GC28-0674-4
File No. S370-37

Program Product

OS/VS2 MVS System Programming Library: Service Aids

MVS/System Product:

JES3 5740-XYN
JES2 5740-XYS

IBM

Fifth Edition (July, 1985)

This is a major revision of, and obsoletes, GC28-0674-3 and Technical Newsletters GN28-0965, GN28-4944, GN28-4920 and GN28-4686 and Supplements GD23-0122-0, GD23-0217-0, and GD23-0178-2.

See the Summary of Amendments following the contents for a summary of the changes 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 1 Release 3.5 of MVS/370 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, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication may be used. Any functionally equivalent program may be used.

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, Programming Publications, Department D58, Building 921-2, PO Box 390, Poughkeepsie, N. Y. 12602. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Preface

The following is a list of requirements for using this publication:

- This publication contains information for the following:

  Interactive Problem Control System (IPCS) - SU57

  OS/VS2 MVS/System Product - JES3 - Program Number 5740-XYN

  OS/VS2 MVS/System Product - JES2 - Program Number 5740-XYS

- Always use the page with the latest date (shown at the top of the page) when adding pages from different Supplements/TNLs.

How to Use the OS/VS2 MVS SPL: Service Aids

This publication explains how, why, and when to use IBM service aids programs for MVS when diagnosing and fixing failures in system or application programs.

Audience

This publication is for system programmers, system operators and IBM program support representatives.

Content

The information in this book describes the following service aids:

- 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.
- AMDPRDMP - Formats and prints dump data sets, which may include GTF trace data, and instruction trace data (created by the console initiated loop recording).
- AMAPTFLE - Updates an operating system by applying PTFs or by generating JCL statements needed to apply PTFs or ICRs in a later step.
- AMDSADMP - Operates as a stand-alone program to produce a dump of instruction trace data (created by the console-initiated loop recording), real and virtual storage.
- AMASPZAP - Verifies and/or replaces instructions and/or data in a load module.

Related service aid components are:

- HMASMP - Provides the user with the facility to apply PTFs or user modifications either selectively or as a group to a VS1 or VS2 system or to the distribution libraries (DLIBs) associated with them. *Note:* SMP is not discussed in this publication. For information on SMP, refer to *OS/VS2 MVS System Modification Program, System Programmer's Guide.*

- IPCS (Interactive Problem Control System) - provides MVS 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; however, an overview of IPCS is provided in the Introduction. For information on IPCS, refer to *OS/VS2 MVS Interactive Problem Control System (IPCS) User's Guide and Reference.*

Note: The system activity measurement facility (MF/1) is not supported with MVS/System Product - JES3 (Program Number 5740-XYN) or MVS/System Product - JES2 (Program Number 5740-XYS).

Organization

Each service aid is explained in a separate chapter and the chapters are arranged in alphabetical order. The chapter page numbers show the names of the programs *minus* the three-character component identifier (such as AMD).

Please 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. This means that you should expect to see AMDPRDMP referred to as simply PRDMP, except in JCL examples and other situations where the full name is necessary. 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.

PRDMP - Print Dump Program.

PTFLE - Program Temporary Fix Link Edit Program.

SADMP - Stand-Alone Dump Program.

SPZAP - Superzap (Data Checker and Modifier).

Related Publications

Three hardware-oriented service aids, IFCDIP00, ISDASDA0, and IFCEREP1, are not documented in this publication, but in publications:

- *OS/VS2 SPL: SYS1.LOGREC Error Recording*, GC28-0677 - describes how IFCDIP00 and ISDASDA0 can be used to initialize and record data from the SYS1.LOGREC data set.

- *OS/VS2 SYS1.LOGREC Error Recording Logic*, SY28-0678 - describes the internal logic of IFCDIP00 and ISDASDA0.

- *OS/VS Environmental Recording Editing and Printing (EREP) Program*, GC28-1178 - describes how EREP can be used to generate reports from records on SYS1.LOGREC.

- *OS/VS Environmental Recording Editing and Printing (EREP) Program Logic*, ZZ28-7032 - describes the internal logic of the IFCEREP1 program and in-storage LOGREC buffer formatter.

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

- *OS/VS2 Service Aids Logic*, SY28-0643 - describes the internal logic of the service aid programs.

- *370/MVS Debugging Handbook Volume 1*, LC28-1385 - describes the dump-type output of the service aids in VS2.

- *OS/VS Message Library: VS2 System Messages*, GC38-1008 - describes the numbered messages issued by the service aids.

- *OS/VS Message Library: EREP Messages*, GC38-1045 - describes the messages produced by the IFCEREP1 programs.

- *OS/VS System Modification Program (SMP) System Programmer's Guide*, GC28-0673 - describes installation and use of SMP.

- *OS/VS System Modification Program (SMP) Logic*, SY28-0685 - describes the internal logic and organization of SMP.

- *MVS Diagnostic Techniques*, SY28-1133 - describes how to print dumps using PRDMP service aids.

- *OS/VS2 MVS Interactive Problem Control System (IPCS) User's Guide and Reference*, GC28-1183.

You should also be familiar with the following publications:

- *OS/VS2 MVS Utilities*, GC26-3902 - describes how to use utility programs to print certain types of service aid output.

- *Operator's Library: OS/VS2 MVS System Commands*, GC28-1031 - describes how to perform certain basic operations in VS2, such as loading a stand-alone program.

- *Operator's Library: OS/VS2 MVS JES2 Commands*, SC23-0048 - describes how to perform certain basic operations in VS2 with JES2.

- *Operator's Library: OS/VS2 MVS JES3 Commands*, GC23-0008 - describes how to perform certain basic operations in VS2 with JES3.

- *MVS JCL Reference*, GC28-1350 - provides background information to understanding the use of parameters, cataloged procedures, spatial allocation of data sets, etc.

- *MVS JCL Users' Guide*, GC28-1349 - describes how to use control statements to override default parameters, use cataloged procedures, allocate space for data sets, etc.

- *OS/VS2 MVS SPL: System Management Facilities (SMF)*, GC28-1030 - describes how a system interfaces with user exit routines to monitor the job stream.

Other publications referenced in the text include the following:

- *IBM S/370 Principles of Operation*, GA22-7000.

- *MVS System Codes*, GC28-1157.

- *OS/VS2 SPL: Initialization and Tuning Guide*, GC28-0681.

- *OS/VS2 SPL: Supervisor*, GC28-1046.

- *OS/VS2 TSO Command Language Reference*, GC28-0646.

- *OS/VS2 SPL: Data Management*, GC26-3830.

- *OS/VS Linkage Editor and Loader*, GC26-3813.

- *OS/VS Linkage Editor Logic*, SY26-3815.

- *VTAM Debugging Guide, ACF/VTAM, Version 2, Diagnosis Guide*, SC27-0615.

- *Data Facility Device Support, DADSM and Common VTOC Access Facility: Diagnosis Guides*, SC26-3968.

Introduction to the OS/VS2 MVS SPL: Service Aids

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

Information Gathering

- To dump real storage, use stand-alone program SADMP. To dump virtual storage, all real storage (including addresses greater than 16 megabytes), and instruction trace data (created by the console-initiated loop recording), use the high-speed version of SADMP. SADMP's output can be formatted and printed using PRDMP.
- To trace system events such as SVC and I/O interruptions, use GTF (the Generalized Trace Facility). Its output can be formatted and printed using the EDIT function of PRDMP.

Formatting and Printing: Mapping

- To summarize and print records in the SYS1.LOGREC data set, use IDASDA0 which is described in the publication *OS/VS2 SPL: 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 and print SADMP output, other system dumps, and GTF trace output, use PRDMP.

Generating and Applying Fixes

- To apply a PTF or an ICR, use PTFLE. SMP may be used to apply PTFs or user modifications. For information on SMP, see the references listed under "Related Publications." PTF tapes distributed after May, 1974 may be in SMP format and, therefore, require modification when using with PTFLE.
- 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 *OS/VS2 SPL: SYS1.LOGREC Error Recording*.

IPCS - Interactive Problem Control System

The OS/VS2 MVS Interactive Problem Control System (IPCS) provides MVS installations with expanded capabilities for diagnosing software failures and facilities for managing problem information and status.

IPCS includes facilities for:

- Online examination of storage dumps.
- Analysis of key MVS system components and control blocks.
- Online management of a directory of software problems that have occurred in the user's system.
- Online management of a directory of problem-related data, such as dumps or the output of service aids.

IPCS runs as a command processor under TSO, allowing the user to make use of existing TSO facilities from IPCS, including the ability to create and execute command procedures (CLISTs) containing the IPCS command and its subcommands.

IPCS supports three forms of MVS storage dumps:

- High-speed stand-alone dumps produced by AMDSADMP.
- Virtual dumps produced by MVS SDUMP on SYS1.DUMP data sets.
- Virtual dumps produced by MVS SDUMP on data sets specified by the SYSMDUMP DD statement.

For information about IPCS, refer to the *OS/VS2 MVS Interactive Problem Control System (IPCS) User's Guide and Reference.*

For complete information about IFCDIP00, and ISDASD0, see *OS/VS2 SYS1.LOGREC (EREP) Program.* For information of SMP, refer to the SMP publications listed under "Related Publication."

Contents

Figures

Summary of Amendments

Summary of Amendments
for GC28-0674-4
as Updated July, 1985

This is a major revision of the *OS/VS2 MVS SPL: Service Aids.* It contains new and updated material in support of MVS/370 System Product Version 1 Release 3.5. It also contains a major reorganization of the topics covered, and minor technical and editorial changes.

Summary of Amendments
for GC28-0674-3
as Updated December 21, 1984.
by Technical Newsletter GN28-0965

This Technical Newsletter contains information about two PRDMP control statements:

- TCAMMAP and
- JES3 (which replaces an old formatting routine).

Also, minor editorial and maintenance changes are included in this document.

Summary of Amendments
for GC28-0674-3
as Updated December 30, 1981
by Technical Newsletter GN28-4944

This technical newsletter contains new and updated information in support of MVS/System Products, and includes the following:

- New trace options in GTF: CCW, CCWP, ASIDP, JOBNAMEP
- New EDIT control options in PRDMP: CCW, CPU
- New 3081 service console in SADMP
- Minor technical and editorial changes

Chapter 1. GTF

Introduction

The generalized trace facility is a service aid program which is available under OS/VS2 to determine and diagnose 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 interrupts and all SVC interrupts,
- specific incidences of one type of system event, such as all I/O interrupts on one particular device,
- user-defined events which are generated by the GTRACE macro.

The GTF output includes system trace records and user trace records. You can direct output to either a data set (IEFRDER) or to buffers in virtual storage. You may use PRDMP's EDIT function to format and print the GTF output you select. See "Chapter 3: PRDMP" and to "Appendix A: Writing EDIT User Programs" for more information about using EDIT to process GTF output.

Note: Affinity has no meaning for GTF. That is, GTF traces events on all processors regardless of the specification for GTF on the AFFINITY macro during system generation.

Using GTF

GTF is an integral part of the OS/VS2 system defined at system generation, and runs as a system task.

Features of GTF

When using GTF, you can select from the options listed below for internal and external tracing. The trace options (shown in parentheses) and associated events are:

- I/O interrupts, including or excluding PCI, from all devices or only particular devices. (IO,IOP)
- All START I/O operations or only those for specific devices. (SIO,SIOP)
- All supervisor call interrupts or only those for specific numbers. (SVC,SVCP)
- All program interrupts or only those for specific interrupt codes. (PI,PIP)
- All external interrupts. (EXT)
- All units of work dispatched by the system, that is SRB, LSR, and TCB. (DSP)
- System recovery routine operations, except those associated with GTF on the first-level interrupt handler, and including STAE/ESTAE operations. (RR)
- All invocations of the system resources manager. (SRM)
- All matching SLIP traps with a tracing action specified or all SLIP traps checked with the SLIP DEBUG option. (SLIP)
- All user/subsystem events defined by GTRACE. (USR)
- All events associated with GTF. (TRC)
- All VTAM remote network activity. (RNIO)
- Channel programs and associated data for I/O events. (CCW,CCWP)

Note: For special considerations in the use of GTF to trace events in indexed VTOC processing, refer to *Data Facility Device Support, DADSM and Common VTOC Access Facility: Diagnosis Guide.*

GTF Trace Output

GTF produces system trace records with two kinds of format, comprehensive and minimal. To see what these records contain for each record type, refer to Figure 1-12, Figure 1-13, and Figure 1-14.

The GTF trace record output can be maintained in virtual storage (internal mode) or directed to the IEFRDER data set on an external storage device (external mode). The external storage device can be either a tape or a direct access device. If there is not enough space, either in storage for internal mode or on the direct access output device for external mode, GTF overlays previously stored or written output beginning at the oldest buffer or physical block.

Retrieving GTF Trace Output

The EDIT function of the PRDMP service aid program provides for formatting and printing internal and external GTF trace records. In addition, the EDIT user format appendage facility allows you to format and print the user trace records generated by the GTRACE macro, and the EDIT user exit facility allows you to inspect all GTF trace records.

If you request that trace data be included in an ABDUMP/SNAP or SVC dump via SDATA=TRT, or in a stand-alone dump, and if GTF is active, EDIT can be used to format the related system records and related user records (created by GTRACE). This occurs independently of the trace mode or options for GTF. The number of buffers formatted is determined by the BUF parameter in the GTF START command. Also, for ABDUMP/SNAP or SVC dumps, only those records directly associated with the failing address space are formatted.

How to Start GTF

You invoke GTF as a system task in storage, 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, time-stamps on records, what action should occur if GTF encounters an error during processing, and the number of buffers that are to appear on ABDUMP/SNAP or SVC dumps. 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. Messages go only to the master console. The START command should be entered only from a console eligible to be a master console. If you should enter the START command from a console not designated as the master console, the command is ignored. An appropriate message is issued.

```
{START}    GTF
{S    }
           [.identifier]
           [,device name | volserial]
           [,(parm)]
           [,MEMBER={GTFPARM|userparm}]

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

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 written by you that invokes GTF.

identifier
specifies the user-determined name identifying this specific GTF session. Note that you use this identifier or the device name in the STOP command to terminate GTF, see Figure 1-5.

device name

specifies an input/output device to contain the trace data set. The device name provided on the IEFRDER DD statement in the cataloged procedure is used unless overridden by the START command. Note that you use this device name or the identifier in the STOP command to terminate GTF; see Figure 1-5.

volserial

indicates the serial number of a magnetic tape or direct access volume that is to contain the trace data set.

(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:

```
(MODE=[{INT|EXT}]) [,subparameter]...)
```

The subparameters are:

```
TIME={YES|NO}
DEBUG={YES|NO}
```

MODE={INT|EXT}

defines where trace data is to be maintained. MODE=INT indicates that trace data is to be maintained in GTF memory. MODE=EXT indicates that trace data is to be maintained in an external trace data set, defined by the IEFRDER DD statement in the cataloged procedure. If you omit this parameter, GTF will assume the default specified in the cataloged procedure, or MODE=EXT if none is specified in the cataloged procedure.

When tracing to an external device, data in the trace data set may be formatted at a later time by the use of the EDIT function of PRDMP.

When you specify SYSM on your TRACE control statement, the data recorded consists of minimal trace records. When you use the SYS option parameter on your TRACE control statement and/or individual keywords, the data recorded consists of comprehensive trace records. If the comprehensive record format is requested, the operator is allowed to select the events to be recorded. The operator is prompted for the trace option parameters. The specific trace options are discussed below in detail.

BUF=nnn

allows you to specify the number of 4096-byte buffers (10 to 255) formatted by EDIT when the records are in a SVCDUMP or stand-alone dump and by the ABEND processor if any task/memory abnormally terminates. If not specified on the START command, the number of buffers is obtained from the EXEC parameter in the cataloged procedure. If you do not specify a value for BUF, a default of 10 4096-byte buffers is used.

TIME = { YES / NO }

TIME = YES requests that every logical 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 *IBM S/370 Principles of Operation.*

When you specify TIME = YES and trace records are formatted and printed by PRDMP, a time stamp record follows each trace record. These time stamp records can be used to calculate the elapsed time between trace entries. The time stamp record is described in *MVS/370 Debugging Handbook, Volume 1.*

The default value is NO when the TIME keyword is not specified either in the START command or on the EXEC statement of the cataloged procedure.

DEBUG = {YES|NO}
requests termination of GTF when any error is encountered, whether or not the error was recoverable.

DEBUG = YES
requests that all error recovery be bypassed, making all errors terminal.

Whenever DEBUG = YES is specified and an error occurs, an error message is issued and GTF immediately terminates.

DEBUG = NO
When DEBUG = NO is in effect, the error message is issued but GTF does not terminate. Instead, GTF attempts to recover from the error and continue.

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.

keyword = option
specifies parameters to override or add specific DD parameters in the IEFRDER DD statement in the cataloged procedure. For example:

- By coding DSNAME = newname, you can specify a different name for the trace data set.
- By coding DSN = NULLFILE when you are specifying MODE = INT, you can prevent the system from sending mount messages to the operator's console.
- By coding DISP = OLD, you can specify an existing data set as the output data set. (Note: If you specify DISP = MOD, GTF changes the data set disposition to OLD.)

- By coding REG=value K, you can specify a REGION parameter. Note that the minimum value is 800 and that K must be included. See Figure 1-6 Storage Requirements for further GTF storage information.

The Cataloged Procedure Supplied by GTF

An IBM-supplied cataloged procedure for the Generalized Trace Facility is supplied in SYS1.PROCLIB with a membername of GTF. The format of the cataloged procedure is shown in Figure 1-2.

```
//GTF        PROC   MEMBER=GTFPARM
//IEFPROC    EXEC   PGM=AHLGTF,REGION=2880K,
//                  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. GTF Cataloged Procedure

The following description explains the statements in the cataloged procedure:

PROC Statement
defines the cataloged procedure named GTF.

EXEC Statement
executes the program AHLGTF, setting th user region size and providing parameters for AHLGTF to use.

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 space to allow the data set to contain twenty 4096-byte physical blocks. When the SYSM option is selected, this allows approximately 1700 trace entries, averaging 46 bytes each. (The 46 bytes include a record prefix containing an 8-byte time stamp value.) For the SYS option, assuming an average record length of 74 bytes, this allows approximately 1100 trace entries. When the primary allocation is filled, recording continues at the beginning of the data set.

If the TRACE data set is directed to tape by the operator, normal end-of-volume processing occurs.

If MODE=INT is specified by the operator, DSN=NULLFILE should be specified here to prevent allocation of a trace data set, because no external recording of trace data occurs.

Note that the data set and attributes on the IEFRDER DD statement may be changed via the START command.

SYSLIB DD Statement (Optional)

defines a member in the SYS1.PARMLIB data set that contains GTF options. If such a member exists, GTF does not prompt you to supply options, but uses the options in the member. If the member does not exist, GTF issues an error message and stops.

A member is supplied in SYS1.PARMLIB, called GTFPARM, that is automatically invoked by the GTF cataloged procedure. GTFPARM causes the following events to be recorded: SVC, IO, PCI, program, and external interrupts; all dispatcher, SIO, system resource manager and recovery routine (RR) events; and all USR entries. A minimal set of data is recorded for each event, except USR events, which are determined by the length specified in the GTRACE macro. All events associated with GTF (TRC) are included in the output. Figure 1-3 shows the format of the GTF-supplied PARMLIB member.

If GTF is started by a user procedure that does not contain a SYSLIB DD statement, message AHL100A is issued requesting that the trace options be supplied from the console.

```
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
```

If you start GTF with a user procedure which 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, 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 1-6.

Summary of GTF Trace Options

The trace option values for GTF are described below:

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

{CCW|CCWP}

requests tracing of channel programs and associated data for I/O events. CCWP requests GTF prompting for the following information:

- tracing CCWs for SIO operations on I/O interrupts or both
- maximum number of CCWs for each event
- maximum number of bytes of data for each CCW
- optional IOSB and EWA tracing
- size of the PCI table

CCW and CCWP only work when the trace options you specify include SIO, SIOP, IO, or IOP.

DSP

requests recording for all dispatchable units of work, (that is, SRB, LSR, TCB and SVC prologue dispatch events). This option is not included by specification of SYS or SYSM; it must be specified additionally. If DSP is specified with SYSM, the trace data is in minimal format; otherwise, it is in comprehensive format.

EXT

requests comprehensive recording of all external interrupts.

{IO|IOP}

requests comprehensive recording of all non-PCI I/O interrupts. Unless the trace option PCI is also specified, PCI interrupts are not recorded. IOP requests GTF prompting to specify devices for which I/O interrupts are to be recorded.

JOBNAMEP

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

PCI

requests that all I/O interrupts be recorded in the same format as other I/O trace records being created. If, as a result of prompting for I/O events, specific devices were selected, the PCI interrupts recorded are only for those devices selected during prompting. PCI may only be requested in conjunction with IO.

{PI|PIP}

requests comprehensive recording for all program interrupts (1-19). PIP requests further GTF prompting for those interrupt codes for which data should be recorded.

RNIO
requests comprehensive recording of all VTAM network activity. This option is not included in specification of SYS. To request minimal trace data recording for this event, specify SYSM and RNIO.

RR
requests that all invocations of recovery routines and STAE/ESTAE routines be recorded. A trace record describing the activity of the recovery routine (RR) is created when control is passed from the recovery routine back to the recovery termination manager (RTM). Data is in minimal format when requested via SYSM; otherwise, it is in comprehensive format.

{SIO|SIOP}
requests comprehensive-format recording for system start I/O operations. SIOP requests prompting for the specific devices for which SIO events should be recorded. When prompting is requested, only the specified devices cause record entries in GTF trace buffers.

SLIP
requests that a trace entry be made each time that a match occurs for a SLIP trap that includes a tracing action 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 are specified on the SLIP command. The SLIP option is not included in the specification of SYS or SYSM; it must be specified separately. Specification of the SYS or SYSM option does not affect the data collected on the SLIP trace record.

SRM
requests that a trace entry be made each time the system resource manager is invoked. This option is not included in specification of SYS or SYSM; it must be specified separately. If SRM is specified with SYSM, records are in minimal format; otherwise, they are in comprehensive format. Further information about this option is in the section "System Performance Factors" of the *OS/VS2 SPL: Initialization and Tuning Guide.*

{SVC|SVCP}
requests comprehensive recording for all SVC interrupts. SVCP requests prompting for those SVC numbers for which data should be recorded.

SYS
requests that comprehensive trace data be recorded for six system events: I/O, SIO, SVC, program and external interrupts, and recovery routines (RR). Any additional event keywords specified, that is, SRM, DSP, and RNIO, result in comprehensive trace entries for those events.

SYSM
requests that only minimal trace data be recorded for the six system events: I/O, SIO, SVC, external and program interrupts, and recovery routines. Any additional trace event keywords must be specified with SYSM (SRM, DSP, RNIO) in order to get minimal format trace entries for those events. For example, coding TRACE=SYSM, DSP, SRM, USR, RNIO results in minimal trace data being recorded for all events associated with SYSM as well as DSP, RNIO, and SRM events. USR entries have the length specified by you in the GTRACE macro.

Note: Coding SYS or SYSM causes the following trace options to be ignored if specified in any form: SIO, I/O, SVC, PI, EXT, and RR.

SYSP
requests the same options as the SYS option, and further GTF prompting for specific system events being recorded during execution. A prompt is issued for selection of specific SVC, I/O, SIO, and PI events being recorded. Data recorded is comprehensive in format. Also, any additional trace event keywords specified (SRM, RNIO, or DSP) result in comprehensive trace entries for those events.

TRC
requests recording of those events being traced that are associated with GTF. Unless this is requested, the GTF-associated events are filtered out and not recorded.

USR
requests that all data passed to GTF via the GTRACE macro be recorded with the system data in the trace data set, or in storage if MODE=INT. 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; the EID's and symbolic names associated with these subsystems are listed in Figure A-5 of Appendix A.

Combining Certain GTF Options

Figure 1-4 shows those TRACE options that GTF does *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	SIOP	SIO
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	CCWP	CCW			

Figure 1-4. Combining Certain GTF Options

Prompting

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

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

The event keywords correspond to those trace option values that request prompting (IOP, SIOP, SVCP, PIP, SYSP, ASIDP, JOBNAMEP, CCWP) and that are listed in the message. Only these keywords are accepted in the reply. If SYSP is specified, all keywords are valid and only those for which specific event recording is desired need be specified. 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, the operator is prompted to continue specification. Event keywords are as follows:

IO = (devaddr1[,devaddr2][...,devaddr50])
specifies up to 50 device addresses for which you want I/O interruptions traced. All other IO interruptions are filtered out. If you have specified IOP or SYSP, and do not specify IO= in response to the prompting messages, no I/O interruption filtering takes place.

IO = SIO = (devaddr1[,devaddr2][...,devaddr50])
only valid after requesting SYSP or both IOP and SIOP, specifies up to 50 device addresses for which you want GTF to trace both I/O and SIO events. All other I/O and SIO events, except those requested specifically by IO= or SIO=, are filtered out.

SIO = (devaddr1[,devaddr2][...,devaddr50])
specifies up to 50 device addresses for which you want SIO operations traced. All other SIO operations are filtered out. If you have specified SIOP or SYSP, and do not specify SIO= in response to the prompting message, no SIO filtering takes place.

SVC = (svcnum1[svcnum2][...,svcnum50])
specifies up to 50 SVC numbers that you want traced. All other SVC numbers are filtered out. If you have specified SVCP or SYSP, and do not specify SVC= in response to the prompting message, no SVC filtering takes place.

PI = (code1[,code2][...,code15,code17,code18])
specifies one to 18 program interrupt codes that you want traced. All other program interrupts are filtered out. If you have specified PIP or SYSP, and do not specify PI= in response to this prompting message, no program interruption filtering takes place.

ASID = (asid1[,asid2][...,asidn])
specifies one to five address space identifiers that you want GTF to trace. The values 'asid1' through 'asidn' 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.

If you use more than one line to specify ASIDs, GTF stacks your replies until it reaches 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 when the jobs that JOBNAMEP identified are running in address spaces that ASIDP did not identify.

JOBNAME=(job1[,job2][...,jobn])
specifies one to five jobnames for which you want GTF tracing to occur. The values 'job1' through 'jobn' must be valid MVS 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.

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 when the address spaces that ASIDP identified contain jobs that JOBNAMEP did not identify.

CCW=(S|I|SI[,CCWN=nnn][,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 defaults are in effect. 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 SIO and resume I/O operations. CCW=S works only if you specify SIO or SIOP 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 works only if you specify IO or IOP as trace options.

SI

specifies GTF tracing of channel programs for both SIO operations and I/O interruptions. CCW=SI works only if you specify either SIO or SIOP and either IO or IOP as trace options.

CCWN=nnn

specifies the maximum number of CCWs that you want traced for each event. The value 'nnn' is an integer from 1 to 512. 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 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 SIO or resume I/O operations, GTF does not trace data for read, read backwards, or sense commands in the channel programs. If the SKIPBIT is on, 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 if it GTF encounters an exceptional condition when tracing a channel program.

PCITAB=n

specifies the 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 interrupt 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 interrupt 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.

PCITAB is meaningful only when the trace options you specify include PCI. When you specify PCITAB and do not specify PCI, GTF does not allocate storage for the PCI table, but PCITAB must have a valid value.

Notes:

- GTF imposes a limit on the number of specific values you can supply through prompting. If you exceed this limit, GTF issues a message and you must respecify all values.

- Up to 50 device addresses may be specified for IO= or SIO=; up to 50 device addresses may be specified by IO=SIO=; however the sum of device addresses specified by IO= and IO=SIO= may not exceed 50; likewise the sum of device addresses specified by SIO= and IO=SIO= may not exceed 50.

- Within a given reply, each keyword specified 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=(asid1, asid2)
Reply #2 ASID=(asid3, asid4, asid5)
```

  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 USR," and "Example 9: Specifying Which System Events GTF Traces, Using Trace Options SIOP, IOP, PCI, CCWP, SVC, and JOBNAMEP."

- Prompting increases GTF storage requirements. Refer to Figure 1-6 GTF Storage Requirements for further information.

GTF Examples

In the examples of message sequences given below, the highlighted areas contain the messages entered, and the shaded portions contain the messages received at the console while using GTF.

Example 1: Starting GTF Using the GTF Catalogued Procedure

When GTF is started using the GTF cataloged procedure supplied by IBM, a set of trace options, specified in the SYS1.PARMLIB member GTFPARM, is used. GTFPARM contains a record that defines these trace options to be placed in effect: TRACE=SYSM, DSP, PCI, SRM, TRC, USR.

This example illustrates the use of the GTF cataloged procedure.

```
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'
AHL1031I GTF INITIALIZATION COMPLETE
```

Example 2: Starting GTF for Internal Tracing Using the GTF Cataloged Procedure

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,SIO,SVC,DSP'
AHL103I TRACE OPTIONS SELECTED -- DSP,SVC,SIO,IO
00 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U
R 00,'U'
AHL031I GTF INITIALIZATION COMPLETE
```

Example 3: Directing GTF Trace 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 on a direct access device. The device name 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 resides on a 2400 tape drive and has a volume serial of TRCTAP. Note that the GTFPARM member of SYS1.PARMLIB is used to specify the trace options.

```
START GTF,2400,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

You can save time when starting GTF by previously storing one or more set combinations of trace options as members in SYS1.PARMLIB, and including a SYSLIB DD statement in the cataloged procedure. If you do this, GTF does not prompt you to supply trace options, but gets them from SYS1.PARMLIB instead.

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=(191,192),SIO=282,PI=15
./       ADD     NAME=GTFB,LIST=ALL,SOURCE=0
TRACE=IO,SIO,TRC
./       ADD     NAME=GTFC,LIST=ALL,SOURCE=0
TRACE=SYS,PCI
/*
```

For a description of each statement, see *OS/VS2 MVS Utilities*, and *OS/VS2 MVS JCL*. For further information about SYS1.PARMLIB see "System Initialization" section of the *OS/VS2 SPL: Initialization and Tuning Guide*.

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 the following:

- SVC and IO interrupts
- All SIO operations
- All matching SLIP traps with a tracing action specified or SLIP traps in DEBUG mode
- All dispatcher events

Also, all issuers of the GTRACE macro are to have their user data recorded in the trace buffers. The trace data is written to the data set ABCTRC. (Note that when the end of the primary extent is reached, writing continues at the beginning).

```
START USRPROC,250,231405,(MODE=EXT),DSN=ABCTRC
00 AHL100A SPECIFY TRACE OPTIONS
R 00,'TRACE=SVC,SIO,IO,DSP,SLIP,USR'
AHL103I TRACE OPTIONS SELECTED--USR,DSP,SVC,SIO,IO,SLIP
01 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U
R 01,'U'
AHL031I GTF INITIALIZATION COMPLETE
```

Example 6: Prompting Keywords Stored in SYS1.PARMLIB

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 (as message AHL103I indicates).

```
AHL103I TRACE OPTIONS SELECTED--SYSP,USR
AHL103I IO=(191,192),SIO=(282),SVC=(1,2,3,4,10)
```

This message may contain more than one line, depending on the number of options selected by the operator. If the set of devices specified for IO and SIO are identical, message AHL103I shows them as if specified by use of IO=SIO.

After being informed of the GTF trace options in effect, you are given an opportunity to change these options:

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

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

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

Record #2 IO=(191,192,193),SVC=(1,2,3)

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

When all prompting is complete, whether from you or from PARMLIB, you are notified which GTF options are in effect as in example two (via message AHL103I). You then are given an opportunity to respecify the trace option, or accept those previously specified (either by you or PARMLIB) via message AHL125A.

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

In this example, GTF is started in external mode to the data set defined in the cataloged procedure. The trace options selected request all system event types be traced as well as user entries generated by use of the GTRACE macro. The 'P' on SYS requests the ability to select only certain events to be traced within some event type. Message AHL101A informs the operator that he may exercise selectivity on SVC, IO, SIO, and PI event types. Those keywords not included in the reply to AHL101A are all recorded. In this example, the operator selects five SVCs, two devices for IO interrupts, and one device for SIO operations. All other SVC, IO, and SIO events are not recorded. All PI, EXT, and RR events are recorded.

```
START MYPROC.EXAMPLE7,,,(MODE=EXT)
00 AHL100A SPECIFY TRACE OPTIONS
R 00,'TRACE=SYSP,USR'
01 AHL101A SPECIFY TRACE EVENT KEYWORDS--SVC=,IO=,SIO=,PI=
R 01,'SVC=(1,2,3,4,10),IO=(191,192)'
02 AHL102A CONTINUE TRACE DEFINITION OR REPLY END
R 02,'SIO=282,END'
AHL103I TRACE OPTIONS SELECTED--SYSP,USR
AHL103I IO=(191,192),SIO=(282),SVC=(1,2,3,4,10)
03 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U
R 03,'U'
```

Example 8: Starting GTF to Trace VTAM Remote Network Activity

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. They are:

- 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--RNIO,IO,USR
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 SIOP, 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, SIO, CCW, and JOBNAME keywords. In reply 01 the operator selected one device for tracing both IO and SIO 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 SIO 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=SIOP,IOP,PCI,CCWP,SVC,JOBNAMEP

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

R 01,JOBNAME=(BACKWARD),IO=SIO=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

AHL103I JOBNAME=(BACKWARD),IO=SIO=(580)

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

03 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U

R 03,U
```

How to STOP GTF

To stop GTF processing, you need to know either the identifier or device address used in the START command. See "Format of the START Command" in this chapter. If you are not sure of the identifier or device address, use the operator display command:

```
D A,LIST
```

This command causes the system to display the number of:

- Active batch jobs.
- Active time sharing users.
- Mount commands in execution.
- Started tasks.

The LIST parameter causes the system to include jobnames and V=R region boundaries in the A display. This parameter can be abbreviated by entering L.

How to Specify the STOP Command

Figure 1-5 shows the general format of the STOP command when used to stop GTF. As with the START command, you must enter this command from a console eligible to be a master console.

```
STOP      identifier
P         device address
```

Figure 1-5. General Format of the STOP Command

The identifier or device address field indicates the identifier or device address specified in a START command, when GTF was started. If GTF was started in internal mode and no identifier was specified, the identifier is 'GTF'.

You may enter the STOP command at any time during GTF processing.

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.

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

The following command stops the GTF session started in the previous example:

```
P EXAMPLE
```

Example 2: Using the Device Name

This example starts a GTF session with trace data recorded on a non-labeled tape mounted on device 282. Each trace record is to be time-stamped. Twenty buffers are to be formatted if a dump is taken.

```
S GTF,282,,(TIME=YES,BUF=20),LABEL=(,NL)
```

The following command stops the GTF session started in the previous example:

```
P 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 is to run in an address space whose maximum size is 1000K.

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

GTF Storage Requirements

Link Pack Area

Fix = Opt + Prmpt + 8K

Fix: Fixed storage in pageable LPA 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	2.5K
SVC, SVCP	8K
SRM, FRR, RNIO	3K
SLIP	8K
USR	1.5K
PCI, TCB	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, SIOP, 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.

16500: 16500 bytes for buffer area, when GTF, is active.

REG: 140 bytes per processor are required for register save areas, regardless of whether or not GTF is active. (If MVS/System Extensions is installed, 204 bytes per processor are required.)

SAVE: 800 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.

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 1-6. 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.

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

EID Assignment for User Events

Events traced by the GTRACE macro 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 by the EDIT function of PRDMP. Usually, user data is printed in hexadecimal, because EDIT cannot format records that are not created by GTF. However, you can write format appendages to format specific types of user data records. For information about writing EDIT format appendages, see Appendix A: Writing EDIT User Programs. (Note: If your installation has format appendages written for use with OS/MFT or OS/MVT, you can still use them in OS/VS. EDIT recognizes and accepts format appendages named IMDUSRxx as well as those named HMDUSRxx and AMDUSRxx.)

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

General Format of the GTRACE Macro

Figure 1-7 shows the general format of the GTRACE macro, standard form.

```
[symbol] GTRACE DATA=address,LNG=number,ID=value[,FID=value][,PAGEIN= NO  |YES]
```

Figure 1-7. General Format of the GTRACE Macro, Standard Form

The individual parameter formats are shown below:

DATA = address	address:	A-type address or register (2)-(12).
LNG = number	number:	symbol, decimal digit, hexadecimal value, or register (2)-(12).
ID = value	value:	symbol, decimal digit, or hexadecimal value.
FID = value	value:	symbol, decimal digit, hexadecimal value, or register (2)-(12).
	Default:	FID = 0
PAGEIN = YES PAGEIN = NO	Default:	PAGEIN = NO

The parameters in the macro are described below.

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

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

ID = value
0 to 1023--user events
1024 to 1535--reserved for program products
1536 to 4095--reserved for IBM components and subsystems

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

FID = value
indicates the format appendage for this record when the trace output is processed by the EDIT function of PRDMP. The format appendage name is formed by appending the 2-digit FID value to the names AMDUSR, HMDUSR, and IMDUSR. FID values are assigned as follows:

```
0 (or FID=parameter omitted)--record to be dumped in hexadecimal
1 to 80--user format identifiers
81 to 255--reserved
```

The default value is X'00'. The value may be specified in decimal or hexadecimal (as X'value').

PAGEIN = {NO|YES}
indicates that paged-out user data is to be processed (YES) or is not to be processed (NO). To make sure that all user data is always traced, specify YES.

Figure 1-8 shows how the GTRACE macro can be coded to record 200 bytes of data, beginning at the address of AREA, with an event identifier of 37, and to be formatted by the format appendage named IMDUSR40.

```
GTRACE DATA=AREA,LNG=200,ID=37,FID=64
```

Figure 1-8. An Example of the GTRACE Macro

List Form of the GTRACE Macro

Figure 1-9 shows the general format of the GTRACE macro, list form.

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

Figure 1-9. General Format of the GTRACE Macro, List Form

The individual parameter formats are shown below:

DATA = address	address:	A-type address.
	Default:	DATA = 0
LNG = number	number:	symbol, decimal digit, or hexadecimal value.
	Default:	LNG = 0
FID = value	value:	symbol, decimal digit, or hexadecimal value.
	Default:	FID = 0.

MF = L

The parameters are described under the standard form of the GTRACE macro, with the following exception:

MF = L
indicates that this is the list form of the GTRACE macro.

Execute Form of the GTRACE Macro

Figure 1-10 shows the general format of the GTRACE macro, execute form.

```
[symbol] GTRACE MF=(E,prob addr),ID=value[,DATA=address]
                [,LNG=number][,FID=value][,PAGEIN=YES|NO]
```

Figure 1-10. General Format of the GTRACE Macro, Execute Form

The individual parameter formats are shown below:

MF=(E,prob addr)	prob addr:	A-type address, or register (1) or (2)-(12).
ID=value	value:	symbol, decimal digit, or hexadecimal value.
DATA=address	address:	A-type address, or register (2)-(12).
LNG=number	number:	symbol, decimal digit, hexadecimal value, or register (2)-(12).
FID=value	value:	symbol, decimal digit, hexadecimal value, or register (2)-(12).
	Default:	FID=0

Note: If the FID value is not specified on the execute form of GTRACE, the FID value defaults to zero. This default overlays any FID value that may have been specified on a list form of GTRACE.

PAGEIN=YES PAGEIN=NO	Default:	PAGEIN=NO

The parameters are described under the standard form of the GTRACE macro, with the following exception:

MF=(E,prob addr)
indicates that this is the execute form of the GTRACE macro using a remote program parameter list.

GTF Error Recovery Handling

GTF recognizes as potentially recoverable all errors that occur while it is 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 does issue 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. It does not issue a message to the operator's console nor disable the function that caused the error; instead, it 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 never causes 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, see *OS/VS2 Service Aids Logic*, and the *MVS/370 Debugging Handbook Volume 1*.

Trace Records

GTF creates trace records for each system event you select. The records have the general format shown in Figure 1-11.

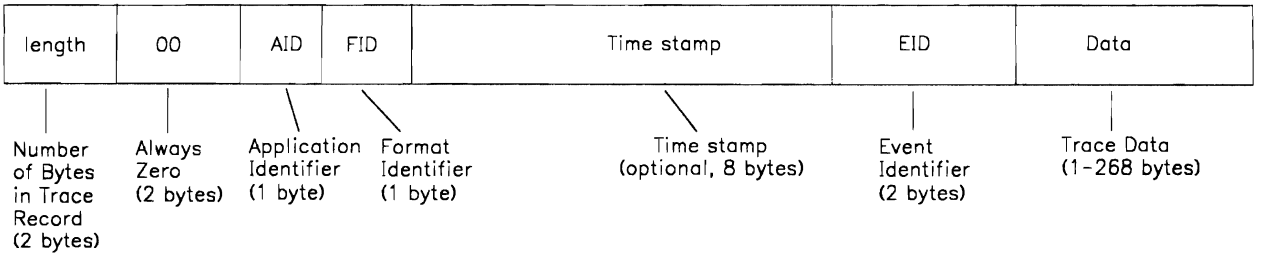


Figure 1-11. Fields in a Trace Record

length
indicates the total length of the record in bytes.

00
always zero.

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

Possible values are:

```
'FF' -- Trace record for data
'00' -- GTF control record
'01' to 'FC' -- reserved
```

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

time stamp

is the eight-byte TOD clock value (local time) given when the record construction is completed. To have this value recorded, specify TIME=YES on the START command. (TOD clock values are described in *IBM S/370 Principles of Operation.*)

PRDMP formats and prints the TOD clock value in a time stamp record that follows each trace record. (The time stamp record is described in *MVS/370 Debugging Handbook Volume 1.*)

EID

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

data

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

Figure 1-12, Figure 1-13, and Figure 1-14 are examples of trace output as processed by the EDIT function of PRDMP. In all the examples, fields identified as h-----h are hexadecimal representations, and fields identified as c-----c are alphameric characters. N/A indicates that the field label does not apply to this particular record.

```
VS2R3 RNIO TEST FOR COMPREHENSIVE RECORDS                    EXTERNAL TRACE - DD TAPE                    PAGE 020

          R15  00F2C850  R0  00000001  R1 FFFD74B8
          PLIST 00028A10 00028A14 00028A18 00028A1C 00028C20 00028C38 00028C80 80028CC8
SIO  0354 ASCB 000106F0  CPU 0000  JOBN *AUXSTM*  R/V CPA 001EC068 00FEC068  CAW 00003BA8  DSID 00000000
          FLGS 00000010 8803       STAT 0600      SK ADDR 00000000 06000F03  CC 4
IO   0354 ASCB 000106F0  CPU 0000  JOBN *AUXSTM*  OLD PSW 070C2000 0002208C  TCB 000108D8  DSID 00000000
          CSW  001EC090 0C000001   SNS  N/A       R/V CPA 001EC068 00FEC068  FLG C0108803  2C880000 00
EXT  1004 ASCB 000106F0  CPU 0000  JOBN *AUXSTM*  OLD PSW 070C1004 0002208C  TCB N/A
                     TQE FIELDS:  ASCB 00000308  FLG/EXI 0384     00FF1018  TCB 00010DF8
SVC   060 ASCB 00FE8720  CPU 0000  JOBN JOBNAME   OLD PSW 070C003C 00E4AB94  TCB 00C9CB28  MODN SVC-RES
          R15  00011610  R0  0000084   R1 00FEC7C0
          PLIST N/A
SVC   010 ASCB 00FE8720  CPU 0000  JOBN JOBNAME   OLD PSW 070C000A 00E4ABD6  TCB 00C9CB28  MODN SVC-RES
          R15  00000000  R0  E50001E0  R1 00C71E20
EXT  1004 ASCB 00FE8720  CPU 0000  JOBN JOBNAME   OLD PSW 070C1004 000678FC  TCB 00C9CB28
                     TQE FIELDS:  ASCB 00000308  FLG/EXI 0384     00FF1018  TCB 00010DF8
RNIO ASCB 00FE8720  CPU 0000 JOBN N/A          IN F1F04002 E8E3C5E2 0000
SVC   035 ASCB 00FE8720  CPU 0000  JOBN N/A       OLD PSW 070C0023 000678FC  TCB 00C9CB28  MODN INTERSIM
          R15  0000002A  R0  00000000  R1 00067E74
          PLIST 00238000 0F0C79AC 000C7A28 10004020
SVC   010 ASCB 00FE8720  CPU 0000  JOBN N/A       OLD PSW 070C000A 00E4A036  TCB 00C9CB28  MODN SVC-RES
          R15  0000002A  R0  E50001E0  R1 80E4A034
EXT  1004 ASCB 00FE8720  CPU 0000  JOBN N/A       OLD PSW 070C1004 0003A5AE  TCB 00C9CB28
                     TQE FIELDS:  ASCB 00FFE318  FLG/EXI 0388     00010DF8  TCB 7004E3A4
PGM   017 ASCB 00FE8720  CPU 0000  JOBN N/A       OLD PSW 070C0011 00E4A038  TCB 00C9CB28  MODN SVC-RES   VPA 00C71E24
          R0 000001E0  R1 00C71E20  R2 FF400001  R3 00011098  R4 00C9CB28  R5 00C9DDD8  R6 00E4A000  R7 00FE8720
          R8 00000000  R9 00C71E20 R10 00FFF710 R11 40E4A028 R12 00E4B027 R13 00067CAC R14 0000F364 R15 00000000
EXT  1004 ASCB 00FE8720  CPU 0000  JOBN N/A       OLD PSW 070C1004 00E4A038  TCB 00C9CB28
                     TQE FIELDS:  ASCB 00FD9B10  FLG/EXI 07C0     13065A58  TCB 00C9CCD0
EXT  1004 ASCB 00FD9B10  CPU 0000  JOBN JES2      OLD PSW 070C1004 00017A68  TCB N/A
                     TQE FIELDS:  ASCB 00000308  FLG/EXI 0384     00FF11C8  TCB 00010DF8


*** DATE   DAY 212    YEAR 1974           TIME 16.49.02.365709                                              **

IO   001F ASCB 0000FB48  CPU 0000  JOBN *MASTER*  OLD PSW 070C1000 00017A68  TCB 00010220  DSID 00C99BCC
          CSW  001F5590 0C000001   SNS  N/A       R/V CPA 001F5580 00FF5580  FLG 40003302  00000000 00
PGM   017 ASCB 00FED720  CPU 0000  JOBN PROC003   OLD PSW 070C0011 00F784D2  TCB N/A       MODN N/A       VPA 0009B0FF
          R0 00FE78A0  R1 00FE7BE8  R2 00F787A6  R3 00FE78CC  R4 0009B000  R5 00000FEA  R6 00FF2FFC  R7 00000FEA
          R8 00FF2FF0  R9 00021022 R10 00000000 R11 00035B40 R12 00F783D0 R13 000703F0 R14 00021022 R15 00F783D0
SVC   072 ASCB 0000FB48  CPU 0000  JOBN *MASTER*  OLD PSW 070C0048 00F2C9AA  TCB 00010220  MODN N/A
          R15  00F2C850  R0  00000001  R1 000289D8
SVC   010 ASCB 0000FB48  CPU 0000  JOBN *MASTER*  OLD PSW 070C000A 00E275BE  TCB 00010220  MODN SVC-RES
          R15  00E27660  R0  E7000060  R1 00D1F768
EXT  1004 ASCB 0000FB48  CPU 0000  JOBN *MASTER*  OLD PSW 070C1004 000125B8  TCB 00010220
                     TQE FIELDS:  ASCB 00000308  FLG/EXI 0384     00FF11C8  TCB 00010DF8
PGM   017 ASCB 0000FB48  CPU 0000  JOBN *MASTER*  OLD PSW 070C0011 00F2CD1A  TCB 00010220  MODN           VPA D561C140
          R0 FA000082  R1 00062F78  R2 00028A10  R3 00000000  R4 00D1DCCC  R5 00028A0C  R6 00028CA4  R7 00028968
          R8 90F2C962  R9 80F2C9AE R10 50F2C856 R11 00028C80 R12 00D1BC8C R13 00028CC8 R14 B0F2CA9A R15 00000000
SVC   036 ASCB 0000FB48  CPU 0000  JOBN *MASTER*  OLD PSW 070C0024 00F2CFD0  TCB 00010220  MODN N/A
          R15  00000000  R0  FA000082  R1 00062F78
PGM   017 ASCB 0000FB48  CPU 0000  JOBN *MASTER*  OLD PSW 070C0011 00DB68D8  TCB 00010220  MODN SVC-RES   VPA 00DB68D8
          R0 FA000082  R1 00062F78  R2 FF400001  R3 00011098  R4 00010220  R5 00FFC840  R6 00DB68D8  R7 0000FB48
          R8 00000000  R9 400122F4 R10 00FFF710 R11 00028C80 R12 00D1BC8C R13 00028CC8 R14 0000F364 R15 00000000
SVC   010 ASCB 0000FB48  CPU 0000  JOBN *MASTER*  OLD PSW 070C000A 00DB68FE  TCB 00010220  MODN SVC-RES
          R15  40DB68DA  R0  000000CE  R1 80DB68FC
PGM   017 ASCB 0000FB48  CPU 0000  JOBN *MASTER*  OLD PSW 070C0011 00DB6902  TCB 00010220  MODN SVC-RES   VPA 00063F34
```

Figure 1-12. Format of Comprehensive Trace Records for SIO, IO, EXT, SVC, RNIO, and PGM

```
                                                    EXTERNAL TRACE - DD TAPE                          PAGE 0004
        PLIST 00973F80
SVC   048 ASCB 00FEBB28  CPU 0000  JOBN BACKWARD  OLD PSW 071C0030 00DF8D4E  TCB 0095A0E8  MODN SVC-RES
        R15  0095FB40  R0  00000408  R1 0095FF30  MAJOR SYSIEFSD  MINOR RPLL
        PLIST 00000000 C0040000 00DF950D 00DF9515 C0040000
SVC   010 ASCB 00FEBB28  CPU 0000  JOBN BACKWARD  OLD PSW 071C000A 00DF8D64  TCB 0095A0E8  MODN SVC-RES
        R15  00000000  R0  E6000408  R1 0095FB40
SVC   048 ASCB 00FEBB28  CPU 0000  JOBN BACKWARD  OLD PSW 075C0030 00D8C718  TCB 0095A0E8  MODN SVC-RES
        R15  00480000  R0  00480001  R1 0096BE80  MAJOR SYSZTIOT  MINOR    o■\
        PLIST 0096BBD8 C0064100 00D8CB98 0096BE8C 00080096
SVC   000 ASCB 00FEBB28  CPU 0000  JOBN BACKWARD  OLD PSW 078D0000 00ABBCD6  TCB 0095A0E8  MODN READBACK  DDN TAPE
        R15  00ABBA50  R0  000E4F80  R1 000E4F80  DCB 000E5FA4  DEB 0096BAD4
SVC   001 ASCB 00FEBB28  CPU 0000  JOBN BACKWARD  OLD PSW 078D0001 00F9B8FA  TCB 0095A0E8  MODN READBACK
        R15  00F9B878  R0  00000001  R1 000E5EB8
        PLIST 000E5EB8
SIO   0580 ASCB 00FEBB28  CPU 0000  JOBN BACKWARD  R/V CPA 003F3E18 00FF2E18  CAW 800174E0  DSID 0096BAD4
        FLGS 00000000 C302       STAT 0400       SK ADDR 00000000 00000000  CC 0   CS 00
        CCW CHAIN  SIO    DEV 0580    ASCB 00FEBB28    CPU 0000    JOBN BACKWARD
        000174E0  C3064BA0 70000001                                                    *                          *
        000174E8  083F3E18 00000000                                                    *                          *
        00FF2E18  013F3B58 24002328                                                    *                          *
            IDAW  00396CD8 0328     00010002 00030004 0005---- ----0190 01910192 01930194  *.........----...J.K.L.M*
            IDAW  0014F000 0800     01950196 01970198 0199---- ----0590 05910592 05930594  *.N.O.P.Q.R----...J.K.L.M*
            IDAW  0014F800 0800     05950596 05970598 0599---- ----0990 09910992 09930994  *.N.O.P.Q.R----...J.K.L.M*
            IDAW  000E7000 0800     09950996 09970998 0999---- ----0D90 0D910D92 0D930D94  *.N.O.P.Q.R----...J.K.L.M*


*** DATE   DAY 225    YEAR 1981           TIME 19.43.57.673859                                                  ***

            IDAW  000E7800 0800     0D950D96 0D970D98 0D99---- ----1190 11911192 11931194  *.N.O.P.Q.R----...J.K.L.M*
IO    0580 ASCB 00FEBB28  CPU 0000  JOBN BACKWARD  OLD PSW 070C2000 00C38E68  TCB 0095A0E8  DSID 0096BAD4
        CSW  803F3E20 0C000000   SNS  N/A        R/V CPA 003F3E18 00FF2E18  FLG 0000C302  A2000580 00 CS 00
        CCW CHAIN  IO     DEV 0580    ASCB 00FEBB28    CPU 0000    JOBN BACKWARD
        00FF2E18  013F3B58 24002328                                                    *                          *
            IDAW  00396CD8 0328     00010002 00030004 0005---- ----0190 01910192 01930194  *.........----...J.K.L.M*
            IDAW  0014F000 0800     01950196 01970198 0199---- ----0590 05910592 05930594  *.N.O.P.Q.R----...J.K.L.M*
            IDAW  0014F800 0800     05950596 05970598 0599---- ----0990 09910992 09930994  *.N.O.P.Q.R----...J.K.L.M*
            IDAW  000E7000 0800     09950996 09970998 0999---- ----0D90 0D910D92 0D930D94  *.N.O.P.Q.R----...J.K.L.M*
            IDAW  000E7800 0800     0D950D96 0D970D98 0D99---- ----1190 11911192 11931194  *.N.O.P.Q.R----...J.K.L.M*
SVC   000 ASCB 00FEBB28  CPU 0000  JOBN BACKWARD  OLD PSW 078D0000 00ABBCD6  TCB 0095A0E8  MODN READBACK  DDN TAPE
        R15  00ABBA50  R0  000E4F80  R1 000E4F80  DCB 000E5FA4  DEB 0096BAD4
SIO   0580 ASCB 00FEBB28  CPU 0000  JOBN BACKWARD  R/V CPA 003F3590 00FF2590  CAW 800174E0  DSID 0096BAD4
        FLGS 00000000 C302       STAT 0C00       SK ADDR 00000000 00000000  CC 0   CS 00
        CCW CHAIN  SIO    DEV 0580    ASCB 00FEBB28    CPU 0000    JOBN BACKWARD
        000174E0  C3064BA0 70000001                                                    *                          *
        000174E8  083F3590 00000000                                                    *                          *
        00FF2590  0C3F3DF8 04002328                                                    *                          *
SVC   001 ASCB 00FEBB28  CPU 0000  JOBN BACKWARD  OLD PSW 078D0001 00F9B8FA  TCB 0095A0E8  MODN READBACK
        R15  00F9B878  R0  00000001  R1 000E5EEC
        PLIST 000E5EEC
IO    0580 ASCB 00FEBB28  CPU 0000  JOBN BACKWARD  OLD PSW 070E0000 00000000  TCB 0095A0E8  DSID 0096BAD4
        CSW  803F3598 0C000000   SNS  N/A        R/V CPA 003F3590 00FF2590  FLG 0000C302  A2000580 00 CS 00
        CCW CHAIN  IO     DEV 0580    ASCB 00FEBB28    CPU 0000    JOBN BACKWARD
        00FF2590  0C3F3DF8 04002328                                                    *                          *
            IDAW  00396CD8 04D9     280F290F 2A0F2B0F 2C0F---- ----1190 11911192 11931194  *.........----  ..J.K.L.M*
            IDAW  003967FF 0800     280B290B 2A0B2B0B 2C0B---- ----230F 240F250F 260F270F  *.........----.........*
            IDAW  001C2FFF 0800     28072907 2A072B07 2C07---- ----230B 240B250B 260B270B  *.........----.........*
            IDAW  001C27FF 0800     28032903 2A032B03 2C03---- ----2307 24072507 26072707  *.........----.........*
            IDAW  002D1FFF 064F     00010002 00030004 0005---- ----2303 24032503 26032703  *.........----.........*
SVC   013 ASCB 00FEBB28  CPU 0000  JOBN BACKWARD  OLD PSW 078D000D 000A0A3E  TCB 0095A0E8  MODN READBACK
        R15  4002BF5E  R0  00000001  R1 000E5EEC  COMP CODE  00000000
```

Figure 1-13. Format of Comprehensive Trace Records for SVC, SIOP, IOP, PCI, CCWP, and JOBNAMEP

```
VS2R3 TEST FOR MINIMAL RECORDS                                  EXTERNAL TRACE - DD TAPE                                  PAGE 0454

SIO  ASCB 000106F0  CPU 0000  CPA 001ECF28 00FECF28  CAW 00003BA8  DEV ADD 0354  STATUS 0C00  CC 0
IO   ASCB 000106F0  CPU 0000  PSW 070C2000 0002208C  TCB 000108D8  DEV ADD 0354  CSW 001ECF50 0C000001  SNS N/A
SRB  ASCB 000106F0  CPU 0000  PSW 070C0000 00018C68  SRB 00FEA560  R15 00018C68  R1 00FEA4F0  TYPE GLOBAL
EXT  ASCB 000106F0  CPU 0000  PSW 070C1004 00005F20  TCB N/A       TQE TCB 00010DF8
SRM  ASCB 0000FB48  CPU 0000  R15 00000000  R0 00010001  R1 00000000
SIO  ASCB 000106F0  CPU 0000  CPA 001ECF28 00FECF28  CAW 00003BA8  DEV ADD 0354  STATUS 0C00  CC 0
IO   ASCB 000106F0  CPU 0000  PSW 070C2000 0002208C  TCB 000108D8  DEV ADD 0354  CSW 001ECF50 0C000001  SNS N/A
SRB  ASCB 000106F0  CPU 0000  PSW 070C0000 00018C68  SRB 00FEA560  R15 00018C68  R1 00FEA4F0  TYPE GLOBAL
EXT  ASCB 000106F0  CPU 0000  PSW 070C1004 00018C68  TCB N/A       TQE TCB 00010DF8
SRM  ASCB 0000FB48  CPU 0000  R15 00000000  R0 00010001  R1 00000000
DSP  ASCB 00FED210  CPU 0000  PSW 070C0000 000124E4  TCB 00C9CB28  R15 00000000  R0 00327766  R1 00C9DEC8
DSP  ASCB 00FED210  CPU 0000  PSW 070C1000 00F6367C  TCB 00C9CB28  R15 00000000  R0 00000002  R1 00000018
EXT  ASCB 00FED210  CPU 0000  PSW 070C1004 00F6854C  TCB 00C9CB28  TQE TCB 00010DF8
SRM  ASCB 0000FB48  CPU 0000  R15 00000000  R0 00010001  R1 00000000
SRM  ASCB 0000FB48  CPU 0000  R15 00000000  R0 00010001  R1 00C00000
SRM  ASCB 0000FB48  CPU 0000  R15 00C00000  R0 00010001  R1 00000000
SRB  ASCB 000106F0  CPU 0000  PSW 070C0000 00008B70  SRB 00FE3004  R15 00008B70  R1 00000000  TYPE LOCAL
SRM  ASCB 0000FB48  CPU 0000  R15 00000000  R0 00010001  R1 00000000
DSP  ASCB 00FED210  CPU 0000  PSW 070C3000 00F6854C  TCB 00C9CB28  R15 00000004  R0 00000001  R1 00C9C94B
SRM  ASCB 0000FB48  CPU 0000  R15 00000000  R0 00010001  R1 00000000
SRB  ASCB 0000FB48  CPU 0000  PSW 070C0000 0004A67E  SRB 00C2C3D8  R15 0004A67E  R1 0068C0FD  TYPE GLOBAL
SRM  ASCB 0000FB48  CPU 0000  R15 00000000  R0 00010030  R1 00C2C3D8
SRM  ASCB 0000FB48  CPU 0000  R15 00000000  R0 00010001  R1 00000000
DSP  ASCB 00FED210  CPU 0000  PSW 070C3000 00F6854C  TCB 00C9CB28  R15 00000004  R0 00000001  R1 00C9C94B
EXT  ASCB 00FED210  CPU 0000  PSW 070C1004 00F6854C  TCB 00C9CB28  TQE TCB 00010DF8
SRM  ASCB 0000FB48  CPU 0000  R15 00000000  R0 00010001  R1 00000000
DSP  ASCB 00FED210  CPU 0000  PSW 070C3000 00F6854C  TCB 00C9CB28  R15 00000004  R0 00000001  R1 00C9C94B
DSP  ASCB 00FED210  CPU 0000  PSW 070C3000 00F6854C  TCB 00C9CB28  R15 00000004  R0 00000001  R1 00C9C94B
EXT  ASCB 00FED210  CPU 0000  PSW 070C1004 00F6854C  TCB 00C9CB28  TQE TCB 00010DF8
SRM  ASCB 0000FB48  CPU 0000  R15 00000000  R0 00010001  R1 00000000
DSP  ASCB 00FED210  CPU 0000  PSW 070C3000 00F6854C  TCB 00C9CB28  R15 00000004  R0 00000001  R1 00C9C94B
DSP  ASCB 00FED210  CPU 0000  PSW 070C0000 00F63706  TCB 00C9CB28  R15 00000000  R0 00000018  R1 00FFD2F0
DSP  ASCB 00FED210  CPU 0000  PSW 070C1000 00F63716  TCB 00C9CB28  R15 00000000  R0 00C9C9B0  R1 00C9C8D0
SRB  ASCB 0000FB48  CPU 0000  PSW 070C0000 000CC2DE  SRB 00FF1A50  R15 000CC2DE  R1 00FF1A7C  TYPE LOCAL
EXT  ASCB 0000FB48  CPU 0000  PSW 070C1004 0000BF1A  TCB N/A       TQE TCB 00010DF8
SRM  ASCB 0000FB48  CPU 0000  R15 00000000  R0 00010001  R1 00000000
DSP  ASCB 0000FB48  CPU 0000  PSW 070C1000 00F2D220  TCB 00010220  R15 6000BB94  R0 00000001  R1 FFFD74B8
PI   ASCB 0000FB48  CPU 0000  PSW 070C0011 00F2D220  TCB 00010220  VPA 00F2D220  R15 6000BB94  R1 FFFD74B8
PI   ASCB 0000FB48  CPU 0000  PSW 070C0011 00F2C000  TCB 00010220  VPA 00F2C000  R15 00F2C000  R1 80028CC8
PI   ASCB 0000FB48  CPU 0000  PSW 070C0011 00E273E0  TCB 00010220  VPA 00E273E0  R15 00F2C850  R1 000289D8
SIO  ASCB 0000FB48  CPU 0000  CPA 001F56D0 00FF56D0  CAW 001F56D0  DEV ADD 001F  STATUS 0C00  CC 0
DSP  ASCB 0000FB48  CPU 0000  PSW 070C1000 00E27992  TCB 00010220  R15 000353B8  R0 00FFFC58  R1 00C9C4E8
DSP  ASCB 0000FB48  CPU 0000  PSW 070C0000 00E2756E  TCB 00010220  R15 00000000  R0 00D1F7C8  R1 00D1F768
DSP  ASCB 0000FB48  CPU 0000  PSW 070C0000 00F2C9AA  TCB 00010220  R15 000353B8  R0 00D1F7C8  R1 00D1F768
DSP  ASCB 00FED210  CPU 0000  PSW 070C1000 00E4AB86  TCB 00C9CB28  R15 00011610  R0 00000000  R1 00FF1A50
DSP  ASCB 00FED210  CPU 0000  PSW 070C1000 00E4AB94  TCB 00C9CB28  R15 00000000  R0 00000014  R1 00C9F228
EXT  ASCB 00FED210  CPU 0000  PSW 070C1004 0002C69E  TCB 00C9CB28  TQE TCB 00010DF8
SRM  ASCB 0000FB48  CPU 0000  R15 00000000  R0 00010001  R1 00000000
LSR  ASCB 00FED210  CPU 0000  PSW 070C2000 0002C69E  TCB 00C9CB28  R15 00001000  R0 FF000010  R1 00C9DBE8
DSP  ASCB 00FED210  CPU 0000  PSW 070C0000 00E4ABD6  TCB 00C9CB28  R15 00000000  R0 FF000010  R1 00C71E20
DSP  ASCB 00FED210  CPU 0000  PSW 070C0000 000678FC  TCB 00C9CB28  R15 00000000  R0 FF000010  R1 0000017A
-RNIO ASCB 00FED210  CPU 0000                  IN F1F040C2 E8E3C5E2 0000
 EXT  ASCB 00FED210  CPU 0000  PSW 070C1004 000678EA  TCB 00C9CB28  TQE TCB 00010DF8
 SRM  ASCB 0000FB48  CPU 0000  R15 00000000  R0 00010001  R1 00000000
 SRB  ASCB 000106F0  CPU 0000  PSW 070C0000 00008B70  SRB 00FE3004  R15 00008B70  R1 00000000  TYPE LOCAL
 SIO  ASCB 000106F0  CPU 0000  CPA 001ECF28 00FECF28  CAW 00003BA8  DEV ADD 0354  STATUS 0C00  CC 0
```

Figure 1-14. Format of Minimal Trace Records for SIO, IO, SRB, EXT, SRM, DSP, PI, LSR, and RNIO

```
                                                    EXTERNAL TRACE - DD TRACE                          PAGE 0002
*** DATE    DAY 289     YEAR 1978            TIME 19.44.36.579936                                              ***
SLIP STD   ASCB 00024870  CPU 0000  JOBN *MASTER*  TID PER1  ASID 0001  JSP N/A       TCB N/A       MFLG 6236  EFLG 0000
           SFLG 00  DAUN 0000    MODN IEANUC01    OFFS 00068F62    IADR 00068F62 INS 904E33500590          EXSIAD N/A
           EXSINS N/A            BRNGA N/A        BRNGD N/A        OPSW 440C1000 00068F66  PI ILC 00040080  PERC 40  TYP 88
SLIP STD   ASCB 00024870  CPU 0000  JOBN *MASTER*  TID PER1  ASID 0001  JSP N/A       TCB N/A       MFLG 6236  EFLG 0000
           SFLG 00  DAUN 0000    MODN IEANUC01    OFFS 00068F80    IADR 00068F80 INS 47F0902858FB          EXSIAD N/A
           EXSINS N/A            BRNGA N/A        BRNGD N/A        OPSW 440C0000 00068F90  PI ILC 00040080  PERC 80  TYP 88
SLIP STD   ASCB 00024870  CPU 0000  JOBN *MASTER*  TID PER1  ASID 0001  JSP N/A       TCB N/A       MFLG 6236  EFLG 0000
           SFLG 00  DAUN 0000    MODN IEANUC01    OFFS 00068FAA    IADR 00068FAA INS 4770908E58A0          EXSIAD N/A
           EXSINS N/A            BRNGA N/A        BRNGD N/A        OPSW 440C3000 00068FF6  PI ILC 00040080  PERC 80  TYP 88
SLIP STD   ASCB 00024870  CPU 0000  JOBN *MASTER*  TID PER1  ASID 0001  JSP N/A       TCB N/A       MFLG 6236  EFLG 0000
           SFLG 00  DAUN 0000    MODN IEANUC01    OFFS 0006901A    IADR 0006901A INS 47F090E89120          EXSIAD N/A
           EXSINS N/A            BRNGA N/A        BRNGD N/A        OPSW 440C1000 00069050  PI ILC 00040080  PERC 80  TYP 88
```

Figure 1-15. Format of SLIP Standard Trace Record

```
SLIP S+U   ASCB 00024870  CPU 0000  JOBN *MASTER*  TID PER2  ASID 0001  JSP N/A       TCB N/A       MFLG 6236  EFLG 0000
           SFLG 40  DAUN 0000    MODN IEANUC01    OFFS 00040698    IADR 00040698 INS D203F07C3014          EXSIAD N/A
           EXSINS N/A            BRNGA 00060000   BRNGD 000C0A5F   OPSW 440C1000 0004069E  PI ILC 00060080  PERC 20  TYP 28
           R0 00000000  R1 0003DB20  R2 0003D428  R3 00000C5C  R4 00FD7D18  R5 00024870  R6 00FD7D70  R7 A0040068
           R8 7003E31C  R9 00000000 R10 00808000 R11 00800000 R12 6004001A R13 000616F8 R14 5004036E R15 000616F8
           46040C00 000002E5 DA000000 00000000 00000257 78000000 00070C00 0000BAA7 28070C10 000006FE 1C440C10 00000406
           9E000000 00000000 00070C00 0000BAA7 280008BC 300C00
SLIP S+U   ASCB 00024870  CPU 0000  JOBN *MASTER*  TID PER2  ASID 0001  JSP N/A       TCB N/A       MFLG 6236  EFLG 0000
           SFLG 40  DAUN 0000    MODN IEANUC01    OFFS 00042B3C    IADR 00042B3C INS 900EF040D203          EXSIAD N/A
           EXSINS N/A            BRNGA 00060000   BRNGD 000C0A5F   OPSW 440C1000 00042B40  PI ILC 00040080  PERC 20  TYP 28
           R0 00000000  R1 0003DB10  R2 0003D428  R3 00000C5C  R4 00FD7928  R5 00024870  R6 00FD7980  R7 A0040068
           R8 7003E31C  R9 00000000 R10 01008000 R11 01000000 R12 6004001A R13 000616F8 R14 5004036E R15 000616F8
           46040C00 000002E5 DA000000 00000000 00000257 78000000 00070C00 0000BAA7 28070C10 000006FE 1C440C10 0000042B
           40000000 00000000 00070C00 0000BAA7 280008BC 300C00
SLIP S+U   ASCB 00024870  CPU 0000  JOBN *MASTER*  TID PER2  ASID 0001  JSP N/A       TCB N/A       MFLG 6236  EFLG 0000
           SFLG 40  DAUN 0000    MODN IEANUC01    OFFS 00042B40    IADR 00042B40 INS D203F07C3014          EXSIAD N/A
           EXSINS N/A            BRNGA 00060000   BRNGD 000C0A5F   OPSW 440C1000 00042B46  PI ILC 00060080  PERC 20  TYP 28
           R0 00000000  R1 0003DB10  R2 0003D428  R3 00000C5C  R4 00FD7928  R5 00024870  R6 00FD7980  R7 A0040068
           R8 7003E31C  R9 00000000 R10 01008000 R11 01000000 R12 6004001A R13 000616F8 R14 5004036E R15 000616F8
           46040C00 000002E5 DA000000 00000000 00000257 78000000 00070C00 0000BAA7 28070C10 000006FE 1C440C10 0000042B
           46000000 00000000 00070C00 0000BAA7 280008BC 300C00
```

Figure 1-16. Format of SLIP Standard/User Trace Record

```
*** DATE    DAY 291     YEAR 1978            TIME 19.51.03.925752                                                    ***
SLIP USR   40000000 0600FF15 6800FD78 380004DA 4000FD73 58000000 0000FECE 0800FD79 D000069F 66500068F 6800000C 5C000000
           0400FD79 D0000000 005007BB 5E00068F 62BE040C 00000002 E5DA0000 00000000 00000002 57780000 0000070C 000000B8
           9000070C 10000006 FE1C440C 10000006 8F780000 00000000 0000070C 30000001 D156001F 68480C00 0001001F 68300002
           5778F56C 063A00FD 7DC0440C 00000006 DB40440C 00000002 7448000C 00000002 FE3A0008 00000003 0440440C 00000006
           DE400000 00000000 10040002 000D0004 008000AA 0FB00004 40000006 8F740000 00060000 00000006 6F081002 0000001E
           AB70FFFF FFFF0000 00000000 00090000
SLIP USR   40000000 0600FF15 6800FD78 380004DA 4000FD73 58000000 0000FECE 0800FD79 D000069F 66500068F 6800000C 5C000000
           0400FD79 D0000000 005007BB 5E00068F 62BE040C 00000002 E5DA0000 00000000 00000002 57780000 0000070C 000000B8
           9000070C 10000006 FE1C440C 00000006 8F7E0000 00000000 0000070C 30000001 D156001F 68480C00 0001001F 68300002
           5778F56C 063A00FD 7DC0440C 00000006 DB40440C 00000002 7448000C 00000002 FE3A0008 00000003 0440440C 00000006
           DE400000 00000000 10040002 000D0006 008000AA 0FB00004 40000006 8F780000 00060000 00000006 6F081002 0000001E
           AB70FFFF FFFF0000 00000000 00090000
SLIP USR   40000000 0600FF15 6800FD78 380004DA 4000FD73 58000000 0000FECE 0800FD79 D000069F 66500068F 6800000C 5C00FF15
           6800FD79 D0000000 005007BB 5E00068F 62BE040C 00000002 E5DA0000 00000000 00000002 57780000 0000070C 000000B8
           9000070C 10000006 FE1C440C 00000006 8F800000 00000000 0000070C 30000001 D156001F 68480C00 0001001F 68300002
           5778F56C 053A00FD 7DC0440C 00000006 DB40440C 00000002 7448000C 00000002 FE3A0008 00000003 0440440C 00000006
           DE400000 00000000 10040002 000D0002 008000AA 0FB00004 40000006 8F7E0000 00060000 00000006 6F081002 0000001E
           AB70FFFF FFFF0000 00000000 00090000
```

Figure 1-17. Format of SLIP User Trace Record

```
                                                    EXTERNAL TRACE - DD TRACE                          PAGE 0008
SLIP STD   ASCB 00024870  CPU 0000  JOBN *MASTER*  TID PER6  ASID 0001  JSP N/A       TCB N/A       MFLG 6236  EFLG 0000
           SFLG 00  DAUN 0000   MODN IEANUC01   OFFS 00068F78   IADR 00068F78 INS D72B337C337C          EXSIAD N/A
           EXSINS N/A           BRNGA N/A       BRNGD N/A       OPSW 440C0000 00068F7E  PI ILC 00060080  PERC 40  TYP 48
SLIP S+U   ASCB 00024870  CPU 0000  JOBN *MASTER*  TID PER6  ASID 0001  JSP N/A       TCB N/A       MFLG 6236  EFLG 0000
           SFLG 80  DAUN 0000   MODN IEANUC01   OFFS 00068F78   IADR 00068F78 INS D72B337C337C          EXSIAD N/A
           EXSINS N/A           BRNGA N/A       BRNGD N/A       OPSW 440C0000 00068F7E  PI ILC 00060080  PERC 40  TYP 48
           020000
SLIP STD   ASCB 00024870  CPU 0000  JOBN *MASTER*  TID PER6  ASID 0001  JSP N/A       TCB N/A       MFLG 6236  EFLG 0000
           SFLG 00  DAUN 0000   MODN IEANUC01   OFFS 00068F7E   IADR 00068F7E INS 18B147F09028          EXSIAD N/A
           EXSINS N/A           BRNGA N/A       BRNGD N/A       OPSW 440C0000 00068F80  PI ILC 00020080  PERC 40  TYP 48
SLIP S+U   ASCB 00024870  CPU 0000  JOBN *MASTER*  TID PER6  ASID 0001  JSP N/A       TCB N/A       MFLG 6236  EFLG 0000
           SFLG 80  DAUN 0000   MODN IEANUC01   OFFS 00068F7E   IADR 00068F7E INS 18B147F09028          EXSIAD N/A
           EXSINS N/A           BRNGA N/A       BRNGD N/A       OPSW 440C0000 00068F80  PI ILC 00020080  PERC 40  TYP 48
           020000
SLIP STD   ASCB 00024870  CPU 0000  JOBN *MASTER*  TID PER6  ASID 0001  JSP N/A       TCB N/A       MFLG 6236  EFLG 0000
           SFLG 00  DAUN 0000   MODN IEANUC01   OFFS 00068F80   IADR 00068F80 INS 47F0902858FB          EXSIAD N/A
           EXSINS N/A           BRNGA N/A       BRNGD N/A       OPSW 440C0000 00068F90  PI ILC 00040080  PERC 40  TYP 48
SLIP S+U   ASCB 00024870  CPU 0000  JOBN *MASTER*  TID PER6  ASID 0001  JSP N/A       TCB N/A       MFLG 6236  EFLG 0000
           SFLG 80  DAUN 0000   MODN IEANUC01   OFFS 00068F80   IADR 00068F80 INS 47F0902858FB          EXSIAD N/A
           EXSINS N/A           BRNGA N/A       BRNGD N/A       OPSW 440C0000 00068F90  PI ILC 00040080  PERC 40  TYP 48
           020000
```

Figure 1-18. Format of SLIP DEBUG Trace Record

Control Records

GTF produces three types of control records: time stamp records, lost event records, and lost block records. The first record in every block of trace output is a time stamp record. A lost data record tabulates trace events that were not recorded because the GTF buffers were full. Figure 1-19 shows the general format of a time stamp record.

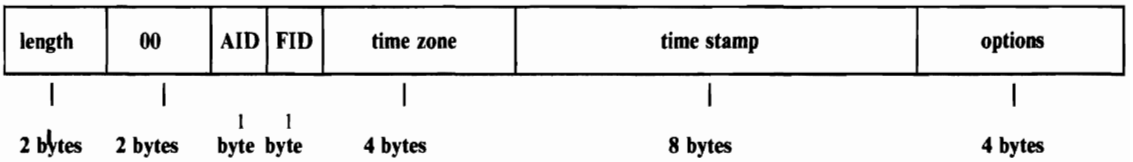


Figure 1-19. General Format of a Time Stamp Control Record

The fields in the record contain the following information:

length
indicates total length of the record in bytes.

00
always zero.

AID
always zero, for control records.

FID
is the format identifier, which for time stamp control records, is always X'01'.

time zone
is a copy of the CVT field CVTTZ and is the difference between local time and Greenwich mean time (GMT) in binary units of 1.048576 seconds.

time stamp
indicates the TOD clock value representing the local time when the control record was constructed. (TOD clock values are described in *IBM S/370 Principles of Operation.*)

PRDMP formats and prints the TOD clock value in a time stamp record that precedes the printout of each GTF trace buffer. The time stamp record is described in *MVS/370 Debugging Handbook, Volume 1.*

options
identifies the GTF options in effect. For detailed information about this field, see Figure A-2 in "Appendix A: Writing EDIT User Programs."

Chapter 2. LIST

Introduction

LIST is a service aid that operates as a problem program named AMBLIST. It produces several kinds of output that you can use 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. You can examine the listing 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.

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 LIST must have the same format as those created by the linkage editor.

JCL Statements

The minimum region for executing AMBLIST is 64K for all functions except LISTLPA, which requires 100K. See the next topic, ***Control Statements*** .

LIST requires the following JCL statements:

JOB Statement
initiates the job.

EXEC Statement
provides for the execution of the program AMBLIST and is used to set the appropriate region size for your program to run.

SYSPRINT DD Statement
defines the message data set.

anyname DD Statement
defines an input data set.

Note: Do not concatenate input DD statements; the results are unpredictable.

SYSIN DD Statement
defines the instream data set 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 does not fit on a single record, end the first record with a comma or a non-blank character in column 72 and continue on the next record. Begin all continuation records in columns 2 - 16. You must not split parameters between two records; the only exception is the MEMBER parameter, 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}
indicates the type of load module listing to be produced.

MODLIST
requests a formatted listing of the control and text records of a load module, including its External Symbol Dictionary and Relocation Dictionary records.

XREF
requests a module map and cross-reference listing for the load module.

BOTH
requests both a formatted listing of the load module and its map and cross-references.

Note: If this parameter is omitted, OUTPUT = BOTH is assumed.

TITLE = ('title',position)

title
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.)

position

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 subparameter to specify the number of characters that should be left blank before the title. If you specify a position greater than 80, the indentation defaults to 1.

Note: Do not punctuate your title with commas; because LIST recognizes a comma as a delimiter, anything that follows an embedded comma in a title is ignored.

DDN=ddname

identifies the DD statement that defines the data set containing the input module. If the DDN parameter is omitted, LIST assumes SYSLIB as the default ddname.

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

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

RELOC=hhhhhh

specifies a relocation or base address of up tosix hexadecimal digits. 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)

title

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.)

position

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 subparameter to specify the number of characters that should be left blank before the title. If you specify a position greater than 80, the indentation defaults to 1.

Note: Do not punctuate your title with commas; because LIST recognizes a comma as a delimiter, anything that follows an embedded comma in a title is ignored.

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

MEMBER = {member|(member1[,membern]...)}
identifies the input object module(s) by membername or alias name. To specify more than one object module, enclose the list of names in parentheses and separate the names with commas. CAUTION: You must include the MEMBER parameter when the input object modules exist as members in a partitioned data set. If you do not include the MEMBER parameter, LIST assumes 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}
indicates whether LIST should print all CSECT identification records or only those containing AMASPZAP data and user data.

ALL
indicates that all IDRs associated with the module are to be printed.

IDENT
indicates that LIST should print only those IDRs that contain SPZAP data or user-supplied data.

Note: If you omit this parameter, LIST assumes a default of OUTPUT = ALL. Do not specify OUTPUT if you specify the MODLIB parameter.

TITLE = ('title',position)

title
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.)

position

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 subparameter to specify the number of characters that should be left blank before the title. If you specify a position greater than 80, the indentation defaults to 1.

Note: Do not punctuate your title with commas; because LIST recognizes a comma as a delimiter, anything that follows an embedded comma in a title is ignored.

Note: There is no point in specifying TITLE if you also specify the MODLIB parameter, because LIST then ignores TITLE.

DDN=ddname

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

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

identifies the input load module(s) by membername or alias name. To specify more than one load module, enclose the list of names in parentheses and separate the names with commas. If you omit the MEMBER parameter, LIST prints 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
```

LISTLPA

lists the link pack area (LPA) only. It does not list the modified link pack area (MLPA). LISTLPA has no parameters.

Note: The LIST reflects only the system currently operating.

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 is produced, and the second page of the listing is the beginning of the formatted output.

The module summary gives the following information:

- the member name (with aliases)
- the entry point
- linkage editor attributes
- system status index (SSI) information

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   PL1LOAD                          MAIN ENTRY POINT  000720

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

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

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

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

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

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

                          MODULE SSI:  NONE
```

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

For LISTOBJ, the listing contains information specific to the object module:

- the object module listing
- its external symbol dictionary (ESD)
- its relocation dictionary (RLD)

Figure 2-4 shows an example of LISTOBJ output.

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

For LISTLPA, the second page of the listing is a map of the link pack area, with modules in alphabetical order. Figure 2-6 shows an example of LISTLPA output.

For complete descriptions of the fields in the formatted output listings, see *Linkage Editor Logic*.

```
                                LISTING OF LOAD MODULE PL1LOAD                                  PAGE 0001


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

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

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

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

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

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

```
                                LISTING OF LOAD MODULE PL1LOAD                                  PAGE 0002


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

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

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

Figure 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 0000063E
            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
                 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       IHEQLW0        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 0003


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

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

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

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

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

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

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

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

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

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

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

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

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

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

Figure 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

```
                LINK PACK MAP - ALPHABETICALLY BY NAME

NAME     LOCATION  LENGTH    EP ADDR  MAJOR LPDE NAME      NAME     LOCATION  LENGTH    EP ADDR  MAJOR LPDE NAME
CHLOADTB FDFFE0    000020    FDFFE0                        DCM2B0   FDFA28    0005B8    FDFA28
DCM2B1   FDF470    0005B8    FDF470                        DCM2D2   FDD000    001280    FDD000
DCM2D3   FDB000    001280    FDB000                        DCM3B0   FDAA48    0005B8    FDAA48
DCM3B1   FDA490    0005B8    FDA490                        DCM3D2   FD8000    001280    FD8000
DCM3D3   FD6000    001280    FD6000                        DCM3E0   FD5A48    0005B8    FD5A48
DCM3E1   FD5490    0005B8    FD5490                        DCM3E2   FD4A48    0005B8    FD4A48
DCM3E3   FD4490    0005B8    FD4490                        DCM4B0   FD3A48    0005B8    FD3A48
DCM4B1   FD3490    0005B8    FD3490                        DCM4D2   FD1000    001280    FD1000
DCM4D3   FCF000    001280    FCF000                        DCM4E0   FCEA48    0005B8    FCEA48
DCM4E1   FCE490    0005B8    FCE490                        DCM4E2   FCDA48    0005B8    FCDA48
DCM4E3   FCD490    0005B8    FCD490                        DEVMASKT FCC678    000988    FCC678
DEVNAMET FCC4B8    0001C0    FCC4B8                        EMODVOL1                       FCC0B8    IFG0552J
IEECB860 FCBCD8    000328    FCBCD8                        IEECVGCI FCB800    0004D8    FCB800
IEELWAIT FCB2F8    000508    FCB2F8                        IEEPALTR FCB270    000088    FCB270
IEEPDISC FCB210    000060    FCB210                        IEEPPRES FCB1A8    000068    FCB1A8
IEEPRTN  FCB0D8    0000D0    FCB0D8                        IEEPRWI2 FCAEE8    000118    FCAEE8
IEEPSN   FCADC0    000128    FCADF0                        IEEQALTR FC6000    003BE8    FC9680
IEERGN   FC5F30    0000D0    FC5F30                        IEESB665 FC5AA8    000488    FC5AA8
IEESMFWR FC54A0    000608    FC54A0                        IEEVDSP1 FC4A08    0005F8    FC4A08
```

Figure 2-6. Sample LISTLPA Output

Examples

The following examples show ways to use LIST.

Example 1: Listing Several Object Modules

In this example, LIST lists all the 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 its members collectively 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 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 produces 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 instream data set 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 are 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, that contains load modules to be processed.

LOADLIB DD Statement
defines a second input data set.

SYSIN DD Statement
defines the instream data set 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 because no DDN parameter was included). LISTIDR 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 formats and lists 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 an instream input data set; in this case, an object deck.

SYSIN DD Statement
defines an instream input data set, which contains LIST control statements.

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 more than one module. Assume that an unsuccessful attempt was 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 an instream input data set that contains the LIST control statements.

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.

```
//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 an instream input data set that contains the LIST control statements.

LISTLOAD Control Statement

instructs LIST to format the control and text records. These records include the external symbol dictionary and relocation dictionary records for the load module IEANUC01, in the data set defined by the SYSLIB DD statement. LISTLOAD 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. PRDMP

Introduction

AMDPRDMP is an OS/VS2 service aid program that provides the facilities to:

- Format and print dump data sets created by AMDSADMP, SVC dump, and SYSMDUMP.
- Edit and print system and user trace records created by GTF.
- Transfer the contents of a SYS1.DUMP data set to another data set, freeing the SYS1.DUMP data set for reuse by SVC dump.

PRDMP only processes dumps taken by OS/VS2 dump programs that are at the same release level as PRDMP.

Using PRDMP

The operation and formatting done by PRDMP is controlled by user-supplied control statements. You enter these control statements either through the console or in the job stream with job control language statements.

In all cases, the PRDMP input is a partial system dump; the content of this dump determines the capability of PRDMP to perform its formatting of control blocks. If input is a real storage dump, PRDMP attempts to provide virtual address formatting and printing by performing address translation for the real storage contents.

The dump data sets which PRDMP processes consist of the following:

- SADMP unformatted, machine-readable dump data set, including instruction trace data (created by the console-initiated loop recording) and page data sets. (**Note:** Address translation is performed on the real storage dump portion if you request any PRDMP format control statement except PRINT REAL. If the translation data (such as the ASCBSTOR, the segment table, or page tables) is invalid, PRDMP cannot print the virtual storage. In this case, you may be able to examine the address space's virtual storage by looking in the page table frame entries (PTFEs) for the real frames assigned to the address space and specifying PRINT REAL=address range to print these pages.

- SYS1.DUMP data set created by SVC dump. This type of dump input includes only virtual addresses and instruction trace data (created by the console-initiated loop recording).

  Note: Up to 10 SYS1.DUMP data sets may be defined to OS/VS2 and must be named SYS1.DUMP00, SYS1.DUMP01, through SYS1.DUMP09.

- Private data set containing a dump produced by SVC dump. This type of dump input includes only virtual addresses and instruction trace data (created by the console-initiated loop recording).

- SYSMDUMP dump data set. This type of dump input is processed by virtual addresses only.

- SADMP formatted dump data set which has been written to tape.

The GTF trace data which PRDMP processes consists of:

- GTF external trace data set (usually called SYS1.TRACE).

- GTF trace data in buffers within a dump of real storage. (**Note:** For GTF trace records to be included in the dump data set processed by PRDMP, GTF must have been active at the time the dump was taken. In addition, if the dump is produced by SVC dump or SNAP macro, the SDATA=TRT option must be specified.)

Figure 3-1 shows the general characteristics of these types of input and how they relate to PRDMP processing. Note that AMDPRDMP does not allow an exit routine to receive control after processing GTF records.

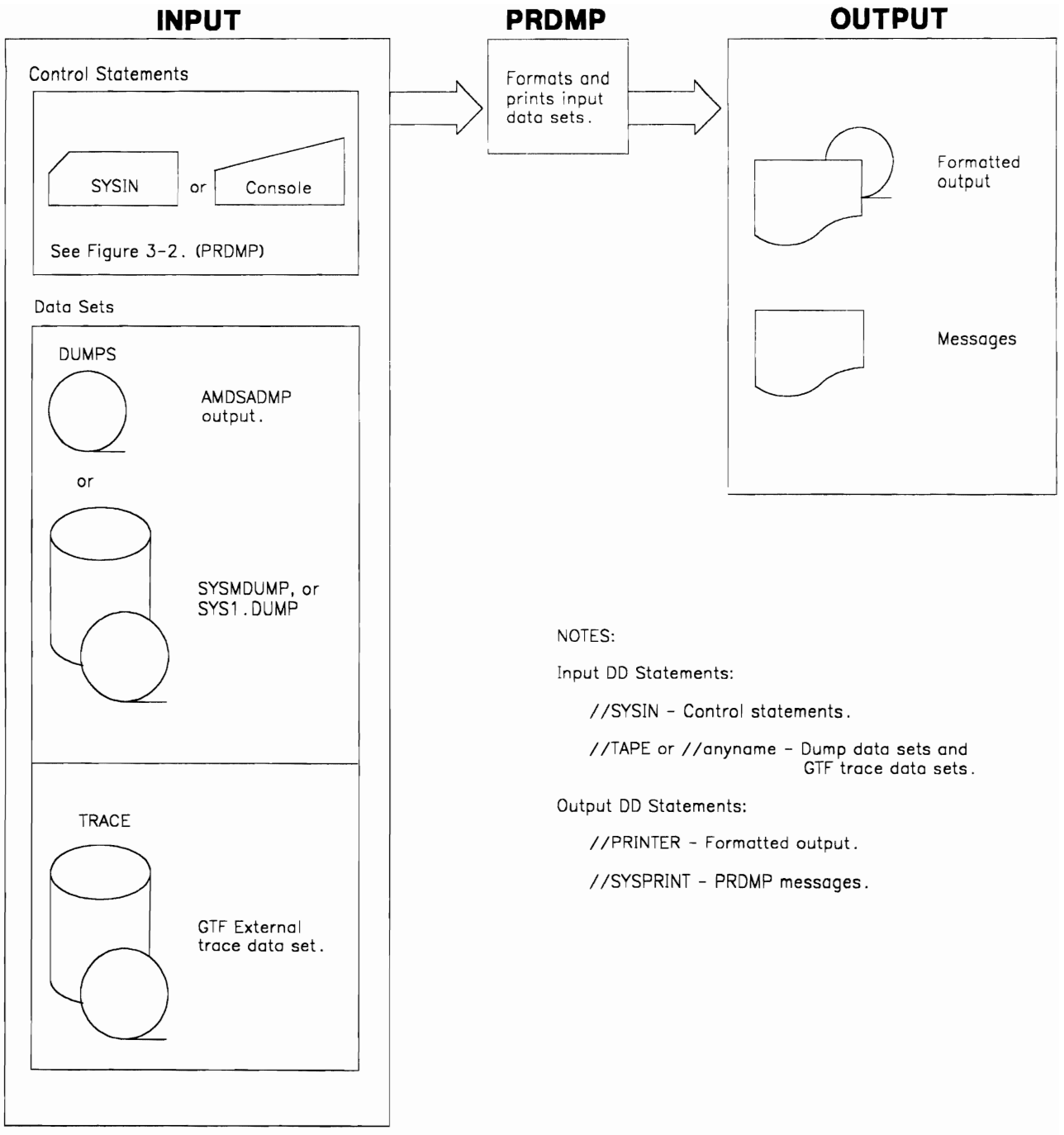


Figure 3-1. PRDMP Input and Output

Functions of PRDMP

PRDMP control statements allow you to select PRDMP functions to vary the formatting and printing of its input. The functions provided by PRDMP control statements are:

Format Control Blocks

Major system control blocks for each virtual address space in the system may be formatted on the output listing. The extent to which formatting can be performed is a function of the input. If the input, for example, consists of all virtual storage related to a single virtual address space and you request formatting of control blocks for all virtual address spaces, your request cannot be completely satisfied. Control blocks may also be formatted on an address space basis by requesting the current address spaces or the address space containing a certain jobname to be printed. (See PRINT CURRENT or PRINT JOBNAME). A printing of the communications vector table and of selected processor-related data are formatting options available to you. (See CVTMAP and CPUDATA control statements). If the input to AMDPRDMP is a low-speed formatted dump, produced by AMDSADMP, then the preformatted records are printed as they appear in the input. No additional formatting is provided by PRDMP. (See the FORMAT control statement).

Provide Address Space Summary Information

A synopsis of the normal and abnormal status of each virtual address space in the system may be provided in order to:

- Limit the amount of output provided by PRDMP.
- Provide gross address space relationships which serve as a starting point for problem debugging.
- Establish a base from which decisions concerning further formatting can be made. (See SUMMARY control statement).

Select Storage Locations to Print

By exercising certain options provided by the PRINT control statements, the user may specify that only those portions of real or virtual storage that are relevant to his problem be printed on the output listing. The user selects what information is to be printed by specifying the beginning and ending real or virtual addresses of those areas of storage. The virtual address ranges are qualified to a particular address space by specifying an address space identifier. If no address space identifier is provided,AMDPRDMP will attempt to print the four cross-memory address spaces that are current at the time the dump was taken. If these address spaces are not in the dump, or if AMDPRDMP cannot identify them the virtual storage ranges default to the address space that was current at the time the dump was taken. The user may also specify that only those areas of storage associated with the nucleus or system queue area be printed. You may also request printing

of those areas of virtual storage associated with a particular address space, the current address space, or a specific job. (See the PRINT control statement).

Map the Link Pack Area Active Queue

A map of the modules comprising the link pack area active queue may be provided on a separate page or pages of the PRDMP formatted listing. This map describes the reenterable system modules from the link pack area that were in use when the dump was taken. It is useful in diagnosing system failures that occurred in program modules residing outside the user's region. The entire link pack area may be mapped using AMBLIST, which is described in Chapter 2. (See the LPAMAP control statement).

Fast Dump Scan

By exercising the options of fast dump scan, the user may display specific areas of storage quickly and conversationally on either a TSO terminal or the system console. This is performed by the DISPLAY function of PRDMP. (See the DISPLAY control statement).

Print Queue Control Blocks

Formatted queue control blocks for all task control blocks in the system are printed on a separate page or pages of the PRDMP listing. The address of the major control blocks associated with global resource serialization and the contents of the control blocks on the global and local resource queues is known as a QCBTRACE. The QCBTRACE may be used to resolve problems arising from task contention or system interlock. (See the GRSTRACE or QCBTRACE control statement.)

Format Generalized Trace Facility (GTF) Trace Output Records

Formatting and printing of GTF trace output records will be performed by the EDIT function of PRDMP. These trace output records may be in either a GTF trace data set or in the GTF trace buffers which are a part of a system dump. (See the EDIT control statement).

Clear SYS1.DUMP Data Sets

You can use AMDPRDMP to transfer the contents of a SYS1.DUMP data set to another data set for later formatting and printing.

When a SYSUT2 DD statement is included in the AMDPRDMP job control statements, PRDMP transfers the contents of the input data set to the data set specified by SYSUT2. If the input data set is a SYS1.DUMP data set, PRDMP makes it available for SVC dump to reuse.

If a SYSUT2 DD statement is present, no other processing of the input data set is performed by PRDMP. PRDMP control statements are ignored.

AMDPRDMP Return Codes

AMDPRDMP return codes are:

Code	Description
0	AMDPRDMP request was successful.
4	The data set processed, or one of multiple data sets input, was empty.
8	An uncorrectable I/O error was encountered.

Specify the Communications Vector Table Address

If you believe that the pointer to the communications vector table (CVT) residing at storage location X'4C' is destroyed or unavailable, you may bypass the pointer and supply the PRDMP with the address of the CVT. (See the CVT control statement).

Print a User-Specified Title

A title may be specified by the user to be printed at the top of each page of the dump listing. This title is provided in addition to the title supplied by the dump program. The AMDPRDMP title may be changed during the execution of the AMDPRDMP program. Page length may also be controlled by specifying the number of lines per page to be printed. (See the title Control Statement and the PARM = T parameter on the START command).

Provide User Interface

The PRDMP user interface allows you to write formatting routines tailored to special interests while taking advantage of the basic PRDMP facilities to retrieve data from the input dump file. A user routine receives control either because its associated control statement (verb) is specified or because a user exit out of PRDMP has been encountered. (See the "Appendix C: PRDMP User Exit Facility").

Provide High-Density Dump Output

High-density dump output can be obtained with 64 bytes of storage per line of dump, as opposed to the normal 32 bytes. Dumps can also be printed at 8 lines per inch, 80 lines per page on 11-inch long paper. The first capability exists only when using the IBM 3800 Printing Subsystem (see PRINTER under "Output DD Statements" later in this chapter).

How to Specify JCL Statements for PRDMP

Job control statements are important in determining what functions PRDMP is to perform. This section describes the JCL statements that are necessary if you use PRDMP. PRDMP is started at the console or by statements in the input stream; the JCL may be in a cataloged procedure, or included in the input stream. Figure 3-2 shows the JCL statements necessary for PRDMP execution. For more complete information about using JCL, refer to the publication, *MVS JCL.*

```
//jobname     JOB
[/*JOBPARM    LINECT=80]
//stepname    EXEC        PGM=AMDPRDMP
                          [,PARM='[T][,LINECNT=nn][,S][,ER=x]']
[//TAPE       DD          Note: Use the parameters that
                          describe the dump data set or
                          the trace data set created by
                          Generalized Trace Facility.
                          Generalized Trace Facility.]
//PRINTER     DD          SYSOUT=x[,CHARS=DUMP][,FCB=STD3]
[//SYSPRINT   DD          SYSOUT=x]

//SYSUT1      DD          UNIT=xxxx,SPACE=(CYL,(xx,yy))
//SYSUT2      DD          VOL=SER=xxxxxx,UNIT=xxxx,DISP=(NEW,KEEP),
                          DSNAME=xxxxxx,SPACE=(CYL,(xx,yy),RLSE)
                          LABEL=([n],xx)
[//SYSIN      DD          *]
          (control statements)
[/*]
```

Figure 3-2. JCL Statements Required for PRDMP Execution

The descriptions which follow explain each statement in Figure 3-2. These descriptions are in three groups: JOB and EXEC statements, input DD statements, and output DD statements.

JOB and EXEC Statements

These statements are coded as follows:

//jobname JOB
identifies the AMDPRDMP job to the system. Because AMDPRDMP executes in a minimum of 128K of virtual storage, a region size of at least 128K should be specified on the JOB statement. If the region is allowed to default, or the user specifies a region size less that 128K, the results are unpredictable. For more complete information on region see "Limiting User Region Size." in *OS/VS2 SPL: Supervisor.*

[/*JOBPARM LINECT=80]
this JES2 statement sets the page overflow line count value to 80 lines per page for the current job when using the IBM 3800 Printing Subsystem. This statement ensures that the expected number of lines per page are printed for explicit or implicit specifications of the PRDMP LINECNT parameter for line count values up to and including 80 lines per page. For more information about the /*JOBPARM control statement, see *MVS JCL.*

//stepname EXEC PGM=AMDPRDMP[,PARM='value']
provides for execution of AMDPRDMP. The operands associated with this statement are:

PGM=AMDPRDMP
supplies the name of the program (AMDPRDMP).

[,PARM='[T][,LINECNT=nn][,S][,ER=x]']
specifies a list of parameters to be passed to PRDMP. If you use the PARM= operand, parameters may appear in any order but must be enclosed in apostrophes and separated by commas with no intervening blanks. (Do not code a comma between PARM= and the first parameter specified.) The valid parameters and their meanings are:

T
specifies that the user wishes a WTOR to be issued to the operator requesting a dump title (message AMD154D REPLY TITLE, 'SAME', or 'END'). The title may contain a maximum of 64 characters. If T is not specified, no prompting will occur.

LINECNT=nn
allows the user to specify the number of lines per page to be printed on the output listing. The value specified for nn may be any decimal integers greater than 10 and less than 81. If this parameter is not specified or is invalid, 58 is the default.

S
If S is specified, message AMD156I REPLY WITH STOP TO TERMINATE CURRENT FUNCTION is issued to the primary console. If at anytime you reply STOP, processing of the current function terminates and the next control statement is processed or message AMD155D is issued to request more control statements.

ER = x

specifies the PRDMP edit program action to be taken if, while editing trace data, an error is detected in an exit routine or a format routine. The valid values for this parameter and their meanings are:

0 -- EDIT displays in hexadecimal the record associated with the error and ignores the faulty routine in subsequent processing. If the error is in a format routine, all subsequent records that require processing by the same format routine are ignored. If the error is in an exit routine, record formatting continues.

1 -- EDIT displays in hexadecimal, the record associated with the error and ignores the faulty routine in subsequent processing. If the error is in a format routine, all subsequent records that require processing by the same format routine are dumped in hexadecimal. If the error is in an exit routine, record formatting continues.

2 -- EDIT displays in hexadecimal the record associated with the error; EDIT then terminates, and the next PRDMP verb is executed.

3 -- EDIT allows ABEND to get control if a program check occurs in an exit or format routine. (If ER = 3 is not specified, EDIT issues the SPIE macro each time before entering the exit routine or format appendage and thus bypasses ABEND processing. Issuance of the SPIE causes a considerable increase in the time it takes to run the EDIT program.) If the recognized error is not a program check, the associated record is dumped in hexadecimal; then EDIT terminates and the next PRDMP verb is executed.

If this value is not included in the PARM = parameter list, a value of ER = 2 is assumed. Note that ER = 0 and ER = 1 are the same for exit programs.

Input DD Statements

The two input statements are coded as follows:

//TAPE DD or
//anyname DD

One of these DD statements is used to describe each of the input dump or GTF trace data sets. If these statements are omitted, PRDMP assumes the dump or trace information is already on the SYSUT1 work data set.

If the input is a GTF trace data set, the ddname must be the same as the one specified in the DDNAME parameter of the EDIT control statement. You can define any number of trace data sets, provided that you identify each data set with a unique ddname and a separate EDIT control statement. (Note that you can use the same ddname for several trace data sets, as long as you provide a new tape volume for each.)

If the input data set defined by one of these statements resides on a direct access device, a SYSUT1 DD statement is not needed.

If the input data set is a dump, you can specify any ddname. Remember, however, that for ddnames other than TAPE, you must use a NEWDUMP control statement to identify the input data set. You can define any number of input data sets, as long as each is identified by a different ddname, and each ddname is specified in a separate NEWDUMP control statement.

When using the anyname DD statement, you must use the NEWDUMP control statement. The ddname(anyname) must be the same as the ddname used on the NEWDUMP control statement. The anyname may not be a ddname normally used to execute PRDMP, such as SYSUT1, SYSPRINT, PRINTER, SYSUT2, or SYSIN. The DD parameters required to define the input data set are as follows:

DSNAME = dsname

required for data sets on direct access devices only. This subparameter supplies the name of the input dump or trace data set.

VOL = SER = (serial-number[,serial-number]...)

specifies the volume serial numbers for the volumes on which the input dump or trace data set is written. You specify these volumes in the order in which they were created.

UNIT = (device-type,P)

informs the operating system that the input dump or trace data set is being entered from a particular I/O device or device type. "P" indicates a request for parallel mounting. If P is coded, the system counts the number of serial numbers specified in the VOLUME parameter and assigns to the data set as many devices as there are serial numbers. It is recommended that P be coded if PRDMP must format dump data sets directly from the tape input.

LABEL = ([data-set-seq-#][,NL|,BLP])

only required for data sets on magnetic tape. The first subparameter is the file number field. When using an unlabelled tape, specify NL; BLP indicates a labelled tape.

DISP = OLD

indicates that the data set associated with this DD statement already exists.

Note: Do not omit the TAPE DD statement unless you supply a NEWDUMP control statement. If you do not define the input data set, PRDMP assumes that the input is in the SYSUT1 data set.

[//SYSIN DD]

describes the PRDMP control statement data set. If control statements are not supplied or are exhausted before an END verb is encountered, the operator will be asked for additional control input (message AMD155D REPLY WITH GO, DESIRED FUNCTION, or END). The most common operand for this DD statement is: *. This operand indicates that the data comprising this data set is contained in the input stream.

Output DD Statements

The five output DD statements are coded as follows:

//PRINTER DD

defines the PRDMP output data set. The DD parameters for this statement are:

SYSOUT = x

specifies the system output class through which the PRDMP output is routed. 'x' is the character that identifies the output class. *Note:* Default blocking is RECFM = FBA, LRECL = 121, BLKSIZE = 121. For the 3800, default blocking is RECFM = FBA, LRECL = 204, BLKSIZE = 204.

[,CHARS = DUMP]

specifies a dump of 64 bytes per line with the LINECNT parameter on the EXEC job control statement determining the number of lines per page. The LINECNT parameter default is 58 lines per page. If CHARS = DUMP is specified on a /*OUTPUT statement for the //PRINTER DD statement, but not on the //PRINTER DD statement itself, a dump of 32 bytes per line is printed with the DUMP character set.

[,CHARS = DUMP,FCB = STD3]

specifies a dump having 64 bytes per line and 8 lines per inch with the LINECNT parameter on the EXEC job control statement determining the number of lines per page. The LINECNT parameter default is 80 lines per page. If CHARS = DUMP,FCB = STD3 is specified on a /*OUTPUT statement for the //PRINTER DD statement, but not on the //PRINTER DD statement itself, a different dump format is obtained. This format is characterized by 32 bytes per line, 8 lines per inch, the DUMP character set and a LINECNT parameter default value of 58 lines per page.

[,FCB = STD3]

specifies a dump having 32 bytes per line and 8 lines per inch with the LINECNT parameter on the EXEC job control statement determining the number of lines per page. The LINECNT parameter default is 58 lines per page. Note that this can be obtained on the 3800 Printing Subsystem or, if your installation has defined an equivalent FCB image named STD3 for a 3211 or other impact printer having the FCB feature, on that impact printer. (FCB modules for the 3800 and 3211 FCB images are not interchangeable, but can have the same 4-character name.)

Notes:

- The CHARS = DUMP parameter is supported only when using the IBM 3800 printer.
- If the value of the LINECNT parameter exceeds the JES2 overflow line count value, undesired results may be obtained. The JES2 page overflow line count

may be overridden by use of the /*JOBPARM control statement. See the "JOB and EXEC Statements" section of this chapter.

- For more information about the /*OUTPUT and /*JOBPARM control statements, see *MVS JCL*.

[//SYSPRINT DD]

directs PRDMP messages to a sequential message data set. This statement is optional; if it is not specified, no message log is kept. The only operand required for this DD statement is:

SYSOUT = x

specifies the system output class to which messages are to be routed. The output class is identified by a single character (x). The class designation may be the same as that associated with the PRINTER DD statement. ***Note:*** Blocking is RECFM = F, LRECL = 120, BLKSIZE = 120.

[//SYSUT1 DD]

describes an optional direct access work data set from which direct reference is made to specific dump information. When the input dump data set is on magnetic tape, the use of a work data set improves the performance of the PRDMP program. If the space allocated for SYSUT1 is not large enough to contain the complete input dump from the tape, PRDMP intercepts the resulting X37 ABEND and continues processing; the dump is processed from the tape and SYSUT1. If the input dump data set is on a direct access storage device, the SYSUT1 work data set is not used.

Note: Performance improvements of as much as 15:1 can result if the space allocated for SYSUT1 is large enough to hold the complete dump.

The DD parameters required to define the direct access work data set are:

UNIT = [device-address|device = type|group-name]

tells the operating system to allocate the SYSUT1 data set on a specific type of direct access device. The subparameter may be a device name, a device type, or an esoteric group name.

SPACE = (CYL,(primary-quantity[,secondary-quantity])

specifies the amount of space to be allocated on the direct access device assigned to the SYSUT1 work set. The subparameters specify the primary and, optionally, the secondary number of cylinders to be allocated.

Note: Do *not* use the SYSUT1 DD statement and the SYSUT2 DD statement in the same step. If both are present, PRDMP ignores SYSUT1 and uses SYSUT2.

[//SYSUT2 DD]

identifies an output data set into which PRDMP transfers the input data set. Whenever the SYSUT2 DD statement is present in the input stream, PRDMP transfers the input data set into the SYSUT2 data set. PRDMP ignores any PRDMP format control statements. If the input data set is a SYS1.DUMP data set, after the contents are transferred, the SYS1.DUMP data set is made available to SVC dump to reuse.

DSNAME = dsname
required for data sets on direct access devices only.

VOL = SER = (serial = number|,serial-number|...)
specifies the volume serial number or numbers for the volumes on which the data set is to be transferred. If more than one tape is indicated, you must specify them in the order in which they were created.

UNIT = device-type
identifies a particular I/O device type.

LABEL = (|data-set-seq-#|,xx)
required for data sets on magnetic tape only. The first subparameter indicates the data set sequence number; xx indicates tape label information.

SPACE = (CYL,(primary-quantity|,secondary-quantity|,RLSE)
required for data sets on direct access devices only. It specifies the amount of space to be allocated on the direct access device assigned to the SYSUT2 data set. The subparameters specify the primary and, optionally, the secondary number of cylinders to be allocated.

RLSE
specifies that space allocated to an output data set that is not used when the data set is closed is to be released.

DISP = (NEW,KEEP)
indicates that this is a new data set (code OLD if the data set already exists) and that the data set should be kept after PRDMP processing is complete.

|//SYSTERM DD TERM = TS|
must be included in the user's JCL and allocated to a TSO terminal when the DISPLAY subverbs are entered from a TSO terminal. If input is from the master console, SYSTERM must not be used for DISPLAY subverbs.

Figure 3-3 shows a sample of the JCL that might be contained in a LOGON procedure for TSO. Example 8 assumes that this procedure is used and allocates the data sets for a print dump run.

```
//IKJACCNT EXEC PGM=IKJEFT01,DYNAMNBR=3
//SYSUDUMP DD SYSOUT=A
//SYSLBC DD DSN=SYS1.BRODCAST,DISP=SHR
//SYSUADS DD DSN=SYS1.UADS,DISP=SHR
//SYSHELP DD DSN=SYS1.HELP,DISP=SHR
//SYSEDIT DD DSN=&EDIT,UNIT=SYSDA,SPACE=(1688,(50,50))
//SYSPRINT DD SYSOUT=A
//SYSIN DD TERM=TS
//SYSTERM DD TERM=TS
```

Figure 3-3. Sample of a TSO LOGON Procedure

Cataloged Procedure Provided by PRDMP

To execute PRDMP, there is a cataloged procedure supplied in SYS1.PROCLIB with a member name of PRDMP. Figure 3-4 the procedure.

```
//PRDMP      PROC   DUMP=DUMP00
//DMP        EXEC   PGM=AMDPRDMP
//SYSPRINT   DD     SYSOUT=A
//TAPE       DD     DSN=SYS1.&DUMP,DISP=OLD
//PRINTER    DD     SYSOUT=A
//SYSUT1     DD     UNIT=SYSDA,SPACE=(4104,(1027,191))
```

Figure 3-4. PRDMP Cataloged Procedure

PROC Statement
defines the default data set number for the symbolic DSN=SYS1.&DUMP parameter of the TAPE DD statement. This value is used if DUMP= is not specified.

EXEC Statement
defines the program to be executed as AMDPRDMP. The EXEC statement that calls this procedure may include DMP.PARM and DMP.COND parameters.

SYSPRINT DD Statement
the SYSPRINT DD statement defines the data set to be used by PRDMP for informational and diagnostic messages.

TAPE DD Statement
the TAPE DD statement defines the input dump or trace data set to be processed by PRDMP. The data set name can be overridden with other data set names.

PRINTER DD Statement
the PRINTER DD statement defines the output data set.

SYSUT1 DD Statement
the SYSUT1 DD statement defines the work data set for PRDMP. The SPACE parameter is defined so that the initial space allocation is large enough to contain 4000K of storage and that each secondary extent can contain 750K of storage, thus allowing enough space to contain a possible 16000K dump. This SPACE size, if not appropriate, may be overridden.

Note: The SYSIN DD statement has been omitted.

To enter control statements in the input job stream, use:

```
//DMP.SYSIN DD *
```

and follow this statement with the appropriate PRDMP control statements. If this statement is omitted, PRDMP prompts the operator to enter control statements through the console.

How to Specify Control Statements for PRDMP

Control statements provide flexibility for you to use PRDMP. These allow you to select specific formatting options and to control the basic operation of PRDMP.

You enter the control statements into the system console or through the SYSIN data set. If no SYSIN is specified, or the supply of control statements is exhausted and an END statement is not encountered, PRDMP issues the message: AMD155D REPLY WITH GO, DESIRED FUNCTION, OR END to the console requesting that the next control statement be entered.

Several control statements, separated by commas, may be specified in one SYSIN record or in a user response to a console message. Certain control statements may appear only in the last position of a single entry. See the section "Combining PRDMP Control Statements." Because each control statement is processed as it is encountered, care must be taken by the user to supply them in the desired sequence. When all statements in one entry have been processed, a new record is read from the SYSIN data set.

PRDMP recognizes two types of control statements: PRDMP-defined control statements and user-defined control statements. The PRDMP control statements control PRDMP processing. The user control statements supplement the PRDMP processing. All control statements consist of two elements: the first element, called a verb, directs the PRDMP or user program to perform a specific function. The second element, called the operand, qualifies, where necessary, a requested function.

PRDMP Control Statements

PRDMP control statements are divided into two sets: function control statements and format control statements. Function control statements determine PRDMP program action. Format control statements determine formatting and printing factors.

Function Control Statements

The function control statements allow you to control certain operations of the PRDMP program, such as input tape handling, dump listing titles, and job termination. See Figure 3-5.

```
{CVT   } = {hhhhhh }
{C     }   {P      }
{SEGTAB} =hhhhhh
{S     }
{NEWDUMP} [{DDNAME } = {TAPE    }]
{ND     }  {DD     }   {anyname}
                [{,FILESEQ} =nn ]
                {,F       }
{NEWTAPE}
{N       }
{GO}
{G }
{ONGO}[ GRSTRACE ]
{O   } [ QCBTRACE ]
       [ Q        ]
       [,LPAMAP ]
       [,L      ]
       [{,CVT} =parm ]
       [{,C  }       ]
       [,FORMAT ]
       [,F      ]
       [{,PRINT}  parm ]
       [{,P    }       ]
       [{,EDIT} parm   ]
       [{,E   }        ]
       [,SUMMARY]
       [,CVTMAP]
       [,CPUDATA]
{TITLE} text
{T    }
{END}
{EN }
```

Figure 3-5. PRDMP Function Control Statements

CVT = { hhhhhh | P }

allows you to specify the address of the communications vector table (CVT) in the dump information. Use this if you think that the CVT pointer in virtual storage location X'4C' of the system that was dumped has been destroyed or is unavailable. If you omit this control statement, and PRDMP cannot locate the CVT at location X'4C', it cannot format most dump information. Once the CVT has been located, it remains in effect until a NEWDUMP, NEWTAPE, or another CVT control statement is encountered.

hhhhhh

is a one to six-digit hexadecimal address specifying the location of the CVT in the input dump information.

P

specifies that the location found at X'4C' in the system on which PRDMP is being executed can be used as a valid pointer to the CVT in the dumped system.

SEGTAB = hhhhhh

allows you to specify the hexadecimal real storage address of the segment table. SEGTAB is optional and need only be specified when PRDMP is unable to find a segment table for the purpose of address translation for a real storage dump. Normally this occurs only when a store status was not performed or the pointer at location X'31C' or the segment table pointed to by location X'31C' is unavailable. The hhhhhh should be replaced by the real address, in hexadecimal, of an available segment table. The SEGTAB= control statement must precede all format control statements to be useful when processing a given dump.

NEWDUMP [DDNAME = { TAPE | anyname }][,FILESEQ = nn]

defines an input data set. If you want to process more than one input data set in a single execution of PRDMP, you must supply a separate NEWDUMP or NEWTAPE control statement for each. If there is only one input data set, defined by the ddname TAPE, NEWDUMP is not needed.

When processing the NEWDUMP function, PRDMP resets all switches and/or reinitializes input data set information pertaining to the former dump. To insure the new input is recognized, all other processing should be done after the execution of the NEWDUMP function.

NEWDUMP has two keyword parameters. When the keyword parameters are omitted, PRDMP assumes that the input is a single data set with only one dump file. The parameters are described below:

DDNAME =

gives the ddname of the input data set. The ddname used in this parameter must differ from the ddnames associated with the permanent data sets used by PRDMP, such as SYSUT1, PRINTER, SYSPRINT, etc. Otherwise, unpredictable results may occur. This

parameter is not required if the TAPE DD statement describes the input data set.

FILESEQ=
identifies the sequence number of an input data set that is one of several data sets on a single magnetic tape volume. If this parameter is omitted, PRDMP assumes a default value of FILESEQ=1.

NEWTAPE
has the same function as the NEWDUMP statement with parameters specified as DDNAME=TAPE and FILESEQ=1.

GO
allows the user to request a group of pre-specified format control statements making it unnecessary to separately specify each statement each time. Unless you specify differently by use of the ONGO verb, the GO verb causes PRDMP to format the dump information as if you specified the format control statements: EDIT, SUMMARY, and PRINT CURRENT. If you use a GO verb, it must appear last in any control statement or console response. See Example 1 and Example 4.

Note: The ONGO-GO combination is not required for PRDMP execution. You need not specify ONGO unless you want to change the latest predefined set of PRDMP verbs. Each PRDMP control statement may be specified directly at any time.

ONGO [QCBTRACE or GRSTRACE][,LPAMAP][,CVT=parm][,FORMAT][,PRINT parm] [,EDIT parm][,SUMMARY][,CVTMAP][,CPUDATA]
allows you to change the pre-specified set of control statements associated with the GO verb. The new set of control statements is executed each time a subsequent GO control verb is encountered. The initial and default ONGO operand consists of the control statements: EDIT, SUMMARY, and PRINT CURRENT. The eligible ONGO parameters are: CVT=parm, QCBTRACE or GRSTRACE, LPAMAP, FORMAT, SUMMARY, CVTMAP, CPUDATA, PRINT parm, ASMDATA, SRMDATA, SUMDUMP, CPUDATA, LOGDATA, UTAMMAP, and EDIT parm.

The PRINT and EDIT parm control statements are in the restricted category and cannot be combined. Therefore, on the ONGO verb, if you specify any parameters on EDIT, do not specify PRINT.
(Note that the rules for combining control statement verbs appear later in this chapter in Figure 3-9).

The ONGO verb without any operand restores the original GO parameters: EDIT, SUMMARY, and PRINT CURRENT. See Example 4.

The data following ONGO (or O) through column 71 is set as the predefined verbs.

The control statements specified on ONGO cannot exceed 66 characters and on O cannot exceed 69 characters.

Note: The ONGO and GO control statements work together. That is, to execute the group of statements set by the ONGO, a GO must follow IBID.

TITLE text

allows you to print a title at the top of each page of the dump listing. If T is specified in the PARM = field of the EXEC statement, the title information can be entered as a reply to message AMD154D REPLY TITLE, SAME, or END. If you enter the control statements from the console, title information is entered as a reply to message AMD155D REPLY GO, DESIRED FUNCTION, or END. If you specify PARM = T on the START command and you enter the control statements from the console, you should enter title information as a reply to message AMD154D; you enter other control information as a reply to message AMD155D. PRDMP considers the first sixty-four (64) characters after the TITLE verb in the control statement or user's console response as the text of the dump listing title. No control statement may appear after the TITLE verb in the same entry. Note that this title is provided in addition to the title provided by the dumping program and that more than one TITLE verb is acceptable in a job. If the TITLE verb is not entered, the title from the dump is printed on each page.

END

allows you to terminate PRDMP processing. When you enter the END statement either via a control statement or in response to a console message, PRDMP closes all data sets and returns control to the system control program. Any PRDMP control statements encountered after an END statement are ignored.

Format Control Statements

Format control statements allow you to choose particular parts of the input to be formatted and printed. These statements can cause AMDPRDMP to generate maps or traces of certain control blocks in the dumped information as well as selectively formatting only those areas of storage in which you are interested. All formatting operations (with the exception of PRINT REAL) are performed using virtual addresses. The extent of formatting and printing that PRDMP can perform is, of course, a function of the input data with which it works. See Figure 3-6.

```
{GRSTRACE}
{QCBTRACE}
{Q       }
{LPAMAP  }
{L       }
{FORMAT  }
{F       }
 ASMDATA
 SRMDATA
 SUMMARY
 SUMDUMP
 CPUDATA
 CVTMAP
 LOGDATA
 VTAMMAP
 JES3 [ASID=nnnn|JOBNAME=jjjjjjjj]
      [,DEBUG]
 TCAMMAP
{PRINT} [CURRENT]
{P    } [C      ]
        [,NUCLEUS ]
        [,N       ]
        [,SQA]
        [,CSA]
        [,STORAGE ] [=ASID  [(asid)                    ]]
        [,S       ]        [(asid:asid)                ]
                           [(asid,asid[,...,asid])     ]
                           [(asid[,NONUC][,NOCOM])     ]
                    [=(addresses)]
        [,REAL  [=(addresses)]  ]
        [,R                     ]
        [,JOBNAME   =(jobnames) ]
        [,J                     ]
{DISPLAY} (see note)
{D      }
   {LIST }{  address.[+offset]      }
   {L    }{  symbolic name[+offset] }
         [L(length)]
         [ASID(n)]
   {EQUATE } symbolic-name address
   {EQ     }
   {HARDCOPY }  [ON          ]
   {HC       }  [OFF         ]
                [user comment]
    C    user comment
   {RETURN }
   {RET    }
{EDIT } parm
{E    }
```

Note: A DISPLAY control statement is required to enter DISPLAY mode. The DISPLAY options (LIST, EQUATE, HARDCOPY, C, or RETURN) may only be used while operating in DISPLAY mode. To terminate the DISPLAY mode, use the RETURN option. Only one option may be entered per TSO transaction. All DISPLAY option parameters must be separated by at least one blank but may contain no embedded blanks.

Figure 3-6. PRDMP Format Control Statements

QRSTRACE or QCBTRACE

causes AMDPRDMP to print the address of the major control blocks associated with global resource serialization and the contents of control blocks on the global resources queue in the input data set. There is no parameter associated with this option. The following information is printed in this order:

- GVT (global vector table) address
- GVTX (global vector table extension) address
- GHT (global hash table) address
- LHT (local has table) address
- GRPT (global resource pool table) address
- LRPT (local resource pool table) address
- the local queue control block data
  - Major name
  - Minor name
  - QCB (queue control block) address
  - QEL (queue element) address
  - TCB (task control block) address
  - ECB/SVRB (event control block/supervisor request block) address
  - scope
  - status
  - SYSNAME
  - ASID (address space identifier)
- the global queue data
  - Major name
  - Minor name
  - ACB address
  - QEL address
  - TCB address
  - ECB/SVRB address
  - scope
  - SYSNAME
  - ASID

Note that if the global resources serialization address space is not dumped, only the GVT and GVTX address is printed, and the following messages appear:

- GHT DATA NOT AVAILABLE
- LHT DATA NOT AVAILABLE
- GRPT DATA NOT AVAILABLE
- LRPT DATA NOT AVAILABLE

LPAMAP

causes PRDMP to format and list a map of the contents (name, entry point, length, and starting address) of those link pack area modules that were on the link pack area active queue at the time the dump was written. There is no operand associated with this verb. If the input data set does not contain this area, LPAMAP causes an error message to be printed.

Note that this control statement maps only the active modules in the reenterable load module area or the link pack area; it does not include any

storage associated with the area. If you want a map of storage, you must use the PRINT STORAGE= control statement. See Example 3.

FORMAT

causes PRDMP to format and print the major system control blocks and data associated with each address space in the dumped system. Information and control blocks that are formatted and printed (if available) are:

- global SRBs
- ASCBs
- local SRBs
- ASXBs
- real address of the segment table
- LT
- ET
- AT
- TCBs
- RBs
- XSBs
- STKEs
- virtual storage information
- load list information
- job pack queue
- DEBs
- task I/O information
- DCBs
- IOBs/ICBs/LCBs
- EXCPD
- UCBs
- RTCT
- RTM2WAs
- EEDs
- SCBs
- FRRs
- IHSA

User exit capabilities are available at the system, address space, and task levels. If you want to limit the formatting of system control blocks to specific address spaces, you should specify the proper parameters in the PRINT control statement instead of using FORMAT. There is no parameter associated with the FORMAT statement.

Note: Only active TCBs are formatted. Check the NTC and LTC pointers in the TCB summary for the addresses of completed TCBs.

Only the current task for the processor from which the dump was taken will be flagged on a multiprocessing system.

The following control blocks are only formatted for failing tasks: DCBs, EXCPD, IOBs/ICBs/LCBs, UCBs, and SCBs.

The MVS/370 Debugging Handbook and MVS Diagnostic Techniques describe the conditions under which the RTM control blocks, such as the RTCT, are formatted.

ASMDATA

causes PRDMP to format and print the contents of the following auxiliary storage management (ASM) control blocks in the input data set:

- ASM vector table (ASMVT)
- Logical group vector table (LGVT)
- Auxiliary storage page correspondence table (ASPCT)
- ASM control element (ACE)
- Task mode element (TME)
- Paging activity reference table (PART)
- Paging activity reference table entry (PARTE)
- I/O request element (IOE)
- Page allocation table (PAT)
- Performance characteristics table (PCT).

There is no operand associated with the ASMDATA verb.

SRMDATA
causes PRDMP to format and print the following SRM (System Resources Manager) control blocks:

- OUCBs (SRM User Control Block) on the WAIT queue
- OUCBs on the OUT queue
- OUCBs on the IN queue
- DMDT (Domain Descriptor Table)

There is no operand associated with the SRMDATA verb.

Note: If an SVC dump generated the input data set, valid queues might appear invalid because the queues can change while the SVC dump is running.

SUMMARY
causes PRDMP to print a summary of the dump input file contents, a system summary, and a problem summary. These summaries are provided to limit the amount of output provided by AMDPRDMP, to provide gross address space relationships which serve as a starting point for problem debugging, and to establish a base from which decisions concerning further formatting can be made. There is no operand associated with the SUMMARY verb.

SUMDUMP
causes PRDMP to locate and print the summary dump data provided by SVC dump. There is no operand associated with the SUMDUMP verb. See the SDUMP macro instruction in *OS/VS2 SPL: Supervisor* for information on the summary dump function The *MVS/370 Debugging Handbook* and *MVS Diagnostic Techniques* also detail the content of the summary dump and examples of the output. See Appendix C, PRDMP User Exit Facility, for more information on formatting summary dump records.

CPUDATA
allows you to request processor-related information which is helpful in debugging multi-processing problems. The information includes all registers for each processor; the current PSW for each processor, unprefixed storage 0-4K, the common system data area, the PSA's, and logical physical CCAs for each PSA. There is no operand associated with the CPUDATA verb. See Example 2.

Note: If input is a SADMP high-speed data set, and the STORE STATUS operation was not performed immediately prior to IPLing SADMP, the STORE STATUS information for the IPLed processor is unpredictable except for the general-purpose registers, which are always valid for the IPLed processor.

CVTMAP
allows you to format the communications vector table (CVT). Fields in the CVT are labelled in the printout. There is no parameter associated with the CVTMAP statement.

LOGDATA

causes PRDMP to format the in-storage LOGREC buffer records in the same manner as EREP formats the SYS1.LOGREC records. There is no operand associated with the LOGDATA verb.

The summary of records found, located at the end of the LOGDATA output, indicates the number and type of records found in the in-storage LOGREC buffer. Not all record types are formatted, so the number of records listed in the summary may not equal the number of records formatted.

VTAMMAP

allows you to request selected VTAM control blocks that are helpful in VTAM problem determination. When VTAMMAP is specified, the VTAM formatted dump routine locates and formats selected VTAM control blocks in dumps produced by the AMDSADMP and SVC dump programs. For a description of the VTAM formatted dump routine and the control blocks selected, see *ACF/VTAM Version 2, Diagnosis Guide*.

```
JES3      [ASID=nnnn|JOBNAME=jjjjjjjj]
            [,DEBUG]
```

JES3

causes formatting of the contents of specific JES3 control blocks. When you code the JES3 control statement and no parameters or you specify the DEBUG parameter alone, PRDMP formats the contents of the JES3 control blocks for the JES3 address space. If you do not code the JES3 control statement at all, PRDMP does not format any JES3 control blocks.

ASID = nnnn

causes the formatting of the contents of JES3 control blocks for the specified address space. The address space identifier is a 1- to 4-digit hexadecimal number. You can format control blocks in either the JES3 address space or an FSS address space using this parameter. Only one ASID can be specified. When you specify ASID, you may specify only DEBUG as a second parameter.

DEBUG

causes a snapshot dump to be written to a JES3SNAP data set if a program check occurs during the formatting of JES3 control block contents. The dump contains register contents, storage contents of the failing module, and the contents of the CSECT (ABNCSECT) workarea at the time the dump was taken. Register 11 contains the address of the exit routine (IATAPBR) that was formatting the control blocks. Register 13 contains the address of the format workarea.

Note: You can code DEBUG alone.

JOBNAME = jjjjjjj

causes the formatting of the contents of JES3 control blocks for an address space associated with the specified job. The jobname is 1 to 8 alpha- numeric characters. Only one jobname can be specified on a single JES3 control statement. When you specify JOBNAME, you may specify only DEBUG as a second parameter. You can format control blocks in either the JES3 address space or an FSS address space using this parameter.

Note: The jobname that you specify is the ID for the FSS once it is started. This is also its FSSNAME.

Note: You must code multiple JES3 control statements when control blocks from the JES3 address space and an FSS address space are to be formatted from the same dump.

NOTES: Two parameters provide you with the compatibility of previous versions of the JES3 dump exit. They are ON and OFF. These parameters allow you to continue to use canned PRDMP procedures that must reference the parameters on the JES3 control statement.

The ON parameter causes the formatting of the contents of the JES3 control blocks for the JES3 address space. The only parameter that can be specified with ON is DEBUG.

The OFF parameter suppresses all JES3 dump exit processing. It must be specified alone.

TCAMMAP

allows you to request selected TCAM control blocks that are helpful in TCAM problem determination. When TCAMMAP is specified, the TCAM formatted dump routine locates and formats selected TCAM control blocks in dumps produced by the AMDSADMP and SVC dump programs. For a description of the TCAM formatted dump routine and the control blocks selected, see *ACF/TCAM Version 2, Diagnosis Guide*.

```
PRINT
     [CURRENT][,NUCLEUS][,SQA][,CSA]
          [,STORAGE[=ASID(asid[:asid])] [=(addresses)]]
          (asid-list)
     [,REAL(=addresses)][,JOBNAME=(jobnames)]
```

PRINT

allows you to select the particular areas of virtual or real storage that you wish to print. When a PRINT control statement is used, it must appear last in the SYSIN data set or user console response. The PRINT control statement has several parameters. Multiple parameters may be specified in a single specification of a PRINT control statement. The first parameter in the parameter string must be separated from the PRINT verb by at least one space. Subsequent parameters are separated from previous parameters by a comma with no intervening spaces.

Note: PRDMP cannot print virtual storage pages which were not in real storage or in a page data set when the dump was taken. Within each specified or implied virtual address range, only those pages which were in real storage at dump time are printed. This limitation on PRDMP is due to SADMP output being restricted to the contents of real storage and specified page data sets.

CURRENT

causes the PRDMP program to print those areas of the dumped system's virtual storage that are associated with the address spaces that were current at the time of the dump. PRINT CURRENT attempts to process the four cross-memory address spaces (primary, secondary, home and cross-memory lock). If any of the four address spaces have the same ASID, that address space is printed only once. If the cross-memory address spaces are not in the dump or if AMDPRDMP cannot identify them, AMDPRDMP determines the address space that was current on the processor when the dump was taken and processes it. The registers, PSW, and the following major control blocks associated with these address spaces are formatted (if available):

- Task Control Block (TCB)
- Request Blocks (RBs)
- Problem Program Boundaries
- Load List
- Job Pack Queue
- Data Extent Block (DEB)
- Task Input/Output Table (TIOT)
- Data Control Block (DCB)
- I/O Control Block (IOB), or Interrupt Control Block for chained scheduling under SAM or BPAM (ICB), or Teleprocessing Line Control Block (LCB)
- EXCP Debugging Area (EXCPD - XDBA)
- Unit Control Block (UCB)
- Recovery Termination Control Table (RTCT)
- RTM2 Work Area (RTM2WA)
- Extended Error Descriptor Block (EED)
- STAE Control Block (SCB)
- Current FRR Stack (FRRS)
- Interrupt Handler Save Area (IHSA)
- The real address of the segment table
- Linkage tables (LT)
- Entry tables (ET)
- Authorization tables (AT)
- Cross Memory status blocks (XSB)
- PCLINK stack elements (STKE)
- The address space second table entry (ASTE)

If the input is a dump created by SVC dump, PRDMP formats the address spaces that were dumped and provides the SVC dump buffer in hexadecimal if the buffer was requested at the time the dump was taken. A format user exit capability is available with the PRINT CURRENT operation.

In a multiprocessing system, only the current task for the processor from which the dump was taken is formatted.

The following control blocks are only formatted for the failing task: DCBs, IOBs/ICBs/LCBs, EXCPD, UCBs, and SCBs.

NUCLEUS
causes PRDMP to print the nucleus portion of the dumped operating system. A format user exit capability is available with the PRINT NUCLEUS statement.

SQA
causes the PRDMP program to print out in hexadecimal the system queue area (SQA) portion of the dumped operating system.

CSA
causes PRDMP to print out in hexadecimal the common service area (CSA) portion of the dumped operating system.

STORAGE[=ASID **[(asid)** **(asid:asid)** **(asid,asid[,...,asid]** **(asid[,NONUC][,NOCOM])]** **][=(addresses)]**
allows you to supply the beginning and ending virtual addresses of the areas that are to be printed. This can be qualified to an address space or group of address spaces by coding a single ASID, a list of single ASIDs separated by commas (asid,asid), a range of ASIDs (asid:asid), or not specifying an asid value which causes PRDMP to default to storage for the address space that was current at the time the dump was taken. The FFFD and FFFE ASIDs are restricted AMDSADMP ASIDs and must be printed using separate print storage statements (for example, P S=ASID(FFFD) and P S=ASID(FFFE)).

The NONUC and NOCOM keywords are optional keywords used in conjunction with the ASID keyword. When used, the two fields must be separated by a comma and must be within parentheses. Either or both keywords may be specified. The NONUC option suppresses printing of the nucleus for this input. The NOCOM option suppresses the printing of the common data area for this input. At least one ASID must appear within the parentheses.

Specify the value for asid in hexadecimal.

The STORAGE parameter with no subparameters causes AMDPRDMP to attempt to process the four cross-memory address spaces (home, primary, secondary, and cross-memory lock). If any of the four cross-memory address spaces has the same ASID as one of the other three, the address space is only printed once. If the dumped data set does not contain the cross-memory address spaces or AMDPRDMP is unable to identify them, AMDPRDMP determines the address space that was current on the processor when the dump was taken and processes it.

When you specify a range of ASIDs, the second ASID value must be greater than or equal to the first ASID value. ASID values of 0 or hexadecimal FFFF are considered invalid ASIDs. When specifying a list of ranges for ASIDs, note that if at least one ASID value is not active, a message

indicates that some ASIDs could not be printed. But, the ones found are printed.

You code beginning and ending virtual addresses, expressed in hexadecimal, in pairs. All address pairs appearing after the STORAGE keyword must be enclosed in parentheses. PRDMP prints the contents of the virtual address space bounded by an address pair. Each specified address is separated from the preceding address by a comma with no intervening spaces. If only a single address is supplied, all of virtual storage for that address space starting at that virtual address and continuing to the upper end of the address space is printed. If you specify a list or range of ASID's and the range of addresses to be printed includes some portions of the nucleus, only the copy of it is printed. If no addresses are specified, the PSW and registers are printed. Examples of correct use of PRINT STORAGE:

```
PRINT STORAGE=ASID(1:20)
PRINT STORAGE=ASID(2,4)=(30000,D0000)
PRINT STORAGE=(4F0000)
PRINT STORAGE=ASID(3,NONUC)
PRINT STORAGE=ASID(1,NOCOM,NONUC)
```

REAL[=(addresses)]

allows you to supply the beginning and ending real storage addresses of the areas that are to be printed. If you specify the REAL= parameter and the input file contains a virtual storage dump, your request for print operation is denied. If you specify PRINT REAL with no addresses, all real storage contents available on the input file are printed. Beginning and ending real storage addresses, expressed in hexadecimal, are coded in pairs. All address pairs must be enclosed in parentheses. The contents of the real storage bounded by an address pair are printed. Each specified address is separated from the preceding address by a comma with no intervening spaces. If only a single address is specified, all real storage contents available on the input file starting at that real storage address are printed. If no addresses are specified, the PSW and registers are printed.

JOBNAME=(jobnames)

allows you to limit the scope of the dump output to the major control blocks and storage associated with the jobs specified in the parameter value field. Jobnames are enclosed in parentheses and delimited by commas. A maximum of ten jobnames may be specified. A format user exit capability is available with the PRINT JOBNAME operation.

DISPLAY

causes PRDMP to enter display mode for interactive use of the DISPLAY subverbs. Through these subverbs, the user may quickly and conversationally scan any dump location and display up to 256 bytes per transaction. See Figure 3-6 for a summary of the DISPLAY subverb operands. When DISPLAY is entered from the system console, output is directed to the system console. When DISPLAY is entered from a TSO terminal, output is directed to the SYSTERM data set. Multiple DISPLAY subverbs are not permitted on one entry. For example, LIST 2AC4 followed by LIST B123 on the same line would not be allowed. *Note:* The parameters for each subverb are separated by at least one blank.

LIST { **address.**[**±offset**] / **symbolic name**[**±offset**] } [**L(length)**][**ASID(n)**]

displays up to 256 bytes of virtual storage. The parameter, address., specifies a one- to six-digit hexadecimal address which identifies the first byte of the display area. A period must be coded after the hexadecimal address to distinguish it from a symbolic address. The symbolic name specifies the relative address of the first byte of the display area. If this parameter is used, it must be preceded by an EQUATE subverb which equates the symbolic name to a virtual address. ±offset specifies a one- to six-digit hexadecimal displacement which when summed with the address supplied by address, or symbolic name yields a one- to six-digit hexadecimal address which has a minimum of 0 and a maximum of FFFFFF. L(length) supplies the length in bytes of the area to be displayed. It has a maximum value of 256 and the default value is 4. If address + length exceeds FFFFFF, the address of the last byte to be displayed is set to FFFFFF. ASID(n) is the address space ID value of the address space to be displayed. The default is the last ASID specified on a previous LIST subverb, or if no ASID was previously specified, the ASID of the address space that was executing at the time of the dump.

EQUATE symbolic-name address

equates a virtual address to a symbolic name. The symbolic name is a one- to eight-character alphameric label. The address is a one- to six-digit hexadecimal address. The maximum number of EQUATEs allowed per display session is 256. A symbolic name must be equated to an address before that name can be used by a LIST subverb as a symbolic name.

HARDCOPY { **ON**|**OFF** / **,user comment** }

requests printing of a hard-copy backup of the displayed storage. The operands are mutually exclusive. ON requests that PRDMP print a hard-copy backup of all requested storage displays. ON remains in effect until HARDCOPY OFF or RETURN is entered. OFF specifies that hard-copy backup of displayed storage not be printed. The HARDCOPY option with a user comment causes PRDMP to print the user's comments and the storage displayed by the previous LIST subverb. The comment may not extend beyond column 80, and continuation statements are not allowed. If no operands are specified, PRDMP prints a backup copy of the storage displayed by the last LIST subverb.

C user comment

requests that hard-copy of the user's comments be printed. The comment may not extend beyond column 80, and continuation statements are not allowed.

RETURN

terminates the display mode and returns control to PRDMP.

EDIT parm

causes PRDMP to attempt to obtain the trace data either from a defined GTF trace data set or from the storage dump currently being processed. See Figure 3-7 for a summary of the EDIT control statement keywords. See also Example 5 and Example 6.

```
{EDIT}  [EXIT=pgmname]
{E   }
[{,DDNAME} =ddname ]
 {,DD    }
 [,START=(ddd,hh.mm.ss)]
 [,STOP=(ddd,hh.mm.ss)]
[{,JOBNAME} =(jobname1[,jobname2]...[,jobname5])]
 {,J      }
 [,ASCB=(address1[,address2]...[,address5])]
 [,EOF]
 [,SYS]
[{,IO    } =(cuu1[,cuu2]...[,cuu50])]
 {,IO=SIO}
 {,SIO=IO}
 {,SIO   }
 [,IO]
 [,SIO]
 {,SVC                              }
 {,SVC=(svcnum1[,svcnum2]...[,svcnum256])}
 {,PI                               }
 {,PI=(code1[,code2]...[,code18][,code19])}
 [,EXT]
 [,DSP]
 [,RR]
 [,RNIO]
 [,SLIP]
 [,SRM]
[,USR= (symbol1[,symbol2]...[,symbol20])  ]
       (idvalue1[,idvalue2]...[,idvalue20])
       (idrange1[,idrange2]...[,idrange20])
       ALL
[,CCW   ]
[,CCW=SI]
[,CCW=S ]
[,CCW=I ]
 [,CPU=n]
```

Figure 3-7. Format of the EDIT Control Statement Showing All Valid Keywords

[EXIT = pgmname]

defines the program name of a user-written exit routine that should be given control to inspect all trace data records. The user-written exit routine is also given control when the logical EOF of the trace data has been reached. If the routine does not exist or cannot be successfully loaded, EDIT execution will terminate and the next control statement is read.

DDNAME = ddname

specifies the name of the DD statement that defines a trace data set to be edited. DDNAME is required for GTF trace data sets; do *not* use it for the storage dump that is being processed.

[START = (ddd,hh.mm.ss)]
[STOP = (ddd,hh.mm.ss)]

specifies that blocks between these times are to be edited; only those records requested by event-oriented keywords will actually be formatted. If no START time is specified, the edit will start at the beginning of the data set. If no STOP time is specified, EDIT will terminate upon reading the end of the data set. 'ddd' is Julian day, and 'hh.mm.ss' is the hours, minutes and seconds as set in the TOD clock.

JOBNAME = (jobname1[,jobname2]...[,jobname5])

allows specification of up to five 8-character jobnames for which trace entries and user records should be formatted. If all jobnames cannot fit on one line, the continuation technique described below under IO should be used. Specifying JOBNAME for editing of minimal trace records (SYSM) is an error condition. If PRDMP control statements are entered from the console, the operator is asked if editing is to continue. If control statements are being entered from SYSIN, EDIT execution is terminated.

ASCB = (address1[,address2]...[,address5])

allows specification of up to five ASCB addresses for which trace entries and user records should be formatted. The ASCB addresses are specified as one- to eight-digit hexadecimal addresses. If all ASCB addresses cannot fit on one line, the continuation technique described below under IO should be used.

EOF

specifies that the exit routine specified by the EXIT = keyword, is to receive control on all EDIT terminating conditions.

SYS

requests that nine system event trace records be formatted (that is, SVC, SIO, IO, PI, EXT, DSP, RNIO, RR and SRM). SYS is the default option when only keywords from the following group are specified:

```
DDNAME=
EXIT=
START=
STOP=
JOBNAME=
ASCB=
EOF
```

If any other keywords are specified, the default is overridden.

{IO / SIO / IO=SIO / SIO=IO} = (cuu1[,cuu2]...[,cuu50])

defines up to fifty different devices for which SIO trace records, IO trace, or both is formatted. Only trace records associated with the specified devices are formatted; all others are ignored. 'cuu' is the three-digit device address. If the specification of this keyword exceeds one line, the first line should be closed with a right parenthesis, followed by a comma, and the keyword respecified with the additional device addresses on a subsequent statement or reply.

Example
```
EDIT DD=SYSTRACE,IO=(191,192,193),
   IO=(283,284,285)
```

IO
SIO

request all IO and SIO trace records to be formatted. If both IO and IO were specified, all IO trace records are formatted. The same is true of SIO and SIO.

SVC
SVC=(svcnum1[,svcnum2]...[,svcnum256])

requests that all SVC trace records be formatted. SVC= selects only those records associated with the specified SVC numbers to be formatted. 'svcnum' is a one- to three-digit decimal SVC number from 0 to 255. If both SVC and SVC= are specified, all SVC trace records are formatted. The same continuation technique should be used for SVC= as described in IO=.

PI
PI=(code1[,code2]...[,code18][,code19])

requests that all program interrupt trace records be formatted. PI= requests that only trace records associated with specific interrupt codes be formatted. 'Code' is a one- to two-digit decimal interrupt code from 1 to 19. Code 16 specifies that trace records for a segment exception should be formatted, while code 17 is for page fault trace records and code 18 is for translation exception records. Code 19 specifies that SSM interrupts should be traced. If both PI and PI= are specified, all PI trace records are formatted. The same continuation technique should be used for PI= as described under IO=.

EXT

requests that all external interrupts trace records be formatted.

DSP

requests that all dispatching event trace records be formatted.

RR

requests that all recovery routine event records be formatted.

RNIO

requests that all VTAM remote network activities be formatted.

SLIP

requests that all SLIP trace records be formatted.

SRM

requests that all system resource manager event records be formatted.

USR= (symbol1[,symbol2]...[,symbol20)]
(idvalue1[,idvalue2]...[,idvalue20)]
(idrange1[,idrange2]...[,idrange20)]
ALL

specifies which user/subsystem trace records should be formatted. These are all records created by use of the GTF GTRACE macro. Up to twenty ID values or ranges or symbols may be specified, to denote all entries belonging to one component or subsystem. The GTRACE data consists of user event trace records and/or IBM subsystem event records. The subsystems are OPEN/CLOSE/EOV, SAM/PAM,DAM,VTAM, and VSAM. The EIDs and symbolic names associated with these subsystems are in Figure A-5. You may indicate up to 20 symbols with USR= keyword. For user trace records, you may indicate up to 20 ID values or ranges. User trace records can also be filtered by use of the jobname and ASCB keywords.

The 'symbol' to be used is defined by component when ID values are assigned to the subsystem.

The 'idvalue' is the three-digit hexadecimal ID which is user specified in the GTRACE macro.

The 'idrange' is the first and last ID values of range separated by a hyphen (for example, USR=(010-040,BFD-BFF).

'ALL' requests that all user and subsystem trace records be formatted. If USR=ALL is specified in addition to selected ID values, ranges or symbols, all user and subsystem trace entries are formatted. The same continuation techniques should be used for USR= as described under IO=; (for example, an ID range should be completed, and followed by a right parenthesis and a comma; the USR= keyword should then be respecified on a subsequent statement or reply).

Valid symbols and their corresponding ids are:

SYMBOL	IDs	FORMATS
DMA1	FFF	OPEN/CLOSE/EOV
AM01	FF5	VSAM
SPD1	FF3	SAM/PAM/DAM
SPD2	FF4	SAM/PAM/DAM
SPD3	FF6	SAM/PAM/DAM
SPD4	FF7	SAM/PAM/DAM
SPD5	FF8	SAM/PAM/DAM
SPD6	FF9	SAM/PAM/DAM
SPD7	FFA	SAM/PAM/DAM
SPD8	FFB	SAM/PAM/DAM
SPD9	FFC	SAM/PAM/DAM
SPDA	FFD	SAM/PAM/DAM
SPDB	FFE	SAM/PAM/DAM
TP10	FEF	ACF/VTAM buffer contents trace (USER)
CL01	FF1	ACF/VTAM buffer contents trace (VTAM)
CL02	FF0	ACF/VTAM SMS (buffer use) trace
LINE	FF2	ACF/VTAM NCP line or TG trace
INT1	FE1	ACF/VTAM internal table
APTH	FE2	TSO/VTAM TGET/TPUT trace
APTR	FE3	VTAM reserved
APTD	FE4	ACF/VTAM NCP line type trace

Figure 3-8. Example of the valid symbols with the associated subsystems

CCW
CCW=SI
CCW=S
CCW=I

requests formatting of channel program trace records. If you specify CCW more than once, PRDMP uses your first specification of CCW. For PRDMP to format CCW trace records, PRDMP must format SIO base records or IO base records or both.

CCW
CCW=SI

requests formatting of all CCW, SIO, I/O, and resumption of I/O trace records in the specified data set. PRDMP turns on formatting for SIO and IO base records even if you do not specify the SIO and IO keywords. When you specify the SIO and IO keywords, and either CCW or CCW=SI, PRDMP formats CCW trace records for events on the devices the SIO and IO keywords identify.

CCW=S

requests formatting of CCW trace records for SIO and resume I/O operations. PRDMP turns on formatting for SIO base records even if you do not specify the SIO keyword. When you specify the SIO keyword and CCW=S, PRDMP formats CCW trace records for events on the devices the SIO keyword identifies.

CCW=I

requests formatting of CCW trace records for I/O events, and, if present, PCI events. PRDMP turns on formatting for IO base records even if you do not specify the IO keyword. When you specify the IO keyword and CCW=I, PRDMP formats CCW trace records for events on the devices the IO keyword identifies.

CPU=n

requests formatting for events occurring on the CPU whose physical identifier is 'n'. The value 'n' is a hexadecimal number from 0 to F. The CPU=n keyword is only effective if you specify the CCW, SIO, or IO keywords. The CPU identifier does not filter any other events.

Defaults for EDIT Keywords

Unless you specify the options illustrated in Figure 3-7, the default options are as follows:

```
SIO, IO, SVC PI, EXT, DSP, RR, SRM, SLIP, RNIO, and CCW
```

Editing Trace from a Trace Data Set

If the DDNAME= keyword is specified, other specified options will take effect as shown in Figure 3-7. Those keywords that appear within the same vertical columns are mutually exclusive and if two or more options within a column are specified, the option that is highest within the column takes effect. For example, if both SVC and SVC= are specified, SVC (indicating ALL) will take effect.

In like manner, if IO=SIO= is specified during the same EDIT execution that IO and SIO are specified, IO and SIO will take effect. However, if in the above case, IO is specified with IO=SIO= but SIO is not specified, those devices indicated in IO=SIO= will remain in effect for SIO=. The converse is true also. For example:

```
EDIT DD=DD1,IO=SIO=(284,285),SIO
```

The above EDIT control statement causes I/O records for devices 284 and 285, and all SIO records, to be formatted. If either START= or STOP= is specified more than once, the first specification takes effect. After all optional input has been processed, a message is written to the SYSPRINT device and/or the console (depending upon where the options were entered: console or control statements) identifying the EDIT options in effect. If SYS has been specified, each system event is individually indicated.

Combining PRDMP Control Statements

If you are controlling PRDMP operation from the system console, you may want to save time by combining control statements in a single reply to a prompting message. This section describes the rules for combining control statements.

PRDMP control statements fall into two categories: restricted and free. The names of the categories refer to the way the control statements in them can be combined with each other. Figure 3-9 shows the categories and the control statements they contain.

FREE		RESTRICTED
ASMDATA	LOGDATA	**DISPLAY
CPUDATA	LPAMAP	*EDIT parm
CVTMAP	NEWTAPE	END
CVT=parm	QCBTRACE or GRSTRACE	GO
*EDIT	SEGTAB=parm	*JES3 parm
FORMAT	SRMDATA	
*JES3	SUMDUMP	NEWDUMP
	SUMMARY	ONGO
	TCAMMAP	PRINT
	VTAMMAP	TITLE

***Note:** A control statement coded with no parameters may fall into the FREE category, while that same control statement with parameters is RESTRICTED.

****Note:** These rules also apply if you are invoking PRDMP via JCL and you want to combine control statements on a single statement in the input stream. The only exception is the DISPLAY control statement. It may not be combined with any other control statements. It cannot be used in the JCL when using the master console, only when running PRDMP via TSO.

Figure 3-9. Combining Free and Restricted Control Statement Verbs

When you reply to a prompting message, the rules for combining control statements in the two categories are:

- You can freely combine control statements in the FREE category with each other.
- You can never combine control statements in the RESTRICTED category with each other.
- You can combine any number of control statements from the FREE category with one control statement from the RESTRICTED category, provided that the control statement from the RESTRICTED category comes last in the reply.

Here are some examples of control statements combined correctly:

```
LPAMAP,EDIT,PRINT NUCLEUS
FORMAT,GRSTRACE,EDIT DDNAME=TRACE,SVC,SIO=IO=ALL,PI
NEWTAPE,CVT=parm,LPAMAP,GO
NEWTAPE,TITLE
LPAMAP,END
SEGTAB=parm,FORMAT,PRINT CURRENT
```

User Control Statements

You can define special control statements to supplement PRDMP processing. The PRDMP user exit facility allows user-written formatting routines to gain control when user control statements are in the input stream. The syntax of these statements is:

```
user control statement      {parameters}
```

The control statement must be one to eight characters in length. You must also include it in the exit control table, AMDPRECT, in order for the appropriate user-written program to execute. See "Appendix C: PRDMP User Exit Facility."

The optional parameters provide flexibility and control of formatting by your routine.

Correlating PRDMP Output with Other Diagnostic Data

When the system encounters a software error or a machine failure that the hardware retry routines cannot correct, it assigns a unique identifier to the error. If an SVC dump is taken, this error identifier (errorid) appears on the SVC dump title page and in the console message that indicates the SVC dump was taken. The same error identifier appears in the SYS1.LOGREC records created as a result of the error. The existence of the same error identifier in these separate sources of diagnostic data indicates that they all apply to the same error condition.

The error identifier in the SVC dump has the following form:

ERROR FOR THIS DUMP=SEQxxxxx CPUyy ASIDzzzz TIMEhh.mm.ss.t

where:

xxxxx	sequence number
yy	CPU identifier
zzzz	address space identifier
hh.mm.ss.t	time stamp(hours,minutes,seconds,tenths of seconds)

If an error identifier is not available for an SVC dump, the message

NO ERRORID ASSOCIATED WITH THIS DUMP

appears where the error identifier would otherwise appear.

Note: MVS Diagnostic Techniques shows examples of dumps formatted by PRDMP and gives advice on when to use the PRDMP control statements.

JCL and Control Statement Examples

The following examples illustrate some of the functions that PRDMP can perform.

Example 1: Using the Cataloged Procedure

IBM supplies a cataloged procedure, called PRDMP, that defines the input and output data sets and a work data set for PRDMP. This example shows how to use the cataloged procedure.

```
//PROCDMP      JOB    MSGLEVEL=(1,1)
//STEP1        EXEC   PROC=PRDMP,PARM.DMP=T
//DMP.SYSIN    DD     *
    GO
    END
//*
```

In this example:

PARM.DMP=T
issues message AMD145D to the operator requesting a dump title.

DMP.SYSIN DD Statement
defines the data set that contains the PRDMP control statements. The data set follows immediately.

GO Control Statement
requests formatting and printing according to the EDIT, SUMMARY and PRINT CURRENT control statements.

END Control Statement
terminates PRDMP processing.

Example 2: Transferring a Dump Data Set and Processing It in a Later Job

If you need to clear a SYS1.DUMP data set to make room for more dump information, you can use PRDMP to transfer its contents to another data set. This new data set is not formatted or printed during execution of PRDMP, but it can be used later as input.

This example shows how to transfer a SYS1.DUMP data set that is a cataloged on a direct access storage device to a tape volume described by the SYSUT2 DD statement. It also shows how to refer to the transferred data set in a later job.

Notes:

1. *When transferring SYS1.DUMP to a SYSUT2 data set, do not use the cataloged procedure PRDMP; the cataloged procedure contains a SYSUT1 DD statement, and the SYSUT1 and SYSUT2 DD statements may never be used in the same step.*

2. *If the SYS1.DUMP data set is date protected, the operator receives message IEC107D requesting permission to proceed. You must respond by entering r 00,'U' to allow PRDMP to continue processing.*

3. *This example does not include any PRDMP control statements except END. If other control statements were included, they would be ignored.*

```
//CLEAR          JOB          MSGLEVEL=(1,1)
//STEP1          EXEC         PGM=AMDPRDMP
//SYSPRINT       DD           SYSOUT=A
//PRINTER        DD           SYSOUT=A
//TAPE           DD           DSNAME=SYS1.DUMP00,DISP=OLD
//SYSUT2         DD           DSN=DUMP1,UNIT=2400,VOL=SER=DUMP,
             LABEL=(,NL),DISP=(NEW,KEEP)
//SYSIN          DD           *
             END
/*
************************************************************
//PROCESS        JOB          MSGLEVEL=(1,1)
//STEP           EXEC         PGM=AMDPRDMP
//TAPE           DD           DSN=DUMP1,VOL=SER=DUMP,LABEL=(,NL),
//           DISP=OLD,UNIT=2400
//SYSUT1         DD           UNIT=3330,SPACE=(CYL,(10,20)),
//           DISP=(NEW,DELETE),VOL=SER=111111
//PRINTER        DD           SYSOUT=A
//SYSPRINT       DD           SYSOUT=A
//SYSIN          DD           *
             FORMAT
             CPUDATA
             END
/*
```

This example consists of two separate jobs. **In the first job:**

SYSPRINT DD Statement
defines the message data set.

PRINTER DD Statement
defines the data set to which PRDMP ordinarily directs its output. This statement must be included, even though its function is not used in this application.

TAPE DD Statement
defines the input data set, SYS1.DUMP00.

SYSUT2 DD Statement
defines the data set to which the contents of SYS1.DUMP00 will be transferred.

SYSIN DD Statement
defines the data set that contains the PRDMP control statements. The data set follows immediately.

END Control Statement
terminates PRDMP processing. Note that this is the only PRDMP control statement needed.

Note: If one of the format control statements (such as QCBTRACE, FORMAT, PRINT, etc.) is included in the input stream, the data transfer takes place, but no formatting is done; any subsequent statements are ignored.

In the second job:

TAPE DD Statement
defines the input data set, which in this case is the transferred dump data set processed in the first job.

SYSUT1 DD Statement
defines a work data set into which the input data will be collected and from which it will be processed.

PRINTER DD Statement
defines the output data set.

SYSPRINT DD Statement
defines the message data set.

SYSIN DD Statement
defines the data set that contains the PRDMP control statements. The data set follows immediately.

FORMAT Control Statement
requests formatting of important system data areas.

CPUDATA Control Statement
requests printing of processor-related information including all registers for each processor, the current PSW for each processor and the common system data area. (For further information, see PRINT CPUDATA description in the FORMAT control statements description in this chapter).

END Control Statement
terminates PRDMP processing.

Example 3: Transferring a SYS1.DUMP Data Set and Processing It in the Same Step

If you want to transfer the contents of a SYS1.DUMP data set to another data set and process the dump immediately, you can use a job stream like the one shown here. Note that the dump is directed to a data set defined by the SYSUT1 DD statement, and the SYSUT2 DD statement is not used.

If the SYS1.DUMP data set is date protected, you receive console message IEC107D requesting permission to proceed. You must respond by entering

r 00'U'

to allow PRDMP to continue.

Note: After the PRDMP run, defined in the example below, is completed, the data set resides in two locations, the new data set created on the specified disk and on the SYS1.DUMP00 data set. To clear SYS1.DUMP00, PRDMP must be run stating that input is on SYS1.DUMP00 and output to tape (SYSUT2). (Utility IEBGENER can be used to clear SYS1.DUMP00 instead of running PRDMP a second time.)

```
//TRANS     JOB     MSGLEVEL=(1,1)
//STEP1     EXEC    PGM=AMDPRDMP
//SYSPRINT  DD      SYSOUT=A
//PRINTER   DD      SYSOUT=A
//TAPE      DD      DSNAME=SYS1.DUMP00,DISP=OLD
//SYSUT1    DD      DSN=DUMP2,UNIT=3330,VOL=SER=666666,
//      DISP=(NEW,KEEP),SPACE=4104,(257,1))
//SYSIN     DD      *
        TITLE SYS1.DUMP00 THURSDAY PM
        LPAMAP
        PRINT STORAGE
        END
/*
```

In this example:

SYSPRINT DD Statement
defines the data set to which PRDMP directs its output, which in this case is the processed dump.

PRINTER DD Statement
defines the output data set.

TAPE DD Statement
defines the input data set, SYS1.DUMP00.

SYSUT1 DD Statement
defines a direct access data set to which the contents of SYS1.DUMP00 will be transferred, and from which PRDMP will process the transferred dump. In this example, the SYSUT1 data set is to be kept, to allow further processing at a later time. When you keep the SYSUT1 data set, do not direct more than one dump data set to it.

SYSIN DD Statement
defines the data set that contains the PRDMP control statements. The data set follows immediately.

TITLE Control Statement
supplies a title for the processed dump.

LPAMAP Control Statement
instructs PRDMP to format and print a map of the contents of the link pack area active queue at the time the dump was created.

PRINT STORAGE Control Statement
instructs PRDMP to print a map of storage for the memory that was current at the time the dump was created.

END Control Statement
terminates PRDMP processing.

If you want to process the transferred dump data set again later, define it using a TAPE DD statement and treat it like any direct access input data set.

Example 4: Processing Multiple Data Sets

PRDMP can process any number of input data sets in a single execution, provided that each data set is properly defined by both DD statements and control statements. This example shows how to process three data sets in the same execution, two of which are on the same tape volume.

```
//NOLINK        JOB          MSGLEVEL=(1,1)
//STEP1         EXEC         PGM=AMDPRDMP,PARM='T'
//SYSPRINT      DD           SYSOUT=A
//PRINTER       DD           SYSOUT=A
//TAPE          DD           UNIT=2400,VOL=SER=DPTAPE,
        LABEL=(,NL),DISP=OLD
//TODAYDMP      DD           UNIT=SYSDA,VOL=SER=DADUMP,
        DSNAME=DMPDS,DISP=OLD
//SYSUT1        DD           UNIT=SYSDA,SPACE=(CYL,(10,20))
//SYSIN         DD           *
    ONGO            QCBTRACE,FORMAT,PRINT CURRENT
    GO
    NEWDUMP         FILESEQ=2
    GO
    NEWDUMP         DDNAME=TODAYDMP
    ONGO
    GO
    END
/*
```

SYSPRINT DD Statement
defines the message data set.

PRINTER DD Statement
defines the output data set.

TAPE DD Statement
defines two input data sets on the same tape volume.

TODAYDMP DD Statement
identifies an input data set on a direct access volume.

SYSUT1 DD Statement
defines the PRDMP work data set; it is required in this example because one of the input data sets is on a direct access volume.

SYSIN DD Statement
defines the data set containing the control statements. The data set follows immediately.

First ONGO Control Statement
alters the default parameters (EDIT, SUMMARY, PRINT CURRENT) for all subsequent GO statements by deleting the SUMMARY and EDIT parameters and adding QCBTRACE and FORMAT.

GO Control Statement #1
instructs PRDMP to process the first data set on the volume described by the TAPE DD statement.

NEWDUMP Control Statement with FILESEQ=2
identifies the second data set to be processed. Because a DDNAME parameter is not specified, PRDMP assumes that the data resides on the volume described by the TAPE DD statement. FILESEQ=2 specifies that the second data set on the volume should be processed.

GO Control Statement #2
instructs PRDMP to process the data set described by the NEWDUMP control statement.

NEWDUMP Control Statement with DDNAME=TODAYDMP
identifies the third data set to be processed. DDNAME=TODAYDMP specifies that the data set is the one described by the TODAYDMP DD statement.

Second ONGO Control Statement with No Parameters
restores the original default parameters for the GO control statement.

GO Control Statement #3
instructs PRDMP to process the data set described by the last NEWDUMP control statement. The original default parameters are used.

END Control Statement
terminates PRDMP processing.

Example 5: Editing GTF Trace Data from Buffers in a Dump

This example shows how to edit GTF trace buffers from a dump of main storage.

```
//EDIT          JOB          MSGLEVEL=(1,1)
//STEP1         EXEC         PGM=AMDPRDMP
//SYSPRINT      DD           SYSOUT=A
//PRINTER       DD           SYSOUT=A
//TAPE          DD           UNIT=2400,VOL=SER=DUMP,
        LABEL=(,NL),DISP=OLD
//SYSUT1        DD           UNIT=SYSDA,SPACE=(CYL,(10,20))
//SYSIN         DD           *
        EDIT
        END
/*
```

In this example:

SYSPRINT DD Statement
defines the message data set.

PRINTER DD Statement
defines the output data set.

TAPE DD Statement
defines the input data set.

SYSUT1 DD Statement
defines the PRDMP work data set. Although it is not required unless the input data set is on a direct access volume or a multivolume tape, it should be included to reduce PRDMP processing time. When it is included, it must specify enough space to contain the entire dump.

SYSIN DD Statement
defines the data set containing the PRDMP control statements. The data set follows immediately.

EDIT Control Statement with No Parameters
instructs PRDMP to format and print GTF trace buffers in the input data set, according to the default option SYS.

END Control Statement
terminates PRDMP processing.

Example 6: Editing a GTF Trace Data Set

When GTF trace data is recorded in an external data set, you can request the editing of selected records only. This example shows how to edit trace records associated with two specific jobs.

```
//EDIT          JOB          MSGLEVEL=(1,1)
//STEP1         EXEC         PGM=AMDPRDMP,PARM='ER=0'
//SYSPRINT      DD           SYSOUT=A
//PRINTER       DD           SYSOUT=A
//TRACE         DD           UNIT=2400,LABEL=(,NL),VOL=SER=TRACE,
        DISP=OLD
//SYSIN         DD           *
        EDIT        DDNAME=TRACE,JOBNAME=X57A
        EDIT        DDNAME=TRACE,JOBNAME=X56B,
                  SIO=IO=(190,191)
        END
/*
```

In this example:

EXEC Statement
invokes PRDMP and specifies the action that PRDMP should take if a program interruption occurs in a user program.

SYSPRINT DD Statement
defines the message data set.

PRINTER DD Statement
defines the output data set.

TRACE DD Statement
defines the input trace data set.

SYSIN DD Statement
defines the data set containing the PRDMP control statements. The data set follows immediately.

EDIT Control Statement #1
instructs PRDMP to edit trace records in the data set defined by the TRACE DD statement. The JOBNAME=X57A parameter requests editing for only those records associated with job X57A.

EDIT Control Statement #2
instructs PRDMP to edit trace records from the data set defined by the TRACE DD statement; that is, the same data set referred to in the first EDIT statement. This time, however, only records associated with job X56B are to be processed; of those, only SIO and I/O interrupt traces for devices 190 and 191 are edited.

END Control Statement
terminates PRDMP processing

Example 7: Processing a Multi-Volume Page Data Set Dump

This example shows how to process an input data set that spans two tape volumes. In this case, the input data set is an AMDSADMP high-speed dump.

```
//PAGEPRT      JOB         MSGLEVEL=(1,1)
//STEP         EXEC        PROC=PRDMP
//TAPE         DD          UNIT=2400,VOL=SER=(TAPE1,TAPE2),LABEL=(,NL),
//          DISP=OLD
//SYSIN        DD           *
          GO
          END
/*
```

In this example:

EXEC Statement
calls the cataloged procedure to execute AMDPRDMP.

TAPE DD Statement
defines the input dump data set, which is contained on two tape volumes.

SYSIN DD Statement
defines the data set containing the PRDMP control statements.

GO Control Statement
requests PRDMP to process the dump data set using the default parameters, which are SUMMARY, EDIT, PRINT CURRENT.

END Control Statement
terminates processing.

Example 8: Fast Dump Scan Display Session

This example illustrates how PRDMP might be run from a TSO terminal emphasizing the DISPLAY verb and its subverbs. DISPLAY mode is entered to find the asid number for use in the PRINT STORAGE command. It is assumed that the user has defined a logon procedure similar to Figure 3-3 and that the dump data set (MYDUMP) is on a direct access device (2314) with an id of DISK01.

```
* LOGON      IBMUSER SIZE(128)
* ALLOC      FILE(PRINTER) SYSOUT(A)
* ALLOC      FILE(SYSUT1) OLD UNIT(2314) DSNAME('MYDUMP')
             VOLUME(DISK01)
* ALLOC      FILE(SYSTERM) DATASET(*)
* CALL       'SYS1.LINKLIB(AMDPRDMP)'
  DISPLAY                    enter DISPLAY mode
  HARDCOPY ON                request hardcopy
  LIST 4C.                   find CVT
  EQUATE CVT xxxxxx          xxxxxx is address returned from
                             LIST 4C.
  LIST CVT+0                 find CVTTCBP
  EQUATE TNEW xxxxxx         xxxxxx is address returned from
                             LIST CVT+0
  LIST TNEW+C                find pointer to current ASCB
  EQUATE CURASCB xxxxxx      xxxxxx is address returned from
                             LIST TNEW+C
  LIST CURASCB L(250)        find asid at ASCB + X'24'
                             and display 250 bytes
  RETURN                     leave DISPLAY mode
  PRINT STORAGE=(asid,NONUC,NOCOM)
  CVTMAP
  END
* LOGOFF
```

*These are TSO commands. For a description of these commands, see the *OS/VS2 TSO Command Language Reference.*

Chapter 4. PTFLE

Introduction

PTFLE is a problem program that you can use to update an operating system without performing another system generation. It has two different, but related, functions:

1. The **application** function updates an operating system by replacing existing load modules with new load modules containing PTFs (program temporary fixes). It does this in a single operation, by generating control statements and dynamically invoking the linkage editor. Note that you cannot use the application function to apply ICRs (independent component releases) that require assembly before invoking the linkage editor.

2. The **generate** function produces a job stream that, when executed, updates an operating system by replacing existing load modules with new load modules consisting of PTFs (program temporary fixes) or ICRs (independent component releases). Note that the generate function does not actually apply PTFs or ICRs; it only produces a job stream, which you must then execute.

This chapter tells how to use both functions.

Application Function

The PTFLE application function produces control statements needed to apply PTFs, and invokes the linkage editor to apply the PTFs, all in one operation. The application function requires the following input:

- JCL to invoke the program AMAPTFLE. IBM provides a cataloged procedure called PTFLE that includes most of the required JCL.
- A PTFLE control statement for each CSECT to be updated with a PTF.
- An object deck for each PTF to be applied.
- An IDENTIFY statement to flag each changed CSECT.
- The output from the generation Stage I SYSGEN from the operating system that is to be updated.

Figure 4-1 shows how PTFLE uses this input to apply PTFs.

The application function requires a region size of 26K, plus the blocksize in bytes for the data set defined by the PCHF DD statement, plus the storage required for the linkage editor.

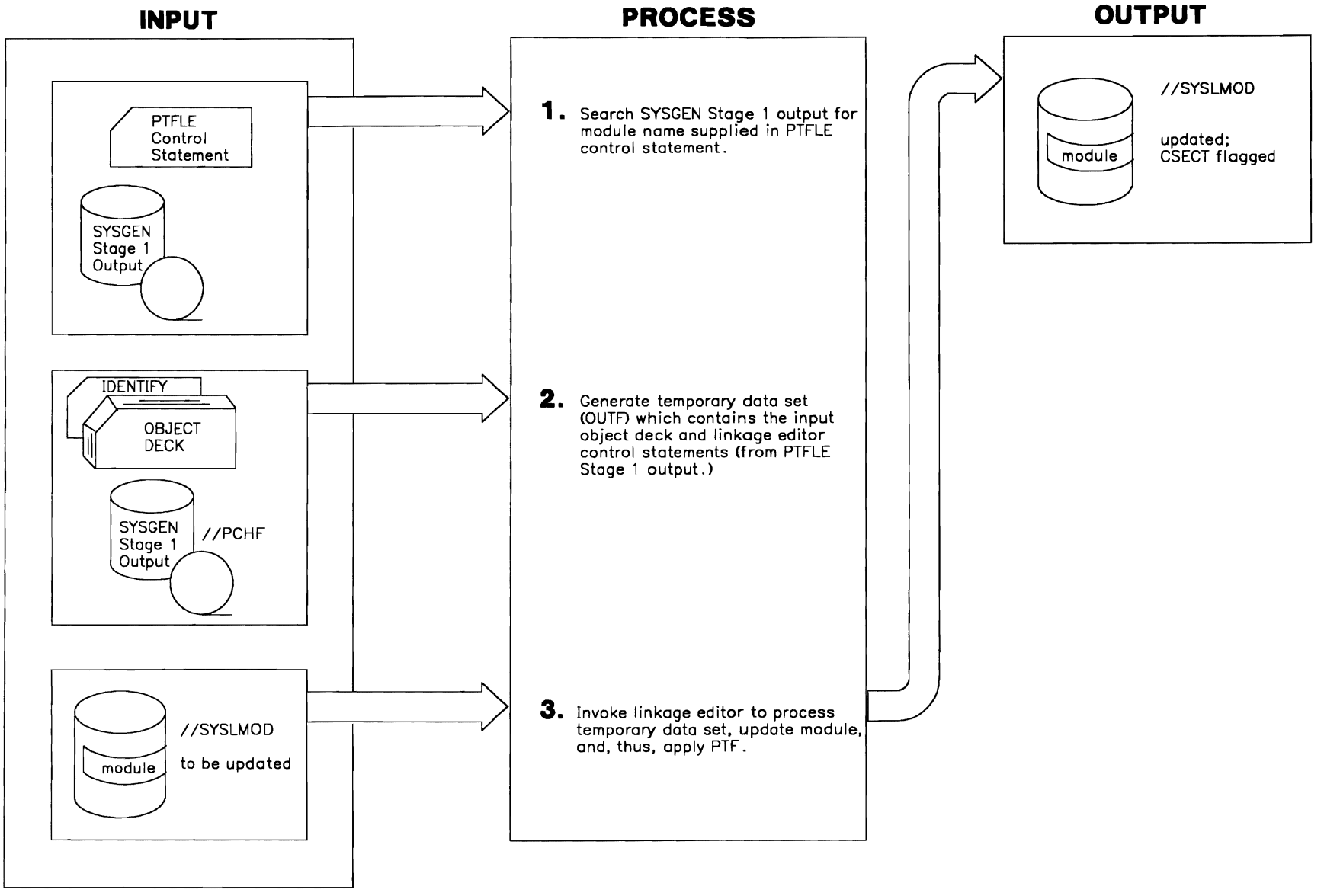


Figure 4-1. Flow of Processing for the Application Function

Using the PTFLE Cataloged Procedure

Figure 4-2 shows the PTFLE cataloged procedure. This IBM supplied procedure assumes the following:

- all system libraries are cataloged
- Stage I output data set consists of unblocked 80-byte records on a non-labeled tape.

```
//PTFJCL      PROC          USE='IEWL',LIB1=LINKLIB
//PTF         EXEC          PGM=AMAPTFLE,PARM=&USE
//PRINT       DD            SYSOUT=A
//PCHF        DD            UNIT=SYSSQ,LABEL=(,NL),DISP=OLD,
//          VOL=SER=STAG1,DCB=(BLKSIZE=80)
//OUTF        DD            UNIT=SYSDA,SPACE=(TRK,(20,20))
//SYSUT1      DD            UNIT=SYSDA,SPACE=(TRK,(20,20))
//SYSUT2      DD            UNIT=SYSDA,SPACE=(TRK,(20,20))
//SYSPRINT    DD            SYSOUT=A
//SYSLMOD     DD            DSNAME=SYS1.&LIB1,DISP=OLD
//PARMLIB     DD            DSNAME=SYS1.PARMLIB,DISP=SHR
```

Figure 4-2. PTFLE Cataloged Procedure -- Application Function Only

PROC Statement
assigns default values for symbolic parameters in the EXEC and SYSLMOD DD statements.

EXEC Statement
invokes AMAPTFLE. The PARM parameter supplies a symbolic name for the linkage editor that PTFLE is to use. The default value, assigned in the PROC statement, is IEWL.

PRINT DD Statement
defines the message data set for PTFLE.

PCHF DD Statement
defines the output data set from the generation Stage I SYSGEN of the system that is to be updated. This output must not contain machine control characters.

OUTF DD Statement
defines a temporary sequential data set used by PTFLE and the linkage editor. This data set can reside on either a magnetic tape or direct access volume. Do not attempt to specify the blocksize.

SYSUT1 DD Statement
defines a work data set for the linkage editor. This data set must reside on a direct access device.

SYSUT2 DD Statement
defines a work data set for PTFLE. This data set must reside on a direct access device. Do not attempt to specify the blocksize.

SYSPRINT DD Statement
defines the message data set for the linkage editor.

SYSLMOD DD Statement
defines the library containing modules that are to be updated. The DSNAME parameter supplies a symbolic name for the library; the default value assigned in the PROC statement is LINKLIB.

PARMLIB DD Statement
defines the SYS1.PARMLIB data set. PTFLE requires this statement whenever it must update the nucleus.

Executing the Application Function

Figure 4-3 is an example of a job stream that executes the application function of PTFLE.

```
//PTFPROC       JOB          MSGLEVEL=(1,1)
//              EXEC         PTFLE
//PTF.MODF      DD           *
IEFSD082 01117251 FIRST PTF
        Insert PTF object deck here
    IDENTIFY CSECT1('LEVEL1PTF'),CSECT5('LEVEL3PTF')
IEFSD085 01117251 SAME PTF
        Insert PTF object deck here
    IDENTIFY CSECT10('HERETOO')
/*
```

Figure 4-3. Sample Job Stream for Executing the Application Function of PTFLE

Application Function Output

When the application function of PTFLE completes processing, all load modules requiring fixes have been updated. No further processing is necessary.

Note: The application function is used before PTFs are applied to a distribution library; to avoid having to re-apply a PTF after system generation, be sure you update the distribution libraries with all PTFs applied to the system.

Generate Function

The PTFLE generate function produces, but does not execute, a job stream that applies PTFs and ICRs. The job stream must be executed in a later, separate step.

The generate function requires the following input:

- JCL to invoke the program AMAPTFLE. Because IBM does not provide a cataloged procedure for this purpose, you must supply your own JCL. (The next section shows you how to write PTFLE JCL.)
- A PTFLE control statement for each CSECT to be updated. (See "Control Statements" in this chapter.)
- (Optional) An IDENTIFY statement to flag each changed CSECT. (See "Control Statements" in this chapter.)
- The output from the generation Stage I SYSGEN of the operating system that is to be updated.

Note: The generate function does not require a PTF object deck.

The generate function also requires that the distribution libraries are updated to contain all PTFs and ICRs that are to be applied to the system. The distribution libraries are input not to PTFLE, but to the program that executes the JCL produced by PTFLE and applies the PTF and ICRs. Use the linkage editor to include PTFs and ICRs in the distribution libraries; for information about using the linkage editor, see the *OS/VS Linkage Editor and Loader*.

Figure 4-4 shows how the generate function uses input.

The generate function requires region size of 47K plus the blocksize in bytes for the data set you define on the PCHF DD statement.

The Stage I output tape requires an index of SYS1. for the data set you specified on the SYSLMOD DD statement that is generated in the output.

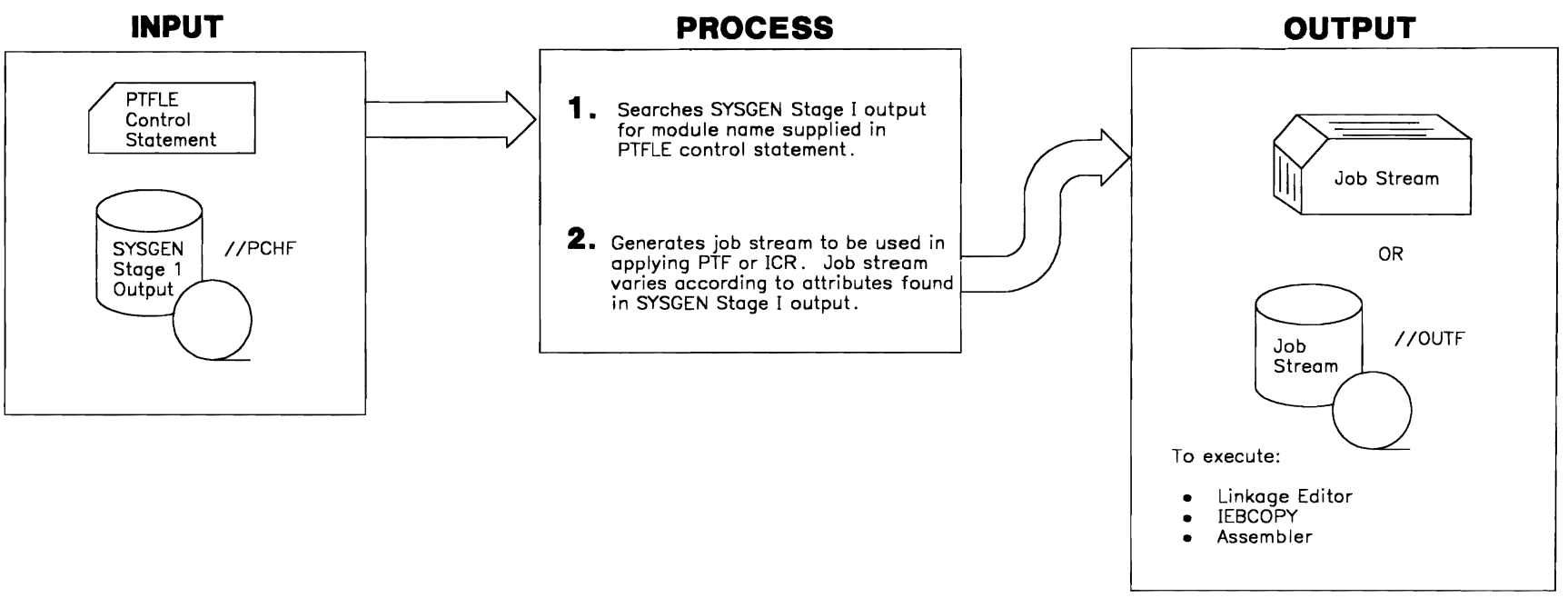


Figure 4-4. Flow of Processing for the Generate Function

Writing JCL for the Generate Function

Figure 4-5 shows the JCL statements that you need to execute the generate function of PTFLE.

```
//GENER          JOB          MSGLEVEL=(1,1)
//               EXEC         PGM=AMAPTFLE
//PRINT          DD           SYSOUT=A
//OUTF           DD           UNIT=2400,LABEL=(,NL),
//       DISP=(NEW,KEEP),VOL=SER=OUTPUT
//PCHF           DD           UNIT=2400,LABEL=(,NL),
         DISP=OLD,VOL=SER=SYSGEN,DCB=(LRECL=80,BLKSIZE=80)
//MODF           DD           *
         Insert Control statements here
/*
```

Figure 4-5. Sample JCL Needed to Execute PTFLE -- Generate Function Only

OUTF DD Statement
defines the output data set, which can be directed to a card punch, a direct access device, or a tape device. Do not specify a block size.

PCHF DD Statement
defines the output from system generation Stage I.

MODF DD Statement
defines an instream data set that contains control statements.

Executing the Generate Function

Figure 4-6 is an example of a job stream you might use to execute the generate function of PTFLE.

```
//PTFJCL         JOB          MSGLEVEL=(1,1)
//               EXEC         PGM=AMAPTFLE
//PRINT          DD           SYSOUT=A
//OUTF           DD           UNIT=SYSDA,VOL=SER=OUTPUT,DISP=(NEW,KEEP),
            DSNAME=DAOUTPUT,SPACE=(TRK,(20,10))
//PCHF           DD           UNIT=2314,DISP=OLD,VOL=SER=SYSGEN,
            DCB=(LRECL=80,BLKSIZE=160)
//MODF           DD           *
IEBGEN03 05199133
IEX51    02150191
   IDENTIFY IEX51000('PTF20191')
IGE0000A 03144004
IGE0000D 02155012
/*
```

Figure 4-6. Sample Job Stream for Executing the Generate Function

For this example, only one module is flagged with an IDENTIFY statement. For the generate function you may omit the IDENTIFY statement; however, the information you supply with the IDENTIFY statement is a valuable diagnostic aid, and it is wise to take full advantage of it.

Generate Function Output

The output of the generate function is a job stream consisting of JCL and control statements. This job stream invokes a program, either the linkage editor, the assembler, or IEBCOPY, to update the target module with a PTF or ICR from the distribution library. If the target module was link edited into the operating system during system generation, the linkage editor is invoked to apply the PTF. If the target module was assembled, first the assembler and then the linkage editor is invoked. If the target module was copied, IEBCOPY is invoked.

PTFLE also provides a listing of the job stream that it produces.

Figure 4-7 is an example of linkage editor-type output produced by the generate function of PTFLE.

```
//SYSGENS     JOB  1,'SYSTEM GENERATION',MSGLEVEL=(1,1)
//SG5   EXEC  LINKS,PARM='NCAL,LIST,XREF,OVLY,XCAL,LET',
//  UNIT='2314',SER=SYSRES,N=SYS1,NAME=LINKLIB,P1=' ',
//  MOD=,P2=' ',OBJ=OBJPDS,CLASS=A
//AOS04  DD   DISP=SHR,VOLUME=(,RETAIN),DSNAME=SYS1.AOS04
//SYSLIN DD   *
  INCLUDE AOS04(AEWLFMAP)
 ENTRY AEWLFROU
 ALIAS IEWK,AEWL
 ALIAS HEWL,HEWLF064
 ALIAS LINKEDIT
  INCLUDE  SYSLMOD(AEWLF064)
 OVERLAY ONE *** VALID EXCLUSIVE CALL TO AEWLFINP ***
 INSERT AEWLFINT,AEWLFOPT
 OVERLAY ONE **VALID EXCL. CALLS TO AEWLFADA,IEWLCFNI,AEWLFADA
 INSERT AEWLFINP,AEWLFESD,AEWLFEND,AEWLFSYM,AEWLFRCG
 INSERT AEWLFSCN,AEWLFRAT,AEWLFINC,AEWLFIDR
 OVERLAY ONE
 INSERT AEWLFMAP
 OVERLAY TWO **VALID EXCL. CALLS TO AEWLFFNL,AEWLFSCD.AEWLFFNL
 INSERT AEWLFADA,AEWLFENT,AEWLFENS,AEWLFOUT
 OVERLAY TWO  *** VALID EXCLUSIVE CALL TO AEWLFFNL ***
 INSERT AEWLFREL,AEWLFSCD,AEWLFSIO
 OVERLAY TWO   *** VALID EXCLUSIVE CALL TO IEWLENAM ***
 INSERT AEWLFFNL,AEWLFBTP
   SETSSI   99999999
 NAME AEWLF064(R)
/*
```

Figure 4-7. Linkage Editor JCL and Control Statements Produced by PTFLE (Generate Function)

Figure 4-8 shows an example of IEBCOPY-type output produced by the generate function of PTFLE.

```
//SG44          EXEC    PGM=IEBCOPY,COND=(8,LT)
//SYSUT3        DD      DISP=SHR,DSNAME=SYS1.UT3
//SYSPRINT      DD      SPACE=(121,(500,100),RLSE),
//      DCB=(RECFM=FB,LRECL=121,BLKSIZE=121),
//      SYSOUT=A
//CI505         DD      DISP=SHR,VOLUME=(,RETAIN),DSNAME=SYS1.CI505
//SVCLIB        DD      DSNAME=SYS1.SVCLIB,VOLUME=(,RETAIN,SER=SYSRES),
//      UNIT=2314,DISP=OLD
//SYSIN         DD      *
    COPY OUTDD=SVCLIB,INDD=CI505
    SELECT MEMBER=((IGE0000A,,R))
    SELECT MEMBER=((IGE0000D,,R))
    SELECT MEMBER=((IGE0000G,,R))
/*
```

Figure 4-8. IEBCOPY JCL and Control Statement Produced by PTFLE (Generate Function)

Figure 4-9 is an example of Assembler and Linkage Editor output produced by the generate function of PTFLE.

```
//SYSGENS    JOB  1,'SYSTEM GENERATION',MSGLEVEL=(1,1)
//SG8   EXEC ASMS,OBJ=OBJPDS,MOD=DCM009,CLASS=A
//SYSIN DD   *
        PRINT ON,NODATA
DCM009  CSECT
        IEECDCM DEVICE=,USE=FC
        END
/*
//SG2   EXEC LINKS,PARM='NCAL,LIST,XREF',
//  UNIT='2314',SER=SYSRES,N=SYS,NAME=LPALIB,P1=' ',
//  MOD=,P2=' ',OBJ=OBJPDS,CLASS=A
//SYSLIN DD  *
    INCLUDE SYSPUNCH(DCM009)
    INCLUDE  SYSLMOD(DCM009)
    SETSSI   33333333
    NAME DCM009(R)
/*
```

Figure 4-9. Example of Assembler and Linkage Editor Output Produced by PTFLE (Generate Function)

Control Statements

Both functions of PTFLE require a PTFLE control statement for each module to be updated. The application function also requires a linkage editor IDENTIFY statement for each module. The IDENTIFY statement is optional in the generate function. The following sections describe how to code these control statements.

PTFLE Control Statement

The PTFLE control statement consists simply of a module name from 1 to 8 characters long, an 8-character system status information (SSI) number, and any comments you may wish to add. The module name must begin in column 1 and be followed by one or more blanks. The SSI number must begin in column 10 and be followed by one or more blanks. Only blanks may be inserted between the module name and the SSI number. Here are several examples of PTFLE control statements coded correctly:

```
IEBGEN04 05199134 THIS IS A MODULE TO BE UPDATED
IEBGEN05 05199135 THIS IS ANOTHER MODULE TO BE UPDATED
MYMOD    06123487 THIS MODULE NAME HAS ONLY FIVE CHARACTERS
MOD1     06134567 NOTICE THAT THE MODULE NAME CAN BE 1 TO 8 CHARACTERS
MOD2     06145678 THE SSI NUMBER HOWEVER MUST ALWAYS START IN COLUMN 10
```

Module Name Parameter

You must supply a PTFLE control statement for each module that you want to update. For modules that have alias names and that were copied rather than link edited during system generation, you must supply a separate control statement for each alias name. Alias-name control statements need not contain SSI numbers. Here is an example of control statements defining a single module with many alias names:

```
MODULE22 05167788 THIS IS THE TRUE MODULE NAME
ALIAS1
ALIAS2
ALIAS3
ALIAS4
```

In any one execution of PTFLE, you may include up to 150 control statements. For the generate function, you must count all alias statements toward this maximum.

If any module being updated has both a component library name and a system library name, you need to include only the component library name in a PTFLE control statement.

With one exception, you can use PTFLE to update a module whose name in the distribution library differs from the CSECT name in the module. The exception is any module that was link edited rather than copied during system generation and

whose overlay structure was defined using INCLUDE statements rather than INSERT statements. The FORTRAN H compiler is an example of such a module.

Modules copied from the distribution library during system generation may be updated using PTFLE, providing the SELECT statement was used in the copy operation.

SSI Number Parameter

The number you specify in the SSI field of a PTFLE control statement must be the number that is listed under the heading "Status Info" on the PTF cover letter. This number is placed in the library directory entry for the updated module to indicate that the PTF was changed. If you omit the SSI field from a control statement containing a true module name, the SSI field in the module is set to zeros; you can, however, omit the SSI field from alias control statements without altering the SSI.

IDENTIFY Control Statement

The IDENTIFY control statement allows you to flag the specific CSECT within a module that is to be updated with a PTF or ICR. PTFLE does not use the IDENTIFY statement directly, but passes it to the linkage editor for processing. For the application function, you must include an IDENTIFY statement for each module that is to be updated; if you omit the IDENTIFY statement for one module, PTFLE issues an error message and terminates processing. For the generate function, the IDENTIFY control statement is optional.

Code the IDENTIFY statement according to the following rules:

- Always begin the IDENTIFY statement in or after column 2.
- You may specify as many as 40 characters of identifying information for each CSECT name.
- To continue the IDENTIFY statement, close the first card with a delimiting comma and a nonblank character in column 72, and start the next card in column 16. Note, however, that PTFLE allows a maximum of 150 IDENTIFY statements in a single execution, and all IDENTIFY continuation statements must be counted toward this total.

Here are some examples of IDENTIFY control statements:

```
IDENTIFY MYCSECT('PTF41392547'),YOURCSCT('PTF12345678')
IDENTIFY CSECT1('***THIS IS A 40 CHARACTER IDENTIFIER****')
   IDENTIFY     CSECT2('PTF1'),CSECT3('PTF2'),CSECT4('PRF3')
IDENTIFY CSECT1('PTFA'),CSECT2('PTFB'),CSECT3('PTFC'),CSECT4('PTFD'), x
              CSECT5('PTFE'),CSECT6('PTFF')
```

Chapter 5. SADMP

Introduction

SADMP is a stand-alone program that produces either

- a formatted dump of real storage on a tape or printer, or
- an unformatted machine-readable dump of real and virtual storage on a tape.

The formatted dump that is produced by low-speed SADMP and directed to tape can be printed by the PRDMP service aid or the IEBPTPCH utility program.

The unformatted, machine-readable dump, which is produced by high-speed SADMP, can be formatted and printed by the PRDMP service aid. Note that the unformatted dump cannot be directed to a printer.

Steps to Generate and Execute SADMP

AMDSADMP is supplied as a macro definition in the system library SYS1.MACLIB. The following is a summary of the steps to generate and execute SADMP:

1. Code the AMDSADMP macro instruction, specifying the type of SADMP program you want: high-speed or low-speed, residing on tape or direct access device, with output to tape or printer.

2. Assemble the AMDSADMP macro instruction. This produces the stage 2 JCL and control statements that are used in the following step.

3. Execute the job stream produced in the previous step. Execution of this step creates the desired SADMP program in executable form and places it on the SADMP residence volume.

4. Execute the SADMP program via an IPL of the SADMP residence volume.

Notice that steps 1 through 3 are all performed under the operating system. Step 4 is a stand-alone operation.

Notes:

1. *Stand-alone dump uses only real, online devices. When dumping to or from devices that have both real and virtual addresses, specify only real addresses to SADMP. Also, stand-alone dump must reside on a real storage device.*

2. *Do not IPL stand-alone dump via a processor from a channel controlled by the channel reconfiguration hardware (CRH).*

3. *SADMP output cannot be directed to its residence volume.*

4. *The SADMP residence volume and operator console must be attached to the same CPU or channel set.*

5. *The output device can be attached to another CPU when the CPU SADMP was IPLd from does not have a device attached to it with the same I/O address.*

6. *The SADMP operator console must be specified on the CONSOLE keyword.*

The remainder of this chapter contains the detailed explanations of how to perform each of these steps.

Coding the Macro Instruction

The SADMP program has the following basic variations:

- High-speed, residing on a direct access device, with output directed to a tape volume.
- High-speed, residing on a tape device, with output directed to a tape volume.
- Low-speed, residing on a direct access device, with output directed to a tape volume.
- Low-speed, residing on a direct access device, with output directed to a printer.

The high-speed version of SADMP dumps real storage and areas of virtual storage in each address space. The output is intended for later machine processing by PRDMP. The output includes: a dump title, the processor store status information for each processor, real storage from address 0 to the top of real storage in real address sequence (some blocks may be missing because of offline main storage), instruction trace data (created by the console - initiated loop recording), and selected virtual storage areas.

The system data areas dumped from real storage are:

- The nucleus
- Subpool 245 - SQA
- Subpools 227 and 228 - CSA

The system data areas dumped from virtual storage are:

- Subpools 231, 239, and 241 - CSA
- Subpool 255 - LSQA
- Subpools 236 and 237 - SWA
- Subpools 229 and 230 - data management areas

The 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 real storage (from 0 to 16 megabytes).

The following sections describe how to code the AMDSADMP macro instruction to produce high-speed and low-speed versions of the dump program. For examples, refer to the Examples section of this chapter.

High-Speed Dump Program

Figure 5-1 shows how to code the AMDSADMP macro instruction to produce a high-speed dump program.

```
[symbol]  AMDSADMP  [TYPE=HI] [,keyword]...

The keywords are:
          IPL={Tunit|Dunit|D3330}
          VOLSER={volser|SADUMP}
          ULABEL={PURGE|NOPURGE}
          CONSOLE={(cnum,ctype)|((cnum,ctype),...)|(01F,3215)}
          SYSUT={unit|SYSDA}
          OUTPUT={Tunit|T282}
```

Figure 5-1. Format of AMDSADMP Macro Instruction Used to Generate a High-Speed Dump Program

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

AMDSADMP
is the name of the macro instruction.

TYPE=HI
specifies the high-speed version of the dump program. If you omit this parameter, TYPE=HI is assumed as the default.

IPL={Tunit|Dunit|D3330}
specifies the device address or the device type of the SADMP residence volume. The first character indicates the volume type; T for tape, D for DASD. The unit character string is used in the stage 2 JCL as the UNIT value to allocate the residence volume for initialization. IPL=D3330 is the default.

VOLSER = {volser|SADUMP}
specifies the VOL = SER value for the stage 2 JCL to allocate the residence volume for initialization. If a tape volume is specified, it must be NL (no labels). VOLSER = SADUMP is the default.

ULABEL = {PURGE|NOPURGE}
specifies whether existing user labels on a DASD residence volume should be deleted (PURGE) or retained (NOPURGE). If you specify NOPURGE, the default, 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.

Note that you must specify ULABEL = PURGE when the residence volume is a 2314 volume that contains user labels.

CONSOLE = {(cnum,ctype)|((cnum,ctype),...)|(01F,3215)}
indicates the device addresses and device types of the system consoles that SADMP is to use while taking the dump. You can specify from 1 to 21 consoles by coding:

CONSOLE = {(cnum,ctype)|((cnum,ctype),...)}

The default, if you omit this parameter, is a 3215 console printer keyboard with the unit address of 01F. The following device types are valid and interchangeable:

- 1052, 2150, 3210, and 3215 (printer keyboards)
- 3066 (system console)
- 3036, 3158, 3277, 3278, and 3279 (console displays)

Note: When SADMP is IPLed, it loads a wait reason code of X'0001FF' into a wait PSW. If one of the consoles from the CONSOLE list is available, press enter to cause SADMP to use this console. If no console from the CONSOLE list is available, ready a tape on the default output device (OUTPUT) and cause an external interrupt on the processor where SADMP was being IPLed. This causes SADMP to proceed without a console using the parameters coded on the AMDSADMP macro to control its execution.

SYSUT = {unit|SYSDA}
specifies the UNIT value for the stage 2 JCL. This is the device to be used for work files during the initialization stage. The device may be specified 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 to be used by SADMP as a default value if the EXTERNAL INTERRUPT key is used to bypass console communication. (Refer to the note under CONSOLE.) High-speed dump output must always be directed to a tape device. This parameter does not allow the same flexibility that the IPL parameter allows (for example, T2400 is not a valid OUTPUT parameter). With a response to message AMD001A, the address that you specify to AMD001A on the OUTPUT

parameter can be overridden at execution time. (Note that a null response causes the OUTPUT value to be used.) The output device need not be attached to the processor from which AMDSADMP is IPLed.

OUTPUT=T282 is the default.

Low-Speed Dump Program

Figure 5-2 shows how to code the AMDSADMP macro instruction to produce a low-speed dump program.

```
[symbol]  AMDSADMP  TYPE=LO [,keyword]

The keywords are:
          IPL={Dunit|D3330}
          OUTPUT={Punit|Tunit|P00E}
          VOLSER={volser|SADUMP}
          ULABEL={PURGE|NOPURGE}
          CONSOLE={(cnum,ctype)|((cnum,ctype),...)|(01F,3215)}
          SYSUT={unit|SYSDA}
          ADDR={REAL|VIRTUAL}
```

Figure 5-2. Format of AMDSADMP Macro Instruction Used to Produce a Low-Speed Dump

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

AMDSADMP
the name of the macro instruction.

TYPE=LO
specifies the low-speed version of the dump program. If you omit this parameter, TYPE=HI is the default.

IPL={Dunit|D3330}
specifies the unit address or the device type of the SADMP residence volume. The first character must be D (for DASD). The unit character string is used in the stage 2 JCL as the UNIT=value to allocate the residence volume for initialization. IPL=D3330 is the default.

OUTPUT={Punit|P00E|Tunit}
specifies the device to which SADMP output is to be written. The first character indicates the output device type; P for printer, T for tape. The unit character string is the unit address to be used when the EXTERNAL INTERRUPT key is used to bypass console communication. (See the note under CONSOLE.)

This parameter does not allow the same flexibility that the IPL parameter allows (for example, T2400 is not a valid OUTPUT parameter). With a response to message AMD001A, the address that you specify to AMD001A on the OUTPUT parameter can be overridden at execution time. (Note

that a null response causes the OUTPUT value to be used.) The output device must be attached to the processor from which AMDSADMP is IPLed.

OUTPUT=P00E (that is, a printer) is the default.

VOLSER={volser|SADUMP}
specifies the VOL=SER value for the stage 2 JCL to allocate the residence volume for initialization. VOLSER=SADUMP is the default.

CONSOLE={(cnum,ctype)|((cnum,ctype),...)|(01F,3215)}
indicates a list of the device numbers and device types for the system console that SADMP may use while taking the dump. You can specify from 1 to 21 consoles by coding:

CONSOLE={(cnum,ctype)|((cnum,ctype),...)

The default, if you omit this parameter, is a 3215 console printer keyboard with the unit address of 01F. The following device types are valid, and are interchangeable:

- 1052, 2150, 3210, and 3215 (printer keyboards)
- 3066 (system console)
- 3036, 3158, 3277, 3278, and 3279 (console displays)

Note: When SADMP is IPLed, it loads a wait reason code of X'0001FF' into a wait PSW. If one of the consoles from the CONSOLE list is available, press enter to cause SADMP to use this console. If no console from the CONSOLE list is available, ready a tape on the default output device (OUTPUT=) and cause an external interrupt on the processor where SADMP was being IPLed. This causes SADMP to proceed without a console, using the parameters coded on the AMDSADMP macro to control its execution.

SYSUT={unit|SYSDA}
specifies the UNIT value for the stage 2 JCL. This is the device to be used for work files during the initialization stage. The device may be specified 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 existing user labels on a DASD residence volume should be deleted (PURGE) or retained (NOPURGE). If 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.

Note that you must specify ULABEL=PURGE when the residence volume is a 2314 volume that contains user labels.

ADDR = {REAL|VIRTUAL}
specifies the default action to be taken by SADMP when the console is unavailable or when the operator specifies end of block to a prompting message for the type of dump desired. REAL indicates that real storage (from 0 to 16 megabytes) is to be dumped in ascending order by real addresses. VIRTUAL indicates that real storage (from 0 to 16 megabytes) is to be dumped in ascending order by virtual addresses. The virtual addresses are in the segment table of the address space in control when the IPLed processor was stopped or if no store status was done. The default is REAL.

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

Assembling the Macro Instruction

The next step in generating the stand-alone dump program is assembling the macro instruction. Figure 5-3 is an example of the JCL statements needed for this operation. This example uses ASMFC, the standard IBM-supplied cataloged procedure for invoking an assembler.

```
//ASMSADMP      JOB         MSGLEVEL=(1,1)
//              EXEC        ASMFC,PARM.ASM='DECK'
//ASM.SYSIN     DD          *
                AMDSADMP    TYPE=HI
                END
/*
```

Figure 5-3. Sample JCL Needed to Assemble the AMDSADMP Macro Instruction

JOB Statement
initiates the job.

EXEC Statement
invokes the cataloged procedure ASMFC, which does the following:

- Invokes an assembler.
- Identifies the system macro library (SYS1.MACLIB), which contains the AMDSADMP macro definition.
- Defines work data sets for the assembler's use.
- Defines two output data sets (SYSPRINT and SYSPUNCH).

The EXEC statement also requests that the assembler output be punched as a deck.

ASM.SYSIN DD Statement
defines the input stream, which in this case consists of the AMDSADMP macro instruction and an END control statement.

Output from this assembly is an object deck and a listing of the statements in the deck. The deck contains JCL and control statements; these constitute a job stream that creates the stand-alone dump program and initializes it on a tape or direct access volume.

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

Error Messages from AMDSADMP Assembly

On the following pages are the error messages that can appear on the output listing when you assemble the AMDSADMP macro. Words beginning with & are replaced with the value that you specified on the macro.

```
IPL=&IPL IS INVALID, IPL=D3330 IS ASSUMED
```

Explanation: The IPL operand is invalid. It is greater than 7 characters, or less than 4 characters, or not prefixed with a 'T' or a 'D'.

Severity Code: 4.

```
CONSOLE ADDR= IS INVALID, CONSOLE ADDR=01F IS ASSUMED
```

Explanation: The console address operand is not three characters.

Severity Code: 4.

```
CONSOLE TYPE= IS INVALID, CONSOLE TYPE=3215 IS ASSUMED
```

Explanation: An invalid console type was specified. The acceptable console types are:

1052	3158	3278
2150	3210	3279
3036	3215	
3066	3277	

The length of the console type is not equal to 4.

Severity Code: 4.

```
TYPE=&TYPE IS INVALID, TYPE=HI IS ASSUMED
```

Explanation: Type operand must be HI or LO.

Severity Code: 4.

```
ADDR = &ADDR IS INVALID. ADDR = REAL IS ASSUMED
```

Explanation: The ADDR operand must be REAL or VIRTUAL.

Severity Code: 4.

```
OUTPUT=&OUTPUT IS INVALID, OUTPUT=P00E IS ASSUMED
```

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.

```
PARAMETERS IPL=&IPL2 AND TYPE=&TYPE ARE INCOMPATIBLE
MACRO PROCESSING TERMINATED
```

Explanation: IPL = Txxx and TYPE = LO are incompatible. A low-speed dump program may reside only on a direct access device.

Severity Code: 8.

```
OUTPUT=&OUTPUT IS INVALID, OUTPUT=T282 IS ASSUMED
```

Explanation: For TYPE = HI the output address was not prefixed by a 'T' or the address was not three characters.

Severity Code: 4.

```
SYSUT=&SYSUT IS INVALID, SYSUT=SYSDA IS ASSUMED
```

Explanation: The SYSUT operand exceeds 6 characters.

Severity Code: 4.

```
VOLSER=&VOLSER IS INVALID, VOLSER=SADUMP IS ASSUMED
```

Explanation: The VOLSER operand exceeds 6 characters.

Severity Code: 4.

```
ULABEL=&ULABEL IS INVALID, ULABEL=NOPURGE IS ASSUMED
```

Explanation: The ULABEL operand is not PURGE or NOPURGE.

Severity Code: 4.

Directing Assembly Output to Tape or DASD

You can override the cataloged procedure ASMFC to direct the object module output from the assembly to a tape or direct access volume. To direct output to tape, add the following statement to the JCL shown in Figure 3-3 on page 3-14.

```
//ASM.SYSPUNCH DD UNIT=2400,LABEL=(,NL),DISP=(NEW,KEEP),
//                VOL=SER=SCRTCH
```

To write the output on a direct access device, use the following statement:

```
//ASM.SYSPUNCH DD UNIT=SYSDA,SPACE=(TRK,(2,1)),DSN=DMPPACK,
//          DISP=(NEW,KEEP),VOL=SER=SCRTCH
```

Assembling Multiple Macro Instructions

If you anticipate need for more than one version of the stand-alone dump program in your installation, you can save time by assembling all applicable variations of the AMDSADMP macro instruction in the same step. Separate the versions by coding a unique symbol at the beginning of each macro instruction. SADMP uses the symbol you code to create a jobname for the initialization program.

Here is an example of a job stream used to assemble four versions of the AMDSADMP macro instruction. Note that you must specify a different residence volume for each program you generate. Output from this step is an object deck and assembly listing for each of the four versions.

```
//ASMSADMP      JOB           MSGLEVEL=(1,1)
//               EXEC          ASMFC,PARM.ASM='DECK'
//ASM.SYSIN     DD            *
HITAPE   AMDSADMP      IPL=T2400,VOLSER=SADMP1
HIDISK   AMDSADMP      VOLSER=SADMP2
LOTAPE   AMDSADMP      TYPE=LO,OUTPUT=T282,VOLSER=SADMP3
LOPTR    AMDSADMP       TYPE=LO,VOLSER=SADMP4
         END
/*
```

Initializing the Residence Volume

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

Execution of the stage 2 JCL initializes the SADMP residence volume. During execution of this stage, a SYS1.PAGEDUMP data set is created (on the residence volume) that is used to contain the SADMP programs.

Physical output from the initialization step is a listing, that may contain the following error messages.

Error Messages from Stage 2

These messages identify errors in the keywords of the AMDSADM2 macro that is generated by the assembly of the AMDSADMP macro. The following list shows the keywords from the error messages and the corresponding keywords on the AMDSADMP macro that require correction.

Message	AMDSADMP Keyword
TYPE2	TYPE
IPL2	IPL
OUTPUT2	OUTPUT
ADDR2	ADDR
CONSOL2	CONSOLE list of device addresses and device types

Words shown in the error messages that begin with an & are replaced with the keyword value in error.

To respond to one of these messages, make sure that the input to the assembly step, output from the assembly step, and input to the initialization step are all correct and that all three correspond. Then run the initialization step again.

```
TYPE2=&TYPE2 INVALID; MACRO PROCESSING TERMINATED
```

Explanation: The TYPE2 operand is not HI or LO.

Severity Code: 12.

```
OUTPUT2=&OUTPUT2 FOR TYPE=&TYPE2 INVALID;
MACRO PROCESSING TERMINATED
```

Explanation: For TYPE2 = HI, OUTPUT2 = Pxxx was specified. OUTPUT2 must be Txxx for high-speed dumps.

Severity Code: 12.

```
OUTPUT2=&OUTPUT2 INVALID; MACRO PROCESSING TERMINATED
```

Explanation: For TYPE2 = HI, the OUTPUT2 operand is not of the form Txxx. For TYPE2 = LO, the OUTPUT2 operand is not of the form Txxx or Pxxx.

Severity Code: 12.

```
IPL2=&IPL2 INVALID; MACRO PROCESSING TERMINATED
```

Explanation: The IPL2 operand is invalid; it must be "D" or "T."

Severity Code: 12.

```
IPL2=&IPL2 AND TYPE2=&TYPE2 INCOMPATIBLE;
MACRO PROCESSING TERMINATED
```

Explanation: IPL2 = Txxx and TYPE2 = LO are incompatible. A low-speed dump must reside on a direct access device.

Severity Code: 12.

```
ADDR = &ADDR INVALID. MACRO PROCESSING TERMINATED
```

Explanation: The ADDR2 operand must be REAL or VIRTUAL.

Severity Code: 12.

Note: An invalid console device type causes an assembler error in the expansion of AMDSADM2.

Executing SADMP

IPLing SADMP

To IPL the stand-alone dump program, AMDSADMP, you must use one of the following procedures.

If you have a 308X console with the system control (SC) operator frame, follow procedure A. Otherwise, follow procedure B.

Procedure A

1. STOP all tightly-coupled processors using the MP/AP global STOP function; do not clear storage.
2. Select a processor that was online at the time of the system failure and that has an operator console attached to it.
3. DO NOT perform a STORE STATUS. The STORE STATUS is done automatically prior to the first IPL of stand-alone dump.
4. Mount the volume containing the stand-alone dump program; 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 stand-alone dump (SADMP) using the SC frame.
6. SADMP loads an enabled wait PSW with wait state code X'1FF' to wait for a console interrupt. Press the ENTER or ATTENTION key on one of the consoles that was specified on the CONSOLE keyword. This causes an interrupt that informs SADMP of the console's address.

7. When message AMD001A TAPE= or PTR= appears, ready an output device (for tapes, be sure it is an NL tape that has been initialized with a tape mark, and the file protect ring is inserted.) Then reply with the tape or printer address. If console communication with SADMP is impossible, the output device must be the device specified on the OUTPUT keyword. When the output device is ready, an external interrupt signals SADMP to begin execution.

8. For TYPE = LO, message AMD008A ADDR= is issued. Respond with R for real or V for virtual followed by the address range to be dumped. For example, the reply format is Rnnnnnn,nnnnnn. You must specify six digits for both the beginning and ending address. (See the ADDR keyword).

Procedure B

1. STOP all tightly-coupled processors, and do not clear storage.

2. Select a processor that was online at the time of the failure and that has an operator console attached to it.

3. IMPORTANT: Perform the STORE STATUS operation as described in the System/370 Operating Procedures manual for your model. Be sure to perform the STORE STATUS operation only on the processor to be initialized. If the STORE STATUS operation is not performed, the program continues processing but most status information is lost. (See the following note on re-IPLing SADMP.)

4. Set the load unit dials on the system control panel to the address of the device where the SADMP residence volume is mounted; or, enter the corresponding data on the operator console screen.

5. Mount the volume containing the SADMP program and ready the device.

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

6. Press the LOAD button or trigger the load operation via interaction with the operator console screen.

7. SADMP loads an enabled wait PSW with wait state code X'1FF' to wait for a console interrupt. Press the ENTER or ATTENTION key on one of the consoles that was specified on the CONSOLE keyword. This causes an interrupt that informs SADMP of the console's address.

8. When message AMD001A TAPE= or PTR= is displayed, ready an output device. (For tapes, be sure it is an NL tape that has been initialized with a tape mark, and the file protect ring is inserted.) Then reply with the tape or printer address. If console communication with SADMP is impossible, the output device must be the device specified on the OUTPUT= keyword. When the output device is made ready, an external interrupt signals SADMP to begin execution.

9. For TYPE=LO, message AMD008A ADDR= is issued. Respond with R for real or V for virtual followed by the address range to be dumped. For example, the reply format is Rnnnnnn,nnnnnn. You must specify six digits for both the beginning and ending address. (See the ADDR keyword.)

Note: If you re-IPL SADMP *do not perform* step 3, the STORE STATUS operation, again. Certain locations might not reflect the original contents of real storage, because these areas could be altered by the initial execution of SADMP.

Restarting SADMP

SADMP can terminate prematurely. For example, if there is a serious I/O error from the output tape or an internal program error, SADMP terminates. However, it is often possible to restart SADMP instead of re-IPLing. A restart is preferable to a re-IPL, because a successful restart maintains a complete dump data set.

To **restart** SADMP, you should follow these steps:

1. STOP all tightly-coupled processors.
2. Perform a PSW RESTART on the processor from which you IPLed SADMP.
3. If the restart succeeds, SADMP reinitializes itself and reloads an enabled wait code X'1FF'.
4. Proceed beginning with step 7 listed above.

If you are taking a high-speed dump, the output tape may stop periodically during the virtual dump execution. This is caused by channel contention between the input and output devices. To avoid channel contention and gain faster operation, you should put input and output devices on different channels.

Operator Communications During Execution

Messages

SADMP loads an enabled wait PSW with code X'1FF' to wait for a console. Press ENTER or ATTENTION on one of the consoles defined on the CONSOLE keyword. If no console is available, the operator can press the EXTERNAL INTERRUPT key. The dump program then bypasses operator communication and attempts to dump to the unit address specified on the OUTPUT keyword.

If the dump program is high-speed, or low-speed with output directed to tape, you receive the following:

```
AMD001A TAPE=
```

This message allows you to accept the tape device specified in the macro instruction by pressing ENTER, or specify a different tape device by entering its device address. SADMP checks the output volume to make sure it is non-labeled. If a label is present, SADMP issues error message AMD002I and reissues message AMD001A, requesting that you identify the address of a tape device that has an unlabeled tape.

If the dump program is low-speed with output directed to a printer, you receive the following:

```
AMD001A PTR=
```

This message allows you to accept the printer device specified in the macro instruction or specify a different printer.

When SADMP accepts an output device specification, it issues message AMD011A. This message requests that you supply up to 100 characters to be used as a dump title. You should use this title to indicate why the dump is required.

After title processing, the low-speed dump program prompts you with the following message for a real or virtual address range to dump:

```
AMD008A ADDR=
```

The answering format has to be a 'V' or 'R' character immediately followed by a six-digit starting address, one comma, and a six-digit ending address. For example, V020000,040000 or R000200,000800 are acceptable. The addresses you specify are rounded (up and down) to a 4K boundary before dumping. The default address range is for all real or virtual storage, as specified in the macro instruction.

When SADMP finishes dumping real storage, it issues this message:

```
AMD005I REAL DUMP DONE
```

If the dump program is high-speed, virtual dumping is performed next.

If you want to terminate SADMP before normal termination, press the EXTERNAL INTERRUPT key any time during virtual storage processing.

Note that when using the 3158 or 3277 operator console in the display mode, the cursor is set automatically to accept operator responses to given messages (for example, printer or tape devices). However, once an operator response is accepted by SADMP, any attempts to modify that response during a later request for input may cause unpredictable results. For example, you should not attempt to change the tape device while entering the dump title: the tape device input will already have been accepted by SADMP during a previous operator response.

Here is a sample exchange between SADMP and an operator during execution of a high-speed SADMP program.

```
AMD001A TAPE=580
AMD011A TITLE=sample high-speed dump
AMD005I REAL DUMP DONE
AMD023I VIRTUAL DUMP COMPLETE FOR CSA
AMD010I PROCESSING ASID=0001 ASCB=030348 JOBNAME=*MASTER*
AMD010I PROCESSING ASID=0002 ASCB=030EE0 JOBNAME=*AUXSTM*
AMD010I PROCESSING ASID=0003 ASCB=FFD4C0 JOBNAME=JES2
AMD010I PROCESSING ASID=0004 ASCB=FF61C0 JOBNAME=GTFSNP
AMD023I STAND-ALONE COMPLETE FOR GTF
AMD010I PROCESSING ASID=0005 ASCB=FD9F20 JOBNAME=INIT
AMD023I STAND-ALONE DUMP COMPLETE - 00
```

In this example, the underlined characters represent the operator's replies.

SADMP also uses wait state codes to communicate with the operator. These are described in *OS/VS2 MVS 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.

Note: 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
MSADMP#N (alternating)
informs the operator that a tape has been rejected and a new tape must be mounted.

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

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

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

SADMP#
indicates that the tape is in use by SADMP.

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

The symbols used in the messages are:

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 has not written on the tape.

M means to mount a new tape.

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

N means the new tape should not be file-protected and should not have a label.

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.

Low-Speed Output

Low-speed SADMP output, if directed to a printer, can be used immediately as a diagnostic aid. Figure 5-4 shows an example of SADMP low-speed output directed to a printer. For a full description of the fields, see the *MVS/370 Debugging Handbook*.

Low-speed SADMP output directed to tape can be printed using either the IEBPTPCH utility or PRDMP. Figure 5-5 and Figure 5-6 show how to use IEBPTPCH and PRDMP, respectively, to print low-speed SADMP output. Note: You can also use the IEBGENER utility program to print low-speed SADMP output. For information about the IEBGENER program, see the *MVS/370 Data Management Utilities*.

High-Speed Output

High-speed SADMP output must be printed using PRDMP. For information about Print Dump, see Chapter 3.

Printing system dumps of five megabytes or larger may result in a completion code of 522, a system abend in print dump. To prevent such abends, use the TIME parameter on the print dump EXEC statement. See *MVS JCL* for further information on the use of the TIME parameter.

```
DUMP TITLE  SAMPLE LOW SPEED DUMP FROM MP SYSTEM

CURRENT PSW          070C0000 00ECA74C            PR        001F5000   CPU ID        00

GR 0-7      80D5BD48 00FF40EC C9C7C5F0 00024868  00CAE2F0 00CAF188 00FF4018 00FF40EC  *.N.... .IGE0......S0..1... ... .*
GR 8-F      80D5BD48 00005D48 00ECA730 80D5BD48  00000000 00CAE530 00CAE4D0 00ECA730  *.N..........N........V...U.....*

CR 0-7      C080EC40 0F082C00 FC000000 00000000  00000000 00000000 00000000 00000000  *... ..............................*
CR 8-F      00000000 00000000 00000000 00000000  00000000 00000000 EFC00000 001EE2F0  *.............................S0*

FR 0-2      00000000 00000000 00000000 00000000                                      *.................               *
FR 4-6      00000000 00000000 00000000 00000000                                      *.................               *

CURRENT PSW          050C1000 00033CCE            PR        001F1000   CPU ID        01

GR 0-7      AD040264 00FF6330 00FFFAD8 FFFFC000  00FFD078 00033B40 00FFFE94 00FFFD20  *...........Q...........  ........*
GR 8-F      00FFFAD8 B00493D4 40049388 00000000  00049544 00000C5C 6004952C 00000000  *...Q...M ......................*

CR 0-7      C080C000 0F1CEC00 FC000000 00000000  00000000 00000000 00000000 00000000  *................................*
CR 8-F      00000000 00000000 00000000 00000000  00000800 01000000 EFC000C0 001E9560  *................................*

FR 0-2      00000000 00000000 00000000 00000000                                      *.................               *
FR 4-6      00000000 00000000 00000000 00000000                                      *.................               *


            STORAGE KEY      06
00000000 V  00080000 0000730C 00000000 00007FF0  0C000000 00007FB8 00000000 00000000  *.............0.............*
00000020 V  00000000 00000000 00000005 E005A932  00000000 00000000 00000000 00000000  *................................*
00000040 V  00007FF0 0C000000 00007FE8 00024868  FFD4CAFF C0FCEDC0 040E0000 00025B6   *...0...........M...........*
00000060 V  040E0000 00001492E 00GE0000 00031BB0  00020000 0010ADF2 040E0000 00030566  *.....................2.........*
00000080 V  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
000000A0 V  00000000 00000000 20000000 00000000  00000000 00000000 00000357 00000000  *................................*
000000C0 V  00000000 00000000 00000000 00000000  00000000 00000000 7FFFFFF2 6015B000  *.........................2....*
000000E0 V  845DF532 8E800000 00000000 00000000  00000000 00000000 00000000 00000000  *..5............................*
00000100 V  050C1000 00033CCE 001F1000 00000000  31000136 60000005 08000110 60000005  *................................*
00000120 V  05007800 60000800 08000140 60000001  00000000 00000000 00000400 00000000  *...........  ...................*
00000140 V  060078C0 60000800 08007800 60000001  00000000 00000000 00000000 00000000  *................................*
00000160 V  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
00000180 V  AD040264 00FF6330 00FFFAD8 FFFFC000  00FFD078 00033B40 00FFFE94 00FFFD20  *...........Q...........  ........*
000001A0 V  00FFFAD8 B00493D4 40049388 00000000  00049544 00000C5C 6004952C 00000000  *...Q...M ......................*
000001C0 V  C080C000 0F1CEC00 FC000000 00000000  00000000 00000000 00000000 00000000  *................................*
000001E0 V  00000000 00000000 00000000 00000000  00000000 00000000 EFC00000 001E9560  *................................*
00000200 V  D7E2C140 00010041 00FFD078 001FD078  00FF6330 001F6330 00000000 00000000  *PSA ...........................*
00000220 V  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*
00000240 V  00000000 00000000 00000000 00000000  000C0000 00000000 00000000 00000000  *................................*
00000260 V  AD000264 00000000 AD00026C 00000000  00000000 00000000 00000000 00000000  *................................*
00000280 V  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  *................................*

000002E0 V  [illegible]                          00000000 00000000  00000000 00000000  *.............................5.*
0000030  [illegible]                                                                  *.............................*
```

Figure 5-4. Sample Low-Speed Dump

```
//PRINTLO       JOB          MSGLEVEL=(1,1)
//             EXEC         PGM=IEBPTPCH
//SYSPRINT      DD           SYSOUT=A
//SYSUT1        DD           UNIT=2400,VOL=SER=DUMPTP,LABEL=(,NL),
//      DISP=OLD,DCB=(BLKSIZE=121,RECFM=F)
//SYSUT2        DD           SYSOUT=A
//SYSIN         DD           *
        PRINT PERFORM=A
/*
```

Figure 5-5. Sample JCL Used to Invoke IEBPTPCH to Print Low-Speed SADMP Output

```
//PTLODUMP      JOB          MSGLEVEL=(1,1)
//              EXEC         PROC=PRDMP
//DMP.SYSIN     DD           *
    PRINT           STORAGE
    END
/*
```

Figure 5-6. Sample JCL Used to Invoke PRDMP to Print Low-Speed SADMP Output

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

In this example, the AMDSADMP macro instruction is used with no parameters to generate a high-speed dump program residing on a direct access volume.

```
DUMP1 AMDSADMP
```

This is equivalent to coding the following parameters:

```
TYPE=HI
IPL=D3330
VOLSER=SADUMP
ULABEL=NOPURGE
CONSOLE=(01F,3215)
SYSUT=SYSDA
OUTPUT=T282
```

Example 2: Generating a High-Speed, Tape Resident Dump Program

In this example, the IPL parameter is coded to specify that the residence volume is to be a tape, and the VOLSER parameter is coded to identify that tape. All other parameters are allowed to default.

```
AMDSADMP IPL=T2400-2,VOLSER=SATAPE
```

The implied defaults are:

```
TYPE=HI
CONSOLE=(01F,3215)
SYSUT=SYSDA
OUTPUT=T282
ULABEL=NOPURGE
```

Example 3: Generating a Low-Speed Dump with Defaults

In this example, only the TYPE parameter is coded to request a low-speed dump. All other parameters are allowed to default.

```
AMDSADMP TYPE=LO
```

The implied defaults are:

```
IPL=D3330
OUTPUT=P00E
VOLSER=SADUMP
CONSOLE=(01F,3215)
ULABEL=NOPURGE
SYSUT=SYSDA
ADDR=REAL
```

Example 4: Generating a Low-Speed Dump Program with Output Directed to Tape

In this example, only the TYPE and OUTPUT parameters are coded. All other parameters are allowed to default.

```
DUMP2 AMDSADMP TYPE=LO,OUTPUT=T282
```

The implied defaults are:

```
IPL=D3330
VOLSER=SADUMP
CONSOLE=(01F,3215)
ULABEL=NOPURGE
SYSUT=SYSDA
ADDR=REAL
```

Example 5: Specifying Multiple Consoles

In this example, one 3215-class and two 327X-class consoles are coded. All other parameters are allowed to default.

```
ANDSADMP CONSOLE = ((009,3215),(3D1,3279),(3E0,3277))
```

The implied defaults are:

```
TYPE   = HI
IPL    = D3330
VOLSER = SADUMP
ULABEL = NOPURGE
SYSUT  = SYSDA
OUTPUT = T282
```

Chapter 6. SPZAP

Introduction

SPZAP is a service aid that operates as a problem program. It is designed to enable authorized personnel to:

- 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.

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 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

Because SPZAP provides the ability to modify data on a direct access storage device, misuse of this program could result in serious damage to both user and system load modules or data sets. To protect against the occurrence of such damage by SPZAP, two means of controlling its use are suggested below:

- One means of exercising control is the System Management Facility (SMF), which provides a system interface with user exit routines to monitor the job stream. This facility affords an internal means of checking to see whether a particular user is authorized to execute the program specified on the EXEC job control language statement. (For further information on the SMF facility, refer to the publication *MVS System Management Facilities (SMF)*).

- A second means of protecting against unauthorized use of SPZAP is to store SPZAP in a "password protected" private library. If SPZAP is located in such a library, any person trying to execute this program would be required to include in his JCL statements a JOBLIB DD statement defining the library, and at initiation time he would be required to give the password associated with the library. Only personnel knowing the password would then be able to execute SPZAP. Note, however, that if SPZAP resides in a private library, the authorized program facility (APF) prevents it from updating a VTOC. Password protected libraries are discussed in the publication *OS/VS2 SPL: Data Management*.

Data Modification and Inspection

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 modification function is controlled by the REP control statement. The REP 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. This function 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 does not be 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 must use 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 processes the first CSECT it encounters in the load module.

SPZAP places descriptive maintenance data in the SPZAP CSECT identification record (IDR) of the load module whenever a REP operation associated with a NAME statement is performed on a CSECT contained in that module. This function is 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 comes from the IDRDATA statement. (See "SPZAP Control Statements" 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 finds 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 use the difference as the displacement of the data.

For a complete description of the control statements mentioned in this discussion, see the section "SPZAP Control Statements" in this chapter.

Figure 6-1 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 1000 8000 00000 00000     381   MVCMSG    MVC     0(1,R1),0(R8)     56200017
                                      382   *                                 56300017
000392 D200 1001 1000 00001 00000     383   MVCBLNKS  MVC     1(1,R1),0(R1)     56400017
                                      384   *                                 56500017




000398                                386   IEFCVOL2  CSECT                     56600017
000398 0590                           387             BALR    R9,0              56700017
00039A                                388             USING   *,R9              56800017
00039A 5820 C010            00010     389             L       R2,LCTJCTAD       56900017

                                          .
                                          .
                                          .
```

Figure 6-1. 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 (cylinder head record) control statement to specify it's direct access address. This 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 AMA121I to provide a record of your request.

```
AMA121I   CCHHR UPDATE BY jjjjjjjj  ON  ser, cchhr, dsn
```

jjjjjjjj	is the name of the job that performed the CCHHR update
ser	is the volume serial number of the direct access device containing the modified data set
cchhr	is the device record address of the record that was modified
dsn	is the name of the modified data set

Accessing a Data Record

When you use the CCHHR control statement, SPZAP is able to read 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 picture of the load module or data record that has been changed, thus allowing you to double check the modifications you have made.

The DUMP and ABSDUMP statements are the control statements used to specify the dumping options. The operation code in the DUMP and ABSDUMP statements indicates 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. (Use of the DUMP and ABSDUMP statements is discussed in detail under the topic "SPZAP Control Statements.")

Updating System Status Information

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

SPZAP also supplies the SETSSI control statement, which 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" in this chapter.

Not all load modules contain system status information. The SSI is located in the last four bytes of the user data field in the directory entry for a load module. Figure 6-2 shows the position of the SSI in load module directory entries.

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

Figure 6-2. SSI Bytes in a Load Module Directory Entry

Figure 6-3 shows the composition of the system status index 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 incremented 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.

Note: Use the LISTLOAD control statement of the LIST service aid to find out what information the SSI contains.

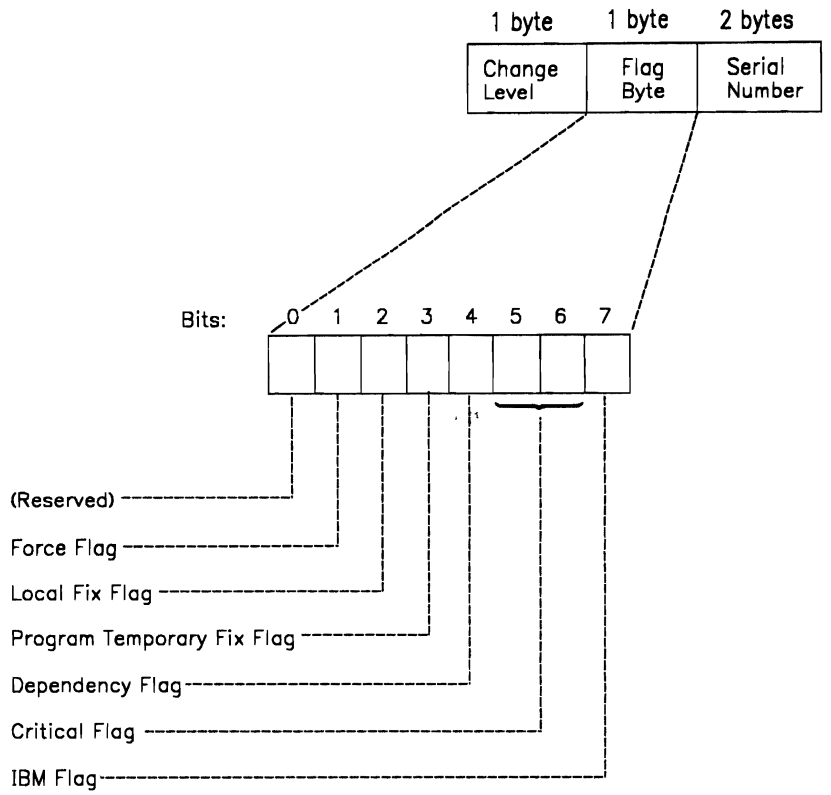


Figure 6-3. Flag Bytes in the System Status Index Field

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:

- The local fix flag contained in bit 2 is used to indicate that the user has modified a particular member. (It is not used to reflect modifications made by IBM-supplied PTFs.) SPZAP sets this local fix flag bit to one after successfully modifying a load module.

- The program temporary fix flag in bit 3 is set to one when an IBM-authorized program temporary fix (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 was enacted, so as not to interfere with the normal system maintenance procedures.

The third and fourth bytes of the system status index are used to store a serial number that identifies the first digit and the last three digits of a PTF number. SPZAP does 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 utilizes system OPEN, and therefore cannot modify "read-only" or inspect "write-only" password protected data sets unless the correct password is provided at OPEN.
- Unexpired data sets such as system libraries cannot be modified unless the operator replies r 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 the following direct access devices: 2314, 2319, 2305, 3330, 3340, 3344, 3350, 3375 and 3380. One of these devices must be specified in the unit parameter of the SYSLIB DD statement.
- SPZAP is a non-reusable module.
- When modifying a system data set, such as SYS1.LINKLIB, DISP=OLD should be specified on the SYSLIB DD statement.

JCL Statements

SPZAP can be executed using the following job control statements. The minimum partition or region for execution 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. Currently the only valid parameter specification is PARM=IGNIDRFULL, which enables SPZAP to override the normal inhibition on CSECT updates (via NAME and REP) when IDR space for the module is found to be full.
Caution: PARM=IGNIDRFULL should not be used indiscriminately and 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 (required for each execution)

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

SYSABEND DD Statement (optional)

defines a sequential output data set to be used in case SPZAP terminates abnormally. This data set can be written to a printer, a magnetic tape volume, or a direct access volume.

SYSIN DD Statement

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

Return Codes

SPZAP termination provides one of the following return codes:

0 Terminates successfully.

4 Warning of condition which may result in future errors if remedial action is not taken.

8 A SPZAP input statement contained an error or was overridden by operator intervention.

12 A requested JCL statement was absent or specified a data set which was not successfully opened. SPZAP terminates immediately.

16 A permanent I/O error, which may be due to a JCL error such as invalid 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 instructions. The program must supply a list of alternate ddnames of data sets 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
specifies the symbolic name of SPZAP.

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 nonzero halfword followed by a character string whose length is given by the halfword. 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 used during SPZAP processing. If all the standard ddnames, SYSPRINT, SYSLIB, SYSIN, are used, then this operand may be omitted.

A ddname list must begin on a halfword boundary. The first two bytes contain a count of the number of bytes in the remainder of the list. The format of the list is fixed with each entry having eight bytes. Each name of less than eight bytes must be left justified and padded with blanks. If a name is omitted 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
40-47	SYSPRINT

{VL / VL=1}
specifies that the sign bit is to be set to 1 in the last word of the address parameter list.

SPZAP Control Statements

The 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.

SPZAP control statements must be coded according to the following rules:

- SPZAP control statements may begin in any column, but the operation name must precede the parameters.
- There must be at least one blank between the specified operation name and the first parameter.
- All parameters must also be separated by at least one blank space.
- Data fields parameters may be formatted with commas for easier visual check, but embedded blanks within data fields are not permitted.
- Data and offset parameter values must be specified as a multiple of two hexadecimal digits.
- The size of a SPZAP control statement is 80 bytes.
- Following the last required parameter and its blank delimiter, the rest of the control statement space can be used for comments. Exceptions to this are the NAME and DUMP control statements. If the CSECT parameter is omitted from either of these statements, the space following the load module parameter should not be used for comments.
- A record beginning with an asterisk and a blank is considered to be a comment statement.

The control statements are described below.

NAME member [csect]

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

member

indicates the member name of the load module that contains the control section in which the data to be inspected and/or modified is resident. The load module must be a member of the partitioned data set defined by the SYSLIB DD statement.

csect

indicates the name of the particular control section that contains the data to be verified or replaced. When this parameter is omitted, it is assumed that the first CSECT contained in the load module is the one to be referenced. If there is only one CSECT in the load module, this parameter is not necessary.

Note: More than one NAME statement can be defined in the input to 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 references the data in the specified record. The parameter is:

record address

provides 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: More than one CCHHR statement can be defined in the 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 contents at a specified location within a control section or physical record to be compared with the data the user supplies in the statement. If the two fields being compared are not in agreement, 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.

offset

provides 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 is 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 parameter, the number of bytes of data defined must be specified as a multiple of two hexadecimal digits. If desired, the data within the parameters may be separated by commas (*never*

blanks), but again, the number of digits between commas must also be a multiple of two. For example, the data may look like this:

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

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

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 replaces the data at the record or CSECT location stipulated in the offset parameter field. (Note that you should always use the VERIFY function to make sure you know what you are going to change with the REP function.) Message AMA122I is issued 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 this offset value is outside the limits of data record (physical block) or CSECT being modified, the replacement operation is not 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.

data

defines the bytes of data to be inserted at a specified location. As with the offset parameter, the number of bytes of data defined must be specified as a multiple of two hexadecimal digits. If desired, the data within the parameter may be separated by commas (never blanks), but again, the number of digits between commas must also be a multiple of two. For example, a REP data parameter may look like this:

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

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

Notes:

- Remember that SPZAP automatically updates the system status index (SSI) when it successfully modifies the associated load module. For a more complete explanation of the value of the SSI to the maintenance of a load module, refer to "Updating System Status Information" in this chapter.

- If you are performing multiple VERIFY and REP operations on a CSECT, you must make sure that all VERIFY statements precede all

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, a 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 has been processed by the linkage editor to include CSECT identification records. The parameter is:

xxxxxxxx
is the eight (or less) bytes of user data (with no embedded blanks) that is to be placed in user data field of the SPZAP IDR of the load module. If more than eight characters are in the parameter field, only the first eight characters are 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 are ignored.

SETSSI xxyynnnn
places user-supplied system status information in the PDS (partitioned data set) directory entry for the library member specified in the preceding NAME statement. The SSI, however, must have been created when the load module was link edited. The parameter is:

xxyynnnn
represents the 4 bytes of system status information the user wishes to place in the SSI field for this member. Each byte is supplied as two hexadecimal digits signifying the following:

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

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

Note: Since all bits in the SSI entry are set (reset) by the SETSSI statement, extreme care must be exercised in its use to avoid altering information vital to the depiction of the maintenance status of the program being changed. Message AMA122I is issued to record the SSI as it was before the SETSSI operation was performed. (See the discussion in this chapter entitled "Updating System Status Information.")

DUMP | **DUMPT** **member** **csect** | **ALL**

used to dump a specific control section or all control sections in a load module. The format of the output of this dump is hexadecimal (see the discussion in this chapter entitled "SPZAP Output"). The DUMPT statement differs from the DUMP statement in that it also gives the user an EBCDIC and instruction mnemonic translation of the hexadecimal data. The parameters are:

member
the member name of the load module that contains the control section(s) to be dumped. (Note: This load module must be a member of a partitioned data set that is defined by the SYSLIB DD statement.)

csect|ALL
the name of the particular control section that is to be dumped. To dump all the CSECTs of a load module, code "ALL" instead of the CSECT name; if the CSECT parameter is omitted entirely, 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) produces no output between the statement printback and the dump-completed message.

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

dumps 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 a record is to be formatted, you can specify DCB=(KEYLEN=nn). The value "n" indicates the length of the record key; it should also be specified by the SYSLIB DD statement.

How to dump a VTOC:

Notes:

1. *If you are dumping a VTOC, you should specify DCB=(KEYLEN=44)*

2. *If you are dumping a PDS directory, you should specify DCB=(KEYLEN=8)*

ABSDUMP produces a hexadecimal printout only; ABSDUMPT prints the hexadecimal data, the EBCDIC translation, and the mnemonic equivalent of the data (see "SPZAP Output" in this chapter). The variables are:

startaddr
the absolute direct access device address of the first record to be dumped. This address must be specified in hexadecimal in the form ccccchhhhrr (cylinder, track and record numbers).

stopaddr
the absolute direct access device address of the last record to be dumped. It must be in the same format as the start address.

Note: Both the beginning and ending 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 you specify in the start address must be a valid record number. If a record number of 0 is specified, SPZAP changes it to 1 because 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 continues until the last record on the track specified in the stop address has been dumped.

membername
the name of a member of a partitioned data set. The member can be a group of data records or a load module. In either case, the entire member is dumped when this parameter is specified.

ALL
indicates that the entire data set defined by the SYSLIB DD statement is to be dumped. The amount of space dumped for the data set depends on how the data set is organized:

- For sequential data sets, 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
adjusts offset values that are specified in any subsequent VERIFY and REP statements. You need this statement when the displacements given in the VERIFY and REP statements for a CSECT are 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, you can specify an offset of X'0408' in the REP statement if you include a BASE statement specifying X'000400' prior to the REP statement in the SPZAP input stream.

When SPZAP processes the REP statement, the base value X'000400' is subtracted from the offset X'0408' to determine the relative address of data within the CSECT. The variable is:

xxxxxx
a six-character hexadecimal displacement to be used as a base for subsequent VERIFY and REP operations. This value reflects 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 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 issued to the operator:

```
AMA116A  ENTER AMASPZAP CONTROL STATEMENT OR END
```

The operator may then enter 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 then continues processing control statements from the input stream until reaching end-of-file.

Note: The control statements can be entered through the console in either upper or lower case letters.

***(Comment)**

annotates the SPZAP input stream and output listing. Any number of comment statements can be included in the input stream. When SPZAP encounters a comment, it writes the entire statement to the SYSPRINT data set.

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

CHECKSUM hhhhhhhh

prints or verifies a fullword checksum (parity-check). — A way to check on the preceding control statements, or on SPZAP initialization, for typographical errors. CHECKSUM translates the previous parameter values into hexadecimal and concatenates them as a single string divided into fullwords. For example, the string 12345678FACE produces the checksum OD025678. The sum of each fullword is then compared with the CHECKSUM you supply, or becomes a CHECKSUM, depending on how you code the the control statement.

hhhhhhhh

is an eight-character hexadecimal value that you want SPZAP to compare with the existing checksum value.

If you do not code this variable, a message is written giving the actual value of the checksum, and indicating that the checksum has been reset.

If the variable is invalid or is not equal to the checksum, SPZAP issues a message indicating an invalid parameter or checksum error. REP and SETSSI statements are suppressed until the next NAME or CCHHR statement. Then it issues a second message giving the actual value of the checksum, and indicating that the checksum has been reset.

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
tells 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 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 does not perform the REP and SETSSI operations requested for the module. It does, however, perform the requested DUMP operation before discontinuing the processing. It can 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 is successful).

SETSSI Control Statement
tells SPZAP to replace the system status information in the directory entry for module IEEVLNKT with the SSI data supplied in the statement (provided the VERIFY statement is 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 (provided the REP operation is successful).

DUMP Control Statement
requests that a hexadecimal image of module IEEVLNKT be dumped to the SYSPRINT data set. Because the DUMP statement follows the REP statement, the image reflects the changes made by SPZAP (provided the VERIFY operation is 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
tells SPZAP that the operations defined by the control statements that immediately following this statement are to be performed on the CSECT IEWFETCH. It is 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 (provided the REP operation 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 is 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 are processed. The new SSI contains:

- 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

tells SPZAP that the operations requested on the control statements immediately following it are to be performed on CSECT IEFQMSSS. This CSECT is contained in load module IEFX5000.

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 (provided the first REP operation is successful).

SETSSI Control Statement #1
tells SPZAP to replace the system status information in the directory entry for module IEFX5000 with the SSI data given. The new SSI contains:

- 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 immediately following 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, 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 (provided the second REP operation is successful).

SETSSI Control Statement #2
causes SPZAP to perform the same function as in 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 (VTOC).

```
//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 shows 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 X'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. Because 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, SPZAP issues message AMA117D 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 IEFCVOL2 to be changed resides.

SYSIN DD Statement

defines the input stream that contains the SPZAP control statements.

NAME Control Statement

tells SPZAP that the operations defined by the control statements that immediately following 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 displacements 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 (provided 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 displacement to determine the relative address 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 offset X'0002' (that is, offset X'039A' minus a displacement of X'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. Because the DUMP statement follows the REP statement, the image reflects the changes made by SPZAP (assuming no verification has been rejected).

SPZAP Output

SPZAP provides two different dump formats for the purpose of checking the data that has been verified and/or replaced. These dumps (written to the SYSPRINT data set specified by the user) may be of the formatted hexadecimal type or the translated form. Both formats are discussed below in detail with examples showing how each type looks.

The Formatted Hexadecimal Dump

When you use the DUMP or ABSDUMP control statement, the resulting printout is a hexadecimal representation of the requested data. Figure 6-4 gives a sample of the formatted hexadecimal dump. A heading line is printed at the beginning of each block. This heading consists of the hexadecimal direct access address of the block, a two-byte record length field, and the names of the member and the control section that contain the data being printed (if the dump is for 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.

The Translated Dump

The DUMPT and ABSDUMPT control statements also provide an operation code translation and an EBCDIC representation of the data contained in the dump. Figure 6-5 shows the format of the translated dump. The first byte of each halfword of data is translated into its mnemonic operation code equivalent, provided such a translation is possible. If there is no equivalent mnemonic representational value to be given, the space is left blank. This translated line of codes and blanks is printed directly under the corresponding hexadecimal line. An EBCDIC representation of each byte of data is printed on two lines to the right of the corresponding line of text, with periods (.) substituted for those bytes that do not translate to valid printable characters.

```
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   200030
 000280   D2052004   301C4122   000C5020   C2009640    C20947F0   [illegible]   C2004143   00[illegible]
 [illegible]                                                                              1C
 [illegible]                                                                              0C218
 000[illegible]   [illegible]   5810C200   41110002   [illegible]                          [illegible]780964C
 000600   41F0C014   D205F000   100441FF   0006D201    96FC[illegible]   48E096FC   12EE4780   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
IHMA113I COMPLETED DUMP REQUIREMENTS
```

Figure 6-4. 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        MVI   *.........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...B@.........*
 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  0000                    9280 1000  0A40                   0002 41A0  *.....
```

Figure 6-5. Sample Translated Dump

Appendix A. Writing EDIT User Programs

Introduction

You may want to code special programs to supplement GTF and PRDMP/EDIT operations. EDIT allows for two types of user programs: exit routines and format appendages. Neither type may occupy more than 10K bytes of main storage.

An exit routine allows you to inspect each input trace record before EDIT begins processing it; on the basis of the inspection you must decide whether EDIT should process the record normally or take special action.

A format appendage allows you to format all user trace records of a specified type. A format appendage can receive control before EDIT processes a record and/or during ABDUMP processing of a USR record. A format appendage must be named AMDUSRxx where xx is the hexadecimal form of the format identifier (FID) specified in the GTRACE macro when the record was created. EDIT also accepts format appendages named IMDUSRxx or HMDUSRxx: thus format appendages that were originally written for use with OS/MFT, OS/MVT, or OS/VS1 need not necessarily be rewritten for use with OS/VS2.

This appendix is meant to help you write efficient, helpful user programs.

Guaranteeing Cross-System Compatibility for Format Appendages

To make sure that an OS/MFT, OS/MVT, or OS/VS1 format appendage is upwardly compatible with OS/VS2 operation, reassemble the module containing the appendage. If your format appendage depends on specific fields in a trace record, be sure that it is not affected by differences in record format between systems.

Interfaces Between User Programs and EDIT

A user program works with the EDIT function of PRDMP in the following ways:

Gaining Control

Until EDIT calls them, user programs reside in SYS1.LINKLIB, or in data sets concatenated to SYS1.LINKLIB. Once your program is loaded into main storage, it remains there until EDIT processing is complete.

You name an exit routine in the EXIT parameter of the EDIT control statement. Your routine gets control every time EDIT reads an input trace record, and always completes its examination of the record before EDIT processes it.

You identify a format appendage in the GTRACE macro. Your format appendage gets control only when EDIT encounters a record that contains an FID field corresponding to the name of the format appendage. The appendage remains in main storage until deleted, For further information about GTRACE, see "User Trace Data Created With GTRACE," in Chapter 1: GTF.

End-of-File Indicator for User Exits

When the PRDMP EDIT facility passes control to your exit, an end-of-file indicator is passed in register two as follows:

```
0 - Normal processing (no end-of-file encountered)
4 - End-of-File or EDIT terminating condition encountered.
```

The end-of-file entrance into user exit routines occurs only when the EOF keyword is specified on the EDIT control statement. (For a description of the EDIT control statement, see Chapter 3: PRDMP). Such an entrance allows the exit routine to perform any necessary summarization and/or cleanup. If a return code of 4 is passed to the user exit, no input trace record exists; all input has already been processed.

Note: If the EOF keyword is not specified on the EDIT statement, there is no need to check for any return codes in register two. In this case, register two contains a zero for all invocations of your routine and your routine does not receive control at EDIT termination.

Using the Parameter List

When EDIT passes control to a user exit routine or a format appendage, register 1 contains the address of a parameter list. The contents of that parameter list, and its related fields are shown in Figure A-1. The exit routine or format appendage uses the parameter list (bytes 0 to 20) to find the record it is to process, determine how to process it, and decide where to put the processed record. (A format appendage also uses bytes 20 to 32.)

Input Record

As shown in Figure A-1, the first four bytes of the parameter list give the address of the input record. The two four-byte fields at offset 12 and 16 point to the event identifier (EID) field and the data area in the input record.

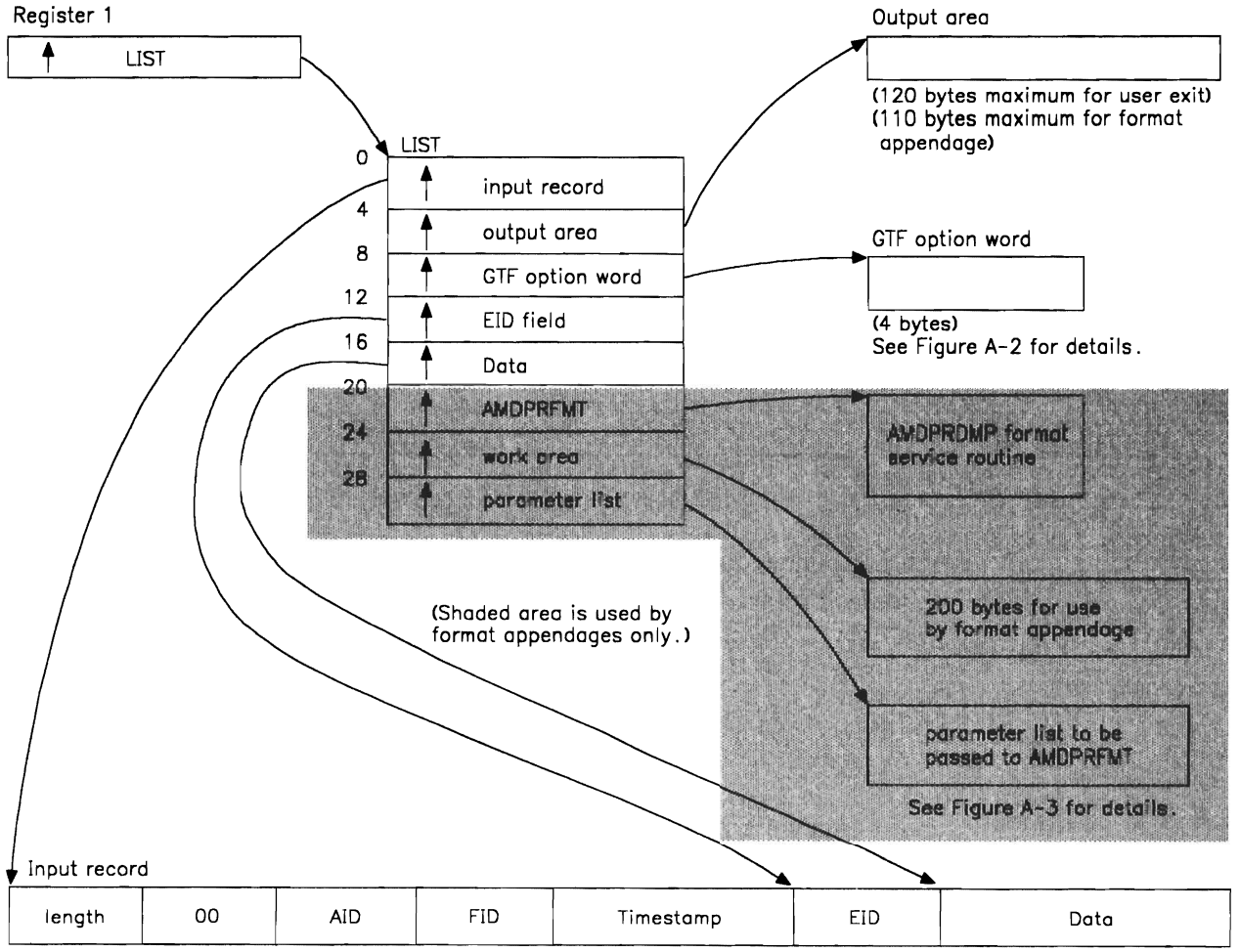


Figure A-1. EDIT Parameter List and Related Fields

For a description of the input record format after GTF processing, see Figure 1-10 and Figure 1-11 in Chapter 1: GTF (Generalized Trace Facility). For a description of the input record format prior to GTF processing, see the chapter on GTF in *OS/VS2 MVS Service Aids Logic*.

GTF Option Word

A four-byte field at offset 8 in the parameter list gives the address of the GTF option word, a four-byte table that summarizes the GTF options in effect when the input trace records were produced. Figure A-2 lists the contents of the GTF option word.

```
BYTES  BITS      OPTIONS IN EFFECT DURING TRACE

Byte 1 1... .... SYSM -- minimal tracing for system events
       .1.. .... SYSP -- maximum tracing, prompting requested
       ..1. .... SYS -- maximum tracing for system events
       ...1 .... USR -- all GTRACE-generated interrupts traced
       .... 1... TRC -- all GTF interrupts traced
       .... .1.. DSP -- all task-switches traced
       .... ..x. Reserved
       .... ...1 PCI -- program-controlled interrupts traced
Byte 2 1... .... SVC -- all SVC interrupts traced
       .1.. .... SVCP -- SVC interrupts selected by prompting
       ..1. .... SIO -- all SIO events traced
       ...1 .... SIOP -- SIO events selected by prompting
       .... 1... PI -- all program interrupts traced
       .... .1.. PIP -- program interrupts selected by prompting
       .... ..1. IO -- all I/O interrupts traced
       .... ...1 IOP -- I/O interrupts selected by prompting
Byte 3 1... .... EXT -- external interrupts traced
       .x.. .... Reserved
       ..1. .... SRM -- all system resource manager events traced
       ...1 .... RR -- all recovery termination events traced
       .... xxx. Reserved bits
       .... ...1 IO=SIO -- identical devices selected for IO & SIO
Byte 4 xxxx xxx. Reserved
       .... ...1 User timestamp requested
```

Figure A-2. Contents of GTF Option Word, Showing GTF Options in Effect during Trace

For more information about any of the GTF options, see Chapter 1: GTF (Generalized Trace Facility).

Writing a Reentrant Format Appendage

Because the user format appendage receives control from both PRDMP and ABDUMP, you can increase efficiency by writing reentrant user format appendages, and placing the routines in the SYS1.LPALIB system data set. This has the advantage of allowing simultaneous execution of the format appendage by both PRDMP and ABDUMP. It also removes the 10K size restriction placed on format appendages by PRDMP; because SYS1.LPALIB routines are not brought into PRDMP's address space, their size need not be restricted.

To be reentrant, word seven of the parameter list passed to the format appendage must address a 200-byte work area. (See Figure A-1.) This work area contains zeros on the first entrance to the format routines for each user trace record. The format appendage may then use this work area for any purpose. On any subsequent entry into the format appendage for that same user trace record, the work area remains intact to allow the format routine to reference any field set by prior calls. When a new trace record is obtained, the work area is zeroed. Figure A-6 shows a reentrant format appendage.

Format Service Routine Used by a Format Appendage

PRDMP's format service routine, AMDPRFMT, is available for use by a format appendage. This routine resides in the link pack area and is pointed to by word six of the parameter list passed to the format appendage. For a complete description of the use of AMDPRFMT, see "The Format Service Routine" in Appendix B: PRDMP User Exit Facility.

Word eight of the PRDMP parameter list contains the address of the parameter list to be passed to the format service routine. Figure A-3 shows the AMDPRFMT parameter list as initialized by ABDUMP and EDIT. See "PRDMP Parameter List" in Appendix B: PRDMP User Exit Facility for a description of the remaining fields of this parameter list.

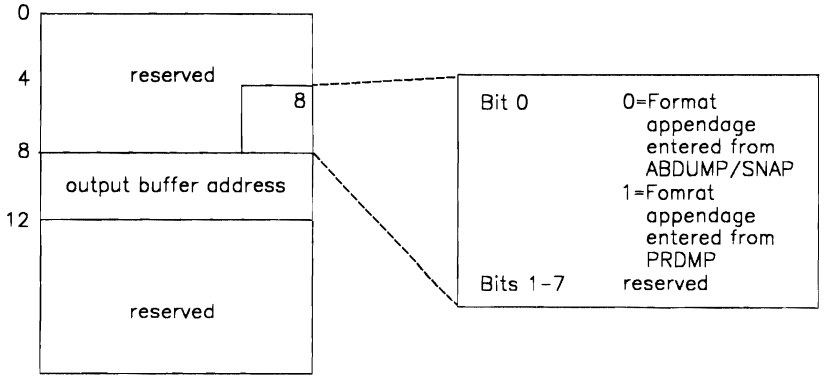


Figure A-3. Format Service Routine Parameter List

The third word of the PRDMP parameter list contains the address of the 120-byte output buffer into which the format service routine formats the data. The actual formatting of this data is directed by the format appendage through the use of format patterns. See "Format Patterns" in Appendix B: PRDMP User Exit Facility for a detailed description of the use of format patterns.

Using the Format Appendage

Note that your format appendage must not put any data into the first ten bytes of the output buffer. These ten bytes are reserved and overlaid by PRDMP or SNAP/ABDUMP before being printed. The maximum output line remains 110 bytes. Therefore, when your format routine assembles its format patterns, you must be sure that all output information (labels and/or data) begins at least ten bytes from the beginning of the output buffer.

Figure A-6 illustrates how a format appendage uses the PRDMP format routine to format its output.

Note that your format appendage does not have to use the PRDMP format routine. A user format appendage may choose to format its own data into the output buffer. In this case, use the output buffer address in the second word of the PRDMP parameter list. This buffer is 110 bytes long, and data can be placed in it starting at offset zero (the first ten bytes need not be preserved). The output buffer may be in a different location when the format appendage receives control again.

Returning to EDIT

Your program (the user-defined exit routine) must return to EDIT with one of the return codes listed below. If EDIT receives an invalid return code from your program, it takes action as specified by the ER subparameter of the PARM parameter of the EXEC statement that invokes PRDMP. This parameter, its values and their meanings are described in Chapter 5: PRDMP in the section "Job Control Language Statements."

Exit Routines Return Codes

An exit routine must return to EDIT with one of the following return codes:

Code	Meaning
0	EDIT should print the contents of the output area, clear the area, and return immediately to the exit routine. This allows the exit routine to print more than one line of output.
4	EDIT should print the contents of the output area and obtain the next logical record.
8	EDIT should format and print the trace record according to the selectivity specified in the EDIT control statement.
12	EDIT should obtain the next logical input trace record without printing the contents of the output buffer.

16	EDIT should print the contents of the output buffer but not re-invoke the exit routine, which is no longer needed.
20	EDIT should format and print the trace record according to the selectivity specified in the EDIT control statement, but not re-invoke the exit routine, which is no longer needed.
24	EDIT should terminate processing and return control to PRDMP so that the next PRDMP control statement can be processed.
28	EDIT should format and print this record as though no selectivity had been specified in the EDIT control statement.

Format Appendage Return Codes

A format appendage must return to EDIT or ABDUMP with one of the following return codes:

Code	Meaning
0	EDIT/ABDUMP should print the contents of the output buffer and return immediately to the format appendage.
4	EDIT/ABDUMP should print the contents of the output buffer and obtain the next logical input trace record.
8	EDIT/ABDUMP should obtain the next logical input trace record without printing the contents of the output buffer.
12	EDIT/ABDUMP should print this record in hexadecimal and obtain the next logical input trace record.

Handling Errors

EDIT is prepared to handle three types of errors: failures in finding or loading your program, invalid return codes, and program checks. Other types of errors and their consequences are discussed later in this appendix, in the section "Avoiding Unrecoverable Errors."

Errors in Finding or Loading Your Program

There are three probable reasons why EDIT might fail to find or load your program:

- Your program is incorrectly identified in the EXIT parameter of the EDIT control statement (for exit routines) or in the FID parameter of the GTRACE macro (for format appendages).
- Your program does not reside in the designated library.
- Your program is larger than 10K bytes and is not in the link pack area (LPA). EDIT does not load an exit routine or format appendage that exceeds this maximum; instead, it issues this message:

```
AMD229I  MODULE mod EXCEEDS 10K LIMIT
```

If, for one of these reasons, EDIT cannot find or load your exit routine or format appendage, it takes action as shown in Figure A-4.

Error Input Type	Exit Routine Not Found	Exit Routine Not Loaded	Format Appendage Not Found	Format Appendage Not Loaded
Dump	A	A	B/C	B/C
Trace Data Set	A	A	B	A

Action A: EDIT terminates processing, returns control to PRDMP and then obtains the next PRDMP control statement.

Action B: EDIT dumps the associated record in hexadecimal and obtains the next input trace record. Any subsequent records that have the same FID are dumped in hexadecimal.

Action C: ABDUMP dumps the associated record in hexadecimal and obtains the next input trace record. Any subsequent records that have the same FID are dumped in hexadecimal.

Figure A-4. EDIT/ABDUMP Actions in Response to Errors in Finding or Loading User Programs

Invalid Return Codes and Program Checks

EDIT's action in response to invalid return codes and program checks depends on the value for ER that you specify in the PARM parameter of the EXEC statement that invokes PRDMP. (**Note:** ER=3 should be specified whenever possible. All other values for ER result in issuance of the SPIE macro prior to every entrance to an EXIT routine or format appendage. The processing is extensive; it can adversely affect performance. For further explanation of the valid values for ER, see "Job Control Language Statements" in Chapter 3.

ABDUMP's action in response to invalid return codes and program checks is described as "Action C" in Figure A-4.

Avoiding Unrecoverable Errors

As shown in the previous sections, EDIT can recover from three kinds of errors: failures in finding or loading your program, invalid return codes, and program checks. EDIT cannot protect you, however, against errors that you may generate, for example, in performing I/O operations or issuing GETMAIN macro instructions. In fact, you should avoid issuing any SVCs in your program. Ordinarily this is not difficult, because EDIT allows your program to examine records, manipulate data, and request formatted output to be printed. If you must issue an SVC, EDIT permits you to do so; you should be prepared, however, for possibly unpredictable results when an error occurs during the SVC operation.

Another type of error arises when you invoke more format appendages than EDIT has room for: when EDIT needs more room to load a new format appendage, it deletes all previously loaded format appendages and starts loading format appendages again as needed.

Deletion of a format routine can be critical if the deleted program issues an OPEN, because the reinitialization that is necessary when the program is reloaded

can cause two DCBs to be open at the same time. (Note that you need not worry about deletion of an exit routine. EDIT provides a separate 10K block of storage for an exit routine, so that an exit routine is never be deleted until EDIT processing is terminated.)

One way to avoid running out of space for a format appendage is to make your program reentrant and place it in the link pack area (LPA). No storage is required and no LPA format appendages are deleted.

If your format appendage must issue a GETMAIN macro, be sure to specify region large enough to include the amount of storage needed. When you no longer need the extra storage, be sure to issue a FREEMAIN macro for all storage that you reserved for your own use. If you do not do this, and your format appendage is deleted, the storage you reserved remains allocated to you and thus will be unavailable to subsequent users.

Examples of Possible Errors

A few examples may further clarify the areas in which EDIT does not provide error recovery:

- Your program, known as module A, issues the LINK SVC for module B. A program check occurs in module B. EDIT attempts error recovery, because the error is a program check, but it knows nothing about module B. Therefore when it produces diagnostic information, it gives the entry point of module A as the entry point of the failing module and the attributes of the registers at the time of the program check to module A.

- Your program issues the OPEN SVC (SVC X'13') unsuccessfully and is posted with a system completion code of 213. EDIT cannot recover, so EDIT, your program and AMDPRDMP are all terminated.

- Your program opens a DCB. Before it can close the DCB, the program is deleted to make room for another user program. When the deleted program is reloaded, it creates a new DCB and opens it. Thus, there are two open DCBs with the same name in storage at the same time. The operating system does not tolerate this situation, so your program is abnormally terminated.

- Your program issues the SPIE SVC, thereby nullifying EDIT's SPIE routine. As a result, any program checks in your program that EDIT would normally handle now go through your own SPIE routine. Results could be unpredictable.

Sample User Exit

Figure A-5 shows a sample exit routine. This routine, named ABENDXIT, aids in diagnosis of abnormal termination condition in a particular job. It scans each input trace record, suppressing printing until it finds a record with the specified jobname. When it finds such a record, ABENDXIT signals PRDMP to print that record. All subsequent records are printed until ABENDXIT encounters an SVC 13 record for the specified jobname; then ABENDXIT instructs PRDMP to print that record and terminate.

```
*************************************************************************
*  ABENDXIT IS AN EDIT USER EXIT ROUTINE DESIGNED TO CONTROL PRINTING   *
*  OF ALL GTF RECORDS ASSOCIATED WITH A PROGRAM THAT HAS                *
*  PROGRAM CHECKED AND ABENDED                                          *
*************************************************************************
ABENDXIT CSECT
*   EQUATE STATEMENTS
FRSTREG       EQU  0
PARMREG       EQU  1
EIDREG        EQU  2
DATAREG       EQU  3
WORKREG       EQU  4
CHAINREG      EQU  9
BASE          EQU  12
SAVEPTR       EQU  13
RETPTR        EQU  14
CODEREG       EQU  15
         STM  RETPTR,BASE,12(SAVEPTR) STORE REGISTERS
         BALR BASE,0                  ESTABLISH ADDRESSABILITY
         USING *,BASE                 USING REGISTER 12
         ST   SAVEPTR,SAVE+4          BACKWARD CHAINING
         LA   CHAINREG,SAVE           MY SAVE AREA POINTER
         ST   CHAINREG,8(SAVEPTR)     FORWARD CHAINING
         LR   SAVEPTR,CHAINREG        REG 13 ADDRESSES SAVE AREA
         IMDMEDIT                     SYMBOLIC EID MACRO
 */* ************************************************************ */
 */*                      IMDMEDIT MAPPING MACRO                  */
 */* THE IMDMEDIT MACRO MAPS THE EID VALUES ASSOCIATED WITH IBM   */
 */* SYSTEM AND SUBSYSTEM EVENTS. THE STORAGE FOR ANY OR ALL OF THE */
 */* MAPPED VALUES MUST BE CONTAINED IN THE MODULE REFERENCING THE */
 */* DESIRED EIDS. IMDMEDIT IS DESIGNED TO BE USED BY IBM-SUPPLIED */
 */* FORMAT APPENDAGES, AND USER-SUPPLIED USER EXIT MODULES. ALL   */
 */* LABELS THAT ARE NOT PREFIXED WITH IMD ARE USED BY THE OTHER   */
 */* COMPONENTS OF GTF AND CORRESPOND TO THE NAMES USED IN THE HOOK */
 */* AND GTRACE MACROS.                                           */
 */* ************************************************************ */
     IMDMPCI     EQU     X'2100'       PCI I/O INTERRUPT
     IMDMSVC     EQU     X'1000'       SVC INTERRUPT
     IMDMDSP     EQU     X'0003'       DISPATCHER
     IMDMI01     EQU     X'5201'       I/O INTERRUPT
     IMDMI02     EQU     X'5200'       I/O INTERRUPT
     IMDMSIO     EQU     X'5100'       SIO OPERATION
     IMDMSSM1    EQU     0             SSM INTERRUPT
     IMDMSSM     EQU     0             OS SSM FOR COMPATIBILITY
     IMDMPIPG    EQU     0             PAGE FAULT PROGRAM INTERRUPT
     IMDMPI      EQU     X'6101'       PROGRAM INTERRUPT
     IMDMEXT     EQU     X'6201'       EXTERNAL INTERRUPT
```

Figure A-5 (Part 1 of 4). Sample Exit Routine

```
IMDMTINT    EQU     X'6200'       PFLIH
IMDMEOS     EQU     X'5101'       IOS
IMDMSTAE    EQU     X'4002'       RTM
IMDMFRR     EQU     X'4003'       RTM
IMDMSLSD    EQU     X'4004'       RTM/SLIP STD
IMDMSLSU    EQU     X'4005'       RTM/SLIP STD+USR
IMDMSLUR    EQU     X'4006'       RTM/SLIP USR
IMDMTP1     EQU     X'8100'       TPIOS
IMDMTP2     EQU     X'8200'       TPIOS
IMDMDSP1    EQU     X'0001'       DISPATCHER
IMDMDSP2    EQU     X'0002'       DISPATCHER
IMDMDSP3    EQU     X'0003'       DISPATCHER
IMDMDSP4    EQU     X'0004'       SVC EXIT PROLOG DISPATCH
IMDMSVCD    EQU     X'F100'       SVCDUMP
IMDMSRM     EQU     X'4001'       SRM
IDAAM01     EQU     X'EFF5'       VSAM
IMDMDMA1    EQU     X'EFFF'       OPEN/CLOSE/EOV
IECPCI      EQU     IMDMPCI       PCI I/O INTERRUPT
IEASVCH     EQU     IMDMSVC       SVC INTERRUPT
IECI01      EQU     IMDMI01       I/O INTERRUPT
IECI02      EQU     IMDMI02       I/O INTERRUPT
IECSIO      EQU     IMDMSIO       SIO OPERATION
IEAPINT     EQU     IMDMPI        PROGRAM INTERRUPT
IEAEINT     EQU     IMDMEXT       EXTERNAL INTERRUPT
IEATINT     EQU     IMDMTINT      PFLIH
IECEOS      EQU     IMDMEOS       IOS
IEASTAE     EQU     IMDMSTAE      RTM
IEAFRR      EQU     IMDMDFRR      RTM
IEAVSLSD    EQU     X'4004'       RTM/SLIP STD
IEAVSLSU    EQU     X'4005'       RTM/SLIP STD+USR
IEAVSLUR    EQU     X'4006'       RTM/SLIP USR
ISPTIO1     EQU     IMDMTP1       TPIOS
ISPTIO2     EQU     IMDMTP2       TPIOS
IEADISP1    EQU     IMDMDSP1      DISPATCHER
IEADISP2    EQU     IMDMDSP2      DISPATCHER
IEADISP3    EQU     IMDMDSP3      DISPATCHER
IEADISP4    EQU     IMDMDSP4      EXIT PROLOG DISPATCH
IEASVCD     EQU     IMDMSVCD      SVCDUMP
IRASRM      EQU     IMDMSRM       SRM
IEAABOF     EQU     X'F200'       ABEND/SNAP-NOT USED BY EDIT
IGGSP169    EQU     X'EFFE'       SAM/PAM/DAM
IGGSP451    EQU     X'EFFD'       SAM/PAM/DAM
IGGSP251    EQU     X'EFFC'       SAM/PAM/DAM
IGGSP145    EQU     X'EFFB'       SAM/PAM/DAM
IGGSP239    EQU     X'EFFA'       SAM/PAM/DAM
IGGSP235    EQU     X'EFF9'       SAM/PAM/DAM
IGGSP119    EQU     X'EFF8'       SAM/PAM/DAM
IGGSP215    EQU     X'EFF7'       SAM/PAM/DAM
IGGSP112    EQU     X'EFF6'       SAM/PAM/DAM
IGGSP008    EQU     X'EFF4'       SAM/PAM/DAM
IGGSP002    EQU     X'EFF3'       SAM/PAM/DAM
ISTLNEID    EQU     X'EFF2'       VTAM LINE TRACE
ISTCLEID    EQU     X'EFF|'       VTAM CONTROL LAYER
ISTRPEID    EQU     X'EFF0'       VTAM BUFFER POOL ALLOCATION
ISTTPEID    EQU     X'EFEF'       VTAM TPIOS BUFFER
ISTVIEID    EQU     X'EFE1'       VTAM INTERNAL
ISTTHEID    EQU     X'EFE2'       VTAM RESERVED
```

Figure A-5 (Part 2 of 4). Sample Exit Routine

```
ISTTREID    EQU     X'EFE3'       VTAM RESERVED
ISTTDEID    EQU     X'EFE4'       VTAM RESERVED
ISTVIEID    EQU     X'EFE1'       VTAM INTERNAL
IMDGPD01    EQU     X'EFAF'       RESERVED
IMDGPD02    EQU     X'EFB0'       RESERVED
IMDGPD03    EQU     X'EFB1'       RESERVED
IMDGPD04    EQU     X'EFB2'       RESERVED
IMDGPD05    EQU     X'EFB3'       RESERVED
IMDGPD06    EQU     X'EFB4'       RESERVED
IMDGPD07    EQU     X'EFB5'       RESERVED
IMDGPD08    EQU     X'EFB6'       RESERVED
IMDGPD09    EQU     X'EFB7'       RESERVED
IMDGPD10    EQU     X'EFB8'       RESERVED
IMDGPD11    EQU     X'EFB9'       RESERVED
IMDGPD12    EQU     X'EFBA'       RESERVED
IMDGPD13    EQU     X'EFBB'       RESERVED
IMDGPD14    EQU     X'EFBC'       RESERVED
IMDGPD15    EQU     X'EFBD'       RESERVED
IMDGPD16    EQU     X'EFBE'       RESERVED
IMDGPD17    EQU     X'EFBF'       RESERVED
IMDGPD18    EQU     X'EFC0'       RESERVED
IMDGPD19    EQU     X'EFC|'       RESERVED
IMDGPD20    EQU     X'EFC2'       RESERVED
IMDGPD21    EQU     X'EFC3'       RESERVED
IMDGPD22    EQU     X'EFC4'       RESERVED
IMDGPD23    EQU     X'EFC5'       RESERVED
IMDGPD24    EQU     X'EFC6'       RESERVED
IMDGPD25    EQU     X'EFC7'       RESERVED
IMDGPD26    EQU     X'EFC8'       RESERVED
IMDGPD27    EQU     X'EFC9'       RESERVED
IMDGPD28    EQU     X'EFCA'       RESERVED
IMDGPD29    EQU     X'EFCB'       RESERVED
IMDGPD30    EQU     X'EFCC'       RESERVED
IMDGPD31    EQU     X'EFCD'       RESERVED
IMDGPD32    EQU     X'EFCE'       RESERVED
IMDGPD33    EQU     X'EFCF'       RESERVED
IMDGPD34    EQU     X'EFD0'       RESERVED
IMDGPD35    EQU     X'EFD1'       RESERVED
IMDGPD36    EQU     X'EFD2'       RESERVED
IMDGPD37    EQU     X'EFD3'       RESERVED
IMDGPD38    EQU     X'EFD4'       RESERVED
IMDGPD39    EQU     X'EFD5'       RESERVED
IMDGPD40    EQU     X'EFD6'       RESERVED
IMDGPD41    EQU     X'EFD7'       RESERVED
IMDGPD42    EQU     X'EFD8'       RESERVED
IMDGPD43    EQU     X'EFD9'       RESERVED
IMDGPD44    EQU     X'EFDA'       RESERVED
IMDGPD45    EQU     X'EFDB'       RESERVED
IMDGPD46    EQU     X'EFDC'       RESERVED
IMDGPD47    EQU     X'EFDD'       RESERVED
IMDGPD48    EQU     X'EFDE'       RESERVED
IMDGPD49    EQU     X'EFDF'       RESERVED
IMDGPD50    EQU     X'EFE0'       RESERVED
            MEND
    TM      TERMSW,X'01'          Q/HAS TERMINATION BEEN REQSTD
    BC      1,FINISH              YES, TELL EDIT TO TERMINATE
    L       EIDREG,12(PARMREG)    GET POINTER TO EID
    L       DATAREG,16(PARMREG)   GET POINTER TO DATA(JOBNAME)
```

Figure A-5 (Part 3 of 4). Sample Exit Routine

```
         TM      PRINTSW,X'01'          Q/HAS JOBN ALREADY BEEN FOUND
         BC      1,PRINTALL             YES, SO PRINT THIS RECORD
         LA      WORKREG,0              GET ZERO CONSTANT
         C       WORKREG,ECB1           Q/HAS THIS ECB BEEN POSTED
         BC      7,MYJOBLAB             YES, CHECK IF JOBN FOUND
         WTOR    'SPECIFY 8-CHARACTER JOBNAME OF ABENDING PROGRAM',
                 MYJOBN,8,ECB1
         WAIT    ECB=ECB1
         LA      WORKREG,MYJOBN         ADDRESS OF JOBNAME SELECTED
         OC      0(8,WORKREG),BLANKS    CONVERT LOWER-CASE CHARS TO
*                                        UPPER CASE
MYJOBLAB CLC     6(8,DATAREG),MYJOBN    Q/IS THIS MY JOBNAME
         BC      7,NOPRINT              NO -- JUST RETURN
*  ONCE JOBNAME FOUND, SET SWITCH AND PRINT ALL RECORDS UNTIL
*  ENCOUNTER AN SVC 13 (ABEND) CONTAINING THIS JOBNAME
         OI      PRINTSW,X'001'         TURN ON JOBNAME FOUND SWITCH
PRINTALL CLC 6(8,DATAREG),MYJOBN  Q/ IS THIS AN SVC RECORD
         BC      7,PRINTREC             NO, SO PRINT AND CONTINUE
         CLI     15(DATAREG),X'0D'      Q/IS THIS AN SVC 13 (ABEND)
         BC      7,PRINTREC             NO, SO PRINT AND CONTINUE
         CLC     0(8,DATAREG),MYJOBN    Q/IS THIS MY JOBNAME
         BC      7,PRINTREC             NO, SO PRINT AND CONTINUE
EXIT     0I      TERMSW,X'01'           INDICATE THAT THIS IS LAST
*                                        RECORD TO BE PRINTED
PRINTREC LA      CODEREG,8              FORMAT AND PRINT THIS RECORD
RETURN   L       SAVEPTR,4(SAVEPTR)     RESTORE SAVE AREA POINTER
         L       RETPTR,12(SAVEPTR)     RESTORE REGISTER 14
         LM      FRSTREG,BASE,20(SAVEPTR) RESTORE OTHER REGS EXCEPT 15
         BCR     15,RETPTR              RETURN TO EDIT
FINISH   LA      CODEREG,24             TERMINATE EDIT PROCESSING
*                                        SINCE SVC 13 WAS LAST RECORD
         B       RETURN                 RESTORE REGISTERS AND RETURN
NOPRINT  LA      CODEREG,12             IGNORE RECORD
         B       RETURN                 RESTORE REGISTERS AND RETURN
SAVE     DC      18F'0'                 SAVE AREA
SVCEID   DC      AL2(IMDMSVC)           ESTABLISH REAL AREA FOR
*                                        EID FROM IMDMEDIT MAP MACRO
TERMSW   DC      X'00'                  INDICATION TO REQUEST TERM
PRINTSW  DC      X'00'                  JOBN FOUND, SO PRINT REC IND
ECB1     DC      F'0'                   FOR POST
MYJOBN   DC      C'        '            PLACE FOR OPR TO PUT JOBNAME
BLANKS   DC      C'        '            TO CONVERT LOWER TO UPPER CASE
         END
/*
```

Figure A-5 (Part 4 of 4). Sample Exit Routine

Some instructions in the sample exit routine require special attention. These are shaded-in in Figure A-5, and they are discussed below.

IMDMEDIT

This mapping macro expands, as shown, into a list of equate statements that supply symbolic names for the event identifiers (EIDs). You should use the symbolic name in your program rather than the hexadecimal value; this is your protection against program failure, if for any reason the EID values are later changed.

L EIDREG,12(PARMREG)
L DATAREG,16(PARMREG)
These two instructions access the EDIT parameter list. (See Figure A-1.)

WTOR 'SPECIFY 8-CHARACTER JOBNAME OF ABENDING PROGRAM', MYJOBN,8,ECB1
This instruction requests information that cannot be obtained from the EDIT parameter list. You can use a WTOR to request any information that the operator is likely to have, such as the EDIT options in effect.

Note: If you issue an SVC in your program, you risk abnormal termination when an error occurs during the SVC operation. For more information about this point, see "Avoiding Unrecoverable Errors" earlier in this chapter.

SVCEID DC AL2(IMDMSVC)
This establishes a storage location for the value equated to IMDMSVC in the expansion of the IMDMEDIT mapping macro.

Sample Format Appendage

Figure A-6 shows how to use the EDIT parameter list and how to handle multiple EIDs. It consists of excerpts from a sample format appendage named AMDUSR01, which formats three different types of user records. For each record AMDUSR01 produces two lines of output. The first line varies according to the record type. The second line is the same for all records.

```
*********************************************************************
* AMDURS01 IS AN EDIT USER FORMAT APPENDAGE MODULE THAT PROCESSES     *
* THREE DIFFERENT TYPES OF INPUT RECORDS, THUS, THREE DIFFERENT EIDS  *
* LINE ONE OF THE FORMATTED OUTPUT VARIES ACCORDING TO THE EID.  LINE *
* TWO OF THE FORMATTED OUTPUT IS THE SAME FOR ALL EIDS, AND IS        *
* PRODUCED IN COMMON CODE.                                            *
*********************************************************************
AMDUSR01 CSECT
*  EQUATE STATEMENTS
FRSTREG  EQU  0
PARMREG  EQU  1
EIDREG   EQU  2
DATAREG  EQU  3
CHAINREG EQU  9
BASE     EQU  12
SAVEPTR  EQU  13
RETPTR   EQU  14
CODEREG  EQU  15
         STM  RETPTR,BASE,12(SAVEPTR)  STORE REGISTERS
         BALR BASE,0                   ESTABLISH ADDRESSABILITY
         USING *,BASE                  USING REGISTER 12
         ST   SAVEPTR,SAVE+4           BACKWARD CHAINING
         LA   CHAINREG,SAVE            MY SAVE AREA POINTER
         ST   CHAINREG,8(SAVEPTR)      FORWARD CHAINING
         LR   SAVEPTR,CHAINREG         REG 13 ADDRESSES SAVE AREA
         L    EIDREG,12(PARMREG)       GET POINTER TO EID
         L    DATAREG,16(PARMREG)      GET POINTER TO FIRST LINE DATA
         TM   SWITCH,X'01'             Q/ HAS FIRST LINE BEEN OUTPUTTED
         BC   1,LINETWO                YES, BRANCH TO FORMAT LINE TWO
*                                      WHICH IS COMMON TO ALL 3 EID RTNS
         CLC  0(2,EIDREG),EID1         NO--Q/IS THIS A RECORD WITH EID1
         BC   8,RTN1                   YES--FORMAT LINE ONE
         CLC  0(2,EIDREG),EID2         Q/IS THIS A RECORD WITH EID2
         BC   8,RTN2                   YES--FORMAT LINE ONE
         CLC  0(2,EIDREG),EID3         Q/IS THIS A RECORD WITH EID3
         BC   8,RTN3                   YES--FORMAT LINE ONE
         LA   CODEREG,8                NO--IF NONE OF THESE EIDS, IGNORE
         B    RETURN                   REC, RESTORE REGS, AND RETURN
     .
     .
     .
RTN1
     .
     .
     .
         B    ZEROCODE                 SET ZERO RETURN CODE
RTN2
     .
     .
     .
         B    ZEROCODE                 SET ZERO RETURN CODE
RTN3
     .
     .
```

Figure A-6 (Part 1 of 2). Sample Format Appendage

```
     .
     .
ZEROCODE OI    SWITCH,X'01'              FIRST LINE COMPLETE INDICATOR
         SR    CODEREG,CODEREG           OUTPUT THIS LINE AND RETURN
*                                        DIRECTLY TO THIS FORMAT APPENDAGE
         B     RETURN                    RESTORE REGISTERS AND RETURN
LINETWO
     .
     .
     .
         NI    SWITCH,X'FE'              TURN OFF LINE 2 INDICATOR
         LA    CODEREG,4                 OUTPUT THIS LINE--COMPLETE
RETURN   L     SAVEPTR,4(SAVEPTR)        RESTORE SAVE AREA POINTER
         L     RETPTR,12(SAVEPTR)        RESTORE REGISTER 14
         LM    FRSTREG,BASE,20(SAVEPTR)  RESTORE OTHER REGS EXCEPT 15
         BCR   15,RETPTR                 RETURN TO EDIT
SAVE     DC    18F'0'                    REGISTER SAVE AREA
SWITCH   DC    X'00'                     READY FOR LINE TWO SWITCH
EID1     DC    X'E001'                   EID1
EID2     DC    X'E002'                   EID2
EID3     DC    X'E003'                   EID3
     .
     .
     .
         END
/*
```

Figure A-6 (Part 2 of 2). Sample Format Appendage

Sample Reentrant Format Appendage, Using the Format Service Routine

Figure A-7 shows how a reentrant format appendage might be written. It handles the same EIDs as the previous example (Figure A-6). However, this format appendage uses the PRDMP format service routine to format its output line. Note that the reentrancy and the use of the PRDMP format routine are not related; a non-reentrant format appendage could also use the PRDMP format routine. See the topic "Format Service Routine" in Appendix B: PRDMP User Exit Facility.

```
*********************************************************************
* AMDUSR02 IS AN USER FORMAT APPENDAGE WHICH IS REENTRANT TO ALLOW    *
* SIMULTANEOUS EXECUTION UNDER BOTH PRDMP EDIT AND ABDUMP/SNAP. It    *
* PROCESSES THREE DIFFERENT TYPES OF INPUT RECORDS, DENOTED BY THE    *
* THREE DIFFERENT EIDS. THE FIRST FORMATTED LINE VARIES ACCORDING     *
* TO THE EID. THE SECOND FORMATTED LINE IS THE SAME FORMAT FOR ALL    *
* THREE EIDS. ALL OUTPUT LINES ARE FORMATTED USING THE PRDMP FORMAT   *
* SERVICE ROUTINE.                                                    *
*********************************************************************
AMDUSR02 CSECT
FRSTREG  EQU  0
PARMREG  EQU  1
EIDREG   EQU  2
DATAREG  EQU  3
TEMPSAV  EQU  4
PARMSAV  EQU  5
TEMPREG  EQU  6
CHAINREG EQU  9
BASE     EQU  12
SAVEPTR  EQU  13
RETPTR   EQU  14
CODEREG  EQU  15
         STM  RETPTR,BASE,12(SAVEPTR)  SAVE CALLERS REGISTERS
         BALR BASE,0                ADDRESSABILITY FOR CODE
         USING *,BASE
         LR   PARMSAV,PARMREG       SAVE FORMAT APPENDAGE PARAMLIST ADDR
         L    TEMPSAV,24(PARMSAV)   LOCATE 200-BYTE WORK AREA
         USING WORKSAV,TEMPSAVE     BASE REGISTER FOR WORK/SAVE AREA
*                                   DSECT, PUTTING ALL DYNAMIC AREAS IN
*                                   PASSED WORK AREA FOR REENTRANCYS IN
         ST   SAVEPTR,SAVE+4        BACK CHAIN, HIS TO MINE
         LA   CHAINREG,SAVE         LOCATE MY SAVE AREA IN WORK AREA
         ST   CHAINREG,8(SAVEPTR)   FORWARD CHAIN,MINE TO HIS
         LR   SAVEPTR,CHAINREG      POINT R13 TO MINE
         L    EIDREG,12(PARMSAV)    POINTER TO EID FIELD OF RECORD
         L    DATAREG,16(PARMSAV)   POINTER TO FIRST DATA BYTE
         TM   SWITCH,X'80'          Q. HAS FIRST LINE BEEN PROCESSED FOR
*                                   THIS INPUT RECORD - SWITCH WILL BE
*                                   ZERO FOR FIRST ENTRY FOR EACH INPUT
*                                   RECORD
         BO   LINETWO               IF SWITCH IS ON, GO FORMAT 2ND LINE,
*                                   WHICH IS COMMON TO ALL THREE EIDS
*********************************************************************
* THE SWITCH WAS ZERO - THE INPUT RECORD CONTAINS A NEW TRACE         *
* RECORD. DETERMINE WHICH EID IS IN THE RECORD, AND BRANCH TO THE     *
* PROPER ROUTINE TO FORMAT THE RECORD.  IF THE RECORD DOES NOT        *
* CONTAIN ANY OF THE THREE EIDS, RETURN A CODE OF 8 TO THE CALLER     *
* TO IGNORE THIS RECORD.                                              *
*********************************************************************
         CLC  0(2,EIDREG),EID1      Q. IS THIS EID1
         BE   RTN1                  IF SO, GO FORMAT IT
         CLC  0(2,EIDREG),EID2      Q. IS THIS EID2
         BE   RTN2                  IF SO, GO FORMAT IT
         CLC  0(2,EIDREG),EID3      Q. IS THIS EID3
         BE   RTN3                  IF SO, FORMAT IT
         LA   CODEREG,8             IF NONE OF THE ABOVE, SET THE
*                                   PROPER RETURN CODE TO IGNORE
         B    RETURN                GO TO COMMON EXIT CODE
```

Figure A-7 (Part 1 of 6). Sample Reentrant Format Appendage

```
*********************************************************************
*        THE FOLLOWING ROUTINE USES THE FORMAT SERVICE ROUTINE TO   *
*        FORMAT THE FIRST LINE FOR EID1. THE OUTPUT FOR THIS LINE IS*
*        TO BE IN THE FOLLOWING FORM:                               *
*                                                                   *
*        LAB1  XXXX    LAB2  CCCCCCCC    LAB3  XXXXXXXX   LAB4  XXXX*
*                                                                   *
*              X=HEXADECIMAL DATA, C=EBCDIC DATA                    *
*                                                                   *
*        THE INPUT DATA CONSISTS OF XXXXCCCCCCCCXXXXXXXXXXXX, AND   *
* MUST BE FORMATTED AS ABOVE, WITH THE LABELS INSERTED. THE DATA    *
* REPRESENTED BY X MUST BE CONVERTED TO PRINTABLE HEXADECIMAL.      *
*********************************************************************
RTN1     MVC  FORMAT(28),FMT1     MOVE FORMAT PATTERN TO WORK AREA -
*                                 DATA ADDRESS MUST BE FILLED IN DUR-
*                                 ING EXECUTION, AND THEREFORE TO
*                                 REMAIN REENTRANT, PATTERN MUST BE
*                                 COPIED.
         ST   DATAREG,FMTDATA     STORE BEGINNING DATA ADDRESS INTO
*                                 PATTERN
         LA   FRSTREG,FORMAT      R0 RUST HAVE FORMAT PATTERNS ADR TO
*                                 BE USED
         L    PARMREG,28(PARMSAV) R1 MUST POINT TO FORMAT ROUTINE's
*                                 PARAMETER LIST - THIS HAS ALREADY
*                                 BEEN INITIALIZED BY OUR CALLER
         L    CODEREG,20(PARMSAV) R15 POINTS TO FORMAT SERVICE ROUTINE
         BALR RETPTR,CODEREG      CALL FORMAT SERVICE ROUTINE
         .
         .
         .
         B    ZEROCODE            SET ZERO RETURN CODE
*********************************************************************
*        THIS ROUTINE USES THE FORMAT SERVICE ROUTINE TO FORMAT THE  *
* FIRST LINE FOR EID2. THE OUTPUT IS TO BE IN THE FOLLOWING FORM:   *
*                                                                   *
*        LAB5  XXXXXXXX  XXXXXXXX  XXXXXXXX  XXXXXXXX              *
*                                                                   *
*        THE INPUT DATA CONSISTS OF A 16 BYTE HEXADECIMAL DATA STRING*
* IT IS TO BE CONVERTED TO PRINTABLE HEXADECIMAL                    *
*********************************************************************
RTN2     MVC  FORMAT(28),FMT2     COPY PATTERN TO WORK AREA
         ST   DATAREG,FMDATA      POINT PATTERN TO INPUT DATA
         LA   FRSTREG,FORMAT      R0 POINTS TO FORMAT PATTERN
         L    PARMREG,28(PARMSAV) R1 POINTS TO PRDMP PARAMETER LIST
         L    PARMREG,20(PARMSAV) R15 POINTS TO FORMAT ROUTINES EP
         BALR RETPRT,CODEREGA     CALL FORMAT SERVICE ROUTINE
         .
         .
         .
         B    ZEROCODE            SET ZERO RETURN CODE
```

Figure A-7 (Part 2 of 6). Sample Reentrant Format Appendage

```
*************************************************************************
*        THIS ROUTINE USES THE FORMAT SERVICE ROUTINE TO FORMAT THE     *
* FIRST LINE FOR EID3. THE OUTPUT IS TO BE IN THE FOLLOWING FORM:       *
*                                                                       *
*        LAB6  CCCCCCCC   LAB7 CCCCCCCC    LAB8  XXXXXXXX XXXXXXXX     *
*                                                                       *
*        HOWEVER, THE INPUT FOR THIS EID IS AS FOLLOWS:                 *
*                                                                       *
*              CCCCCCCCXXXXXXXXXXXXXXXXCCCCCCCC                         *
*                                                                       *
*        NOTE THAT THE INPUT DATA IS NOT IN THE SAME SEQUENCE AS IT     *
* APPEARS IN THE OUTPUT. THE FORMAT PATTERNS AT FMT3 ALLOWS FOR THIS    *
* AGAIN, C+EBCDIC DATA,X=HEXADECIMAL DATA                               *
*************************************************************************
RTN3     MVC  FORMAT(36),FMT3     COPY PATTERN TO WORK AREA
         ST   DATAREG,FMDATA      STORE FIRST DATA AREA POINTER
         LA   TEMPREG,24(DATAREG) FIND LOCATION OF SECOND DATA
         ST   TEMPREG,FMTDATA1    STORE INTO FORMAT PATTERN
         LA   TEMPREG,8(DATAREG)  FIND LOCATION OF THIRD DATA
         ST   TEMPREG,FMTDATA2    STORE INTO PATTERN
         LA   FRSTREG,FORMAT      R0 POINTS TO FORMAT PATTERN
         L    PARMREG,28(PARMSAV) R1 POINTS TO PRDMP PARAMETER LIST
         L    CODEREG,20(PARMSAV) R15 POINTS TO FORMAT ROUTINE
         BALR RETPTR,CODEREG      CALL FORMAT SERVICE ROUTINE
         .
         .
         .
ZEROCODE OI   SWITCH,X'80'        INDICATE FIRST LINE HAS BEEN DONE
         SR   CODEREG,CODEREG     SET RETURN CODE TO CALLER TO
*                                  OUTPUT THIS LINE AND RETURN TO THIS
*                                  FORMAT APPENDAGE WITHOUT GETTING
*                                  ANOTHER INPUT RECORD
RETURN   L    SAVEPTR,4(SAVEPTR)  RESTORE HIS SAVE AREA
         L    RETPTR,12(SAVEPTR)  RESTORE RETURN ADDRESS
         LM   FRSTREG,BASE,20(SAVEPTR) RESTORE REMAINING REGISTERS
         BR   RETPTR              RETURN TO CALLER
*************************************************************************
*        THIS ROUTINE FORMATS THE SECOND LINE FOR EACH TRACE RECORD.    *
* THE FORMAT IS AS FOLLOWS:                                             *
*                                                                       *
*        COMMONL2  CCCCCCCCCCCCCCCC                                     *
*                                                                       *
*        THE INPUT FOR THIS LINE IS ALWAYS 32 BYTES FROM THE START OF   *
* THE INPUT DATA, REGARDLESS OF THE EID. ITS IN THE SAME FORM AS THE    *
* OUTPUT                                                                *
*************************************************************************
LINETWO  MVC  FORMAT(16),COMFMT2  MOVE PATTERN TO WORK AREA
         LA   TEMPREG,32(DATAREG) FIND DATA
         ST   TEMPREG,FMTDATA     POINT PATTERN TO DATA
         LA   FRSTREG,FORMAT      R0 POINTS TO FORMAT PATTERN
         L    PARMREG,28(PARMSAV) R1 POINTS TO PRDMP PARAMETER LIST
         L    PARMREG,20(PARMSAV) R15 POINTS TO FORMAT ROUTINE
         BALR 14,15               CALL FORMAT SERVICE ROUTINE
         .
         .
         .
```

Figure A-7 (Part 3 of 6). Sample Reentrant Format Appendage

```
         LA    CODEREG,4           SET RETURN CODE TO TELL CALLER TO
*                                  OUTPUT THIS LINE AND GET NEW INPUT
*                                  RECORD. WORKAREA WILL BE ZERO WHEN
*                                  ENTRY IS MADE TO THIS ROUTINE FOR
*                                  THE NEW RECORD, SO NO NEED TO TURN
*                                  OFF SWITCH
         B     RETURN
WORKSAV  DSECT
SAVE     DS    F18                 REGISTER SAVE AREA
SWITCH   DS    XL1                 LINE 1 OR LINE 2 INDICATOR
FORMAT   DS    OF                  AREA INTO WHICH FORMAT PATTERN IS
*                                  COPIED
FMTFLGS  DS    XL1                 FLAG FIELD
FMTLNGS  DS    XL1                 LABEL AND/OR DATA LENGTHS
FMTLBOFF DS    XL1                 LABEL OFFSET
FMTDAOFF DS    XL1                 DATA OFFSET
FMTLBADR DS    F                   ADDRESS OF LABELS
FMTDATA  DS    F                   ADDRESS OF DATA TO BE FORMATTED
         DS    F                   FILLER FOR FLAGS,ETC
FMTDATA1 DS    F                   2ND DATA ADDRESS
         DS    F                   FILLER
FMTDATA2 DS    F                   3RD DATA ADDRESS
AMDUSR02 CSECT                     RESTART MAIN CSECT
EID1     DC    X'E001'             EID1
EID2     DC    X'E002'             EID2
EID3     DC    X'E003'             EID3
LABELS1  DC    C'LAB1LAB2LAB3LAB4' LABELS FOR LINE 1, EID1
LABELS2  DC    C'LAB5'             LABEL FOR LINE 1, EID2
LABELS3  DC    C'LAB6LAB7LAB8'     LABELS FOR LINE 1, EID3
COMLABEL DC    C'COMMONL2'         LABEL FOR COMMON LINE 2
FMT1     DS    0F                  START OF FORMAT PATTERN FOR LINE 1
         DC    XL1'1F'             SET FLAGS TO INDICATE:
*                                      1.DATA IS TO BE CONVERTED
*                                      2.ADDRESS OF LABEL IS SUPPLIED
*                                        AND A LABEL IS TO BE PLACED IN
*                                        OUTPUT BUFFER
*                                      3.ADDRESS OF DATA IS SUPPLIED
*                                        AND DATA IS TO BE PLACED IN
*                                        OUTPUT BUFFER
         DC    XL1'31'             LABEL IS 4 BYTES, DATA 2
         DC    FL1'10'             LABEL STARTS AT OFFSET 10 IN OUTPUT
*                                     BUFFER - FIRST 10 BYTES WILL BE
*                                     OVERLAYED BY EDIT/ABDUMP/SNAP
         DC    FL1'16'             DATA BEGINS AT OFFSET 16 IN OUTPUT
         DC    A(LABELS1)          CHARACTER STRING OF LABELS TO BE
*                                     USED
         DC    A(0)                DATA ADDRESS FOUND DYNAMICALLY
         DC    XL1'35'             SET FLAGS TO INDICATE:
*                                     1.DATA NOT TO BE CONVERTED
*                                     2.NO LABEL OR DATA ADDRESS IS
*                                       SUPPLIED, BUT THEY ARE TO BE
*                                       INCLUDED IN THE OUTPUT. THEY
*                                       ARE FOUND BY INCREMENTING ADDR-
*                                       ESS IN PRECEDING PATTERN
```

Figure A-7 (Part 4 of 6). Sample Reentrant Format Appendage

```
         DC    XL1'37'             4 BYTE LABEL,8 BYTE DATA
         DC    FL1'24'             LABEL AT OFFSET 24
         DC    FL1'30'             DATA AT OFFSET 30
         DC    XL1'15'             SET FLAGS AS ABOVE, BUT CONVERT
*                                     DATA TO PRINTABLE HEXADECIMAL
         DC    XL1'33'             4 BYTE LABEL, 4 BYTE DATA
         DC    FL1'42'             LABEL TO APPEAR 42 BYTES FROM BUFFER
*                                     START
         DC    FL1'48'             DATA AT OFFSET 48 FROM BUFFER START
         DC    XL1'15',XL1'31',FL1'60',FL1'66' FORMAT PATTERN FOR LAB4
*                                           AND ITS DATA
         DC    F'0'                INDICATES END OF THIS FORMAT CALL
FMT2     DS    0F                  FORMAT PATTERNS FOR LINE 1 - EID2
         DC    XL1'1F'             INDICATE BOTH LABEL AND DATA ADD-
*                                     RESSES FOLLOW
         DC    XL1'33'             LABEL IS 4 BYTES, DATA 4 BYTES
         DC    FL1'10'             LABEL IN COLUMN 10 AGAIN
         DC    FL1'16'             DATA TO GO IN COLUMN 16
         DC    A(LABELS2)          ADDRESS OF LAB5
         DC    A(0)                DATA ADDRESS FILLED IN DURING
*                                  EXECUTION
         DC    XL1'14'             SET FLAGS TO INDICATE ONLY CONVERTED
*                                     DATA IS TO BE PLACED IN BUFFER
         DC    XL1'03'             NO LABEL,4 BYTES OF DATA
         DC    FL1'0'              NO LABEL
         DC    FL1'26'             DATA TO GO INTO OFFSET 26
         DC    XL1'14',XL1'03',FL1'0',FL1'36'  NEXT DATA AT 36
         DC    XL1'14',XK1'03',FL1'0',FL1'46'  NEXT DATA AT 46
         DC    F'0'                END OF PATTERN
FMT3     DS    0F                  START OF LINE 1 - EID3
         DC    XL1'3F'             LABELS AND DATA ADDRESSES FOLLOW,
*                                     THIS DATA NOT TO BE CONVERTED
         DC    XL1'37'             4 BYTE LABEL, 8 BYTE DATA
         DC    FL1'10'             LABEL STARTS OFFSET 10
         DC    FL1'16'             DATA TO START OFFSET 16
         DC    A(LABELS3)          ADDRESS OF LABEL STRING
         DC    A(0)                ADDRESS OF FIRST DATA TO BE FOR-
*                                    MATTED, TO BE FILLED IN LATER
         DC    XL1'3D'             SET FLAGS TO INDICATE NO CONVERSION,
*                                    THAT THERE IS A NEW DATA ADDRESS,
*                                    AND THAT A LABEL IS TO BE USED
*                                    FROM THE PREVIOUS PATTERN
         DC    XL1'37'             4 BYTE LABEL, 8 BYTE DATA
         DC    FL1'28'             LABEL AT OFFSET 28
         DC    FL1'34'             DATA AT OFFSET 34
         DC    A(0)                ADDRESS OF 2nd SECTION OF DATA, TO
*                                    BE FILLED IN DURING EXECUTION
         DC    XL1'1D'             SET FLAGS TO INDICATE DATA TO BE
*                                    CONVERTED, AND ITS ADDRESS GIVEN
         DC    XL1'33'             4 BYTE LABEL, 4 BYTE INPUT DATA
         DC    FL1'46'             LABEL GOES IN AT OFFSET 46
         DC    FL1'52'             DATA TO START AT OFFSET 52
         DC    A(0)                DATA ADDRESS, TO BE FILLED IN LATER
         DC    XL1'14'             SET FLAGS TO INDICATE ONLY DATA
         DC    XL1'03'             NO LABEL,4 BYTES OF INPUT DATA
         DC    FL1'0'              NO LABEL
         DC    FL1'62'             DATA STARTS AT OFFSET 62
         DC    A(0)                END OF THIS PATTERN
```

Figure A-7 (Part 5 of 6). Sample Reentrant Format Appendage

```
COMFMT2  DS    0F                  FORMAT PATTERN FOR COMMON 2nd LINE
         DC    XL1'3F'             DATA NOT TO BE CONVERTED,AND ADDRESS
*                                      OF BOTH LABEL AND DATA IS PRESENT
         DC    XL1'7F'             8 BYTE LABEL,16 BYTES OF DATA
         DC    FL1'10'             LABEL, AS ALWAYS, AT OFFSET 10
         DC    FL1'21'             DATA STARTS AT OFFSET 21
         DC    A(COMLABEL)         POINT TO LABEL ON COMMON 2ND LINE
         DC    A(0)                DATA ADDRESS IS FOUND DURING
*                                     EXECUTION
         DC    A(0)                END OF FORMAT PATTERN
         END   AMDUSR02
```

Figure A-7 (Part 6 of 6). Sample Reentrant Format Appendage

Debugging a User Program

Figure A-8 shows a sample ABEND dump of the user exit routine ABENDXIT, shown in Figure A-6. Certain important fields are highlighted in the figure and marked with numbers; the numbers see the explanations below:

1. PSW for the abnormally-terminating program. The address in the second half of the PSW is an address in the abnormally-terminated program. To find the entry point and name of the program, compare this address to the entry point addresses in the contents directory entry list. The abnormally-terminating program is the one whose entry point address is closest to and greater than the address in the PSW.

   Note: If the address in the PSW does not immediately indicate the entry point address of the failing program, you can locate the beginning address of the abnormally-terminating program by tracing PRDMP's save area chain. See point 4, below.

2. Part of a contents directory entry (CDE). This shows the name of the abnormally-terminating program, ABENDXIT, its entry point, X'05D080', and the pointer to the appropriate entry in the extent list.

3. An extent list entry. This shows the beginning address (not necessarily the entry point) of the abnormally-terminating program. Subtract this address from the address in the PSW to find the address of the instruction following the instruction that failed.

   For example, in this case:

   ```
   address in PSW - beginning address = offset (hex)
           5D092 - 5D080 = 12
   ```

   The failing instruction in ABENDXIT can be found at offset X'12' in the program. (See part 2 of Figure A-8, number 3.)

4. The first save area in the save area trace table (system save area) is chained to the following AMDPRDMP module save areas:

```
AMDPRCTL - AMDPRDMP control routine
AMDPRMSC - AMDPRDMP scan routine
AMDPRFRM - EDIT control routine
AMDPRFLT - EDIT trace record selection routine
AMDPREXT (or AMDPRAPP) - EDIT user program selection
                         routine.
```

5. The user program's registers are stored in AMDPREXT's or AMDPRAPP's save area. Add the contents of register 12 to X'6AC' to get the address of a fullword that points to an EDIT communication table. At Offset X'1D0' into this table are the following:

   a. The 8-byte EBCDIC name of the current user program (the failing program).

   b. The entry point address of the current user program (the failing program).

   These fields are shown in part 3 of Figure A-8.

6. Register 1 in AMDPREXT's or AMDPRAPP's save area points to the parameter list that EDIT passes to the user program. (See Figure A-1.)

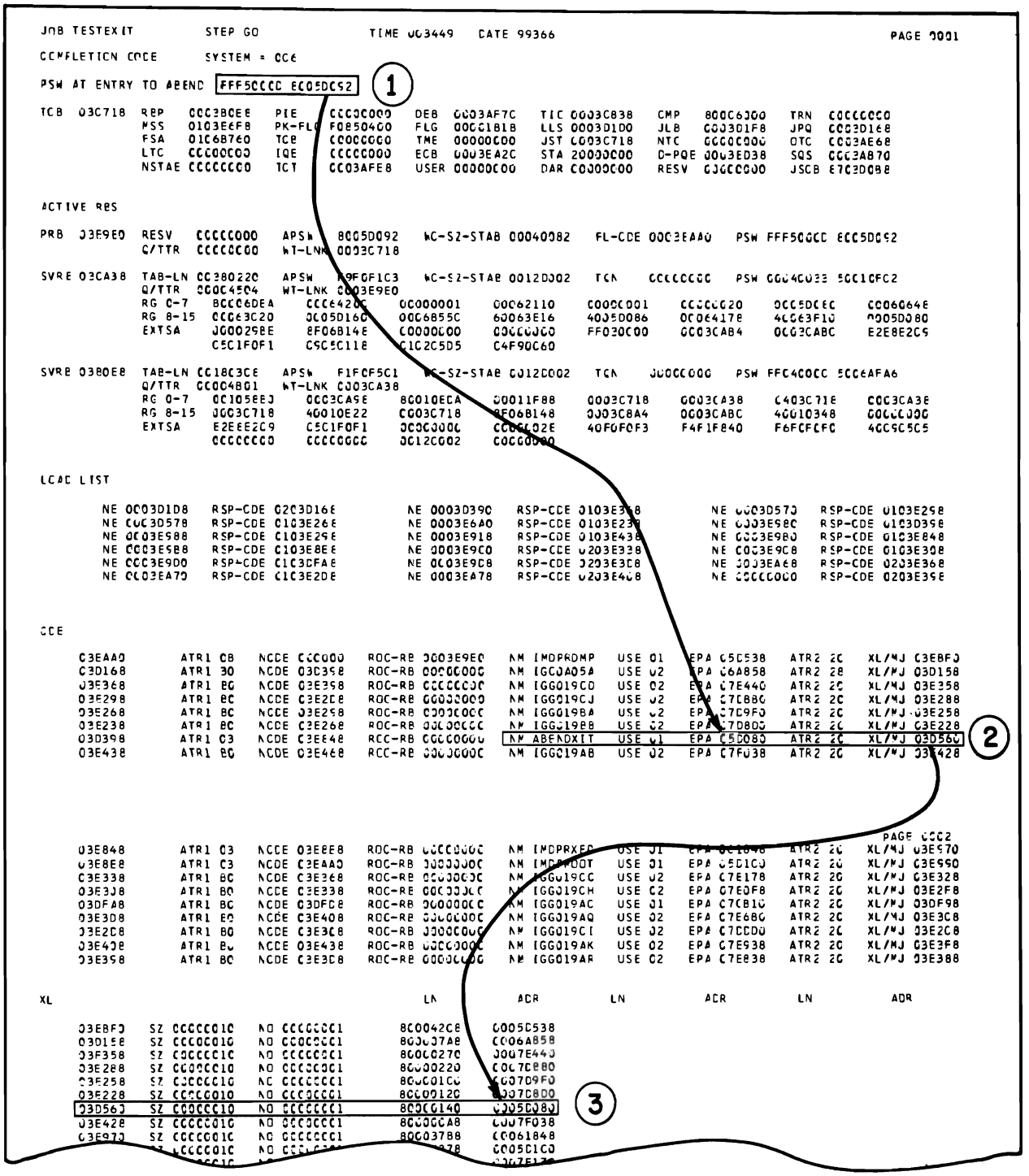
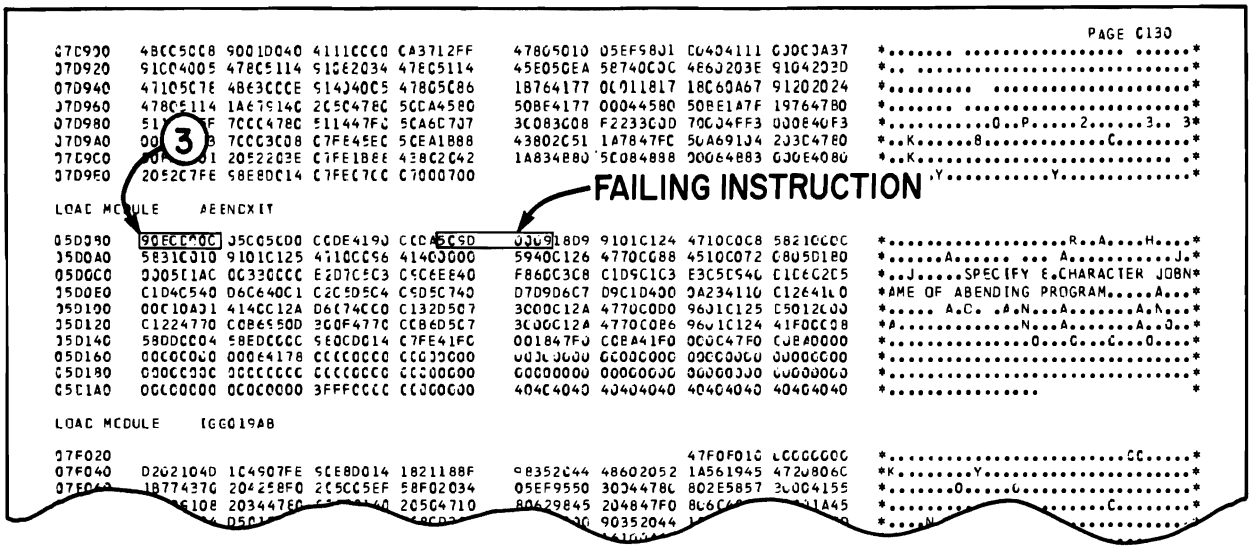


Figure A-8 (Part 1 of 3). Sample ABEND Dump Showing Fields Needed for Debugging User Exit Routine ABENDXIT



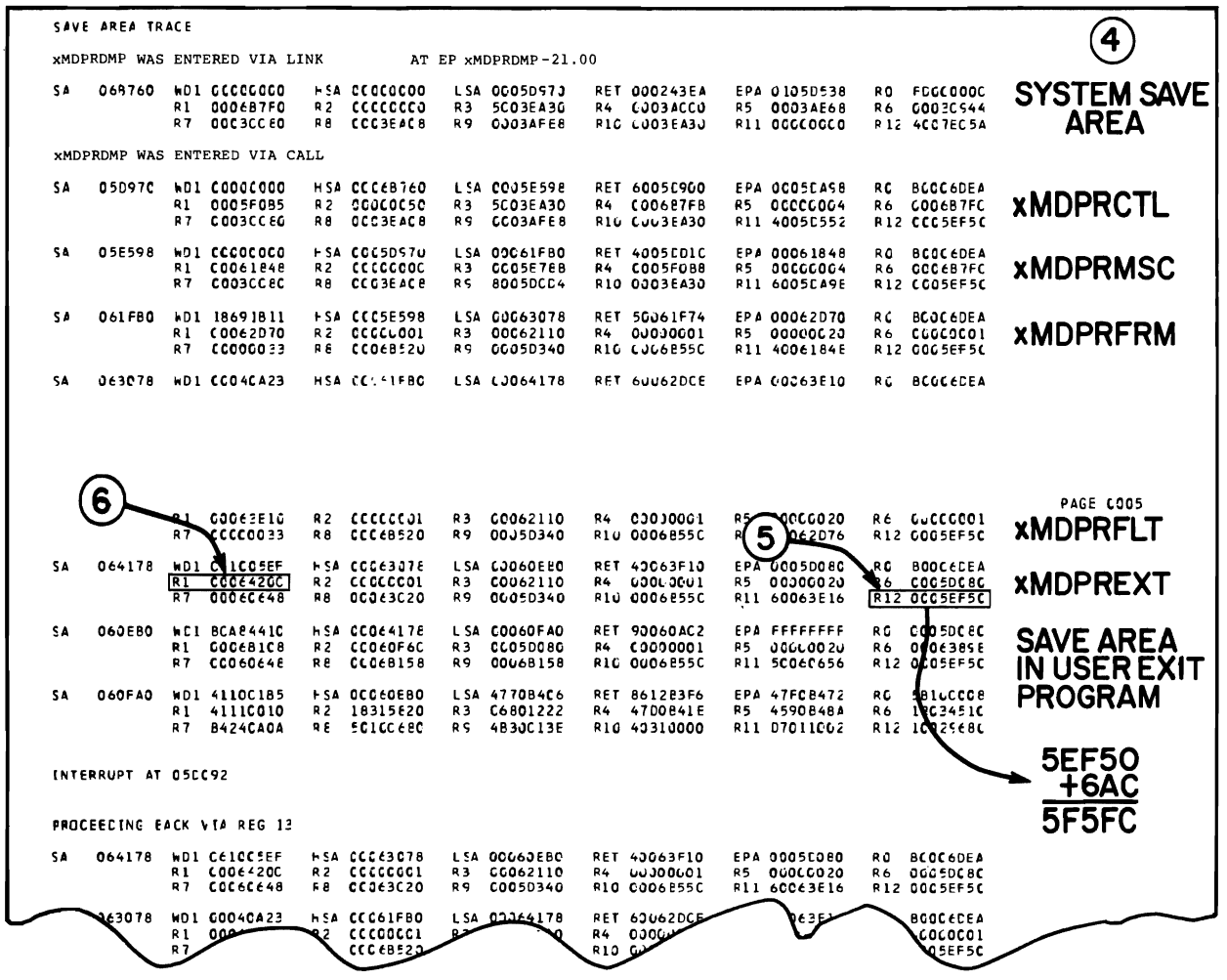
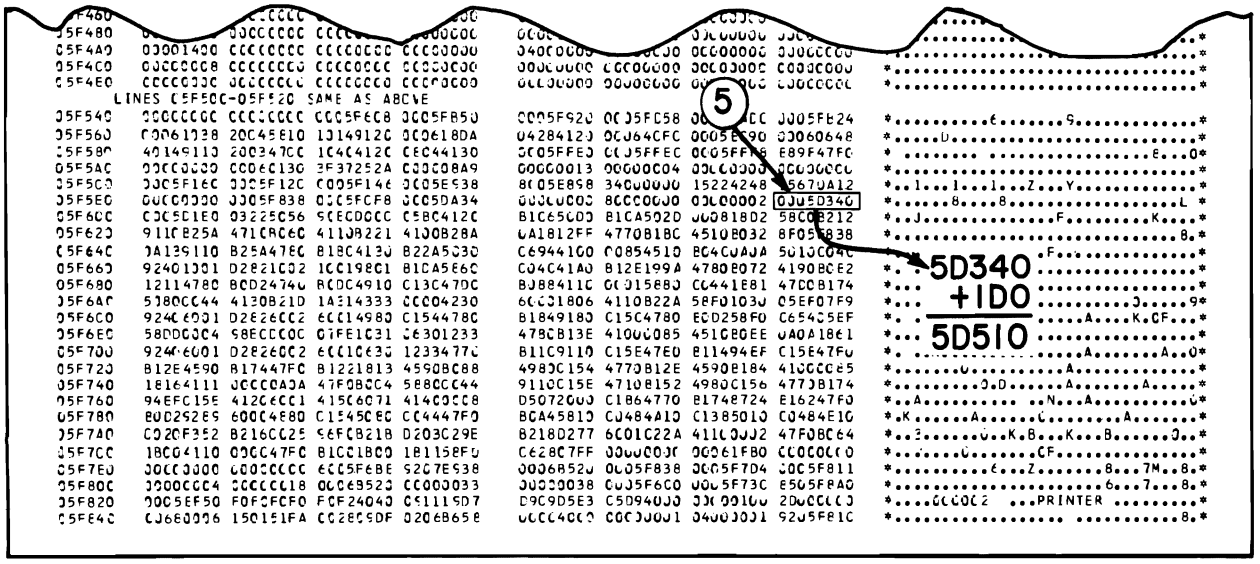
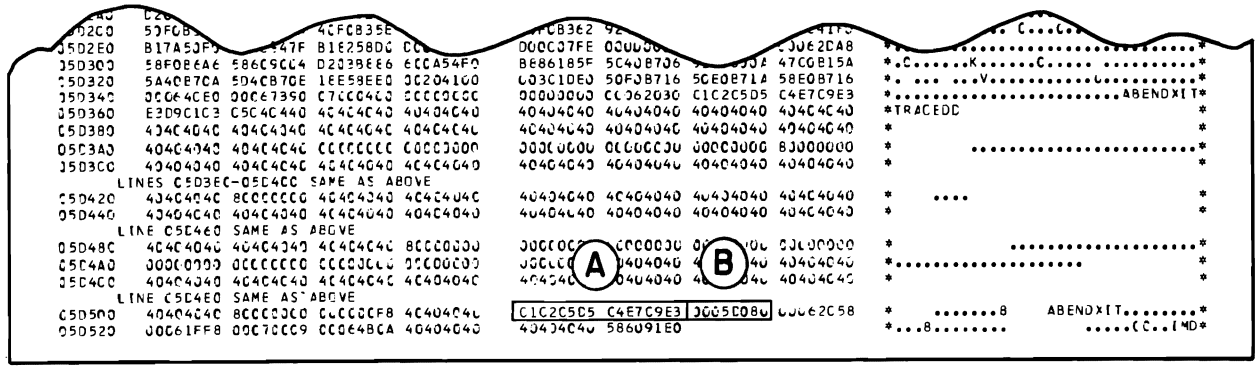


Figure A-8 (Part 2 of 3). Sample ABEND Dump Showing Fields Needed for Debugging User Exit Routine ABENDXIT





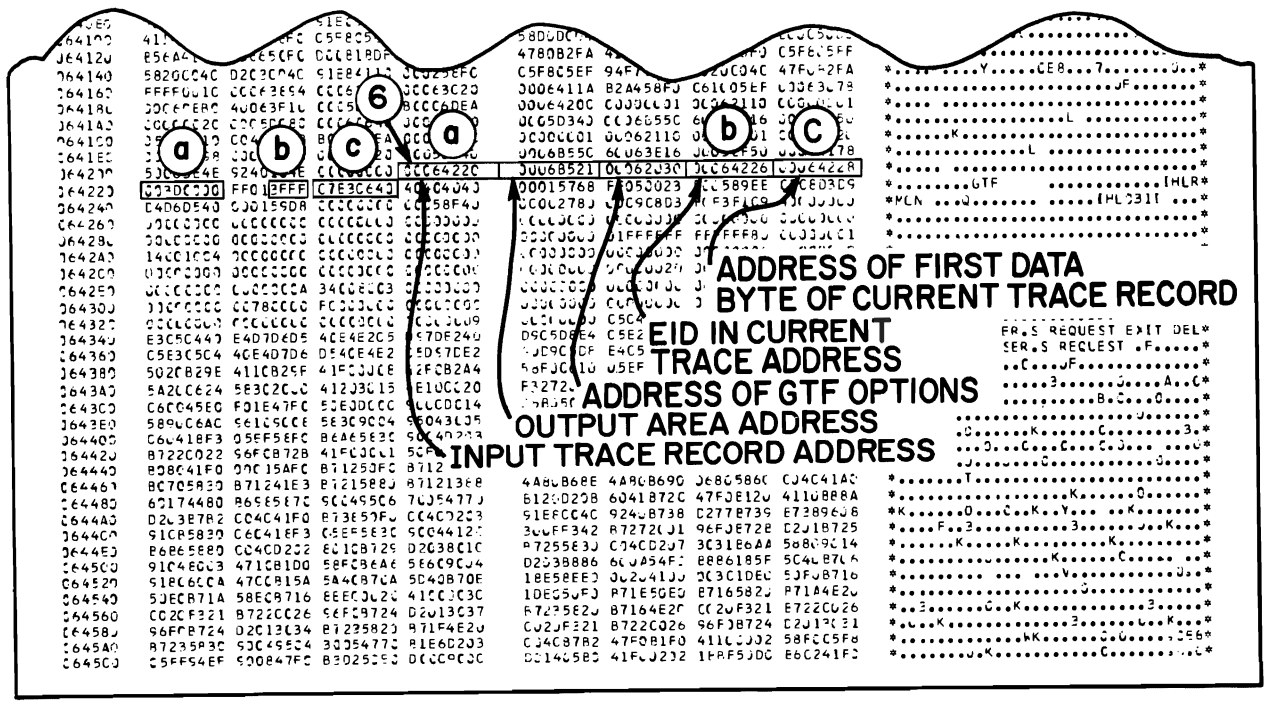


Figure A-8 (Part 3 of 3). Sample ABEND Dump Showing Fields Needed for Debugging User Exit Routine ABENDXIT

JCL and Control Statement Examples

The following examples show how to test a user program.

Example 1: Link Editing a User Exit Routine into a Library

This example shows how to make a user exit routine available to PRDMP by link-editing it into a system library.

```
//LKUSRPGM      JOB     MSGLEVEL=(1,1)
//              EXEC    PGM=IEWL,PARM='XREF,LET,LIST,NCAL'
//SYSPRINT      DD      SYSOUT=A
//SYSLMOD       DD      DSNAME=SYS1.LINKLIB,DISP=OLD
//SYSLIN        DD      *
    object deck
         NAME       EXITNAME
/*
```

In this example:

EXEC Statement
invokes the linkage editor and requests maximum diagnostic listings.

SYSPRINT DD Statement
defines the message data set.

SYSLMOD DD Statement
defines the output data set, in this case the linkage library, SYS1.LINKLIB. The output data set can also be a permanent library to be invoked later by a JOBLIB or STEPLIB DD statement; in that case the SYSLMOD DD statement should be coded as follows:

```
//SYSLMOD DD    DSNAME=MYLIB,UNIT=2314,VOL=SER=231400,
//              DISP=(NEW,KEEP),SPACE=(1024,(20,2,1))
```

SYSLIN DD Statement
defines the input data set, in this case, the object deck for the user program.

NAME Control Statement
specifies the member name, and thus the program name, to be assigned to the user program. In this case, the member name is EXITNAME; to invoke this program in a later execution of PRDMP, you would have to specify EXIT=EXITNAME on the EDIT control statement.

Example 2: Testing a User Exit Routine

This example shows how to link edit a user exit routine into a library for testing:

```
//TSEXTRTN      JOB         MSGLEVEL=(1,1)
//STEP1         EXEC        PGM=IEWL,PARM='XREF,LET,LIST,NCAL'
//SYSPRINT      DD          SYSOUT=A
//SYSLMOD       DD          DSNAME=MYLIB,UNIT=2314,VOL=SER=231400,
//       DISP=(NEW,KEEP),SPACE=(1024,(20,2,1))
//SYSLIB        DD            *
    object deck
         NAME      MYEXIT
/*
//STEP2         EXEC        PGM=AMDPRDMP,PARM='ER=1'
//STEPLIB       DD          DSNAME=MYLIB,UNIT=2314,VOL=SER=231400,
//       DISP=OLD
//SYSPRINT      DD          SYSOUT=A
//PRINTER       DD          SYSOUT=A
//TRACEDD       DD          DSNAME=TRACE2,UNIT=2400,VOL=SER=TRC2TP,
//       LABEL=(,NL),DISP=OLD
//SYSIN         DD   *
         EDIT      DDNAME=TRACEDD,SYS,EXIT=MYEXIT
/*
```

This example consists of two steps.

In the first step:

EXEC Statement
invokes the linkage editor and requests diagnostic information.

SYSPRINT DD Statement
defines the message data set.

SYSLMOD DD Statement
defines the output data set, in this case a permanent job or step library named MYLIB.

SYSLIB DD Statement
defines the input data set, in this case an object deck containing the user program.

NAME Control Statement
specifies a member name (program name) to be assigned to the user program. Specify this program name on the EDIT control statement (EXIT = MYEXIT) when you need the exit routine for a particular PRDMP execution.

In the second step:

EXEC Statement
invokes PRDMP and specifies that, if an error occurs in the exit routine, EDIT should print the record associated with the error and delete the exit routine. (See the discussion of the EXEC statement in the section "Job Control Language Statements" earlier in this chapter.)

STEPLIB DD Statement
defines the data set that contains the exit routine, which, in this case, is MYLIB, a data set defined in STEP1 by the SYSLMOD DD statement.

SYSPRINT DD Statement
defines the message data set.

PRINTER DD Statement
defines the data set to which PRDMP output will be directed.

TRACEDD DD Statement
defines the data set containing trace records to be processed by the exit routine.

SYSIN DD Statement
defines the data set that contains the PRDMP control statement. The data set follows immediately.

EDIT Control Statement
invokes the EDIT function of PRDMP, specifies that the trace data exits as an external trace data set, and supplies the name of the exit routine. Note that this name is the same as the member name specified in the NAME control statement in STEP1.

Example 3: Testing a User Format Appendage

This example shows how to add a user format appendage to a temporary data set for testing.

```
//TSTFMT        JOB          MSGLEVEL=(1,1)
//STEP1         EXEC         PGM=IEWL,PARM='XREF,LET,LIST,NCAL'
//SYSPRINT      DD           SYSOUT=A
//SYSLMOD       DD           DSNAME=&TEMPLIB,UNIT=SYSDA,
//       SPACE=(1024,(20,2,1)),DISP=(NEW,PASS)
//SYSLIN        DD           *
    object deck
         NAME       AMDUSR01
/*
//STEP2         EXEC         PGM=AMDPRDMP,PARM='ER=3'
//STEPLIB       DD           DSNAME=&TEMPLIB,DISP=OLD
//SYSPRINT      DD           SYSOUT=A
//PRINTER       DD           SYSOUT=A
//TRACEDD       DD           DSNAME=TRACE,UNIT=2400,VOL=SER=TRCTPE,
//       LABEL=(,NL),DISP=OLD
//SYSIN         DD           *
         EDIT      DDNAME=TRACEDD,USR=ALL
/*
```

This example consists of two steps.

In the first step:

EXEC Statement
invokes the linkage editor.

SYSPRINT DD Statement
defines the message data set.

SYSLMOD DD Statement
defines a temporary data set that contains the format appendage.

SYSLIN DD Statement
defines the input data set, in this case the object deck containing the format appendage.

NAME Control Statement
specifies a member name (program name) for the format appendage. Note that the name shown in this example conforms to the convention for naming format appendages; that is, it is formed from the prefix AMDUSR concatenated with the format identifier (FID) to be specified in the GTRACE macro when user records are created.

In the second step:

EXEC Statement
invokes PRDMP and specifies that ABEND processing should not be suppressed if a program check occurs in the format appendage. (See the discussion of the EXEC statement in the section "Job Control Language Statements" earlier in this chapter.)

STEPLIB DD Statement
defines the data set where the format appendage resides.

SYSPRINT DD Statement
defines the message data set.

PRINTER DD Statement
defines the data set to which the format appendage will direct its output.

TRACEDD DD Statement
defines the trace data set containing the records that the format appendage will process. In this case, the trace data set is on tape.

SYSIN DD Statement
defines the data set containing PRDMP control statements. The data set follows immediately.

EDIT Control Statement
invokes the EDIT function of PRDMP, specifies that the trace data exists as an external trace data set, and specifies that EDIT is to process all user-created records.

Appendix B. PRDMP User Exit Facility

PRDMP provides exits for user-written formatting routines that supplement normal PRDMP formatting and diagnosis of control blocks. These exits make it possible for you to tailor PRDMP output to your needs.

The advantages of the PRDMP user exit facility are:

- It allows you to have your own set of control statements to format and diagnose any system or user control blocks, and
- It permits you to supplement normal PRDMP processing and control block formatting.

The six types of exits available to a user are:

- TCB exit — gets control after PRDMP processes a TCB.
- ASCB exit — gets control after PRDMP processes an ASCB.
- FORMAT exit — gets control when PRDMP processes the PRDMP control statement FORMAT.
- JOBNAME/CURRENT exit — gets control when PRDMP processes the PRDMP control statement PRINT JOBNAME or PRINT CURRENT.
- NUCLEUS exit — gets control when PRDMP processes a PRINT NUCLEUS statement.
- User-control-statement exit — gets control when PRDMP processes a user-defined control statement.

The interface between PRDMP and the user formatting routines consist of the exit control table (ECT) and the PRDMP parameter list.

The ECT is a table that identifies user exits and associated user module names. You fill it in and link-edit it into SYS1.LINKLIB. PRDMP uses the ECT to pass control to your modules. See Chapter 3 for more details.

PRDMP passes a two-part parameter list to your user exit routines. All user exit routines use the first part; it includes the address of a TCB being processed, the addresses of PRDMP-supplied routines that access storage, format, and print control blocks or data areas, and the address of the second part of the parameter list. Only user-control-statement exit routines use the second part of the PRDMP parameter list (the Extension). The Extension contains the address of the parameter list your exit routine is passing to PRDMP. See Chapter 3 for details.

PRDMP permits your exit routine to gain control during its execution. Your routine receives control when there is an exit flag setting in the ECT or when PRDMP processes a control statement that you defined for this exit.

The Exit Control Table (ECT)

The exit control table (ECT) resides in SYS1.LINKLIB as the module AMDPRECT. You indicate, in the ECT, a module name and the point in PRDMP processing at which this module should receive control.

Format of the ECT

The ECT contains a 20-byte entry for each user exit module. Figure B-1 shows an entry in the ECT.

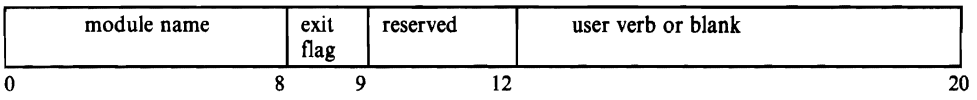


Figure B-1. ECT Entry Format

The **module name** is an eight-character load module name that PRDMP uses to load your program. The name is left-justified and padded with blanks.

The **exit flag** is a one-byte field that, if any bit is on, causes your routine to receive control at that exit. The exits are defined by different flag settings. Shown in Figure B-2.

Flag setting		Meaning
X'00'	-	No exit
X'80'	-	TCB exit
X'40'	-	ASCB exit
X'20'	-	FORMAT exit
X'10'	-	PRINT JOBNAME/CURRENT exit
X'08'	-	PRINT NUCLEUS exit

Figure B-2. ECT Exit Flag Settings

The **reserved field** is not used. It contains binary zeros.

The **user verb** is an eight-character name of your control statement. Your routine receives control every time PRDMP encounters an ECT entry that contains that verb. The user verb is left-justified and padded with blanks.

Setting Up the ECT

You build an entry in the ECT using the SPZAP service aid program. This is a two-step procedure.

First you find a free entry using the DUMPT function of SPZAP. You check the DUMPT output to find a free entry. If it exists, you can use the SPZAP VERIFY function later. An entry is free if there is no module name, the exit flag is zero, and the user verb is blank. ECT entries do not have to be consecutive so you cannot assume that a free entry exists at a 20-byte offset from the last known entry. Use the following JCL to list the contents of the ECT module AMDPRECT:

```
//              EXEC        PGM=AMASPZAP
//SYSPRINT      DD          SYSOUT=A
//SYSLIB        DD          DSNAME=SYS1.LINKLIB,DISP=SHR
//SYSIN         DD          *
                DUMPT AMDPRECT
/*
```

Second, you activate an exit using the REP function of SPZAP. You supply a module name and either include a user verb or leave the verb field blank and turn on a bit in the exit flag. The JCL and control statements to execute SPZAP are the same as specified above. You add the following SPZAP control statements, preceding the DUMPT statement in the input stream:

```
NAME AMDPRECT
VERIFY offset old-data
REP offset ECT-entry
```

You replace the following with appropriate values:

offset
: is the displacement from the beginning of the ECT where you found a free entry.

old data
: is the contents at that displacement as shown in DUMPT output.

ECT entry
: consists of a 20-byte field, in the format of an ECT entry, that contains a module name, exit flag, reserved field (three bytes as shown in the DUMPT output), and a user verb or blank field. See Figure B-3 Sample ECT.

The DUMPT statement produces a dump showing the new entry.

For more information on the NAME, VERIFY, REP and DUMPT control statements, see Chapter 6: SPZAP.

Only one type of user exit should be associated with a module name. However, in case multiple flag settings occur or a user-verb exit also has an exit flag set, you

should code your exit routine so its processing does not depend on what kind of exit caused it to receive control.

Note that, when you set up ECT entries, PRDMP takes exits sequentially. That is, if four user modules use the TCB exit, then each gets control according to the list order in the ECT.

Figure B-3 is an example of an ECT containing '2' active entries.

```
AMDPRECT CSECT
*                                                    *
*          E X I T  F L A G  D E F I N I T I O N S   *
*                                                    *
TCB          EQU   X'80'           EXIT WHEN TCB PROCESS ASSOCIATED BLOCKS
FORMAT       EQU   X'20'           EXIT WHEN FORMAT STMT PROCESSED
CURJOB       EQU   X'10'           EXIT WHEN PRINT JOB/CURR STMT PROCESSED
NUCLEUS      EQU   X'08'           EXIT WHEN PRINT NUCLEUS STMT PROCESSED
VERB         EQU   X'00'           EXIT ONLY IF USER VERB PROCESSED
*          E X I T  C O N T R O L  T A B L E         *
ECTNTR01     DS    0CL20          ACTIVE ENTRY FOR TCB EXIT
             DC    CL8'TSTRAFD1'   EXIT USER MODULE NAME
             DC    AL1 (TCB)       TCB EXIT FLAG SET
             DC    3XL1'00'        RESERVED
             DC    CL8' '          USER VERB NAME EMPTY
ECTNTR02     DS    0CL20          ACTIVE ENTRY FOR USER VERB
             DC    CL8'TSTMYBLK'   EXIT USER MODULE NAME
             DC    AL1(VERB)       NO EXIT FLAG SET
             DC    CL8'USERBLOC'   USER CONTROL STMT VERB IS USERBLOC
ECTNTR03     DS    0CL20          INACTIVE ENTRY
             DC    CL8' '          NO EXIT USER MODULE NAME FILLED IN
             DC    AL1(VERB)       NO EXIT FLAG SET
             DC    3XL1'00'        RESERVED
             DC    CL8' '          NO USER VERB
*  SEVENTEEN ENTRIES, ECTNTR04 - ECTNTR20, FOLLOW WHICH MATCH ECTNT03  *
```

Figure B-3. Sample ECT

Note that the ECT initially contains twenty 20-byte entries (400 bytes). If you need to activate more than 20 entries, you can expand the ECT by re-link editing AMDPRECT, increasing its size.

User-Written Formatting Routines

Your formatting routine can be a single module or a set of modules. The routine, including dynamically allocated storage and excluding modules loaded into the LPA, should not exceed 8K. PRDMP guarantees a user routine 8K of storage; however, more may be available depending on the type of PRDMP processing involved.

Your module must be resident in SYS1.LINKLIB or a private library, JOBLIB or STEPLIB. For all user modules, PRDMP uses a LOAD macro to get the module, then passes control via standard linkage.

Conditions on Entry to a User Formatting Routine

- A FORMAT or PRINT JOBNAME/CURRENT/NUCLEUS exit user routine gets control when PRDMP processes the respective PRDMP control statement. At the time of entry to your routine, only a title has been printed and no TCB address or operand information is available.

- For all types of user exits, PRDMP uses standard linkage to pass control to your routines: register 13 contains the address of a save area, register 14 contains a return address to PRDMP, and register 15 contains the entry point address in your routine. Register 1 points to the PRDMP parameter list which is further described, along with a format diagram, as the next topic.

  Other conditions on entry to your routine vary depending on the type of exit:

- A TCB routine gets control after PRDMP processes a TCB, including the case when PRDMP processes a FORMAT, PRINT JOBNAME, or PRINT CURRENT statement (if these control statements all require TCB processing).

  At the time of entry to a TCB exit routine, PRDMP has printed a title and TCB with its associated control blocks. The TCB address is available to you in the PRDMP parameter list.

- An ASCB routine gets control after PRDMP processes control blocks for a new ASID and including the case when PRDMP processes a FORMAT, PRINT JOBNAME, or PRINT CURRENT statement (if these control statements require address-space control block formatting).

  At the time of entry to an ASCB exit routine, PRDMP has formatted the control blocks associated with one ASCB.

- A user-control-statement exit routine gets control when PRDMP processes a user control statement that matches a user verb in the ECT. At the time of entry to your routine, PRDMP has started a new page. The routine should print a title to indicate that it received control. The parameter list extension points to a list of parameters that appear on the user control statement being processed.

PRDMP Parameter List

The parameter list pointed to by register 1 begins on a fullword boundary and is fourteen fullwords long. It serves as a communications area for PRDMP, your routine, and the PRDMP service routines. Its format is shown in Figure B-4, 'The PRDMP Parameter List and Extension.'

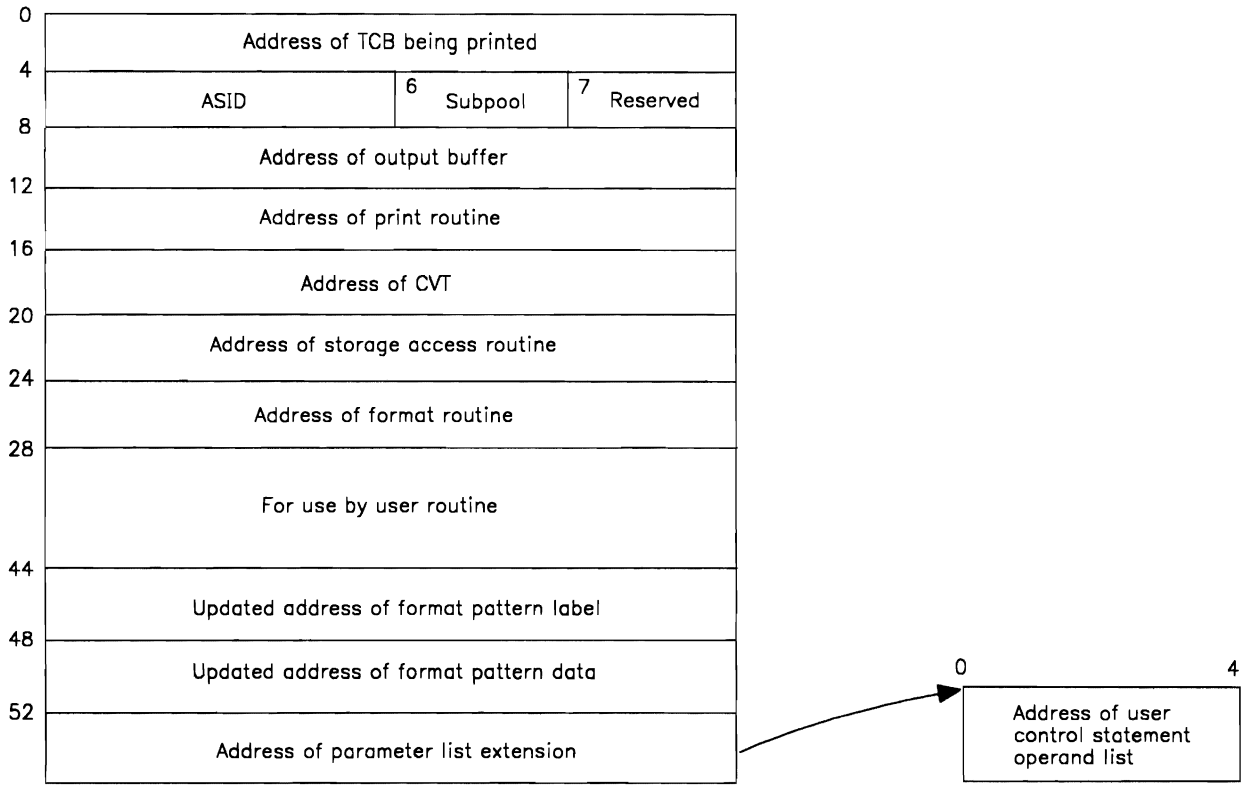


Figure B-4. The PRDMP Parameter List and Extension

The IHAABDPL mapping macro maps the PRDMP parameter list, which is the same as the parameter list passed to user-written formatting routines for ABEND dumps taken to SYSABEND or SYSUDUMP data sets.

- The first word in the parameter list contains the TCB address if PRDMP is processing a TCB at the time the parameter list is passed. This field is always used for TCB exits.
- The second word of the parameter list consists of three fields:

  1. The first field is the two-byte ASID field which identifies the ASID being processed.

  2. The second field is a one-byte subpool field which contains the number of the subpool your routine can use for dynamical storage allocation.

3. The last field is a byte field that indicates the version of PRDMP that is processing. The settings are:

Bit	Setting and Meaning
0	0 - module loaded by SNAP 1 - module loaded by PRDMP
1	0 - system is OS/VS2 1 - system is OS/VS1
2	For data management formatter under SNAP: 0 - format the DEB 1 - format the DEB, DCB, and IOB
3	Applies to OS/VS2 only: 0 - identifies VS2 PRDMP 1 - identifies MVS/IPCS
4-7	Reserved

- The third word of the parameter list contains the address of an output buffer. It is needed by PRDMP service routines and by your module when it does its own formatting.

- The fouth word of the parameter list contains the address of the print routine. This is a PRDMP-provided service routine that all user modules must use to print an output line. See 'The Print Service Routine.' later in this section.

- The fifth word of the parameter list contains the address of the CVT. Note that when PRDMP is processing a selective dump that does not contain the CVT, this field might not be filled in. If your routine needs to reference the CVT, this address must be used.

- The sixth and seventh words of the parameter list are the addresses of two other PRDMP-supplied service routines: **the storage access routine** and **the format routine**. Your module must use the storage access routine to obtain data from the dumped data set. This routine is described in detail in 'The Storage Access Service Routine' later in this section.

  Your user module may optionally invoke the format routine to have the output buffer automatically formatted. This service routine is described in detail in "The Format Service Routine" later in this section.

- The parameter list has four fullwords of work area, starting at the eighth word.Your module can use this work area to construct a GETMAIN parameter list in order to have reentrant code.

- Following the work area are two words used by the format service routine, starting at the twelfth word. The twelfth word stores the updated address of a label for the output line. The thirteenth word stores the updated address of a data item for the output line. If your routine does not supply either of these addresses in a format pattern, the format service routine uses these fields to calculate the address. For more information about the format routine and format patterns, see "The Format Service Routine" later in this section.

- The fourteenth and last word in the parameter list contains a pointer to the parameter list extension. The extension consists of one full word that points to another parameter list associated with the user control statement being processed. The parameter extension is a fullword of binary zeros when there is no control-statement parameter list. The extension to the PRDMP parameter list allows your control-statement exit routine to receive parameters during execution.

Returning to PRDMP

All types of user routines must return control to PRDMP using standard linkage. Register 15 should contain a return code of 0 indicating processing is complete or a return code of 4 indicating the storage necessary to process could not be obtained.

A user-control-statement exit routine may specify an additional return code in the PRDMP parameter list extension. If an parameter on a user statement contains an error, the operand list pointer in the extension should be replaced by a code of 4, 8, or 12. This code causes PRDMP to print an error message on the SYSPRINT data set.

The following are the error codes and their corresponding messages:

Code	Message
4	DELIMITER ERROR IN OPERAND FIELD OF verb
8	INVALID KEYWORD IN OPERAND FIELD OF verb
12	SYNTAX ERROR IN OPERAND FIELD OF verb

The verb field of these messages contains the user verb associated with the user control statement.

Guidelines for Writing a User Formatting Routine

You can design a routine to format a specific area of storage, print non-IBM control blocks, or print IBM control blocks that PRDMP does not process. The function of your routine should depend on the type of exit where it receives control; conditions vary as described in the previous section, "Conditions on Entry to a User Formatting Routine."

Guidelines for User Modules

For all user modules, some guidelines apply. The following is a summary of these guidelines:

- Your module must be identified to PRDMP in the ECT and either an exit flag or user verb or both must indicate when control should be passed.

- Your routine must not exceed 8K of storage including a dynamically acquired area. To have reentrant code, you may use twenty bytes in the parameter list.

- Your routine must use the storage access service routine and the print service routine when you need to perform their designated functions. If your routine performs its own formatting, it must use the buffer pointed to in the PRDMP parameter list.

- If you use the storage access routine, remember that the address boundary you request (that is, byte, word, 4K), determines the length of data that you can reference (1 byte, 1 word, 4K). The maximum length available is 4096 bytes on a 4K boundary.

- Your routine may print comments at any time. Informative messages can be used to identify the module processing. Error messages diagnose conditions encountered in processing or in analysis of data areas.

- You must provide recovery processing for unexpected incidents. Return codes generated by any SVC issued within the module and by service routines invoked within a module should be checked.

- If your module has an associated user control statement, and the statement has parameters, you must check for valid parameters and be prepared to handle error or default conditions. PRDMP passes your routine the address of a parameter list in the parameter list extension. To indicate an error condition to PRDMP, your routine must replace the address in the extension with a return code of 4, 8, or 12. See "Returning to PRDMP" earlier in this section.

- Prior to returning control to PRDMP, your module must free any dynamically allocated storage and release any other resources it obtained.

  When your routine returns control to PRDMP, it must pass a return code of 0 or 4 in register 15; a code of 0 indicates user module processing is completed and a code of 4 indicates that your module was unable to obtain the storage necessary to process.

The Storage Access Service Routine

The address of the PRDMP storage access routine is the fifth word in the PRDMP parameter list.

Your routine must pass control to the PRDMP-supplied storage access routine for all references to system storage because any reference to storage in a dumped system is really referring to a record in the dump data set.

You pass control to the storage access routine using standard linkage conventions. Upon entry, register 0 must contain the virtual address to be referenced and register 1 must contain the original parameter list address.

When the storage access routine returns control to your module, register 0 contains the address of the data requested and register 15 contains a return code.

The length of data available for reference depends on the boundary of the address requested. If the address is on a byte boundary, only one byte can be referenced. The maximum length available is 4096 bytes on a 4K boundary.

In the example in Figure B-5, the subroutine MEMORY uses the storage access routine, then checks to be sure the data is returned.

Note that your routine must test the contents of register 15 for a return code of 0 to make sure that the storage access routine was successful. A return code of 4 indicates that the requested storage is not in the dump or the service routine is unable to access it.

You must invoke the storage access routine each time you wish to format system storage. This is because the address you are requesting may have been overlaid by another routine (such as the format service routine) that does a storage access in the interim between your two requests.

The ASID used for accessing storage is the current ASID being processed by PRDMP, or in the case of exits called via a verb, the ASID in the dump header. Any ASID could be specified by the exit program by storing the ASID in the print dump parameter list. See also the The PRDMP Parameter List and Extension, Figure B-4.

```
* THIS ROUTINE IS USED FOR ALL REFERENCES TO STORAGE IN THE SYSTEM.
* 'PREG' ON ENTRY CONTAINS THE ADDRESS WANTED FROM THE SYSTEM; IT
* ALSO USES 'PREG' TO RETURN TO THE CALLER THE ADDRESS OF THAT
* REQUESTED DATA.  "R1" ALWAYS CONTAINS THE ADDRESS OF THE
* ORIGINAL PARAMETER LIST.
* IF THE REQUESTED ADDRESS IS NOT AVAILABLE, CONTROL IS PASSED
* TO THE ADDRESS IN 'ERROR'.
MEMORY   LR     R0,PREG         SET REQUESTED ADDR TO REGISTER 0
         L      R15,PARMMEMA    GET STORAGE ACCESS ADDRESS FROM
*                               PARM LIST
         BALR   R14,R15         GO TO STORAGE ROUTINE
         LR     PREG,R0         RETURN REQUESTED ADDR IS SAME REG
         LTR    R15,R15         IS REQUESTED ADDRESS AVAILABLE
         BC     8,USEDATA       YES--USE IT
         B      ERROR           NO--GO TO ERROR ROUTINE
USEDATA  L      RADDR,0(PREG)   GET DATA AT REQUESTED ADDRESS
```

Figure B-5. Example Using the Storage Access Routine

The Format Service Routine

The PRDMP format routine is available to your module to format its output. This service routine performs two functions: it converts data to printable hexadecimal (if necessary) and it formats data in the output buffer.

At each invocation, the format service routine can format one output line. Your module must provide this input:

- The address of the output buffer in the PRDMP parameter list
- The addresses of data items and/or labels by using the format patterns.

(See the topic "Format Patterns for PRDMP Format Routine" following in this section.)

The PRDMP parameter list contains the address of the 121-byte buffer where the output line is to be formatted. It also contains two fullwords of work area needed by the format service routine. The format patterns indicate data and labels to be inserted in the output line.

Your module can invoke the format routine two different ways. You may specify format patterns to create one output line. Or you may invoke the format routine more than once. Each time, you can specify patterns for part of an output line. Note that if you choose to format a portion of a line, you must be careful not to overlay previously-formatted portions.

When a line is formatted, your module must invoke the PRDMP print service routine to perform the print operation. The format service routine cannot print the buffer.

Your module obtains the service routine address from the sixth word of the PRDMP parameter list and must use standard linkage to pass control. You must pass the address of the first format pattern in register 0 and the address of the PRDMP parameter list register 1.

When the format routine passes control back to your module, register 15 contains a return code of 0 or 4. The value of the return code is the same as that received by the format routine when it used the storage access routine; the meanings of the codes are the same as for the storage access routine.

Figure B-6 shows how a format routine is invoked. Figure B-8 shows format patterns associated with that invocation. Also, in Appendix A: Writing EDIT User Programs, Figure A-7 provides a full example of a user program that uses the format routine and format patterns.

```
*  THIS SECTION ACTUALLY OUTPUTS TWO LINES OF A TAPE UCB THEN SKIPS A LINE
OUTPUTIT  LA R0,TAPE1          SET ADDR OF LINE 1 FORMAT PATTERN
          BAL RLINK, FORMAT    GO FORMAT LINE
          BAL RLINK, PRINTIT   GO TO PRINT IT WITH THE SERVICE RTN
          LA R0, TAPEL2        SET ADDR OF LINE 2 FORMAT PATTERN
          BAL RLINK, FORMAT    GO FORMAT LINE
          BAL RLINK, PRINTIT   GO TO PRINT IT WITH THE SERVICE RTN
          BAL RLINK, PRINTIT   GO PRINT A BLANK LINE
*THIS SUBROUTINE CALLS THE FORMAT ROUTINE TO AUTOMATICALLY
*FORMAT AN OUTPUT LINE OF THE UCB.  INPUT TO THIS SECTION OF CODE
*IS THE ADDRESS OF THE FIRST FORMAT PATTERN IN REG 0.  REG 1 MUST
*CONTAIN THE ADDRESS OF THE ORIGINAL PRDMP PARAMETER LIST.  IF
*DATA CANNOT BE OBTAINED BY THE SERVICE ROUTINE, CONTROL IS
*PASSED TO AN ERROR SUBROUTINE TO PRINT ERROR MESSAGE
FORMAT    L     R15,PARMFRMT   GET RTN ADDR FROM PARM LIST
          BALR  R14,R15        GO TO SERVICE ROUTINE TO FORMAT LINE
          LTR   R15,R15        WAS FORMAT SUCCESSFUL
          BCR   8,RLINK        YES, GO PRINT LINE IMMEDIATELY
          B     ERROR          NO, GO TO ERROR ROUTINE
```

Figure B-6. Example Using the Format Routine

Format Patterns for PRDMP Format Routine

The format service routine uses the format pattern to locate information and position it in the output buffer. A format pattern consists of four to seven fields. One field indicates how the format routine should process the pattern. Three fields describe the position, length, and address of a label. Three other fields describe the position, length, and address of a data item.

If a series of format patterns are set up by your routine, the patterns that describe one line must be contiguous and have as a last entry a fullword of binary zeros.

The second and subsequent data and label addresses in a series are optional. If these fields are not provided, the service routine uses the last address plus the length to locate a data item or label. To keep track of updated addresses, the format routine uses the twelfth and thirteenth words in the PRDMP parameter list to store label and data pointers. See Figure B-4 for a diagram of the PRDMP parameter list.

See Figure B-7 to assist you in defining fields in a format pattern.

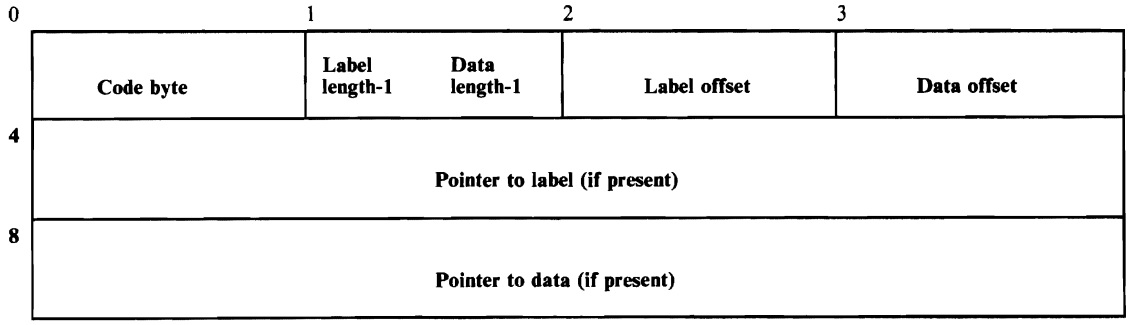


Figure B-7. Format Pattern Description

Code byte:

This one-byte field identifies the contents of a format pattern. The possible bit settings and their meanings are:

Bit	Setting and Meaning
0	- Unused.
1	- Unused.
2	- Data is not to be converted to printable hexadecimal.
3	- Data is in the caller's storage (not in the dump).
4	- Data pointer follows (bit 5 must also be on). Either a dump address or a storage address. (See bit 3)
5	- Data is to be placed in buffer.
6	- Label pointer follows (bit 7 must also be on).
7	- Label is to be placed in buffer.

If you do not set bits 4 and/or 6, then you should omit the label pointer and/or data pointer field(s).

Label length and data length: Each of these two fields is 4 bits in length and contains a length count that is one less than the actual input length. PRDMP uses these fields to update label/data addresses in the parameter list when the addresses are not included in a pattern.

Label offset and data offset - Each of these one-byte fields indicate the position of labels and data in the buffer. To indicate the first character of the buffer, specify 0 as the offset.

Label pointer: This four-byte field is optional. It contains the address of a character string to be used for labels. If you include a label pointer in the pattern, you must set bits 6 and 7 of the code byte. If you do not include this pointer, PRDMP updates the current label address in the PRDMP parameter list and uses it.

Data pointer: - This four-byte field is optional. It contains the address of the first byte of data to be placed in the output line. Note that if this data is not in the caller's storage, you must set bit 3 of the code byte to 0; this causes the format routine to use the storage access service routine to obtain the data. If a data pointer is in the pattern, bits 4 and 5 of the code byte must be set. If this pointer is not included, the format routine updates the current data address in the PRDMP parameter list and uses it. If bit 2 is 0, the format routine converts this data to hexadecimal.

```
* THIS CHARACTER STRING DEFINES LABELS TO BE USED BY THE SERVICE ROUTINE
* FOR THE FORMATTED UCB'S
*
TAPELB   DC   C'SNSVOLISTABFSCTDSCTDSNORESVOPTXTN'       UCB LABELS
* THE FOLLOWING FIVE FORMAT PATTERNS DESCRIBE THE FIRST OF THE
* TWO LINES FOR A TAPE UNIT CONTROL BLOCK (UCB).  THE FIRST TWO
* PATTERNS ARE EXPLAINED FOR EACH FIELD.
* THIS PATTERN IDENTIFIES A LABEL AND DATA ITEM THAT WILL BE
* INSERTED IN THE FIRST LINE OF A TAPE UCB
TAPEL1   DS   OF          THIS PATTERN SETS UP THE SNS FIELD IN
         DC   X'0F'       THIS CODE BYTE IS SET FOR: CONVERSION,
*                         DATA AND LABEL POINTERS FOLLOWING, AND LABEL
*                         AND DATA ITEM TO BE PLACED IN BUFFER.
         DC   X'23'       LABEL LENGTH-1 IS 2.DATA LENGTH-1 IS 3
         DC   FL1'12'     LABEL OFFSET IN OUTPUT LINE IS 12
         DC   FL1'17'     DATA OFFSET IN OUTPUT LINE IS 17
         DC   A(TAPELB)   POINTER TO STRING OF LABELS
         DC   A(0)        ADDR OF DATA TO BE FILLED IN DURING EXECUTION
*
* NEXT PATTERN SETS THE VOL1 FIELD IN THE OUTPUT LINE.  THIS
* PATTERN USES THE SAME LABEL STRING AND DATA ADDRESS.  THE
* FORMAT ROUTINE UPDATES ITS LABEL AND DATA ADDRESSES USING
* LENGTH FIELDS.  TWO WORDS IN THE PRDMP PARAMETER LIST STORE THE UPDATED
* ADDRESSES.
*
         DC   X'25'       CODE BYTE SET FOR: NO CONVERSION, PLACE
*                         LABEL AND DATA IN BUFFER(NO ADDRESSES-DO UPDATE
         DC   X'35'       LABEL LENGTH(4)-1 IS 3.  DATA LENGTH(6)-1 IS 5.
         DC   FL1'28'     LABEL OFFSET IN OUTPUT BUFFER IS 28.
         DC   FL1'33'     DATA OFFSET IN OUTPUT LINE IS 33.
*
* THE REMAINING PATTERNS IN THIS LINE ARE CONDENSED
*
         DC   X'05',X'31',FL1'44',FL1'52'                STAB FIELD
         DC   X'05',X'33',FL1'59',FL1'65'                FSCT FIELD
         DC   X'05',X'35',FL1'76',FL1'81'                DSNO FIELD
         DC   X'00',X'00',FL1'0',FL1'0'                  INDICATES END
*                                                         OF LINE
* THE SECOND LINE OF THE TAPE UCB IS DEFINED BY A SECOND SERIES OF
* FORMAT PATTERNS, TAPEL2.
```

Figure B-8. Sample Format Patterns

The Print Service Routine

The fourth fullword in the PRDMP parameter list is the address of a PRDMP-supplied print routine that your module must use to print the contents of the output buffer.

Your module must pass control to the print routine using standard linkage conventions. Upon entry to the print routine, register 1 must contain the address of the original PRDMP parameter list.

The print routine causes the output buffer pointed to by the parameter list to be printed and returns a new 121-character buffer (set to blanks) to your module.

PRDMP Output Buffer

In the PRDMP parameter list, the output buffer address points to a 121-character work area where a print line can be constructed. This buffer is blank on entry. The first 120 characters of the buffer are used as data for the line. The last character is a pad character to allow unpacking directly to the buffer. PRDMP supplies the carriage control character. When a print line is built, the print routine must be given control to cause the line to be printed. To print a blank line, your module must pass control with a blank buffer. The PRINT service routine will not print blank lines at the top of a new page. You define the output data set on the PRINTER DD statement.

In the example in Figure B-9, the subroutine, PRINT, uses the PRDMP-supplied subroutine.

```
* THIS SECTION OUTPUTS A LINE OF A TAPE UCB
OUTPUTIT   LA     R0,TAPEL2          SET ADDR OF LINE 1 FORMAT PATTERN
           BAL    RLINK,FORMAT       GO TO FORMAT LINE
           BAL    RLINK,PRINTIT      GO PRINT IT WITH SERV RTN
           BAL    RLINK,PRINTIT      GO PRINT A BLANK LINE
           B      NEXTLINE           GO TO PROCESS NEXT LINE
* THIS SECTION OF CODE IS USED TO CALL THE PRINT SERVICE
* ROUTINE. REGISTER 1 MUST CONTAIN THE ADDRESS OF THE ORIGINAL
* PRDMP PARAMETER LIST.
PRINTIT    L      R15,PARMPRNT       GET PRINT RTN ADR FROM PARM LIST
           BALR   R14,R15            GO TO PRINT ROUTINE
           BR     RLINK              RETURN TO SECTION OF CODE ABOVE
```

Figure B-9. Example Using the PRINT Routine

The Summary Dump Data Access Service Routine

The summary dump data access service routine (IEAVTFRD) allows your formatting routine to access the summary dump data contained in an SVC dump. IEAVTFRD is reenterable and can be loaded from SYS1.LINKLIB.

Your routine passes control to IEAVTFRD via the CALL macro, using standard linkage conventions. Register 1 must contain the address of the PRDMP parameter list. IEAVTFRD uses the eighth word in the parameter list during its processing. This word must be set to zero prior to the first invocation of IEAVTFRD, and must not be changed between invocations.

One summary dump record is read for each invocation of IEAVTFRD. The records are returned in the same order that they were dumped. IEAVTFRD reads the summary dump records sequentially from the beginning of the data to the end. The summary dump data is returned as variable-length records. Each record contains a header that describes the data in the record. This header is described by mapping macro IHASMDLR.

You should invoke IEAVTFRD to read all of the available records, rather than stopping when a particular record has been read. Storage obtained by

IEAVTFRD is not freed until the end of the summary dump data is reached (return code 8 or greater) or upon step termination. If all the records are not read, storage is not freed until step termination.

When IEAVTFRD returns control to your module, register 0 contains the address of a reconstructed summary dump record (that is, the record appears exactly as it did when the dump was taken). Register 15 contains a return code. For nonzero return codes, IEAVTFRD prints an explanatory message in the dump output.

Possible return codes from IEAVTFRD are:

Return Code	Explanation
0 -	Register 0 contains the address of a completely reconstructed summary dump record.
4 -	Register 0 contains the address of a partially reconstructed summary dump record. During summary dump processing, it was necessary to omit parts of this record. The missing portions of the record are replaced with asterisks (hexadecimal 5C).
8 -	No data is returned. The normal end of the summary dump data was reached.
12 -	No data is returned. The end of the summary dump data was reached before the expected normal end.
16 -	There is no summary dump data available in the dump.
20 -	The CVT could not be located in the dump.*
24 -	The GDA (global data area) could not be located in the dump.*
28 -	IEAVTFRD was unable to obtain sufficient storage for its processing.

*Information in the GDA is used to locate the summary dump data. The CVT is used to locate the GDA.

Specifying Format Routines for Summary Dump Records

The FMTTABLE table in CSECT IEAVTFTM of load module IEAVTFSD contains the names of format subroutines. These subroutines format the summary dump records for an SVC dump when the SUMDUMP control statement is specified. FMTTABLE contains an entry for each summary dump record identifier. (The identifiers are contained in the record header as described by mapping macro IHASMDLR.) The first entry is for the first identifier, the second entry is for the second identifier, and so on. Each entry is 28 bytes in length and contains the following fields:

- The 8-character load module name of the format subroutine to be loaded and called, or blanks
- The 4-byte address of a default format subroutine to be invoked if the load module name is blank or cannot be loaded
- 16 printable characters that describe the data associated with this summary dump record identifier.

IEAVTFSD passes control to the format subroutines using standard linkage conventions. Register 1 contains the address of a 3-word parameter list:

- The address of a pointer to the PRDMP parameter list
- The address of a pointer to the summary dump record to be formatted
- The address of the original virtual address of the data when it was accessed by summary dump processing

Prior to calling the format subroutines, IEAVTFSD provides a record title containing information from the summary dump record header and blank lines to separate the output of the format subroutines.

You can change the format subroutine for a particular summary dump record identifier in FMTTABLE. Use SPZAP to put the load module name of the new format subroutine into FMTTABLE (see Figure B-10 and Figure B-11), and add the load module to SYS1.LINKLIB or the equivalent. The load module name must be left-justified in the first eight bytes of the entry, and must be padded with blanks if the name has less than eight characters.

To determine the relative address of FMTTABLE, use the sample JCL in Figure B-10. The table can be found immediately following its printable identifier FMTTABLE.

```
//        EXEC PGM=AMASPZAP
//SYSPRINT DD SYSOUT=A
//SYSLIB   DD DSN=SYS1.LINKLIB,DISP=SHR
//SYSIN    DD *
     DUMPT IEAVTFSD IEAVTFTM
/*
```

Figure B-10. JCL to Locate FMTTABLE

The sample SPZAP control statements in Figure B-11 verify the table identifier (FMTTABLE), and verify and replace the module name portion of the second entry with the characters 'MODULE', assuming that the table identifier is at location 3EC. (The table identifier is eight bytes long, and each entry is 28 (hexadecimal 1C) bytes long.)

```
NAME IEAVTFSD  IEAVTFTM
VER  03EC  C6D4E3E3C1C2D3C5
VER  0410  4040404040404040
REP  0410  D4D6C4E4D3C54040
```

Figure B-11. Modifying FMTTABLE

Appendix C. 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
CAW	channel address word
CCH	channel check handler
CCT	common control table
CCW	channel command word
CDE	contents directory entry
CESD	composite external symbol dictionary
COM	common communication area
CS	control section name
CSCB	command scheduling control block
CSD	common system data area
CSECT	control section
CSID	channel set identifier
CSW	channel status word
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
ECT	exit control table
ECTE	exit control table entry
EID	event identifier
EOF	end of file
EOV	end of volume
EP	entry point name
EPA	entry point address
ERB	error recovery block

ESD	external symbol dictionary
FID	format identifier
FXTAB	fix table
GSMQ	global service manager 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
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	job name
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 manager 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
PCB	print control block
PCI	program controlled interruption
PDS	partitioned data set
PER	program event recording

PICA	program interruption control area
PRDMP	AMDPRDMP service aid program
PSW	program status word
PTF	program temporary fix
PTFLE	AMAPTFLE service aid program
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
SD	section definition
SDATA	service data area
SIO	start input/output
SLE	save list element
SLIP	serviceability level indication processing
SPZAP	AMASPZAP service aid program
SQA	system queue area
SR	subroutine
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

OS/VS2 MVS System Programming Library: Service Aids

GC28-0674-4

S370-37

Printed in U.S.A.

OS/VS2 MVS
System Programming
Library: Service Aids

GC28-0674-4

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. 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:

_____

_____

_____

_____

Note: Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.

Cut or Fold Along Line

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.)

OS/VS2 MVS System Programming Library: Service Aids

GC28-0674-4 S370-37

Reader's Comment Form

Cut or Fold Along Line

Fold and tape Please Do Not Staple Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department D58, Building 921-2
PO Box 390
Poughkeepsie, New York 12602

Fold and tape Please Do Not Staple Fold and tape

Printed in U.S.A.

IBM®