LY28-1199-4
File No. S370-37

Program Product

MVS/Extended Architecture Diagnostic Techniques

MVS/System Product:

JES3 Version 2	**5665-291**
JES2 Version 2	**5740-XC6**

IBM

Fifth Edition (June, 1987)

This is a major revision of LY28-1199-3. See the Summary of Amendments following the Contents for a summary of the changes made to this manual. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

This edition applies to Version 2 Release 2.0 of MVS/System Product (5665-291 and 5740-XC6) and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. The previous edition still applies to Version 2 Release 1.7 and may be ordered using the temporary order number LT00-2123. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370 Bibliography*, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product in this publication is not intended to state or imply that only IBM's product may be used. Any functionally equivalent product may be used instead.

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

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Information Development, Department D58, Building 921-2, PO Box 390, Poughkeepsie, NY 12602. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

This document contains restricted materials of International Business Machines Corporation.

Preface

This publication describes diagnostic techniques and guidelines for isolating problems on MVS systems. It is intended for the use of system programmers and analysts who understand MVS internal logic and who are involved in resolving MVS system problems.

This publication is intended for use only in debugging. None of the information contained herein should be construed as defining a programming interface.

Organization and Contents

This publication stresses a three-step debugging approach:

1. Identifying the external symptom of the problem.

2. Gathering relevant data from system data areas in order to isolate the problem to the component level.

3. Analyzing the component to determine the cause of the problem.

In support of this approach, the publication has been organized into three basic parts consisting of four sections and three appendixes as follows:

Part 1

"Section 1. General Introduction" describes the debugging approach that is used and defines the external symptoms that are used to identify a system problem.

"Section 2. Important Considerations" describes concepts and functions that should be understood prior to undertaking system diagnosis. Included are: global system analysis, system execution modes and status saving, locking, use of recovery work areas, effects of MP, trace analysis, debugging hints, and general data gathering techniques.

"Section 3. Diagnostic Materials Approach" provides guidelines for obtaining and analyzing storage dumps of data areas related to the problem.

Part 2

"Section 4. Symptom Analysis Approach" describes how to identify an external symptom (loop, wait state, TP problem, performance degradation, or incorrect output), and provides an analysis procedure for what kind of problem is causing the symptom.

Part 3

Appendixes

"Appendix A" provides a step-by-step approach to analyzing a stand-alone dump.

"Appendix B" provides diagnostic information for SVC dump titles.

"Appendix C" contains definitions of abbreviations used throughout the publication.

Referenced and Related Publications

The following publications either are referenced in this publication or provide related reading:

370-XA Principles of Operation		SA22-7085
MVS/System Product General Information		GC28-1118
MVS/Extended Architecture System Programming Library:		
Initialization and Tuning		GC28-1149
System Macros and Facilities (2 volumes)	GC28-1150,	GC28-1151
Service Aids		GC28-1159
System Modifications		GC28-1152
31-Bit Addressing		GC28-1158
MVS/Extended Architecture Operations:		
System Commands		GC28-1206
JES2 Commands		SC23-0064
JES3 Commands		SC23-0063
MVS/Extended Architecture Message Library:		
System Messages (2 volumes)	GC28-1376,	GC28-1377
System Codes		GC28-1157
JES2 Messages		GC28-1353
JES3 Messages		GC28-0062
Dump Output Terminal Messages		GC28-1336
MVS/Extended Architecture Debugging		
Handbook (6 volumes)	LC28-1164 through	LC28-1169
MVS/Extended Architecture SYS1.LOGREC Error Recording		GC28-1162
Environmental Record Editing and Printing (EREP) Program		
User's Guide and Reference		GC28-1378
MVS/Extended Architecture JES3 Diagnosis		SC23-0061
ACF/VTAM Version 2 Diagnosis Guide		SC27-0615
ACF/VTAM Version 2 Diagnosis Reference		LY38-3053
ACF/TCAM Version 2 Diagnosis Guide		SC30-3137
MVS/Extended Architecture System Logic Library (multiple volumes)		
Master Table of Contents and Index (has order numbers)		SY28-1600
MVS/Extended Architecture Service Aids Logic		LY28-1189
MVS/Extended Architecture System Initialization Logic		LY28-1200
MVS/Extended Architecture JES2 Logic		LY24-6008
MVS/Extended Architecture JES3 Logic		LY24-6007
MVS/Extended Architecture TSO Command Processor Logic, Volume IV		SY28-0652
MVS/Extended Architecture Resource Measurement Facility (RMF)		
Version 3 Program Logic Manual (3 volumes)	LY28-1170 through	LY28-1172
MVS/Extended Architecture Catalog Diagnosis Guide		LY26-3955
MVS/Extended Architecture BDAM Logic		LY26-3893
MVS/Extended Architecture CVOL Processor Logic		LY26-3895
MVS/Extended Architecture Open/Close/EOV Logic		LY26-3966
MVS/Extended Architecture SAM Logic		LY26-3967
MVS/Extended Architecture VIO Logic		LY26-3900
MVS/Extended Architecture VSAM Logic		LY26-3970
OS/VS2 MVS Mass Storage System Communication (MSSC) Logic		SY35-0013
OS/VS2 MVS VTIOC and TCAS Logic		SY27-7269
IBM 3704/3705 Communications Controllers Emulation Program		
Generation and Utilities Guide and Reference Manual		GC30-3008
IBM 3704/3705 Communications Controller NCP/VS		
Generation and Utilities Guide and Reference Manual		GC30-3007
Resource Access Control Facility (RACF): Program Logic Manual		LY28-0730
OS/VS2 MVS Programmed Cryptographic Facility:		
Program Logic Manual		LY28-0958
MVS/Extended Architecture Interactive Problem Control System (IPCS)		
User's Guide		GC28-1407
MVS/Extended Architecture Interactive Problem Control System (IPCS)		
Command Reference		GC28-1408

Contents

Figures

Summary of Amendments

Summary of Amendments
for LY28-1199-4
MVS/System Product Version 2 Release 2.0

This major revision contains changes to support MVS/System Product Version 2 Release 2.0.

The changes include:

- Throughout the book, as applicable, information that indicates the IPCS subcommands that can be used to perform part or all of the diagnostics described.
- Updated information on the recording control buffers (RCBs) used for SYS1.LOGREC messages and WTO messages.
- Information about the symptom record, which is a software record that can be initiated and written to SYS1.LOGREC by programs that are authorized to use the SYMREC macro instruction.
- Information to reflect that the MT statement in the SCHEDxx member of SYS1.PARMLIB defines the size of the master trace table.
- Information to reflect that SLIP commands can reside in the IEASLPxx, IEACMD00, and COMMNDxx members of SYS1.PARMLIB.
- The addition of the ACTION=STDUMP option on the SLIP command.
- A change to the ACTION=WAIT option on the SLIP command.
- New and changed SVC dump titles.

Additionally, several technical corrections and editorial changes were made throughout the book.

Summary of Amendments
for LY28-1199-3
MVS/System Product Version 2 Release 1.7

This major revision contains changes to support MVS/System Product Version 2 Release 1.7.

The changes include:

- Changes to fields in the address space control block (ASCB).
- For the Vector Facility Enhancement, additions to the machine check interruption code.
- For the Availability Enhancement, the addition of the AVMDATA control statement for formatting major availability manager control blocks, and the addition of several SVC dump titles.

Additionally, several technical corrections and editorial changes were made.

Summary of Amendments
for LY28-1199-2
as Updated December 31, 1985
by Technical Newsletter LN28-1107

This Technical Newsletter, which supports MVS/Extended Architecture Data Facility Product Version 2 Release 1.0, adds a new dump title to the "Appendix B. SVC DUMP Title Directory."

Summary of Amendments
for LY28-1199-2
as Updated December 27, 1985
by Technical Newsletter LN28-5098

This Technical Newsletter, which supports Version 2 Release 1.5 of MVS/System Product, includes HASPRAS as the issuing module of a $SDUMP macro instruction for JES2.

Additionally, several technical corrections were made.

Section 1. General Introduction

This section introduces basic MVS problem analysis and provides an overview of the interactive problem control system (IPCS).

Basic MVS Problem Analysis Techniques

Problem isolation and determination are significantly more complex in MVS than in previous operating systems because of:

- ***Enabled System Design*** which has made the internal and environmental status-saving functions more extensive than those of previous systems.
- ***Multiprocessing (MP)*** which potentially allows the execution of code in sequences not encountered in a uniprocessing (UP) environment. MP can also cause contention for serially reusable resources.
- ***Locking Mechanism*** which facilitates enabled system design and multiprocessing functions and maintains data integrity.
- ***Subsystems*** which are responsible for processing work requested from the system. They maintain their own work queues, control block structures and dispatching mechanisms - all of which must be understood in order to effectively pursue problems in the MVS operating system.
- ***Software Recovery*** which attempts to keep the system available despite errors.
- ***The number of components*** which provide new functions and whose internal logic must be understood for effective problem determination.

As a result of this complexity, MVS problem solvers have made two adjustments in their diagnostic outlook:

- Rather than learning the system logic at an instruction or module level, they have learned the system in terms of component interactions at the interface level.
- They have learned that the most effective problem analysis at a system level is obtained from a disciplined, almost formal, diagnostic approach.

This publication contains those debugging techniques and guidelines that have proven the most useful to problem solvers with several years experience in analyzing MVS system problems. These techniques are presented in terms of a debugging "approach" that can be summarized in three steps:

1. Identifying the external symptom of the problem.

2. Gathering relevant data from system data areas in order to isolate the problem to a component.

3. Analyzing the component to determine the cause of the problem.

The most important step in this approach is often the first - correctly identifying the external symptom of a problem. To do this, it is best to get a description of the problem as it was perceived by an eyewitness. You will want a description that provides a context from which to start, such as:

"System is looping; can't get in from console."
"Job abended with 213."
"I/O error on 251."
"Console locked out."
"Terminal hung, keyboard locked."
"System in wait, nothing running."
"Bad output."
"Job won't cancel."
"System degrading. Very slow."
"System died."
"0C4 in component abc."

The list is endless, of course. Your objective is to fit one (or more) of these descriptions to one of the following external symptoms.

- ***Enabled wait*** - The system is not executing any work and when it takes interrupts, nothing happens. Something appears to be stuck.

- ***Disabled wait*** - The system freezes with a disabled PSW that has the wait bit on. This can be either an explicit and intentional disabled wait or a situation that occurs because the PSW area has been overlaid.

- ***Disabled Loop*** - This is normally a small (fewer than 50 instructions) loop in disabled code.

- ***Enabled loop*** - This is normally a large loop in enabled code (and may include disabled portions - loops as a result of interrupts).

- ***Program check*** - The program is automatically cancelled by the system, usually because of improper specification or incorrect use of instructions or data in the program. If a SYSABEND, SYSMDUMP, or SYSUDUMP DD statement was included in the JCL for the job, a dump of the problem program will be taken.

- ***ABEND*** - The system issues an SVC 13 with a specific code from 1 to 4095 to indicate an abnormal situation.

- ***Incorrect output*** - The system is not producing expected output. Incorrect output can be categorized as: missing records, duplicate records or invalid data that has sequence errors, incorrect values, format errors, or meaningless data. If a program has apparently executed successfully, incorrect results will not be detected until the data is used at some future time.

- ***Performance degradation*** - A bottleneck or system failure (hardware or software) has severely degraded job execution and throughput.

- ***TP problem*** - A problem, usually detected by the operator or terminal user, that indicates malfunctions are affecting one or more terminals, lines, etc.

The chapters in Section 4 (Symptom Analysis Approach) will help you identify these symptoms. The main rule at this stage of your analysis is to proceed carefully. When first screening a problem, do not assume too much. Don't even assume that the original eye witness description was correct. Keep all initial information about the problem as a reference for your later analysis.

In the course of identifying the correct external symptom, you will begin gathering data that will lead you to other sections of the publication. Specific data gathering techniques are contained in Sections 2 and 3. Section 2 describes the major MVS debugging areas such as LOGREC records and recovery work areas. Section 3 describes how to use a storage dump effectively as your main source of diagnostic material. Appendix A contains a step-by-step procedure that can be used as a guide for analyzing a stand-alone dump.

Eventually you should have gathered enough data to isolate the problem to a particular component or process. For component-related diagnostic aids, refer to *System Logic Library* or other appropriate logic books or debugging guides. For information that is useful to diagnose problems during system initialization, such as the IPL diagnostic area, refer to *System Initialization Logic.*

Note: Before you begin using this publication for problem analysis, scan through it to find out where the various types of information are located. Depending on your current debugging skill level, various sections will be more important than others.

Always keep in mind that trouble-shooting a system of the internal complexity of MVS is not always an "If A, then B" procedure. The guidelines and techniques presented in this publication define "generally" what the analyst will discover. The nature of the debugging process is such that the problem solver does not perform the same analysis for every problem.

IPCS - Interactive Problem Control System

The interactive problem control system (IPCS) provides MVS installations with expanded capabilities for diagnosing software failures.

IPCS includes facilities for:

- Online examination of storage dumps using either line mode TSO or an IPCS/ISPF dialog management full screen facility.
- Analysis of key MVS system components and control blocks.

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.DUMPxx data sets.
- Virtual dumps produced by MVS SDUMP on data sets specified by the SYSMDUMP DD statement.

Where appropriate, this publication names the IPCS subcommand that you can use to do part or all of the diagnostics being described.

For information about IPCS, see the *Interactive Problem Control System (IPCS) User's Guide* and the *Interactive Problem Control System (IPCS) Command Reference*.

Section 2. Important Considerations

This section describes concepts and functions that are useful to problem analysis. It also contains miscellaneous debugging hints and general data gathering techniques.

The chapters in this section are:

Global System Analysis

In trying to isolate a problem to an internal symptom, a global system analysis often uncovers enough data to provide a starting point for the actual problem isolation and debugging. This chapter discusses the main considerations the analyst should be aware of when analyzing a stand-alone dump, including:

- The system areas that should be inspected to understand the current system state at the time of a dump
- The system areas that should be examined to understand the current state of the work in the system and the current disposition of storage and tasks

Global Indicators that Determine the Current System State

The following areas should be examined to help determine the current state of the system:

1. **PSA** - The prefixed save area is 4K bytes of key 0 storage that contains important system information. The second 2K bytes of the 4K area are fetch protected and can only be referenced by key 0 routines. The first 512 bytes are write protected (low-address protection). During NIP processing, the PSA(s) for the processor(s) are initialized and the prefix register(s) are initialized to point to them. The PSA(s) contain old and new PSWs, register save areas used by system routines, and pointers to other key system areas. There is one PSA for each processor in the configuration.

   You can use the IPCS subcommand VERBEXIT CPUDATA to format all PSAs, and the IPCS subcommand

   CBFORMAT 0 CPU(d) STRUCTURE(PSA)

   to format a single PSA.

   Special Notes About Stand-alone Dumps:

   - If you IPL the stand-alone dump program from the system control (CC-012) frame, it is not necessary to perform the STORE STATUS operation as noted in the following paragraphs. Status is automatically stored when stand-alone dump is invoked from the CC-012 frame and automatic store status is on.
   - Before taking a stand-alone dump, it is necessary to perform a STORE STATUS operation. This hardware facility does *not* use prefixing; instead it stores values such as the current PSW, registers, CPU timer, and clock comparator in the unprefixed PSA (the one used before NIP initialized the prefix register). The dump program subsequently saves these values and, in an MP environment, issues a SIGP instruction to other processors requesting a STORE STATUS operation. As a result, these values in the unprefixed PSA are *overlaid* by another processor's values.

     Therefore, in an MP environment the status in the unprefixed PSA is always that of a non-IPLed processor, not the one on which the stand-alone dump was IPLed.

- The IPLing of the stand-alone dump program causes absolute storage (X'0'-X'18' and storage beginning at X'110') to be overlaid with CCWs. You should be aware of this and not consider it as a low storage overlay.

- In an MP environment, the STORE STATUS operation must be performed only from the processor to be IPLed for the stand-alone dump program.

- IPLing the stand-alone dump program twice causes the storage dump to contain a dump of the dump program itself because it was read in for the first IPL. This causes the dump program to overlay a certain portion of real storage and the general purpose registers to contain values associated with the stand-alone dump program and *not* MVS.

- If the operator does not issue the STORE STATUS instruction before IPLing a stand-alone dump, the message "ONLY GENERAL PURPOSE REGS VALID" might appear on the formatted dump. The PSW, control registers, etc., are not included. This greatly hampers the debugger's task.

2. **Registers and PSW** - The current PSW and the general, the control, and the floating point registers associated with each processor help to determine the program executing on each processor. (The print dump program formats these items.)

   You can use the IPCS subcommand STATUS CPU REGS WORKSHEET to get an indication of what is executing.

   If the current PSW is 070E0000 00000000 and the GPRs are all 0, you are in the no-work wait condition, which indicates no ready work is available for this processor to execute.

   If there is or should be work remaining, an invalid wait condition results. (Refer to the chapter on "Waits" in Section 4.)

   If the registers are not equal to zero and the PSW does not contain the wait bit (X'0002'), there is an active program. If the wait task is dispatched, the system is in the no-work wait condition.

   In the PSW - when bit 32 (the A bit) is 0, the program in control is executing in 24-bit addressing mode; when bit 32 is 1, the program in control is executing in 31-bit addressing mode.

3. **ILC/CC** - These fields are at location X'84' for external interrupts; location X'88' for SVC interrupts; location X'8C' for program interrupts. These fields indicate the last type of interrupt associated with each interrupt class for each processor. The work active when each interrupt occurs is represented by the old PSWs at locations: X'18' (external); X'20' (SVC); X'28' (program). Common contents of these fields are:

   X'84' - 00001004 clock comparator
   - 00001005 CPU timer
   - 000x1201 SIGP-emergency signal
   - 000x1202 SIGP-external call

   Where x indicates the processor issuing the SIGP instruction.

X'88' - 000200xx where xx is the SVC number. This field should be inspected for unusual SVCs such as:

1 - WAIT:	can indicate an enabled wait situation
D - ABEND:	can indicate program error processing
F - ERREXCP:	can indicate a problem in I/O error processing
10 - PURGE:	can indicate a problem in the swap process
38 - ENQ:	can indicate a resource contention problem
4F - STATUS:	can indicate a non-dispatchability problem

X'8C' - 000x0011 indicates a page fault and 000x0010 indicates a segment fault interrupt. Anything other than a code of 10 or 11 must be inspected further. Also with a code of 10 or 11, the program check old PSW (location X'28') must be enabled (mask = X'07') because disabled page faults or segment faults are not allowed in MVS and it is an error if one occurs.

- 000x0016 indicates a trace table overflow.

- 00040040 indicates a monitor call (if GTF is active).

4. **PSA + X'204' (CPU ID)**

5. **PSA + X'208' (address of PCCA - 1 per processor)** - The PCCA contains information about the physical facilities of a processor.

6. **PSA + X'210' (address of LCCA - 1 per processor)** - The LCCA contains many of the status-saving areas that were located in low storage in previous systems. It is used for software environment saving and indications. The registers associated with each of the interrupts you find in the PSA are saved in this area. In addition, the system mode indicators for a processor are maintained in the LCCA.

7. **PSA + X'224' (PSAAOLD)** - This is the address of the ASCB of the work last dispatched on each processor. This field indicates the address space that is currently executing.

   You can use the IPCS subcommand STATUS WORKSHEET to get a summary of the PSA fields.

8. **PSA + X'21C' (PSATOLD)** - This is the address of the TCB of the work last dispatched on each processor. This field in conjunction with PSAAOLD isolates to a task within an address space. **Note:** PSATOLD = 0 when SRBs are dispatched.

9. **PSA + X'228' (PSASUPER)** - This is a field of bits that represent various supervisory functions in the system. If a loop is suspected, these bits should be checked in an attempt to isolate the looping process.

   Note: Because of SRM timer processing in MVS, the external first level interrupt handler bit (X'20') or the dispatcher bit (X'04') may be set in this field even in the enabled wait situation.

10. **PSA + X'2F8' (PSACLHS)** - This field indicates the current locks held on each processor. Knowing which locks are held helps isolate the problem, especially in a loop situation. By determining the lock holders you can isolate the current process. (See the chapter on "Locking" later in this section.)

11. **PSA+X'380' (PSACSTK)** - This is the address of the active recovery stack which contains the address of the recovery routines to be routed control in case of an error. If the address is other than X'C00' (normal stack), the type of stack (for example, program check FLIH or restart FLIH) is meaningful, especially in the loop situation.

    You can use the IPCS subcommand STATUS CPU to get an FRR stack summary.

    If the pointers to the current entry and the empty entry are not equal, you might get an idea of the current processing by searching the normal stack (X'C00') and associating the recovery routine to the active mainline routines.

    Note: If a loop is suspected, scan the first word following each routine's address in the current stack. A X'80' indicates that the routine is in control. A X'40' indicates that the routine is a nested recovery routine. And, a X'08' indicates that the routine is not allowed to retry.

    If X'28' into the stack is non-zero, also check for an SDWA address in the RT1WRTCA field of the RTM1WA associated with the active stack This block is mapped by the SDWA DSECT and is described in the *Debugging Handbook* (RTCA and SDWA are different names for the same control block). If an SDWA address is present, an error has occurred and it can be related to the problem you are analyzing. If trapping via RTM's SLIP facility, the registers at entry to RTM are contained in this area.

Work Queues, TCBs and Address Space Analysis

For information about how to use IPCS for online examination of control blocks, see the *Interactive Problem Control System (IPCS) User's Guide*.

Examine the following areas to help determine the current state of work in the system.

TCB Summary

The TCB summary report contains a summary of the address spaces and their associated tasks. You can use the IPCS subcommand SUMMARY to get this report (or you can use AMDPRDMP, the print dump program).

A quick scan of the completion (CMP) field for each task reveals any abnormal terminations that have occurred. Discovery of an error completion code warrants further investigation as to the cause. Remember, however, that these codes are residual and the job or task might have recovered from the problem.

Also investigate multiple abnormal completion codes which all relate to the same area of the system, or many tasks that all have the same completion code. These completion codes can all relate to one area of the system and perhaps to the problem you are investigating. Again, LOGREC should provide further documentation in an error situation such as this.

You can use the IPCS subcommand VERBEXIT LOGDATA to format the in-storage LOGREC buffer.

SRB Dispatching Queues

The print dump program formats the SRB dispatching queues. Elements on any of these queues should be investigated, especially in cases where no work appears to be progressing through the system.

You can use the IPCS subcommand SUMMARY FORMAT to format the SRBs.

Elements on the global, SVT, or address space local services management queues (SVTGSMQ, SVTLSMQ, or ASCBLSMQ) can indicate that the dispatcher has not received control since these SRBs were scheduled. This is an unusual condition that should be investigated to determine why the SRBs have not been dispatched.

Elements on the global/local service priority lists (GSPLs/LSPLs) should be explained. It is possible the dump was taken before the SRB routines were able to execute. But it more likely indicates some other system problem such as an enabled wait or disabled loop. If there are SRBs on an LSMQ/LSPL, you should determine if the associated address space is swapped into storage and if it is not, why not. (Possible causes are real frame shortage or a problem in the paging/swapping mechanism.) Again this is an indication of a potential system problem. The chapter on "Waits" in Section 4 contains additional information on the dispatching queues.

If, at this point, you can isolate the problem to a component, refer to diagnostic information for the component in the *System Logic Library* or other appropriate logic books or diagnosis guides.

Address Space Analysis

If you have isolated the error to a given address space or want to determine the state of a given address space, analyze the ASCB.

You can use the IPCS subcommand CBSTAT ascb-addr STRUCTURE(ASCB) to get the status of the address space.

Important indicators in the ASCB are:

- ASCBLOCK (ASCB+X'80') - to determine the specific state of the local lock. If it contains 7FFFFFFF, FFFFFFFF, or 4FFFFFFF (the lock suspend/interrupt/ready-to-run IDs), refer to the chapter on "Locking" later in this section for an explanation.

  You can use the IPCS subcommand ANALYZE to do suspend lock analysis.

  Note: When holding a suspend lock, a routine can only be suspended because it attempts to obtain an unavailable cross memory services lock or because of a page fault, segment fault, synchronous page fix, or the routine specifies SUSPEND=YES on the SDUMP macro. To find the reason for the suspension, refer to the discussion of Task Analysis later in this chapter and to the chapter on "Locking" later in this section.

- ASCBEWST (ASCB+X'48') - to determine the TOD clock value when the address space last executed. This field helps you determine how long an address space has been swapped-out. By subtracting this field from the last timer value in the system trace table and converting to seconds, you can discover the approximate swap-out time. (See the chapter "MVS Trace Analysis" later in this section.)

- ASCBTNEW (ASCB+X'1C') - identifies highest-priority TCB that is dispatchable. Explicit wait sets a non-dispatchability flag (TCBFLGS4=X'04').

- ASCBCPUS (ASCB+X'20') - number of processors running tasks in this address space.

- ASCBTRQP (ASCB+X'2C') - dispatching queue forward pointer.

- ASCBRCTF (ASCB+X'66'), ASCBFLG1 (ASCB+X'67') - current status of the address space.

- ASCBASXB (ASCB+X'6C') - pointer to the ASXB that anchors the TCBs.

- ASCBDSP1 (ASCB+X'72') - address space non-dispatching flags.

- ASCBSRBS (ASCB+X'76') - number of SRBs currently suspended in the address space.

- ASCBOUCB (ASCB+X'90') - pointer to the OUCB, which is helpful when determining why an address space is swapped-out.

- ASCBFMCT (ASCB+X'98') - number of real frames currently occupied by the address space.

- ASCBTCBS (ASCB+X'D8') - number of ready TCBs not requiring the local lock.

- ASCBTCBL (ASCB+X'DC') - number of ready TCBs requiring the local lock.

- ASCBLOCI (ASCB+X'E8') - contains either the ASCB address of the address space holding this ASCB's local lock as a CML lock, or zero.

- ASCBCMLH (ASCB+X'EC') - contains either the address of the suspended TCB or SSRB holding this ASCB's local lock, or zero. The high-order bit on in field ASCBCMLH indicates an SSRB.

- ASCBRSM (ASCB+X'178') - contains either the address of the RSM address space block (RAB), or zero.

Task Analysis

After you understand the ASCB, you should analyze the associated task structure. Once again, scan the TCBs associated with your address space and look for an abnormal completion field. While doing so, check the RB structure for each task. Remember that the region control task, dump task, and STC/LOGON are represented by the first three TCBs. "Normally" they will be waiting during task execution. If one of them is not, you should determine why.

Assuming the first three TCBs are not obvious problem areas, continue inspecting the remaining TCBs. You are trying to explain each RB. Starting with the last RB created (the first RB, pointed to by the TCB+0), determine what work is represented. If work is waiting, find out why.

You can use the following IPCS subcommands when doing task analysis:

- CBSTAT tcb-addr ASID(asid) STRUCTURE(TCB) to get the status of the task.
- CBFORMAT tcb-addr ASID(asid) STRUCTURE(TCB) to format a single TCB.
- SUMMARY FORMAT ASID(asid) to format all TCBs in an address space.

Note: The master scheduler address space has system task TCBs that differ from other address spaces. Refer to the diagrams for Master Scheduler Initialization, Start Initiator, and Job Execution in the topic "General System Flow" in the *Debugging Handbook,* Volume 1 for details of the TCB structures.

The RBOPSW indicates the issuer of an explicit WAIT. TCBFLGS4 indicates an explicit WAIT. If it is not an explicit WAIT, consider the following suspension possibilities and their associated key indicators:

1. If ASCBLOCK=X'7FFFFFFF', X'FFFFFFFF', or X'4FFFFFFF', the status (registers and PSW) of the suspended, interrupted, or ready-to-run task is saved in the IHSA of the locally locked address space (ASCB+X'6C' points to ASXB; ASXB+X'20' points to IHSA).

   You can use the IPCS subcommand

   CBFORMAT ihsa-addr STRUCTURE(IHSA)

   to format a single IHSA.

   The IHSA is serialized by the address space's local lock. The reason for suspension is important. If it is for a lock, find out what address space or task owns that lock and what the owners' state is. (The chapter on "Locking" later in this section shows how to determine lock owners.) If it is for a page fault, segment fault, or synchronous page fix, determine the state of that page fault, segment fault, or synchronous page fix. Note also that while the RBTRANS field points to the address that caused the page fault, the RBWCF is 0.

An SRB can be suspended requesting an unavailable suspend lock (local or cross memory services), or because of a page fault, segment fault, or a page fix. Once an executing SRB is suspended for any of the above reasons, an SSRB (see the *Debugging Handbook*) is constructed. Also, an SRB can be delayed by the dispatcher if the local lock is unavailable and the SRB is to receive control with the local lock held. No status is saved for a delayed SRB; instead the SRB is placed on the local lock suspend queue. If suspended for a page fault or segment fault, the SSRB is pointed to by the corresponding PCB+X'20'. If suspended for a page fix, the SSRB is pointed to by the corresponding FCB+X'18'. PCBs can be found in the following locations:

- For pages with I/O in progress, PCBs are pointed to by the PFTE for the frame backing the page (PFTPCB, PFT+X'18').
- For cross memory page faults or segment faults, or common area page faults, PCBs are queued on the notification PCB queue for the home address space. The NPQ is anchored in the RAB for the home address space (RABNPQF, RAB+X'6C').
- For pages that cannot be backed because a frame is not available, PCBs are queued on the defer PCB queue. The DPQ is anchored in the RIT (RITDPQF, RIT+X'8C').

For a cross memory services lock request, the SSRB is on the requested cross memory services lock's suspend queue. See the chapter on "Waits" in Section 4 for details on how to locate the SSRB. For Local lock suspensions, the SRBs and SSRBs are chained together on a queue anchored in the ASCB field ASCBLSQH (ASCB+X'84').

You can use the IPCS subcommand ANALYZE to do suspend lock analysis.

A locked TCB can be suspended for the same reasons as an SRB. The save area is the IHSA of the locally locked address space (described in the *Debugging Handbook*). The TCBLLH (TCB+X'114') is set to X'01' if the task was locally locked at the time of the page fault.

The IHSA is valid for a cross memory services lock suspension if the ASCB is on the cross memory services lock's suspend queue. The CMSSMF lock suspend queue header is at label CMSFRSQH, the ENQ/DEQ cross memory services lock suspend queue is at label CMSEDLK+X'4', and the general cross memory services lock suspend queue is at label CMSSQH in CSECT IEAVESLA. If there is a page fault, the TCB could be suspended while holding both the local lock and at least one cross memory services lock. An indication of this is that the flag for cross memory services locks (ASCB+X'2A') is turned on, and the ASCB address is in at least one of the cross memory services locks. The cross memory services lockword contains the ASCB address of the locally locked address space. The requester of the cross memory services lock can own a cross memory local (CML) lock.

Note: The local and cross memory services lock bits in ASCBHLHI are set at suspend and are never reset.

2. If ASCBLOCK = X'00000000' and the memory/task is waiting, the status is saved in the RB/TCB. (See the chapter on "System Execution Modes and Status Saving" later in this section.)

   A task can issue the SUSPEND macro to cause the wait count of an RB to be non-zero. The RBOPSW indicates the issuer of a SUSPEND RB=CURRENT request. However, an RB can also issue SUSPEND RB=PREVIOUS. In this case, the only clue to the issuer is the interrupt code in the RB. If the interrupt code indicates a type 2, 3, or 4 SVC, then this SVC routine could have issued the suspend for this RB; but it is also possible that some IRB had executed on behalf of the task and issued a suspend for its previous RB.

   If the RBOPSW does not indicate the issuer of a WAIT or SUSPEND, and the RB is not in either page fault wait, segment fault wait, or page fix wait, then a SUSPEND RB=PREVIOUS might have been issued.

   Note: The explicit wait flag in the TCB (in TCBFLGS4) will not be on for a suspended RB.

3. Suspended SRBs can cause bottlenecks. The chapter on "System Execution Modes and Status Saving" can aid in locating any suspended SRBs that relate to the address space.

   Note: Do not spend time looking for them unless other facts about the problem indicate a potential problem in this area.

By far the most important consideration in task analysis is the RB structure of each task. Generally if you have isolated the problem to an address space, RB analysis shows a potential problem in the way of:

- Long RB chains
- Contention caused by an ENQ (SVC 38) request
- Page fault, segment fault, or synchronous page fix waits
- I/O waits
- Abnormal termination processing, that is, SVC D RB

After you have analyzed the RB structure, you might want to go back and further analyze the TCBs. Following are additional important fields in the TCB:

1. TCBFLGS (TCB+X'1D') - indicators of how the system currently considers this task.

2. TCBGRS (TCB+X'30') - general purpose registers (0-15) saved when a TYPE 1 SVC is issued or for an interruption for a non-locked task.

3. TCBSCNDY (TCB+X'AC') - additional system indicators for this task that help to determine why this task is not executing.

4. TCBRTWA (TCB+X'E0') - pointer to the RTM2 work area (mapped in the *Debugging Handbook*) which contains information similar to the SDWA but also data for RTM processing.

Summary

This chapter contains major considerations you must be aware of when analyzing a stand-alone dump in MVS. A disciplined approach is important; resist the tendency to go off on tangents upon finding the first unexplainable condition. After gathering all the facts, try to resolve the "cause and effect" situations you are bound to uncover. Generally, at this point you will have isolated the error and can start a detailed component/process analysis.

System Execution Modes and Status Saving

MVS has multiple execution modes. Status is saved and restored from many different locations depending upon the execution mode at the time control was lost. This chapter explains those modes and how they affect problem analysis.

System Execution Modes

MVS has four execution modes:

- Task mode
- SRB mode
- Physically disabled mode
- Locked mode

Code always executes in one of these modes or, in certain cases, in a combination of modes. For instance, code running in task or SRB mode can also be either locally locked or physically disabled.

You can use the IPCS subcommand STATUS CPU to determine the mode of the unit of work on each CPU.

In general, the supervisor dispatches units of work according to the following priority: SRB, locked, and task mode. Because a unit of work that is disabled is already executing, disabled mode work is not dispatched as such.

When a unit of work is running, the locally locked ASCB is found through PSALOCAL or PSAAOLD. If PSALOCAL = 0 and PSACLHS indicates that a local lock is held, then PSAAOLD points to the locked ASCB. If PSALOCAL ≠ 0 then PSALOCAL points to the CML locked address space.

In conjunction with the four execution modes, a unit of work can execute in cross memory mode. Cross memory mode is defined by control registers 3 and 4 and the PSW S-bit. The S-bit (bit 16 of the PSW) indicates whether current addressability is to the primary address space (S-bit = 0) or the secondary address space (S-bit = 1). The primary and secondary address spaces are defined by the ASIDs in control registers 3 and 4. The home address space is the address space in which the unit of work resides (indicated by PSAAOLD) when that unit of work is executing. When primary and secondary addressability is to the home address space and the S-bit = 0, then the unit of work is not in cross memory mode.

Task Mode

Task mode describes code that is executing in the system because the dispatcher selected work from the task control block (TCB) chain or from the interrupt handler save area (if the interrupted TCB held a local lock). To start execution, the dispatcher sets up the environment (registers, PSW, cross memory state, PCLINK stack, and FRR stack) and then passes control to the code to be executed.

1. Information for an unlocked task dispatch environment is found as follows:

TCB+X'30' (TCBGRS)	- General purpose registers.
TCB+X'0' (TCBRBP)	- Address of the RB.
RB+X'10' (RBOPSW)	- Old PSW.
RB-X'20' (RBXSB)	- Address of the XSB.
XSB+X'8' (XSBXMCRS)	- Cross memory status.
XSB+X'18' (XSBSTKE)	- PCLINK stack header.
TCB+X'E4' (TCBNSSP)	- Address of the NSSA.
NSSA+X'8' (NSSAFRRS)	- FRR stack for an enabled unlocked task mode FRR.

2. Information for a locally locked or a CML locked task dispatch environment is found in the locally locked address space as follows:

   - From the ASCB of the locally locked address space:

ASCB+X'E8' (ASCBLOCI)	- Contains either the address of the ASCB holding this ASCB's local lock as a CML lock, or zero if this ASCB's local lock is held as a LOCAL lock.
ASCB+X'EC' (ASCBCMLH)	- Address of the TCB holding this ASCB's local lock.

   - From the task holding a local lock:

TCB+X'E8' (TCBXLAS)	- Contains either the address of the ASCB of the locally locked address space, or zero if holding the LOCAL lock.
ASCB+X'6C' (ASCBASXB)	- Address of the ASXB.
ASXB+X'20' (ASXBIHSA)	- Address of the IHSA.
IHSA+X'38' (IHSAGPRS)	- General purpose registers.
IHSA+X'10' (IHSACPSW)	- PSW for the redispatched task.
IHSA+X'80' (IHSAXSB)	- Address of the XSB.
XSB+X'8' (XSBXMCRS)	- Cross memory status.
XSB+X'18' (XSBSTKE)	- PCLINK stack header.
IHSA+X'88' (IHSAFRRS)	- FRR stack.

Task mode is probably the most common execution mode. All programs given control via ATTACH, LINK, and XCTL operate in this mode.

SRB Mode

SRB (service request block) mode describes code that is executing in the system because the dispatcher found an SRB on one of the SRB queues. SRB set-up is started by the SCHEDULE macro. SCHEDULE is a macro that calls a service routine, which places the requestor-furnished SRB directly on the queue. SRBs are generally placed on the service management queue (SMQ), unless both the SMQ and the service priority list (SPL) are empty, in which case the SRB is placed on the SPL. The global services management queue (GSMQ) is located at SVTGSMQ (SVT+X'20'). It is also pointed to by CVTGSMQ (CVT+X'264'). The global service priority list (GSPL) is located at SVTGSPL (SVT+X'24') and can also be found from CVTGSPL (CVT+X'26C'). The SVT local service management queue (LSMQ) is located at SVTLSMQ (SVT+X'28'), and can be found from CVTLSMQ (CVT+X'268'). Finally, there is one local SMQ and one local SPL per address space. ASCBLSMQ is located at ASCB+X'D0', and

ASCBLSPL is located at ASCB+X'D4'. An SRB scheduled globally for a swapped-out address space is moved to one of the local queues.

SRBs are selected from the SPLs by the dispatcher in order to start execution. The dispatcher loads registers 0, 1, 14, and 15 from information in the SRB and builds the PSW. The PSW key and address are the responsibility of the scheduler of the SRB and are specified in the SRB. SRB mode has the characteristics of being enabled, supervisor state, key requested and non-preemptable. Non-preemptable means that the interrupt handler should return control to the interrupted service routine (code running under SRB mode). However, service routines can be suspended because of a page fault, segment fault, or because a lock (cross memory services or local) is unavailable.

SRB is interrupted. SRBs are non-preemptable. The registers, PSW, and cross memory status are saved in the PSA and/or LCCA during interrupt processing. When the system has handled the interrupt, the SLIHs return to the FLIHs, the status is restored, and control is returned to the interrupted SRB routine.

SRB is suspended. SRBs that are suspended must have their status saved in a unique area. The process that suspends an SRB is responsible for obtaining an SSRB (suspended SRB) and XSB (extended status block), which will contain the interrupted status used to reschedule the service routine once the reason for suspension has been resolved. See "Locating Status Information in a Storage Dump" later in this chapter for a detailed description of how to find these SSRBs and XSBs.

Physically Disabled Mode

Disabled mode is reserved for high-priority system code whose function is the manipulation of critical system queues and data areas. It is usually combined with supervisor state and key 0 in the PSW, and assures that the routine running disabled is able to complete its function before losing control. It is restricted to just a few modules in MVS (for example, interrupt handlers, the dispatcher, and programs holding a global spin lock).

Physically disabled mode is used for one of two reasons:

1. To assure that data remains static while the code is referencing or updating the data.

2. To assure that non-reentrant code does not lose control while performing critical system functions. For example, IOS must run disabled while enqueueing and dequeueing requests to UCBs and while updating UCBs at the start and end of I/O operations.

In the MVS system, physical disablement on a system basis because of MP must be accompanied by locking in order to guarantee serialization. MVS disabled code is usually accompanied by either a global spin lock or code executing under a "super bit." The "super bits" are located in each processor's PSA (X'228'). They are used primarily for recovery reasons - they allow RTM to recognize that a disabled supervisory function was in control at the time of error even though global locks were not held. This indicates that FRR recovery processing should be initiated by RTM.

Note that type 1 SVCs do not execute disabled in MVS. Instead they are entered with the local lock. Thus they are considered to be task mode physically enabled, holding the local lock.

Type 6 SVCs execute disabled. They are considered to be logical extensions of the SVC FLIH and execute with all the restrictions (that is, cannot page fault, etc.) of a disabled function.

Locked Mode

Locked mode describes code executing in the system while owning a lock. (See the chapter on "Locking" later in this section.) A lock can be requested during any execution mode (SRB, TCB, physically disabled).

Status saving while in a locked mode requires unique considerations from the system. An example is a program that invokes a type 1 SVC, such as EXCP or WAIT, that executes in locked mode. When a type 1 SVC is enabled, it can be interrupted. However, if the SVC is interrupted, the registers cannot be saved in the TCB because it is being used to save registers active at the time of the SVC request for return to the requestor. Therefore, status must be saved elsewhere.

Status saving while in locked mode is described under the previous topics "Task Mode" and "SRB Mode."

Determining Execution Mode from a Stand-Alone Dump

Knowing the system's execution mode at the time a stand-alone dump was taken is important in analyzing a disabled coded wait state or a loop. You can use the IPCS subcommand STATUS CPU WORKSHEET to get information about execution modes and a summary of useful fields from the PSA and LCCA. The following areas may help determine the mode of execution:

LCCA Indicators

There is an important dispatcher flag byte at LCCA+X'21D'. For a global SRB, the LCCAGSRB and LCCASRBM flags are set on. For a local SRB, only the LCCASRBM flag is set on.

PSA Indicators

- Super Bits - Flags in the supervisor control field located at PSA+X'228' (PSASUPER) indicate whether the dump was taken while in one of the interrupt handlers or dispatcher. The dispatcher's super bit is left on when the wait task is dispatched.

- PSAMODE - PSA+X'49F' indicates the mode of the system:

  X'00' - Task mode
  X'04' - SRB mode
  X'08' - Wait mode
  X'10' - Dispatcher mode
  X'20' - Non-preemptive bit (can be on with any of the above bits)

- Recovery Stack - If the first two words of the RTM stack vector table (PSA+X'380') are *not* equal, then control is in one of the interrupt handlers

or the dispatcher. The dispatcher stack is current when the wait task is dispatched. Compare the address at PSA+X'380' with each entry in the FRR stack vector table starting at PSA+X'384' to determine the owner of the active stack. (See the chapter on "Use of Recovery Work Areas for Problem Analysis" later in this section for stack vector table analysis.)

- Current Work - PSA+X'218' contains the addresses of the new TCB, old TCB, new ASCB and old ASCB consecutively in a four-word area. If the system is in SRB mode, the address of the old TCB equals 0. If the addresses of the new and old ASCBs are *not* equal, then the stand-alone dump was taken between the time that an address space switch was requested and the time that the dispatcher dispatched an address space, a global SRB was dispatched, or the wait task was dispatched. In all cases, the old TCB and ASCB indicate the current work.

- Locks - The PSA also contains the lock indicators. (See the chapter on "Locking" later in this section for a description of how to determine the lock mode.)

ASCB Indicators

You can use the IPCS subcommand

CBFORMAT ascb-addr STRUCTURE(ASCB)

to format an ASCB.

The following ASCB locations help determine execution mode:

X'66-67' - RCT flags.

X'72-73' - Non-dispatchability flags.

X'76' - Count of SRBs suspended in this address space.

X'80' - Local lock (see "Locking" later in this section for how to interpret this field when ≠0).

X'84' - Address of the SRB suspend queue for local lock requestors.

X'D0' - Local service management queue (contains SRBs that have not been staged). When the high-order bit of this field is 1, it indicates that a "user-ready" SYSEVENT is required to swap in the address space.

X'D4' - Local service priority list (contains SRBs that have been staged).

X'D8' - Number of ready TCBs that do not require the local lock.

X'DC' - Number of ready TCBs that require the local lock.

X'E8' - Contains either the ASCB address of the address space holding this ASCB's local lock as a CML lock, or zero. If nonzero, then the lock is owned by a unit of work in the address space pointed to by this field. If zero, then the lock is not owned or is owned by a unit of work within this address space.

X'EC' - Contains either the address of the suspended TCB or SSRB that is holding this ASCB's local lock, or zero. The high-order bit (bit 0) on indicates an SSRB.

Keep in mind that mixed modes frequently occur. For example, a local SRB can obtain a lock, be interrupted, and the stand-alone dump taken while disabled in the I/O supervisor. Depending on the system mode at the time of the interrupt, a task's status (registers, PSW, etc.) can be saved in one of several places.

Locating Status Information in a Storage Dump

Status information is located in a storage dump depending on the conditions under which it was saved.

Task and SRB Mode Interruptions

Status saving is required whenever the code gives up control, whether voluntarily or involuntarily. Initial status is saved by the first level interrupt handler (FLIH) as follows:

SVC FLIH - Initially:

- Registers 7-9 saved at PSA+X'8F0' (PSAGPREG)
- If an error condition is found, registers saved at LCCA+X'380' (LCCASGPR)

Then for all SVCs, status is saved in the TCB and the requestor's RB and XSB:

- Registers 0-15 saved at TCB+X'30' (TCBGRS)
- PSW saved at requestor's RB+X'10' (RBOPSW)
- Cross memory status saved at XSB+X'8' (XSBXMCRS)
- PCLINK stack header saved at XSB+X'18' (XSBSTKE)

Then for Type 2, 3, and 4 SVCs:

- Registers 0-15 saved at SVRB+X'20' (RBGRSAVE)

I/O FLIH - Initially:

- Registers 0-15 saved at PSA+X'868' (PSAIOGPR)

Then for unlocked tasks, status is saved in the TCB, RB, and XSB:

- Registers 0-15 saved in TCB+X'30' (TCBGRS)
- PSW saved at RB+X'10' (RBOPSW)
- Cross memory status saved at XSB+X'8' (XSBXMCRS)

For locally locked tasks, status is saved in the IHSA and XSB of the locked address space:

- Registers 0-15 saved at IHSA+X'38' (IHSAGPRS)
- PSW saved at IHSA+X'10' (IHSACPSW)
- Cross memory status saved at XSB+X'8' (XSBXMCRS)

For SRBs and non-preemptive TCBs:

- Register 0-15 saved at PSA+X'868' (PSAIOGPR)
- PSW saved at PSA+X'38' (FLCIOPSW)
- Cross memory status saved at PSA+X'5A8' (PSAIOXMS)

External FLIH - Initially:

- Registers 8-10 saved at PSA+X'950' (PSASLSA)

Then for locally locked tasks, status is saved in the IHSA and XSB of the locked address space:

- Registers 0-15 saved at IHSA+X'38' (IHSAGPRS)
- PSW saved at IHSA+X'10' (IHSACPSW)
- Cross memory status saved at XSB+X'8' (XSBXMCRS)

For unlocked tasks, status is saved in the TCB, RB, and XSB:

- Registers 0-15 saved at TCB+X'30' (TCBGRS)
- PSW saved at RB+X'10' (RBOPSW)
- Cross memory status saved at XSB+X'8' (XSBXMCRS)

For SRBs and non-preemptive TCBs:

- Registers 0-15 saved at LCCA+X'630' (LCCAXGR1)
- PSW saved at LCCA+X'618' (LCCAXPSW)
- Cross memory status saved at LCCA+X'620' (LCCAXXM1)

If first recursion:

- Registers 0-15 saved at LCCA+X'E0' (LCCAXGR2)
- PSW saved at LCCA+X'748' (LCCAXPS2)
- Cross memory status saved at LCCA+X'3C8' (LCCAXXM2)

If second recursion:

- Registers 0-15 saved at LCCA+X'120' (LCCAXGR3)
- PSW remains at PSA+X'18' (FLCEOPSW)
- Cross memory status saved at LCCA+X'3D0' (LCCAXXM3)

Program check - Initially:

For nonrecursive program interruptions:

- Registers 0-15 saved at LCCA+X'48' (LCCAPGR2)
- PSW saved at LCCA+X'88' (LCCAPPSW)
- ILC/PINT saved at LCCA+X'90' (LCCAPINT)
- TEA saved at LCCA+X'94' (LCCAPVAD)
- Cross memory status saved at LCCA+X'1C0' (LCCAPXM2)

For recursive program interruptions:

- Registers 0-15 saved at LCCA+X'08' (LCCAPGR1)
- PSW saved at LCCA+X'6B0' (LCCAPPS1)
- ILC/PINT saved at LCCA+X'6B8' (LCCAPIC1)
- TEA saved at LCCA+X'6BC' (LCCAPTE1)
- Cross memory status saved at LCCA+X'1B8' (LCCAPXM1)

For monitor call interruptions that occur during page fault or segment fault processing:

- Registers 0-15 saved at LCCA+X'A0' (LCCAPGR3)
- PSW saved at LCCA+X'750' (LCCAPPS3)
- ILC/PINT saved at LCCA+X'758' (LCCAPIC3)
- TEA saved at LCCA+X'75C' (LCCAPTE3)
- Cross memory status saved at LCCA+X'1C8' (LCCAPXM3)

For all trace buffer full interruptions:

- Registers 0-15 saved at LCCA+X'6C0' (LCCAPGR4)

For page faults or segment faults that require I/O the following occurs:

- Unlocked tasks:
  - Registers moved to TCB
  - PSW moved to RB
  - Cross memory status moved to XSB
- Locked tasks:
  - Registers moved to IHSA
  - PSW moved to IHSA
  - Cross memory status moved to IHSA's XSB
- SRBs:
  - Are suspended: see "SRB Suspension" later in this chapter

Locally Locked Task Suspension

Status saving is the same as for locked task interruptions (described earlier under "I/O FLIH") except that IHSA of the locally locked address space also contains the floating point registers, the FRR stacks, and the PSW. The ASCBLOCK field is updated to contain X'7FFFFFFF'. The XSB contains cross memory status, which includes control registers 3 and 4.

SRB Suspension

An SRB can be suspended in three cases. If a service routine encounters a page fault or a segment fault and a page-in is required, then the SRB routine must give up control. In that event, an SSRB (suspended SRB) must be obtained and the status saved in that control block. Then the SSRB is queued from the page control block (PCB) in the real storage manager. When the paging I/O completes, the SSRB is scheduled to the local service priority list (LSPL) where it is found later by the dispatcher. The SSRB must be obtained because the original SRB was not retained after the dispatch. Status saved in an SSRB must include the current FRR stack.

In the second case, a service routine requests a page fix and a page-in is required. This suspension is handled as in case one, except that the SSRB is queued from the function control block (FCB).

The third case of SRB suspension is an unconditional request for an unavailable lock. Status saving for SRB suspension for a lock differs from the page fault where the SSRB is queued and where control returns after the redispatch of the SSRB. For a request for the LOCAL lock when it is unavailable, the SSRB is queued from the ASCB. For a request for an unavailable CML lock, the SSRB is queued from the ASCB whose lock is requested. For a request for an unavailable cross memory services lock, the SSRB is queued on that cross memory service lock's suspend queue. (For more detail see the chapter on "Locking" later in this section.) In the cross memory services case of SRB suspension, resumption is at the appropriate entry in the lock manager to try to acquire the lock. Upon release of the cross memory services lock by the holder, any SSRBs are rescheduled. Upon release of the local lock by the holder, and if all suspended elements are for LOCAL lock requests rather than CML lock requests, then the first SSRB that was suspended is given the local lock and rescheduled. The SRB is given control at the next sequential instruction following the lock manager call. If any one of the elements on the local lock suspend queue is suspended as a result of a CML lock request, then all queue elements are dequeued and rescheduled to retry the lock request.

Suspend SRB queues can be summarized:

Page Faults and Segment Faults

PCBs can be found in the following locations:

- For pages with I/O in progress, PCBs are pointed to by the PFTE for the frame backing the page (PFTPCB, PFT+X'18').
- For cross memory page faults or segment faults, or common area page faults, PCBS are queued on the notification PCB queue for the home address space. The NPQ is anchored in the RAB for the home address space (RABNPQF, RAB+X'6C').
- For pages that cannot be backed because a frame is not available, PCBs are queued on the defer PCB queue. The DPQ is anchored in the RIT (RITDPQF, RIT+X'8C').
- PCB+X'20' points to SSRB.

Page Fix

- PCB is chained as for page fault.
- PCB+X'24' points to FCB.
- FCB+X'18' points to SSRB.

Local Lock Requests

- SSRB is queued from ASCBLSQH(ASCB+X'84').

CML Lock Requests

- SSRB is queued from ASCBLSQH of the ASCB whose lock is requested.

Cross Memory Services Lock Requests

- The SSRB is queued from a cross memory services lock's suspend queue in IEAVESLA as shown in Figure 2-4 in the following topic "Locking."

Locking

Serialization of resources to provide data integrity and protection is a necessary function of operating systems. In pre-MVS systems, resource serialization was accomplished by physical disablement and by the ENQ/DEQ component. Physical disablement controls only one processor and thus, in MP systems, does not guarantee serialization.

To achieve these requirements the locking facility provides:

- Serialization in an MP system
- Serialization across address spaces for common resources
- Serialization within address spaces

A lock manager function acquires and maintains all locks. Use of the lock manager is restricted to key 0 programs running in supervisor state, which prevents unauthorized problem programs from interfering with the serialization process. Locking functions are provided by two nucleus resident modules: IEAVELK, the spin lock manager; and IEAVESLK, the suspend lock manager.

Categories of Locks

MVS locks are divided into two categories:

- *Global Locks*, which protect serially reusable resources related to more than one address space. These resources provide system-wide services or use control information in the common area. Examples of resources protected by global locks are UCBs and RSM control blocks.

  You can use the IPCS subcommand STATUS CPU to determine which locks are held on each CPU.

- *Local Locks*, which protect serially reusable resources assigned to a particular address space. When a task or SRB holds a local lock, the queues and control blocks serialized by that lock can be used only by the task or SRB holding the lock.

  You can use the IPCS subcommand ANALYZE to do suspend or local lock analysis.

Figure 2-1 defines the locks. All locks, except the LOCAL and CML locks, are global locks.

Lock Name	Category	Type	Description (See note 1.)
RSMGL	Global	Spin	- Real storage management global lock - serializes RSM global resources.
VSMFIX	Global	Spin	- Virtual storage management fixed subpools lock - serializes VSM global queues.
ASM	Global	Spin	- Auxiliary storage management lock - serializes ASM resources on an address space level.
ASMGL	Global	Spin	- Auxiliary storage management global lock - serializes ASM resources on a global level.
RSMST	Global	Spin	- Real storage management steal lock - serializes RSM control blocks on an address space level when it is not known which address space locks are currently held.
RSMCM	Global	Spin	- Real storage management common lock - serializes RSM common area resources (such as page table entries).
RSMXM	Global	Spin	- Real storage management cross memory lock - serializes RSM control blocks on an address space level when serialization is needed to a second address space.
RSMAD	Global	Spin	- Real storage management address space lock - serializes RSM control blocks on an address space level.
RSM	Global	Spin	- Real storage management lock (shared/exclusive) - serializes RSM functions and resources on a global level.
VSMPAG	Global	Spin	- Virtual storage management pageable subpools lock - serializes the VSM work area for VSM pageable subpools.
DISP	Global	Spin	- Global dispatcher lock - serializes the ASVT and the ASCB dispatching queue.
SALLOC	Global	Spin	- Space allocation lock - serializes receiving routines that enable a processor for an emergency signal or malfunction alert.
IOSYNCH	Global	Spin	- I/O supervisor synchronization lock - serializes, using a table of lockwords, IOS resources.
IOSUCB	Global	Spin	- I/O supervisor unit control block lock - serializes access and updates to the UCBs. There is one IOSUCB lock per UCB.
SRM	Global	Spin	- System resources management lock - serializes SRM control blocks and associated data.
TRACE	Global	Spin	- Trace lock (shared/exclusive) - serializes the reading (shared) and writing (exclusive) of the system trace buffer.
CPU	Global	Spin	- Processor lock - provides system-recognized (legal) disablement. (See note 2.)
CMSSMF	Global	Suspend	- System management facilities cross memory services lock - serializes SMF functions and control blocks. (See note 3.)
CMSEQDQ	Global	Suspend	- ENQ/DEQ cross memory services lock - serializes ENQ/DEQ functions and control blocks. (See note 3.)
CMS	Global	Suspend	- General cross memory services lock - serializes on more than one address space where this serialization is not provided by one or more of the other global locks. The CMS lock provides global serialization when enablement is required. (See note 3.)
CML	Local	Suspend	- Local storage lock - serializes functions and storage within an address space other than the home address space. There is one CML lock per address space. (See note 4.)
LOCAL	Local	Suspend	- Local storage lock - serializes functions and storage within a local address space. There is one LOCAL lock per address space. (See note 4.)

Notes:

1. *All locks are listed in hierarchical order, with RSMGL being the highest lock in the hierarchy. (See also notes 2, 3, and 4.)*
2. *The CPU lock has no hierarchy in respect to the other spin type locks. However, once obtained, no suspend locks can be obtained.*
3. *The cross memory services locks (CMSSMF, CMSEQDQ, and CMS) are equal to each other in the hierarchy.*
4. *The CML and LOCAL locks are equal to each other in the hierarchy.*

Figure 2-1. Definition and Hierarchy of Locks

Types of Locks

Two types of locks exist. The type determines what happens when a processor makes an unconditional request for a lock that is unavailable. The types are:

- Spin locks - prevent the requesting processor from doing any work until the lock is released by the owning processor. The requesting processor enters a loop in the spin lock manager (IEAVELK) that keeps testing the lock until the owning processor releases it. As soon as the resource is free, the spinning processor can obtain the resource and continue processing.

- Suspend locks - prevent the requesting unit of work from doing work until the lock is available, but allow the processor to continue doing other work. The suspend lock manager (IEAVESLK) queues the request by suspending the requesting task or SRB. The requesting processor is then dispatched to do other work. Upon release of the lock, all of the queued requesters are made dispatchable to retry the lock request, except in the case of the local lock. Upon release of the local lock, the first SSRB will be given the lock and rescheduled; unless there are CML lock requesters on the suspend queue, in which case, all requesters are rescheduled to retry the lock request.

Combining Categories and Types of Locks

Combining categories and types of locks provide the following three groups of locks.

- Global spin locks
- Local suspend locks
- Global suspend locks

Global Spin Locks

A *global spin lock* is used to provide serialization in MP systems. While code is executing under a global spin lock, it is physically disabled for I/O and external interrupts. An unconditional request for an unavailable lock will cause the processor to spin in the spin lock manager. Upon release of the global spin lock, the looping processor acquires ownership and returns control to the requestor.

The supported global spin locks are listed in Figure 2-1. Additional information for the shared/exclusive locks (RSM and TRACE) and the CPU lock is provided in the following topics.

Shared/Exclusive Locks

A shared/exclusive lock is used to serialize the reading or the updating of a global resource. The RSM and TRACE locks are shared/exclusive locks. Up to 16 processors can own a shared/exclusive lock as shared at one time. Only one processor can own a shared/exclusive lock as exclusive at one time. In general, the owners of a shared/exclusive lock as shared have "read only" authority to the resource, and the owner of a shared/exclusive lock as exclusive has "write" authority to the resource. While code is executing under a shared/exclusive lock, it is physically disabled.

Results of unconditional requests for shared/exclusive locks are:

- An unconditional request for a shared/exclusive lock as shared, while held as shared by other processors, will result in the requester obtaining shared ownership of the lock.

- An unconditional request for a shared/exclusive lock as shared, while held as exclusive by another processor, will result in the processor spinning for the lock until the exclusive owner releases the lock.

- An unconditional request for a shared/exclusive lock as exclusive, while held as shared by other processors, will result in the processor spinning for the lock until all shared owners release the lock. New requests for shared ownership of a shared/exclusive lock are not allowed until the spinning exclusive request is satisfied.

- An unconditional request for a shared/exclusive lock as exclusive, while held as exclusive by another processor, will result in the processor spinning for the lock until the exclusive owner releases the lock.

In general, processors that are spinning for exclusive ownership of a shared/exclusive lock are given priority over processors that are spinning for shared ownership of the lock.

CPU Lock

The CPU lock provides system-recognized (legal) disablement for units of work (requesters). When the CPU lock is obtained, the requester is physically disabled for I/O and external interruptions. (Note that the CPU lock does not provide serialization between processors.)

There is one CPU lock per processor and it can be obtained by any number of requesters. However, a unit of work executing on one processor cannot obtain another processor's CPU lock.

The CPU lock has no hierarchy in respect to other spin type locks, but it does have hierarchy in respect to suspend type locks. A requester can obtain the CPU lock before or after obtaining any spin type lock; however, once the CPU lock is obtained, the requester cannot obtain a suspend type lock.

The contents of the CPU lockword (at PSA+X'2E0') represents the cumulative count of lock holders for a processor. The CPU lockword does not contain a processor ID (as other global spin lockwords do).

Local Suspend Locks

A *local suspend lock* is used to serialize resources within an address space. There is one local suspend lock per address space and it is located in the ASCB. An unconditional request for the LOCAL or CML lock when it is not available causes the suspension of the requesting task or SRB until the lock is released.

Global Suspend Locks

A *global suspend lock* is used to serialize resources that are commonly addressable from any address space. The requestor remains physically enabled while owning the lock. The general cross memory services lock (CMS), the ENQ/DEQ cross memory services lock (CMSEQDQ), and the SMF cross memory services lock (CMSSMF) are the only supported global suspend locks. A local lock must be held in order to obtain a cross memory services lock. An unconditional request for any cross memory services lock when it is unavailable causes suspension of the requesting task or SRB.

Locking Hierarchy

To prevent a deadlock between processors, MVS locks are arranged in a hierarchy, and a processor may unconditionally request only locks higher in the hierarchy than locks that it currently holds. The locking hierarchy is the order in which the locks are listed in Figure 2-1 with RSMGL being the highest lock in the hierarchy.

Some locks are single system locks (for example, DISP), and some locks are multiple locks in which there is more than one lock within the lock level (for example, IOSUCB). For those global lock levels that have more than one lock, a processor may only hold one lock of each level. For example, if a processor holds an IOSUCB lock, it may not request a different IOSUCB lock.

A unit of work can hold only one local lock at a time. A unit of work cannot hold both its own LOCAL lock and CML lock of another address space. Note that the CML lock of an address space, obtained from another address space, is the same as the LOCAL lock of the address space.

A local lock must be held by the caller when requesting any cross memory services lock. Also, a local lock cannot be released while holding any cross memory services lock.

It is not necessary to obtain all locks in the hierarchy up to the highest lock needed. Only the needed locks have to be obtained, but in hierarchical sequence.

The caller may obtain the three cross memory services locks (CMSSMF, CMSEQDQ, and CMS) only by requesting all of them in a single lock manager request. If a caller holds any one and requests another, an abend will result.

Determining Which Locks Are Held On a Processor

To diagnose certain MVS problems, such as wait states and performance degradation, it is necessary to determine the lock status of the system as well as the back-up of work caused by lock contention.

You can use the IPCS subcommand STATUS CPU to determine which locks are held on each CPU.

Locks held by a particular processor are indicated in the processors PSA (prefixed save area). There is a bit map in the PSA which the lock manager checks when a request is made for a lock. This map is called PSACLHS (PSA current locks held string). Each bit corresponds to a particular lock in the hierarchy. The bit positions in PSACLHS do not represent the hierarchy of the locks. When a bit is on, it means that lock is held by the current unit of work executing on the corresponding processor. Figure 2-2 shows the bit assignments.

When the local lock bit is on in the PSACLHS, either the LOCAL lock or a CML lock is held. To determine which lock is held by the current unit of work, check the contents of PSALOCAL. If PSALOCAL is zero, then the LOCAL lock of the home address space (pointed to by PSAAOLD) is held. If PSALOCAL is nonzero, the local lock of the address space pointed to by PSALOCAL is held as a CML lock.

Note: When a holder of the local lock or a cross memory services lock is suspended, the corresponding bit in the PSACLHS field is copied to the ASCBHLHI and the PSACLHS is set to 0 even though the lock is still held.

PSACLHS (location X'2F8' in PSA)

2F8	2F9	2FA	2FB	Lock	
80	00	00	00	CPU	
40	00	00	00	Reserved	
20	00	00	00	Reserved	
10	00	00	00	Reserved	
08	00	00	00	RSM	
04	00	00	00	TRACE	
02	00	00	00	Reserved	
01	00	00	00	Reserved	
00	80	00	00	Reserved	
00	40	00	00	Reserved	
00	20	00	00	Reserved	
00	10	00	00	RSMCM	
00	08	00	00	RSMGL	
00	04	00	00	VSMFIX	
00	02	00	00	ASMGL	
00	01	00	00	RSMST	
00	00	80	00	RSMXM	
00	00	40	00	RSMAD	
00	00	20	00	VSMPAG	
00	00	10	00	DISP	
00	00	08	00	ASM	
00	00	04	00	SALLOC	
00	00	02	00	IOSYNCH	
00	00	01	00	Reserved	
00	00	00	80	IOSUCB	
00	00	00	40	Reserved	
00	00	00	20	TPNCB	
00	00	00	10	TPDNCB	See note
00	00	00	08	TPACBDEB	
00	00	00	04	SRM	
00	00	00	02	CMS/CMSEQDQ/CMSSMF	
00	00	00	01	LOCAL/CML	

Note: TP-related locks, but not used by the system.

Figure 2-2. Bit Map to Show Locks Held on a Processor

Content of Lockwords

Each lock is represented by a lockword that defines the availability and status of the lock. The contents of lockwords differ according to the category and type of lock they describe:

Global Spin Lockword

For global spin lockwords except the shared/exclusive locks (RSM and TRACE) and the CPU lock:

- X'00000000' - Lock is available.
- X'0000004n' - Lock is held on processor n.

For shared/exclusive locks (RSM and TRACE):

- Bit 0 indicates exclusive ownership.
- Bit 1 indicates an exclusive request is pending.
- Bits 2-15 are reserved and set to zero.
- Bits 16-31 indicate processors 0-15 respectively.

- X'00000000' - Lock is available.
- X'00008000' - Lock is held shared by processor 0.
- X'4000C001' - Lock is held shared by processors 0, 1, and 15, and one or more other processors are waiting for exclusive ownership.
- X'80008000' - Lock is held exclusively by processor 0.
- X'C0008000' - Lock is held exclusively by processor 0, and one or more other processors are waiting for exclusive ownership.

For the CPU lock:

- X'00000000' - Lock is not held.
- X'00000001' - Lock has been obtained by a unit of work.
- X'0000002F' - Lock has been obtained 47 times.

Global Suspend Lockword (Cross Memory Services Locks)

- X'00000000' - Lock is available.
- X'xxxxxxxx' - ASCB address of the locally locked address space. If an address space holds a cross memory services lock but is interrupted or suspended, ASCBHLHI of the locally locked address space will be set and the cross memory services lock-held bit in PSACLHS is turned off until the address space is redispatched. The ASCB address remains in the cross memory services lock until the lock is released.

Local Suspend Lockword (Local Lock)

- X'00000000' - Lock is available.
- X'0000004n' - Lock is held on processor n.
- X'4FFFFFFF' - Task holding a CML lock is now dispatchable or an SSRB holding either the LOCAL or a CML lock is now dispatchable.
- X'7FFFFFFF' - Task or SRB suspended while holding the lock. The reason for suspension is:
  - A page fault.
  - Waiting for a synchronous page fix to complete.

  - An unconditional request for a cross memory services lock while it was unavailable.

  - SUSPEND=YES was specified on the SDUMP macro.

- X'FFFFFFFF' - Task holding the LOCAL lock was suspended or interrupted but is now dispatchable. The reasons for this state are:

  - A page fault or page fix has been resolved for a locked task.
  - The cross memory services lock, at one time unavailable, is now available.
  - A task holding the LOCAL lock has been preempted.

How to Find Lockwords

Lockwords for single system locks are located in a system lock area called IEAVESLA. PSA+X'2FC', PSALITA, points to the lock interface table (LIT) in module IEAVELIT; LIT+X'28' points to IEAVESLA. Lockwords for single system locks can also be located at the label IEAVESLA in a NUCMAP.

You can use the IPCS subcommand

```
LIST IEAVESLA STRUCTURE(IEAVESLA)
```

to locate the lockwords.

Lockwords for multiple system locks are supplied by the requestor of the lock. The addresses of these are placed in the PSA for each processor.

The locations of lockwords are shown in Figure 2-3. Note that all lockwords must reside in fixed common storage.

Lock Name	Category	Type	Number of Locks	Location of Lock	Location of Address of Lock (when actually held)
RSMGL	Global	Spin	1 per system	RIT+X'14'	PSA+X'2AC'
VSMFIX	Global	Spin	1 per system	IEAVESLA+X'48'	PSA+X'2B0' (See note 2)
ASM	Global	Spin	1 per address space	ASMHD+X'14'	PSA+X'284'
ASMGL	Global	Spin	1 per system	ASMVT+X'24'	PSA+X'2B4'
RSMST	Global	Spin	1 per address space	RAB+X'1C'	PSA+X'2B8'
RSMCM	Global	Spin	1 per address space	RAB+X'1C'	PSA+X'2C8'
RSMXM	Global	Spin	1 per address space	RAB+X'1C'	PSA+X'2BC'
RSMAD	Global	Spin	1 per address space	RAB+X'1C'	PSA+X'2C0'
RSM	Global	Spin	1 per system	IEAVESLA+X'50'	PSA+X'2D0' (See note 2)
VSMPAG	Global	Spin	1 per system	IEAVESLA+X'58'	PSA+X'2C4' (See note 2)
DISP	Global	Spin	1 per system	IEAVESLA+X'0'	PSA+X'280' (See note 2)
SALLOC	Global	Spin	1 per system	IEAVESLA+X'8'	PSA+X'288' (See note 2)
IOSYNCH	Global	Spin	1 per IOS process	IOCOM+X'38' (See note 1)	PSA+X'28C'
IOSUCB	Global	Spin	1 per UCB	UCB-X'8'	PSA+X'294'
SRM	Global	Spin	1 per system	IEAVESLA+X'10'	PSA+X'2A8' (See note 2)
TRACE	Global	Spin	1 per system	IEAVESLA+X'60'	PSA+X'2D4' (See note 2)
CPU	Global	Spin	1 per processor	PSA+X'2E0' (Count of CPU lock holders)	--
CMSSMF	Global	Suspend	1 per system	IEAVESLA+X'18'	--
CMSEQDQ	Global	Suspend	1 per system	IEAVESLA+X'28'	--
CMS	Global	Suspend	1 per system	IEAVESLA+X'38'	--
CML	Local	Suspend	1 per address space	ASCB+X'80'	PSA+X'2EC' (See note 3)
LOCAL	Local	Suspend	1 per address space	ASCB+X'80'	--

Notes:

1. *Points to an IOS lock table.*
2. *For these single system locks, the fields in the PSA point directly to the lockword, whether it is owned or not owned.*
3. *For the CML lock, this field in the PSA (PSALOCAL) points to an ASCB, not a lockword.*

Figure 2-3. Classification and Location of Locks

Results of Requests For Unavailable Locks

The results of requests for unavailable locks are described in the following topics.

Global Spin Locks

An unconditional request for an unavailable global spin lock results in a disabled loop in the spin lock manager (IEAVELK). While in the disabled spin loop, the lock manager will periodically enable for EMS or MFA interrupts. The lock manager spins until the global lock is released by the owning processor. In this case, register 11 contains the address of the requested lock and PSALKR14 contains the address of the requestor.

If the lock manager spins for an excessive period of time, then message IEE331A is issued to the operator when the lock manager invokes the excessive spin notification routine, IEEVEXSN. If the operator does not initiate an ACR condition, the lock manager continues to spin until the lock becomes available.

Local Locks

Tasks requesting an unavailable LOCAL lock are suspended. In each case, the request block old PSW (RBOPSW) is set to re-enter the lock manager, and the registers are saved in the TCB. A flag is set in the TCB (TCBLLREQ) to indicate to the dispatcher that the task should not be dispatched until the LOCAL lock is available.

SRBs requesting an unavailable LOCAL lock are suspended. In each case, the suspend lock manager calls the STOP/RESET service (IEAVESRT) to have an SSRB obtained and status saved. The lock manager then queues the SSRB on the LOCAL lock suspend queue.

If the SRB was scheduled with the LOCAL lock option, the LOCAL lock will be obtained for the SRB by the dispatcher. The SRB will get control with the LOCAL lock held. If the LOCAL lock is unavailable, the dispatcher will delay the SRB, that is, the SRB will be queued to the LOCAL lock suspend queue and will not be dispatched until the LOCAL lock is available.

Tasks that unconditionally request the LOCAL lock when it is unavailable are suspended by the suspend lock manager (IEAVESLK). IEAVESLK saves the registers, PSW, and cross memory status in the TCB, RB, and XSB. The TCBLLREQ flag is turned on to indicate that this task requires the LOCAL lock. The suspend lock manager exits to the dispatcher, which will not dispatch this task until the LOCAL lock can be obtained.

Tasks that request an unavailable CML lock are stopped via a call by the suspend lock manager to the STOP/RESET service (IEAVESRT). The registers, PSW, and cross memory status are saved by IEAVESRT in the TCB, RB, and XSB. The lock manager obtains an SRB, initializes it to run in the requester's address space, and queues it to the local lock suspend queue header (ASCBLSQH) of the address space whose CML lock was requested. This SRB has the SRBCMLRQ bit set on to indicate that a CML request was made for this ASCB's local lock.

SRBs that request an unavailable CML lock are suspended. The suspend lock manager calls the STOP/RESET service (IEAVESRT) to have an SSRB obtained and status saved. The registers, PSW, and cross memory status are saved in this SSRB and its XSB. The lock manager then queues the SSRB to the CML address space's local lock suspend queue header (ASCBLSQH). This SSRB has the SRBCMLRQ bit set on to indicate that this address space's lock was requested as a CML lock.

Notes:

1. *The FRR stack can be used to help recreate the process leading up to the point of suspension by interpreting the recovery routines that are currently active. SSRBs for local lock suspensions can be found by inspecting the local lock suspend queue anchored in the ASCB from field ASCBLSQH (ASCB+X'84'). SSRBs are obtained from SQA. SSRBs and delayed SRBs on the local lock suspend queue are chained together at SRB+X'04'.*

2. *When interrogating a given address space, if the ASCBLOCK field is not X'00000000', check the ASCBLSQH to determine the SRB work being delayed in this address space because of lock contention.*

3. *If the ASCBLOCK field is not X'00000000', check the ASCBLOCI field (ASCB+X'E8'). If ASCBLOCI is X'00000000', then the address space's lock is held as a LOCAL lock and not a CML lock.*

4. *If the ASCBLOCK field is not X'00000000' and not a processor ID, then the ASCBCMLH field (ASCB+X'EC') contains the address of the unit of work that was suspended while holding the address space's local lock.*

   For a unit of work that is suspended and holds the address space's local lock as a CML lock, the ASCBLOCI field (ASCB+X'E8') points to the ASCB where the lock owner resides, and the ASCBCMLH field (ASCB+X'EC') points to the owning unit of work. When the high-order bit of ASCBCMLH is on, the unit of work is an SSRB.

   When the local lock is released, the suspend queue is scanned until the first suspended (oldest) element is found or until an element representing a CML lock request is found. If no requesters for a CML lock exist on the suspend queue, the first suspended (oldest) element is dequeued, the ready-to-run ID is placed in the lockword (ASCBLOCK=X'4FFFFFFF'), the SRB/SSRB is given the lock by turning on the SRB local lock held flag (SRBLLHLD), and the SRB is scheduled locally. If a requester for the CML lock exists on the suspend queue, then all suspended elements (SRBs and SSRBs) are dequeued and rescheduled so that the obtain request is retried. The SRBs on the suspend queue that represent the CML lock requester cause the task to be resumed so that the lock request is retried.

Cross Memory Services Locks

Tasks unconditionally requesting a cross memory services lock when it is unavailable are suspended. For each task:

- GPRs are saved in the IHSA which is pointed to from ASXB+X'20' of the locally locked address space.
- The resume PSW in the IHSA is set to re-enter the lock manager.
- The cross memory status is saved in the XSB pointed to by IHSA+X'80'.
- The locally locked ASCB is queued on that cross memory services lock's suspend queue. The suspend queue header for the SMF cross memory services lock follows the lockword in CSECT IEAVESLA at offset X'1C', the suspend queue header for the ENQ/DEQ cross memory services lock follows the lockword at offset X'2C', and the suspend queue header for the general cross memory services lock follows the lockword at offset X'3C'.

  Note: When a NUCMAP is not available, locate the IEAVESLA through PSA+X'2FC' which contains the address of the lock interface table in module IEAVELIT, the lock interface table+X'28' contains the address of IEAVESLA.

The tasks suspended on a cross memory services lock suspend queue are represented by the ASCBs whose local locks they own. For example, if task A in address space A owned the CML lock of address space B and was suspended on the cross memory services lock suspend queue, then the cross memory services suspend queue header would contain the ASCB address of address space B. The ASCBLOCI field of address space B would point to address space A and the ASCBCMLH field of address space B would point to task A. The ASCBs are chained together at field ASCBCMSF (forward pointer).

Note: When an ASCB is on the cross memory services lock suspend queue, the local lock (ASCBLOCK) contains X'7FFFFFFF'.

When the cross memory services lock is released, the ASCBLOCK field of the locally locked address spaces on the suspend queue is changed to one of the following values:

- X'FFFFFFFF' - the LOCAL lock was held by a task that is now ready to run.

- X'4FFFFFFF' - the lock was held by either (1) a task holding the lock as a CML lock and the task is now ready to run, or (2) an SSRB holding a CML or LOCAL lock and the SSRB is now ready to run.

SRBs unconditionally requesting a cross memory services lock when it is unavailable, are suspended. For each SRB, the lock manager calls the STOP/RESET service (IEAVESRT) which:

- Obtains an SSRB from SQA
- Saves GPRs and the FRR stack in the SSRB
- Sets the local lock (ASCBLOCK) to X'7FFFFFFF'
- Saves cross memory status in the XSB of the SSRB.

Then the lock manager chains the SSRB on that cross memory services lock-suspend queue located in IEAVESLA.

The offsets for the cross memory services lock suspend queues are:

- CMSSMF - IEAVESLA+X'1C'
- CMSEQDQ - IEAVESLA+X'2C'
- general - IEAVESLA+X'3C'

There is one suspend queue per cross memory services lock and the requestor is chained on the queue associated with the unavailable lock.

The SSRBs and ASCBs are chained on the respective suspend queues using either ASCBCMSF (ASCB+X'C') or SRBFLNK (SSRB+X'4'). There are no backward pointers. Thus the cross memory services lock suspend queues could appear as shown in Figure 2-4.

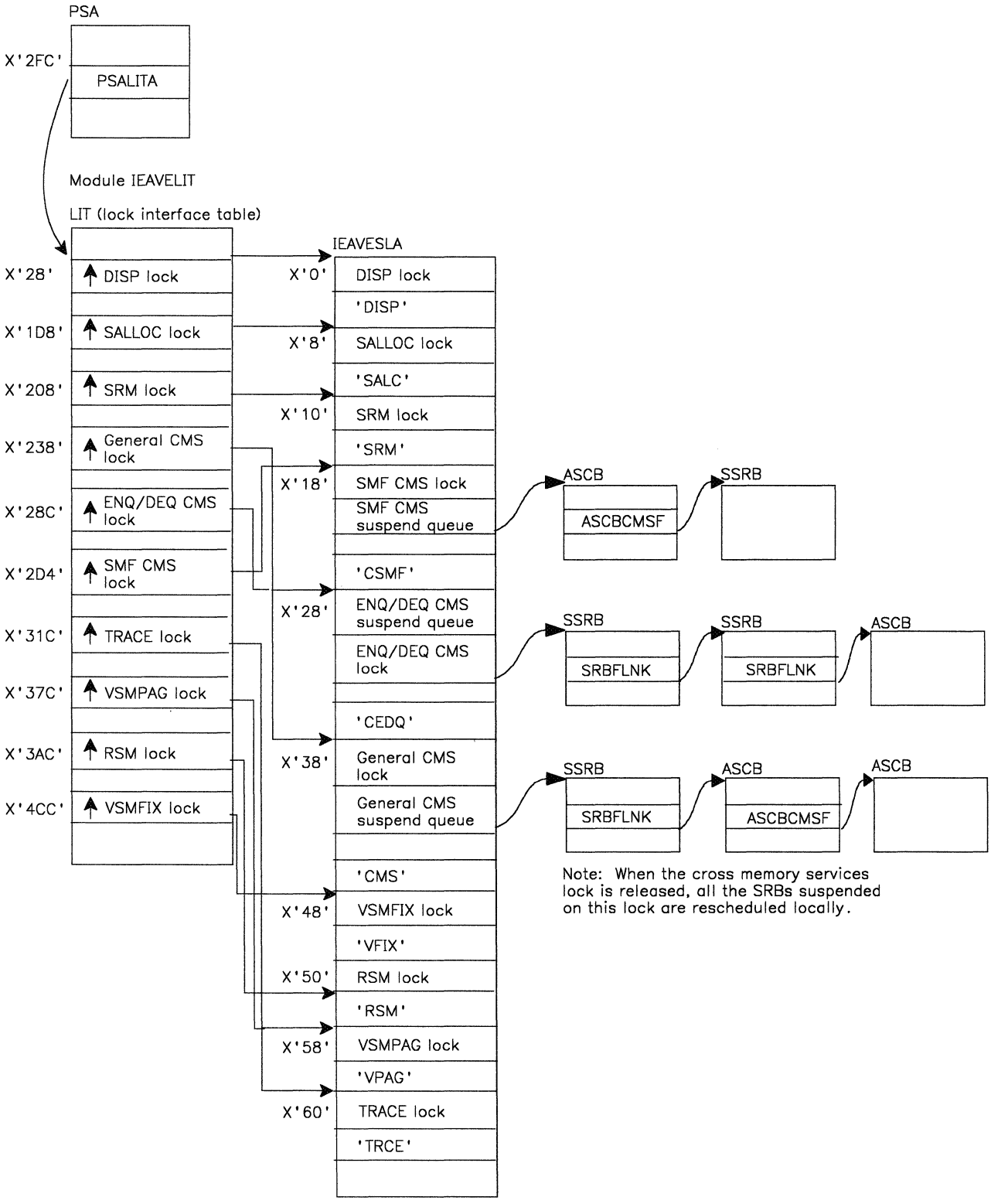
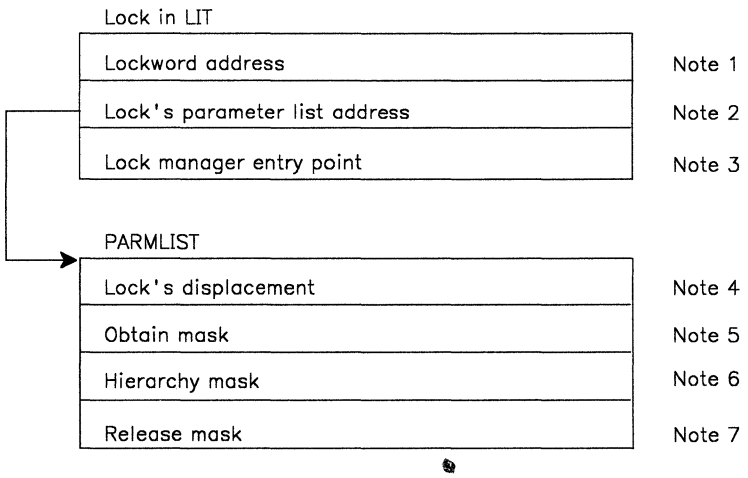


Figure 2-4. Cross Memory Services Lock Suspend Queues

Lock Interface Table

The SETLOCK macro uses the lock interface table (LIT) in module IEAVELIT to initialize the entry registers and to call the appropriate entry point in the lock manager (routines IEAVELK and IEAVESLK). PSALITA points to the LIT.

Each defined lock has a three-word structure in the LIT and an associated four-word parameter list as follows:



Notes:

In the LIT:

1. *Contains the address of the lockword for a single system lock, or zero for a multiple system spin lock.*

2. *Contains the address of a four-word parameter list, which contains information needed to process the lock request.*

3. *Contains the entry point address of the lock manager routine that will process the obtain/release lock request.*

In the PARMLIST:

4. *Contains the displacement of the lock into the locks held table (PSACLHT), which contains the address of the lockword.*

5. *Contains a four-byte bit mask (representing this lock) that is ORed with the PSACLHS field when the lock is obtained to indicate the lock is owned.*

   Example: *The obtain mask for the DISP lock is X'00001000'.*

6. *Contains a four-byte bit mask (representing this lock) that is used to ensure that the requester is not requesting a lock that would violate the lock hierarchy. The hierarchy mask is ANDed with the PSACLHS field to test for a violation.*

In the hierarchy mask, the following bits are set to one:

- *All reserved bits*
- *The bit representing this lock*
- *All bits representing locks higher in the hierarchy than this lock*

Refer to Figure 2-1 for the lock hierarchy, and Figure 2-2 for the lock bit assignments in the PSACLHS field.

Example: *The hierarchy mask for the DISP lock is X'7BFFF940'. When the DISP lock is requested, the four bytes in the hierarchy mask have the following meaning:*

X'7B' - indicates that the RSM lock cannot be held; but that the CPU and TRACE locks can be held.

X'FF' - indicates that the RSMCM, RSMGL, VSMFIX, ASMGL, and RSMST locks cannot be held.

X'F9' - indicates that the RSMXM, RSMAD, VSMPAG, and ASM locks cannot be held; but that the SALLOC and IOSYNCH locks can be held. (Note that the DISP lock itself may or may not be held by the requester.)

X'40' - indicates that the IOSUCB, the unused TP locks, SRM, CMS/CMSEQDQ/CMSSMF, and LOCAL/CML locks can be held.

7. *Contains a four-byte bit mask (representing this lock) that is ANDed with the PSACLHS field when the lock is released to indicate the lock is no longer owned.*

   Example: *The release mask for the DISP lock is X'FFFFEFFF'.*

Intersect

Locking serializes resources between routines and processors. The intersect function is used in conjunction with locking to serialize dispatcher control blocks between specific routines and the dispatcher.

There are two levels of intersection:

1. Global intersect, which serializes the ASCB dispatching queue and address space dispatchability flags, first requires that the dispatcher lock be held (the lock serializes between routines, the intersect serializes only with the dispatcher).

2. Local intersect, which serializes the TCB dispatching queue and TCB dispatchability flags, first requires that the local lock of the target address space be held.

Note: The lock associated with each intersect must be obtained before the intersect is requested and the intersect must be reset before releasing the lock to ensure proper serialization.

Determining if Intersects are Held on a Processor

Intersect contention can exist just like locking contention. To check for this condition it is necessary to know where the intersect words are and what they should look like. The global intersect word is in the SVT. PSA+X'B4C' (PSASVT) contains the address of the SVT. If any bits are on in the SVTDSREQ word (SVT+X'1C'), then the global intersect is held. Refer to the SVT mapping in the *Debugging Handbook* to determine who holds the intersect.

A local intersect word exists at X'B4' into each ASCB. Bits in this word are defined in the same way as the global intersect. Refer to the ASCB mapping in the *Debugging Handbook* to determine who holds the local intersect.

The dispatcher has a four-word field of its own which it sets when processing. The dispatcher active field (SVTDACTV) is at offset X'160' into the SVT. The first byte of the field corresponds to CPU0, the second to CPU1, and so forth. Each byte should have one of the following settings:

- X'00' - Dispatcher not active on the corresponding processor
- Logical CPUID with high-order bit off - Dispatcher is executing on this processor
- Logical CPUID with high-order bit on - Dispatcher is in recursion mode on this processor.

Requesting the Intersect

A routine requests the intersect via the INTSECT macro. The macro turns on the requestor's intersect bit, then, if the dispatcher active field is not zero, the macro passes control to the intersect service routine (IEAVEINT) which spins, waiting for all dispatcher active bytes to go to zero.

This intersect spin can be detected by examining the registers:

- R15 - address of IEAVEINT
- R1 - address of SVT
- R2 - physical CPU ID of processor on which dispatcher is active
- R3 - contents of dispatcher active byte
- R14 - return address of caller

Ensure that the contents of R3 are valid (that is, that SVTDACTV has not been overlaid). If the caller was disabled, then intersect would set LCCASPN1 with its spin bit (X'02').

Use of Recovery Work Areas For Problem Analysis

Recovery processing enhances the reliability of the MVS operating system. When an error occurs, "active recovery" is given control, one routine at a time, in an attempt to isolate the error to a unit of work. Recovery terminates that work instead of the entire operating system and then continues normal system operation. This process occurs whether the error is in the system or an application.

Because system operation is not halted at the point of error, the resulting storage dumps represent system status sometime after the original error(s). Often the system can encounter numerous errors, fully recover, and continue. At other times it can be a recovery failure that causes the system to cease operations. In either case, the obvious problem and its associated tracks have been covered over. This makes the back-tracking process extremely difficult.

However, experience has shown that although recovery causes this difficulty, it can very often provide valuable clues for the problem analyst. This chapter points out important recovery areas and explains how they can be used in the debugging process.

You can use the IPCS subcommand CBSTAT for an ASCB or a TCB to get the status of any recovery that might be in progress for that address space or task.

CAUTION: Recovery is *not* designed to aid the problem solver; it is designed as a means by which the system can prevent total loss. Because recovery maintains system status information, its work areas often provide the same information to the analyst. However, once recovery is invoked, the system is in a tenuous position; it is attempting to maintain operation despite an error. It is possible that the recovery process itself can encounter the same error or bad data. Most often this is not the case; the system does recover and continues normal operation. But the possibility of recursive errors in the recovery process does exist, in which case the new error becomes of prime consideration. If you are dependent on internal recovery control blocks and queues, be aware of this possibility. Don't get caught following a chain of blocks for some subsequent or unrelated problem that will hinder your own error-finding efforts. This danger is most prevalent when you use recovery work areas without following the normal work-related debugging techniques. Do not immediately use the RTM2 work area without analyzing the Task/RB structure and associated indicators.

The following work areas should be used carefully and only after traditional techniques have failed. The exceptions to this rule are:

- When the dump is taken as a result of a trap (for example, SLIP) and the analyst understands that the current status at the time of error can only be found by using the recovery save areas.

- When there are problems in the recovery process itself.

In other instances, be aware of the total environment so that what you discover in these areas bears some relationship to the problem you are analyzing. These areas are of great importance if used with understanding.

SYS1.LOGREC Analysis

For effective problem analysis, use the information in SYS1.LOGREC to understand the error history of the system. Because of recovery processing, MVS does not halt operation when an error occurs. Dump analysis must be performed using a snapshot of storage as it appears sometime after the error and recovery have occurred; therefore, some type of recording mechanism is needed in order to trace the error.

The entries in SYS1.LOGREC provide information about a potential problem. The SYS1.LOGREC entries serve as a diagnostic trace of the problem encountered by the operating system; they usually provide a history of events leading up to a system incident. Use this information to understand system problems, the recovery actions that are taken as a result of these problems, and the outcome of the recovery attempt. The entries in SYS1.LOGREC are described in *SYS1.LOGREC Error Recording*.

Often more than one record exists for the same software incident. You must be able to relate these records in the proper sequence and understand the progress of recovery the various records indicate. Knowing the errors that have occurred since the last IPL helps you understand the system behavior and explains your findings at dump analysis time.

In stand-alone dump analysis you should always inspect the in-storage LOGREC buffer for entries that recovery routines have made but which were not written to the SYS1.LOGREC data set because of a system problem. Very often it is these records that are the key to the problem solution. (There is a discussion of LOGREC buffer analysis later in this chapter.)

You can use the IPCS subcommand VERBEXIT LOGDATA to format the in-storage LOGREC buffer.

Information that is written by recovery routines to the SYS1.LOGREC data set is used primarily to monitor incidents both when retry is attempted and when percolation to the next recovery routine takes place.

SDWA Records in SYS1.LOGREC: Generally, functional recovery routines (FRRs) will write a software error record to SYS1.LOGREC (via RECORD=YES on the SETRP macro) when they are entered as the first recovery routine for the abend. The default for ESTAE routines, however, is to *not* write a record. This means that unless the ESTAE routine specifically requests recording, no SYS1.LOGREC record will be built.

Symptom Records in SYS1.LOGREC: A symptom record (which is one type of software record) can be initiated and written to SYS1.LOGREC by programs that are authorized to use the SYMREC macro instruction. (*SYS1.LOGREC Error Recording* describes the entries in SYS1.LOGREC.) A mainline module that detects an error condition can initiate a symptom record to record the error in SYS1.LOGREC. Therefore, it is possible for SYS1.LOGREC to contain records that were produced without an ABEND occurring and without the involvement of the system's recovery routines (RTM).

Listing the SYS1.LOGREC Data Set

To get a listing of the SYS1.LOGREC data set, use EREP as described in the *Environmental Record Editing and Printing (EREP) Program User's Guide and Reference*. (The JCL required to print the SYS1.LOGREC data set is contained in the chapter "Additional Data Gathering Techniques" on page 2-95.)

Using EREP, you can get the following reports:

- The *system summary* report, which gives an overview of errors related to an installation's processors, channels, subchannels, I/O subsystem, storage, and operating system.
- The *event history* report, which contains chronologically ordered, one-line abstracts of information from each record. Because this report shows happenings in their context (that is, the order, frequency, and pattern in which errors and events occur), it can help the problem solver recreate the sequence of events leading up to a failure.
- The *detail edit* report, which gives the complete contents of an error record. Each formatted record appears on a separate page, along with a hexadecimal dump of the record.
- The *detail summary* report, which gives a summary of selected data for each type of record. This report can be used to identify problems that occur frequently.

SDWAVRA Key-Length-Data Format

The SDWA variable recording area (SDWAVRA) can optionally be mapped in a key-length-data format. Recovery routines use the SDWAVRA to construct messages and provide data that often contains valuable debugging information. Some MVS recovery routines use the key-length-data format to provide standardized diagnostic information for software incidents. This formatted information allows you to more easily screen duplicate errors.

Constants for the key field have been defined to describe data such as: return and/or reason codes, parameter lists, registers, and control block information. For example, a key of X'10' indicates a recovery routine parameter area. The SDWAVRAM bit (in the fixed portion of the SDWA) indicates that the SDWAVRA has been mapped in the key-length-data format as described by the IHAVRA mapping macro. (SDWAVRAM is the third bit in field SDWADPVA at X'192'.) Refer to the *Debugging Handbook* for the format of the SDWA that includes the SDWAVRA, and the format of the VRAMAP, which includes a list of key values.

Some MVS components have assigned specific meanings to key fields for use by their recovery routines. Refer to the module listings of the individual recovery routines that use the key-length-data format for detailed information.

A VRA key, VRADAE, allows a recovery routine to specify to dump analysis and elimination (DAE) that duplicate dumps can be suppressed. No data is associated with this key. Refer to *System Modifications* for information on DAE.

Figure 2-5 shows an example of the SDWAVRA formatted in the key-length-data format. The example starts at offset X'190' in a hexadecimal dump of the SDWA.

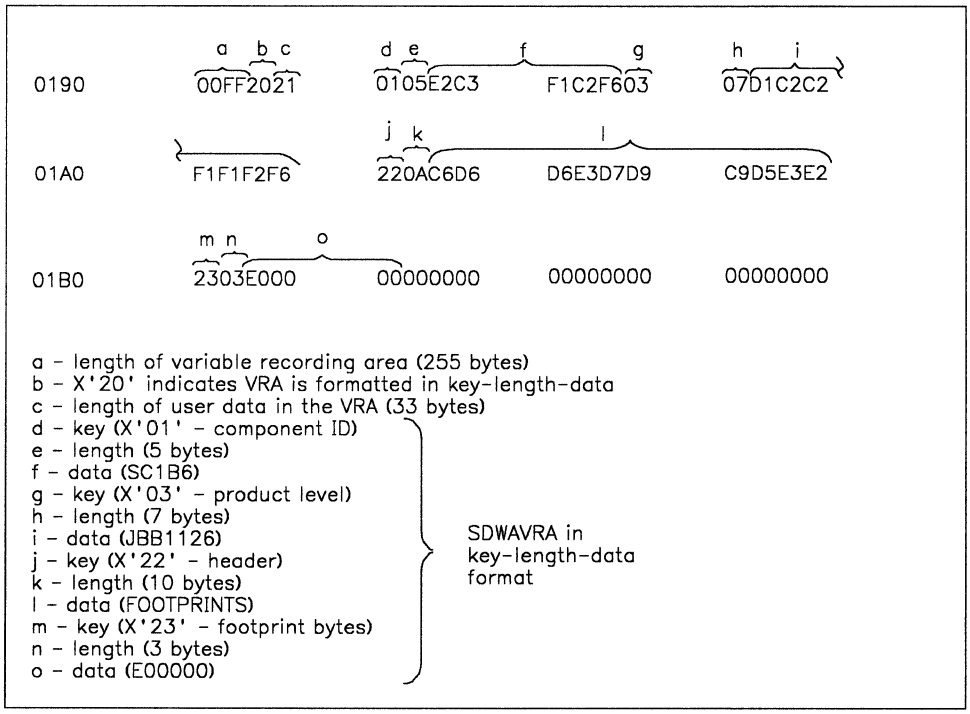


Figure 2-5. Example of SDWAVRA in Key-Length-Data Format

SDWA Recordable Extensions

In addition to the SDWA standard area and the SDWAVRA, the SDWA recordable extensions also contain valuable debugging information, as follows:

- *SDWARC1* (recording extension 1) contains additional component service data (such as the component ID, the component name, and the address of the TCB representing the task that incurred the failure).
- *SDWARC2* (recording extension 2) contains additional I/O machine check data (such as the machine check interruption code).
- *SDWARC3* (recording extension 3) contains locking information (such as the locks to be freed, and the addresses of lockwords).

Note: The SDWA that is in the LOGREC buffer is a compressed SDWA in which the recordable extensions start directly after the used portion of the SDWAVRA. The SDWAURAL field contains the length of the SDWAVRA.

Important Considerations About SYS1.LOGREC Records

The LOGREC records are mostly SDWAs the system supplies, plus variable user data areas the individual recovery routines supply.

Following are some special considerations pertaining to specific portions of LOGREC entries:

- Compare the time stamp at the top of the incident records with those in adjacent records. If the system is percolating through FRRs, these times are either identical or just a fraction of a second apart.

- Jobname - If the jobname is "NONE-FRR," this indicates that the record is generated by an SRB's FRR (functional recovery routine) or the current ASCB was invalid.

- "EC PSW from ESTAE RB (0 for ESTAI)" - This field has the following possible meanings:

  - If the ESTAE is associated with an RB level other than the one encountering the error, this is the PSW at the time that the RB level associated with the ESTAE last gave up control. If this is the case, the "RB of ESTAE Not in Control" flag should also be set.

    If the ESTAE is associated with the RB level in error, the PSW is equal to the "EC PSW at Time of ABEND" because the last time the RB level gave up control was when the error occurred.

    If the error occurred locked, disabled, or in SRB mode and is covered by an ESTAE, the two PSWs might not match even for the top RB. "At Abend" is the locked PSW, and "ESTAE RB" is the PSW for the last unlocked interruption.

  - If the record was generated by an FRR, this is the PSW used to pass control to the FRR and is therefore the address of the FRR.

  - If the record was generated by an FRR (that is, a locked/disabled routine is in control, or the system is in SRB mode), and the "EC PSW at Time of ABEND" is equal to the EC PSW from ESTAE RB, this is a system-generated record.

- "Regs of RB Level of ESTAE Exit or Zero for ESTAI":

  - If the ESTAE exit is associated with the RB level that encountered the error, these registers are the same as "Regs at Time of Error."

  - If the ESTAE is associated with an RB level other than the one encountering the error, then these are the registers at the time that RB last gave up control.

  - If this is an FRR-generated record, the two sets of registers are identical. However, if the FRR or ESTAE has updated the registers for retry, these registers are the new, updated registers.

- "SVC by Locked or SRB Routine" - This indicator can be misleading. A forced SVC 13, which is often the way FRR-protected code passes control to recovery, also causes this flag to be set if the SVC occurred in locked, disabled, or SRB mode. Although the flag is set, this situation is not a key error indication in itself. The analyst must investigate why the issuing routine invoked SVC 13.

- Error Identifier - This field contains pertinent information regarding the error described by this SYS1.LOGREC entry, and provides a correlation to other SYS1.LOGREC entries. Related software and MCH records have the same sequence (SEQ) number that allows the correlation of records written in a particular recovery path (that is, FRR and/or ESTAE percolation, or MCH and subsequent software entries). For locked, disabled, or SRB routines, the processor identifier (CPU) indicates the processor on which the routine was running when it encountered an error. A zero processor identifier indicates that the record was written by an ESTAE routine (that is, the processor identifier is not uniquely identifiable). ASID indicates the current ASID at the time of the error. TIME indicates the time that the ERROR ID was generated. It is normally very close to the time that the record was written, as indicated in the first line of the record. TIME can be used to chronologically order related SYS1.LOGREC entries that contain the same SEQ number. This ordering is useful in reconstructing the environment as it was at the time of the error. Note that TIME changes only during recursion; percolation does not change TIME.

  If an SVC dump is taken, the ERROR ID as it appears in the SYS1.LOGREC record, will also appear in the SVC dump output and associated IEA911I message. Do not be concerned if the ERROR ID sequence numbers seem to have an increment of more than one. Although the RTM adds one to the sequence number of each unique entry (not percolation or recursion), there may be no associated recording of the error, thus, the sequence number is updated internally but is not always externally written. In particular, SDUMP and SNAP might get many expected program checks (which are not recorded) when determining which storage areas can be dumped.

- The dump status indicator (field SDWASDRC in the SDWA) indicates if an SVC dump was requested in association with the LOGREC record. EREP formats the following values:

Value	Meaning
0	SVC dump was not requested.
1	SVC dump was successfully started.
2	SVC dump was suppressed because another SVC dump was in progress.
3	SVC dump was suppressed by installation request via IPL or CHNGDUMP specification.
4	SVC dump was suppressed by ACTION=NODUMP on the SLIP command.
5*	SVC dump was suppressed because a SYS1.DUMP data set was not available.
6*	SVC dump was suppressed because an I/O error occurred while initializing the SYS1.DUMP data set.

8	SVC dump was suppressed because SRBs could not be scheduled to activate the dump tasks in the requested address spaces.
9	SVC dump was suppressed because a terminating error occurred in SDUMP before the first record of the dump could be written.
10	SVC dump was suppressed because a STATUS STOP SRB condition was detected, preventing I/O operations. (This indicator might not be formatted by EREP.)
11	SVC dump was suppressed by dump analysis and elimination (DAE) processing. (This indicator might not be formatted by EREP.)
12-254	Reserved.
255	SVC dump was suppressed for some other reason.

*Applies only for synchronous dump requests.

The SYS1.LOGREC data set is a vital tool in debugging. At times, the information in the LOGREC printout can be used to describe the entire problem situation. A search of the APAR data base on Retain for the CSECT, recovery routine, abend code, and reason code will often identify the problem as a known one.

SYS1.LOGREC Recording Control Buffer

This is one of the most important areas to be used when analyzing problems in MVS. The previous discussion of LOGREC records analysis generally applies to the in-storage LOGREC buffer as well.

This buffer serves as the interim storage location for hardware and software error records that are queued to be written to the SYS1.LOGREC data set. The buffer is significant because of the error history it contains. Also, any records in the buffer that have *not* reached SYS1.LOGREC are almost certainly related to the problem you are trying to solve.

Formatting the LOGREC Buffer

You can use the IPCS subcommand VERBEXIT LOGDATA to format the in-storage LOGREC buffer. Or, you can specify the LOGDATA verb under AMDPRDMP, as documented in *Service Aids*. In either case, the entries that are still in the buffer will be formatted in the same way as entries that are printed in the EREP detail edit report.

Finding the LOGREC and WTO Recording Control Buffers

There are two recording control buffers (RCBs) in the SQA. The system uses one buffer for LOGREC messages, and the other for WTO messages. The CVT+X'16C' (CVTRBCB) points to the recording buffers control block (RBCB). The RBCB contains the following information about the two recording control blocks (which are also referred to as RCBs or buffers):

For the LOGREC RCB:

- RBCB+X'10' (RBCBLRCB) points to the LOGREC buffer.
- RBCB+X'14' (RBCBLLEN) contains the length of the LOGREC buffer.

For the WTO RCB:

- RBCB+X'18' (RBCBWRCB) points to the WTO buffer.
- RBCB+X'1C' (RBCBWLEN) contains the length of the WTO buffer.

The LOGREC and WTO recording control buffers reside in fetch-protected SQA. Entries in these buffers have timestamps (8-byte TOD clock values) that allow you to look at a dump and create a chronological list of the LOGREC events and WTO messages.

Format of the LOGREC Recording Control Buffer

The LOGREC recording control buffer is a "wrap-table" similar to the system trace table. The entries are variable in size. The latest entries are the most significant especially if they have not yet been written to SYS1.LOGREC. Knowing the areas of the system that have encountered errors and the actions of their associated recovery routines, information obtained from SYS1.LOGREC and the LOGREC recording control buffer, helps provide an overall understanding of the environment you are about to investigate.

The *Debugging Handbook* describes the format of the RCB. However, a typical SDWA-type software entry in the LOGREC buffer has the following structure:

Contents of the Entry	Number of Bytes
Record entry header	16
Standard LOGREC entry header	24
Entry data, as follows: Job name SDWA (see the following note) Error identifier	variable, as follows: 8 variable 10
Length of the entry	4
Unused or free area	4

Note: The SDWA that is in the LOGREC buffer is a compressed SDWA in which the recordable extensions start directly after the used portion of the SDWAVRA. The SDWAURAL field contains the length of the SDWAVRA.

You can find the oldest entry in the buffer by locating the end of the unused or free area, obtained from RCBFREE+RCBFLNG. (If this sum brings you to a point beyond the end of the buffer, subtract RCBTLNG from the sum.) You can also read the buffer backwards by using the entry length at the end of each entry. The latest entry appears directly before the free or unused area of the buffer.

FRR Stacks

The FRR (functional recovery routines) stacks are often useful for understanding the latest processes on the processors. Entries are added and deleted dynamically as processing occurs. The PSA+X'380' contains the pointer to the current stack. The format is described in Data Areas section of the *Debugging Handbook* under FRRS. Experience has shown that the normal stack (located at X'C00' in each PSA) is perhaps the most useful, although all stacks have been beneficial on occasion.

The FRR stack +X'C' (FRRSCURR) points to the current recovery stack entry. (Unless FRRSCURR matches FRR stack +0 (FRRSEMP), in which case no recovery is present on the stack.) This entry +0 (FRRSFRRA pointed to by FRRSCURR) points to the recovery routine that is to gain control in case of error. The entry +4 (FRRSFLGS) contains flags used for RTM processing; a X'80' indicates this FRR is currently in control, a X'40' indicates that the FRR entry represents a nested FRR, and a X'08' indicates that the FRR is not allowed to retry. The next 24 bytes (FRRSPARM) serve as a work area for the mainline function associated with the FRR pointed to by this entry. This parameter area may contain footprints useful to your debugging efforts. The previous entry in the stack (X'20' bytes in front of the current) represents the next most current recovery routine. The stacks contain residual information associated with recovery that was previously active but is no longer valid. You should not rely on any information beyond the current entry.

Also consider the case where:

A gains control and establishes recovery;
A passes control to B;
B establishes recovery, performs its function, deletes recovery, and passes control to C;
C establishes recovery and subsequently encounters an error.

The FRR stack will contain entries for module A's and C's recovery routines. There is no indication from the FRR stack that B was ever involved in the process although it might have contributed to or even caused the error. The debugger gains an insight into the process but is not presented with the *exact* flow. Although you can get an idea of the general process or flow, do not make assumptions based solely on the FRR stack contents.

If you have trapped a specific problem, the stacks often contain valuable information. The same is true of a stand-alone dump taken because of a suspected loop. If RT1TLPN (a one-byte field) at FRR stack +X'28' is not zero, the FRR stack contains current, valid data. Following are some of the more valuable fields in the FRR stacks from a debugging viewpoint:

1. FRR stack+X'28' (FRRSRTMW) - Portion of RTM 1 work area (RT1W) resident in the stack

   In the case of an error, the RT1TENPT field (FRRSRTMW+2) indicates the error type as follows:

   X'01' - program check
   X'02' - restart key
   X'03' - SVC error (SVC was issued while in locked, disabled, or SRB mode)
   X'04' - DAT error
   X'05' - machine check
   X'06' - STERM reentry

2. RT1WRTCA (in RT1W work area pointed to by FRRSRTMA) - address of system diagnostic work area (SDWA)

3. RT1WMODE (in RT1W work area pointed to by FRRSRTMA) - mode at entry to RTM1

   X'80' - supervisor control mode (PSASUPER≠0)
   X'40' - physically disabled mode
   X'20' - global spin lock held
   X'10' - global suspend lock held
   X'08' - local lock held
   X'04' - Type 1 SVC mode
   X'02' - SRB mode
   X'01' - unlocked task mode

   This is the system mode at the time of entry to RTM1. The mode may change as processing continues through recovery; the current mode is at RT1WSRMD.

Extended Error Descriptor (EED)

The extended error descriptor (EED) passes error information between RTM1 and RTM2 and also between successive schedules of RTM1. The EED address is found at RT1W+X'3C' (RT1WEED), at TCBRTM12 (TCB+X'104'), or in the RTM2 SVRB at X'7C'. The EED, pointed to by RTM's SVRB, is generally not valid because RTM2 releases it early in its processing. The EED is described in the *Debugging Handbook*. Important EED fields are:

EED+0 (EEDFWRDP) - pointer to the next EED on the chain, or zero
EED+4 (EEDID) - description of contents of the rest of the EED
BYTE 0 = 1 - register and PSW information EED
= 2 - dump parameters EED
= 3 - machine check handler EED
= 4 - reserved
= 5 - dump storage range EED
= 6 - subpool list EED
= 7 - original error data EED (includes errorid)

For a software EED:

EED+X'C' (EEDREGS) - registers 0-15 at the time of the error

EED+X'4C' (EEDPSW) - PSW/instruction length code (ILC)/translation exception address (TEA) at time of error

EED+X'5C' (EEDXM) - control registers 3 and 4 at the time of the error.

RTM2 Work Area (RTM2WA)

This is the work area used by RTM2 to control abend processing. Registers, PSW, abend code, etc. at the time of the error are recorded in the RTM2WA. This area is often useful for debugging purposes and is described in the *Debugging Handbook* by RTM2WA. This work area can be found through TCB+X'E0' (TCBRTWA), or RTM2 SVRB+X'80'.

For information about how to use IPCS for online examination of control blocks, see the *Interactive Problem Control System (IPCS) User's Guide*.

Formatted RTM Control Blocks

RTM control blocks are formatted by AMDPRDMP as a TCB exit with the SUMMARY FORMAT, PRINT CURRENT, and PRINT JOBNAMES control statements, by IPCS as a TCB exit with the SUMMARY FORMAT subcommand, or with the ERR option under SNAP/ABEND. The formatted control blocks are all TCB-related, and are formatted only when they are associated with the TCB. The formatted control blocks are:

- FRRS (functional recovery routine stack) - points to the RT1W and is formatted with the current TCB if the local lock is held. (This control block is formatted only by AMDPRDMP and it is mutually exclusive of the IHSA).
- IHSA (interrupt handler save area) - has the normal FRR stack saved within it and is formatted with the TCB pointed to by the IHSA, if the address space was interrupted or suspended while the TCB was holding the local lock. (This control block is formatted only by AMDPRDMP and it is mutually exclusive of the FRRS.)
- RTM2WA (RTM2 work area) - formatted if the TCB pointer to it is not zero.
- ESA (extended save area of the SVRB) bit summary - formatted only if the RTM2WA formatted successfully and the related SVRB could be located.
- SDWA (system diagnostic work area) - formats the registers at the time of error only if the ESA formatted successfully and the SDWA could be located.
- EED (extended error descriptor block) - formatted if the TCB or RT1W pointer to it is not zero.
- SCB (STAE control block) - formatted under AMDPRDMP for abend tasks only. It is formatted under SNAP/ABEND whenever the TCB pointer to it is not zero.
- XSB (extended status block) - formatted if the XSB pointer in the the IHSA is not zero.
- STKE (stack element) - formatted if the STKE pointer in the XSB is not zero.

System Diagnostic Work Area (SDWA) Use in RTM2

RTM2 uses the SDWA to pass information to ESTAE recovery routines. RTM2 obtains two SDWAs: (1) the SDWA pointed to by RTM2WA+X'3DC' (RTM2SDW2), which is RTM2's copy of the SDWA, and (2) the SDWA pointed to by RTM2WA+X'354' (RTM2RTCA), which is the ESTAE routine's copy of the SDWA. (RTM2WA is pointed to from SVRB+X'80'.) Also, when a recovery routine is entered, register 1 contains the address of the ESTAE routine's SDWA. RTM2's SDWA is available when RTM2WA is available. If RTM2 cannot obtain storage for the ESTAE routine's SDWA, RTM2 enters the ESTAE routine without the SDWA.

RTM2 uses RTM2's SDWA for processing. Just before entering an ESTAE routine, RTM2 copies RTM2's SDWA into the ESTAE routine's SDWA. When the ESTAE routine returns to RTM2, RTM2 copies the ESTAE routine's SDWA back into RTM2's SDWA.

Effects of Multiprocessing On Problem Analysis

The multiprocessing (MP) capability of MVS allows multiple processors to share real storage using one control program. In MP, each processor has addressability to all of main storage and executes under the control of one set of supervisor routines.

Because various queue structures must be processed in a serial fashion, interlocking facilities are implemented in both the hardware and software to allow serialization of portions of the control program where conflicts may arise. Queue structures that don't require serialization are processed in parallel, that is, without regard to other processors.

Features of an MP Environment

The main features of a multiprocessing configuration are:

PSA - Each processor has a unique real storage frame, called a prefixed save area (PSA), referenced with addresses from 0 to 4K. (The second 2K of these 4K bytes are fetch protected.) Its location in real storage is in the processor's prefix register.

Inter-Processor Communication - Malfunction alerts (MFA) are automatically generated by failing processors before entering the check-stop state. Other inter-processor signaling is accomplished with the SIGP instruction. (This feature is discussed in detail later in this chapter.)

CONFIG Command - Performs three functions: (1) dynamically add or remove a processor from the configuration; (2) dynamically increase or decrease the amount of useable real storage; (3) control the availability of channel paths.

QUIESCE Command - Quiesces the system so that I/O pools or two channel switches or both can be reconfigured.

Locking - Access to various supervisory services is serialized by means of a software locking structure.

Dispatching - Assures that highest-priority ready work is processed by available processors.

PTLB (purge translation lookaside buffer) - When an entry is to be invalidated in a segment table, the translation lookaside buffer (TLB) on every processor must be purged before permitting subsequent references to the corresponding virtual segment.

IPTE (invalidate page table entry) - When an entry is to be invalidated in a page table, the IPTE is used to purge the entry in the TLB for the page being invalidated.

RMS - When components of the hardware operating system fail, it becomes the responsibility of the recovery management support (RMS) to help define the extent of the damage.

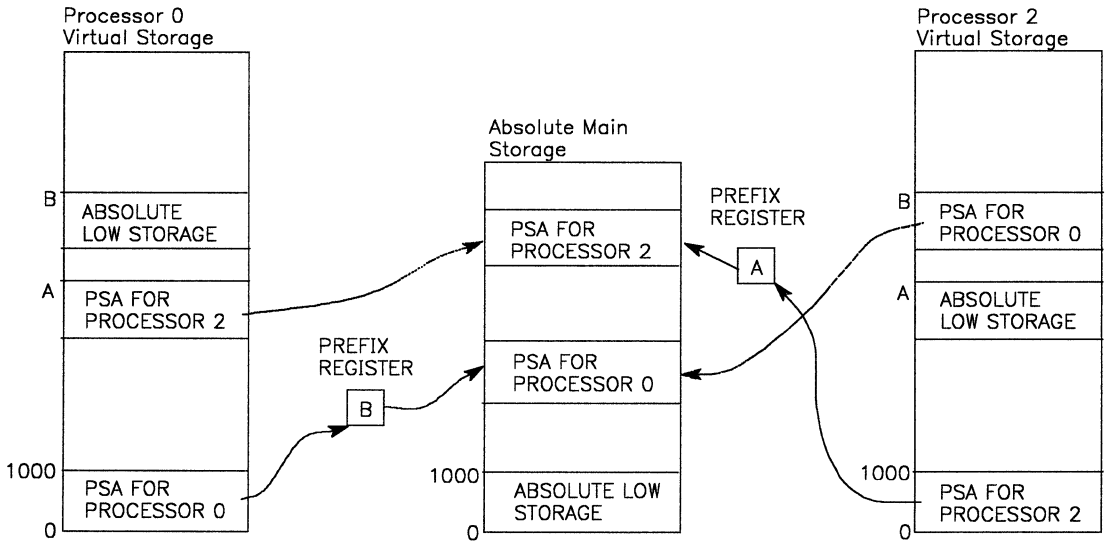
Compare and Swap - Two instructions assure interlocked update operations. They are Compare and Swap (CS) and Compare Double and Swap (CDS). References to storage for these instructions are interlocked the same way as the Test and Set (TS) instruction.

IOS - has the ability to initiate I/O activity to a device from any processor.

ACR - When one processor fails in an MP configuration, the alternate CPU recovery (ACR) function takes the failing processor offline and attempts to release any global resources held on that processor so that system operation can continue with the remaining processors. (See Miscellaneous Debugging Hints.)

CPU Affinity - The ability to force a job step to execute on a particular processor is a feature of MVS.

MP Storage Usage - The following diagram shows storage relationships.



Note that a processor's virtual PSA maps to real address 0. A processor can access another processor's PSA by using the virtual address of the other processor's PSA.

MP Dump Analysis

The first step of MP dump analysis is to determine conclusively that it is an MP dump. To do this, you must find the common system data area (CSD). The CSD address is located at offset X'294' in the CVT. The halfword CSDCPUOL, at offset X'A' in the CSD, gives the number of processors currently active. If this number is more than one, you are looking at an MP dump. For the rest of this discussion, we will assume that CSDCPUOL = 2.

Several other fields in the CSD are informative. For example, the byte CSDACR at offset X'16', indicates whether or not ACR is in progress. ACR in progress (X'FF' in CSDACR) indicates that one of the processors in the configuration is becoming inactive. If this is the case, the problem may be the result of a failure during ACR processing, and the MP dump will probably present at least two problems:

1. A failure causing ACR to be invoked.

2. A failure during ACR processing. (See the discussion on ACR processing in the "Miscellaneous Debugging Hints" chapter later in this section.)

Note that you can use the IPCS subcommand STATUS WORKSHEET to determine the number of CPUs in the system and whether alternate CPU recovery is in progress.

Data Areas Associated With the MP Environment

There are several processor-related areas with which you should be familiar:

1. The PCCA (physical configuration communication area)
2. The LCCA (logical configuration communication area)
3. The PSA (prefixed save area)

You can use the IPCS subcommand CBFORMAT to selectively locate and format a PCCA, LCCA, or PSA. (For example, CBFORMAT PCCA0 STRUCTURE(PCCA) formats the PCCA for CPU 0.) To format all PCCAs, LCCAs, and PSAs in the system, use the IPCS subcommand VERBEXIT CPUDATA.

There is a set of these control blocks for each processor located as follows:

CVT+X'2FC' points to the PCCAVT (contains the address of a PCCA for each processor)

CVT+X'300' points to the LCCAVT (contains the address of an LCCA for each processor)

PCCA+X'18' is the virtual address of the PSA for that processor

PCCA+X'1C' is the real address of the PSA for that processor

PSA+X'208' is the virtual address of the PCCA for that processor

PSA+X'20C' is the real address of the PCCA for that processor

PSA+X'210' is the virtual address of the LCCA for that processor

PSA+X'214' is the real address of the LCCA for that processor

The PSA is the "low storage area" (first 4K bytes of storage) and it contains, among other things, the hardware-assigned storage locations. *Principles of Operation* details the prefixing mechanism the hardware uses to reassign a block

of real storage for each processor to a different block in absolute main storage. Prefixing permits processors to share main storage and operate concurrently.

The PCCA contains information about the physical facilities associated with its processor, the LCCA contains save areas for use by the first level interrupt handlers (FLIHs). The need for processor unique areas arises, for example, because external interrupts could occur simultaneously on each processor, and therefore a processor-related area must exist for status saving by the external FLIH. Such areas are in the processor's PSA and LCCA. After locating these control blocks, you can determine several things about the status of each processor.

- The PSWs at the time of the last program, I/O, SVC, external, and machine check interrupts for each processor (PSA)
- The general purpose registers at each program check and machine check interrupt (LCCA)
- The mode (SRB or task) of each processor (LCCA or PSA)
- The address of the device causing the last I/O interrupt on each processor (PSA)

In addition, a work/save area vector table (WSAVTC) pointed to at LCCA+X'218' is associated with each processor. This vector table contains pointers to processor-related work/save areas. For example, there is a large save area for use by ACR, which is pointed to in the processor's WSAVTC. It is important to be aware of the existence of these processor-related areas because GTF, SRM, ACR, IOS, etc., use them; but you must narrow your problem to one of these processes (such as GTF, SRM, etc.) before the information in the associated work/save areas becomes helpful.

Parallelism

The most important characteristic of the MP capability is parallelism. In looking at MP dumps, you must always remember that several processes might run in parallel and reference the same main storage locations. As a result, queue structures and common data areas are vulnerable. In order to preserve their integrity, the system must insure that they are accessed serially. The resources that must be serialized in order to guarantee their integrity are called serially reusable resources (SRRs). The use of shared resources is the key item to be kept in mind in debugging an MP dump. There are various mechanisms available for serializing SRRs:

- ENQ/DEQ
- WAIT/POST
- Disablement
- Locking
- Compare and Swap instructions (CS and CDS)
- Nondispatchability
- Test and Set (TS) instruction
- RESERVE/RELEASE
- Intersect

You can use the IPCS subcommand ANALYZE for contention analysis on suspend locks, ENQs, and I/O devices, and some aid in RESERVE/RELEASE analysis.

Obviously, all users of a particular SRR must use the same serialization mechanism. The integrity of an SRR is reduced if one user uses locking and another uses ENQ/DEQ. You need to understand the processes going on in all processors at the time of the failure. The processor on which the failure occurred might *not* be the one that caused the problem.

Use of the work/save areas pointed to from the ASXB is a good example. These areas are serialized with the local lock. The following diagram shows what could happen if the same address space is running on two processors and one of the processes involved fails to serialize properly.

PROCESSOR 0	**PROCESSOR 2**
•	•
•	•
•	•
•	Gets local lock
Branch enters validity check routine	Branch enters validity check routine
•	
•	Releases local lock
	•
	•

In this example, assume that the process executing on processor 0 fails to get the local lock before it branch enters the system validity check routine. The validity check routine uses the local lock to serialize one of the save areas mentioned above in order to save the caller's registers. The registers saved by the validity check routine on processor 2 can be overlaid by the registers saved by the validity check routine on processor 0. Thus, the failure would be encountered on processor 2, but the processor 0 process would be the one that caused the failure.

OI/NI (OR Immediate and AND Immediate) instructions also illustrate this phenomenon. These instructions take more than one machine cycle to complete (that is, the operand is fetched, altered, and then stored). In previous operating systems, physical disablement and UP environments were enough to insure the completion of one instruction before another was executed. In MVS, with multiple processors, this is no longer true.

For example, suppose processor 0 issues OI and the operand has been fetched. Before processor 0 stores the changed byte, processor 2 executes the fetch cycle of an NI instruction to change a different bit in the same byte. Now, processor 0 stores the original status plus the OI change; subsequently the NI instruction completes, which erases the effect of the OI on the same byte. In MVS, locking is used to solve some of the problems arising from such multi-cycle instructions. When locking is not an appropriate solution, the Compare and Swap instructions could be the appropriate solution. CS serializes the word containing the byte against other processors. CSD serializes a doubleword. The point is that in debugging an MP dump, *all* processors must be considered because interaction between processes and shared resources is generally the key to solving the problem.

When a program serializes a resource incorrectly, other programs can alter the resource before the first program completes its update. The other programs may be running on other processors, or they may have received control on the same

processor because the first program was preempted (for example, SRB suspension because of a page fault) before completing its update. Proving that a problem resulted from incorrect serialization is accomplished by finding both the "other" program and the interval in which a program opens a serialization exposure.

The system trace table can sometimes be used to find potential "other" programs. If the occurrence of the error has not been overlaid in the trace table, it may be possible to reconstruct the series of events leading up to the failure by:

1. Listing all events on that processor, in order, using the logical processor address field in each event's trace entry
2. Making a similar list of all of the events on the other processor(s)
3. Comparing the lists to see if the processes executing in parallel on the processors are altering a common resource

Try to relate these processes that are executing in parallel to the serialization problem that caused the dump.

General Hints For MP Dump Analysis

The following is a list of general hints to help you analyze an MP dump.

1. The use of PRIORITY and DPRTY parameters no longer ensures the order in which tasks are dispatched. First, the SRM, when attempting to handle resources, can allow a task or job with a lower DPRTY to run prior to a job with a higher priority. Second, as the dispatcher dispatches tasks on other processors, tasks of different priority may be executing on multiple processors simultaneously.
2. The CHAP (change priority) SVC does not ensure that tasks are dispatched in the expected order when they are dispatched on other processors.
3. Attached tasks can execute at the same time as the mother task on different processors. Therefore, if both tasks reference the same data, serialization of the data is required.
4. Any references made to system control blocks that change dynamically after IPL must be serialized to preserve the integrity of the data. The serialization technique for the data item must match that employed by the system.
5. Tasks can be redispatched on a different processor from the one on which they were previously operating. Therefore, do not use the LCCA, PCCA, WSA, or PSA when enabled for interrupts, because redispatch on a different processor results in different data being referenced.
6. If subpools are shared between tasks, users must serialize the use of any data in the subpools common to the two tasks.
7. SRBs can be dispatched on any processor unless they are scheduled with affinity for a particular processor.
8. Asynchronous appendages on one processor can operate simultaneously with the task on another processor.

9. Enabled recovery routines can run on any processor, not necessarily the one on which the error was detected.

10. STATUS STOP SRB request does not prevent SRBs from being added to the local queue; it merely quiesces the address space after any currently executing or suspended SRBs have completed.

11. When access methods allow sharing of data sets between tasks in the same address space, access to the data sets must be serialized between the tasks.

Inter-Processor Communication

MVS uses the inter-processor communication (IPC) function in doing its inter-processor related work. The IPC function uses the SIGP (Signal Processor) instruction to provide the necessary hardware interface between processors.

The functions provided by the SIGP instruction depend on the order code that is specified and when the SIGP is issued. For a detailed description of the SIGP instruction and order codes, refer to *Principles of Operation.*

The functions are divided into two classes, direct and remote. The following briefly describes IPC's implementation. For more information, refer to *SPL: System Macros and Facilities* and *System Logic Library.*

1. ***Direct*** - The direct service is defined for those control program functions that require the modification or sensing of the physical state of a processor. The direct service function consists of a macro (DSGNL) and a service routine (IEAVEDR).

2. ***Remote*** - The remote services are defined for those control program functions that require the execution of a software function on a processor. They provide the hardware interface and interruption mechanism to initiate the desired program on the proper processor. The remote class is subdivided into the remote pendable service and the remote immediate service.

   - The **remote pendable service** consists of a macro (RPSGNL) and a service routine (IEAVERP) that executes on the sending processor, and an interruption processing routine (IEAVEXS) that executes on the receiving processor. The sending routine establishes the interface and issues the SIGP instruction with the external call order code. The receiving routine gets control from the external FLIH (IEAVEEXT) when the processor is enabled for the class of interruption. The communication interface established is in a field in the PCCA of the receiving processor. This allows multiple requests to be handled via one interruption.

   - The **remote immediate service** consists of a macro (RISGNL) and a service routine (IEAVERI) that executes on the sending processor, and an interruption processing routine (IEAVEES) that executes on the receiving processor. The sending routine establishes the interface and issues the SIGP instruction with the emergency signal order code. The receiving routine gets control from the external FLIH (IEAVEEXT) when the processor is enabled for the class of interruption. The interface used to communicate between the processors involved is a buffer in the PCCA of the sending processor.

Note that for both the direct and remote services the following applies:

- The initiated function can be directed to any processor, including the processor initiating the request.
- The service routines invoke a common routine (IEAVESGP) that issues the SIGP instruction.

MP Debugging Hints

1. Apparent disabled loop in IEAVERI on processor A

   This is probably caused when processor A sends an EMS (emergency signal) to processor B, but the EMS SLIH (IEAVEES) on processor B has not yet turned off the serial or parallel bit in processor A's PCCA. Thus, processor A is in the "window spin" state in IEAVERI.

   To find what processor A wanted processor B to do, locate processor A's PCCA using one of the following:

   - Field PSAPCCAV (PSA+X'208') in processor A's PSA contains the virtual address of the PCCA for processor A.
   - Field CVTPCCAT (CVT+X'2FC') points to the PCCAVT. The virtual address of the PCCA for processor A is found by indexing into the PCCAVT using four times the CPUID in PSACPUPA (PSA+X'204').

   PROCESSOR A's PCCA

X'88'	RISP	EMS2	EMS3	
X'8C'	Receiving Routine PARM address			
X'90'	Receiving Routine EP address			
X'94'	Receiving Processor's PCCA address			

   RISP field

   X'80' - Parallel request

   X'40' - Serial request

   EMS2 field

   X'80' - Serial pending

   EMS3 field

   X'80' - Serial receiving routine failed

By locating the proper PCCA (in this case processor A's), you can determine whether the EMS request was parallel or serial, the entry point, and therefore, the name of the receiving routine. The external interrupt is taken when the receiving processor is enabled. The EMS SLIH processes the request and indicates the state of the processing by updating the first word of the PCCA buffer of the sending processor.

The following are possible situations that can cause a disabled loop:

- Processor B, if disabled for EMS interrupts, would never take the EMS interrupt; therefore the receiving routine would never get control and the parallel or serial bit would never get turned off.
- There could be a hardware problem with the SIGP circuitry. For example, if IEAVERI got condition code 0 as a result of issuing the SIGP instruction on processor A, but the SIGP was never received on processor B, there would be a loop in IEAVERI.
- If the receiving routine loops or hangs for a serial request, IEAVERI will also loop with the serial bit on and the serial pending bit off.

2. Locate External Call buffers

The external call buffer is located at offset X'84' in the PCCA. Normally, the buffer is clear, but it is worthwhile to check to make sure that there is no external call work to process, as indicated by the request codes below:

Request Code (PCCA + X'84'):

X'80' - SWITCH
X'40' - RQCHECK
X'10' - GTFCRM
X'04' - MODE
X'01' - MEMSWT

The code is set in the *receiving* processor's PCCA so that a bit on in processor B's PCCA, for example, means that another processor initiated the request. (Note that multiple bits can be on at the same time.)

Note: If the external call or EMS events are available in the system trace table, those table entries contain event-related information (for example, the external call buffer). For details see the following topic "MVS Trace Analysis" and the TTE in the *Debugging Handbook*.

MVS Trace Analysis

This chapter describes the system trace, the master trace, and the message processing facility.

The generalized trace facility (GTF) also provides tracing of system events and can execute concurrently with system trace. For a description of GTF, see *Service Aids*; and for the format of GTF trace records, see the *Debugging Handbook*.

At system initialization, both system trace and master trace are activated. The system trace options are explicit tracing, branch tracing, and address space tracing. After system initialization, you use the TRACE operator command to change the status of system trace and master trace. For a description of the TRACE operator command, see *System Commands*.

System Trace

System trace provides an on-going record, in the system trace table, of significant software and hardware events that you can use in the determination of system problems. The system trace table is located in the private area of the TRACE address space.

You can use the IPCS subcommand VERBEXIT TRACE to format the system trace table.

For each processor, there is a queue of four 4K buffers in the system trace table. The amount of space allocated for the trace buffers can be changed using the TRACE ST operator command.

The following topics contain:

- The types of system trace entries
- The format of the system trace table
- An example of a formatted trace table
- Notes for traces
- The unformatted system trace table

Types of System Trace Entries

The types of trace entries are: explicit, branch, and address space.

The explicit trace entries are:

SSCH	Start subchannel
MSCH	Modify subchannel
HSCH	Halt subchannel
CSCH	Clear subchannel
RSCH	Resume subchannel
EXT	External interruption
EMS	EMS external interruption
SS	Service signal external interruption
CALL	External call external interruption
CLKC	Clock comparator external interruption
SVC	SVC interruption
SVCR	SVC return
SVCE	SVC error

PGM	Program interruption
SPER	SLIP/PER interruption
I/O	I/O interruption
DSP	Task dispatch
SRB	Initial SRB dispatch
SSRB	Suspended SRB dispatch
WAIT	Wait task dispatch
MCH	Machine check interruption
RST	Restart interruption
ACR	Alternate CPU recovery
SUSP	Lock suspension
ALTR	Trace options alteration
USRn	User event trace

The branch trace entry is:

BR	Branch trace (BALR, BASR, and BASSM)

The address space trace entries are:

SSAR	Set Secondary Address Space
PC	Program Call
PT	Program Transfer

For the detailed format of these entries, see the trace table entry (TTE) in the *Debugging Handbook*.

Format of the System Trace Table

The format of each trace entry and a brief description of each field are shown in Figure 2-6. An example of a formatted system trace table is shown in Figure 2-7. The trace table headers, shown at the top of the figures, have the following meanings:

PR
Indicates the processor identifier (ID) of the processor that produced the trace table entry (TTE).

ASID
Indicates the address space identifier (ASID) associated with the TTE. For explicit TTEs (except subchannel-related TTEs), this is the home ASID of the address space under which the event was traced. For the subchannel-related TTEs, this is the ASID associated with the I/O. For branch and address space TTEs, this is the home ASID from the most recent explicit TTE in the trace table.

TCB-----
Indicates the TCB address (or a related TCB address) of the task in progress for which the TTE was produced.

IDENT
Indicates the identification of the type of TTE. An asterisk (*) indicates an entry that has significant debugging value, for example:

- I/O alert
- I/O path not operational
- SVC error
- Program check (other than a segment or page fault)

- Machine check
- SVC 13 (other than a normal end of task)
- Restart
- Alternate CPU recovery
- Change in trace options
- SSCH nonzero condition code
- Malfunction alert (MFA)
- External interruption key

If the trace formatter cannot identify an explicit entry, ?EXPL is used for the TTE identification.

CD/D
Contains an interruption code for EXT and PGM entries, an SVC number for SVC entries, a device number for subchannel and I/O entries, or a physical processor's address for the ACR entry.

PSW----- ADDRESS
For many explicit TTEs, these two fields contain the first and second words of the PSW related to the TTE. (See Figure 2-6 for the contents of this field for the various entries.)

UNIQUE-1/UNIQUE-2/UNIQUE/3
UNIQUE-4/UNIQUE-5/UNIQUE/6
Contains unique information for the various TTEs. (See Figure 2-6 for the contents of this field for the various entries.)

PSACLHS-
Contains the PSA current lock held string at the time of the trace (or other locking information). For example, on I/O interrupts the UCB lock appears to be held.

PSALOCAL
Contains the locally locked address space indicator. If the CML lock was held when the TTE was created, this field contains the ASCB address of the CML lock.

PASD
Contains the primary ASID (PASID) at the time of the trace entry.

SASD
Contains the secondary ASID (SASID) at the time of the trace entry.

TIMESTAMP-RECORD
Contains the time-of-day (TOD) clock value at the time that the explicit entry was created. (This timestamp-record is formatted the same as the timestamp-record on SYS1.LOGREC records, which facilitates correlation of trace entries to LOGREC records.)

SYSTEM TRACE TABLE

PR	ASID	TCB-----	IDENT	CD/D	PSW-----	ADDRESS-	UNIQUE-1	UNIQUE-2	UNIQUE-3	PSACLHS-	PSALOCAL	PASD	SASD	TIMESTAMP-RECORD
							UNIQUE-4	UNIQUE-5	UNIQUE-6					
pr	asid	tcb-addr	SSCH	dev	cc di	iosbaddr	ucb-addr	orb-wrd2	orb-wrd3					timestamp-record
pr	asid	tcb-addr	MSCH	dev	cc	iosbaddr	ucb-addr	flf2pmom	mbi-t2lb					timestamp-record
pr	asid	tcb-addr	HSCH	dev	cc di	iosbaddr	ucb-addr	ioq-addr	asc-iosb					timestamp-record
pr	asid	tcb-addr	CSCH	dev	cc di	iosbaddr	ucb-addr	ioq-addr	asc-iosb					timestamp-record
pr	asid	tcb-addr	RSCH	dev	cc di	iosbaddr	ucb-addr							timestamp-record
pr	home	tcb-addr	EXT	code	ext-old-	psw-----	psaeepsw			psaclhs-	psalocal	pasd	sasd	timestamp-record
pr	home	tcb-addr	EMS		ext-old-	psw-----	psaeepsw	pccaemsi	pccaemsp	psaclhs-	psalocal	pasd	sasd	timestamp-record
							pccaemse							
pr	home	tcb-addr	SS		ext-old-	psw-----	psaeepsw	psaeparm	msf-bcmd	psaclhs-	psalocal	pasd	sasd	timestamp-record
							flg-brsp	mssfasid	mssfatcb					
pr	home	tcb-addr	CALL		ext-old-	psw-----	psaeepsw	pccarpb-		psaclhs-	psalocal	pasd	sasd	timestamp-record
pr	home	tcb-addr	CLKC		ext-old-	psw-----	psaeepsw	tge-tcb-	tge-asid	psaclhs-	psalocal	pasd	sasd	timestamp-record
pr	home	tcb-addr	SVC	code	svc-old-	psw-----	gpr15---	gpr0----	gprl----					timestamp-record
pr	home	tcb-addr	SVCR	code	ret-new-	psw-----	gpr15---	gpr0----	gprl----					timestamp-record
pr	home	tcb-addr	*SVCE	code	svc-old-	psw-----	gpr15---	gpr0----	gprl----	psaclhs-	psalocal	pasd	sasd	timestamp-record
							psamodew							
pr	home	tcb-addr	PGM	code	pgm-old-	psw-----	ilc-code	tea-----		psaclhs-	psalocal	pasd	sasd	timestamp-record
pr	home	tcb-addr	SPER	code	pgm-old-	psw-----	ilc-code	tea-----	trap	psaclhs-	psalocal	pasd	sasd	timestamp-record
							per-addr							
pr	home	tcb-addr	I/O	dev	i/o-old-	psw-----	flg-ctl-	ccw-addr	dvch-cnt	psaclhs-	psalocal	pasd	sasd	timestamp-record
							ext-stat	ucb-addr						
pr	home	tcb-addr	DSP		dsp-new-	psw-----	psamodew	gpr0----	gprl----	psaclhs-	psalocal	pasd	sasd	timestamp-record
pr	home	rel-tcb-	SRB		srb-new-	psw-----	safnasid	gpr0----	gprl----	srbhlhi-		pasd	sasd	timestamp-record
							flg-srb-							
pr	home	rel-tcb-	SSRB		ssrb-new	psw-----	safnasid		gprl----	psaclhs4	psalocal	pasd	sasd	timestamp-record
pr	home	tcb-addr	WAIT											timestamp-record
pr	home	tcb-addr	*MCH		mch-old-	psw-----	machine-	chk-code	psasuper	psaclhs-	psalocal	pasd	sasd	timestamp-record
pr	home	tcb-addr	*RST		rst-old-	psw-----	gpr15---	gpr0----	gprl----	psaclhs-	psalocal	pasd	sasd	timestamp-record
							psasuper	psamodew						
pr	home	tcb-addr	*ACR	cpu			psaeepsw	flg-crex	psacstk-	psaclhs-	psalocal			timestamp-record
							psasuper	psamodew						
pr	home	tcb-addr	SUSP			return--	rb-addr-	suspndid	reladdr-	psaclhs-	psalocal			timestamp-record
							ssrbaddr							
pr	home	tcb-addr	*ALTR				tobtropt	gpr0----	gprl----			pasd	sasd	timestamp-record
							pol-buf-							
pr	home	tcb-addr	USRn			return--	word1---	word2---	word3---			pasd	sasd	timestamp-record
							word4---	word5---						
pr	home	tcb-addr	USRn			return--	idc- rbc	data----	data----			pasd	sasd	timestamp-record
							data----	data----	data----					
pr	home	tcb-addr	USRn			return	idc- rbc	data----	etc.			pasd	sasd	timestamp-record
pr	home	tcb-addr	BR		address-	address-	address-	address-	address-	address-	etc.			
pr	home	tcb-addr	SSAR		newsasid								sasd	
pr	home	tcb-addr	PC		psw-key-	pc-addr-	pc#-----							
pr	home	tcb-addr	PT		psw-key-	pt-addr-	pt-asid-					pasd	sasd	

Figure 2-6 (Part 1 of 3). Format of Trace Table Entries

Note: If the trace formatter cannot identify an explicit entry, ?EXPL is used for the TTE identification in the trace entry.

address-
successful branch address, repeated for consecutive branches on the BR entry.
- A 4-byte address indicates a 31-bit address.
- A 3-byte address indicates a 24-bit address.

asc-iosb
IOSB address of the associated SSCH request.

asid
address space identifier related to the I/O.

cc
condition code associated with the I/O.

ccw-addr
CCW address associated with the I/O.

code
external, PER, or program interruption code, or SVC number.

cpu
PSACPUPA field from the PSA (failing processor address).

data----
user-defined data.

dev
device number associated with the I/O.

di
driver identifier associated with the I/O.

dsp-new- psw-----
PSW to be dispatched.

dvch-cnt
dv = device status.
ch = subchannel status.
-cnt = residual count.

ext-old- psw-----
external old PSW.

ext-stat
extended status word.

flg-brsp
flg- = MSSF hardware flags.
brsp = MSSF response code.

flg-crex
LCCACREX field of the LCCA for the failing processor.

flg-ctl-
flg- = IRBFLAGS field.
ctl- = subchannel control bytes.

flg-srb-
SRBFLGS field from the SRB.

flf2pmom
the following from the SCHIB associated with the I/O:
f1 = SCHFLG1 flags field.
f2 = SCHFLG2 flags field.
pm = SCHLPM field.
om = SCHPOM field.

gpr0----/gpr15---
contents of general purpose register 0/15.

home
home ASID associated with the entry. For ACR entry, this is the home ASID of the failing processor. For BR entries, this is the last home ASID in the trace buffer.

i/o-old- psw-----
I/O old PSW.

idc- rbc
continuation information:
idc- = PTRACE identification count.
rbc = relative byte count.

ilc-code
ilc- = instruction length code.
code = interruption code.

ioq-addr
IOQ address associated with the I/O

iosbaddr
IOSB address associated with the I/O.

machine- chk-code
machine check interruption code from FLCMCIC field in the PSA.

mbi-t2lb
mbi- = SCHMBI field from the SCHIB.
t2 = IOSOPT2 field from the IOSB.
lB = IOSFLB field from the IOSB.

mch-old- psw-----
machine check old PSW.

msf-bcmd
service processor command word.

mssfasid
service processor ASID.

mssfatcb
service processor TCB address.

newsasid
new SASID from the SSAR instruction.

orb-wrd2
word 2 of the ORB associated with the I/O.

orb-wrd3
word 3 of the ORB associated with the I/O.

pasd
primary ASID (PASID) at the time of the entry.

pc-addr-
return address from the PC instruction.

pc#-----
PC number from the PC instruction.

pccaemse
PCCAEMSE field from the PCCA.

pccaemsi
PCCAEMSI field from the PCCA.

pccaemsp
PCCAEMSP field from the PCCA.

pccarpb-
PCCARPB field from the PCCA.

per-addr
SLIP/PER status address.

pgm-old- psw-----
program old PSW.

pol-buf-
pol- = TOBTRPOL field from the TOB, containing the number of processors with tracing active or suspended.
buf- = TOBTRBUF field from the TOB, containing the number of trace buffers per processor.

pr
processor identifier.

psaclhs-
PSACLHS field from the PSA. (For the ACR entry, from the failing processor.)

psaclhs4
PSACLHS4 field from the PSA.

psacstk-
PSACSTK field from the PSA (from the failing processor).

psaeepsw
PSAEEPSW field from the PSA. For EMS and CALL entries, bytes one and two contain the issuing processor's address. For the ACR entry, bytes one and two contain the failing processor's address. For all entries, bytes three and four contain the external interruption code.

Figure 2-6 (Part 2 of 3). Format of Trace Table Entries

psaeparm
PSAEPARM field from the PSA, containing the MSSF buffer address.

psalocal
PSALOCAL field from the PSA. (For ACR entry, from the failing processor.)

psamodew
PSAMODEW field from the PSA. (For ACR entry, from the failing processor.)

psasuper
PSASUPER field from the PSA. (For ACR entry, from the failing processor.)

psw-key-
PSW key.

pt-addr-
new instruction address as updated by the PT instruction.

pt-asid-
new ASID specified on the PT instruction.

rb-addr-
suspended RB address.

rel-tcb-
related TCB address.

reladdr-
address associated with the type of lock suspension:
- 0 for LOCL lock.
- ASCB address for CML lock requested.
- lockword address for CMS/CEDQ/CSMF locks.

ret-new- psw-----
PSW to receive control on the redispatch of the SVC.

return--
return address of the caller.

rst-old- psw-----
restart old PSW.

safnasid
safn = LCCASAFN field from the LCCA.
asid = related ASID.

sasd
secondary ASID (SASID) at the time of the entry.

srb-new- psw-----
PSW to receive control on the SRB dispatch.

srbhlhi-
SRBHLHI field from the SRB.

ssrb-new psw-----
PSW to receive control on the SSRB redispatch.

ssrbaddr
SSRB address.

suspndid
lock suspension type identifier: LOCL, CML, CMS, CEDQ, or CSMF.

svc-old- psw-----
SVC old PSW.

tcb-addr
TCB address of the current task.

tea-----
translation exception address. (The high order bit indicates primary (0) or secondary (1).

timestamp-record
the eight-byte TOD clock value at the time the entry was made.

tobtropt
TOBTROPT field from the TOB, containing trace options in control register 12 format.

tqe-asid
ASID of the associated TQE.

tge-tcb-
TCB address of the associated TQE.

trap
SLIP/PER trap id.
ID=xxxx

ucb-addr
UCB address associated with the I/O.

word1---/word6---
user-defined data.

Figure 2-6 (Part 3 of 3). Format of Trace Table Entries

```
------------------------------------------------------ SYSTEM TRACE TABLE ------------------------------------------------------
--                                                     ------------------                                                       --
--                                                     ------------------                                                       --
PR ASID TCB-ADDR  IDENT CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD TIMESTAMP-RECORD
                                               UNIQUE-4 UNIQUE-5 UNIQUE-6

02 000F E3C3C27C   SSCH  0FF 00  0F   00007BF4  D6D9C2F1 D6D9C2F2 D6D9C2F3                                    93320B120C73AE60
00 000F E3C3C27C   MSCH  0FF 00       00007BF4  D6D9C2F1 D6D9C2F2 D6D9C2F3                                    93320B120C74DC40
00 000F E3C3C27C   HSCH  0FF 00  0F   00007BF4  D6D9C2F1 D6D9C2F2 D6D9C2F3                                    93320B120C755440
00 000F E3C3C27C   CSCH  0FF 00  0F   00007BF4  D6D9C2F1 D6D9C2F2 D6D9C2F3                                    93320B120C75C220
02 000F E3C3C27C   RSCH  0FF 00  0F   00007BF4  00007A2E                                                    93320B120C761480
00 0009 009FF268   EXT   1211 070E0000 00000000  00011211                    80808080 44444444 0001 0009 93320B120C783240
02 0009 009FF268   EMS         070E0000 00000000  00001201 89999999 88888888 80808080 44444444 0001 0009 93320B120C79DC80
                                                  77777777
02 0009 009FF268   SS          070E0000 00000000  00012401 00007B64 00000000 80808080 44444444 0001 0009 93320B120C7B0000
                                                  00000000     0000 00000000
00 0009 009FF268   CALL        070E0000 00000000  00011202 06666666           80808080 44444444 0001 0009 93320B120C7F3820
00 0009 009FF268   CLKC        070E0000 00000000  00011004 00000000     0000  00001000 00000000 0001 0009 93320B120C812460
00 0009 009FF268   SVC     8A  070C1000 00006748  D9C5C7C6 D9C5C7F0 D9C5C7F1                                 93320B120C81B020
02 0009 009FF268   *SVC     D  070C1000 00006748  D9C5C7C6 D9C5C7F0 F7FFFFFF                                 93320B120C822A60
00 0009 009FF268   SVCR    8A  070C1000 00006748  D9C5C7C6 D9C5C7F0 D9C5C7F1                                 93320B120C830E20
02 0009 009FF268   *SVCE    FF C6D3C3E2 D6D7E2E6  D9C5C7C6 D9C5C7F0 D9C5C7F1 80808080 44444444 0009 0009 93320B120C845240
                                                  03333333
00 0009 009FF268   PGM     011 01010101 02020202  00040011 03030303           80808080 44444444 0009 0009 93320B120C864080
00 000B 009FF458   SPER     C3 D7C7D460 D7E2E640  D3C3D7C3 E3C5C140 ID= ABCD  80808080 44444444 000B 000B 95A164D2A354CA20
                                                  D7C5D9C1
00 0009 009FF268   I/O    0FF C6D3C3C9 D6D7E2E6  40404040 D6D9C2F1 D6D9C2F2  80808080 44444444 0009 0009 93320B120C896060
                                                  D6D9C2F3 00007A2E
00 0009 009FF268   DSP         C6D3C3E2 E5D7E2E6  03333333 E3C3C2F0 E3C3C2F1  80808080 44444444 0009 0009 93320B120C8B5C60
02 0009 E3C3C27C   SRB         070C0000 8104B380  0041000F 00007C60 00000FFF        77          0009 0009 93320B120C8BFE00
00 0009 E3C3C27C   SSRB        070C1000 00006748  FFFF7777           00000FFF        00          0000 7974 93320B120C8CBC40
00 0009 009FF268   WAIT                                                                                    93320B120C8CBD00
02 0009 009FF268   *MCH         C6D3C3D4 D6D7E2E6  C6D3C3D4 C3C9C340 99999999  80808080 44444444 0009 0009 93320B120C8DCA40
00 0009 009FF268   *RST         C6D3C3D9 D6D7E2E6  D9C5C7C6 D9C5C7F0 D9C5C7F1  80808080 44444444 0009 0009 93320B120C8EE660
                                                  99999999 03333333
02 0001 67676767   *ACR  0000                     00017777 FF       45454545  80808080 44444444           93320B120C912E20
                                                  99999999 03333333
00 0009 009FF268   SUSP                 00007832  009FF1C0 LOCL     00F7B438  00001000 00000000           93320B120C926460
                                                  00000000
00 0009 009FF268   *ALTR                          80000003 FFFFFFFF AAAAAAAA                 0009 0009 93320B120C933460
                                                  00020004
00 0009 009FF268   USR1                 81A00722  00000001                                  0009 0009 93320B310124AA40
00 0009 009FF268   USRA                 81A00A02  0005 000 00000001 00000002              0009 0009 93320B3103FACA20
                                                  00000003 00000004 00000005
00 0009 009FF268   USRA                 81A00A02  0005 014 00000006                       0009 0009 93320B3103FB0A80
00 0009 009FF268   BR          0100A308 00FF62DA  010047E0 010ADDB0 01031AC8  01003400 0103B3C8
                   BR          01031AC8 01003400  0103B3C8 00FF9170
00 0009 009FF268   BR            FF8E24 00FFF8E8
00 0009 009FF580   PC      ...    0      81023340  00108
00 0009 009FF580   PT      ...    0      81023340  0009                                      0009 0009
00 0009 009FF580   SSAR    ...      0009                                                          0009
```

Figure 2-7. Example of Formatted System Trace Table

The TRACE Address Space

The initialization routines for the system trace address space (TRACE) activate the system trace with default options. The system trace can be deactivated or the tracing options changed by the TRACE operator command, processed from the COMMNDxx parmlib member at IPL, or issued by the operator after IPL.

The TRACE address space is terminated when (1) the operator issues the FORCE command for the TRACE address space or (2) the system detects a critical problem in the address space. A subsequent issuance of the TRACE operator command with any ST option (except OFF) will recreate the system trace address space and activate the system trace with the specified or default options.

Notes For Traces

System trace provides a history of some of the events that lead to a storage dump. Trace interpretation is one of the most important aspects of debugging.

Tracing Procedure

When attempting to recreate the process that was occurring on the processor(s) when the dump was taken, start at the current entry in the trace table (identified by the highest clock value in the last column) and scan upwards. While scanning, look for unexpected events. These include:

- Unit check, unit exceptions on I/O devices
- Non-CC=0 on SSCHs and RSCHs
- Non-type 10 or 11 program checks
- SVC D, SVC 33, SVC errors - (see number 5 under "Cautionary Notes" later in this chapter)
- Malfunction alerts (X'1200' external interrupt)
- Entries that show both processors executing the same code as indicated by the ICs (instruction counter) in the entries
- Large time gaps in the TOD clock value (which are accompanied by a time-gap message in the formatted output)
- MP environment and only one processor doing anything

These entries indicate a potential for errors. Do not be distracted if you discover an entry of this type. Record the incident for future use. Then continue scanning back through the trace and try to determine what was happening in the system that might have caused the failure. Remember to conduct the scan by unique processor. Separate the processes that occur on each processor and watch for any obvious interactions in the processes.

You can further subdivide the activity by address space (as depicted by ASID) or by task (TCB address; remember to stay under the same ASID). As you recreate the situation, remember that you are relating individual entries to real events that must occur in order to accomplish work. Do not be distracted. For example, do not look for an I/O interrupt just because you see a SSCH. The two events should be associated, but you should also determine the following:

- Why the I/O is occurring;
- If the I/O is related to the process, address space, task, page fault, etc. that you are concerned with;
- If the I/O completion should trigger another event. This is the way work is accomplished in MVS, that is, events triggering more events. As you become familiar with trace coding you learn to expect this "event causing" sequence. Certain sequences occur very frequently; you learn to recognize these and to look for less familiar sequences.

As you are searching trace entries, watch for repeating patterns, which can indicate a loop in the system. These patterns can appear as constantly repeating ICs (generally the case in a tight enabled loop), or as a repeating sequence of entries (often the case in a process loop, such as an ERP constantly retrying an I/O operation). Note that in the latter case, other entries from other processes can intervene periodically in the trace table, especially in an MP environment.

If you reach a point in the trace analysis where you are somewhat comfortable with the processes you are uncovering and recreating, and you feel you have a fair understanding of the activity in the system, pause. Try to understand what you have found. Is there any way you can relate your findings to the reason you have taken the dump in the first place? Do the unexpected events have anything to do with the problem, or are they unrelated to the problem? It can happen that the events you have discovered are unrelated to the problem causing the dump and you have exhausted the scope of the trace. In this case, you probably have to go into the system and study the address space and task structures, queues, and global data areas in order to zero in on the problem.

However, if the events you have discovered are related to the problem causing the dump, you must then attempt to isolate the erroneous process. Try to understand how the unexpected events relate to the process. Look on both sides of the event: did the event trigger the bad process, or is it a result of the bad process?

It is also necessary in trace analysis in MVS to understand whether you are looking at the primary error or at some secondary problem. Is this a mainline failure or a failure because of a problem in the recovery? Also, you must decide if the problem is caused by a previous error from which the system has recovered. Always be sure that it was not something several pages earlier in the trace that caused recovery to be activated and eventually led to the current problem. If this is the case you must now decide which error to pursue. The original error is probably more important; however, much of the required information might be lost because of recovery and the subsequent recovery failure. Also keep in mind that if you must attack the secondary error condition, your search of the dump and the recovery areas can often uncover information about the first error.

The trace is one of the most useful tools available for back-tracking through a problem sequence. You must use it in conjunction with system control blocks and indicators in order to recreate the error sequence.

Cautionary Notes

Listed below are some items the problem solver should understand when analyzing an MVS trace table.

1. *I/O Processing:*

   - Much I/O is accomplished in MVS by the branch entry interface to IOS and without the use of SVC 0 (EXCP). Therefore, you often find I/O entries (SSCH-I/O interruptions) that are not accompanied by SVC 0.

   - Back-end I/O processing can result in an SRB schedule of IECVPST. This trace entry should appear soon after an I/O interrupt. The register 1 slot will contain the IOSB address. The IOSB is the key to tracking the I/O request.

   - Some I/O is accomplished in MVS by invoking IOSRSAIO, which is a synchronous I/O routine. Because IOSRSAIO runs disabled, it cannot hook to a trace routine to make a trace entry. Therefore, you will not see any system trace or GTF trace entries for functions (such as disabled console support) that use the IOSRSAIO routine.

2. *Timer Value:*

   The last field of each trace entry contains the eight-byte TOD clock at the time the entry was made. The eighth digit (from the left) represents the value to be increased approximately every second.

   The following steps show how to determine the elapsed time between two trace entries (such as from a SSCH to the I/O interruption).

   a. Using bytes 3-6 of the TOD clock value, find the difference between the two hexadecimal timer values.

   b. Convert the difference to a decimal value.

   c. Divide the decimal value by 16 (result is microseconds).

   d. Divide by 1,000,000 (result is seconds).

3. *Enabled Wait State:*

   Because of recovery, the end symptom of many problems is an enabled wait state. For tracing, the wait state presents particular problems in MVS. SRM maintains a timer interval that periodically causes a clock comparator interrupt. Also, an SRB is periodically dispatched in the master scheduler's address space to run a section of SRM code which updates the page frame tables unreferenced interval counts (UICs). In addition, there are external calls issued between processors requesting that the receiver look for ready work. These calls will be followed by a re-dispatch of the no-work wait on

the receiving processor. In short, the wait state is a combination of dispatches of the no-work wait task, clock comparator interrupts, and SIGP external calls. The IC (instruction counter) will always be 0.

After the dispatch of an enabled wait is traced, system trace is suspended on the processor to prevent overlaying meaningful trace data with entries for timer interrupts (which might make no work ready) and entries for the re-dispatches of enabled waits. System trace resumes tracing system events when the next system event occurs that is not a timer interrupt and not a re-dispatch of an enabled wait.

4. *MP Activity:*

   The communication between processors in the MP environment is traced as the external interrupts are accepted by the receiving processor. (The previous chapter, "Effects of MP on Problem Analysis," explains this communication process.)

5. *SVC D Entries:*

   SVC D is the means by which termination is invoked. RTM2 is the mechanism for normal end-of-task processing as well as for abnormal termination; RTM2 is invoked via SVC D. Consequently, SVC D for normal termination is a valid situation and is traced. You can determine whether SVC D implies normal or abnormal termination by inspecting the register 1 slot associated with the SVC D entry. If the first byte contains a X'08', RTM2 is being invoked for normal termination and this is not an error situation.

   MVS does not allow SVC routines to be invoked from code in one of the following states: cross memory mode, non-task mode, any lock held, disabled for I/O or external interruptions, or with any enabled unlocked task mode FRR established. However, SVC D issued in one of these states is a common means to enter RTM1 to invoke recovery. RTM indicators show the SVC error, and the system trace entries for the SVC and SVC error show the issuer's state, but the real problem is why SVC D was issued.

6. *Unit exception I/O interrupt on a 3705 communications controller:*

   The presence of unit exception conditions from the 3705 is a common occurrence while running VTAM. This is a normal situation and should not be considered erroneous. The host processor has issued a set of read commands to the 3705, and the channel program has been terminated before all the reads have completed because the NCP did not have enough data to satisfy each read command.

7. *GETMAIN, FREEMAIN - SVC X'A', SVC X'78':*

   For SVC X'A', inspect the register 1 slot of the associated trace entry. A value of X'80' in the high-order byte indicates GETMAIN; a value of X'00' indicates FREEMAIN. SVC X'78' uses a code in register 15 (see the *Debugging Handbook*.) If a GETMAIN is indicated, the register 1 slot of the associated re-dispatch of the SVC issuing code can be used to locate the storage allocated by the GETMAIN process.

8. *A GETMAIN for X'9B8' bytes is often seen soon after an SVC D is issued:*

   This is RTM2's request for storage for an RTM2WA, a SNAPTRC parameter list, and an SDWA for RTM2. Register 1 points to the RTM2WA. By locating the re-dispatch of RTM2 and inspecting the register 1 slot, you can locate the RTM2WA.

The Unformatted System Trace Table

The system trace table is formatted in all dumps when the system trace data is requested. However, if you need to look at the unformatted system trace table, this topic describes how to locate the trace buffers and current trace data.

The system trace table is located in the private area of the trace address space. For each processor, there is a queue of 4K buffers (initially four) in the system trace table. For each 4K buffer, there is a trace buffer vector table (TBVT) that resides in the TRACE address space and which contains buffer status and queuing information.

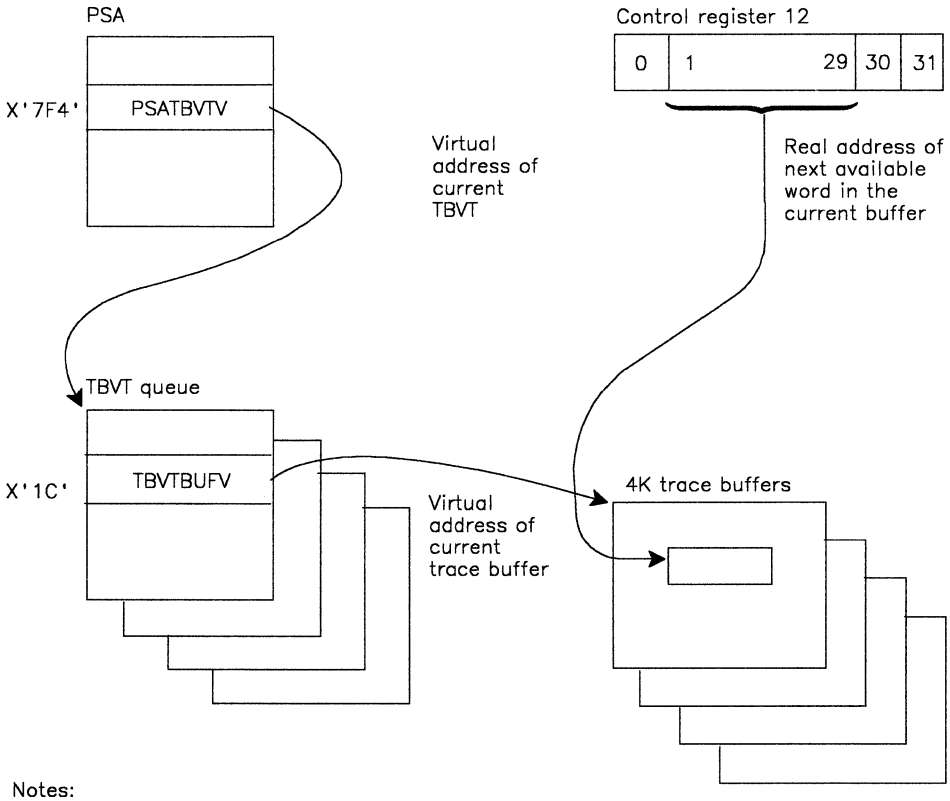
Control register 12 contains the real address of the next available word in the buffer. The bits in control register 12 have the following meaning:

Bit	Meaning
0	Bit 0 = 1 indicates branch tracing is active
1-29	Real address of the next available word in the trace buffer for a TTE
30	Bit 30 = 1 indicates address space tracing is active
31	Bit 31 = 1 indicates explicit tracing is active

Figure 2-8 shows an overview of the trace control blocks. Figure 2-9 is a list of the TTEs showing the contents of byte 0 and bytes 10 and 11 (for explicit entries). Figure 2-10 shows an example of an unformatted system trace table and contains comments on the figure pointing out various areas and trace entries.

The first byte of a TTE identifies the TTE as a branch, address space, or explicit entry. Explicit entries can be identified by the explicit entry type in field TTEXPTYP (TTE+X'A-B').

TTEs vary in content and length, therefore, use the TTE format in the *Debugging Handbook* to determine the content and length of the unformatted entries.



Notes:

1. Field PSATBVTR (PSA+X'7F0') contains the real address of the current TBVT.
2. Field TBVTCR12 (TBVT+X'8') contains the real address of the current trace buffer in control register 12 format (including the trace control bits).
3. Fields TBVTNXTR (TBVT+X'10') and TBVTNXTV (TBVT+X'14') contain the real and virtual addresses of the next TBVT. Field TBVTBWRD (TBVT+'18') contains the virtual address of the previous TBVT.

Figure 2-8. Trace Buffer Control Block Overview

For branch entries:

TTE	Byte 0	Length
BR	00 (24-bit address)	4
BR	80-FF (31-bit address)	4

For address space entries:

TTE	Byte 0	Length
SSAR	10	4
PC	21	8
PT	31	8

For explicit entries:

TTE	Byte 0	Bytes 10-11 (X'A-B')	Length Dec (Hex)
SSCH	76	0001	40 (28)
EXT	77	0003	44 (2C)
SVC	76	0005	40 (28)
PGM	78	0007	48 (30)
SPER	7B	0009	56 (38)
I/O	7B	000B	60 (3C)
DSP	79	000F	52 (34)
MCH	79	0013	52 (34)
RST	7B	0015	60 (3C)
ACR	78	0017	48 (30)
SUSP	78	0019	48 (30)
ALTR	76	001B	40 (28)
MSCH	76	0101	40 (28)
EMS	7A	0103	56 (38)
SVCR	76	0105	40 (28)
SRB	76	010F	40 (28)
HSCH	76	0201	40 (28)
SS	7C	0203	64 (40)
SSRB	77	020F	44 (2C)
CSCH	76	0301	40 (28)
CALL	78	0303	48 (30)
RSCH	74	0401	32 (20)
CLKC	79	0403	52 (34)
SVCE	7A	0F05	56 (38)
WAIT	71	0F0F	20 (14)
USRn	7n	0n7F	32 (20) + user-data

Figure 2-9. List of TTEs - Ordered by Type

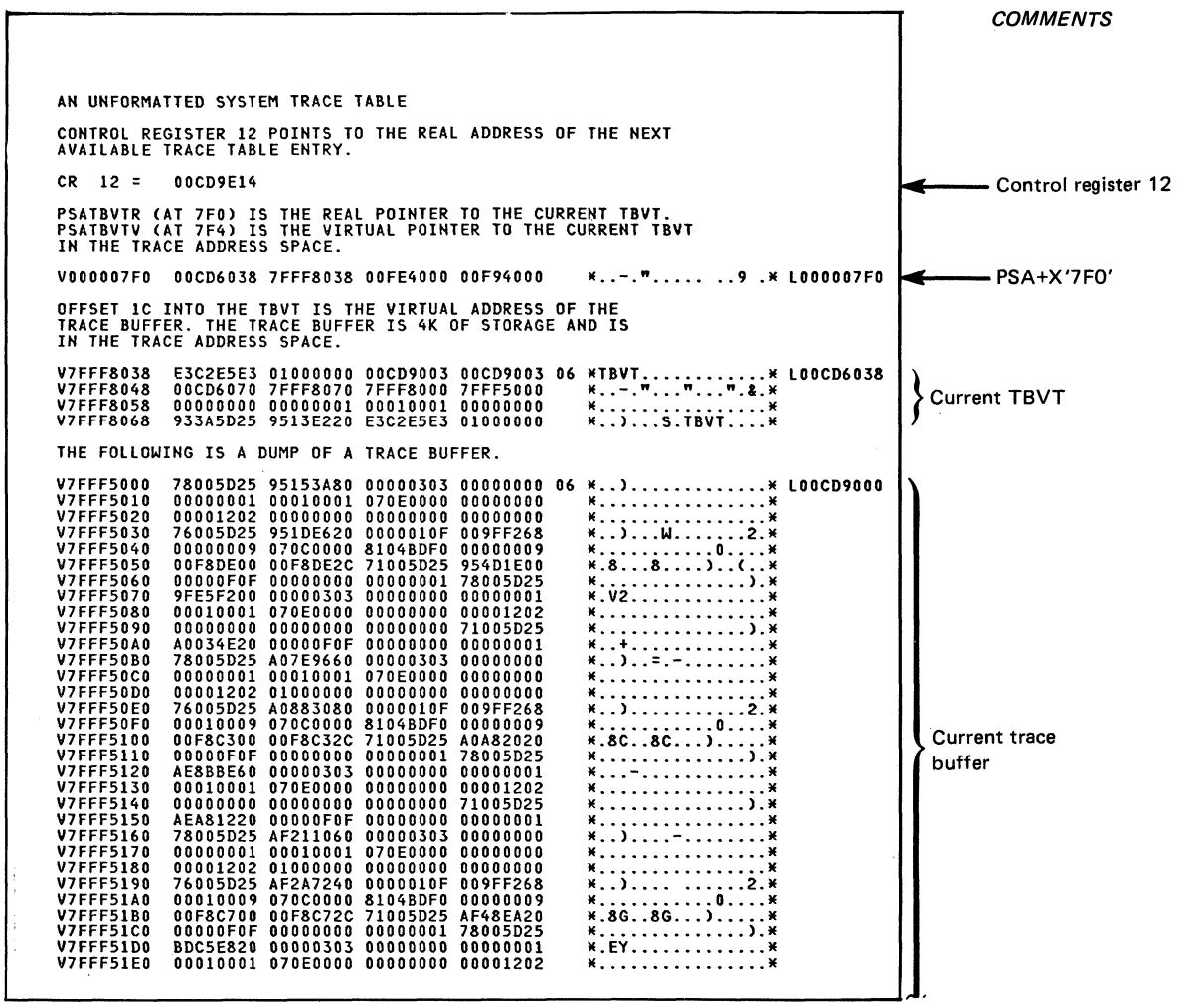
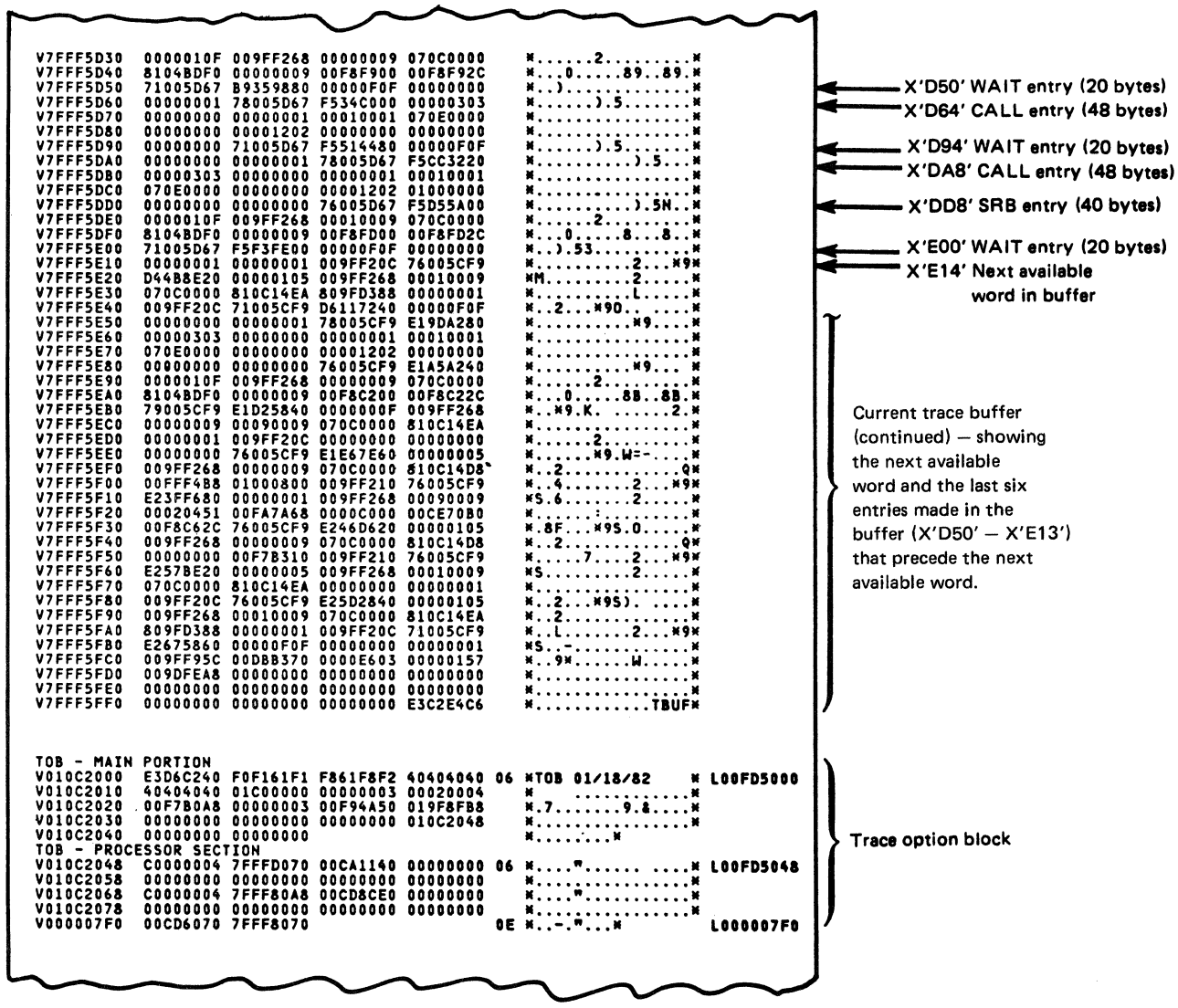


Figure 2-10 (Part 1 of 3). Example of Unformatted Trace Table




```
V7FFF5D30  0000010F 009FF268 00000009 070C0000    ¤......2.........¤
V7FFF5D40  8104BDF0 00000009 00F8F900 00F8F92C    ¤...0.....89..89.¤
V7FFF5D50  71005D67 B9359880 00000F0F 00000000    ¤..)..............¤
V7FFF5D60  00000001 78005D67 F534C000 00000303    ¤......).5.......¤
V7FFF5D70  00000000 00000001 00010001 070E0000    ¤................¤
V7FFF5D80  00000000 00001202 00000000 00000000    ¤................¤
V7FFF5D90  00000000 71005D67 F5514480 00000F0F    ¤......).5.......¤
V7FFF5DA0  00000000 00000001 78005D67 F5CC3220    ¤..........).5...¤
V7FFF5DB0  00000303 00000000 00000001 00010001    ¤................¤
V7FFF5DC0  070E0000 00000000 00001202 01000000    ¤................¤
V7FFF5DD0  00000000 00000000 76005D67 F5D55A00    ¤..........).5N..¤
V7FFF5DE0  0000010F 009FF268 00010009 070C0000    ¤......2.........¤
V7FFF5DF0  8104BDF0 00000009 00F8FD00 00F8FD2C    ¤...0.....8...8..¤
V7FFF5E00  71005D67 F5F3FE00 00000F0F 00000000    ¤..).53..........¤
V7FFF5E10  00000001 00000001 009FF20C 76005CF9    ¤...........2...¤9¤
V7FFF5E20  D44B8E20 00000105 009FF268 00010009    ¤M.........2.....¤
V7FFF5E30  070C0000 810C14EA 809FD388 00000001    ¤..........L.....¤
V7FFF5E40  009FF20C 71005CF9 D6117240 00000F0F    ¤..2...¤9O..  ....¤
V7FFF5E50  00000000 00000001 78005CF9 E19DA280    ¤..........¤9....¤
V7FFF5E60  00000303 00000000 00000001 00010001    ¤................¤
V7FFF5E70  070E0000 00000000 00001202 00000000    ¤................¤
V7FFF5E80  00000000 00000000 76005CF9 E1A5A240    ¤...........¤9... ¤
V7FFF5E90  0000010F 009FF268 00000009 070C0000    ¤......2.........¤
V7FFF5EA0  8104BDF0 00000009 00F8C200 00F8C22C    ¤...0.....8B..8B.¤
V7FFF5EB0  79005CF9 E1D25840 0000000F 009FF268    ¤..¤9.K. ......2.¤
V7FFF5EC0  00000009 00090009 070C0000 810C14EA    ¤................¤
V7FFF5ED0  00000001 009FF20C 00000000 00000000    ¤......2.........¤
V7FFF5EE0  00000000 76005CF9 E1E67E60 00000005    ¤......¤9.W=-....¤
V7FFF5EF0  009FF268 00000009 070C0000 810C14D8    ¤..2.............Q¤
V7FFF5F00  00FFF488 01000800 009FF210 76005CF9    ¤..4.......2...¤9¤
V7FFF5F10  E23FF680 00000001 009FF268 00090009    ¤S.6.......2.....¤
V7FFF5F20  00020451 00FA7A68 0000C000 00CE70B0    ¤.....:..........¤
V7FFF5F30  00F8C62C 76005CF9 E246D620 00000105    ¤.8F...¤9S.O.....¤
V7FFF5F40  009FF268 00000009 070C0000 810C14D8    ¤..2.............Q¤
V7FFF5F50  00000000 00F7B310 009FF210 76005CF9    ¤.....7....2...¤9¤
V7FFF5F60  E257BE20 00000005 009FF268 00010009    ¤S.........2.....¤
V7FFF5F70  070C0000 810C14EA 00000000 00000001    ¤................¤
V7FFF5F80  009FF20C 76005CF9 E25D2840 00000105    ¤..2...¤9S). ....¤
V7FFF5F90  009FF268 00010009 070C0000 810C14EA    ¤..2.............¤
V7FFF5FA0  809FD388 00000001 009FF20C 71005CF9    ¤..L.......2...¤9¤
V7FFF5FB0  E2675860 00000F0F 00000000 00000001    ¤S..-............¤
V7FFF5FC0  009FF95C 00DBB370 0000E603 00000157    ¤..9¤......W.....¤
V7FFF5FD0  009DFEA8 00000000 00000000 00000000    ¤................¤
V7FFF5FE0  00000000 00000000 00000000 00000000    ¤................¤
V7FFF5FF0  00000000 00000000 00000000 E3C2E4C6    ¤............TBUF¤


TOB - MAIN PORTION
V010C2000  E3D6C240 F0F161F1 F861F8F2 40404040 06 ¤TOB 01/18/82    ¤ L00FD5000
V010C2010  40404040 01C00000 00000003 00020004    ¤   .............¤
V010C2020  00F7B0A8 00000003 00F94A50 019F8FB8    ¤.7.......9.¢....¤
V010C2030  00000000 00000000 00000000 010C2048    ¤................¤
V010C2040  00000000 00000000                      ¤........¤
TOB - PROCESSOR SECTION
V010C2048  C0000004 7FFFD070 00CA1140 00000000 06 ¤....".....  ....¤ L00FD5048
V010C2058  00000000 00000000 00000000 00000000    ¤................¤
V010C2068  C0000004 7FFF80A8 00CD8CE0 00000000    ¤....".....¤.....¤
V010C2078  00000000 00000000 00000000 00000000    ¤................¤
V000007F0  00CD6070 7FFF8070                   0E ¤..-.".../¤         L000007F0
```

Figure 2-10 (Part 2 of 3). Example of Unformatted Trace Table

```
THIS IS THE TBVT QUEUE FOR PROCESSOR 0.

**** TBVT ****
V7FFFD070  E3C2E5E3 01000000 00CA6003 00CA6FBF 06 *TBVT.....-...?.* L00CB4070
V7FFFD080  00CB40A8 7FFFD0A8 7FFFD038 7FFFB000    *.. .".....".....*
V7FFFD090  00000000 00000001 00010001 00000000    *................*
V7FFFD0A0  933A5F35 2CDABA60                      *..¬....-*
**** TBVT ****
V7FFFD0A8  E3C2E5E3 01000000 00CAF003 00CAFFB7 06 *TBVT.....0.....* L00CB40A8
V7FFFD0B8  00CB4000 7FFFD000 7FFFD070 7FFFC000    *.. .".....".....*
V7FFFD0C8  00000000 00000008 00080008 00000000    *................*
V7FFFD0D8  933A5F36 3FE5BE40                      *..¬..V. *
**** TBVT ****
V7FFFD000  E3C2E5E3 01000000 00CA1003 00CA1003 06 *TBVT...........* L00CB4000
V7FFFD010  00CB4038 7FFFD038 7FFFD0A8 7FFF9000    *.. .".....".....*
V7FFFD020  00000000 00000008 00080008 00000000    *................*
V7FFFD030  933A5F7C E206EA40                      *..¬aS.. *
**** TBVT ****
V7FFFD038  E3C2E5E3 01000000 00CA4003 00CA4FC7 06 *TBVT..... ...|G* L00CB4038
V7FFFD048  00CB4070 7FFFD070 7FFFD000 7FFFA000    *.. .".....".....*
V7FFFD058  00000000 00000001 00010001 00000000    *................*
V7FFFD068  933A5F33 CA69D060                      *..¬....-*

THIS IS THE TBVT QUEUE FOR PROCESSOR 2.

**** TBVT ****
V7FFF80A8  E3C2E5E3 01000000 00CD7003 00CD7FCB 06 *TBVT.........".* L00CD60A8
V7FFF80B8  00CD6000 7FFF8000 7FFF8070 7FFF7000    *..-.".....".....*
V7FFF80C8  00000000 00000008 00080008 009FF570    *..............5.*
V7FFF80D8  933A5F35 1930A080                      *..¬.....*
**** TBVT ****
V7FFF8000  E3C2E5E3 01000000 00CDA003 00CDAFBF 06 *TBVT...........* L00CD6000
V7FFF8010  00CD6038 7FFF8038 7FFF80A8 7FFF4000    *..-.".....".. .*
V7FFF8020  00000000 00000008 00080008 009FF570    *..............5.*
V7FFF8030  933A5F35 CC5EAC20                      *..¬..;..*
**** TBVT ****
V7FFF8038  E3C2E5E3 01000000 00CD9003 00CD9FCF 06 *TBVT...........* L00CD6038
V7FFF8048  00CD6070 7FFF8070 7FFF8000 7FFF5000    *..-.".....".....&.*
V7FFF8058  00000000 00000008 00080008 009FF570    *..............5.*
V7FFF8068  933A5F36 92B63620                      *..¬.....*
**** TBVT ****
V7FFF8070  E3C2E5E3 01000000 00CD8003 00CD8003 06 *TBVT...........* L00CD6070
V7FFF8080  00CD60A8 7FFF80A8 7FFF8038 7FFF6000    *..-.".....".....*
V7FFF8090  00000000 00000006 00060006 009FF598    *..............5.*
V7FFF80A0  933A5F37 AA596E60                      *..¬...>-*
```

TBVT queue for processor 0

TBVT queue for processor 2

Figure 2-10 (Part 3 of 3). Example of Unformatted Trace Table

Master Trace

Master trace is useful for debugging problems when you need to know the content of recently issued messages. Master trace maintains a wraparound table of the messages that are routed to the hardcopy log. When a message becomes eligible for the hardcopy log, it is entered into the master trace table by the communications task queuing routines. The trace table resides in pageable virtual storage in the master scheduler's private area.

The size of the master trace table is specified by the MT operand of the TRACE command and by the MT statement in the SCHEDxx member of SYS1.PARMLIB. If a size is not specified, the default size is 24K bytes. The master trace table is created during master scheduler initialization. After system initialization, master trace may be activated and deactivated by using the TRACE command.

The master trace table is included in SVC dumps as a part of the SDATA=TRT option. The default size of 24K bytes accommodates approximately 336 messages (with an average length of 40 characters).

To locate the master trace table: field CVTMSER (at CVT+X'94') points to IEEBASEA (master scheduler resident data area) and field BAMTTBL at offset X'8C' in the IEEBASEA points to the master trace table.

When submitting an APAR, the SVC dump may be submitted for the hardcopy log if the master trace table contains the required messages. For example:

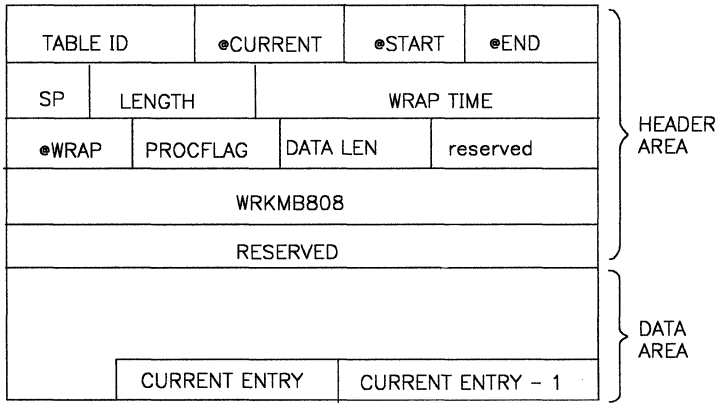
- The master trace table has wrapped at 9:00.
- A message related to a problem was issued at 9:20.
- An SVC dump is taken at 9:30.

In the example, the required messages are in the dump because the problem occurred between the time that the master trace table was wrapped and the dump was taken.

Master Trace Table

Master trace data is maintained in a wraparound table that is anchored in the master scheduler resident data area. The format of the master trace table and entries is described in the *Debugging Handbook*.

The table contains the following header and data areas:



Where:

TABLE ID - A fullword field with a constant of 'MTT' to identify the beginning of the master trace table.

@CURRENT - Address of the first byte of the current (most recently stored) entry.

@START - Address of the first byte of the data area.

@END - Address of the first byte beyond the end of the data area.

SP - Subpool in which this table resides.

LENGTH - Length of the header and data areas combined - the size specified on the TRACE command.

WRAP TIME - Either the time that the table was initialized or the time at which the last table wrap occurred. The form is:

xx - where IT indicates initialize time and WT wrap time
HH - hours
MM - minutes
SS - seconds
T - tenths of a second

@WRAP - Address of the first byte of the last entry stored before the most recent table wrap. It is initialized to zero and remains so until the first table wrap.

PROCFLAG - Processing flags used by IEEMB808.

DATA LEN - Fullword field containing the size of the data area of the table.

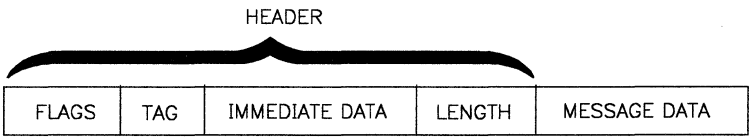
reserved - Reserved fullword.

WRKMB808 - Sixteen word work area for IEEMB808 and IEEMB809 -- serialized by CMS and local locks.

RESERVED - Reserved four word area.

Master Trace Table Entry

The master trace table entries (located in the data area of the master trace table) contain the following fields:



Where:

FLAGS - Halfword containing the flags set by the caller in the parameter list passed to IEETRACE.

TAG - Halfword value indicating the identity of the caller. Values are defined in macro IEZMTPRM, which maps the parameter list.

IMMEDIATE DATA - Fullword containing 32 bits that are defined by the caller. This area is stored in the table without validity checking by the trace service routine.

LENGTH - Halfword containing the length of the message data.

MESSAGE DATA - Variable length field containing message data provided by the caller. The maximum size is the length of the data area less 10 bytes, or 65,535 bytes, whichever is less.

The table entries are of variable length. If the length of the data is specified as zero by the caller, only the 10 byte header is entered in the table and used to store immediate data. The significance of the immediate data is defined by the caller.

Entries are placed into the table from back to front in the data area. Thus the current entry immediately precedes the entry stored on the previous call to IEETRACE.

Formatting the Master Trace Table

You can use the IPCS subcommand VERBEXIT MTRACE to format the master trace table. Also, you can use the print dump MTRACE control statement when you print the dump. (See also the topic "How to Print Dumps" on page 2-99.)

The entries in the master trace table are listed in first-in, first-out (FIFO) order, which resembles a hardcopy log. (Note that the messages might not be in chronological order because the messages might not have been put into the master trace table in the order the messages were issued.)

The title *** MASTER TRACE TABLE *** is followed by the master trace table entries. For each trace entry, the halfword tag, the fullword immediate data, and the message data are listed. If an error occurs during processing, a line following the title will indicate the error.

The formatted output obtained by the MTRACE control statement is a suitable replacement for the hardcopy log, provided that the required messages appear in the dump (as described earlier in this topic).

See the *Debugging Handbook* for an example of the formatted master trace table and *Service Aids* for a description of the MTRACE control statement.

The Message Processing Facility Table (MPFT)

By using the SET MPF (message processing facility) system command, you can suppress messages from being displayed on the operator's console.

The message processing facility builds the message processing facility table (MPFT, mapped by macro IEEZB809), which contains an entry for each message ID to be suppressed. The table includes the following:

- Table header:
  - Subpool number (where the table resides)
  - Size of the table (in number of bytes)
  - Number of entries in the table following the header
  - PARMLIB suffix from which the table was built
  - Address of the first entry in the table
  - Length of each entry
- Table entry (one 10-byte entry for each message to be suppressed):
  - Message ID
  - Length of the message
  - Flag byte

The MPFT is located in the common service area (CSA) and is pointed to by field UCMFMPFP in the UCM (from the fixed extension base).

Miscellaneous Debugging Hints

This chapter is a collection of miscellaneous debugging hints to aid the problem solver in specific situations not covered elsewhere in this book. It includes the following topics:

- Record of SVC Table Updates
- Alternate CPU Recovery Problem Analysis
- Supervisor Analysis Router
- Pattern Recognition
- Debugging Machine Checks
- Debugging Problem Program ABEND Dumps
- Debugging From Summary SVC Dumps
- Dump Analysis and Elimination (DAE)

Record of SVC Table Updates

The system maintains an SVC update recording table in parallel with the SVC table. The SVC update service (IEAVESTU) uses the SVC update recording table to track all changes that are made in the SVC table. Note, however, that at any given time during an IPL, an entry in the SVC update recording table contains information about only the ***most recent*** change to the corresponding SVC table entry. If an SVC routine that was installed using the SVC update service causes a problem, you can use the SVC update recording table to determine when and how the SVC routine was added to the system or modified.

The address of the SVC update recording table is in the secondary CVT field named SCVTSVCR. The SVC update recording table has 256 entries, each initialized to zero. When either the SVCUPDTE macro or the IEASVCxx member of SYS1.PARMLIB modifies an SVC table entry, the corresponding SVC update recording table entry is modified with the following data:

- Old SVC table entry
- Date of update (CVTDATE in format YYDDD)
- Count of updates to this entry
- Return address of the caller of IEAVESTU
- The 2-character suffix if the update was from IEASVCxx
- Entry point address specified in the new SVC table entry

An entry in the SVC update recording table is mapped by the IHASVC macro assembled with the UPDATE=YES option. (See the SVC update recording table entry, which is shown in the SVCTABLE data area in the *Debugging Handbook*.)

Alternate CPU Recovery (ACR) Problem Analysis

Alternate CPU recovery (ACR) is the process by which MVS dynamically adjusts to the unexpected failure of a processor in a multiprocessing (MP) configuration. ACR is initiated by the failing processor. If the failing processor's hardware detects the failure, it issues a malfunction alert (MFA) external signal to another processor. If the failing processor generates the severe machine check interrupt (recursive or invalid logout) type, the machine check interrupt handler will initiate ACR via the SIGP instruction with the emergency signal (EMS) order code, which generates an external interrupt on the receiving processor.

When a running processor detects that a failing processor is requesting ACR, it places X'FF' in the CSDACR byte (CSD+X'16') in the CSD control block. The byte will be restored to X'00' after ACR is complete.

You can use the IPCS subcommand STATUS WORKSHEET to determine if ACR is in progress.

ACR works in three phases: pre-processing, intermediate, and post-processing phase. Pre-processing is the initialization phase: the running processor copies the PSA and normal functional recovery routine (FRR) stacks of both processors and places them in the area pointed to from their respective LCCA's WSACACR pointer. The WSACACR pointer is located at X'10' beyond the area pointed to by LCCACPUS. Additionally, LCCAs are marked so that in both processor's LCCAs, LCCADCPU points to the LCCA of the failing processor and LCCARCPU points to the LCCA of the running processor. By means of the CSDCPUAL field in the CSD (CSD+X'8'), you can determine which processor has failed and which are still running. A system trace table entry is created to indicate that ACR has begun and to identify the failing processor.

Note that in a storage dump, the PSA of the failed processor is the same as it was when the processor decided that ACR should be initiated. The normal FRR stack, pointers to other FRR stacks, locks, PSASUPER bits etc. all reflect the state of the processor at the time it failed. This will be useful for solving problems in the recovery initiated for the process on the failed processor.

The ACR intermediate phase gets control from the MVS dispatcher, or lock manager global spin lock routine. In this phase, ACR switches from the process on one logical processor to the process on the other logical processor. This switching continues until the RTM1 recovery (routing to FRRs) completes on behalf of the process on the failed processor. At this point, the ACR post-processing phase is entered.

ACR post-processing notifies SRM of the loss of the processor, invokes the MSSFCALL SVC (IEAVMSF) to cause the service processor to physically vary the processor offline, issues message IEA858E 'ACR - COMPLETE', and resets the CSDACR flag to X'00'.

Supervisor Analysis Router

The supervisor analysis router is called by:

- The restart FLIH when the operator requests system diagnostic and repair actions.
- The recovery termination manager (RTM) as a result of a SLIH mode entry to RTM by one of the interruption handlers.

The supervisor router routine (IEAVESAR) aids in the recovery of a failing processor by refreshing critical system control block pointers and fields (such as PSALCCAV and PSASCWA) that might have been overlaid.

IEAVESAR also calls component analysis routines that continue the refresh/repair process (such as the lock repair functions, IEAVELKR and IEAVESLR). The repair actions taken by the supervisor router (and routines called by the router) are recorded in the SDWAVRA of a software SYS1.LOGREC record. The LOGREC record is produced by IEAVESAR via the RECORD function and is not associated with recovery or dump processing.

These LOGREC records can be identified by the following header printed by EREP:

```
JOBNAME                          IEAVESAR
ABENDING PROGRAM NAME            N/A
NAME OF MODULE INVOLVED          NUCLEUS
NAME OF CSECT INVOLVED           IEAVESAR
FUNCTIONAL RECOVERY ROUTINE      NONE
```

The entries in the SDWAVRA of the LOGREC record are formatted in the key-length-data format and include:

- The name of the control block containing the repaired field
- The name of the repaired field
- The content of the field prior to the repair.

If a system control block pointer or field cannot be corrected, IEAVESAR sets the current processor in a disabled wait with wait state code 083.

Pattern Recognition

When analyzing a dump you should always be aware of the possibility of a storage overlay. System incidents in MVS are often caused by storage overlays that destroy data, control blocks, or executable code. The results of such an overlay vary. For example:

- The system detects the problem and issues an abnormal completion code, yet the error can be isolated to an address space.
- Referencing the data or instructions can cause an immediate error such as a specification or op-code exception.
- The bad data can be used to reference a second location, which then causes an evident error.

When you recognize that the contents of a storage location are invalid and subsequently recognize the bit pattern as a certain control block or piece of data, you generally can identify the erroneous process/component and start a detailed analysis. This section discusses pattern recognition and potential causes of storage overlays, and points out common patterns that aid the debugger.

Once you recognize an overlay, analyze the bit pattern. If you do not recognize the pattern at all, try to determine the extent of the damaged area. Look at the data on both sides of the obviously bad areas. See if the length is familiar; that is, can you relate the length to a known control block length, data size, MVC length, etc.? If so, check various offsets to determine their contents and, if you recognize some, try to determine the exact control block/data. Even if you do not recognize the pattern, take one more step. Can you determine the offset from

some base (X) that would have to be used in order to create the bit pattern? If so, the fact that there is a certain bit pattern at a certain offset (Y) can be helpful. For example, a BALR register value (X'40D21C58') at an offset X'C' can indicate that a program is using this storage for a register save area (perhaps caused by a bad register 13). Another field in the same overlaid area might trigger recognition.

For routines that are executing in 24-bit addressing mode, look at the overlaid area and scan for familiar addresses such as device addresses, UCB addresses, and BAL/BALR register values (fullword with high-order byte containing some "1" bits). If you find any of these, try to determine what components or modules are involved or what control blocks contain these addresses.

Repetition of a pattern can indicate a bad process. If you can recognize the bad data you might be able to relate that data to the component or module that is causing the error. This provides a starting point for further analysis.

Common Bad Addresses

Common bad addresses are:

- X'C0000', X'040C0000', or X'070C0000', and one of these addresses plus some offset. These are generally the result of some code using 0 as the base register for a control block and subsequently loading a pointer from 0 plus an offset, thereby picking up the first half of a PSW in the PSA.

  Look for storage overlays in code pointed to by an old PSW. These overlays result when 0 plus an offset cause the second half (IC) of a PSW to be used as a pointer.

- X'C00', X'D00', X'D20', X'D28', X'D40', and other pointers to fields in the normal FRR stack. Routines often lose the contents of a register during a SETFRR macro expansion and illegally use the address of the 24-byte work area returned from the expansion.

- Register save areas. Storage might be overlaid by code doing an STM (Store Multiple) instruction with a bad register save area address. In this case, the registers saved are often useful in determining the component or module at fault.

Debugging Machine Checks

The machine check interruption is the hardware's method of informing the MVS control program that it has detected a hardware malfunction. Machine checks vary considerably in their impact on software processing. Some machine checks notify software that the processor detected and corrected a hardware problem that required no software recovery action (software calls these errors soft errors). Hard errors are hardware problems detected by a processor but that require software-initiated action for damage repair. Hard errors also require software recovery to verify the integrity of the process that experienced the failure. Obviously, if there are software problems after a machine check, it is more likely that the machine check was a hard error. It is important to get a feeling for which software components are affected by particular hardware failures.

The machine check interrupt code (MCIC), located in the PSA (at X'E8'), describes the error causing the interrupt. (Refer to *Principles of Operation* for a complete description of the MCIC.) The following discussion shows how to find MCICs and how to interpret them for subsequent software processing. Machine checks can be found in a LOGREC buffer (LRB), the SYS1.LOGREC data set, or in the storage area used as a buffer prior to writing records to SYS1.LOGREC (see the discussion of SYS1.LOGREC analysis in the "Recovery Work Areas" chapter earlier in this section). Also, a pointer to the LRB that describes the last machine check that occurred on a processor can be found in that processor's PCCA at PCCALRBV (PCCA+X'A0'). The LRB contains the machine check interrupt code (MCIC), except when:

- The machine check old PSW is zero. The MCIC is also zero. The LRBMTCKS bit (field LRBTERM at LRB+X'20') is turned on by software.
- MCIC is zero and the machine check old PSW is non-zero. The LRBMTINV bit (field LRBTERM at LRB+X'20') is turned on by software.

The MCIC is the principal driver of software processing after a machine check. It must be examined to determine the actions that MVS should take. The MCIC contains bits describing the conditions that caused the interrupt. Note that more than one failing condition can be described by a machine check at one time. Software performs repair processing for each condition found; software recovery processing is initiated if any hard error conditions are found (except in the cases described on the following pages).

Because hard errors require FRR and ESTAE processing, identifying a hard error is important. Important MCIC bits follow with a description of their hardware significance and impact on software.

Bit 0 (System damage) - The processor is still useable, but damage has occurred which cannot be isolated to one or more of the less severe machine checks. Software recovery routines (FRRs) are entered for this hard error.

Bit 1 (Instruction processing damage) - The processor is still useable, but an instruction has failed to operate as intended. Software recovery is initiated for this hard error, unless the backed-up bit is on with storage error or key error uncorrected on refreshable storage (see Bit 16 description).

Bit 2 (System recovery) - The processor detected and corrected a potential hardware problem. The interrupted process is completely restored by software for this soft error; no repair is performed and no recovery routines are entered.

Bit 4 (Timing facility damage) - Damage has occurred to the CPU timer, clock comparator, or time-of-day clock. The particular clock facility that is damaged is described by MCIC bits 46 and 47. A first failure to a facility results in an attempt to reuse it. Subsequent failures result in taking the facility offline (described in the PCCA fields PCCATODE, PCCACCE, or PCCAINTE). If no clock of a particular type remains in the system, any task which requests timing using that type of clock is sent through software recovery. This is treated as a soft error for the process current on the processor at the time of the interrupt.

Bit 6 (Vector Facility failure bit) - The Vector Facility has failed. Software terminates the work in progress, disconnects Vector Facility processing, takes the Vector Facility offline, and issues message IGF970E to the operator.

Bit 7 (Degradation) - The system has detected that elements of the high-speed buffer (cache) or translation look-aside buffer have had bit (parity) errors. The bad elements are automatically reconfigured out of the buffer. The system might perform at a reduced rate because of increased storage access time (cache element deletion) or increased time to translate virtual addresses (because of translation look-aside buffer element deletion). However, because no damage has been done to any software process or data, this soft error is merely recorded in SYS1.LOGREC. The system state at the time of the error is re-established, ignoring the occurrence of the buffer bit error. It is treated as a soft error and no software recovery is initiated.

Bit 8 (Warning) - Damage is imminent; there is a cooling loss or a power drop, etc.

Bit 9 (CRW pending) - The channel subsystem has indicated that CRWs (channel report words) are pending. Software retrieves the CRWs and processes them in order.

Bit 10 (Service processor damage) - The service processor or MSSF has failed. Software issues a message indicating that there is a problem in the processor controller.

Bit 11 (Channel subsystem damage) - The channel subsystem has failed. No further communication with the channel subsystem can occur. The system is stopped with a disabled wait state of A19.

Bit 13 (Vector Facility source bit) - If the instruction processing damage bit (bit 1) is also on, bit 13 indicates that the machine check might be associated with the Vector Facility. Software attempts to save the vector environment and restart the work in progress (if not successful, terminates the work), and then updates the threshold value for vector source machine checks. When the threshold value is exceeded, software takes the Vector Facility offline and issues message IGF970E to the operator. Bit 13 is not meaningful when bit 6 (Vector Facility failure) is also on.

Bit 16 (Storage error uncorrected) - There is a block in storage with a double bit error that is located at the real, prefixed address stored in PSA location X'F8'. If the frame's page is refreshable, that is, unchanged, pageable, and in the current address space, it is marked invalid so a future reference will cause a fresh copy to be paged into a new frame. (*Note:* More than one error can occur before the page goes offline.) In all cases, an attempt is made to take the damaged frame offline (unless the frame is in the nucleus). When a storage error uncorrected condition occurs in conjunction with a system recovery error, it is treated as a soft error and no recovery routines are entered. If the storage error occurs in conjunction with instruction processing damage when the backed-up bit (bit 14) and storage logical validity bit (bit 31) are on, and the frame's page is refreshable, the error is treated as soft and no recovery routines are entered.

Any other occurrences of storage error uncorrected are treated as hard errors and software recovery is initiated for the error.

Bit 17 (Storage error corrected) - A single-bit storage error was detected and successfully corrected by hardware. Software treats this error as a soft error. This error sometimes appears in conjunction with system recovery (bit 2). For a double bit storage error, see bits 16 and 19.

Bit 18 (Storage key error uncorrected) - Hardware has detected a bit error in a storage key. Software attempts to reset the storage key to its original value. If the key is successfully reset, and the storage key error occurs in conjunction with instruction processing damage when the backed-up bit (bit 14) and the storage logical validity bit (bit 31) are on, the error is treated as soft and no recovery routines are entered. When the storage key error occurs in conjunction with a system recovery error, it is also treated as a soft error and no recovery routines are entered. Change bits are set to one in case the frames have been altered. Any other occurrences of storage key error are treated as hard errors and software recovery is initiated for the error.

Bit 19 (Double bit storage error) - If the storage error corrected bit (bit 17) is also on, bit 19 indicates that a double bit storage error was detected and successfully corrected by hardware. If the page containing the data can be paged, software obtains a new frame, moves the data from the frame that has the indicated double bit error correction into the new frame, and then marks the frame that had the double bit error offline. If the page containing the data cannot be paged, software marks the associated frame as intercepted to go offline, which causes the frame to be taken offline when the page is freed.

In addition to these error description bits there are other MCIC fields that describe the time-of-occurrence of the machine check interrupt, or the validity of the registers, PSW, and other data logged out during the machine check interruption process.

The two time-of-occurrence bits are bits 14 and 15. The backed-up bit (bit 14), when set to 1, indicates that the machine check occurred before actual damage occurred. Note that in this case, the machine check old PSW points to the instruction, not to the next instruction. The delayed bit (bit 15) is set to 1 when the processor has been disabled for one or more of the interrupt conditions described in the MCIC. The processor had been processing after damage was detected.

Validity bits describe the validity of the associated field logged out during the machine check interrupt. If a validity bit is 0, the associated data logged out is incorrect. Validity bits are:

- Bit 20 (PSW EMWP mask validity)
- Bit 21 (masks and key validity)
- Bit 22 (program mask and condition code validity)
- Bit 23 (instruction address of machine check old PSW validity)
- Bit 24 (failing storage address validity)
- Bit 27 (floating point register validity)
- Bit 28 (general purpose register validity)
- Bit 29 (control register validity)
- Bit 30 (processor model-dependent logout validity)
- Bit 31 (storage logical validity)
- Bit 46 (CPU-timer validity)
- Bit 47 (clock comparator validity)

Debugging Problem Program Abend Dumps

The following steps may provide some initial assistance in this debugging process:

1. Locate the RTM2 work area (RTM2WA), which is pointed to by the TCBRTWA field in the TCB and the ESART2WA field in the abend SVRB. It provides a summary of the abend as follows:

Name	Offset	Explanation
RTM2CC	1D	Abend completion code.
RTM2CRC	3E0	Reason code associated with completion code.
RTM2ABNM	8C	Abending program name. This is the name of a load module or an external entry point (ALIAS) in the load module.
RTM2ABEP	94	Abending program address (the beginning of the load module or an ALIAS in the load module).
RTM2EREG	3C	Registers at time of error.
RTM2APSW	7C	EC PSW at time of error.
RTM2ILC1	85	Instruction length code for PSW at time of error.
RTM2ERAS	36C	Error ASID.
RTM2ERRA	B4	Error type.

   Notes:

   a. *The RTM2ABNM and RTM2ABEP fields do not contain information about the abending program if an SVC has abended.*

   b. *In a recursive abend (an abend occurring while the original abend is being processed by an ESTAE or other recovery routine), more than one RTM2WA may be created, and the RTM2PREV or RTM2PRWA field points to other RTM2WAs associated with the problem. The system diagnostic work area (SDWA) is pointed to by the RTM2RTCA field during recovery routine processing, and has register contents at time of error stored in the SDWAGRSV field. These register contents may differ from those in the RTM2WA after a recursive abend.*

2. To find the abend code and its explanation, look at the completion code at the top of the abend dump. A user completion code is printed as a 4-digit decimal number and a system completion code is printed as a 3-digit hexadecimal number.

   If the user code is non-zero, a user program has specified the completion code in an abend macro instruction. Looking up the name of the abending program in the RTM2WA, and investigating why the program would issue this completion code, should lead directly to the cause of the error in the user program.

   Usually the *system* code is non-zero. This indicates that a system routine issued the abend but a problem program might indirectly have caused the abnormal termination. For example, a problem program might have branched to an invalid storage address, specified an invalid parameter on a macro instruction, or requested too much storage space.

Often the explanation of the system code gives enough information to determine the cause of the termination. The explanations of system completion codes, along with a short description of the action for the programmer to take to correct the error, are contained in *System Codes*. A summary of system codes is in the *Debugging Handbook* Volume 1.

3. To find the name of the abending program look in the RTM2 work area. System routines usually start with the letters A or I; and module prefixes for system routines are listed in the *Debugging Handbook* Volume 1.

   Note: If the RTM2 work area is not available, or if the name of the abending program is not given in the RTM2 work area, the routine name can be obtained from the contents directory entry (CDE) queued to the program request block (PRB). If the ABEND dump was taken to a data set (or to SYSOUT) specified with a SYSABEND, SYSMDUMP, or SYSUDUMP DD statement, the last two RBs are SVRBs for the SNAP and SYNCH SVCs used to take the dump. The SVC numbers can be checked by obtaining the hexadecimal SVC number from the interruption code of the WC-L-IC field in the RB. The *Debugging Handbook* contains a list of SVC numbers. The SNAP SVC is hexadecimal '33', and the SYNCH SVC is hexadecimal '0C'. The RB for the program that caused the abend is immediately before these two RBs.

   CSECTs within load modules in the private area of an address space can be located using a linkedit map produced by the AMBLIST service aid. CSECTs in load modules in the nucleus, FLPA, or PLPA can be located using a nucleus or link pack area map, also produced by AMBLIST.

4. To find the instruction that caused a program interrupt (program check) completion code (0Cx) in a problem program, examine the PSW at the time of error. It is at the top of the abend dump, in the RTM2 work area, and in the RB for the program that caused the abend. The instruction address field in the PSW contains the address of the next instruction to be executed.

   The length of the abend-causing instruction is printed following the instruction length code's title 'ILC' at the top of some abend dumps. It is also located in the RTM2ILC1 field (see the RTM2 work area), and is formatted in the third and fourth digits (00xx0000) of the WC-L-IC field in the PRB. The address of the instruction that caused the termination can be found by subtracting the instruction length from the address in the PSW.

   Subtract the program address found in the RTM2WA (and in the last PRB) from the instruction address. The resulting offset can be used to find the matching instruction in the abending program's assembler listing for this CSECT.

5. To find the cause of a program interrupt, check the explanation of the system completion code and the instruction that caused the interrupt. Also check the registers from the time of error which are saved in the RTM2WA and in the SVRB following the RB for the program that caused the abend. The formatted save area trace can be used to check the input to the failing CSECT.

6. To find the cause of an abend code from an SVC or from a system I/O routine, check the explanation of the system completion code, then find the last instruction executed in the failing program and examine the related SVC and I/O entries in the trace table or GTF trace records.

   The last PRB in the formatted RBs has a PSW field containing the address of the instruction following the instruction that issued the SVC. For I/O requests, check the entry point address ('EPA') field in the last PRB. The formatted save area trace gives the address of the I/O routine branched to, and the return address in that save area is the address of the last instruction executed in the failing program.

   The system trace information can be checked for SVC entries that match the formatted SVRBs, or for I/O entries issued from addresses in the failing program. The trace information is formatted in the dump if the installation has specified it as a dump option. The system trace table is frequently overlaid with entries for other system activity by the time the dump is produced.

   If the dump contains system trace records, begin at the most recent entry and proceed backwards to locate the most recent SVC entry indicating the problem state. From this entry, proceed forward in the table. Examine each entry for an error that could have terminated the SVC or I/O system routine. The format of system trace table entries is described in the *Debugging Handbook* under the heading 'TTE Trace Table Entry.' The format of GTF trace records is also described in the *Debugging Handbook*.

7. In a cross memory environment, many services are requested by use of the Program Call (PC) instruction rather than by SVCs or SRBs. When an abend is issued by the PC routine, it can be confusing trying to identify the caller and exactly where the PC instruction was issued. This is because the RB old PSW contains the instruction address of the PC routine issuing the abend and the abend SVRB contains the registers of the PC routine.

   To determine if a program is in cross memory mode, examine the SASID and PASID fields in the XSB control block. If they are not equal, the program is executing in cross memory mode. To locate the XSB:

   - In a formatted dump, the XSB is printed following the RB with which it is associated.
   - In storage, field RBXSB (RB-X'20') points to the XSB.

   In cross memory mode, you can determine the caller of a PC routine by examining the PCLINK stack. To locate the PCLINK stack element (STKE):

   - In a formatted dump, the STKEs are printed following all of the RBs. If there are more than one STKE, the pointer to the one you want is contained in field XSBSTKE (XSB+X'18') of the XSB associated with your RB.
   - In storage, field RBXSB (RB-X'20') points to the XSB and field XSBSEL (XSB+X'1C') points to the current STKE.

Important fields in the STKE are:

- STKERET (STKE+X'18') - contains the return address of the caller of the PCLINK service.
- STKEPR15 (STKE+X'1C') - contains parameter register 15 passed to the PC routine.
- STKEPRM0 (STKE+X'20') - contains parameter register 0 passed to the PC routine.
- STKEPRM1 (STKE+X'24') - contains parameter register 1 passed to the PC routine.
- STKESA (STKE+X'14') - contains the address of the previous save area passed by the caller of the PC service.

Debugging from Summary SVC Dumps

The SUMDUMP option on the SDUMP macro causes SVC dump to produce a summary dump, which is formatted by the IPCS SUMDUMP verb exit and by the PRDMP (print dump) SUMDUMP control statement. It is strongly recommended that the SUMDUMP output be reviewed prior to investigating the usual portions of the dump. The SUMDUMP option provides different output for SVC and branch entries. For example, branch entries generally dump PSA, LCCA, and PCCA control blocks, and SVC entries generally dump RTM2WA control blocks. Each output type is indicated by the header "----tttt---- RECORD ID X'nnnn'," where *tttt* is the title for the type of SUMDUMP output, and *nnnn* is the hexadecimal record identifier assigned to the type. The record id values are described in the table below. They are also described by the IHASMDLR mapping macro in the *Debugging Handbook*.

SUMDUMP Output For SVC-Entry SDUMP

The following table summarizes the SUMDUMP output types for an SVC entry to SDUMP:

SVC-ENTRY TABLE

Record ID Dec	Hex	Title	Mapping Macro	Fields Used to Dump PSW or Register Areas
46	2E	SUMLIST RANGE	-	-
48	30	REGISTER AREA	-	-
53	35	NORMAL DATA END	-	-
57	39	RTM 2 WORK AREA	IHARTM2A	RTM2NXT1 RTM2EREG
60	3C	ASID INFO	-	-
66	42	SUMLSTA RANGE	-	-

For an SVC entry to SDUMP, the SUMDUMP output can contain information that is not available in the remainder of the SVC dump if options such as region, LSQA, nucleus, and LPA were not specified in the dump parameters.

For each address space that is dumped, the SUMDUMP output is preceded by a header with the ASID, plus the jobname and stepname for the last task created in the address space. The SUMDUMP output contains RTM2 work areas for tasks in address spaces that are dumped. Many of the fields in the RTM2WA provide valuable debugging information. (See "Debugging Problem Program ABEND Dumps" for more details.)

The enabled summary dump is produced by module IEAVTSSE. The summary dump data is dumped in the following manner:

1. The ASID record (ID X'3C') is dumped for the address space in which IEAVTSSE is running.

2. The SUMLIST (ID X'2E') or SUMLSTA (ID X'42') ranges are dumped next. These contain information that is helpful in debugging the problem, and should be examined carefully.

3. All RTM2 work areas pointed to by all TCBs in this address space are dumped (ID X'39').

4. An address range table is built containing the following ranges:

   - 4K before and after the PSW at the time of error (RTM2NXT1)

   - 4K before and after each register at the time of error (RTM2EREG).

   Duplicate storage is eliminated from this address range table to reduce the amount of storage dumped. In the summary dump output, this storage is written in ascending order to make finding addresses easier. This storage is dumped with record ID X'30'.

SUMDUMP Output for Branch-Entry SDUMP

For branch entry to SDUMP, there are two types of summary dumps:

- Disabled summary dump - which is taken by module IEAVTSSD and performs the summary dump with the system disabled for interruptions. This means that all data to be dumped must be paged in at the time of the summary dump.

- Suspend summary dump - which is taken in two parts. The first part is similar to the disabled summary dump and dumps some of the global system control blocks. The second part is performed by module IEAVTSSV and runs with the system enabled for interruptions. This allows data to be dumped that is currently paged out, but was going to be modified by the recovery routine that called SDUMP.

The SUMDUMP output for a branch entry to SDUMP might not match the data that is at the same address in the remainder of the dump. The reason for this is that SUMDUMP is taken at the entry to SDUMP while the processor is disabled for interruptions. The system data in the remainder of the dump is often changed because other system activity occurs before the dump is complete. The SUMDUMP output follows a header that contains the ASID of the address space from which the data was obtained.

The following conditions can occur that prevent SDUMP from taking a disabled or suspend summary dump.

- RSM is not able to obtain the necessary locks to serialize the real storage buffer (RSB).
- RSM is in the process of modifying the storage queues and cannot satisfy the request for a RSB.
- No frames are available for a RSB.
- SDUMP encounters an error while holding RSM serialization for the RSB.
- A critical frame shortage causes RSM to steal the pages of the RSB.
- The SDUMP timer disabled interruption exit determines that SDUMP has failed and frees the RSB.

The following table summarizes, in the order in which they are printed, the SUMDUMP output types from a branch entry to SDUMP.

BRANCH-ENTRY TABLE

Record ID Dec	Hex	Title	Mapping Macro	Fields Used to Dump PSW or Register Areas:
73	49	Suspend Summary Dump in Error	-	-
67	43	XMEM ASID record	-	-
56	38	SDWA record	IHASDWA	SDWAGRSV,SDWANXT1
57	39	RTM2 work area	IHART2WA	-
3	3	PSA	IHAPSA	FLCIOPSW,FLCPOPSW FLCEOPSW,FLCROPSW
1	1	PCCA	IHAPCCA	-
2	2	LCCA	IHALCCA	-
68	44	Caller's ASCB	IHAASCB	-
69	45	Caller's TCB	IHARB	-
71	47	Caller's SSRB	IHASRB	-
5	5	Super FRR stacks	IHAYSTAK	-
80	50	SLIP reg/PSW area	-	-
47	2F	IHSA	IHAIHSA	-
64	40	XSB from IHSA	IHAXSB	-
65	41	PCLINK stacks from XSB	-	-
50	32	Global WSA vector table	IHAWSAVT	-
77	4D	Global work save areas	-	-
51	33	CPU WSA vector table	IHAWSAVT	-
78	4E	CPU work save areas	-	-
52	34	Local WSA vector table	IHAWSAVT	-
79	4F	Local work save areas	-	-
72	48	SDUMP caller's save area	-	-
46	2E	SUMLIST ranges	-	-
66	42	SUMLSTA ranges	-	-
48	30	Storage around regs/PSW at time of error	-	-
49	31	Storage around PSWs	-	-
53	35	End of Summary Dump	-	-

For **disabled summary dumps**, records are dumped in the following order:

1. If a suspend summary dump was requested but could not be taken, a disabled summary dump is done instead. If this occurs, an error record is written to that effect.

2. The XMEM ASID record is written that gives the ASID that is home, primary, secondary, and CML (if the CML lock is held).

3. The SUMLIST or SUMLSTA address ranges are dumped.

4. The PSA, PCCA, and LCCA for each processor are dumped.

5. The current PCLINK stack (pointed to by PSASEL) is dumped (if it exists).

6. If this is a SLIP request for a dump (ACTION=SVCD), then the SLIP reg/PSW area (pointed to by the SUMDUMP parameter list SDURGPSA) is dumped. This area contains the following data from the time the SLIP trap was entered:

Offset	Description of area
0	PSW at the time SLIP interrupted the program
8	Control register 3 (contains the secondary ASID)
C	Control register 4 (contains the primary ASID)
10	Registers 0-15 at the time SLIP interrupted the program.

   The following address ranges are added to the address range table:

   - 4K before and after the PSW address at the time of the SLIP trap.
   - 4K before and after each address in the registers at the time of the SLIP trap.

   Duplicate storage is eliminated from this address range table to reduce the amount of storage dumped.

   Note that if the primary and secondary ASIDs are different, the above address ranges are added to the table for both ASIDs.

7. The IHSA is dumped along with its associated XSB and PCLINK stack. The PSW and register addresses from the IHSA are added to the range table. This causes 4K of storage to be dumped around each address.

8. The caller's SDWA is dumped, if one exists. The PSW and register addresses from the SDWA are added to the range table. This causes 4K of storage to be dumped around each address.

9. The addresses in the address range table are dumped.

10. The super FRR stacks are dumped.

11. The global, local, and CPU work save area (WSA) vector tables are dumped. The save areas pointed to by each of these WSA vector tables are also dumped.

12. 4K of storage on either side of the address portion of the I/O old PSW, the program check old PSW, the external old PSW, and the restart old PSW saved in the PSA for all processors, are dumped.

For **suspend summary dumps**, records are dumped in the following order:

1. The ASID record; the PSA, PCCA, LCCA records; and the IHSA, XSB, and the PCLINK stack are all dumped with the system disabled in the same way they are dumped in steps 2, 4, and 5 for the disabled summary dump.

   At this point, an SRB is scheduled to the DUMPSRV address space and the current unit of work (SDUMP's caller) is suspended by using the STOP service. The rest of the summary dump is done by module IEAVTSSV running in SRB mode in the DUMPSRV address space. Data dumped at this point does not have to be paged in because the system is enabled. Cross memory functions are used to gain access to data in the caller's address space.

2. The SUMLIST or SUMLSTA address ranges are dumped.

3. The caller's ASCB is dumped.

4. The suspended unit of work (SDUMP's caller) is dumped. This is either a TCB or an SSRB. The related PCLINK stacks are also dumped.

5. For TCB mode callers, the caller's SDWA is dumped. The PSW and register addresses from the SDWA are added to the range table. This causes 4K of storage to be dumped around each address. All RTM2 work areas pointed to by this TCB and any associated SDWAs are all dumped.

   For SRB mode callers, the SDWA is dumped. The PSW and register addresses from the SDWA are added to the range table. This causes 4K of storage to be dumped around each address. Also, the caller's register save area is added to the range table and the storage dumped.

   Duplicate storage is eliminated from the address range table to reduce the amount of storage dumped.

6. After all the storage is saved in a virtual buffer in the DUMPSRV address space, the caller's unit of work is reset by using the RESET service. This allows SDUMP to complete and return to the caller. The rest of the dump is then scheduled from the DUMPSRV address space.

Dump Analysis and Elimination (DAE)

Dump analysis and elimination (DAE) reduces the number of unneeded dumps by suppressing duplicate SDUMPs and SYSMDUMPs. If the recovery routine enables dump suppression (by supplying the VRADAE in the SDWA) and the installation allows dump suppression (by specifying SUPPRESS in the ADYSETxx member of SYS1.PARMLIB), DAE suppresses dumps when selected symptom data in the dump's SDWA (and SDWAVRA) matches the symptom data from a dump (of the same dump type) previously taken.

System data set SYS1.DAE contains the symptom data of the unique problems for dumps that have previously occurred. When DAE is started (via the SET DAE= operator command), DAE makes a copy of this symptom data in the SQA. Before processing an SDUMP or SYSMDUMP dump, DAE searches the symptom data in the SQA for duplicate symptom data. When a duplicate set of symptom data is found, DAE informs SDUMP to suppress the dump. For suppressed SYSMDUMPs, message IEA838I is issued by ABDUMP and contains information about the SYSMDUMP.

Note: SLIP actions of SVCDUMP (SVCD), trace dump (TRDUMP), and no suppression (NOSUP) override DAE suppression processing. Dumps affected by these options are not suppressed by DAE.

Dump Header Record

When an SDUMP or SYSMDUMP is taken, the symptom record portion of the dump header record contains the symptoms related to the dump in both MVS and RETAIN format. (The DAE dump header record format routine (ADYHDFMT) formats the symptom data.) Informational messages indicate why the dump was not suppressed.

You can use the IPCS subcommand VERBEXIT DAEDATA or VERBEXIT SYMPTOMS to format the DAE-generated symptoms.

This symptom data is useful for (1) searching the FE RETAIN data base, (2) communicating with FE, and (3) starting the problem determination process. See the *Debugging Handbook* for an example of a dump header.

Additional Data Gathering Techniques

This chapter describes additional techniques for gathering data and circumventing certain system problems.

It contains the following topics:

- Using the CHNGDUMP, DISPLAY DUMP, DUMP, and DUMPDS Operator Commands
- Using IPCS to Analyze Dumps
- How to Print Dumps
- How to Automatically Establish System Options for SVC Dump
- How to Copy PRDMP Tapes
- How to Rebuild SYS1.UADS
- How to Print SYS1.DUMPxx
- How to Clear SYS1.DUMPxx Without Printing
- How to Print the SYS1.COMWRITE Data Set
- How to Print an LMOD Map of a Module
- How to Re-create SYS1.STGINDEX
- Software LOGREC Recording
- Using the PSA as a Patch Area

Using the CHNGDUMP, DISPLAY DUMP, DUMP, and DUMPDS Operator Commands

A dump obtained from MVS contains those storage areas specified in the dump request and those defined as system defaults in SYS1.PARMLIB for SYSABEND, SYSMDUMP, and SYSUDUMP. Normal system defaults are:

SYSABEND: CB, ENQ, TRT, ALLPA, SPLS, LSQA, PSW, REGS, SA, DM, SUM, IO, and ERR

SYSMDUMP: LSQA, NUC, RGN, SQA, SWA, SUM, and TRT

SYSUDUMP: SUM

For an SVC dump, the normal system defaults are SQA, ALLPSA, and SUMDUMP.

The **CHNGDUMP command** is used to dynamically alter the options specified originally by SYS1.PARMLIB or by previous CHNGDUMP commands. Dump mode may be set to ADD, OVER, or NODUMP. System action for each setting is:

ADD - merges the options specified on the dump request with the options in the system dump options list.

OVER - ignores the options specified in the dump request and uses only the options in the dump options list.

NODUMP - ignores the request and does not dump.

To determine the current system dump options, use the DISPLAY DUMP,OPTIONS command. If an error is made while specifying the CHNGDUMP command, the system rejects the command and issues an error message.

The topic "How to Automatically Establish System Options For SVC Dump," which appears later in this chapter, describes how to issue the CHNGDUMP command during IPL. See *System Commands* for the format of the CHNGDUMP command.

The **DISPLAY DUMP command** is used for the following:

- To display the effects of the CHNGDUMP command or to determine the current system dump options. (DISPLAY DUMP,OPTIONS)
- To determine which dump data sets are full and which are available. (DISPLAY DUMP,STATUS)
- To obtain a list of dump titles from full dump data sets. (DISPLAY DUMP,TITLES)
- To obtain specific information about the dumps that is useful in debugging, searching the APAR data base, or developing SLIP traps to isolate the problem. (DISPLAY DUMP,ERRDATA,DSN=xx)

See *System Commands* for the format of the DISPLAY DUMP command.

The **DUMP command** must be used carefully if the desired dump is to be obtained. For instance, the following typical error can occur when requesting a dump. The operator enters DUMP COMM=(title). The system responds with message IEE094 requesting the dump parameters. If the operator replies 'U' to this message, the system dumps the current address space which is the master scheduler address space. The operator must reply with ASID, Jobname, or TSOname. See *System Commands* for the format of the DUMP command.

The **DUMPDS command** is used to add and delete SYS1.DUMP data sets and tapes that are available to SDUMP. It is also used to clear full dump data sets when the dump is no longer needed. The following scenarios are possible uses of the DUMPDS command:

- Adding dump data sets after IPL.

  1. For example, the system is IPLed with the system parameter DUMP=(DASD,L) but the volume containing the dump data sets is offline and not available.

  2. The IPL is continued, and the volume is made available later.

  3. DUMPDS ADD,DSN=ALL is issued to make all cataloged dump data sets available to SDUMP.

- Deleting a dump data set.

  1. For example, dump data set SYS1.DUMP03 keeps getting partial dumps because of an I/O error on the data set.

  2. DUMPDS DEL,DSN=03 is issued to delete the failing dump data set.

- Directing a dump to a specific dump data set or tape:

  1. For example, when a problem occurs and the dump contains incomplete data, the dump must be recreated requesting the additional data that exceeded the capacity of the normal dump data sets.

  2. DUMPDS DEL,DSN=ALL is issued to delete all dump data sets from SDUMP.

  3. DUMPDS ADD,UNIT=570 is issued to add a tape on unit 570 as a dump data set.

  4. The dump is then recreated by running the job that causes the problem or by setting a SLIP trap requesting the additional data to be dumped.

  5. After the dump has been recreated, DUMPDS DEL,UNIT=570 is issued to delete the tape from use by SDUMP and to restore it to system use. DUMPDS ADD,DSN=ALL is then issued to reestablish the dump data sets that were deleted in the second step.

Note that the dump data sets added by the DUMPDS command are lost if the dumping services address space (DUMPSRV) fails. In this case, you must add dump data sets again via the DUMPDS command after DUMPSRV is restarted.

See *System Commands* for the format of the DUMPDS command.

Using IPCS to Analyze Dumps

For complete information on using IPCS for online examination of dumps, see the *Interactive Problem Control System (IPCS) User's Guide* and the *Interactive Problem Control System (IPCS) Command Reference.*

IPCS contains many subcommands that do dump analysis as well as dump formatting. To take advantage of the analysis capability of IPCS, consider the following suggested approach for starting problem diagnosis on a dump.

While in the IPCS dialog, choose the SUBMIT option on the IPCS primary option menu. Then, choose the appropriate option to process a stand-alone dump or an SVC dump. These actions activate a process that copies the dump from tape to a designated data set and runs a diagnostic CLIST containing IPCS subcommands against the dump data set.

For stand-alone dumps, IPCS runs the BLSCSCAN CLIST, which contains the following subcommands:

```
STATUS SYSTEM WORKSHEET CPU CONTENTION REGISTERS
ANALYZE EXCEPTION
SUMMARY TCBERROR
COMCHECK
DIVDATA EXCEPTION
VERBX IOSDATA 'EXCEPTION'
VERBX SRMDATA
CONTROL NOMSG
VERBX ASMDATA TERM NOPRINT
CONTROL MSG
VERBX TRACE
VERBX MTRACE
VERBX SYMPTOM
```

These subcommands provide the maximum amount of analysis with the minimum amount of output, and they provide a good starting point for dump analysis. The SYMPTOM verb executed at the end of the CLIST formats any RETAIN symptoms that were generated by previous subcommands.

Note: If the dump has already been loaded into a DASD data set, you can run the appropriate CLIST, depending on the type of dump:

- For SADMP, run BLSCSCAN.
- For SVC Dump, run BLSCBSVB.
- For SYSMDUMP, run BLSCBSYB.

How to Print Dumps

This section discusses the DD statements and control statements used in the following example:

```
//ASIDDMP JOB MSGLEVEL=1
//DMP EXEC PGM=IKJEFT01,PARM=AMDPRDMP
//SYSTSIN DD DUMMY,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSTSPRT DD DUMMY
//INDEX DD SYSOUT=A
//PRINTER DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//TAPE DD UNIT=TAPE,LABEL=(1,NL),VOL=SER=ABCTPE,DISP=OLD
//SYSUT1 DD UNIT=251,SPACE=(CYL,(20,1)),DISP=NEW
/* PRINT STORAGE=ASID(X)=(X,X,X,X,X,X) IS PROPER FORMAT
   CVTMAP
   CPUDATA
   SUMMARY FORMAT
   QCBTRACE
   SUMDUMP
   LPAMAP
   NUCMAP
   VSMDATA
   SRMDATA
   RSMDATA
   ASMDATA
   IOSDATA
   MTRACE
   DAEDATA
   LOGDATA
   VTAMMAP
   TRACE
   JES3
   JES2
   AVMDATA
   SADMPMSG
   TCAMMAP
   EDIT
   PRINT CURRENT,SQA
   PRINT STORAGE=ASID(X)=(xxxx,xxxx,xxxx,xxxx)
   PRINT JOBNAME=(jobnames)
   PRINT REAL=(xxxx,xxxx)
   END
```

See *Service Aids* for a complete description of PRDMP DD and control statements.

The INDEX DD statement defines the output data set containing the PRDMP table of contents. If the INDEX and PRINTER DD statements specify the same SYSOUT, putting INDEX before PRINTER causes the table of contents to be printed first in the listing.

The PRINTER DD statement defines the output data set for the dump itself. It should be directed to a SYSOUT class as shown.

The SYSPRINT DD statement defines the data set for PRDMP messages, etc.

The TAPE DD statement defines the input data set to PRDMP. It can define one of the SYS1.DUMPxx data sets, a stand-alone dump tape, or a GTF output data set on either tape or DASD.

The SYSUT1 DD statement defines work space to PRDMP. It can be used to define the input data set. It is not required if the input data set is defined by the TAPE DD statement. It does, however, significantly enhance the performance of PRDMP when it is used in conjunction with the TAPE DD statement and when the input is a tape data set.

The SPACE parameter is determined by the size of the dump. Generally 5 cylinders or 95 tracks or 285 4104 records should be specified for each megabyte of real storage dumped by SADMP.

Control Statements

The placement of the control statements determines the sequence in which the dump is printed. Refer to the "Dump and Trace Formats" section of the *Debugging Handbook* for examples of how these statements format a dump.

Note: To reduce the volume of output, select only those PRDMP control statements that provide the desired control blocks and output.

The following statements can be specified with PRDMP:

CVTMAP - formats the CVT and can be an aid in finding other significant control blocks in the system.

CPUDATA - formats the CSD, PSA, PCCA and LCCA for each active processor.

SUMMARY - defines and prints the dump ranges of the dump, active processor, active tasks, etc.

QCBTRACE, Q, or GRSTRACE - formats the ENQ/DEQ control blocks in use at the time the dump was taken.

SUMDUMP - locates and prints the summary dump data provided by SVC dump. Use SUMDUMP on all SVC dumps and stand-alone dumps. (For stand-alone dumps, SUMDUMP formats any summary dump records it finds in the real storage buffers. Such records could exist in the buffers if an SVC dump was in progress when a stand-alone dump occurs.)

LPAMAP - provides a listing of the contents of the modules that were in the link pack area, the directory entries, and the LPA active queue.

NUCMAP - formats the contents of the modules in the nucleus when the dump was generated.

VSMDATA - formats the major VSM control blocks.

SRMDATA - formats the major SRM control blocks.

RSMDATA - formats the major RSM control blocks.

ASMDATA - formats the major ASM control blocks.

IOSDATA - formats the major IOS control blocks.

MTRACE - formats the master trace table.

DAEDATA - formats and prints the DAE data for the dumped system.

LOGDATA - formats the in-storage LOGREC buffer records.

VTAMMAP - formats selected ACF/VTAM control blocks.

TRACE - formats the system trace table.

JES3 - formats specified JES3 control blocks.

JES2 - formats specified JES2 control blocks.

AVMDATA - formats the major availability manager control blocks.

SADMPMSG - formats the stand-alone virtual storage dump (SADMP) message log.

TCAMMAP - formats selected ACF/TCAM control blocks.

The EDIT statement formats and prints the GTF buffers (that is, all internal trace buffers or those external trace buffers that have not been written to the TRACE data set) if GTF is active at the time the dump is taken. If GTF is not active, only an error message is printed.

The PRINT statement can be used several ways:

- PRINT CURRENT,SQA - should be included in the initial run of PRDMP. It formats and prints the address space and task-related control blocks of the address space active at the time the dump is taken. SQA should be printed for the valuable data it contains such as LOGREC buffers. PRINT CURRENT prints only the current address space of the processor from which the SADMP program was IPLed; except in cross memory mode, PRINT CURRENT also includes the home, secondary, primary, and CML address spaces.

- PRINT NUC,CSA - should not be included in the initial run of PRDMP because of the volume of data it produces. Once a problem is suspected in this area, the PRDMP program should be rerun specifying only these parameters.

- PRINT STORAGE=ASID(x)=(xxxx,xxxx) - should not be included in the initial run of PRDMP. Once a problem is isolated to an address space or a range of storage addresses, rerun PRDMP specifying only these parameters. Several ASIDs and several address ranges can be requested with one run of PRDMP. PRDMP does not duplicate address ranges for every ASID but prints all storage dumped (NUC, CSA, SWA, LPA in storage) if only ASIDs are specified without address ranges. PRINT STORAGE is useful for printing SVC dumps. See the discussion "How to Print SYS1.DUMPxx" later in this chapter.

- PRINT JOBNAME=(jobnames) - produces output equivalent to PRINT CURRENT except it prints the private address space of job(s) requested. It should not be used for the initial run of PRDMP unless the jobname is known from another source, such as the system log.

- PRINT REAL=(xxxx,xxxx) - prints real storage in specified address range pairs. Use this option only when the system cannot find adequate data to format the dump.

How to Automatically Establish System Options For SVC Dump

A potential problem is that the SVC dumps written to the SYS1.DUMPxx contains only those address ranges that the FRR or ESTAE routine passes to SDUMP. When these dumps are subsequently printed by PRDMP, the PRDMP formatting program might not find sufficient data to format the dump properly. This can make it difficult to find data in an SVC dump and it can provide erroneous indicators to the problem solver.

The CHNGDUMP command can be used to alter the SVC dump system options and provide a complete dump. The following job updates the COMMND00 member of SYS1.PARMLIB to issue the CHNGDUMP command automatically at IPL time. The CHNGDUMP command can also be entered by the operator. (See *System Commands* for a description of the CHNGDUMP command.)

```
//UPDAT JOB (,,5,5),MSGLEVEL=1,REGION=100K
// EXEC PGM=IEBUPDTE
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=SYSDA,VOL=SER=SYSRES,DISP=OLD,
//   DSN=SYS1.PARMLIB
//SYSUT2 DD UNIT=SYSDA,VOL=SER=SYSRES,DISP=OLD,
//   DSN=SYS1.PARMLIB
//SYSIN  DD DATA
./ REPL NAME=COMMND00,LIST=ALL
./ NUMBER NEW1=10,INCR=20
COM='CD SET,SDUMP=(NUC,RGN),Q=YES,ADD'
./ ENDUP
```

How to Copy Dump Tapes

It is sometimes necessary to copy dump tapes to supply another location with a copy of the dump while retaining your own. It is particularly useful to be able to supply a dump tape with an APAR.

Following are several methods that you can use to copy a dump tape:

1. You can use the IPCS COPYDUMP subcommand. The following example shows how this is done in batch mode:

```
//IPCSCOPY JOB MSGLEVEL=1
//COPYDUMP EXEC PGM=IKJEFT01,REGION=600K
//IPCSDDIR DD DSN=IPCSU1.DUMPDIR.DEBUG,DISP=(OLD,KEEP)
//TAPEIN   DD UNIT=TAPE,LABEL=(1,NL),VOL=SER=DMPIN,
//   DCB=(DSORG=PS,RECFM=F,BLKSIZE=4104,LRECL=4104)
//TAPEOUT  DD UNIT=TAPE,LABEL=(2,NL),VOL=SER=DMPOUT,
//   DCB=(DSORG=PS,RECFM=F,BLKSIZE=4104,LRECL=4104)
//SYSTSPRT DD DUMMY
//SYSTSIN  DD *
IPCS NOPARM
COPYDUMP INFILE(TAPEIN) OUTFILE(TAPEOUT) NOPRINT NOCONFIRM
END
/*
```

2. Also, you can use the IEBGENER utility program. The following example shows how this is done:

```
//COPYDMP JOB MSGLEVEL=1
// EXEC PGM=IEBGENER
//SYSIN DD DUMMY
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=TAPE,LABEL=(2,NL),VOL=SER=DMPIN,
//   DCB=(RECFM=FB,LRECL=4104,BLKSIZE=4104)
//SYSUT2 DD UNIT=TAPE,LABEL=(1,NL),VOL=SER=DMPOUT,
//   DCB=(RECFM=FB,LRECL=4104,BLKSIZE=4104)
/*
```

3. And, you can use PRDMP as a copy program. Define the input tape with the TAPE DD statement and the output tape with the SYSUT2 DD statement. It is also possible to put several dumps on one tape or take one dump from a multiple dump tape by manipulating the file number parameters in the label parameter. The following example shows how this is done:

```
//ASIDDMP JOB MSGLEVEL=1
//DMP EXEC PGM=IKJEFT01,PARM=AMDPRDMP
//SYSTSIN DD DUMMY,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSTSPRT DD DUMMY
//INDEX DD SYSOUT=A
//PRINTER DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//TAPE DD UNIT=TAPE,LABEL=(2,NL),VOL=SER=DMPIN,DISP=OLD
//SYSUT2 DD UNIT=TAPE,LABEL=(,NL),VOL=SER=DMPOUT,
//   DISP=(NEW,KEEP)
//SYSIN DD *
  END
/*
```

After copying a dump tape, a quick run through PRDMP to verify that the CVT can be formatted and printed will prove that the copy was successful.

```
//ADMP JOB MSGLEVEL=1
//DMP EXEC PGM=IKJEFT01,PARM=AMDPRDMP
//SYSTSIN DD DUMMY,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSTSPRT DD DUMMY
//INDEX DD SYSOUT=A
//PRINTER DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//TAPE DD UNIT=TAPE,LABEL=(1,NL),VOL=SER=DMPTPE,DISP=OLD
//SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(400,20)),DISP=NEW
//SYSIN DD *
  CVTMAP
 END
/*
```

How to Rebuild SYS1.UADS

The loss of the SYS1.UADS data set can significantly impact a TSO environment. However, it is possible to run the TMP as a batch job and recreate SYS1.UADS in the background. The following is an example of a job that has been run successfully to scratch and recreate a SYS1.UADS data set.

```
//BLDUADS JOB MSGLEVEL=1
// EXEC PGM=IEFBR14
//DD2 DD VOL=SER=SYSRES,DISP=(OLD,DELETE),UNIT=3330,
//  DSN=SYS1.UADS
//  EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=A
//SYSUADS DD DSN=SYS1.UADS,DISP=(NEW,KEEP),
//   SPACE=(800,(20,9,30)),UNIT=3330,
//   VOL=SER=SYSRES,DCB=(RECFM=FB,
//   DSORG=PO,LRECL=80,BLKSIZE=800)
//SYSLBC DD DSN=SYS1.BRODCAST,DISP=SHR
//SYSIN DD *
ACCOUNT
SYNC
ADD (USER01 TSOTE01 * IKJACC01) UNIT(SYSDA) ACCT OPER JCL MOUNT
ADD (USER02 TSOTE02 * IKJACC02) UNIT(SYSDA) OPER JCL MOUNT
ADD (USER03 TSOTE03 * IKJACC03) UNIT(SYSDA) JCL MOUNT
ADD (USER04 TSOTE04 * IKJACC04) UNIT(SYSDA) JCL MOUNT
ADD (USER05 TSOTE05 * IKJACC05) UNIT(SYSDA) JCL MOUNT
ADD (USER06 TSOTE06 * IKJACC06) UNIT(SYSDA) JCL MOUNT
ADD (USER07 TSOTE07 * IKJACC07) UNIT(SYSDA) JCL
ADD (USER08 TSOTE08 * IKJACC08) UNIT(SYSDA) JCL
ADD (USER09 TSOTE09 * IKJACC09) UNIT(SYSDA) OPER
ADD (USER0A TSOTE0A * IKJACC0A) UNIT(SYSDA)
ADD (USER0B TSOTE0B * IKJACC0B) UNIT(SYSDA)
ADD (USER0C TSOTE0C * IKJACC0C) UNIT(SYSDA)
LIST (*)
END
/*
```

How to Print SYS1.DUMPxx

See the discussion under "How to Print Dumps" earlier in this chapter to define the control statements required. The same rules apply except in this case the TAPE DD statement points to one of the SYS1.DUMPxx data sets. These are cataloged data sets and require no further definition. Note that DISP=SHR must be specified for SYS1.DUMPxx data sets allocated to SVC dump.

Be aware that the dump data sets contain only those address ranges passed to SVC dump by the dump requestor and might not contain sufficient data for PRDMP to properly format all requested control blocks.

See the next topic "How to Clear SYS1.DUMPxx Without Printing" for a description of how to clear the dump data sets for reuse.

The following example shows how to print SYS1.DUMP00:

```
//ASIDDMP JOB MSGLEVEL=1
//DMP EXEC PGM=IKJEFT01,PARM=AMDPRDMP
//SYSTSIN DD DUMMY,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSTSPRT DD DUMMY
//INDEX DD SYSOUT=A
//PRINTER DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//TAPE DD DSN=SYS1.DUMP00,DISP=SHR
//SYSUT1 DD UNIT=SYSDA,DISP=NEW,SPACE=(CYL,(10,5))
//SYSIN DD *
  SUMMARY
  FORMAT
  CVTMAP
  CPUDATA
  SUMDUMP
  LPAMAP
  PRINT STORAGE
/*
```

How to Clear SYS1.DUMPxx Without Printing

The dump data sets can be cleared at 'SPECIFY SYSTEM PARAMETERS' time during IPL by specifying the LIST option in the DUMP=(DASD,L) parameter. Also, dump data sets can be cleared by using the DUMPDS command. Some examples of the DUMPDS command are:

- DUMPDS CLEAR,DSN=03 - clears dump data set SYS1.DUMP03.
- DUMPDS CLEAR,DSN=(00-20) - clears the range of DASD dump data sets from SYS1.DUMP00 through SYS1.DUMP20.
- DUMPDS CLEAR,DSN=ALL - clears all DASD dump data sets.
- DUMPDS CLEAR,UNIT=570 - clears the dump tape at device number 570.

They can also be cleared and made available for reuse by using PRDMP to copy the data set to tape with the SYSUT2 DD statement pointing to the output data set. This must be a separate job step from printing the dump. If it has been determined that the SYS1.DUMPxx data set need not be saved, it can be cleared and made available for reuse by running PRDMP with the SYSUT2 DD

statement defined as DUMMY. The following example shows how to clear SYS1.DUMP00. See the example in the discussion "How to Copy PRDMP Tapes" earlier in this chapter for how to define the SYSUT2 DD statement to unload the SYS1.DUMPxx data sets.

```
//ASIDDMP JOB MSGLEVEL=1
//DMP EXEC PGM=IKJEFT01,PARM=AMDPRDMP
//SYSTSIN DD DUMMY,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSTSPRT DD DUMMY
//INDEX DD SYSOUT=A
//PRINTER DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//TAPE DD DSN=SYS1.DUMP00,DISP=SHR
//SYSUT2 DD DUMMY
//SYSIN DD *
  END
```

Another, and faster way to clear a SYS1.DUMPxx data set without printing is to use the IEBGENER utility program. The following example shows how to clear SYS1.DUMP00:

```
//CLEARDMP JOB MSGLEVEL=1
//      EXEC PGM=IEBGENER
//SYSIN DD DUMMY
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=SYS1.DUMP00,DISP=SHR,DCB=SYS1.DUMP00
//SYSUT2 DD DUMMY,DCB=SYS1.DUMP00
/*
```

How to Print the SYS1.COMWRITE Data Set

The following job will format and print the TCAM SYS1.COMWRITE data set. Note that the PARM fields in the EXEC statement define the traces to be formatted and printed. See *ACF/TCAM Diagnosis Guide* for more information on the use of the SYS1.COMWRITE data set.

```
//COMWRITE JOB MSGLEVEL=1
//STEP1 EXEC PGM=IEDQXB,PARM='STCB,IOTR,BUFF'
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=SYS1.COMWRITE,DISP=SHR
/*
```

How to Print an LMOD Map of a Module

The following job produces a module cross-reference of the nucleus, module IEFW21SD, and a link pack area map. In addition, AMBLIST produces an IDR listing or a complete hexadecimal dump of an object module. If you include the RELOC parameter, the cross-reference listing is based at the address the module is loaded in LPA.

Note that the JCL must contain a DD statement for every data set containing a module you referenced in the control card section.

For more information about AMBLIST, see *Service Aids*.

```
//AMBLIST JOB MSGLEVEL=1
//  EXEC PGM=AMBLIST
//SYSLIB DD DSN=SYS1.LPALIB,DISP=OLD
//LOADLIB DD DSN=SYS1.NUCLEUS,DISP=OLD
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  LISTLOAD OUTPUT=XREF,MEMBER=IEANUC01,DDN=LOADLIB
  LISTLPA
  LISTLOAD OUTPUT=XREF,MEMBER=IEFW21SD
/*
```

How to Re-Create SYS1.STGINDEX

It is possible for the SYS1.STGINDEX data set to be destroyed because of system failure or operator intervention during an IPL with the cold start (CLPA,CVIO) option. Loss of this data set prevents warm starting the system or restarting jobs using VIO data sets.

The following job can recreate this data set. Remember to change the VOLUME and CYLINDERS parameters to apply to your system.

```
//STGINDEX JOB MSGLEVEL=1
// EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//VOL DD DISP=OLD,UNIT=3330,VOL=SER=SYSRES
//SYSIN DD *
 DEFINE SPACE(VOL(SYSRES)FILE(VOL)CYL(7))
 DEFINE CLUSTER-
 (NAME(SYS1.STGINDEX)-
 VOLUME(SYSRES)-
 CYLINDERS(7)-
 KEYS(128)-
 BUFFERSPACE(5120)-
 RECORDSIZE(2041 2041)-
 REUSE)-
 DATA-
 (CONTROLINTERVALSIZE(2048))-
 INDEX-
 (CONTROLINTERVALSIZE(1024))
```

Software LOGREC Recording

The following JCL defines a two-step job. The first step prints an event history report for all SYS1.LOGREC records. The second step formats each software, IPL, and EOD record individually. The event history report is printed as a result of the EVENT=Y parameter on the EXEC statement of the first step. It can be a very useful tool to the problem solver because it prints the records in the same sequence they were recorded and therefore shows an interaction between hardware error records and software error records.

```
//EREP JOB MSGLEVEL=1
//EREPA EXEC PGM=IFCEREP1,PARM='EVENT=Y,ACC=N',
//   REGION=128K
//SERLOG DD DSN=SYS1.LOGREC,DISP=SHR
//TOURIST DD SYSOUT=A
//EREPPT DD SYSOUT=A,DCB=BLKSIZE=133
//EREPB EXEC PGM=IFCEREP1,PARM='TYPE=SIE,PRINT=PS,ACC=',
//   REGION=128K
//SERLOG DD DSN=SYS1.LOGREC,DISP=SHR
//TOURIST DD SYSOUT=A
//EREPPT DD SYSOUT=A,DCB=BLKSIZE=133
/*
```

See the discussion on LOGREC analysis in the "Use of Recovery Work Areas" chapter earlier in this section.

Using the PSA as a Patch Area

There are areas in the PSA reserved for future expansion. They can be used for quick implementation of a trap without having to consider base registers. Check the mapping of the PSA (IHAPSA) for possible areas to be used. Once an area is chosen, verify that the value of this storage is zero.

CAUTION: Use extreme care when you use this method. Patches should be made only to disabled code unless the patch is completely reentrant. Saving registers and data in the PSA while the system is enabled could produce unpredictable results, especially in an MP environment where more than one PSA exists and the code could be interrupted and subsequently redispatched on the other processor. Extreme care must be used when considering a system alteration in order to gather additional data about a problem. No superzaps should be applied before the system programmer has verified the logic being zapped and the trap logic itself. Remember if any one location or offset within the module or trap changes, all offsets and base registers *must* be verified.

SLIP Command

The SLIP (serviceability level indication processing) command is designed to intercept or trap system events. You specify the kinds of events you want trapped and the system action to be taken when the trap matches. See *System Commands* for the syntax of the SLIP command.

Using the SLIP Command

The SLIP command provides a way of getting information concerning an error prior to ESTAE or FRR recovery processing. This is in addition to the information ordinarily supplied by dumping services during abnormal termination. The SLIP command is also used to establish PER monitoring for instruction fetch, storage alteration, and successful branch PER events within a range of virtual addresses. When the requested PER event occurs, a PER interrupt causes control to be given to the SLIP processor. The purpose of the SLIP command is to establish SLIP traps that describe the system conditions which must exist at the time of the error or interrupt so that an action will be taken.

The SLIP command is usually entered by a system programmer, either at the console or via the input stream. It can also reside in the IEASLPxx, IEACMD00, and COMMNDxx members of SYS1.PARMLIB. The SLIP command can also be entered as a subcommand of the OPERATOR command from a TSO terminal or TSO CLIST. Information about SLIP traps can be displayed by using the DISPLAY operator command.

As long as enough system queue area storage is available, SLIP traps may be established at any time. The recovery termination manager (RTM) compares the SLIP trap event qualifiers with the dynamic system conditions at the time of the error or interrupt. If RTM detects a match, the requested action is taken.

SLIP Event Qualifier Keywords

When specified on a trap, an event qualifier keyword is checked against current system conditions to determine a match or no match condition. The following keywords are described in this topic.

ADDRESS	LPAMOD
ASID	MODE
ASIDSA	NUCEP
COMP	NUCMOD
DATA	PVTEP
ERRTYP	PVTMOD
JOBNAME	RANGE
JSPGM	RBLEVEL
LPAEP	REASON

Because the RTM/SLIP processor runs in one of four environments, the checking done for each event qualifier keyword is described in terms of the applicable environment. The four environments are:

RTS - an error has occurred that will cause routing to FRRs.

RT2 - an error has occurred while in enabled unlocked task mode, which will cause routing to ESTAEs but not FRRs.

RTM - an abnormal address space termination has occurred.

PER - a PER interrupt has occurred.

Note: The names of these four environments are taken from the names of the modules that call the RTM/SLIP processor; namely IEAVTRTS, IEAVTRT2, IEAVTRTM, and IEAVTPER.

ADDRESS Event Qualifier Keyword

The checking done for the ADDRESS keyword is:

RTS - the address from the PSW at the time of the error (SDWANXT1) is used in the comparison.

RT2 - the RBLEVEL keyword determines which RB is used to get the address (RBPSWNXT) used in the comparison.

RTM - the address of the instruction that branched to RTM is used in the comparison.

PER - the address of the instruction that caused the PER interrupt (LCCAPERA) is used in the comparison.

ASID Event Qualifier Keyword

The checking done for the ASID keyword is:

RTS - The PASID of the failing address space (field SDWAPRIM) is used in the comparison.

RT2 - The PASID in the XSB of the RB located from the RBLEVEL keyword (field XSBPASID) is used in the comparison.

RTM - The PASID of the address space being terminated is used in the comparison.

PER - The PASID at the time of the interruption is used in the comparison.

ASIDSA Event Qualifier Keyword

The ASIDSA keyword associates an address space with a storage alteration (SA). ASIDSA is only valid when it is specified with SA on the SLIP command. The PASID, SASID, HASID, and the S-bit at the time of the interruption are used to determine the address space where the storage alteration occurred. On an SA trap, ASID refers to the address space where the instructions are being executed, and ASIDSA refers to the address space where the storage being altered is located.

COMP Event Qualifier Keyword

The checking done for the COMP keyword is:

RTS - field SDWACMPC is used in the comparison.

RT2 - field RTM2CC is used in the comparison.

RTM - the completion code for the address space being terminated (ASCBMCC) is used in the comparison.

Because many recovery routines change the abend completion code to make it more specific, the value supplied on the COMP keyword must be the original value before a recovery routine changes the code. This is because RTM/SLIP checking is done before processing by the recovery routines.

For example, a SLIP trap will not match if any of the following completion codes are specified: 11A, 12E, 15D, 15F (reason codes 12 and 16), 200, 212, 25F, 279, 282, 402, 42A, 57D, 6FC, 700, 72A, A00, B00, and E00. Most of these codes were originally a program check (0C4) that has been converted to a more specific value. If you want to specify a program check, use COMP=0C4 or ERRTYP=PROG. To avoid having the SLIP action occur for all program checks, you should also specify some other event qualifier such as program name or module name.

Similarly, specification of 13E or 33E might prevent a trap from matching if these completion codes occur for any active subtasks associated with a task that is abending. These secondary abends occur for the purpose of clean-up and therefore the SLIP processor is not called when they occur.

Note: The SDUMP and ABDUMP dumping programs might cause many 0C4 program checks while taking a dump. Therefore, when you specify COMP=0C4 on a trap, you should avoid these unwanted matches by also specifying another event qualifier, such as MATCHLIM.

DATA Event Qualifier Keyword

The DATA keyword checks data in the system at the time of the error or interrupt against the data conditions specified in the SLIP trap. Addressing is established to the address space specified with the address. Indirect addresses are resolved by using the registers at the time of the error or interrupt.

For some errors, register contents at the time of error are not valid. In such a case, if registers are used, the data is considered unavailable and the trap does not match. Other conditions that can cause the data unavailable situation are: the address space specified with the address does not exist; data is paged out; or a pointer required for an indirect address is paged out.

When data is unavailable, a counter associated with the trap is incremented and message IEA413I is sent to the SLIP user. For a PER trap, message IEA413I is issued only the first time that the data unavailable situation occurs. However, the counter is made available by displaying the trap. The data unavailable counter is also part of the SLIP standard, SLIP standard/user, and SLIP DEBUG GTF trace records.

The SLIP command processor does not perform any reasonability checking on the data tests requested. For example, the SLIP command processor would allow the following DATA keyword specification even though its specification prevents the trap from ever matching.

```
DATA=(H.CD300,EQ,00,HASID.CD300,NE,00)
```

The DATA keyword may be used as a validity check for RANGE addresses or LPAMOD offsets when used on an IF (instruction fetch) or SB (successful branch) PER trap. For example, if RANGE=(CD300,CD303) is used to establish the range of addresses for an IF trap, you can ensure that the expected instruction is being monitored by specifying DATA=(CD300,EQ,47F0B020) where 47F0B020 is the expected instruction. If the wrong instruction is being monitored (for example, due to a typing error or a change in the system), the trap would not match because the DATA keyword does not match. This technique can be especially useful on traps that take potentially disruptive actions (for example, WAIT or RECOVERY actions) in order to ensure the action is taken only when desired.

To make additional validity checks on the same PER trap to ensure that the action is taken only when desired, you can AND or OR several sets of data, such as:

```
DATA=(3R,EQ,0,AND(2R,EQ,00BF12C0,OR,2R,EQ,00BF12C1))
```

You can compare the contents of a register or a storage location to:

- A constant value, or
- The contents of another register or storage location, or
- The address determined by an indirect address definition.

ERRTYP Event Qualifier Keyword

The checking done for the ERRTYP keyword is:

RTS - the RT1TENPT field is used in the comparison. The value field RT1TENPT of the RTM1 work area indicates the reason for entry into RTM1:

1=PROG	3=SVCERR	5=MACH
2=REST	4=DAT	10=PGIO

The SLIP processor recognizes an SVC error for SVC 13 as either an ABEND or SVCERR error and allows a match for the ERRTYP keyword if either is specified.

RT2 - the RTM2ERRA field is used in the comparison. The reason for entry into RTM2 is indicated by flags in the RTM2 work area as follows:

RTM2MCHK=MACH	RTM2ABTM=ABEND
RTM2PCHK=PROG	RTM2TEXC=DAT
RTM2RKEY=RESTART	RTM2PGIO=PGIO
RTM2SVCD=ABEND	

RTM - an abnormal address space termination (MEMTERM) causes a match.

JOBNAME Event Qualifier Keyword

The checking done for the JOBNAME keyword is:

RTS - if the failing address space has been identified by RTM (field SDWAFMID), both the failing and current address space job names are tested for a match. The job names pointed to by fields ASCBJBNI and ASCBJBNS are tested and either job name may match the job name specified in the trap.

- if the failing address space has not been identified, then only the current address space is tested for a job name match.

RT2 - if the failing address space has been identified by RTM (field RTM2FMID), both the failing and current address space job names are tested for a match. The job names pointed to by fields ASCBJBNI and ASCBJBNS are tested and either job name may match the job name specified in the trap.

- if the failing address space has not been identified, then only the current address space is tested for a job name match.

RTM - the job names for the address space being terminated are used in the comparison.

PER - the job names for the current address space are used in the comparison.

Note: The job, logon, or started task named by JOBNAME need not be active when the trap is set.

JSPGM Event Qualifier Keyword

The checking done for the JSPGM keyword is:

RTS - if a job step program name is available (field JSCBPGMN), it is compared to the job step program name specified in the trap.

- if a job step program name is not available (PSATOLD or TCBJSCBB=0), the trap will not match.

- if the reason for entry is a DAT error, the trap will not match.

RT2 - if a job step program name is available (field JSCBPGMN), it is compared to the job step program name specified in the trap.

- if a job step program name is not available (PSATOLD or TCBJSCBB=0), the trap will not match.

RTM - the trap will not match because the job step program name in the address space being terminated is not available.

PER - if a job step program name is available (field JSCBPGMN), it is compared to the job step program name specified in the trap.

- if a job step program name is not available (PSATOLD or TCBJSCBB=0), the trap will not match.

LPAEP Event Qualifier Keyword

LPAEP is the same as LPAMOD, except that SLIP checks from the entry point address or alias address to the end of the module.

LPAMOD Event Qualifier Keyword

The checking done for the LPAMOD keyword is:

RTS - the address from the PSW at the time of error (field SDWANXT1) is used in the comparison.

RT2 - the RBLEVEL keyword determines which RB is used to get the address (field RBPSWNXT) to be used in the comparison.

RTM - the address of the instruction that branched to RTM is used in the comparison.

PER - the address of the instruction that caused the PER interrupt (field LCCAPERA) is used in the comparison. (For additional information, refer to the note under the description of the RANGE keyword.)

Note: If the name specified on the LPAMOD keyword is an alias of a load module name, all monitoring is done by SLIP as though the load module name was specified.

MODE Event Qualifier Keyword

The system mode at the time of error is indicated in the MODEBYTE as follows:

1...	MODESUPR	Supervisor control
.1..	MODEDIS	Physically disabled
..1.	MODEGSPN	Global spin lock held
...1	MODEGSUS	Global suspend lock held
.... 1...	MODELOC	Locally locked
.... .1..	MODETYP1	Type 1 SVC
.... ..1.	MODESRB	SRB mode
....1	MODETCB	Task mode (unlocked)

The checking done for the MODE keyword is:

RTS - as a part of error processing, RTM determines the mode of the system (MODEBYTE value in field RT1WMODE). Also, the PSW at the time of the error (field SDWAEC1) is examined to determine key and state. The SDWASTAF bit indicates if the error occurred while a recovery routine was in control. The PASID at the time of the error (SDWAPRIM) is compared to the HASID. If they are equal, the instruction executed in home mode.

RT2 - the system mode, as determined by RTM, is obtained from the ABEND SVRB extended save area (MODEBYTE value in field ESAMODE). Also, the PSW (field RBOPSW) in the RB (as determined by RBLEVEL processing) is examined for key and state. The RTM2RECR and RTM2XIP bits indicate if a recovery routine was in control at the time of error. The PASID at the time of the error (obtained by the RBLEVEL keyword, field XSBPASID) is compared to the HASID. If they are equal, the instruction executed in home mode.

RTM - because RTM has not determined the system mode for an address space termination, various fields are tested by the RTM/SLIP processor to determine the system mode at the time of the MEMTERM request. (The mode will always indicate supervisor state and system key because these are requirements for issuing a MEMTERM request.)

PER - various fields are tested to determine the system mode at the time of the interrupt. In the SLIP trace record (system mode indicators), all bits are filled in. However, no attempt is made to determine if a recovery routine was in control when the interrupt occurred. Therefore, the recovery-routine-in-control bit will always be zero for a PER interrupt. Because of this, MODE=RECV is invalid for a PER trap and if MODE=ALL is specified for a PER trap, ALL does not include RECV (recovery-routine-in- control). If the PASID and HASID are equal at the time of the interruption, the home mode indicator is set.

NUCEP Event Qualifier Keyword

NUCEP is the same as NUCMOD.

NUCMOD Event Qualifier Keyword

The checking done for the NUCMOD keyword is:

RTS - the address from the PSW at the time of error (field SDWANXT1) is used in the comparison.

RT2 - the RBLEVEL keyword determines which RB is used to get the address (field RBPSWNXT) to be used in the comparison.

RTM - the address of the instruction that branched to RTM is used in the comparison.

PER - the address of the instruction that caused the PER interrupt (field LCCAPERA) is used in the comparison. (For additional information, refer to the note under the description of the RANGE keyword.)

Note: If the name specified on the NUCMOD keyword is an alias of a load module name, the address of the alias name or alternate entry point is used.

PVTEP Event Qualifier Keyword

PVTEP is the same as PVTMOD, except that SLIP checks from the entry point address or alias address to the end of the module.

PVTMOD Event Qualifier Keyword

The checking done for the PVTMOD keyword is:

RTS - if RTM has identified a failing address space (field SDWAFMID) and it is not current, the trap will not match.

- to check for a private area module, the local lock must be obtained. If it cannot be obtained, the trap will not match. If the local lock is already held, the chain that is to be searched for a private area module may be in the process of being changed. The search is performed, but the results may not be valid. The address obtained from the PSW at the time of error (field SDWANXT1) is used in the comparison.

RT2 - the RBLEVEL keyword determines which RB is used to get the address (field RBPSWNXT) used in the comparison.

RTM - this keyword test will not match because the private area chain in the failing address space is not available for searching.

PER - the trap will not match for the PVTMOD keyword test if the interrupt occurs in the nucleus or FLPA because these areas cannot contain private area modules.

- to check for a private area module, the local lock must be obtained. If it cannot be obtained, the trap will not match. If the local lock is already held, the chain that is to be searched for a private area module may be in the process of being changed. The search is performed, but the results may not be valid. The address of the instruction that caused the PER interrupt (field LCCAPERA) is used in the comparison.

In a cross-memory environment, the primary address space will be searched. Then, if needed and the corresponding local lock can be obtained, the home address space will be searched.

If offsets are specified on the PVTMOD keyword, the RTM/SLIP processor does not check to make sure that the offsets define an area wholly within the private area module.

RANGE Event Qualifier Keyword

The checking done for the RANGE keyword is:

PER - the address of the instruction that caused the PER interrupt (field LCCAPERA) is used in the comparison. Note that if the first address specified is greater than the second, the monitored range wraps storage addresses.

Notes:

1. *Consider the range carefully on any PER trap. A wide range could cause performance degradation due to the processing overhead for many PER interrupts. For example, for an address range that wraps storage, such as (700,600), PER events might occur too quickly for the system to disable the SLIP trap. If this happens, you can reset to zero control registers 9, 10, and 11. This action disables PER and also defines a minimum address range.*

2. *For successful branch monitoring, hardware PER processing does not check the address range specified on the RANGE and LPAMOD keywords. This means that a branch taken by an instruction anywhere in the system would cause a successful branch PER interrupt. However, to simulate an address range for successful branch monitoring, SLIP initially sets up instruction fetch monitoring for the desired address range. Then when instruction processing enters the requested range (indicated by an instruction fetch PER interrupt), PER monitoring is automatically switched to successful branch mode. You should be aware that the first PER event that occurs when instruction processing enters the requested range may not be a successful branch event. This "extra" event (instruction fetch) may affect values supplied for other keywords such as MATCHLIM. When instruction processing leaves the requested range, PER monitoring returns to instruction fetch monitoring on the requested range to avoid unnecessary PER interrupts. If the instructions being monitored are enabled for I/O and/or external interrupts, control may leave and then re-enter the monitored range due to normal interrupt processing.*

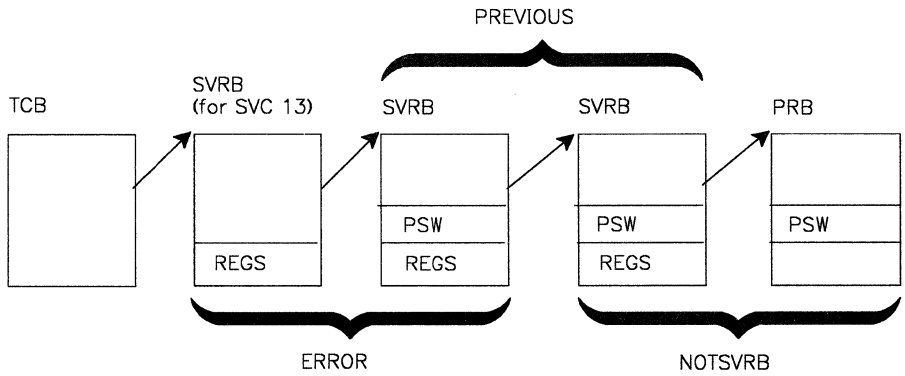
   This note applies to processing on behalf of a non-IGNORE successful branch PER trap. Mode switching does not occur for successful branch PER traps with ACTION=IGNORE specified. This means that if the initial entry into a monitored area matches an IGNORE trap, the mode remains instruction fetch and the "extra" event is delayed. Also, output that appears to be a contiguous successful branch trace may not actually be contiguous if an IGNORE trap matches intermittently.

3. *For successful branch monitoring, if an EXECUTE instruction has a successful branch target, the location of the EXECUTE instruction is used to determine whether or not the branch was within the monitored area without regard to the location of the executed branch.*

RBLEVEL Event Qualifier Keyword

The RBLEVEL keyword applies only to enabled unlocked task mode errors. It is used to direct the SLIP processor to the registers and PSW of interest for a particular error. The SLIP processor will use the PSW identified by the RBLEVEL keyword when processing the LPAMOD, PVTMOD, ADDRESS, and MODE keywords. The SLIP processor will use the registers identified by the RBLEVEL keyword when processing the DATA, TRDATA, LIST, and SUMLIST keywords.

The following diagram shows which RBs are chosen by the three RBLEVEL keyword options for an example RB chain.



The RBLEVEL keyword can be used in an error situation where several nested services are involved. For example, program A calls service B which calls service C. If an error occurs in service C, the default RBLEVEL=ERROR can be used to qualify the error or obtain information concerning the error. However, if the error in service C is the result of incorrect input supplied by service B or program A, the RBLEVEL=PREVIOUS or RBLEVEL=NOTSVRB can be used to qualify the error or obtain information concerning the input supplied by service B or program A respectively.

REASON Event Qualifier Keyword

The checking done for the REASON keyword is:

RTS - field SDWACRC is used in the comparison.
RT2 - field RTM2CRC is used in the comparison.
RTM - field ASCBARC is used in the comparison.

RTM/SLIP checking is done before processing by recovery routines. Therefore, the source of a reason code used in the comparison is from the REASON parameter coded on the ABEND or CALLRTM macro.

Using the ACTION Keyword

The ACTION keyword is used to specify the action to be taken when a SLIP trap matches the specified system conditions. The following ACTION options are described in this topic:

ACTION=SVCD	- schedule an SVC dump.
ACTION=WAIT	- put the system in a wait state.
ACTION=TRACE	- write a GTF trace record.
ACTION=TRDUMP	- write a GTF trace record and then schedule an SVC dump.
ACTION=STRACE	- write an SPER type of entry in the system trace table (PER traps only).
ACTION=STDUMP	- write an SPER type of entry in the system trace table and then schedule an SVC dump (PER traps only).
ACTION=RECORD	- write a record to SYS1.LOGREC (non-PER traps only).
ACTION=NODUMP	- suppress all dump requests (non-PER traps only).
ACTION=NOSUP	- bypass the suppression of dumps (non-PER traps only).
ACTION=NOSVCD	- suppress SVC dump requests (non-PER traps only).
ACTION=NOSYSA	- suppress SYSABEND dump requests (non-PER traps only).
ACTION=NOSYSM	- suppress SYSMDUMP dump requests (non-PER traps only).
ACTION=NOSYSU	- suppress SYSUDUMP dump requests (non-PER traps only).
ACTION=IGNORE	- take the IGNORE action.
ACTION Keyword	- with RECOVERY option (PER traps only), initiate recovery processing.

ACTION=SVCD Option

ACTION=SVCD indicates that an SVC dump will be scheduled for the failing or home ASID. This is the default option if ACTION is not specified. If the SVC dump cannot be taken, message IEA412I is issued and SLIP processing continues. No attempt is made to reschedule the SVC dump.

One of the advantages of the SVC dump over one taken by a recovery routine is that nothing has been done to correct the error situation. Although the bulk of the SVC dump is not taken until later, the summary dump portion preserves as much volatile data as possible. An SVC dump also contains more data (for example, more than one address space can be dumped) than a SYSABEND or SYSUDUMP, and because it is machine readable, it can, if necessary, be copied onto a tape to accompany an APAR, or used with interactive dump display programs. SYSMDUMP also provides a machine readable dump. The biggest advantage is in situations where no dump was occurring.

When ACTION=SVCD is specified or defaulted, the default SDATA parameters are: SQA, RGN, TRT, LPA, CSA, NUC, ALLPSA, and SUM. These default SDATA parameters are affected by the current CHNGDUMP command settings which may add to or override the requested dump options. The SDATA parameters can be changed by the SLIP user. Refer to "Dump Tailoring" later in this section.

If an SVC dump is already in progress, another dump cannot be taken and message IEA412I is issued. When an SVC dump is scheduled on behalf of a SLIP trap by the SLIP processor, debugging information is placed in the SDUMP 4K buffer (if the buffer is available). This buffer is pointed to by field CVTSDBF and contains:

Offset	Length	Content
0(0)	4	The characters 'TYPE' to identify the following field.
4(4)	4	RTM/SLIP processor environment indicator: X'00000001' - RTS X'00000002' - RT2 X'00000003' - RTM X'00000004' - PER
8(8)	4	The characters 'CPU' to identify the following field.
12(C)	4	Logical CPUID.
16(10)	4	The characters 'REGS' to identify the following field.
20(14)	64	Registers at the time of error or interrupt. (R0-R15)
84(54)	4	The characters 'PSW' to identify the following field.
88(58)	8	The PSW at the time of error or interrupt.
96(60)	4	The characters 'PASD' to identify the following field.
100(64)	2	The primary ASID at time of error or interruption.
102(66)	4	The characters 'SASD' to identify the following field.
106(6A)	2	The secondary ASID at time of error or interruption.
108(6C)	variable	The SDWA if offset 4 is 1 (RTS). The RTM2WA if offset 4 is 2 (RT2). The ASCB if offset 4 is 3 (RTM). The PER interrupt code if offset 4 is 4 (PER).

If ASIDLST is not specified with the SLIP trap, the following information describes which address space will be dumped depending on the environment of the RTM/SLIP processor.

RTS - the failing address space (field SDWAFMID) or the home address space is dumped.

RT2 - the failing address space (field RTM2FMID) or the home address space is dumped.

RTM - the master address space is dumped because the address space that is terminating cannot be used to take the dump. The summary dump information is collected in the home address space (of the issuer of the CALLRTM TYPE=MEMTERM macro), and the asynchronous dump runs later in the master address space.

PER - the home address space is dumped.

If a dump request for a failing address space fails (such as SDUMP returning a bad return code), then a second attempt is made to schedule a dump in the home address space. In the second attempt, no information is put in the SDUMP 4K buffer. If the second attempt fails, the message IEA412I is issued.

If a summary dump is requested, it may be suppressed under certain conditions. Refer to the topic "Placement of PER Traps" later in this section.

The entire dump may be suppressed if the operator has chosen the CHNGDUMP NODUMP option.

ACTION=WAIT Option

ACTION=WAIT indicates that the system is to stop and display the following information in message IEE8371I:

- SLIP id
- Type of trap and related information:
  - RTM1 - the SDWA address
  - RTM2 - the RTM2WA address
  - MEMTERM - the ASCB address
  - PER - the PER code and address
- PSW at the time of error or PER event
- Contents of control registers 3 and 4
- Contents of general purpose registers 0 through 15

The system remains stopped until the operator or system programmer enters "U" to request that the system continue processing.

If the system either cannot display the message or cannot be restarted following the display of the message, the system enters a restartable wait state (code X'01B').

Note that the time spent in the stopped state is attributed to the PER interrupt that caused the wait state. This makes it look as if a lot of time has been spent processing PER interrupts. Therefore, if the trap is intended to be used so that the system is restarted and the trap is to remain enabled, you may want to use PRCNTLIM=99 on the trap. Otherwise, the trap may be disabled after the system has been restarted. Use PRCNTLIM=99 with caution because limit checking is not performed while waiting for the trap to match.

Once in the wait state, you can use the debugging work area provided by SLIP to begin debugging a problem. This area is pointed to by PSA+X'40C' and contains:

Offset	Length	Content
0(0)	1	RTM/SLIP processor environment indicator: X'01' - RTS X'02' - RT2 X'03' - RTM X'04' - PER
1(1)	2	Logical CPUID.
3(3)	1	System mask (if offset 0 is 2).
4(4)	4	Pointer to registers at the time of error or interrupt. (R0-R15)
8(8)	4	Pointer to PSW at the time of error or interrupt.
12(C)	4	Pointer to SDWA if offset 0 is 1. Pointer to RTM2WA if offset 0 is 2. Pointer to ASCB being terminated if offset 0 is 3. Pointer to PER code if offset 0 is 4.
16(10)	4	Pointer to cross memory information (control registers 3 and 4) at the time of the error or interruption.

ACTION=TRACE Option

ACTION=TRACE indicates that a GTF SLIP trace record is written each time that the SLIP trap matches. GTF must be active and the GTF SLIP option specified in order for the record to be built and recorded. (Use the TRDATA keyword if you want to tailor the GTF trace records.)

The TRACE option is designed for those situations where a relatively small amount of data is required each time that a matching event occurs. Such a situation might occur when you are trying to determine the path through a module. But the TRACE option can handle a relatively large amount of data when required. Refer to the TRDATA keyword.

The registers at the time of the error or interrupt are used to resolve indirect addresses specified for the trace record fields. Under some circumstances, registers at the time of error may not be available. If this is the case, indirect addresses that contain a register value cannot be resolved and related fields cannot be collected. A zero length field is used in the user portion of a SLIP standard/user or SLIP user record to indicate that the requested field was not available. Also, a field is not available if it is paged out or if one of the pointers to it is paged out. When using indirect addresses, use the REGS keyword to get the contents of the registers used to resolve indirect addresses.

Another technique that is useful when using long indirect addresses is to request the same field twice if there is more than one path to the field. In this way, if a pointer is bad or is paged out in one path to the data, it may be available via the other path.

GTF Considerations: You may want to choose other GTF trace options in addition to SLIP to obtain other valuable diagnostic data. GTF options SYS or SYSM can be used to have GTF collect information similar to that collected by the normal system trace. Note that using the internal GTF trace instead of the external trace helps to reduce system overhead. Be sure to stop GTF after the traps which require the TRACE option are disabled or deleted.

ACTION=TRDUMP Option

ACTION=TRDUMP is a combination of the SVCD and TRACE options. While the trap remains enabled, a GTF SLIP trace record is written when the trap matches. When the trap is disabled (automatically by MATCHLIM or PRCNTLIM or via the SLIP MOD operand) or deleted (via the SLIP DEL operand), a dump is scheduled. When you use the SLIP MOD or DEL operand to disable or delete a trap that has the TRDUMP option specified, the dump does not contain diagnostic data in the SDUMP 4K buffer.

The default SDATA parameters are TRT, NOSQA, NOALLPSA, and NOSUM. These defaults are affected by the current CHNGDUMP command settings which may add to or override the requested dump options. The SDATA parameters can be changed by the SLIP user. Refer to "Dump Tailoring" later in this section.

When used in conjunction with the MATCHLIM keyword, the TRDUMP option can be useful in getting an idea of what events lead up to an error. For example, a problem is narrowed to a particular module. You could use a successful branch PER trap and the TRDUMP option. The trace records that are written could trace the fields that are critical to the operation of the module. An estimate of

the number of successful branches that would enable you to determine the path through the module could be specified on the MATCHLIM keyword in order to automatically disable the trap and initiate the dump.

The TRDUMP option can also be used to obtain GTF SLIP trace records without tracing to an external data set (and then using PRDMP to print the data set). When starting GTF, specify the number of 4K GTF trace buffers (on the BUF parameter) to be saved for a dump. When the dump is taken, the trace records are passed to the SVC dump routine and become a part of the dump.

ACTION=STRACE

ACTION=STRACE indicates that an SPER (SLIP/PER) type entry is written to the system trace table each time that the PER trap matches. SLIP issues the PTRACE macro to write the SPER entry. STRACE cannot be specified with any other ACTION option.

ACTION=STRACE helps to isolate a problem using PER events without the need to place the processor in a stopped state (such as ACTION=WAIT) when the PER event occurs.

Low-overhead SLIP processing occurs when no keywords other than the following are specified on the trap:

- For IF and SB events: ENABLE/DISABLE, ID, MATCHLIM, RANGE/NUCMOD/NUCEP/LPAMOD/LPAEP, and END.
- For SA events: ENABLE/DISABLE, ID, RANGE, MATCHLIM, and END.

When a keyword other than one of those listed is used on the trap, normal SLIP processing occurs.

ACTION=STDUMP

ACTION=STDUMP indicates that while the trap is enabled, an SPER (SLIP/PER) type entry (also known as a SLIP system trace record) is to be written to the system trace table each time that the PER trap matches. In this respect, ACTION=STDUMP is similar to ACTION=STRACE. However, when the trap is disabled (automatically by MATCHLIM or PRCNTLIM or by use of the SLIP MOD operand) or it is deleted (by use of the SLIP DEL operand), the system is to schedule an SVC dump to obtain the data recorded in the trace table.

The SVC dump includes the registers and PSW for the current task. Note that when you use the SLIP MOD operand to disable the trap or the SLIP DEL operand to delete the trap, the SVC dump does not contain diagnostic data in the SDUMP 4K buffer.

ACTION=STDUMP helps to isolate a problem using PER events without placing the processor in a stopped state (as with ACTION=WAIT) when the PER event occurs.

STDUMP cannot be specified with any other ACTION option.

To tailor the dump, use the ASIDLST, LIST, SUMLIST, and SDATA keywords. The default SDATA parameters are TRT, NOSQA, NOALLPSA, and NOSUM. These defaults are affected by the current CHNGDUMP command settings, which can add to or override the requested dump options. The SDATA parameters can be changed. For more information on SDATA and the other dump tailoring keywords, see the "Dump Tailoring" topic later in this section.

Low-overhead SLIP processing occurs when no keywords other than the following are specified on the trap:

- For IF and SB events: ENABLE/DISABLE, ID, MATCHLIM, RANGE/NUCMOD/NUCEP/LPAMOD/LPAEP, ASIDLST, LIST, SDATA, SUMLIST, and END.
- For SA events: ENABLE/DISABLE, ID, RANGE, MATCHLIM, ASIDLST, LIST, SDATA, SUMLIST, and END.

When a keyword other than one of those listed is used on the trap, normal SLIP processing occurs.

ACTION=RECORD

ACTION=RECORD indicates that a record is written to SYSl.LOGREC when a recovery routine is executing and the trap matches. You can specify RECORD with another option, such as ACTION=(SVCD,RECORD).

ACTION=NODUMP Option

ACTION=NODUMP indicates that SLIP is to set a flag in the RTM work area which is checked by the dump programs ABEND and SVC dump. If the bit is on, all dump requests are ignored. Because the bit is in the RTM work area, only dumps requested during processing of this error by RTM (requested by an FRR and/or an ESTAE) are suppressed. Should the error involve recursive entry into RTM, the bit setting is propagated to the next RTM work area.
ACTION=NODUMP applies only to non-PER traps for errors in the RTS and RT2 environments.

This action is useful for preventing dumps that may not be needed (for example, X37, etc.) because accompanying messages provide sufficient information. It can also be used to prevent duplicate dumps for known problems which have already been documented.

ACTION=NOSUP Option

ACTION=NOSUP indicates that dumping services is to bypass any suppression of the dump (such as suppression of the dump by dump analysis and elimination, DAE). This option ensures that the dump is taken for the event that was trapped.

ACTION=NOSVCD Option
ACTION=NOSYSA Option
ACTION=NOSYSM Option
ACTION=NOSYSU Option

When the SLIP trap matches, these options on the ACTION keyword indicate:

NOSVCD	Suppress SVC dumps
NOSYSA	Suppress SYSABEND dumps
NOSYSM	Suppress SYSMDUMP dumps
NOSYSU	Suppress SYSUDUMP dumps

These options allow you to selectively suppress dumps for particular errors when a message or other data provides all the information needed to debug the problem.

You can specify one or more of the options on the ACTION keyword of the SLIP trap. This allows you, for example, to suppress a SYSUDUMP dump for a particular abend (such as an "out of space" condition, code 80A) but permit other dumps.

Note that if, for a particular abend, you want to suppress all SVC and SYSUDUMP dumps but allow SYSABEND and SYSMDUMP dumps, you must use one trap (rather than two traps) to suppress the dumps. For example, use ACTION=(NOSVCD,NOSYSU). This is because the SLIP processor stops examining SLIP traps for a match when a matching trap has been found.

Also, note that the IEACMD00 member in SYSl.PARMLIB contains SLIP commands that suppress dumps for selected abends.

ACTION=IGNORE Option

ACTION=IGNORE indicates that the SLIP processor is to take the IGNORE action. The IGNORE action does not result in any specific action being taken but a match is indicated for the trap and other processing for the trap occurs normally (such as messages being issued, and processing for the MATCHLIM, PRCNTLIM, RECOVERY, and DEBUG options).

This option is generally used on a trap to prevent a different, and more general trap, from matching. (Note that for any event, the SLIP processor stops examining SLIP traps for a match condition when a matching trap is found.) Because SLIP traps are tested in last-in-first-out order, IGNORE traps used in this way must be entered after the more general non-IGNORE trap. Also, the traps should be specified in the disabled state to prevent the non-IGNORE trap from matching while the IGNORE traps are being specified. After the non-IGNORE and all related IGNORE traps have been set, they can be enabled in a last-in-first-out order by using the MOD operand of the SLIP command.

For PER traps, the IGNORE trap must be of the same type (IF, SA, or SB) as the non-IGNORE trap or it will not be tested. For IF and SB PER traps, IGNORE traps can be used to simulate multiple ranges for monitoring as shown in Example 14 in the following topic "Examples of Using the SLIP Command." This technique cannot be used for SA PER traps. The use of the IGNORE trap with a more general IF or SB PER trap does not prevent PER interrupts from occurring in the range specified on the IGNORE trap. You should consider this when you are selecting a percent limit value.

In general, there is no limit to the number of IGNORE traps that can be set to work in conjunction with a non-IGNORE trap. You should be aware that IGNORE traps are considered as independent traps, and the SLIP command

processor does not know when IGNORE traps are being used in conjunction with a non-IGNORE trap. For example, at the time a trap is being set, no checking is done between traps to ensure that the range to be ignored falls within the range specified on the non-IGNORE PER trap. Such checking is the responsibility of the user.

ACTION Keyword With RECOVERY Option (PER Traps Only)

Normal processing of a PER interrupt causes control to be returned to the next sequential instruction after the PER interrupt is processed. The RECOVERY keyword can be used to force recovery processing to be initiated after the PER interrupt has been processed.

The RECOVERY keyword is used to initiate recovery processing in those situations when an error is occurring, but the error is not being detected by the system or it is being detected too late for recovery routines to adequately handle the error situation. By initiating recovery processing via a SLIP trap, you have a way to use the error correction function that is built into MVS recovery routines.

To avoid unexpected results when using the RECOVERY keyword, you should be thoroughly familiar with the MVS recovery concepts and ensure that:

- Recovery is initiated at an appropriate point in the program.
- The recovery routine is designed to handle the error situation that exists at that point.

The RECOVERY action initially causes an 06F abend code to be generated.

Dump Tailoring

When ACTION=SVCD, or ACTION=STDUMP, or ACTION=TRDUMP is specified on a SLIP trap, the ASIDLST, SDATA, SUMLIST, and LIST keywords can be used to tailor the dump to the particular problem that is being trapped.

ASIDLST Keyword

The ASIDLST keyword is used to specify the address spaces that are to be dumped. Note that a specification of zero indicates the home address space (pointed to by PSAAOLD).

SDATA Keyword

The SDATA keyword is used to specify the system data areas that are to be dumped. If the default SDATA specification is used, the current system CHNGDUMP setting can affect (add to or override) the areas requested. The system CHNGDUMP settings do not affect (add to or override) the areas specified on SDATA except when CHNGDUMP has been set with the NODUMP option. In this case, the SLIP trap does not produce a dump when the trap matches.

When SDATA is specified, the areas specified to be dumped completely replace the default specification on the dump request. For example, on an ACTION=SVCD dump, if SDATA=(NOSQA) is specified, the NOSQA completely replaces the default SDATA specification of SQA, RGN, TRT, LPA,

CSA, NUC, ALLPSA, and SUM. The dump request of NOSQA is presented to SDUMP which merges it with its own defaults (SQA, SUM, and ALLPSA) and, in this case, produces a dump that contains only a summary dump and ALLPSAs.

Also, the SLIP command processor does not make any reasonability checks on the SDATA options specified. For example, SDATA=(SQA,NOSQA) is allowed even though the SQA and NOSQA options are contradictory. In this case, SQA would not be part of the dump produced.

If ENQ/DEQ control blocks are needed, use the GRSQ option, which obtains the appropriate information from the global resource serialization address space.

SUMLIST and LIST Keywords

The SUMLIST and LIST keywords are used to dump user-defined areas of storage. The storage areas are defined by specifying address space qualifiers followed by address pairs that specify the beginning and ending addresses of storage to be dumped. Address space qualifiers can be either explicit or symbolic. They specify the address space to which the address pairs refer. If a qualifier is not specified, the previous qualifier is used as the default. If the first address pair does not have a qualifier, CURRENT is used as the default. The beginning address must be less than or equal to the ending address. If the beginning address is greater, then the characters *A1>A2* are dumped instead of the requested area. Direct or indirect addresses can be used to specify the address pairs and can be mixed. Indirect addresses are resolved using the registers at the time of error or interrupt. If for any reason an indirect address cannot be resolved, the characters *RC=4* are dumped rather than the requested area.

The difference between SUMLIST and LIST is the point in time when the requested information is collected. SUMLIST collects information as a part of SDUMP summary dump processing. This is close to the time of error or PER interrupt and the information that is collected will probably be unchanged since the time of error or interrupt. (Note that storage areas must be paged in; and if not paged in, they are ignored by SDUMP.) LIST collects information when the scheduled dump is processed. (Note that storage areas are paged in if necessary.) This is some time after the error or interrupt and the information that is collected may have been changed since the time of error or interrupt.

Examples of Using the SLIP Command

The following examples briefly describe a system problem and show the SLIP command that can be used to match on a system event. The resulting dump or GTF SLIP record can then be used by the debugger to obtain diagnostic information in order to solve the problem.

Example 1: Match on Storage Alteration

Problem: An unknown program is incorrectly modifying location CD3010 in the LPA.

Action: The debugger sets the following SLIP trap:

```
SLIP SET,SA,ENABLE,ACTION=SVCD,
     RANGE=(CD3010),END
```


Result: When location CD3010 is altered, a SLIP match occurs and an SVC dump is scheduled. For this PER trap, MATCHLIM defaults to 1 which prevents a dump from being taken each time that location CD3010 is altered.

Example 2: Match on Storage Alteration

Problem: Same as Example 1 except location CD3010 is normally modified by JES2 (ASID=3) but should not be modified by any other program (in ASIDs 1, 2, 4, 5, 6, 7, 8, and 9).

Action: The debugger sets the following SLIP trap:

```
SLIP SET,SA,ENABLE,ACTION=SVCD,
     RANGE=(CD3010),ASID=(1,2,4,5,6,7,8,9),END
```

Result: When any program in an address space specified on ASID= alters location CD3010, a SLIP match occurs and an SVC dump is scheduled.

Example 3: Match on Storage Alteration

Problem: Same as Example 2 except there is an unknown number of address spaces (in addition to JES2 in ASID 3).

Action: An IGNORE trap is used in conjunction with the non-IGNORE PER trap. The debugger sets the following SLIP traps:

```
SLIP SET,SA,ID=TRP1,DISABLE,ACTION=SVCD,
     RANGE=(CD3010),END
SLIP SET,SA,ID=TRP2,DISABLE,ACTION=IGNORE,
     ASID=(3),END
```

Then issues the following SLIP commands:

```
SLIP MOD,ENABLE,ID=TRP2
SLIP MOD,ENABLE,ID=TRP1
```

Result: Any alterations to location CD3010 by JES2 (ASID=3) are ignored. Alterations to location CD3010 by programs in any other address space result in a SLIP match, and an SVC dump is scheduled. Note that the SLIP traps are inspected in a last-in-first-out (LIFO) order.

Example 4: Match on Storage Alteration

Problem: Location CD3010 contains an address that is normally modified by many programs. Intermittently, it is set to zero and causes an error.

Action: The debugger sets the following trap:

```
SLIP SET,SA,ENABLE,ACTION=SVCD,
     RANGE=(CD3010),DATA=(CD3010,EQ,00000000),END
```

Result: When location CD3010 is set to zero, a SLIP match occurs and an SVC dump is scheduled.

Example 5: Match on Instruction Fetch

Problem: An LPA routine is consistently abending and the debugger does not know if the routine is in error or it is being passed bad parameters. The LPA routine entry point is CD3100.

Action: The debugger sets the following SLIP trap:

```
SLIP SET,IF,ENABLE,ACTION=SVCD,
     RANGE=(CD3100),END
```

Result: The routine still abends. However, when entry is made to the routine at location CD3100, a SLIP match occurs and an SVC dump is scheduled. Information in the SVC dump allows the debugger to determine the validity of the parameter data.

Example 6: Match on Successful Branch

Problem: The debugger needs a "branch trace" of the instruction path taken through module MOD01 starting at offset X'108' through X'4FC' during the execution of the JOBX.

Action: GTF must be active with the GTF trace option SLIP and MODE=EXT specified in order to collect the GTF SLIP trace records in an external data set. The debugger sets the following SLIP trap:

```
SLIP SET,SB,ENABLE,ID=PER1,ACTION=TRACE,
     LPAMOD=(MOD01,108,4FC),JOBNAME=JOBX,
     MATCHLIM=20,END
```

Result: When 20 successful branch events have occurred during the execution of MOD01 when JOBX is in control, the trap is automatically disabled because MATCHLIM=20. The collected GTF SLIP standard trace records may be printed by the debugger via the EDIT function of the AMDPRDMP service aid.

Example 7: Match on Successful Branch

Problem: For Example 6, if the debugger wants to collect the GTF SLIP standard trace records in GTF's address space and obtain these records as a part of an SVC dump, then GTF must be active with GTF trace option SLIP and MODE=INT specified.

Action: The debugger sets the following SLIP trap:

```
SLIP SET,SB,ENABLE,ACTION=TRDUMP,
     LPAMOD=(MOD01,108,4FC),JOBNAME=JOBX,
     MATCHLIM=20,END
```

Result: When 20 successful branch events have occurred in the range of addresses from 108 to 4FC in MOD01 while JOBX is in control, an SVC dump is scheduled. The dump contains the GTF trace buffers (assuming the SDUMP TRT trace option is in effect). The number of GTF buffers dumped is determined by the BUF parameter on the GTF START command. Note that if the CHNGDUMP command has been invoked, then the latest areas defined by CHNGDUMP are dumped.

Example 8: Match on Instruction Fetch

Problem: The debugger needs to collect specific data (as a part of a summary dump) when the instruction at offset X'200' in MOD02 is executed in ASID 27.

Action: The debugger sets the following trap:

```
SLIP SET,IF,ENABLE,ACTION=SVCD,
     LPAMOD=(MOD02,200),ASID=(27),
     SUMLIST=(2R%%,2R%%+DCF,4R%%+28,4R%%+1C9),
     SDATA=SUMDUMP,MATCHLIM=1,END
```

Result: When the instruction at offset X'200' in module MOD02 is executed in ASID 27, the selected data (specified on SUMLIST) is gathered. Control will resume at the next sequential instruction after the point of interrupt. Additionally, the trap is disabled after a single match occurs because MATCHLIM = 1.

Note: To obtain the same data, the SUMLIST keyword could have been specified as:

```
     SUMLIST=(2R%%,+DCF,4R%%+28,+1C9),
```

Example 9: Match on Program Check

Problem: The debugger wishes to take a dump of the current private region when an 0C7 program check occurs during task mode processing in module MOD03.

Action: The debugger sets the following SLIP trap:

```
SLIP SET,ENABLE,ERRTYP=PROG,ACTION=SVCD,
     COMP=0C7,PVTMOD=MOD03,MODE=TCB,
     SDATA=RGN,END
```

Result: When the 0C7 program check occurs in TCB mode, the trap matches and an SVC dump is scheduled. Normal recovery processing then takes place.

Example 10: Match on Completion Code

Problem: The debugger wishes to take an SVC dump when a 806 completion code occurs for a job TEST99 when program PGM5 is in control.

Action: The debugger sets the following trap:

```
SLIP SET,ENABLE,COMP=806,ACTION=SVCD,
     JOBNAME=TEST99,JSPGM=PGM5,END
```

Result: When the job step that executes program PGM5 of job TEST99 is abended with a completion code of 806, the trap matches and an SVC dump is scheduled.

Example 11: Match on SVC Error

Problem: The debugger wishes to force the system into a wait state when an SVC error occurs in job TEST98 to take a stand-alone dump.

Action: The debugger sets the following trap:

```
SLIP SET,ENABLE,ERRTYP=SVCERR,ACTION=WAIT,
     JOBNAME=TEST98,END
```

Result: When job TEST98 encounters an SVC error, all processors in the system are put into the wait state. The operator may then initiate a Stand-alone dump (SADMP).

Example 12: Match on Data

Problem: The debugger wishes to force entry into recovery processing for LPA module MODX when MODX is processing a specified input (X'0105').

Action: The debugger sets the following SLIP trap:

```
SLIP SET,IF,ENABLE,ACTION=(RECOVERY),
     LPAMOD=(MODX,16),DATA=(1R%,EQ,0105),
     MATCHLIM=1,END
```

Result: When the input parameter pointed to by general purpose register 1 is equal to X'0105', the trap matches and the SLIP processor forces the recovery path to be taken. The trap is then disabled because MATCHLIM = 1.

Example 13: Match on Storage Alteration

Problem: The debugger wants to monitor a common storage location in a production system for alteration to X'F1F2F3F4'.

Action: To keep the trap overhead to a minimum, the debugger specifies the JOBNAME and ASID keywords. Also, because the trap is being set on a production system, the PRCNTLIM keyword is specified to prevent the trap from using more than 20% of the available system processing time. The debugger sets the following trap:

```
SLIP SET,SA,ENABLE,ACTION=SVCD,
     ASID=(7,9),JOBNAME=JOBX,
     RANGE=(CD3100,CD3103),
     DATA=(CD3100,EQ,F1F2F3F4),
     PRCNTLIM=20,END
```

Result: When the specified area is altered by JOBX in ASID 7 or 9 and the pattern of data is X'F1F2F3F4', then the trap matches and an SVC dump is scheduled. The processing time used by the PER interrupts is being monitored and if the time exceeds 20% of the available system time, the trap is disabled and the debugger is notified.

Example 14: Match on Instruction Fetch

Problem: The debugger wants to monitor a range of instruction addresses in LPA module MODX but ignore instructions that form an iterative loop within a subset of this range.

Action: The debugger sets the following traps:

```
SLIP SET,IF,DISABLE,ID=TRP1,ACTION=TRACE,
     LPAMOD=(MODX,110,1FB),JOBNAME=JOB1,
     TRDATA=(STD,REGS),MATCHLIM=500,END
SLIP SET,IF,DISABLED,ID=TRP2,ACTION=IGNORE,
     LPAMOD=(MODX,1C4,1D7),END
```

Then the debugger issues the following SLIP commands:

```
SLIP MOD,ENABLE,ID=TRP2
SLIP MOD,ENABLE,ID=TRP1
```

Result: PER interrupts are taken for each instruction that is executed within the range specified on trap TRP1, but those interrupts that fall within the range specified on trap TRP2 are ignored. Therefore, tracing occurs in MODX for those instructions that fall in the ranges of X'110' to X'1C3' and X'1D8' to X'1FB'. Note that the IGNORE trap must be defined after the non-IGNORE trap because the traps are processed for match tests in last-in-first-out order.

Note: The use of IGNORE traps with non-IGNORE PER traps effectively allows you to discard selected events that occur in one or more subsets of the monitored range. Be aware that PER interrupts occur within the ignored ranges and cause system degradation.

Example 15: Match on Instruction Fetch

Problem: The debugger wants to force the system into a wait state when JOB1 executes in address space 5, 7, or 10 within a given address range.

Action: The debugger sets the following trap:

```
SLIP SET,IF,ENABLE,ID=PNK1,ACTION=WAIT,MODE=(HOME),
     JOBNAME=JOB1,ASID=(5,7,10),RANGE=(E300,E000),END
```

Result: When job JOB1 starts in address space 5, 7, or 10, and an instruction is fetched within the specified address range, then the trap matches and the system is put in a wait state.

Note: Because MODE=HOME was specified, if job JOB1 starts in address space 5, issues a program call to address space 7, and then an instruction is fetched within the specified address range, the trap will not match.

Example of SLIP Command From TSO Terminal

The following example shows the use of the SLIP command from a TSO terminal and the prompting that can occur.

Problem: A debugger suspects that a module is being passed an improper parameter list that causes the module ISTAPC11 to abend during job APPLE1. By the time the abend occurs, all evidence of the cause has been eliminated by recovery processing. A history of the caller's parameter list can be obtained by using the SLIP IF PER function with ACTION=TRDUMP.

Action: The debugger issues the following commands:

```
tso user:   OPERATOR

system:     OPERATOR

tso user:   slip set,if,enable,id=per1,action=trdump,
            jobname=apple1,lpamod=(sstapc11,16),
            trdata=(std,regs,1r%,1r%+32),
            matchlim=5,end

system:     IEE736D SLIP ID=PER1,SSTAPC11 IS NOT IN THE LPA.
            ENTER KEYWORD, NULL LINE, OR 'CANCEL'

tso user:   lpamod=(istapc11,16)

system:     IEE727I SLIP TRAP ID=PER1 SET BUT GTF IS NOT ACTIVE

tso user:   send 'please start gtf with mode=int and trace=slip and notify me when done'

system:     OPER GTF IS ACTIVE

tso user:   send 'please start apple1'

system:     IEA992I SLIP TRAP ID=PER1 MATCHED

system:     IEA411I SLIP TRAP ID=PER1 DISABLED FOR MATCHLIM

system:     IEA911A COMPLETE DUMP ON SYS1.DUMP01 TSO USERID (D10XYZ1)

tso user:   send 'please stop gtf'
```

Result: The dump has been taken. The debugger may now want to copy the dump to another data set, clear the SYS1.DUMP01 data set, and obtain a hardcopy of the dump. This additional processing can be accomplished from the TSO terminal by using TSO commands to execute the print dump (AMDPRDMP) program.

Note: You may want to establish a cataloged procedure to invoke GTF. This procedure could specify all of the desired GTF options and keywords for using GTF with the SLIP command. Using such a procedure would allow you (when working from a TSO terminal) to pass only the name of the procedure to be started to the operator instead of the GTF parameters as shown in the example. This reduces the burden on the operator and is more effective when communicating with the operator.

Designing an Effective SLIP Trap

The design of a SLIP trap requires knowledge of the error conditions and what makes the error unique. An effective trap should catch only the intended error. To do this, the description should be as specific as possible.

Note that SLIP does not detect any events that occur while running in TRASMODE mode. (Control register 1 points to the segment table origin for an address space other than the current address space.)

The best way to design a trap is from a dump of the error. In the case of the NODUMP action, a dump should be available. In other cases, an approximate dump (one taken near the time of the error) or one without sufficient information to debug might be available.

It should be understood that for error events (non-PER traps), SLIP operates as a subroutine within the RTM. SLIP is called from either RTM1 or RTM2, depending on whether the error environment allowed FRR or only ESTAE recovery respectively. The level of RTM in control affects the data areas available. The calls to SLIP are prior to calls to any error recovery routine, therefore it is possible that the data areas contained in a dump may have been changed since SLIP examined them. This is especially true of the COMP keyword value. Many recovery routines change the abend completion code to make it more specific. For example, a system service that receives a bad address from a user parameter list will get an 0C4 which it converts to its own completion code meaning a bad parameter list.

Controlling SLIP Traps

This topic describes how the MATCHLIM and PRCNTLIM keywords are used to limit the system resources that a SLIP trap is allowed to use. Also, for PER traps, it includes some performance hints.

MATCHLIM Keyword

The MATCHLIM (match limit) keyword provides a way to set an upper limit on the number of times that a trap is allowed to match. This keyword can be specified to ensure that resources are not used unnecessarily (for example, using dump data sets when ACTION=SVCD is chosen) or to remove the overhead associated with a PER trap as soon as possible.

The MATCHLIM keyword should always be specified for enabled traps that are unattended in order to avoid undesirable results, such as filling several dump data sets for multiple occurrences of the same error.

With a few exceptions, if MATCHLIM is not specified on a trap, the number of times that the match can occur is not limited. For the following exceptions, the system disables a trap to limit the number of matches that can occur for that trap:

- After one match occurs on a PER trap that specifies ACTION=SVCD.
- After 50 matches occur on a PER trap that specifies:
  - Either ACTION=STDUMP or ACTION=STRACE, and
  - Only those keywords that result in low-overhead processing

  The descriptions for ACTION=STDUMP and ACTION=STRACE indicate which keywords result in low-overhead processing.

If a MATCHLIM value has been specified for a trap, you can tell if the trap is matching and approaching the limit set by displaying the trap via the DISPLAY command.

When the MATCHLIM keyword is used on an IGNORE type trap, it can provide the effect of ignoring a specified number of events before an action is taken by the associated non-IGNORE trap.

PRCNTLIM Keyword

The PRCNTLIM (percent limit) keyword provides a way to set a limit on the amount of system time that is committed to processing on behalf of a PER trap. The percentage of time that is computed is based on the amount of time spent processing PER interrupts and space switch interrupts (as compared to the amount of time elapsed since the first PER interrupt for the trap). The percentage that is computed is related only to software processing and does not include any PER hardware processing. The percentage is computed each time that a PER interrupt is processed and is compared to the percent limit specified on the PRCNTLIM keyword.

Percent limit checking is not performed for the first 33 seconds (approximately) after the first interrupt is taken for the trap. This avoids a high initial percentage which might disable the trap immediately.

The accuracy of the percent limit calculation is affected by the instructions that are executed on behalf of the PER interrupt and space switch interrupt but are not included in the calculation. These instructions include:

- instructions that are executed before the first timestamp is taken. (For example, instructions in the program check first level interrupt handler.)
- instructions that are executed after the last timestamp is taken. (For example, the percent limit calculation itself.)
- instructions that are indirectly related to the PER interrupt. (For example, instructions used for PER trap messages.)

Because these instructions are not accounted for in the percent limit calculation, the accuracy of the calculation may vary between different traps. For example, if there are many very explicit traps to be checked and one of them takes an action, the large number of instructions executed between the timestamps taken will result in a fairly accurate percentage calculation even though some instructions were not accounted for in the calculation. Conversely, if there is one very simple trap that does not match, the instructions that are not accounted for in the calculation represent a large portion of the instructions executed and their exclusion makes the percentage calculation more inaccurate.

Also, the percentage calculation is affected because the calculated percentage is truncated to an integer.

If you have any doubt as to whether or not a PER trap will work properly in a system, a conservative value (such as the default PRCNTLIM = 10) should be chosen. Thus, if the trap should consume large amounts of processing, it will be quickly disabled. After approximately 33 seconds, you can display the PER trap (via the DISPLAY command) and obtain the current system percent utilization by the trap.

Specifying PRCNTLIM = 99 indicates that percent limit checking is not to be performed. It is intended for non-critical environments (such as testing) and certain special situations (see the ACTION = WAIT option). In general, all non-IGNORE PER traps should have a reasonable value specified for percent limit.

Performance Hints For PER Traps

For PER traps, consider the following ways to limit PER monitoring and minimize performance degradation:

- Choose values for the RANGE and LPAMOD keywords that reduce the range of storage monitored by the PER hardware. This action avoids unnecessary PER interrupt processing.
- For non-IGNORE PER traps:
  - Specify the JOBNAME event qualifier keyword to limit PER monitoring to the address spaces in which the specified job runs. If the unit of work executes in an address space other than the one in which it was dispatched, PER monitoring will also be active in that address space for that particular job.
  - Specify the ASID event qualifier keyword to limit PER monitoring to the address space(s) identified on the keyword.
  - Specify HOME on the MODE keyword (MODE=HOME) to indicate that PER monitoring is to be active only when the unit of work executes in the address space in which it was dispatched.

Placement of PER Traps

The PER support provided by SLIP is designed to be non-disruptive at the possible expense of not collecting data or performing a user requested action. Several aspects of this non-disruptive characteristic are discussed in this topic:

Certain parts of the system cannot tolerate PER interrupts. For those parts, the PSW PER bit is set off to prevent interrupts. Most notably, the PER bit is set off in the program check, machine check and restart new PSWs. PER remains off in such critical paths until processing reaches a point where a PER interrupt is considered "safe." For example, if a SLIP IF PER trap is set on for the first instruction in the program check FLIH, no PER interrupts would occur and the trap would not match. Note, however, that you are not prevented from setting such a trap.

Some PER interrupts that occur are not always processed by the SLIP processor. The SLIP processor ignores (that is, does not process) PER interrupts if the interrupt:

- occurred while DAT was off (PER support for SLIP applies only to virtual addresses).
- is redundant. (Refer to *Principles of Operation* for a description of redundant PER interrupts.)
- occurred while an enabled non-IGNORE PER trap does not exist. Note that this implies that PER interrupts caused by a non-SLIP tool which has set up the PER control registers is ignored by SLIP and the PER bit is turned off by SLIP in the resume PSW before returning to the program check FLIH.

Because the SLIP processor uses certain system services, SLIP is sensitive to the recursive use of those services where a recursive entry could cause an error. A recursion is the result of placing a PER trap in a function and then causing SLIP to use that function. For example, suppose a SLIP trap is placed in GTF entry code and the action specified is TRACE. If the trap matched and SLIP tried to take the action, a GTF trace record would not be written because of the recursive checks within GTF. A similar situation exists with other tracing actions, dump actions, and wait. In general, recursions result in the action not being taken. Such recursions can be avoided by the appropriate choice of another SLIP action. Note that you are not prevented from setting a trap that can cause a recursion.

PER Monitoring and Checkpoint/Restart

For PER monitoring, specific PER support is not included in the checkpoint/restart function. Therefore, two possibilities exist for a checkpointed program.

Case 1. A program is running in an address space that has not been selected for PER monitoring and the program is checkpointed.

Case 2. A program is running in an address space that has been selected for PER monitoring and the program is checkpointed.

These two cases could result in one of three conditions when the checkpointed program is restarted.

- In Case 1, if the program is restarted in an address space that has been selected for PER monitoring, the restarted program is not monitored.
- In Case 2, if the program is restarted in an address space that has not been selected for PER monitoring, but other address spaces are being monitored, unwanted PER interrupts may occur (depending on the PER control register settings). If unwanted PER interrupts occur in the restarted program, the PSW PER bit is turned off in the restarted program. This may eventually remove all possible degradation due to the unwanted PER interrupts from the restarted program.
- In Case 2, if the program is restarted and PER monitoring is not active in the system (that is, control register nine is zero), the system may suffer some degradation due to the PSW PER bit being on in the restarted program as the restarted program is running.

SLIP Command Keyword Summary

Figure 2-11 is a summary of the SLIP command keywords. The keywords are shown across the top of the figure and are grouped by the function that they perform. For each keyword, a brief description is given for the use of the keyword, the valid options, required and default options, restrictions, comments and other information.

	SLIP Control Keywords			Trap Type Keywords		
	SET	DEL	MOD	IF	SB	SA
PER	Set a PER trap.	Delete one or more PER traps.	Enable or disable one or more PER traps.	Set an instruction fetch PER trap.	Set a successful branch PER trap.	Set a storage alteration PER trap.
Non-PER	Set a non-PER trap.	Delete one or more non-PER traps.	Enable or disable one or more non-PER traps.	---	---	---
Valid Options	ACTION ADDRESS ASID ASIDLST ASIDSA COMP DATA DEBUG DISABLE ENABLE END ERRTYP IF ID JOBNAME JSPGM LIST LPAEP LPAMOD MATCHLIM MODE NUCEP NUCMOD PRCNTLIM PVTEP PVTMOD RANGE REASON RBLEVEL SA SB SDATA SUMLIST TRDATA	ALL ID	ALL DISABLE ENABLE ID	ACTION ASID ASIDLST DATA DEBUG DISABLE ENABLE END ID JOBNAME JSPGM LIST LPAEP LPAMOD MATCHLIM MODE NUCEP NUCMOD PRCNTLIM PVTEP PVTMOD RANGE SDATA SUMLIST TRDATA	Same as IF.	ACTION ADDRESS ASID ASIDLST ASIDSA DATA DEBUG DISABLE ENABLE END ID JOBNAME JSPGM LIST LPAEP LPAMOD MATCHLIM MODE NUCEP NUCMOD PRCNTLIM PVTEP PVTMOD RANGE SDATA SUMLIST TRDATA
Required	END	ID or ALL	ID or ALL, and ENABLE or DISABLE	RANGE or LPAMOD or NUCMOD or PVTMOD or LPAEP,PVTEP,NUCEP	RANGE or LPAMOD or NUCMOD or PVTMOD or LPAEP,PVTEP, NUCEP	RANGE (unless ACTION=IGNORE is specified)
Default	ENABLE,ACTION=SVCD, and RBLEVEL=ERROR	---	---	---	---	---
Restrictions	Must follow SLIP.	Must follow SLIP.	Must follow SLIP.	Must follow SET.	Must follow SET.	Must follow SET.
Comments	---	---	---	Qualify with ASID and/or JOBNAME to minimize performance degradation.	Same as IF.	Same as IF.
Minimum Value	---	---	---	---	---	---
Maximum Value	---	---	---	---	---	---
Special Value	---	ALL indicates all SLIP traps.	ALL indicates all SLIP traps.	---	---	---
Abbreviation	---	---	---	---	---	---
Example	SLIP SET,...	SLIP DEL,...	SLIP MOD,...	SLIP SET, IF,...	SLIP SET, SB,...	SLIP SET, SA,...

Figure 2-11 (Part 1 of 8). SLIP Command Summary

	Trap Control Keywords		Specialized Keywords		
	ENABLE	DISABLE	ID	END	DEBUG
PER	Enable a PER trap.	Disable a PER trap.	Provide an identifier for a PER trap.	Indicates the trap definition is complete.	Causes the GTF SLIP DEBUG record to be written when the trap is checked.
Non-PER	Enable a non-PER trap.	Disable a non-PER trap.	Provide an identifier for a non-PER trap.	Indicates the trap definition is complete.	Causes the GTF SLIP DEBUG record to be written when the trap is checked.
Valid Options	---	---	Four-character identifier.	---	---
Required	---	---	---	---	---
Default	---	---	System supplied.	---	---
Restrictions	---	---	---	Specify at end of the command.	---
Comments	Only one non-IGNORE PER trap may be enabled.	---	Supplied by system if not specified on SET.	Use with SET.	GTF must be active and with the SLIP option chosen.
Minimum Value	---	---	---	---	---
Maximum Value	---	---	---	---	---
Special Value	---	---	For SLIP MOD, * in any character position means don't care, and **** means ALL.	---	---
Abbreviation	EN	D	---	E	---
Example	SLIP..., EN,...	SLIP..., D,...	ID=TRP1	SLIP SET,..., END	DEBUG

Figure 2-11 (Part 2 of 8). SLIP Command Summary

	Trap Event Qualifier Keywords				
	ADDRESS	**ASID**	**ASIDSA**	**COMP**	**DATA**
PER	For SA, specifies the address range of the instruction that must cause the storage alteration.	Specifies the ASIDs (address spaces) to be monitored.	Specifies the ASIDs (address spaces) in which the storage being altered resides.	---	Specifies condition of storage or registers.
Non-PER	Specifies the address range that must be in control when the error occurs.	Specifies the ASIDs (address spaces) that must be in control when the error occurs.	---	Specifies a user or system completion code.	Specifies condition of storage or registers.
Valid Options	Start address End address	---	---	Udddd - user code hhh - system code	Direct address Register Indirect address Operator - EQ, NE, LT, GT, NG, NL
Required	---	---	---	---	---
Default	---	---	---	---	---
Restrictions	For PER traps, SA only. Start address must be less than or equal to end address.	Maximum of 16 ASIDs can be specified.	---	---	---
Comments	Specify virtual addresses in hexadecimal.	Limits PER monitoring to the ASIDs specified.	---	Specify user code as four decimal digits. Specify system code as three hexadecimal digits.	---
Minimum Value	0	0	---	0	---
Maximum Value	00FFFFFF	Installation defined (ASVTMAXU).	Maximum of 16 ASIDs can be specified.	U9999 or FFF.	---
Special Value	---	---	---	x means don't care.	---
Abbreviation	AD	AS	ASA	C	DA
Example	AD=(200,300)	AS=(1,6,12A)	ASA=(5,S,3,HASID)	C=06F	DA=(2R,EQ,00)

Figure 2-11 (Part 3 of 8). SLIP Command Summary

	Trap Event Qualifier Keywords				
	ERRTYP	**JOBNAME**	**JSPGM**	**LPAEP**	**LPAMOD**
PER	---	Specifies the job to be monitored.	Specifies the job step program name that must be in control.	Same as LPAMOD except checking is done from entry point or alias.	For IF and SB, defines the range to be monitored. For SA, specifies the LPA module that must cause the storage alteration.
Non-PER	Specifies system-detected error type.	Specifies the job that must be in control when the error occurs.	Specifies the job step program name that must be in control when the error occurs.	Same as LPAMOD except checking is done from entry point or alias.	Specifies the LPA module that must be in control when the error occurs.
Valid Options	ABEND DAT MACH MEMTERM PGIO PROG REST SVCERR	Any valid job name, TSO ID, or started task name.	Any valid job step program name.	Entry point name or alias Start displacement End displacement	Name Start displacement End displacement
Required	---	---	---	Entry point name or alias	Name
Default	---	---	---	From entry point or alias to end of module	Entire module
Restrictions	---	---	---	---	---
Comments	Logical "or" if more than one error is specified.	Limits PER monitoring to the ASID of the job.	---	Specify displacement in hexadecimal.	Specify displacement in hexadecimal.
Minimum Value	---	---	---	---	---
Maximum Value	---	---	---	---	---
Special Value	ALL indicates all valid options.	---	---	* as last character of entry point name indicates X'C0'.	* as last character of name indicates X'C0'.
Abbreviation	ER	J	JS	---	L
Example	ER=(DAT)	J=MY JOB	JS=IFOX00	---	L=MYMOD2

Figure 2-11 (Part 4 of 8). SLIP Command Summary

	Trap Event Qualifier Keywords				
	MODE	**NUCEP**	**NUCMOD**	**PVTEP**	**PVTMOD**
PER	Specifies system mode.	Same as NUCMOD.	For IF and SB, defines the range to be monitored. For SA, specifies the nucleus module that must cause the storage alteration.	Same as PVTMOD except checking is done from entry point or alias.	For IF and SB, defines the range to be monitored. For SA, specifies the private module that must cause storage alteration.
Non-PER	Specifies system mode	Same as NUCMOD.	Specifies the nucleus module that must be in control when the error occurs.	Same as PVTMOD except checking is done from entry point or alias.	Specifies the private module that must be in control when the error occurs.
Valid Options	ALL TCB DIS TYP1 GLOC --- GLOCSD with GLOCSP ANY or HOME EVERY LLOC LOCK PKEY PP RECV SKEY SRB SUPER SUPR	Entry point name Start displacement End displacement	Name Start displacement End displacement	Entry point name or alias Start displacement End displacement	Name Start displacement End displacement
Required	---	Entry point name	Name	Entry point name or alias	Name
Default	ANY (if EVERY is not specified with an option).	From entry point to end of module	Entire module	From entry point or alias to end of module.	Entire module
Restrictions	RECV cannot be specified for a PER trap. If ALL is specified for a PER trap, RECV is not included.	---	---	---	---
Comments	HOME is not included when ALL is specified. If HOME is specified, the unit of work must be executing in HOME even if ANY is also specified.	Specify displacement in hexadecimal.	Specify displacement in hexadecimal.	Specify displacement in hexadecimal.	Specify displacement in hexadecimal.
Minimum Value	---	---	---	---	---
Maximum Value	---	---	---	---	---
Special Value	EVERY - logical "and" ANY - logical "or" ALL - all modes	* as last character of entry point name indicates X'C0'.	* as last character of name indicates X'C0'.	* as last character of entry point name indicates X'C0'.	* as last character of name indicates X'C0'.
Abbreviation	M	---	N	---	P
Example	M=(GLOC)	---	N=IEAVTRTS	---	P=MYMOD1

Figure 2-11 (Part 5 of 8). SLIP Command Summary

	Trap Event Qualifier Keywords		
	RANGE	**RBLEVEL**	**REASON**
PER	Specifies the range of addresses to be monitored.	---	---
Non-PER	---	Specifies the source RB for the registers and PSW. at the time of error.	Specifies a user or system reason code.
Valid Option	Start Address End Address	ERROR NOTSVRB PREVIOUS	---
Required	Start Address	---	---
Default	---	ERROR	---
Restrictions	For SA PER traps, cannot be specified with ACTION=IGNORE.	---	COMP must also be specified.
Comments	Specify virtual addresses in hexadecimal. Start address may be greater than, less than, or equal to end address.	Applies to unlocked task mode errors only.	Specify one to eight hexadecimal digits.
Minimum Value	0	---	0
Maximum Value	00FFFFFF	---	FFFFFFFF
Special Value	---	---	x means don't care.
Abbreviation	RA	RB	RE
Example	RA=(600,700)	RB=NOTSVRB	RE=8

Figure 2-11 (Part 6 of 8). SLIP Command Summary

	Action Keyword	Dump Tailoring Keywords			
	ACTION	ASIDLST	LIST	SDATA	SUMLIST
PER	Specifies the action to be taken when the trap matches.	Specifies the ASIDs (address spaces) to be dumped.	Specifies a list of storage ranges to be included in the dump.	Specifies the system information to be included in the dump.	Specifies a list of storage ranges to be included in the summary dump.
Non-PER	Specifies the action to be taken when the trap matches.	Specifies the ASIDs (address spaces) to be dumped.	Specifies a list of storage ranges to be included in the dump.	Specifies the system information to be included in the dump.	Specifies a list of storage ranges to be included in the summary dump.
Valid Options	IGNORE NODUMP NOSUP NOSVCD NOSYSA NOSYSM NOSYSU RECORD RECOVERY STDUMP STRACE SVCD TRACE TRDUMP WAIT	Hexadecimal ASID value.	Direct address pairs. Indirect address pairs.	ALLNUC ALLPSA CSA GRSQ LPA LSQA NOALLPSA/NOALL NOSQA NOSUMDUMP/NOSUM NUC PSA RGN SQA SUMDUMP/SUM SWA TRT	Direct address pairs. Indirect address pairs.
Required	---	---	---	---	---
Default	SVCD	---	---	For ACTION=SVCD, SDATA=(ALLPSA,CSA, LPA,NUC,RGN,SQA, SUMDUMP,TRT). For ACTION=TRDUMP, SDATA=TRT.	---
Restrictions	WAIT - not available to TSO user. RECOVERY - PER traps only. STRACE or STDUMP - PER traps only; cannot be specified with another option. RECORD - non-PER traps only.	Specify a maximum of 15 ASIDs. ACTION=SVCD or ACTION=STDUMP or ACTION=TRDUMP must be specified or defaulted.	Start storage range address must be less than or equal to end storage range address. ACTION=SVCD or ACTION=STDUMP or ACTION=TRDUMP must be specified or defaulted.	ACTION=SVCD or ACTION=STDUMP or ACTION=TRDUMP must be specified or defaulted.	Start storage range address must be less than or equal to end storage range address. SUMDUMP/SUM must be specified or defaulted. ACTION=SVCD or ACTION=STDUMP or ACTION=TRDUMP must be specified or defaulted.
Comments	Choose only one option - except RECOVERY and RECORD, which can be specified with another option; and NOSVCD, NOSYSA, NOSYSM, and NOSYSU, which can be specified in any combination.	---	---	CHNGDUMP settings are overridden when SDATA is specified (except for NODUMP option).	---
Minimum Value	---	0	---	---	---
Maximum Value	---	Installation-defined (ASVTMAXU).	---	---	---
Special Value	---	0 - indicates the current ASID.	---	---	---
Abbreviation	A	AL	LS	SD	SL
Example	ACTION=(TRACE)	ASIDLST=(0,1,LLOC,P)	LIST=(H.200,300, 2R%%,+20)	SDATA=(SQA,NUC)	SUMLIST=(S.400,500, 2R%%,+20)

Figure 2-11 (Part 7 of 8). SLIP Command Summary

	Trace Tailoring	Automatic Control Keywords	
	TRDATA	**MATCHLIM**	**PRCNTLIM**
PER	Specifies the type and content of the GTF records to be collected.	Specifies the maximum number of times that the trap can match.	Specifies the maximum amount of system time that PER processing is allowed.
Non-PER	Specifies the type and content of the GTF records to be collected.	Specifies the maximum number of times that the trap can match.	---
Valid Options	Direct address pairs. Indirect address pairs. REGS STD	Decimal integer.	Decimal integer.
Required	---	---	---
Default	STD	1 - for PER traps with ACTION=SVCD specified or defaulted. 50 - for PER traps with ACTION=STRACE or ACTION=STDUMP and only other keywords that result in low-overhead processing.	10
Restrictions	ACTION=TRACE or ACTION=TRDUMP must be specified.	---	Applies to non-IGNORE PER traps only.
Comments	GTF record is 256 bytes maximum. STD uses 120 bytes. REGS uses 65 bytes. GTF must be active with the SLIP option.	---	---
Minimum Value	---	1	1
Maximum Value	---	65535	99
Special Value	STD - standard record information. REGS - registers.	---	99 - no checking is performed.
Abbreviation	TD	ML	PL
Example	TRDATA=(STD,REGS,20,30,2R%,+40)	MATCHLIM=40	PRCNTLIM=20

Figure 2-11 (Part 8 of 8). SLIP Command Summary

Section 3. Diagnostic Materials Approach

This section provides guidelines for analyzing storage dumps to find which data areas were affected by the error and to isolate internal symptoms of the problem.

The three chapters in this section are:

Stand-Alone Dumps

The stand-alone dump provides the problem solver with a larger quantity of data than system-initiated dumps because it contains areas that belong to the entire operating system rather than just a single address space or component. One of the major problems for the analyst is finding the important data for his problem and then isolating the problem area. Once this isolation is achieved, the debugger uses unique system/component techniques to gain further insight into the exact cause of the problem.

This chapter points out where to look in a stand-alone dump to determine various problem symptoms. The general approach is to analyze a stand-alone dump to find out what the system is doing (or not doing). Important areas will be described and possible reasons for their current state/contents will be explained. The analysis starts at the global system level and, by gathering data and gaining an understanding of the environment, works down to the address space and task level.

The experienced problem solver realizes that under certain conditions it may be necessary or advantageous to omit interpreting various areas. For example, if during system operation you observe that a given segment of the system (such as ACF/VTAM) is not functioning (other areas appear okay - jobs are executing, SYSIN/SYSOUT is appearing, etc.), you may decide to take a stand-alone dump. In this case, the current state of the system is probably not important. You would not be interested in current PSW, registers, etc.; you would be interested only in the address spaces that are using ACF/VTAM and the state of the TP network. The dump is not taken for a problem that is "active" now, but to give you data with which to determine a problem that appears to have originated some time ago. The point is that knowing why the dump was taken will often govern which, if any, of the stand-alone dump areas are of significance for a given problem.

See "Using IPCS to Analyze Dumps" on page 2-98 for a way to use the IPCS-provided CLISTs to gather useful information when starting to diagnose a problem.

Information contained in the chapter on "Waits" in Section 4 can be used as a supplement to the following discussions. (Also, a step-by-step approach to analyzing a stand-alone dump is contained in Appendix A of this manual.)

To analyze a stand-alone dump, you should always ask the following questions:

1. *Why was the dump taken?*

   Console sheets/logs are very important in stand-alone dump analysis. They are often the key to solving "enabled wait" situations and may present valuable information about system activity prior to taking the dump. Messages concerning I/O errors, condition code = 3, SVC dumps, abnormal job terminations, device mounts, etc. should be thoroughly investigated to determine if they could possibly contribute to the problem you are tracking.

   The dump title gives an indication of the problem's external signs or, possibly, a specific situation that must be investigated, such as "ACF/VTAM NOT FUNCTIONING."

2. *What is the current state of the system?*

   Examine the available global data areas to determine what the system is currently doing. The "Global System Analysis" chapter in Section 4 aids in this process. Remember that at this point, you are gathering information and trying to understand the system environment in order to isolate the *internal* symptom; you are not ready yet to debug.

3. *Has your global analysis isolated the problem to an internal symptom?*

   If so, refer to the discussion of that symptom in Section 4 of this manual.

4. *What previous errors have occurred within the system; could they possibly have any affect on your current problem?*

   The interpretation of SYS1.LOGREC and the in-storage LOGREC buffers are most important in determining error history. See the chapter on "Use of Recovery Work Areas" in Section 2.

5. *What is the recent system activity?*

   The chapter on "MVS Trace Analysis" in Section 2 aids in trace table interpretation.

6. *What is the work status within the system?*

   Your objective is to determine if the system has for some reason not completed all scheduled work. Determining what that work is and why it is not progressing can provide insight into the problem as well as answer some questions that may have arisen during an earlier analysis. Understanding the major control block structure and work queue status should aid in determining the possible source of the error. Refer to the discussion of "Work Queues and Address Space Status" in the "Global System Analysis" chapter of Section 2.

At this point, you should have gathered enough data to have a definition of the internal problem symptom. You should also have considerable information about the system's state, error history, and job status. You should refer to the appropriate chapter in Section 4 "Symptom Analysis Approach."

SVC Dumps

SVC dumps (invoked by the SDUMP macro) are usually taken as a result of an entry into a functional recovery routine (FRR) or ESTAE routine. The component recovery routine specifies the addresses that will be dumped.

"Appendix B: SDUMP Title Directory" lists the titles of SVC dumps initiated by system components and provides diagnostic information for the modules that issue the SDUMP macro.

SDUMP options SQA, ALLPSA, and SUMDUMP are the defaults for all requests. The SUMDUMP option of SDUMP provides a summary dump within an SVC dump. There is a twofold purpose for this. First, since dump requests from disabled, locked, or SRB-mode routines cannot be handled by SVC dump immediately, system activity destroys much useful diagnostic data. With SUMDUMP, copies of selected data areas are saved at the time of the request and then included in the SVC dump when it is taken. Second, SUMDUMP provides a means of dumping many predefined data areas simply by specifying one option.

You can use the IPCS subcommand VERBEXIT SUMDUMP to view the data areas that are saved in the summary dump, and you can use the AMDPRDMP control statement SUMDUMP to print the summary dump data. This summary dump data is not mixed with the SVC dump because in most cases it is chronologically out of step. Instead, each data area selected in the summary dump is separately formatted and identified.

For information on print dump statements needed to print the summary dump, and multiple address-space output from SVC dump, see *Service Aids.*

The RTM2WA pointed to by the TCB upon whose behalf the dump is being taken is the most valid system status indicator available. The dump task is usually the current task; the task upon whose behalf the dump is being taken will contain a completion code in the TCB completion code field. It is possible for the ESTAE routine to issue SVC D itself, in which case the current task is also the failing task.

Because of MVS recovery (retry and percolation), the SVC dump may be only part of the documentation at the problem solver's disposal. The problem solver should attempt to obtain:

1. The system log for the time the dump was taken to ascertain if:

   - Any other SVC dumps were taken before or after the one he is investigating.

   - Any task subsequently abended. If so, a system dump that displays other areas of storage that have meaningful data may be available.

2. The LOGREC formatted listing for the time immediately preceding the time of the SVC dump. If the component analysis procedure fails to determine the cause of the problem, analyze the dump as you would a stand-alone dump. Keep in mind that the information obtained via the CPUDATA option on AMDPRDMP is probably meaningless. Refer to the "Global System

Analysis" chapter in Section 2 for information on how to do a task analysis of available address-space-related control blocks.

Keep in mind that the system has detected the error and has attempted recovery, at least on a system basis. Therefore, there will be a good indication of the type (internal symptom) of error (loop, abend, problem check, etc.) that caused the problem. (See Section 4, "Symptom Analysis Approach.")

How to Change the Contents of an SVC Dump Issued by an Individual Recovery Routine

At times, SVC dump contents are not sufficient to solve a problem. The most convenient way to change the contents is the CHNGDUMP command. It can be used to establish system options to be added to the options on each SDUMP request, or to totally override the SDUMP options. See "Using the CHNGDUMP Command" in Section 2. If you do not want to affect all SVC dumps or if storage lists are involved, you may want to change the parameter list in a particular ESTAE exit instead.

You can usually find the name of the recovery routine by looking at the user data (or title) on the SVC dump printout. If not, search the ESTAE's PRB for the virtual address of the SDUMP SVC instruction.

The following description of SDUMP's parameter list can help you decide which bits will provide the data you want. The SDUMP macro expansion generates the parameter list and puts the address of the list in register 1.

SDUMP Parameter List

Offset		
0	1...	user-supplied DCB=
	.1..	BUFFER=YES
	..1.	user-specified STORAGE= or LIST=
	...1	user-specified HDR= or HDRAD=
	.... 1...	user-specified ECB=
	.... .1..	user-specified ASID=
	.... ..1.	QUIESCE=YES
	1	BRANCH=YES
1	1...	indicates SDUMP (as opposed to SNAP)
	.1..	indicates a SYSMDUMP request
	..1.	indicates the MVS/SP level of the SDUMP macro expansion
	...1	user-specified ASIDLIST=
	.... 1...	user-specified SUMLIST=
	.... .1..	ignore the change dump options (used by SLIP)
	.... ..1.	dump came from TSO user
	1	parameter list applies for MVS/SP
2		SDATA options
	1...	ALLPSA
	.1..	PSA
	..1.	NUC
	...1	SQA
	.... 1...	LSQA
	.... .1..	RGN
	.... ..1.	LPA
	1	TRT (system trace table, master trace table, and GTF buffers if GTF is active)

3		more SDATA options
	1...	CSA
	.1..	SWA
	..1.	SUMDUMP
	...1	NOSUMDUMP
	.... 1...	NOALLPSA
	.... .1..	NOSQA
	.... ..1.	ALLNUC
	other	reserved

Offset		
4		DCB address
8		address of storage list (STORAGE, LIST, LISTA)
C		address of header record (HDR, HDRAD)
10		address of ECB
14		caller's ASID
16		target ASID of scheduled dump
18		address of ASID list (ASIDLST)
1C		address of summary dump storage list (SUMLIST/SUMLSTA)
20		address of SYSMDUMP 4K SQA area (or TSO USERID if the DUMP command was from TSO)
24		address of SYSMDUMP CSA work area
28		SDUMP control flags
	1...	LISTA option specified
	.1..	SUMLSTA option specified
	..1.	SUSPEND=YES option specified
	...1	SUBPLST option specified
	.... 1...	KEYLIST option specified
	.... .1..	SYSMDUMP for authorized user
	others	reserved
29		new release control flags
	1...	parameter list is 64 bytes
	others	reserved
2A		TYPE=parameter options
	1...	TYPE=XMEM specified
	.1..	TYPE=XMEME specified
	..1.	TYPE=NOLOCAL specified
	others	reserved
2B		reserved
2C		SDATA options for component exits
	1...	GRSQ - global resource serialization exit
	others	reserved
2D		reserved
2E		reserved
2F		reserved
30		address of subpool list
34		address of key list
38		pointer to SLIP reg/PSW area
3C		pointer to SYSMDUMP symptom area
40		reserved

SYSABENDs, SYSMDUMPs, and SYSUDUMPs

SYSABENDs, SYSMDUMPs, and SYSUDUMPs are produced by the system when a job abnormally terminates and a SYSABEND, SYSMDUMP, or SYSUDUMP DD statement was included in the JCL for the terminating step. The output produced is dependent on parameters supplied in the SYS1.PARMLIB members IEAABD00, IEADMR00, and IEADMP00 for SYSABENDs, SYSMDUMPs, and SYSUDUMPs, respectively. See *Initialization and Tuning* for the IBM-supplied defaults and options that are available.

If the IBM defaults are used, a hexadecimal dump of LSQA is produced when the SYSABEND DD statement is specified. MVS systems do not dump the nucleus or SQA as a default for SYSABEND or SYSUDUMPs. SYSMDUMP defaults include NUC and SQA.

With a SYSABEND, SYSMDUMP, or SYSUDUMP, the system has detected the error and therefore provided a starting point (such as a job step completion code) for analysis. The analyst should always look at the JCL and allocation messages that accompany the dump. The allocation messages contain error messages that can sometimes be helpful. There will also be a JES2 job log that shows the operator messages and responses that relate to the job. The error messages also contain valuable information about the error and should always be investigated.

SYSABEND, SYSMDUMP, and SYSUDUMP errors can generally be divided into two categories: software-detected errors and hardware-detected errors.

Software-Detected Errors

Software-detected errors are those in which one or more of the following occurs:

- A module detects an invalid control block queue.
- A called module returns with a bad return code.
- A program check occurs in system code and a recovery routine changes the program check to a completion code and abnormally terminates the task.

The best approach for a software-detected error is:

1. Use the JES2 job log and allocation messages to investigate all error messages produced. (Refer to the appropriate *Message* manual to determine the causes and corrective action of each message.)
2. Check the abend code defined in the dump. (Refer to *System Codes* to determine causes and corrective actions of the code.) Some abend codes define problem determination areas that can be used to help define the problem.
3. In the event that sufficient data is not available in the *Messages* and *Codes* manuals to resolve the problem, the analyst can go directly to the program listing. The diagnostic sections of most PLMs contain a message/module and abend/module cross-reference. Once the correct module has been located, the program list (supplied in the system microfiche) helps to define the problem.

SYSABENDs, SYSMDUMPs, and SYSUDUMPs normally do not produce system-related data areas other than those which are formatted. Because of this and the fact that error recovery will attempt to reconstruct invalid control block chains before terminating the task, any error that does not occur in the private area may be difficult to resolve from a SYSABEND, SYSMDUMP, or SYSUDUMP alone.

Because of the recovery and percolation aspects of MVS, the SYSABEND, SYSMDUMP, or SYSUDUMP could be the end result of an earlier system error. If so, the analyst should determine if any LOGREC entries were made pertaining to this task and if any SVC dumps were taken while this task was running. The system error is normally reflected in either the LOGREC entries, the dump data sets, or both.

Hardware-Detected Errors

A hardware-detected error is a program check that is not interrupted by a recovery routine. This is identified by a system completion code of X'0Cx' where x is the program check type. For this type of error, the analyst needs to know the address of the module where the program check occurred, and the register contents when the program check occurred. The best place to locate this information is in the RTM2WA that is pointed to by the abending TCB.

Given the registers and PSW at the time of the error, the analyst should determine the module that program checked by using the load list link edit maps of the program. (If the module is outside the private area, a NUCMAP or LPA map may be necessary.) Then the analyst should examine the program listing for the module until the cause of the program check is defined.

Section 4. Symptom Analysis Approach

This section describes how to identify correctly an external symptom, and provides an analysis procedure for determining what kind of problem is causing the symptom.

Each external symptom is described in a separate chapter, as follows:

"Using IPCS to Analyze Dumps" on page 2-98 names the IPCS CLISTs that generate symptoms for problems found in a dump. You can use the IPCS subcommand VERBEXIT SYMPTOM to print these symptoms when you start your analysis.

Waits

Wait states may be either disabled or enabled. The characteristics of and an analysis approach for each type are described in the following topics. Also, the system termination facility is described later in this chapter.

Note: Be aware that hardware problems, such as waiting on I/O or the timer, can be the cause of enabled waits. Use the information in these topics if you feel that the cause of the wait is a software problem.

Characteristics of Disabled Waits

Situations can develop during execution of the MVS system that require the software to abruptly terminate the system by loading a disabled PSW with the wait bit set to 1. In previous systems, this occurred much more frequently than it does in MVS because, in MVS, many of these situations were removed from the code and replaced with software error recovery. However, a few cases still remain that cause this symptom. To understand these situations better, refer to the 'Wait State Codes' section of *System Codes.*

A more critical situation for the analyst is a disabled wait that is caused when data areas containing PSWs referenced by the dispatcher, RTM, or hardware are overlaid and subsequently fetched for use in an LPSW, or for interrupt processing (interrupt new PSWs). This might occur when a PSA overlay condition exists, that is, these PSWs have been inadvertently overlaid by a program running in supervisor state key 0. Other data areas, such as PRBs, may contain PSWs used by the dispatcher and are also potential sources of the disabled wait state. Loading of bad PSWs are difficult to track down. Low-address protection prevents inadvertent destruction of the PSWs used by the hardware to give control to the first level interrupt handlers. Bytes 0 through 511 of storage are protected by low-address protection which should eliminate many occurrences of invalid disabled wait conditions. The most common MVS uses of the LPSW are:

- hardware loading from low storage for an interruption-processing sequence
- dispatcher loading from fields X'420' and X'468' in the PSA. The PSW loaded by the dispatcher may have come from an RB or an SSRB.
- RTM (IEAVTRTS) passing control to FRRs
- the system termination routine
- I/O and external FLIH's LPSW to resume task or SRB.

Storage overlays resulting in wait state PSWs are approached in the same manner as other storage overlays. The important step is to realize the storage overlay has occurred, then re-create the process that was possibly responsible. The discussion of pattern recognition in the chapter "Miscellaneous Debugging Hints" in Section 2 should be helpful.

Analysis Approach For Disabled Waits

The following is a list of objectives that provides a systematic approach to analyzing a disabled wait.

You can use the IPCS subcommand STATUS CPU for wait state analysis.

Objective 1 - Determine positively that an actual disabled wait condition exists. Is the PSW the type that is used when MVS loads an explicit wait or is this an overlaid PSW with the wait bit on?

Analysis - Examine the *current* PSW contained in the dump according to the technique described in the chapter "Stand-alone Dumps" in Section 3. The PSA overlay should also be analyzed to determine if key PSWs have been overlaid.

If the PSW shows an explicit wait, look up the wait state in *System Codes* to find what conditions could cause the explicit wait. You may need to do some extra analyzing before the condition can be related to a component. (*Note:* No further analysis for explicit wait situations is discussed in this book.)

If the PSW suggests an overlaid PSA or some other error source, proceed to Objective 3; otherwise proceed to Objective 2.

Objective 2 - Determine if the situation has been improperly diagnosed as a disabled wait. This will eliminate a situation in which the locked console is diagnosed as a disabled wait.

Analysis - In previous operating systems, the operator's inability to communicate with the system through the console was an external indication of a disabled wait condition. In MVS, this same external symptom is often not a true disabled wait. Console communication is dependent upon other services of the operating system, such as paging, and the I/O subsystem. A problem in any of these services often terminates console activity and causes an apparent "disabled wait" situation, when the PSW does not actually reflect a disabled wait.

If the current PSW is not disabled for external and I/O interrupts or if the wait bit (X'0002') in the PSW is not set to one (PSW = X'070E0000 00000000'), you should proceed to either the "Enabled Wait Analysis" topic later in this chapter or to the chapter on "Loops" later in this section.

Objective 3 - Once you know that the disabled PSW is the result of an overlay in low storage or in another data area, you must gather specific data about the overlay. Ask such questions as: What was the damage to the PSW? When did the overlay most likely occur? Where did the PSW come from?

Analysis - It is important to try to find out how the PSW was overlaid - was it a byte, an entire word or doubleword, a single bit, or was a large portion of the surrounding area destroyed along with the PSW? (The discussion of Pattern Recognition in the chapter "Miscellaneous Debugging Hints" in Section 2 will help you determine this.) Much of this analysis depends on your experience and familiarity with the normal data for the subject PSW and the surrounding area. You should try to gather enough data to know, for example, that "n" bytes were overlaid beginning at location xyz.

Also, examine the system trace table, if available, and try to determine when the PSW was probably last valid. Look for interrupts and unusual conditions in the trace entries to try to reconstruct the process(es) leading up to the incorrect PSW.

If the trace indicates the overlay occurred *after* the most recent trace entry, the registers are important because they may show recent BALs and BALRs and they may contain the address of a routine or control block that was used to overlay the subject PSW. This is actually a good situation because it will not take long to relate the overlay to some bad pointer in a control block and, hopefully, your analysis will proceed to a specific component.

If the overlay occurred several trace entries earlier, determine a possible save area that might contain the registers that were active at the time of the overlay by examining interrupt entries or dispatch entries in the trace table.

If there is no trace table, it is almost impossible to define when the overlay occurred. You might try to analyze, for example, TCB save areas, hoping for a clue as to when the overlay occurred and to gather information concerning the problem. However, this process is basically undefined and undisciplined. In most cases, a trap for the overlay can be generated at this point and used as soon as possible.

Objective 4 - Determine which component most likely caused the overlay and choose a likely set of modules from that component to analyze at an instruction level. Determine which data area field contains the bad address and who set up the field.

Analysis - As mentioned earlier, by using the registers and trace table it is possible to identify which code actually overlaid the PSW, but the source of the error must still be found. This mostly involves screening code to reconstruct the path which caused the overlay and locating the data that generated the bad address. At this point, you want to learn which module set the bad field so you can start backtracking.

Shortcuts are possible according to the analyst's familiarity with the modules that are involved. Certainly the main objective should be to decide which component is most likely responsible.

Characteristics of Enabled Waits

Enabled waits have traditionally been the most difficult problem to analyze because of the lack of an obvious failure. The enabled wait provides no indication of error other than that the system apparently has nothing to do. In fact the enabled wait has been accurately described as an end symptom of a problem with no obvious causes. The task of determining the possible cause is left to the debugger. Other types of software failures - abends, program checks, loops, messages - provide a starting point for analysis; that is, software or hardware has indicated a violation of interfaces or data integrity and has halted the erroneous process at the point of error. The enabled wait provides none of these.

You can use the IPCS subcommand ANALYZE for contention analysis on locks, ENQs, and I/O devices. This analysis examines all address spaces, QCBs, QELs, and UCBs, and identifies units of work that appear to be causing a bottleneck in the system.

Note: The subsystem design of many components includes a dispatching mechanism and internal control block structure not generally recognized by the operating system. When these subsystems (for example, ACF/VTAM, ACF/TCAM, JES2) malfunction, work through these components is often halted. Because of the critical nature of these processes, external signs of the problem are often detectable. Within this debugging discussion, these problems are often treated as wait states, that is, the system may be capable of running batch work, but the TP network appears "hung-up." This general discussion of analysis-approach applies for problems such as "permanently" swapped-out address spaces, TP network hung, and no batch running. The advantage is that the external symptoms may allow you to more easily isolate the problem component or at least a starting point - it may be obvious that ACF/TCAM is not responding, or that JES2 is not processing input.

Experience has shown that in MVS a much greater percentage of re-IPL situations are caused by enabled waits than in previous systems. One reason for this characteristic of MVS is software recovery. Software recovery attempts to repair the damage caused by a failure and allow the system to continue meaningful operation. The general philosophy of recovery is to isolate the error to a job, terminate the job, and allow the system to continue. This philosophy dictates that under certain conditions innocent work may be forcefully terminated.

Software recovery obviously may cause the termination of some critical process which in turn causes dependent processes to wait indefinitely. For example, assume that while processing a page-fault, an error occurred during the I/O interruption processing; software recovery was invoked and subsequently caused a cleanup of the bad control blocks, but did not post the I/O requestor. It is possible that the paging mechanism will wait indefinitely for the missing interrupt. This in turn could cause a problem program to wait indefinitely for the paging operation to complete. The end result is no work accomplished and also no external problem symptom, although a problem clearly exists. The debugger must find the bottleneck - the paging exception - and subsequently back-track enough to determine why the bottleneck still exists. Very often, this back-tracking requires analysis of several components in order to determine the original cause.

Analysis Approach For Enabled Waits

It is most important that you understand the actions that must take place in order to accomplish work in the operating system. This requires a basic understanding of the key system processes in MVS - paging, I/O, dispatching, locking, WAIT/POST, ENQ/DEQ, ACF/VTAM, ACF/TCAM, SRM, JES2/3. These areas of the system are responsible for directing work through MVS; a malfunction in any one may cause global system problems. Several, if not all, must be investigated in order to determine why work is not progressing.

This investigation requires a disciplined approach. The relationships of component interfaces and their mutual dependencies must be understood. With this in mind, the debugger should proceed to gather information about the various processes and try to integrate his findings with his other information and

assumptions about the problem, always trying to isolate one cause of the bottleneck. He must avoid the tendency to guess, assume, and go off on tangents once the first irregular item is uncovered. Instead, he should continue to gather known facts and piece them together in some logical pattern that recreates the situation.

In the majority of wait state cases, more than one key process will appear backlogged. The challenge is to determine how these problem processes relate and which is the fundamental cause of the wait situation. After you gather the facts and understand the bottlenecks, you must answer one question. If I "pull the cork" on this given bottleneck will all the other intertwined situations resolve themselves? In *every* problem there is only one bottleneck for which the answer to this question is "yes." The other problems are consequences of this key process's failure to complete its designed function. Isolating the process is half your battle; the other half is determining the cause of this *one* process's failure.

Following is a suggested disciplined approach for the problem solver who is approaching a system wait problem. The approach involves three distinct stages of problem analysis:

Stage 1 - Preliminary global system understanding, including

- system externals
- current system state
- LOGREC analysis
- trace analysis
- determining the reason for waiting

Stage 2 - Key subsystem analysis - an in-depth analysis of the MVS components that are responsible for accomplishing work.

Stage 3 - System analysis - using the information gathered in Stages 1 and 2 the problem solver must "step back," get perspective about the known facts by piecing them together in a logical fashion, and isolate the error to a process, component, module, etc.

This approach is described in detail in the following sections.

Stage 1: Preliminary Global System Analysis

1. ***System Externals*** - Completely understand the system externals of the situation. Console sheets and the system log should be inspected.

   - For any enabled wait (operators call it "system hung") find out if a display requests command was issued. (Lack of operator action can cause system bottlenecks.)

     You can use the IPCS subcommand COMCHECK to get a list of the outstanding WTORs.

   - Often many pages of console sheets must be investigated to uncover operational problems and explain events uncovered in the dump. Scanning provides a feeling for the events, jobs, requests, etc. leading up to the problem.

   - Make sure all DDR SWAP requests, I/O error messages, SQA shortage messages, etc. can be explained.

Always take the time to examine these external areas because a small effort here could save many hours of detailed dump analysis. Do not overlook obvious items such as a MOUNT PENDING message in the console log that can cause system problems.

- If an IPL for an alternate nucleus was executed, make sure that the correct nucleus was specified. Additionally, CVTNUCLS can be checked to verify that the IPL was for the correct nucleus. CVTEXT2(CVT+X'148') points to the CVT extension. CVTNUCLS is located at offset X'04' in the CVT extension.

  You can use the IPCS subcommand STATUS SYSTEM to determine the alternate nucleus used at IPL.

2. ***Current System State*** - Investigate fully the current situation as depicted by the dump.

   For enabled waits, the PSW should equal X'070E000000000000' (often called the "no-work" wait). The last entry should be the wait or there should be a considerable recurrence of the no-work wait in the system trace table - see the chapter on "MVS Trace Analysis" in Section 2. If this is not the case, use the disabled wait analysis approach (earlier in this chapter).

   If the PSW indicates the no-work wait situation, you have an enabled wait. You should now check other global system data areas indicators to get the whole picture. Following are key global indicators:

   - PSASUPER field (PSA+X'228') should have the dispatcher super bit (PSADISP) set. PSADISP is left on when wait is dispatched and will be turned off on the next I/O or external interrupt. If any other bits are set, some supervisor routine is in control. This situation can indicate incomplete processing by the associated routine. All possibilities should be pursued until the situation can be explained.

   - Because of SRM timer/analysis processing, even when the system is in the enabled wait situation, the state of the processor at the very instant the dump was taken can indicate, via the "super bits" or locks indicator (PSACLHS), that some process was occurring. You must determine in this case that these fields being set is normal and continue with wait analysis. If the fields cannot be explained, you have isolated the error.

   - There should be no locks held, as indicated by PSACLHS on either processor. This situation is similar to the one described just above. You must try to discover the owner of the lock and determine why it is still held despite the fact that the system is waiting. Often the purpose of the lock will provide insight as to who the owner might be. The chapter on "Locking" in Section 2 should be of help in your analysis.

     You can use the IPCS subcommand STATUS CPU to determine, for each CPU, if any locks are held.

3. ***LOGREC Analysis*** - Determine if key components have encountered difficulty; determine previous errors encountered by the system. This can be accomplished by inspecting SYS1.LOGREC as well as the in-storage LOGREC buffer. Errors encountered in any of the key processes noted

earlier (RSM, ASM, IOS, JES2/3, SRM, ENQ/DEQ, etc.) may provide further information. If you do find an error associated with any of these areas, determine whether it could lead to the bottleneck.

The LOGREC records generally contain the names of the error-encountering routines and often the job on whose behalf the system was processing at the time of the error. If the routine names are not present, you may have to use system maps and the PSW/register information in the LOGREC records in order to associate errors with components. The discussion of LOGREC analysis in the "Use of Recovery Work Areas" chapter in Section 2 should be helpful in your analysis.

4. ***Trace Analysis*** - Determine the last activity within the system.

   Because of SRM's timer processing, the trace table for most wait conditions is not useful. However, on the rare occasion that the system has been stopped or if for some reason the trace is not overlaid with timer interrupts, the trace should be analyzed to ensure normal processing, for example, page faults are being processed, I/O is being accomplished. Be suspicious of large (relative to most entries) time gaps in the trace table. If the table has not wrapped-around, process re-creation may be of some use in determining what the system was doing up to the point of incident. (The chapter on "MVS Trace Analysis" in Section 2 should be helpful.)

5. ***Determine the reason for waiting*** - Once it has been determined that the system is waiting, it is always useful to determine what the various address spaces or jobs are waiting for. This is accomplished by inspecting and scanning the various tasks and their associated RB structure in a formatted stand-alone dump. Remember the RCT, started task control (STC)/LOGON, and dump task may all be waiting in each address space - this is normal. The question you should ask is: Why are the subtasks below the STC/LOGON waiting?

   Generally in an active system more than one address space will be waiting for the same or similar resource in a problem situation. Therefore, as you scan and analyze address space status, look for suspensions in common modules (RB resume PSWs containing similar addresses):

   - Many tasks in page-fault wait can indicate the paging or I/O mechanism is faulty.
   - The RCE can indicate a real frame shortage.
   - Many tasks in terminal I/O wait can indicate something is wrong with the TP access method or some part of the network.
   - Several Resume PSWs pointing into the ENQ/DEQ routine, IEAVENQ1, can indicate an ENQ resource contention problem.

   In general, be on the look-out. Try to compare and relate the system activities as you encounter them. Often more than one process or address space is held up because of a common bottleneck. It may be a global resource required by more than one address space, for example, a lock or data set. It is important that the *exact* cause be determined.

Stage 2: Key Subsystem Analysis

As part of this investigation, if nothing can be easily determined from a cursory address space scan, you may have to delve into the key components. Following are some highlights of the important and potentially suspect areas:

1. ***I/O Subsystem*** - Check for unprocessed I/O requests, bottlenecks in the I/O process will almost always log-jam the system. Since IOS is the central facility for controlling I/O operations, I/O problems should always be suspected in an enabled wait condition. Therefore, the IOS component and its associated queues should be analyzed early in the subsystem analysis stage of debugging. Important IOS control blocks will indicate whether problems exist in the I/O process.

   You can use the IPCS subcommand IOSCHECK to analyze IOS control blocks and determine if there are problems in IOS.

   Unit control blocks (UCBs) are a logical representation of each I/O device containing I/O active indicators at offset X'6' and X'7'. If any indicator is set, this device must be further investigated. This condition can indicate either a hardware or software problem.

   The active (UCB/IOQE) requests must be further investigated to determine the associated requestors and what effect their I/O not being serviced will have on system operation (for example, if paging I/O or console I/O is not being serviced, the system will usually stop).

   The UCB contains indicators for DDR, intervention required, and missing interrupt handler processing. Any such indication must be further investigated.

   An ENQ on the SYSZEC16 resource is an indication of a waiting condition generally associated with swapping. The swapping process cannot complete until active I/O finishes. In a quiesced system, an ENQ on this resource must be further investigated.

2. ***Paging Mechanism*** - Check for unserviced page faults. ASM, RSM, and SRM are closely related and depend upon each other to maintain real storage, the swapping process, and page fault resolution. If, when you determined the reason for waiting as described in stage 1, you discovered several page fault wait conditions, be suspicious. You can use the IPCS subcommands ASMCHECK and VERBEXIT ASMDATA for analysis and formatting of ASM control blocks.

   Some key indicators in determining page fault waits are:

   - ASCBLOCK = X'7FFFFFFF' - indicates suspension while holding a local lock. If in task mode at the time of suspension, the resume PSW instruction address (saved at IHSA + X'10' of the locally locked address space) should be checked. When the instruction address = RBRTRAN(X'C' offset) and RBXWAIT = 0, it indicates the task is suspended while it waits for a page fault resolution. Note that a data reference page fault might have occurred; in this case, the instruction address does not equal RBTRAN. RBTRAN may be checked against the register and displacement as determined from the page-faulting

instruction. The page fault occurred when a new module (paged-out) was referenced. If in SRB mode at the time of suspension, an SSRB will be queued from a PCB.

PCBs can be found in the following locations:

- For pages with I/O in progress, PCBs are pointed to by the PFTE for the frame backing the page (PFTPCB, PFT+X'18').
- For cross memory page faults or segment faults, or common area page faults, PCBs are queued on the notification PCB queue for the home address space. The NPQ is anchored in the RAB for the home address space (RABNPQF, RAB+X'6C').
- For pages that cannot be backed because a frame is not available, PCBs are queued on the defer PCB queue. The DPQ is anchored in the RIT (RITDPQF, RIT+X'8C').

- ASCBLOCK=X'00000000' - indicates the local lock for this address space is not held. The RB structure can reveal the same situation as described above for RBOPSW instruction address = RBRTRAN, RBXWAIT=0 *and*, in addition, an RB wait count =1. Note that a data reference page fault might have occurred; in this case, the instruction address does not equal RBTRAN. RBTRAN may be checked against the register and displacement as determined from the page-faulting instruction. If you find several tasks in this state, check the dump for the page represented by RBTRAN. Is it in storage? (Remember for private area addresses to be sure that the address space you are investigating is printed.) If the page is not in storage you may have a potential paging problem. Again, if in SRB mode at page fault time, the SSRB must be found to determine more about the process.

If you suspect that paging is a potential problem in the system, several SRM and RSM/ASM control block fields can be examined.

You can use the IPCS subcommand VERBEXIT RSMDATA to analyze and format RSM control blocks, and the IPCS subcommand VERBEXIT SRMDATA to format the SRM control blocks. Periodically, SRM calculates the total paging rate (swap, VIO, and demand), the demand paging rate (separately), and the average maximum unreferenced interval count (UIC). These values are saved in the resource management control table (RCT) for use by the algorithms that adjust the multiprogramming level (MPL) of the domains. The RCT can be found by locating the SRM control table (RMCT) which is pointed to by the CVT (field CVTOPCTP at offset X'25C'). The RCT, like many other SRM control blocks, follows the RMCT and can be located by examining the program listing for module IRARMCNS. The total paging rate is identified by field name RCVPAGRT. The demand paging rate is identified by field name RCVDPR. The average maximum UIC is identified by field name RCVUICA. Two other fields of interest in the RCT are RCVASMQA, which is the average number of requests in progress or queued by the ASM, and RCVMSPP, which is the average time in milliseconds for the ASM to process a page request.

Also examine the RSM control and enumeration area (RCE), which can be found from the CVT (field CVTRCEP at offset X'490'). The available frame count is at RCE+X'88'. Previously a low available frame count (less than 15-20) was an indication that storage contention or paging was a problem. The SRM attempts to better utilize the real storage resource and a low available frame count is not uncommon, although it can still be a source of concern. The combination of low available frame count, a low UIC, and a high demand paging rate indicate storage contention great enough to adversely affect response time and throughput.

ASM maintains a count of the number of paging requests received and the number for which processing has completed in the ASMVT. If these counts are not equal, ASM is backed-up and page faults have not been resolved. This can be caused by an I/O problem or some internal ASM problem. Finding unprocessed work on these queues will aid in determining whether ASM is the problem component. But again be careful: you are still gathering data about the wait state. Your purpose now is not to debug ASM - it may not be the problem. Note the apparent ASM problems and continue your investigation. Later when you piece together your findings and find the real source of the problem, detailed debugging and logic flow will be required.

3. ***ENQ/DEQ*** - Check for unresolvable resource contention. Finding an ENQ/DEQ interlock and determining what work is being held up because of this interlock can provide important information about the overall problem.

   You can use the IPCS subcommand ANALYZE to do ENQ contention analysis. Or, you can use the QCBTRACE, Q, or GRSTRACE option of AMDPRDMP to get a formatted structure of the resources and the work that is in contention for them. (Also, you can use the DISPLAY GRS,CONTENTION command to display resource contention information for the current global resource serialization complex.)

   Determining who owns the resources and the current status of the owners (if swapped-out, why? or if waiting, for what?) often provides important clues in understanding the bottleneck.

   Also in your scanning process, you should be on the alert for address spaces that contain subtasks (usually below the STC/LOGON level) with multiple RB levels, and with the lowest RB containing a resume PSW with an address somewhere within ENQ code (nucleus resident) and with the RB wait count RBWCF=1. The previous RB should be an RB with the ENQ SVC (SVC X'38') indication in the "WC-L-IC" portion of the RB prefix (-4 offset). This indicates that this task and probably the address space are suspended because of an unsatisfied ENQ request. If several address spaces or tasks are found in this state you should find out why.

   You can use the IPCS subcommand VERBEXIT GRSTRACE to format ENQ data for all resources. Or, you can use the GRSTRACE, Q, or QCBTRACE option of AMDPRDMP. An illustration follows:

   An investigation of data produced by either IPCS or AMDPRDMP data shows that there are many requests backed-up on resource A. The analyst notes this and determines what ASID or TCB owns resource A at this time (in this example, ASID 9). The other resources represented in the data are now scanned. If ASID 9 is backed-up behind someone else (ASID 10)

waiting for another resource (B), you must now determine ASID 10's status with respect to other resources, including resource A. Essentially you are looking for cases where:

- An address space has resource A and is waiting for resource B *and* a second address space has resource B and is waiting for resource A. This indicates a deadlock. You must determine the faulty process. In this case you have probably isolated the error to the ENQ process and the way it is being used. You must analyze the task structure of each address space to determine how this situation occurred. Do not forget the SYS1.LOGREC buffers. They may contain clues like errors in ENQ/DEQ or one of the tied-up address spaces (jobs). Faulty recovery should be suspected if the latter is the case.

  It may be that a job requests control (via ENQ) of a resource and subsequently encounters a software error. The task's associated recovery gains control and "recovers" from the error but does not dequeue (DEQ), and therefore does not release the resource. Eventually, the contention for this resource, depending on its importance, could cause severe problems.

- An address space has control of a resource and a lot of address spaces are queued-up behind this address space. In this case, you must find out why the holder is not releasing the resource. Also know your system. It is not unusual to see activity on the master catalog resource: "SYSIGGV1 - Master Catalog Name." But be suspicious of most resources. Determine from the holder's task structure what process it is attempting. Determine whether the address space is waiting or swapped-out and why. If it is not waiting or swapped-out, check the non-dispatchability bits and the possibility that the address space is looping.

  This second case is much more likely to be a sign of some other system problem. Your clue is what is preventing the holder's execution; this will point you to another process which *must* be investigated and may lead to the detection of the final problem.

Note: When analyzing a dump of a quiesced system you should be suspicious of "unusual" ENQ resource names - resources that should not be a contention factor in a quiesced system. The presence of these names should be understood and explained because they very often will point you to the problem area. Common resource names are:

SYSZEC16 - PURGE	- Can indicate a problem in the I/O process related to the resource holders address space
	Can indicate a bottleneck in the swapping process
SYSZVARY - x	- indicates the reconfiguration component has been invoked - why is it not completing?

4. ***Dispatching*** - Determine if there is work to do in the system. A common trouble indicator is an MVS dispatching queue containing elements that indicate work is ready to execute in a waiting system. The SVTGSMQ, SVTGSPL, SVTLSMQ, and each ASCBLSMQ and ASCBLSPL should be empty. (The chapter on "System Modes and Status Saving" in Section 2 contains details of these queues and how to find them.) Generally it is not a

problem in the dispatching mechanism itself but merely an error indication. Often the most useful information is just that 'yes, there is work.' Why is it not being dispatched? Is there a problem in some other area of the system? Is the address space swapped out? Yes, there may be a real storage problem delaying swap-in. Or perhaps SRM has not been told to swap-in the address space via a "user-ready" SYSEVENT. (The ASCBURR bit indicates that a "user-ready" SYSEVENT is required). In summary, investigate the OUCB for the address space you are concerned with.

You can use the IPCS subcommand SUMMARY FORMAT to get information on ASID-related control blocks.

Another useful point is to find out what problems could arise if this work were not dispatched. Investigating the queued work will indicate what would be accomplished if this work were executed. This is usually important because it can clear up much of the "smoke" you may be encountering in your overall system investigation.

Likewise, investigate the task structure. Generally, you can ask the same questions as above, but you must look in different places for the key indicators. Among the most important indicators are:

- The ASCB, which contains two counts of ready TCBs in the address space: ASCBTCBL, which is a count of ready TCBs that require the local lock; and ASCBTCBS, which is a count of ready TCBs that do not require the local lock.
- The TCB non-dispatchability flags.
- The RTM work area, which contains status at time of error.
- The RB structure. Look for long RB chains or unusual SVCs and interrupt codes. Look for page fault waits.

Again, use this information to lead you to processes or problems that hold-up the system.

5. ***Locking*** - Determine if there is a locking conflict. The locking mechanism causes system bottlenecks when it is not used properly. The global spin locks cause obvious problem symptoms such as one processor spinning in the spin lock manager (IEAVELK) in an MP environment. In a UP environment, global spin locks are generally not a problem unless a lock word or interface is overlaid or bad, causing a disabled spin. The enabled suspend locks (local/cross memory services, processed by IEAVESLK) are generally the problem ones. The chapter "Locking" in Section 2 describes in detail the considerations with which you should be concerned. Elements on the local/cross memory services suspend queues may indicate a problem. The technique you adopt to resolve the conflicts is exactly the same as the ENQ interlock or logjam situation.

   You can use the IPCS subcommand ANALYZE to do suspend lock analysis, and the IPCS subcommand STATUS CPU to determine which disabled locks are held.

6. ***Teleprocessing*** - Determine if the TP network is responding. Problems in the TP network often manifest themselves as waiting network or waiting terminals, even waiting systems. The chapter "TP Problems" in Section 4 contains a description of TP problem analysis.

   You can use the IPCS subcommand VERBEXIT VTAMMAP to format VTAM control blocks, and the IPCS subcommand VERBEXIT TCAMMAP to format TCAM control blocks.

   An important fact for the problem solver here is that these are subsystems. As such, they maintain their own control blocks, queues, and dispatching mechanisms. They are responsible for work being processed once it enters the subsystem and they often have little direct dependency on MVS. That is, normal MVS problem indicators will not generally solve the problem. You must understand the subsystem's work-processing mechanism in order to be an effective analyst. For example, ACF/VTAM has its own address space with a number of tasks used primarily for network start-up, shut-down, and operator commands. In most ACF/VTAM problems, a look at the ACF/VTAM address space will show these tasks are waiting. However, this is normal when no operator processing is required. Even though ACF/VTAM is waiting, this is not the place to be distracted. Again, remember this ACF/VTAM task structure, put it aside as part of your information gathering.

7. ***Console Communication*** - Determine whether console communication is possible. The system can appear or actually prove to be waiting because the operator is not able to communicate with MVS. This could be the sign of a problem almost anywhere in MVS, but it often indicates an error in the communications task or its associated processing.

   You can use the IPCS subcommand COMCHECK for communications task analysis.

   The communications task (comm task) runs in its own address space and is usually represented by the TCB in the formatted portion of the stand-alone dump that is identified by a X'FD' in the TCBTID field (TCB+X'EE'). By inspecting the RB structure associated with this task, you can determine the current status. It is not unusual to find one RB with a resume PSW address in the LPA and an RB wait count of one. If more than one RB is chained from the TCB and you were not able to enter commands, analyze the RB structure because this is not a normal condition.

   The key control block is the unit control module (UCM) which is located in the nucleus. CVTCUCB (CVT+X'64') points to the base UCM. The base UCM-4 contains the address of the UCM MCS prefix and the base UCM-8 contains the address of the UCM extension. From the UCM you can determine the status of the various consoles. The following should be considered and can warrant further investigation:

   - Important WTORs are outstanding.
   - An out-of-buffer (WQEs, OREs) situation exists.
   - There are unusual flags in the UCM.
   - There is a full-screen condition.
   - There is a console out of ready.

Remember that comm task processing is dependent on the rest of the operating system. Most likely, some external service or process has caused comm task to back-up, and this possibility should be investigated. Remember the debug process: gather all the facts, then proceed with analysis.

8. ***SDUMP*** - During SDUMP processing, enabled waits can occur if SDUMP terminates unsuccessfully or is suspended because of a system service it uses. SDUMP does two things that can cause a wait:

   - SDUMP sets the system nondispatchable.
   - SDUMP sets the tasks in the address space being dumped nondispatchable. If this is a critical system address space, a wait can occur.

   If the system is in a wait state because all ready work is set nondispatchable, check the following SDUMP control blocks to determine if SDUMP is responsible.

   - CVTSDBF (CVT+X'24C') - If the high order bit of this word is on, SDUMP is active. If this bit is off, SDUMP might have terminated without resetting other tasks dispatchable, so still check the other indicators.

     In the following items, CVTRTMCT (CVT+X'23C') points to the RTCT. You can use the IPCS subcommand

     CBFORMAT RTCT STRUCTURE(RTCT)

     to format the RTCT.

   - RTCTSDPL (RTCT+X'9C') - This is the address of the SDUMP parameter list. By looking at the parameter list, you can determine the type of SDUMP requested. By finding the dump header (offset X'C' in the parameter list points to the header), you can determine which system component called SDUMP.
   - RTCTSDND (RTCT+X'108') - If the second bit in this flag is on, it indicates that SDUMP has set the system nondispatchable.
   - RTCTSDF3 (RTCT+X'10A') - This address space array has a maximum of 16 four-byte entries (for 16 address spaces), where the first byte indicates the ASID. If bit 2 (RTCTSDAN) in the third byte is on, SDUMP has set the tasks in the address space nondispatchable.

Stage 3: System Analysis

At this point you should have a detailed understanding of the system and its key components. You should know which components or processes are back-logged and, correspondingly, what work (jobs) is not being processed by the system because of these back-logs. You must now stand back from the problem.

Answer this question: Which of these problems and situations can be related to or attributed to each other? For example, if I/O is queued for the paging devices (indicated by IOQs on the paging devices' UCBs) and you also found several address spaces are in "page-fault wait," you can now relate these findings. And if

one of these address spaces performed an ENQ for a resource and did not yet DEQ because of the page-fault suspension, it is very likely other address spaces are also backlogged behind this job's processing. Initially your ENQ/DEQ analysis showed the problem, but at this point you can attribute the ENQ contention problem to the page-fault suspension problem that you have already attributed to the I/O problem.

This process must be repeated for all the potential error situations you uncovered in your investigation. Do not forget to use the system indicators in your attempt to arrive at the source of the problem. And most importantly, ask yourself: If I unplug this bottleneck, will all the other intertwined situations resolve themselves? In the previous example, resolving the ENQ situation *will* allow the work queued in the ENQ/DEQ component to execute *but* the "page-fault waiting" job will still be hung. That is, ENQ/DEQ is not the problem to pursue. Indeed, if you resolve the I/O problem, this page fault is resolved, the DEQ will be performed, and *all* work in the system will resume normal operation. Yes, the I/O problem is the important consideration in this case. The I/O problem is the one that must be pursued. When this problem is resolved, the enabled wait state condition has been resolved. Global system areas, recovery work areas, LOGREC analysis, and IOS component analysis will be necessary to further isolate, and eventually solve, the problem.

System Termination Facility Wait State Codes

For some I/O errors, the system termination facility puts into the PSW both the wait state code found in the LOGREC recording buffer (LRB) and a reason code indicating why module IGFPEMER is in the wait state.

The PSW appears as: X'00xE0000rrrrwwww'

Where:

x — The reason IGFPEMER is in the wait state. IGFPEMER is waiting for one of the following:

- 1 - an interruption that indicates the channel path is cleared.
- 3 - an interruption that indicates I/O is completed.

wwww — The wait state code.

rrrr — The reason code.

The wait state code and the reason code are obtained from the LRB passed to IGFPTERM.

When you see a wait state code in the PSW with this format, you can locate the message that was to be displayed at the operator's console and the LOGREC record that was to be written to SYS1.LOGREC. When the wait state code is loaded, register 1 points to a two-word parameter list, where the first word contains the address of the WTO message, and the second word contains the address of the LOGREC record.

Loops

Loops are defined as disabled or enabled, depending upon their external appearances. A *disabled* loop can be recognized externally by a solid system light or 100% CPU busy on the system activity display, and the inability to communicate with the system through the consoles (that is, no input or output). Usually, a disabled loop indicates a hardware and/or software malfunction. There are several cases in MVS however in which a disabled loop is purposely used and is not an error indication. These cases are discussed later in this chapter.

An *enabled* loop is generally much larger than a disabled loop. Observed from the console it appears as a bottleneck: the system seems to be slowing down periodically, suggesting performance degradation. The operator may notice that a particular job remains in the system for a long time and does not terminate.

Common Loop Situations

There are three common loop situations:

1. Processors of an MP environment communicate via the signal processor (SIGP) instruction. Often the SIGP-issuing processor enters a disabled loop until the receiving processor either accepts the SIGP-caused interrupt or performs the operation requested by the issuing processor. This loop serializes the processors in the MP configuration. The SIGP-issuing processor loops in a nucleus-resident module, IEAVERI or IEAVESGP.

   You can use the IPCS subcommand STATUS CPU to determine the active module on each CPU.

   Often during an MP dump analysis you will find that one processor was in this loop. This is not an error if:

   - The operator stopped one processor and not others to investigate a suspected problem.
   - The receiving processor is disabled for external interrupts, thereby preventing the SIGP-issuing processor from proceeding.

   If this situation continues for an extended period, it means there is a system problem but the loop is a result of that problem and is not an error itself. Most often, other processors' activities must be analyzed to determine the problem. For a more detailed discussion of MP communication, refer to the chapter "Effects of MP on Problem Analysis" in Section 2.

2. The spin lock manager (IEAVELK), which resides in the nucleus and controls part of the locking mechanism of MVS, contains a section of code that enters a disabled loop when a global spin lock is requested but is not available. On a UP this is an invalid condition and always signifies an overlaid lockword or invalid lockword address. On an MP system, this usually indicates that another processor is holding the lock and not releasing it. But it may indicate an overlaid lockword; if not, the problem is definitely on the processor that is not releasing the lock. In either case, register 11 contains the pointer to the requested lockword and PSALKR14 contains the address of the requestor.

Check the value in the lockword. If the lockword is not valid, it is necessary to identify who overlaid the lockword. It is possible that the lockword was overlaid in conjunction with some other problem. Again, since the disabled loop may not be the problem but a symptom of a possible error on another processor, determine why the requested lock is not available. For a detailed discussion of "Locking" see Section 2.

3. The intersect service routine (IEAVEINT), which resides in the nucleus, controls the intersect serialization in a manner similar to the lock manager. Intersect provides the serialization between a routine holding either the dispatcher or local lock and the dispatcher (IEAVEDS0). The intersect service routine contains a disabled loop which is entered when a routine requests the intersect on one processor and the dispatcher is active on another processor. The dispatcher active field is located via PSA+X'B4C', which points to the SVT. SVT+X'160' (SVTDACTV) contains one byte per processor and is indexed by the processor address. If the loop in the intersect routine is in control, register 2 contains the physical processor address where the dispatcher is processing, and register 14 contains the return address of the routine requesting the intersect. It is possible that the dispatcher active field is overlaid; its contents should be checked for validity. Each byte of the dispatcher active field corresponds to a possible online processor. Valid contents are logical processor address, or the logical processor address with the high-order bit on.

   Because the disabled loop might not be itself a problem, but a symptom of a possible error, determine how the dispatcher active field became overlaid or why the dispatcher is looping on the other processor.

Analysis Procedure

Generally for loop analysis, you will have a stand-alone dump if the operator considered the problem serious enough to re-IPL the system, or an SVC dump, SYSUDUMP, SYSMDUMP, or SYSABEND (provided by the software recovery) if the operator pressed the RESTART key in order to break the apparent loop. For the SVC dump, SYSUDUMP, SYSMDUMP, and SYSABEND dumps there is an abnormal completion code of X'071' associated with the looping task of a job if the RESTART key was pressed when the program was actually looping. In addition, a formatted SYS1.LOGREC listing should be available.

Note: With the loop recording option, you can record information about the loop by selecting the option on the system frame. This option records several hundred values of the instruction counter in the form of an instruction trace. After recording the instruction trace, the processor is put in the STOP state. The information becomes available when you take a stand-alone dump or when the SVC dump is invoked.

Before you can determine what problem is causing the loop, you must determine first that a loop really exists, and second whether it is enabled or disabled.

First, verify that a loop exists. The *disabled* loop situation is fairly straightforward. The PSW contains a disabled mask (X'44' or X'04') and all other system activity will have stopped.

Recognizing that there is an *enabled* loop is often the most difficult step. Enabled loops are often quite large and may encompass several distinct operations - I/O events, SVCs, module linkage, etc. Because the loop is enabled, it is often interrupted, preempted and eventually resumed many times. This makes it difficult to recognize the loop pattern. Following are some indicators of a potential enabled loop:

- The current PSW has an enabled mask, X'07', in the first byte and the instruction address portion $\neq 0$. This alone does not prove there is a loop, but the information may help your analysis of the problem later.

- The system trace table shows a repetitive pattern of events, for example, SVCs issued from the same virtual addresses, or dispatcher entries for virtual addresses that are relatively close together. Determine if the entries are related to the same address space by using the ASID field. If so, you can now examine the task and control block structure indicated by the trace entries. The chapter on "MVS Trace Analysis" in Section 2 should prove helpful.

- Many tasks (TCBs) or address spaces (ASIDs) appear to be bottlenecked waiting for some resource(s). This can be determined by using the ANALYZE subcommand of IPCS or by using the QCBTRACE or GRSTRACE option of AMDPRDMP and analyzing the output. If there appears to be a bottleneck, determine what job owns the resource(s) and what that job is currently doing. It may be that the job that acquired the resource(s) is in an infinite enabled loop; therefore, when other jobs request the same resource(s), their requests cannot be satisfied, which eventually causes a major performance throughput problem. See the chapter on "System Execution Modes and Status Saving" in Section 2 for how to recreate the job's current status. A reconstruction of the PSW and registers helps you to determine if there was an enabled loop.

- TCB/RB structure analysis. Look for unusual or long RB structures chained from TCBs. These may indicate a loop that includes several levels of supervisor linkage.

Enabled Loop Exception:

The system resources manager (SRM) constantly monitors resources, gathers data, and analyzes the system. Periodically, SRM uses a timer interrupt in order to gather its statistics. This timer interrupt occurs even when the system is in an enabled wait condition. Because of this, the enabled wait is often referred to by operators as an enabled loop. (They observe the "WAIT" indicator from the console, followed by a burst of activity (SRM processing), followed by the "WAIT" indicator, etc. It may even be possible to enter certain operator commands.) However, this is really an enabled wait condition and analysis should proceed according to the discussion on "Enabled Waits" in the "Waits" chapter earlier in this section.

The dump you are analyzing may show the system trace table containing a no-work wait (070E0000 00000000) PSW followed by a timer interrupt, SRB dispatch, MP communication, etc. This pattern indicates an enabled wait condition, not an enabled loop. (See the "Pattern Recognition" topic in the "Miscellaneous Debugging Hints" chapter in Section 2.)

Once you have determined the type of loop, the following analysis procedure should help determine what problem is causing the loop.

Objective 1 - *Who is looping?*

The PSW and registers saved at the time of the dump indicate the active work. (See the chapter "Global System Analysis" in Section 2.) The register save areas in the LCCA/PSA indicate important environmental data at the time of the last I/O interrupt, external interrupt, etc. (See the chapter "System Execution Modes and Status Saving" in Section 2.)

The PSA indicators contain valid information about disabled loops. Also remember the recovery areas active at the time of the loop are valid and may provide hints as to the current process. (See the chapter "Use of Recovery Work Areas" in Section 2.)

Unlike the disabled loop situation, the enabled loop may not have the current registers associated with it. This is true if the loop was interrupted and new processing was initiated before the dump was taken. For the enabled loop, find the current registers and status from the ASCB/ASXB/TCB/RB structure and the associated save areas (for example, IHSA). The chapter "System Execution Modes and Status Saving" in Section 2 will be helpful for this phase.

Objective 2 - *What is the system mode?*

It is important to know whether the system is in SRB or task mode and the implications of these modes. In all cases of true disabled loops, the PSW, LCCA, and PCCA contain valid status indicators such as the last dispatched routine. The old PSWs reflect the last interrupt status. The register save areas in the LCCA are valid. The LCCA+X'21D' set to 1 indicates SRB mode; set to 0 indicates task mode. Additionally, PSAMODE (PSA+X'49F') indicates the state of the system. If set to X'00', the system is in task mode; if set to X'04', the system is in SRB mode. This byte can also contain flags to indicate wait mode,

spin mode, or non-preemptive mode. The ASCB NEW/OLD and TCB NEW/OLD pointers reflect the current task. If the TCB OLD pointer is zero, the system is in SRB mode or possibly in supervisor mode - that is, dispatcher or supervisor recovery. The discussion in the "System Execution Modes and Status Saving" chapter in Section 2 is useful.

You can use the IPCS subcommand STATUS CPU to determine the mode of the unit of work on each CPU.

By scanning the system trace table, you will be able to determine system events leading up to the loop. See the chapter on "MVS Trace Analysis" in Section 2.

SYS1.LOGREC and the in-storage LOGREC buffer may contain indications of previous occurrences of the loop (records with X'071' completion codes) or records of previous errors in the currently looping process that could possibly contribute to the current loop. See the "Locking" chapter and the discussion on LOGREC in the "Use of Recovery Work Areas" chapter in Section 2.

Objective 3 *- What is the extent of the loop and why is the system looping?*

Using the current PSW and the global data areas in combination with the general purpose registers, you should be able to determine the extent of the loop. One register often contains the key to a loop-causing value. Try to isolate that one register. It may be necessary to inspect the actual object/source to determine the basic logic in case there is an encoded loop that is supposed to end when a certain value is reached. If that value cannot be reached for some reason, the loop will not end.

Isolating the cause of the loop is important in loop analysis. Once the cause is isolated, you can proceed the same as with a system-detected error such as a program check.

Objective 4 *- Determine the cause of the error - how is the value that is causing the loop developed?*

To determine how the bad value was developed, it is necessary to back through the logic leading to the loop. Be aware of bad control blocks. Look at the bad value itself and the areas from which it was developed. Try to determine if the value is the result of a storage overlay or if it was calculated from bad logic. See the "Pattern Recognition" topic in the "Miscellaneous Debugging Hints" chapter of Section 2 to help make this determination.

In addition to bad control blocks and data fields, consider the loop control mechanism used for encoded loops. Often a common cause of problems is that the BCT instruction is used and the loop control register contains a negative value. Scanning the active registers at the time of the dump often aids in discovering this type of problem.

Figuring out how the erroneous field could possibly contain the value it does is the most challenging part of the process. Again, the contents of the field often provide the clue to determining the error-causing process.

Also, consider how serialization is accomplished for the field in question. Is it possible for several processes to be updating the field simultaneously? The system trace helps you recreate recent processes, but you also must understand the modes and structure of the code that updates the field. (Your work in Objective 2 should be helpful.)

It is possible that the code setting up the field was physically interrupted and, because it was non-reentrant or the logic was faulty, another process updated the field or control fields and subsequently caused the first process to encounter unexpected data.

TP Problems

A common problem in teleprocessing (TP) environments is incorrect data, which may affect one terminal or an entire component. The symptoms include no data, wrong data, or too much data, but the general problem symptom is that something is wrong with one or more messages. The problem is usually not tied directly to a component or access method, as a program check would be; often an error message is from a component not directly causing the problem.

Typical symptoms are:

- An error response from an application that suggests incorrect data was entered from a terminal, when in fact the data was correct
- A "hung" terminal - the system will not respond
- Wait states, in which message traffic gradually dies off
- Incorrect characters in a message (the data may be going into or out of the system)

This chapter discusses TP problem analysis, in the following topics:

- Message Flow Through the System
- Types of Traces

Message Flow Through the System

Data exchanged between programs in the system and terminals follows a route through several components. The first step in solving "typical" problems is to determine where along that route something is happening incorrectly.

By far the most valuable tools for doing this are the traces in the various components. To use the traces effectively it is necessary to understand how messages flow through the system. For example, consider a message from an ACF/TCAM application to an ACF/TCAM terminal. The path might go from the application program buffer to an ACF/TCAM queue data set, to an ACF/TCAM buffer while ACF/TCAM processes it, over the channel into the 3705, then into an NCP buffer, and finally over a communications line to a terminal. Traces allows the message to be checkpointed at certain spots along the path; therefore, understanding the path is vital to knowing what traces to use and what you should see for a message that flows correctly.

To use traces effectively you must also understand how components refer to terminals or lines and how they communicate with each other regarding these terminals and lines. Terminals or lines are identified in traces by a line control block (LCB), a logical name, a network address, a polling/addressing sequence, or a subchannel address. Not only must these relationships be known in order to use multiple traces, but certain correspondences must be correct in order for data to move through the network.

When using traces, the general approach to a problem of incorrect data is to track the data flow from a point where everything was all right to a point where the messages stopped or were incorrect. Messages that are flowing correctly can be used to establish time relationships between different traces. Then each message can be followed along its route past each checkpoint, with the goal of isolating a gap between two checkpoints where the message stopped or became bad. The next step is to focus on this narrower area to learn what is wrong.

If a message stops, what is wrong or what is missing? How does the flow up to that point compare with a normal flow? You must understand what resources and what processes are required for a message to move from where it appeared last to where it should have appeared next. What buffers and/or control blocks would have been used? Were they available? A single terminal or all terminals may encounter a "wait state," and it is necessary to dig into the component to determine what processing has taken place and what condition or resource is preventing further processing. The *ACF/TCAM Diagnosis Guide* should be referenced for problem isolation in ACF/TCAM.

If a part of the data moving through the system becomes bad, the traces should isolate a component or an interface over which it was transformed. Comparison with normal message flow will indicate whether any change at all should have taken place. If no change should have occurred, an overlay ("clobber") or incorrect pointers to data buffers may be the problem. The exact amount and positioning of bad data should be determined, for it might provide an obvious correlation with other known variables such as a buffer length. If some transformation normally occurs to a message, the controlling process under which it is performed must be examined. What could cause an incorrect transformation of data? Examples are translate/edit tables or mappings from one resource name into another name, such as mapping the logical network name into the network address.

Types of Traces

The following is a summary of TP-related traces.

GTF Traces

The following GTF trace options can be used to provide traces of TP-related activities. (Refer to *Service Aids* for additional information.) The GTF traces include:

- SSCH and IO trace options - record start subchannel and resume subchannel operations and I/O interruptions for either an emulation mode subchannel or a 3705 controller.

- RNIO trace option - records the header and first few bytes of data for each PIU coming into or going out of ACF/VTAM.

You can correlate the GTF traces with the ACF/VTAM, ACF/TCAM, and NCP and EP traces to compare relationships of the system and TP activity.

ACF/VTAM Traces

ACF/VTAM provides several kinds of traces to record the flow of path information units (PIUs) between nodes in the ACF/VTAM network. (Refer to *ACF/VTAM Diagnosis Guide* for additional information.) The ACF/VTAM traces include:

- Buffer contents trace - records the contents of inbound and outbound message buffers.
- I/O trace - provides a chronological history of all I/O activity between ACF/VTAM and a particular network node.
- Line trace - records the status of a line each time that data is sent or received along the line.
- Generalized PIU trace - records the PIU trace data obtained by the NCP.
- Transmission group trace - records the sequence of PIUs being sent through a transmission group.
- SMS (buffer use) trace - records information about the use of buffers.

ACF/TCAM Traces

ACF/TCAM provides several service aids for diagnosing ACF/TCAM problems. (Refer to *ACF/TCAM Diagnosis Guide* for additional information.) The ACF/TCAM traces include:

- Channel I/O interrupt trace - provides a sequential record of the I/O interrupts occurring for specified non-NCP line addresses or for the channel addresses of 3704/05 controllers.
- NCP line trace - provides a sequential record of the level two (I/O) interrupts occurring for specified NCP lines.
- PIU trace - provides a sequential record of all PIUs sent to or received from NCP network resources.
- Buffer trace - provides a record of ACF/TCAM buffer contents before and after message handler (MH) processing.

NCP and EP Traces

The network control program (NCP) and emulation program (EP) provide several diagnostic aids to diagnose difficulties in network operations. (Refer to *IBM 3704 and 3705 Control Program Generation and Utilities Guide and Reference* for additional information.) The NCP and EP traces include:

- Address trace facility for network control mode - records the contents of selected areas of communications controller's storage and registers for successive interrupts in the NCP.
- Line trace facility for network control mode - records the operating parameters of a line each time that a level two interrupt occurs for the line.
- Line trace facility for emulation mode - records the operating parameters of a line each time that a level two interrupt occurs for the line.

Performance Degradation

This chapter describes how to investigate performance degradation problems. It is not intended to serve as a tuning guide or as a reference for general performance analysis (which should be performed through SMF, GTF, etc.)

The following points should be considered when a problem is suspected in the operating system itself or in the manner in which applications use the operating system.

Operator Commands

When a bottleneck or system failure, hardware or software, is degrading throughput, the following DISPLAY operator commands can help identify the source of degradation and, possibly eliminate it.

D A,L — Displays current system status. A job step with a name of STARTING indicates initiation has not successfully completed.

D R,L — Displays any outstanding requests. Operator action is required (for example, to mount a volume). Other jobs may need to wait until action has been taken.

D M — Displays configuration information. The loss of a hardware component (for example, a channel path) may have been noted on a hard copy console and missed by the operator.

D SLIP — Displays the name and status of current SLIP traps. Display those traps that are enabled (D SLIP=xxxx (where xxxx is the trap id) to see if any are PER traps (PER-IF, PER-SA, or PER-SB). If an enabled PER trap with an action other than IGNORE exists, check with the system programmer to determine if it should remain enabled or if it can be disabled.

If a resource queue "snooper" program exists, it should be started and output examined to find any ENQ bottlenecks.

If no such program is available:

- You can issue the DUMP command with options GRS, SQA, CSA, and NUC. Use the IPCS ANALYZE subcommand to locate contention caused by ENQs, suspend locks, or I/O devices.
- Or, you can obtain a dump of the global resources serialization address space and the nucleus. Use the PRDMP service aid (with the GRSTRACE, Q, or QCBTRACE option) to print the dump, and examine the resource queue.

Use the job entry subsystem display commands to find the status of jobs, queues, printer setups, requirements of SYSOUT data sets, etc. to find reasons why JES2 is not able to schedule work. Some JES2 commands that may be useful are shown in Figure 4-1.

$D	J1-9999 S1-9999 T1-9999	Status of jobs, started tasks, or time-sharing users. If a range of jobs has been held they may be released using $AJ.
$AJ		Release jobs.
$DF		Status of output forms queue.
$DU,PRTS		Status of printer setup characteristics.
$TPRTN		Change setup to needs of queue output.
$LJ1-9999,H		List held SYSOUT data sets.
$OJ1		Release held SYSOUT data sets.
$DQ		Display queue.
$AQ		Release queue.
$AA		Release all jobs held by a $H command.

Figure 4-1. JES2 Commands for Status Information

If the use of previous commands does not make it obvious why JES2 is not scheduling work, take a dump of the JES2 address space. Print the SYS1.DUMPxx data set to help determine the problem.

Note that you can use the IPCS subcommand VERBEXIT JES2 to format JES2 control blocks.

Find the number of the IPS member that should be active and issue T IPS=na to ensure that it is active. Print the IPS member in SYS1.PARMLIB and analyze the IPS for an explanation of degraded service. Then, enter the W command to print the system log to obtain the history of system execution.

Figure 4-2 shows important hardware components used by the system that should be understood when a degradation problem is suspected.

Dump Analysis Areas

The following areas in a storage dump may provide a starting point for further analysis. Problems in these areas may indicate a bug or some unexpected use of system services.

1. ENQ/DEQ - A check of ENQ/DEQ's processing queues may indicate contention problems. A queue of many QELs off a particular QCB should be explained. An indication of a possible problem is a mixture of shared and exclusive requests intertwined for one resource. The state (running/waiting/swapped-out/etc) of the holder of the resource should be determined.

   You can use the IPCS subcommand ANALYZE to do ENQ contention analysis, and the IPCS subcommand VERBEXIT GRSTRACE to format global resource serialization control blocks and all QCB and QEL data.

The QCBs defining all enqueued resources are chained from either the local or global hash table. To locate the hash tables:

- CVTGVT (CVT+X'1B0') points to the global resource serialization vector table (GVT), which resides in the nucleus.
- GVTGVTX(GVT+X'10') points to the GVT extension (GVTX), which resides in the GRS address space.
- GVTXGQHT (GVTX+X'A4') points to the global queue hash table in the GRS address space.
- GVTXLQHT (GVTX+X'A8') points to the local queue hash table in the GRS address space.

Additionally, if the ENQ/DEQ request was initiated from this system, you can locate the resources and the requesters of resources (QELs) for an address space by following the queue of QELs chained from the fields ASCBGQEL (ASCB+X'110') or ASCBLQEL (ASCB+X'114'). If the request was initiated from another GRS system, the QELs are chained from the SYSID/ASID hash table, which is pointed to by GVTXSAHT (GVTX+X'AC').

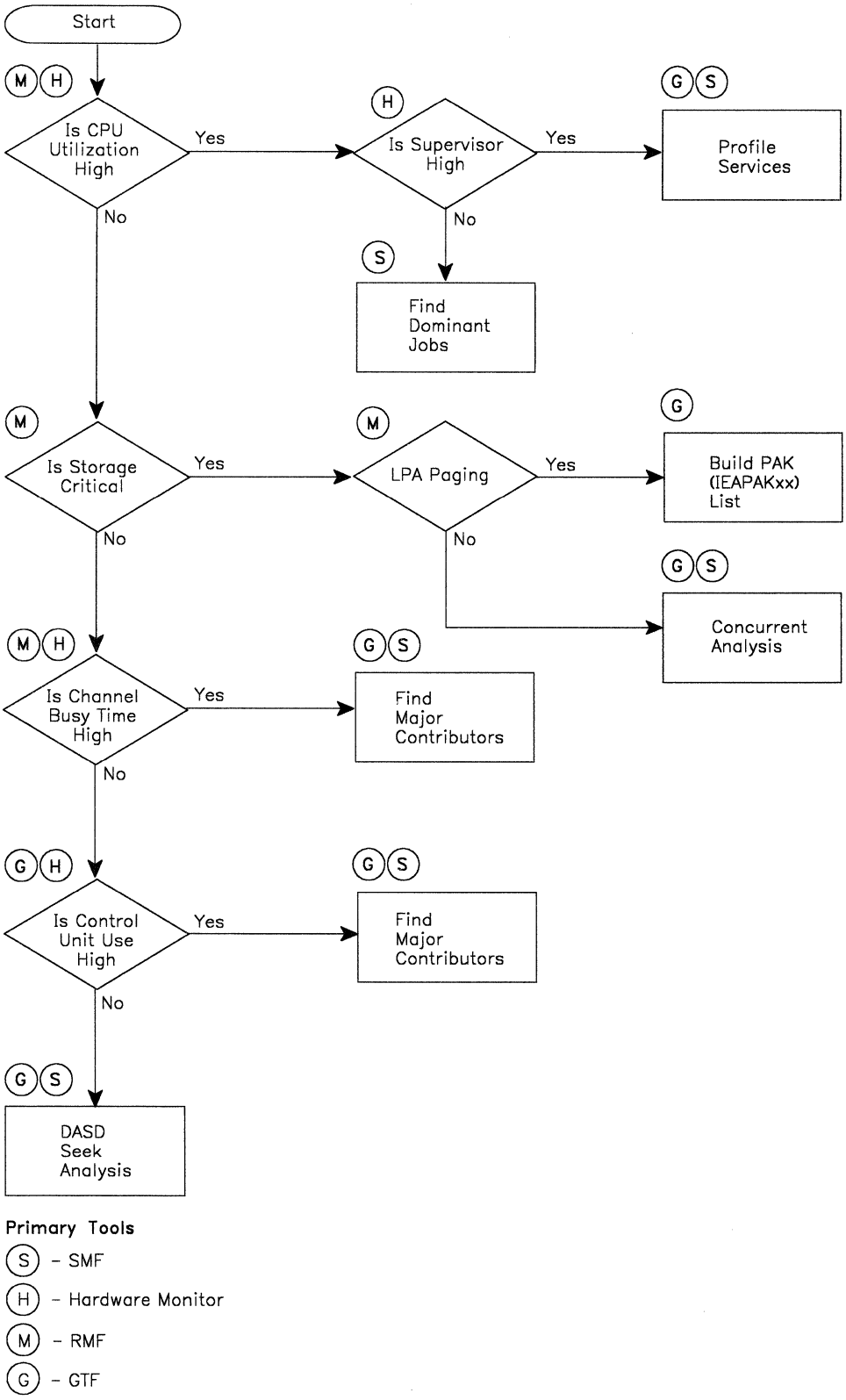


Figure 4-2. System Use of Hardware Components

2. IOS storage manager queues should be inspected. The anchors for the various pools (small, medium, and large block pools) are located in IOSVSMGR at external symbol IOSVSHDR, which should show up in a NUCMAP pointed to by the IOCSMHDR field in the IOCOM (CVT+X'7C'). Generally there should be one 4K page of small blocks (used for IOQEs) and two 4K pages of medium blocks (used for RQEs). Examine the large blocks in detail. If the system was quiesced, there should be six or more 4K pages of large blocks and all blocks should be on the free queue. Many heavily-loaded systems require 8-10 pages of large blocks. If the actual number is much higher than this, determine the ASID that each in-use block is assigned to. The first two bytes of the storage manager header contains the ASID address. The storage manager header is the last eight bytes of the block. System address spaces can have many blocks, but any user address space with a large number of blocks should be explained.

   You can use the IPCS subcommand IOSCHECK for analysis and formatting of IOS control blocks.

   Common problems are: I/O loop, I/O errors, and storage not being freed at I/O termination time. These page frames occupy real storage, which depletes the pool of available real storage and possibly causes excessive paging.

3. Check page frame table entries (PFTEs) for large fix counts. The CVT+X'164' contains the address of the PVT; PVT+X'88' contains the address of the address of the PFTE origin. Large fix counts may indicate a page fix macro loop, or page fix without page free. Frames allocated to a private area space may indicate a user error. Try to analyze the contents of the page for a clue as to who is page fixing without page freeing.

   You can use the IPCS subcommands ASMCHECK and VERBEXIT ASMDATA for analysis and formatting of ASM control blocks.

4. Check PFTEs for bad frames caused by hardware storage errors that rendered these frames unusable (in PFTE+X'9', the X'01' flag should be set if this is the case). Contact hardware personnel to determine if a machine malfunction has occurred.

5. CDE (contents directory entry). These blocks represent modules that are loaded into virtual storage. CDEs on the active link pack area queue reside in SQA and the queue is anchored by a header field at CVTQLPAQ (CVT+X'BC'). The loaded module's name and starting address reside in the CDE. Modules with starting addresses in the range CVTFLPAS to CVTFLPAE or the range CVTEFLPS to CVTEFLPE were members of an IEAFIXnn list. Modules with starting addresses in the range CVTMLPAS to CVTMLPAE or the range CVTEMLPS to CVTEMLPE were members of an IEALPAnn list. Members of these lists occupy virtual storage even when the modules are not in use. Members of an IEAFIXnn list occupy fixed virtual storage. If fixed storage or fragmentation is a problem, moving these modules to LPA can provide a partial solution.

6. In a quiesced system, the number of paging requests received should equal the number of paging requests completed by ASM. The fields ASMIORQR (ASMVT+X'28') and ASMIORQC (ASMVT+X'2C') in the ASMVT represent the number of requests received and completed, respectively. The difference between the two counts represents requests not completed. A large number of uncompleted requests can indicate ASM is either not processing at all or is taking considerable time for each operation. Examine the PAT (page allocation table) to determine whether the page data sets are almost full. Also examine ASMERRS (ASMVT+X'7C'), PAREFLG1 (PARTE+X'9'), and the IOSB for paging requests (IOSCOD=X'D') to determine if I/O errors have occurred and the data sets are no longer in use.

7. The use of ECSA and CSA should be examined. If ESQA is depleted, requests for ESQA storage are filled from ECSA. If ECSA is also depleted, then requests are filled from SQA, and finally from CSA. SQA overflow into CSA can be determined by examining field GDACSACV. This field indicates the total amount of overflow for both the extended and non-extended areas. To locate field GDACSACV:

   - CVT+X'230' points to the GDA (global data area)
   - GDACSACV is at offset X'8C' into the GDA

   Field GDACSACV, when divided by 4096, indicates the number of frames that were page fixed due to the SQA overflow. Also, many users of pageable CSA use page fixes. In this case, fragmentation, if present, can cause performance degradation.

8. A possible real frame shortage can be indicated by inordinately large counts in the RCE fields: RCERSQA (a count of the number of times the SQA reserved frame was allocated), and RCEDFRS (a count of the number of times a frame allocation that was deferred, because of a lack of frame availability, has been satisfied). These counts by themselves mean little, but can be of some use when analyzing an overall problem.

Incorrect Output

The problem of missing, unexpected, or erroneous output is one of the most difficult. This incorrect output might take the form of a message on the console log or in SYSOUT, or an incorrect total in a report. There is usually very little documentation that assists the debugger in analyzing incorrect output.

Initial Analysis Steps

To resolve the problem of missing or incorrect output the analyst must have a complete understanding of the job environment. There is no fast, clear cut approach to these errors. This section only tries to assist your thought processes as you begin to work on a problem of this type.

There are four basic categories of incorrect output: missing, unexpected, erroneous, or a combination of these. The steps in resolving the problem must take the category into account.

Initially, consider the following steps:

1. Gather all possible documentation. You will probably need additional information as you begin to understand the problem in more detail.

2. Consider all recent hardware and software changes to the system and to the application(s) if relevant. A change to an application that updates a data base affects all other data base users.

3. Remember that output requires input. Consider the possibility of bad input.

4. Consider whether the problem is associated with some new function or application. Most incorrect output errors occur in the installation and test phase.

Isolating the Component

Next, attempt to locate the component causing the error. Do this by thinking through the flow. Listed below are some questions that might assist you.

- Is the problem related to a user function or application? If yes, have there been recent changes or is testing still in progress?

- Is the job control language correct? Have there been recent changes to the JCL?

- Have any user exits been added or modified?

- Have any user supervisor calls (SVCs) been added or modified?

- Are there operator interactions that could affect the input/output?

- From which access method or function is the output expected? Some examples are: JES, VSAM, BTAM, ACF/TCAM, and WTO.

- Was there any cross-address-space communication involved in the data movement? In MVS, most telecommunication requires data passing between address spaces.

- Is there any evidence of I/O error activity? Refer to the console log and LOGREC data.

- Do you have a storage dump, or should you obtain one? See the chapter on "Additional Data Gathering" in Section 2.

- Would a trace be helpful in understanding the flow? Consider tracing the activity with system trace or GTF.

Many of the above questions have to be answered in order to get a better understanding of the problem area. In many cases, the problem has to be recreated with various traces or traps. These questions help to determine what data is needed to solve the problem.

Analyzing System Functions

To solve an incorrect output problem, you must understand the mode of operation and the processes required to accomplish the function in question.

The first question must be the following: where does the output originate? Then you must be able to verify that the activity did occur. There must be some means for understanding the path the data should take from the origin to the final location (device).

Consider the following example:

1. A TSO user invokes his program which should write a message to the terminal and then wait.

2. The program waits after the I/O but no message appears.

3. What are the system functions involved?

   a. A language translator and the linkage editor that created the load module.

   b. OPEN code necessary to complete the link between the device and the user PUT macro.

   c. TSO TIOC flow. The user issues PUT which branches to the TIOC module IGG019T4. This module issues TPUT. What is the TPUT path through TIOC?

   d. TSO TIOC interfaces with TCAM. What is the data path through TCAM?

   e. TCAM interfaces with the I/O supervisor. Can evidence be found of the SSCH? What types of trace would be helpful?

In this example it may be necessary to take a series of dumps to resolve where the message was lost. But first be certain that the correct message is in the correct buffer at the time of the user PUT macro.

It could be necessary to apply this type of thinking all the way down to the CSECT level.

Appendix A. Stand-Alone Dump Analysis

This appendix contains a procedure that has been used successfully in stand-alone dump analysis. It is part of the course material in Field Engineering classes that teach MVS problem determination. This procedure does not attempt to cover all situations but it can be used as a guide through major status areas until you become thoroughly familiar with the system.

Overview

Stand-alone dumps are generally taken by the operator when:

- The system has stopped in a solid wait state with a wait state code.
- There appears to be a system loop.
- The system is not running or is running slowly.

Usually the 'Title From Dump' reflects what the operator thought happened. You can use the IPCS subcommand LIST TITLE to see the title of the dump.

Before becoming too involved in the problem itself, it is a good practice to get some feel for the status of the system at the time the dump was taken. Some valuable system status indicators can be obtained from the formatted section of the dump. Indicators can be obtained from the formatted portion of the dump under "System Summary" (produced by the SUMMARY control statement) and CSD, PSA, LCCA, and PCCA (produced by the CPUDATA control statement). Although it is seldom that any one indicator definitely points out the problem, when all indicators are noted and analyzed, a pattern might emerge that points the problem solver to the proper area for further investigation.

The enabled wait generally occurs as a result of the lack of some critical system resource.

You can use the IPCS subcommand STATUS CPU to determine whether the system is in an enabled wait. If the PRINT statement of PRDMP is used, PRDMP identifies the current task. If the current task is the wait task, the message "Current Task = Wait Task" might appear.

If it appears you have an enabled wait condition, read the chapter on "Waits" in Section 4 of this book before proceeding with your analysis.

The system can appear or actually prove to be bottlenecked because the operator cannot communicate with MVS. This is the sign of a problem almost anywhere in MVS, but an error in the communication task or its associated processing might be the direct cause. The communication task runs as a task in its own

address space and is identified by a X'FD' in the TCBTID field (TCB+X'EE'). By inspecting the RB structure associated with this task, you can determine the current status. It is not unusual to find one RB with a resume PSW address in the LPA and an RB wait count of one. If more than one RB is chained from the TCB and you could not enter commands, analyze the RB structure as this is not a normal condition.

You can use the IPCS subcommand COMCHECK for communications task analysis.

Remember that communications task processing is very dependent on the rest of the operating system. Probably some external service or process has caused the communications task to back-up, and this possibility should be investigated.

For the system to continue execution, the major components must be operational. If any critical system components such as the master scheduler, ASM, and JES2 terminate abnormally and fail to recover, the system cannot continue normal operation. Usually this can be determined from the records in SYS1.LOGREC. However, check the TCB summary in the formatted section for completion codes.

The presence of a TCB completion code does not positively identify the associated task as being inoperative. It is possible that the completion code is residual and the task has recovered. The presence of a completion code makes the task suspect, however, and should be investigated.

Unless the operator STORE STATUS command was issued before taking the dump or the "Title from Dump" reflects a WSC (wait state code), it can be difficult to determine if a WSC exists and what it is if it does.

If however, the WSC PSW is dispatched by NIP during IPL, it is generally located in one of two places:

- In the MCH new PSW if a program check occurred prior to RTM initialization.
- In the nucleus vector table (NVT+X'E0') in the case of a system-detected error during the NIP process.

The other WSCs (they are few in number) issued by the system are dispatched by the communications task and ASM. The current address space should identify who loaded the WSC PSW; WSC PSWs are issued when the system determines that it cannot continue. They are usually preceded by other error indicators that should be investigated along with the WSC.

Note: A valid WSC looks like: X'000A0000 nnnnnxxx'

where:

nnnnn is the reason code
xxx is the wait state code.

A disabled wait normally has a wait state code associated with it. If so, the messages and codes should contain a problem description.

You can use the IPCS subcommand STATUS CPU to obtain wait state codes and reason codes.

If there is no wait state code, the system trace table should indicate the last sequence of events leading to the wait state condition. Probably a bad PSW (wait bit on) has been loaded.

If no valid WSC exists and if the PSW reflects the wait bit, is disabled, and the STORE STATUS registers are not equal to zero, suspect: a user or Field Engineering trap or a SLIP trap (with a wait state code of X'01B'), a bad branch, or system damage. Examine the trace table and attempt to define the events that led up to the wait condition. Was the last entry an SRB dispatch or an SVC I/O interrupt? Using the PSW address, determine the entry point of the routine if possible.

The PSA is a system area whose status indicators are dynamically changing. The status indicators reflect the condition of the system the instant the dump was taken. Taken out of context, they can be misleading.

Therefore, find out if the operator entered a STORE STATUS command and keep in mind that the status could have been stored at any time, but was probably stored just before the dump.

Note: If you IPL the stand-alone dump program from the system control (CC-012) frame, it is not necessary to perform the STORE STATUS operation. Status is automatically stored when stand-alone dump is invoked from the CC-012 frame and automatic store status is on.

The best evidence that the operator issued STORE STATUS is the content of 'Current Registers and PSW at Time of Dump.' This is because the stored status is put in the PSA +X'100' and the registers are put at PSA +X'160- 1FF', and SADMP program reads this area as the current PSW and registers and writes them to the dump data set. On a UP, the formatted current data will be the same as in the PSA. On an MP system, however, the SADMP program issues SIGP to another processor to store status. The STORE STATUS command always stores in the unprefixed PSA at location X'100'. This means that the unprefixed PSA will contain the registers and PSW from another processor. If the SADMP program did not save the STORE STATUS data before issuing the SIGP instruction to the other processor, the data from the operator's STORE STATUS command would be overlaid and the contents lost.

If PSW+X'01' = xE or xA, the PSW is a wait PSW. If PSW+X'00' = X'04', the system was disabled. If PSW+X'00' = X'07', the system was enabled. Determine whether the PSW contains a WSC or an address. Then determine what key the PSW reflects. PSW+X'01' = X'xC' or X'xE' where the x = key, as follows:

0 = supervisor
1 = scheduler/JES2/JES3
5 = IOS, data management, actual block processor, O/C/EOV
6 = TCAM/VTAM
7 = IMS
8 = virtual problem program
9-F = V=R problem program

In the PSW - when bit 32 (the A bit) is 0, the program in control is executing in 24-bit addressing mode; when bit 32 is 1, the program in control is executing in 31-bit addressing mode.

Loops can be either disabled or enabled. The best way of determining which has occurred is by examining the PSW, the loop recording option table, and the system trace table.

Recorded addresses that fall within the SRM code are usually not indicative of a loop because this code is entered periodically as a result of a timer interrupt. This signifies, however, that the system does enable for interrupts and you can treat the error as an enabled loop.

Note: If the only addresses the operator furnished or the loop recording option recorded are in the timer or SRM code, check that it is not really an enabled wait condition.

The typical disabled loop is quite short, whereas the enabled loop covers a wide range of addresses. Be careful that the recorded addresses that may reflect a short loop are not a 'loop within a loop'. Scan the system trace table and try to determine if a pattern of activity exists. Look for SSCHs to the same device, SVCs from the same address, program checks occurring frequently for other than page or segment faults, or any repetitive activity. If no pattern exists, try to correlate the last trace entry with what you already know about the loop (for example, I/O interrupts, a loop in an IOS or SRB dispatch, and a loop in the nucleus in some routine which is entered via an SRB).

The enabled loop usually reflects a wide range of addresses and can even span address spaces between a user and the system address spaces. (However, an enabled loop can be shown by repeated I/O and external interrupts for the same task within a small range of addresses.) An examination of the system trace table usually shows some pattern of activity that is recognizable as a loop.

Be especially suspicious of a SVC 0D or SVC 0A for the same size area, SVC 33, SVC 4C, and SSCHs to the same device with the same IOSB address in register 1.

System trace table entries with SVC 0D and/or SVC 33 in a stand-alone dump usually mean that some task is abending and the system is attempting to recover and purge the task from the system.

If any address within the loop points to the spin lock manager (module IEAVELK), the problem is probably caused by someone requesting an unavailable spin lock. This signifies that another processor is holding the lock and failing to release it. There is a strong possibility that this indicates an overlaid lockword also. If not, the problem is on another processor. In either case, register 11 can point to the lockword requested and register 14 is the address of the requestor. Check the value in the lockword. See the topic "Locking" in Section 2 for valid lockword values.

If the lockword is overlaid, you must identify who overlaid it. It is possible that the lockword was overlaid in conjunction with some other problem.

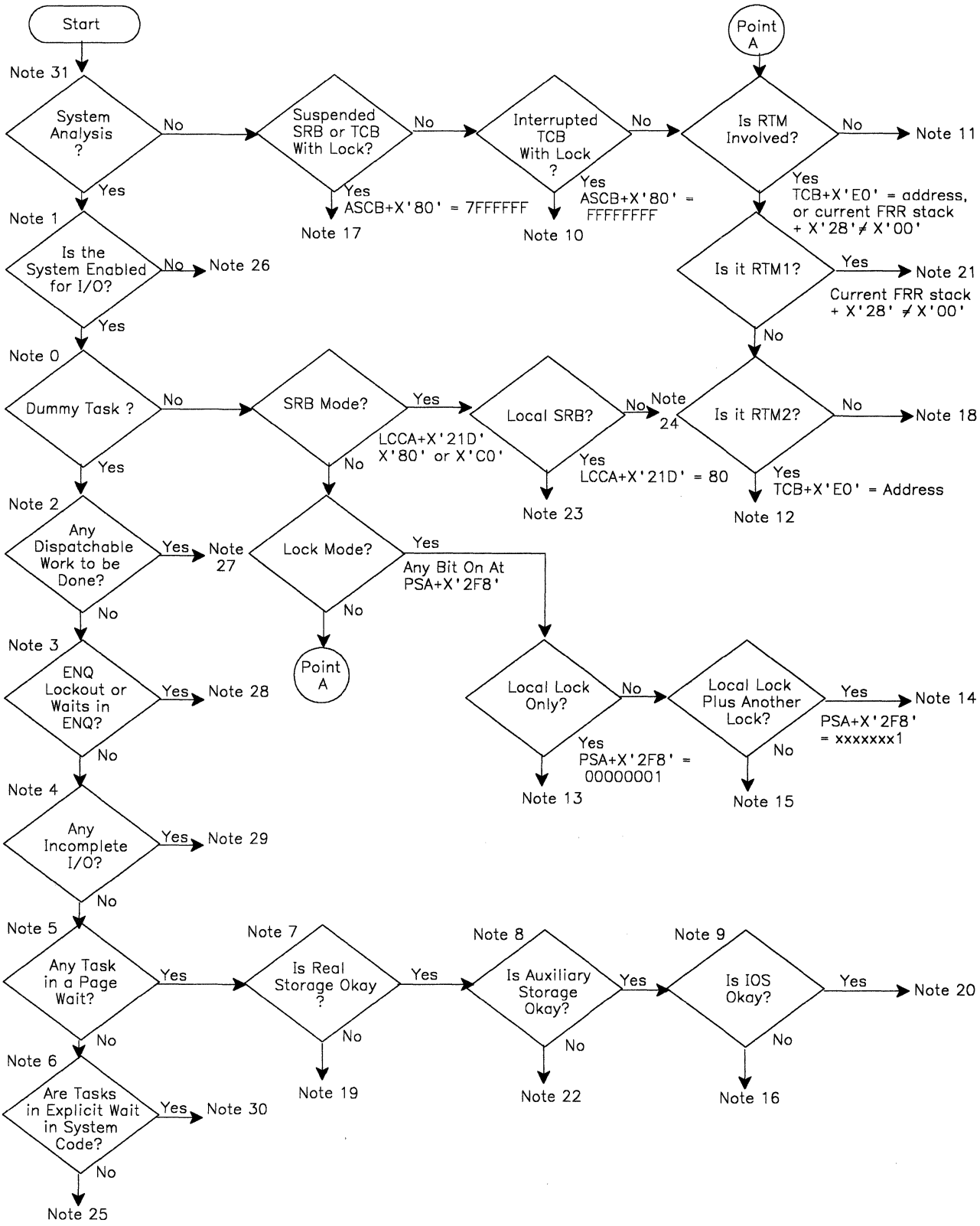


Figure A-1. Stand-Alone Dump Analysis Flowchart

Analysis Procedure

The following explanations correlate to the "Notes" in Figure A-1.

Note 0 - Dummy Task?

The dummy wait has been dispatched on the processor if the following fields contain the values shown:

STORE STATUS PSW = 070E0000 00000000
STORE STATUS GPRs = 0
PSATNEW = PSATOLD = 0
PSAANEW points to ASID 0
PSAAOLD points to ASID 1
PSASUP1 = 04 (dispatcher)
PSAMODE = 08 (wait)

Note 1 - System Enabled for I/O?

Is bit 6 on in the current PSW?

On any processor in the system, is control register 6 properly loaded to allow I/O interruptions for the associated I/O-interruption-subclass queue?

Note 2 - Dispatchable Work to be Done?

1. One of the first places to check for system dispatchability is the common system data area (CSD). For example, CSD+X'C'=X'40' indicates that most of the system is non-dispatchable. This bit can be set by SDUMP. Is any address space abending and in the process of taking an SDUMP? Check the TCB summary for completion codes.

   You can use the IPCS subcommand STATUS WORKSHEET to determine if SDUMP has set the system non-dispatchable.

2. Dispatchable work within an address space is indicated by:

   ASCB+X'80' = X'FFFFFFFF' or X'4FFFFFFF'

   ASCB+X'80' = X'00000000' (indicates the LOCAL lock is available)

   ASCB+X'D0' = SRB on the address space service management queue (ASCBLSMQ)

   ASCB+X'D4' = SRB on the address space service priority list (ASCBLSPL)

   ASCB+X'D8' $\neq$ 0 indicates ready TCBs not requiring the LOCAL lock.

   ASCB+X'DC' $\neq$ 0 indicates ready TCBs requiring the LOCAL lock.

3. The JES2/JES3 address space can contain work that should be passed to a waiting initiator or interface that has an address space for SYSIN or SYSOUT data.

4. Dispatchable work at the system level is indicated by SRBs queued to the global service manager queue (GSMQ) and the global service priority list (GSPL).

For (1), you must determine who set the bit on, who should have reset it and why the bit was set. It might be necessary to trap on the setting of this bit.

For (2), a X'7FFFFFFF' indicates that the holder of the local lock is suspended (for example, a page fault or CMS lock request). A CPUID value (such as X'00000040') indicates that the unit of work holding the local lock is currently running on that processor. Any nonzero value indicates that the lock is held and that TCBs requiring the LOCAL lock cannot be dispatched.

For (3), check the JES control blocks more closely.

For (4), determine why the dispatcher is not functioning.

Note 3 - Enqueue Lockout?

As in other systems, an exclusive enqueue prevents other tasks from using the same resource. However, in MVS, locks are now used frequently instead of an enqueue.

1. You can use either the IPCS ANALYZE subcommand or the IPCS VERBEXIT GRSTRACE subcommand to do ENQ contention analysis. Or, you can use the QCB format function (QCBTRACE, Q, or GRSTRACE option of print dump) to print the QCBs and check for exclusive enqueues.

2. CVT+X'1B0' points to the GVT.
   GVT+X'10' points to the GVTX.
   GVTX+X'A4' points to the global queue hash table.
   GVTX+X'A8' points to the local queue hash table.
   GVTX+X'AC' points to the SYSID/ASID hash table.
   ASCB+X'110' points to the global QEL queue.
   ASCB+X'114' points to the local QEL queue.

   Note: The GVTX, the hash tables, and the QEL queues reside in the GRS address space.

3. Any QEL reflecting exclusive control prevents any other task from using that resource. Any QEL reflecting shared status prevents any task requesting exclusive control from using that resource.

4. The *Debugging Handbook* defines some of the major and minor ENQ names.

Note 4 - Incomplete I/O?

You can use the IPCS subcommand IOSCHECK to do IOS analysis.

Label IOSVSHDR in IEANUC01 points to a pool of cells used by IOS to build the IOQ (I/O queue element). An IOQ chained to the UCB-X'4' indicates an I/O operation is in progress or has completed on that device. The flag bytes at UCB+X'6' and X'7' determine the current state of the device. The device is normally available when the flag bytes are zero.

IOQs chained to UCB-X'2C' indicate the I/O requests that are waiting to get started and also the I/O request in operation.

Note 5 - Is Any Task in a Page Wait?

Check the TCB RBs for a wait count not equal to zero.

RB+X'1C' = wait count
RB-8 ≠ 40 (FLAG1)
TCBFLGS4 ≠ '04'. This indicates a page-fault wait or a synchronous page fix wait.

Note 6 - Explicit Wait in System Code?

Does the address in the PSW fall within the nucleus or LPA code? Compare the address with a NUCMAP or LPA map.

Check the load list and CDEs for system modules that have been loaded into the private area.

You can use the IPCS subcommand WHERE on which you specify an address to obtain the name of the module that contains that address.

Note 7 - Real Storage Okay?

If a task remains in a page wait, it could indicate a shortage of page frames or a real storage failure.

You can use the IPCS subcommand VERBEXIT RSMDATA to format RSM control blocks.

The control blocks that contain status about the use of real storage are:

1. RSM control and enumeration area (RCE)

   RCE+X'84' = number of available preferred area frames below 16MB real

   RCE+X'88' = total number of available frames

2. RSM internal table (RIT)

   RIT+X'80' = free RPB count. Note that PCBs are obtained from RSM's RPB pool.

   RIT+X'8C' = address of the first PCB of a chain waiting for a frame.

   The RIT starts at the end of the RCE.

3. Page frame table (PFT) - shows the use of each frame of real storage available for paging.

Note 8 - Is Auxiliary Storage Okay?

If tasks are in a page wait and real storage is not a problem, the trouble could be within the auxiliary storage manager (ASM).

You can use the IPCS subcommands ASMCHECK and VERBEXIT ASMDATA for analysis and formatting of ASM control blocks.

ASM status indicators are:

1. ASMVT+X'28' = the number of paging I/O requests received
2. ASMVT+X'2C' = the number of paging I/O requests completed
3. ASMVT+X'30' = the number of swap I/O requests received
4. ASMVT+X'34' = the number of swap I/O requests completed
5. ASMVT+X'58' = the SRB address used to redrive work through ASM
6. SRB+X'1C' = the address of an 8-byte parameter list that contains the addresses of the first and last I/O requests to be redriven through ASM
7. IORB+X'03' indicates whether the I/O requests on the IORB have been passed to IOS.

If the number of I/O requests completed is equal to the number of I/O requests received, ASM has no outstanding work. If I/O requests have been started but not completed, determine what has happened to the I/O. If ASM's redrive SRB parameter list is nonzero, the SRB has been scheduled. Determine what the dispatcher has done with the SRB.

Note 9 - Is IOS Okay?

If all IORBs are idle (IORB+X'03', bit 0 is zero), then IOS has completely processed all the I/O that ASM has started.

You can use the IPCS subcommand IOSCHECK for analysis and formatting of IOS control blocks.

Note 10 - Interrupted TCB?

The condition that caused the TCB holding the local lock or local lock and at least one cross memory services lock to be suspended has been resolved. The save area to be restored upon dispatching is the IHSA.

A TCB holding the local lock, or local lock and at least one cross memory services lock has been interrupted by a higher priority task. The save area used for redispatching is the IHSA. See "System Execution Modes and Status Saving" in Section 2 of this manual.

Note 11 - Not RTM?

Without the detection of a failure by MVS, which would have caused entry into RTM, check the following. If the system trace table in the stand-alone dump reflects the same task in most of its entries, this could be normal operation or the task could be in a loop. Check the following for status information:

LCCA
PSA
PCCA
System trace table
TCB
RB/SVRB

You can use CBSTAT for selected ASCBs and TCBs to check for RTM activity on a task-level address space.

If no failure information is found (the system appears to be running normally), the problem might be that another task or address space should be running and is unable to. Check the following for status information:

1. Check each address space that is expected to be running to find out why it is not running. The information about each address space and task within that address space can be found in: ASCB, ASXB, TCB, and RB/SVRB.

2. Or, check the total system to find out why other work is not being run. Check the status of the system resources:

   ENQ lockout of data sets
   I/O failures
   RSM or ASM failure
   Waits in system code for other system resources (such as buffers)

If you are checking other than the current task, the TCBs could be dispatchable, but not yet dispatched. If the task is non-dispatchable (non-dispatchability bits on in the TCB), this can indicate an error situation. Or the task could be simply waiting (indicated by a wait count in the current RB). Check the dispatchability flags in the following control blocks for status of this task or select another address space or task and continue at Point A.

Status information can be found in: ASCB, ASXB, TCB, and RB/SVRB.

Note 12 - RTM2, Yes.

The most important place to find information about abend codes is *System Codes.*

The RTM2 work area address is stored by RTM2 in TCB+X'E0'. Every system dump (SYSABEND/SYSMDUMP/SYSUDUMP) should have at least one TCB with an RTM2WA address at TCB+X'E0'. The error indicators contained in the RTM2WA are described in the *Debugging Handbook.*

If an ESTAE routine is in control when an error occurs, RTM builds an SDWA (described in the *Debugging Handbook*) and places its address at the RTM2WA+X'D4' (RTM2RTCA).

Additional information about the failure may be found in the LOGREC buffer. The CVT+X'16C' points to the RBCB; RBCB+X'10' points to the LOGREC buffer.

If recursion occurs during RTM processing, other RTM2WAs may exist. If other work areas were obtained, the last one is pointed to by the TCB+X'E0'. The last RTM2WA points to the previous work area (RTM2WA+X'16C'(RTM2PRWA),'170'(RTM2PREV)).

RTM2WA is obtained from LSQA. It is therefore unique to each address space. If you are looking at a stand-alone dump, be sure that the area you are looking at belongs to the failing address space.

If the abending task is one of several abending tasks it is important to decide which task to look at first. There could be several failures or one failure causing all the others. Any failure in one of the system address spaces (such as JES2 or master scheduler) are important because they might have caused the user address spaces to terminate.

For the system to continue execution, the major components must be operational. If any of the critical system components (master scheduler, ASM, JES2, etc.) abend and fail to recover, the system cannot continue normal operation. Usually this can be determined from the records in LOGREC. However, check the TCB summary in the format section for completion codes.

The presence of a TCB completion code does not positively identify the associated task as being inoperational. It is possible that the completion code is residual and the task has recovered. The presence of a completion code makes the task suspect however, and should be investigated.

Simplify your choice of address spaces by using:

- SYS1.LOGREC external and internal entries
- Hardcopy log
- System trace table or GTF (check for SVC D or program check entries)

Once you have selected an address space and TCB, continue at Point A.

In addition to the RTM2WA, status indicators related to the problem can be found in:

- System trace table
- ESTAE control block (SCB)
- RB/SVRB
- TCB

Note 13 - Local Lock Only?

The current ASCB+X'80' contains the CPU ID. The current TCB+X'F0' also contains the CPU ID. The loop is within this task. Status is saved (if a STORE STATUS was done) in:

- PSA
- LCCA
- Current stack
- Local SDWA (ASXB+X'6C') - if the task abended while holding the lock
- System trace table
- In-storage LOGREC buffer

You can use the IPCS subcommand ANALYZE to do suspend lock analysis, and the IPCS subcommand STATUS CPU to determine which disabled locks are held and the active module on each CPU.

Is this task looping in the lock manager's code? Check the PSA+X'228' and LCCA+X'20C'. If the task is looping, another processor could be causing the loop by not freeing a spin lock that it is currently holding. The failure to free or obtain a lock can be caused by the lockword being overlaid.

If all processors are looping in lock manager code, then the failure could be in that code. If only one processor is in lock manager code, then the failure is likely to be in the processor currently holding the lock.

Where is the task looping? Why doesn't it free the locks? Is RTM involved with this task? If it is, continue at Point A.

See the chapters on "Locking" and "Effects of Multi-Processing on Problem Analysis" in Section 2 of this manual.

Note 14 - Local Lock Plus Another Lock.

The current ASCB+X'80' contains the CPU ID. The current TCB+X'F0' contains the CPU ID. The loop could be a spin loop waiting for another processor to release a global lock. In this case, determine why the lock has not been released.

Status indicators can be found in the following areas (if a STORE STATUS was done):

- PSA
- LCCA
- Current stack
- Local SDWA (ASXB+X'6C') - if the task abended while holding the local lock and at least one cross memory services lock
- System trace table
- In-storage LOGREC buffer

See Note 13 for additional information. Also see the chapters "Locking" and "Effects of Multiprocessing on Problem Analysis" in Section 2 of this manual.

Note 15 - Global Lock Held.

A global lock loop in an MP system could be normal. The spin loop continues until the global lock is released by the other processor. Determine why the other processor has not released the lock.

Error status indicators can be found in the following areas if a STORE STATUS was done:

- PSA (current PSW)
- LCCA
- Current stack
- Global SDWA (if there was an abended failure while the global lock was held)

The global SDWA for the super stacks is located at the respective super stack + X'458'. For the normal stack, the global SDWA immediately follows the RESTART super stack SDWA at + X'488'.

Now continue at Point A in the procedure. See Note 13 for additional information. Also see the chapters "Locking" and "Effects of Multiprocessing on Problem Analysis" in Section 2 of this manual.

Note 16 - IOS Not Okay.

Check the requests sent to IOS from auxiliary storage manager (ASM). Control blocks containing information are:

1. PART (paging activity reference table) - One entry per page data set. Each PART entry contains a pointer to an IORB (I/O request block) at X'1C' and a pointer to a UCB at X'2C'.

2. IORB contains the following I/O related data:

   IORB+X'1' = number of IORBs for this page data set
   IORB+X'3' = indicates whether IORB is in use
   IORB+X'4' = pointer to next IORB for this page data set
   IORB+X'8' = pointer to the first PCCW
   IORB+X'C' = pointer to the IOSB
   IORB+X'2C' = pointer to the last CCW in the channel program.

Note 17 - Suspended SRB or TCB With Lock Held.

An SRB can be suspended because of a page fault or segment fault, a synchronous page fix, or a request for a cross memory services lock when it is being held by another address space. The save area for the suspended SRB is the SSRB. If interrupted by a page fault, the SSRB is pointed to by the corresponding PCB+X'20'.

For a general cross memory services lock request, the SSRB is on the requested CMS lock suspend queue, which can be located in the system lock area. (See the topic on locking to locate the system lock area.)

For an ENQ/DEQ cross memory services lock request, the SSRB is on the CMSEQDQ lock suspend queue (CMSEDSQH).

A locked TCB can be suspended for the same reasons as an SRB. The save area is the IHSA of the locally locked address space (described in the *Debugging Handbook*). The IHSA is valid for a cross memory services (CMS) lock suspension if the ASCB is on the CMS lock suspend queue (SQH) for that cross memory service lock requested (either the general CMS, the CMSEQDQ, or the CMSSMF).

Note 18 - Not RTM2.

The presence of a TCB completion code does not positively identify the associated task as being inoperational. It is possible that the completion code is residual and the task has recovered. The presence of a completion code makes the task suspect however, and it should be investigated.

The save areas have been released. The status of the error has been written to SYS1.LOGREC. Continue at Point A with other TCBs in the dump. Another abending task is likely. If this is a stand-alone dump, it very likely has the needed LOGREC entry in the in-storage buffer. The CVT+X'16C' points to the RBCB; RBCB+X'10' points to the LOGREC buffer.

Note 19 - Real Storage Not Okay.

If page waits seem to be caused by the lack of real frames, check their usage. The PFT contains information about each frame currently being used. Important items to check are:

Which ASID holds the most real storage?
What are the frames being used for?
Is it valid that they be held or is there a problem with the freeing of the frames?

Status information might be found in the RCE, PFT, and RIT and ASCB (X'98') for the ASID that is holding all the frames.

Note 20 - IOS Okay.

Either something was missed along the way or the failure might be in one of the following areas:

- The IOS interrupt handler has failed to schedule the SRB/IOSB to the address space.
- The dispatcher has not handled the SRB correctly.
- RESET has not functioned properly.

Information on these errors might be found in the system trace table or the in-storage LOGREC buffers.

Note 21 - RTM1 Involved.

If there is an address at TCB+X'104' there might be two problems to resolve:

- The failure that caused the system to enter RTM initially.
- A loop between RTM1 and RTM2 since the pointer at TCB+X'104' normally lasts for only a short time.

  The pointer at TCB+X'104' is the EED (described in the *Debugging Handbook*). This data area is used to pass information from RTM1 to RTM2. Once RTM2 receives control the information is moved to the RTM2 work area and the EED is deleted. Therefore, because of its short duration the presence of an EED is unusual.

A SLIP trap may be required to solve the RTM loop. This loop is of course the most important problem.

If the loop is in the current task, check these status indicators:

- LCCA
- PSA
- Current stack
- RTM1WA
- RTM2WA
- SDWA pointed to by RTM1WA
- EEDs
- LOGREC buffer
- System trace table

If the loop is not in the current task, all the indicators above except the LCCA, PSA, and current stack are valid. The current FRR stack is also a valid status indicator. Remember that all disabled or locally locked code runs under the protection of an FRR routine.

Check the current stack pointer at PSA+X'380'. If the current stack pointer points to a super FRR it is almost certain that system damage has occurred.

The normal stack at X'C00' contains a record of FRR activity for the current address space. Location X'C0C' is the pointer to the current entry on the normal FRR stack. An address at X'C0C' equal to the address at X'C00' indicates an empty stack. Any address at X'C0C' greater than the address at X'C00' indicates that the system is currently under FRR protection and the first word in each FRR entry is a pointer to the FRR routine. Because the FRR routine is usually embedded within the routine it protects, identifying the FRR routine identifies the "looper."

If the byte at X'C28' is not 0 and there is an address in the RT1WRTCA field of the RTM1WA, there has been an entry into RTM1 and an SDWA has been obtained. The loop could be occurring in the FRR routine itself. The first word in the FRR stack entry points to the FRR routine. The SDWA (pointed to by RT1WRTCA) is the input passed to the FRR. Examine the code for the FRR and the module and consider the input passed to it in the SDWA to gain some insight into the cause of the loop.

Note 22 - Auxiliary Storage Not Okay.

If the count of I/O requests received (ASMVT+X'28' for paging requests, and ASMVT+X'30' for swapping requests) differs from the count of I/O requests completed (ASMVT+X'2C' for paging requests, and ASMVT+X'34' for swapping requests), and the IORBs are all idle (IORB+X'03', bit 0 is zero), locate those paging I/O requests (represented by an AIA) that ASM has received but not completed. Control blocks containing information are:

1. PARTE + X'3E' = count of outstanding I/O requests on a page data set

2. AIA-X'40' = part of PCB (PCBAIA) which contains RSM-related data

3. PART+X'24-28' = queue of AIAs waiting for PCCWs

4. ASMHD+X'0C' = swapout AIAs waiting to be processed

5. ASMHD+X'10' = queue of completed swap requests waiting to be returned to RSM

6. The following fields in the ASMVT point to SRBs to be processed:

   ASMVT+X'58' = the SRB whose parameter field (SRB+X'1C') points to a parameter list that points to I/O requests to be redriven through ASM

   ASMVT+X'5C' = the SRB whose parameter field (SRB+X'1C') has swap requests to be started

   ASMVT+X'60' = the SRB whose parameter field (SRB+X'1C') has error requests to be handled.

7. Each active IORB (PART entry+X'1C') contains a chain of PCCWs (IORB+X'8'). Each of these PCCWs that is active points to an AIA (PCCW+X'10').

8. If the AIA cannot be found by the above means (that is, it was lost by ASM), PCB/AIA is pointed to by the PFTE for the frame that is backing the page (PFTEPCB at PFT+X'18').

Note 23 - Local SRB Mode.

This indicates a loop (or enabled wait) within a single address space.

The SRB code cannot be pre-empted. If a loop occurs in the SRB routine, no higher priority task can be dispatched.

For an MP system there is a second possibility. Determine if the loop is in the lock manager code. If so, see notes 13, 14, and 15 for additional information. Continue at Point A.

Status Indicators

- System trace table.
- PSA (current PSW).
- LCCA.
- Current stack.
- RTM1WA (SDWA) - if abend occurred during SRB processing.
- ASCB.

Note: If the system is an MP and the loop is in the lock manager code, then another processor might be at fault. See notes 13, 14, and 15 for additional information. Continue at Point A.

Status Indicators

- PSA (current PSW).
- LCCA.
- Current stack.
- RTM1WA (SDWA) - if failure occurred during SRB processing.
- System trace table.

See the chapters "Locking," "System Execution Modes and Status Saving," and "Effects of MP on Problem Analysis" in Section 2 of this manual.

Note 24 - Global SRB Mode.

This indicates an enabled loop (or enabled wait) within a single address space.

The SRB code cannot be pre-empted. If a loop occurs in the SRB routine, no higher priority task can be dispatched.

For an MP system there is a second possibility. Determine if the loop is in the lock manager code. If so, see notes 13, 14, and 15 for additional information. Continue at Point A.

Status Indicators

- System trace table.
- PSA (current PSW).
- LCCA.
- Current stack.
- RTM1WA (SDWA) - if ABEND occurred during SRB processing.
- ASCB.

Note: If the loop is in the lock manager code, then another processor might be at fault. See notes 13, 14, and 15 for additional information. Continue at Point A.

Status Indicators

- PSA (current PSW).
- LCCA.
- Current stack.
- RTM1WA (SDWA) - if failure occurred during SRB processing.
- Trace table.

See the chapters "Locking," "System Execution Modes and Status Saving," and "Effect of MP on Problem Analysis" in Section 2 of this manual.

Note 25 - Wait in User Code.

This could be normal operation for an explicit wait (SVC 1) issued by a user routine. Determine if the event waited upon has completed. Check the TCB non-dispatchability flags to determine the reason. The flags normally indicate the area of the problem. For example, if TCBFLGS4 = X'04', this indicates an explicit wait was issued; TCBFLGS5 = X'80' means the task was terminated.

Note 26 - Non-enabled System.

A disabled wait normally has a wait state code associated with it. If so, the messages and codes should contain a problem description.

If there is no wait state code, the trace table should indicate the last sequence of events leading to the wait state condition. Probably a bad PSW (wait bit on) has been loaded.

Status Indicators

- LCCA
- PSA
- Current stack
- System trace table
- In-storage LOGREC buffer

If no valid WSC exists, if the PSW reflects the wait bit and is disabled, and if the STORE STATUS registers are not equal to zero, suspect a user/FE trap, bad branch, or system damage. Examine the trace table and attempt to define events that lead up to the wait condition. Was the last entry an SRB dispatch or an SVC or I/O interrupt? Using the PSW address, determine the entry point of the routine if possible and go to the chapter "MVS Trace Analysis" in Section 2 of this manual.

If the wait state occurs during system initialization, see the NIP vector table for error information. If the system is in a disable loop, determine what code is in control and why it is not returning to the enabled state.

A disabled loop in the lock manager on an MP system could be okay. Read notes 13, 14, and 15. A disabled loop in the SIGP processor on an MP system could be okay. (Another processor should turn off its PCCA's parallel/serial bit.)

If the system is looping (no wait bit), follow the SRB mode path. Check if RTM is involved and if it is, go to Point A.

*Note 27 - **Dispatchable Work Available.***

If the system is dispatchable and an address space has dispatchable work, the following are possible causes:

- The dispatcher is not functioning.
- CPU affinity may have been requested.
- JES2 might not be sending work to the initiators. In this case take a closer look at JES2.

*Note 28 - **Enqueue Lockout.***

Determine why the top task of a series of exclusive enqueues is not running or has not dequeued from the resource.

Note: It is valid for the top task to be swapped out. If it does not get swapped back in, then the failure might be in the system resource manager (SRM).

*Note 29 - **Incomplete I/O.***

This is a probable hardware error.

*Note 30 - **Explicit Wait in System Code.***

Check in the program listings (on microfiche) for the reason of the wait. Then determine which resource is being waited upon.

Once the resource is identified, determine if the wait should have been satisfied. If the wait appears to be a normal operation, continue at Point A for this TCB.

If the last thing done before the wait was an SVC 23 (WTO), related information can be found in the UCM base, prefix UCM, UCM extension and the chain of used WQEs.

*Note 31 - **System Analysis.***

If the failing task or component is not known, continue on the "yes" path of the flowchart.

To determine status about a TCB without doing a total system analysis, continue on the "no" path of the flowchart.

For a complete system analysis, start with low storage. Check the PSA for a low storage overlay. Critical fields are the CVT pointer at X'10', and the PSW new locations at location X'58-78' and at location X'00'. Be especially critical of the interrupt handler new PSWs. Any change to any new PSW will cause the next interrupt handler for that event to be dispatched in the wrong mode or key or to the wrong address. Subsequent results can be very unpredictable.

Keep in mind that the old PSWs are constantly updated by the hardware. They could have been overlaid at one time and still look okay in the dump from an MP system.

Other important fields in the PSA are as follows.

The interrupt code for the various classes of interrupts are located at:

- X'84' external interrupt
- X'88' SVC interrupt
- X'8C' program interrupt

These fields indicate the last type of interrupt associated with each interrupt class for each processor.

PSA+X'210' - address of the LCCA (1 per processor). The LCCA contains many of the status-saving areas that were located in low storage in previous systems. It is used for software environment saving and indicators. The registers associated with each of the interrupts you have discovered in the PSA are saved in this area. In addition, the system mode indicators for each processor are maintained in the LCCA.

The ASCB and TCB NEW/OLD pointers in the PSA (locations X'218-227') indicate the currently dispatched task. *Note:* PSATOLD can equal zero if an SRB is dispatched.

PSA+X'228' - PSASUPER. This is a field of bits that represent various supervisory functions in the system. If a loop is suspected, check these fields to isolate the looping process. Note that the dispatcher's super bit (X'04') will be left on when the wait task is dispatched.

PSA+X'2F8' - PSACLHS. This field indicates the current locks held on each processor. Knowing which locks are held may help isolate the problem, especially in a loop situation. By determining the lock holders you can isolate the current process.

PSA+X'380' - PSACSTK. This is the address of the active recovery stack that contains the addresses of the recovery routines to which control will be routed in case of an error. If the address is other than X'C00' (normal stack), determining the type of stack (for example, program check FLIH, restart FLIH) should aid in debugging the loop situation.

Another thing to consider in systems analysis is the possibility of a storage overlay of some critical system code such as IOS or GETMAIN.

Because of the recovery aspects of MVS (percolation and retry), evidence of storage overlays often can be found in the LOGREC recording buffer.

To manually find the LOGREC recording buffer in a dump:

1. Use the CVT+X'16C' to locate the RBCB.
2. RBCB+X'10' contains the address of the LOGREC buffer.

Note that you can also use the following IPCS subcommands to locate the LOGREC buffer:

```
EQUATE RBCB CVT+16C%
EQUATE LRCB RBCB+10%
```


To format the in-storage LOGREC buffer, you can use the IPCS subcommand VERBEXIT LOGDATA. Or, you can specify the LOGDATA verb under AMDPRDMP, as documented in *Service Aids*.

Identify the last entry in the buffer. Then, work backwards through the buffer to see the events that lead up to the dump. It is a good idea to have the SYS1.LOGREC records for the time leading up to the dump. Scan the trace table for SVC 4C. This represents a call to the LOGREC recording task and identifies a record being written to SYS1.LOGREC. If SVC 4Cs appear in the trace, it is certain that there are SYS1.LOGREC records that may more closely define the problem. (See the discussion of LOGREC records in the chapter "Use of Recovery Work Areas for Problem Analysis" in Section 2 of this manual.)

As a general approach, follow the flow of FRR activity from the last entry backwards until a pattern is recognizable or the first entry is found.

If you can define a function that is consistently failing (IOS, a program check, etc.), examine the system trace table for evidence of successful completion of this function. If the function completed successfully, the search for the function that caused the overlay is narrowed to those functions appearing in the trace between the last successful completion and the first evidence of error. This should at least narrow the search to the address space and task level.

Analyze the contents of the overlaid storage. If it appears to contain registers, determine what data areas or modules the registers are pointing at. This helps to identify the failing code.

If there is no evidence of a storage overlay, return to your system analysis at the beginning of Note 31.

If a storage overlay exists, further examination of the reported problem is usually non-productive until the cause of the system damage is explained.

It might be necessary to build a trap to identify the cause of the overlay. The chapter "SLIP Command" in Section 2 of this manual helps in building such a trap.

Appendix B. SVC DUMP Title Directory

This directory lists the titles of SVC dumps produced by MVS components via the SDUMP macro instruction. It also provides a module name to dump title cross-reference.

The directory has the following topics:

- **System-Defined SVC Dump Titles** - lists, in alphameric order, the titles of SVC dumps and provides diagnostic information for the modules that initiate the SVC dump via the SDUMP macro.
- **Operator- and Caller-Defined SVC Dump Titles** - provides diagnostic information for the modules that initiate SVC dumps via the SDUMP macro but where the dump title is defined by the system operator or the caller of SVC dump.
- **SVC Dumps Without Titles** - provides diagnostic information for the modules that initiate SVC dumps but where no titles are supplied on the SDUMP macro.
- **Module to SVC Dump Title Cross-Reference** - lists, in alphameric order, the MVS modules that issue the SDUMP macro and provides the titles of the SVC dumps specified by the modules on the SDUMP macro.

System-Defined SVC Dump Titles

This topic lists, in alphameric order, the titles of SVC dumps and provides diagnostic information for the modules that initiate the SVC dump via the SDUMP macro.

ABDUMP ERROR,COMPON=ABDUMP,
COMPID=SC1CM,ISSUER=IEAVTABD

Component: RTM - ABDUMP (5752-SC1CM)

Issuing Module: IEAVTABD

PLM: *System Logic Library*

Explanation: An error has occurred during RTM processing of a SYSABEND, SYSMDUMP, or SYSUDUMP dump. The error occurred when (1) ABDUMP attempted to set up dump processing, or (2) SNAP or SDUMP processing encountered an error while taking the dump. The areas dumped are LSQA, TRT, LPA, GRSQ, and subpools 230 and 250.

Problem Determination: A software record is written to SYS1.LOGREC. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA (in the LOGREC record).

ABEND IN IEAVTGLB

Component: RTM - PER Activation/Deactivation (5752-SC1CM)

Issuing Module: IEAVTGLB

PLM: *System Logic Library*

Explanation: An error has occurred then the SLIP processor attempted to activate or deactivate PER in the system. Message IEA415I is also issued. The areas dumped are PSA and SQA. The summary part of the dump requested by IEAVTGLB contains information relevant to the error.

Problem Determination: A software record is written to SYS1.LOGREC. Also refer to message IEA415I in *System Messages.*

ABEND IN IEAVTJBN

Component: RTM - PER Activation/Deactivation (5752-SC1CM)

Issuing Module: IEAVTJBN

PLM: *System Logic Library*

Explanation: An error has occurred when the SLIP processor attempted to determine if PER should be active for a new address space, started task, logon, mount, or job. Message IEA422I is also issued. The areas dumped are PSA and SQA. The summary part of the dump requested by IEAVTJBN contains information relevant to the error.

Problem Determination: A software record is written to SYS1.LOGREC. Also refer to message IEA422I in *System Messages.*

ABEND IN IEAVTLCL

Component: RTM - PER Activation/Deactivation (5752-SC1CM)

Issuing Module: IEAVTLCL

PLM: *System Logic Library*

Explanation: An error has occurred when the SLIP processor was attempting to activate or deactivate PER in an address space. Message IEA415I is also issued. The areas dumped are PSA, SQA, and LSQA. The summary part of the dump requested by IEAVTLCL contains information relevant to the error.

Problem Determination: A software record is written to SYS1.LOGREC. Also refer to message IEA415I in *System Messages.*

ABEND IN SMF INTERVAL PROCESSING - ROUTINE IEEMB836
JOBNAME = xxxxxxxx

Component: SMF (5752-SC100)

Issuing Module: IEEMB836 - FRR

PLM: *System Logic Library*

Explanation: An abend has occurred during SMF interval processing. xxxxxxxx indicates the name of the affected job. The areas dumped are SQA, ALLPSA, NUC, LSQA, RGN, LPA, TRT, SWA, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC. The SDWACSCT field in the SDWA contains the name of the module in control at the time of the error.

ABEND IN SMF INTERVAL PROCESSING - ROUTINE IFAEASI
JOBNAME = xxxxxxxx

Component: SMF (5752-SC100)

Issuing Module: IFAEASI - FRR

PLM: *System Logic Library*

Explanation: An abend has occurred during SMF interval processing for the early address spaces that do not go through full function start. xxxxxxxx indicates the name of the affected job. The areas dumped are SQA, ALLPSA, NUC, LSQA, RGN, LPA, TRT, SWA, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWACSCT - CSECT in error
SDWAMODN - Load module in error
SDWAREXN - ESTAE routine name

ABEND code AT hhhhhhhh (nnnnnn) + X'nnnn' cc- - -cc

Component: JES2 (5752-SC1BH)

Issuing Module: HASPTERM or HASPRAS

PLM: *JES2 Logic*

Explanation: An abend has occurred during JES2 processing. Fields in the dump title are:

code	abend code
hhhhhhhh	failing module name
nnnnnn	entry point address
X'nnnn'	offset to the failing instruction
cc- - -cc	brief description of the abend code and the JES2 release level

Abend codes that start with S are system codes, and those that start with $ are JES2 codes. The areas dumped are PSA, NUC, RGN, TRT, SQA, CSA, LPA, and SWA.

Problem Determination: For information on system abend codes, refer to *System Codes*; for JES2 abend codes, refer to message $HASP095 in *JES2 Messages*.

ABEND=aaa,COMPON=CONVERTER,COMPID=SC1B9,
ISSUER=IEFNB9CR

Component: Converter (5752-SC1B9)

Issuing Module: IEFNB9CR - Converter Recovery Routine

PLM: None (refer to microfiche)

Explanation: IEFNB9CR was entered due to an expected error (0B0 abend or program check) during converter processing. The areas dumped are LSQA, RGN, LPA, and SWA.

Problem Determination: A software record is written to SYS1.LOGREC.

ABEND=aaa,COMPON=INTERPRETER,COMPID=SC1B9,
ISSUER=IEFNB9IR

Component: Interpreter (5752-SC1B9)

Issuing Module: IEFNB9IR - Interpreter Recovery Routine

PLM: None (refer to microfiche)

Explanation: IEFNB9IR was entered due to an expected error (0B0 abend or program check) during interpreter processing. The areas dumped are LSQA, RGN, LPA, and SWA.

Problem Determination: A software record is written to SYS1.LOGREC.

ABEND=aaa,COMPON=PC/AUTH-PCLINK
UNSTACK,COMPID=SCXMS,
ISSUER=IEAVXSTK

Component: PC/AUTH Services (5752-SCXMS)

Issuing Module: IEAVXSTK

PLM: *System Logic Library*

Explanation: An error has occurred while the PCLINK service routine (IEAVXSTK) was processing an UNSTACK request. The areas dumped are NUC, LSQA, SQA, and SUM.

Problem Determination: A software record is written to SYS1.LOGREC. The SDWAVRA includes the following key-length data formatted entries:

- The 24-byte FRR parameter area
- The caller's return address
- The current TCB address (if not task mode, 0)
- The PCLINK work areas in the LCCA (56 bytes beginning at LCCASREG)
- The PSASTKE and PSAKPSW fields

ABEND=aaa,REASON=xxyy,GPRrr=zzzzzzzz,COMPON=PC/AUTH-macro,
COMPID=SCXMS,ISSUER=IEAVXPCR

Component Program Call/Authorization (PC/AUTH) Services

Issuing Module IEAVXPCR - PC/AUTH Services FRR

PLM: *System Logic Library*

Explanation: An error has occurred while a PC/AUTH service routine was processing. IEAVXPCR issues the SDUMP macro. The areas dumped are ALLPSA, NUC, SQA, SUMDUMP, RGN, and LSQA. Also, subpools 255 (LSQA) and 229 (pageable private area) of the PC/AUTH address space are synchronously dumped via SUMLSTA. Fields in the title are:

aaa	system completion code
xxyy	low-order two bytes of register 15 at the time of the error
rr	general purpose register
zzzzzzzz	contents of general purpose register rr
macro	name of the macro that invoked the PC/AUTH service

Notes:

1. *If the system completion code is 053, refer to System Codes for a description of the reason code xxyy and the diagnostic register value. For other system completion codes, the contents of register 14 at the time of the error are shown as GPR14=zzzzzzzz.*

2. *If register contents are not available at the time of error, then the title text contains N/A or UNAVAIL.*

3. *If the FRR is entered with an invalid service-in-control code value in PCRASERV, the title will contain 'PCRAEERC=nnnn' where nnnn is the PCRAEERC field at entry to the FRR.*

Problem Determination: A software record is written to SYS1.LOGREC and includes key-length-data formatted entries in SDWAVRA containing:

- The program call recovery area (PCRA) at entry to the FRR.
- Diagnostic information, if any, from the control block validation and/or queue verification routines.
- The service routine recovery area (SRRA). (As much as possible is included in the SDWAVRA.)

It is suggested that you initially print the dump using the PRDMP options LOGDATA, SUMMARY, SUMDUMP, and PRINT CURRENT to obtain sufficient diagnostic data.

ABEND=xxx,REASON=nnnn,MODULE=IEAVSPDM,
COMPON=RECONFIGURATION - SPDM,
COMPID=SC1CZ,ISSUER=IEAVSPDM

Component: Reconfiguration (5752-SC1CZ)

Issuing Module: IEAVSPDM

PLM: *System Logic Library*

Explanation: An abend has occurred in module IEAVSPDM during either system initialization processing (at IPL time) or during IEAVSPDM's attempt to process the machine check interruption handler's post of the service processor damage ECB. In the latter case, an MSSF machine check interruption has occurred and the MSSF (or processor controller) is no longer functioning.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IEAVSPDM (load module in error)
SDWACSCT - IEAVSPDM (CSECT in error)
SDWAREXN - IEAVSPDM (CSECT containing recovery routine)
SDWARRL - SPDMRECV (recovery routine)

ABP:IDA121A2 - ABEND FROM ABP FRR

Component: Block Processor (5665-28419)

Issuing Module: IDA121A2 - FRR

PLM: *VSAM Logic*

Explanation: An abnormal termination has occurred during VSAM block processing. The FRR routine in IDA121A2 issues the SDUMP macro. The areas dumped are PSA, NUC, RGN, TRT, CSA, and SQA.

Problem Determination: A VSAM request was being processed in the actual block processor (ABP), initiating I/O, when the error occurred. Register 3 points to the IOMB for the request. For information on how the IOMB relates to the VSAM control block structure, refer to *VSAM Logic*.

ABP:IDA121A3 - ABEND FROM NORMAL END FRR

Component: Block Processor (5665-28419)

Issuing Module: IDA121A3 - FRR

PLM: *VSAM Logic*

Explanation: An abnormal termination has occurred while IDA121A3 was processing a VSAM request. RTM passes control to the FRR in IDA121A3 (at entry point IDA121F3), which issues the SDUMP macro. The areas dumped are PSA, NUC, RGN, TRT, CSA, and SQA.

Problem Determination: I/O for a VSAM request had completed normally when the error occurred. Register 3 points to the IOMB for the request. For information on how the IOMB relates to the VSAM control block structure, refer to *VSAM Logic*.

ABP:IDA121A4 - ABEND FROM ABNORMAL END FRR

Component: Block Processor (5665-28419)

Issuing Module: IDA121A4 - FRR

PLM: *VSAM Logic*

Explanation: An abnormal termination has occurred while IDA121A4 was processing a VSAM request. RTM passes control to the FRR in IDA121A4 (at entry point IDA121F4), which issues the SDUMP macro. The areas dumped are PSA, NUC, RGN, TRT, CSA, and SQA.

Problem Determination: I/O for a VSAM request had completed abnormally when the error occurred. Register 3 points to the IOMB for the request. For information on how the IOMB relates to the VSAM control block structure, refer to *VSAM Logic*.

ABP:IGC121 - ABEND FROM SIOD FRR

Component: Block Processor (5665-28419)

Issuing Module: IGC121 - FRR

PLM: *VSAM Logic*

Explanation: An abnormal termination has occurred while IGC121 was processing a VSAM request. RTM passes control to the FRR in IDA121 (at entry point IDA121F1), which issues the SDUMP macro. The areas dumped are PSA, NUC, RGN, TRT, CSA, and SQA.

Problem Determination: The I/O manager was processing a VSAM request when the error occurred. Register 3 points to the IOMB for the request. For information on how the IOMB relates to the VSAM control block structure, refer to *VSAM Logic.*

AHL007I GTF TERMINATING ON ERROR CONDITION

Component: GTF (5752-SC111)

Issuing Module: AHLGTFI

PLM: *Service Aids Logic*

Explanation: An error has occurred during GTF initialization. The ESTAE routine AHLIESTA in AHLGTFI requests a retry action which issues the SDUMP macro, writes message AHL016I, and frees storage and other resources that were allocated to GTF. GTF terminates itself. The areas dumped are RGN, SQA, and MCHEAD control block.

Problem Determination: A software record is written to SYS1.LOGREC. All control blocks allocated to GTF are dumped.

COMMON AUTHORIZATION CHECK ROUTINE ERROR,
ABEND=xxx,COMPON=SCHR-CMF,COMPID=BB131,
ISSUER=IEFCMAUT

Component: Scheduler (5752-SC1B6)

Issuing Module: IEFCMAUT

PLM: *System Logic Library*

Explanation: An abend has occurred during authorization checking. ESTAE routine SETESTAE in IEFCMAUT sets up the recovery environment. If no previous abend has occurred, recovery routine RECOVERY in IEFCMAUT requests a retry. If there was a previous abend, the recovery routine issues a SETRP to indicate that RTM should percolate the error to the next level of recovery.

Problem Determination: If an SDWA was obtained, a software record is written to SYS1.LOGREC which includes:

SDWAMODN - IEFCMAUT (load module in error)
SDWACSCT - IEFCMAUT (CSECT in error)
SDWAREXN - IEFCMAUT (CSECT containing recovery routine)
SDWARRL - RECOVERY (recovery routine)
SDWACID - SC1B6 (component identifier)
SDWARCDE - return code
SDWAMLVL - product level

COMP=DATA IN VIRTUAL, COMPID=SCDIV, ISSUER=ITVRD, DATA IN VIRTUAL GENERAL ESTAE RECOVERY FAILURE

Component: Data-in-Virtual (5752-SCDIV)

Issuing Module: ITVDEST - ESTAE

PLM: *System Logic Library*

Explanation: An error has occurred during data-in-virtual general ESTAE recovery processing. The areas dumped are SUMDUMP, LSQA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC (if SDWA is available) and includes:

SDWAMODN - ITVDIV
SDWACSCT - CSECT in error
SDWAREXN - ITVRD (recovery routine CSECT)
SDWACID - SCDIV
SDWARRL - ITVRDEST (recovery routine label)
SDWASC - DIV ESTAE RECOVERY
SDWAVRA - Variable data in key-length-data format; includes the DRA.

Note: IPCS subcommands, DIVDATA and VERBEXIT (with the DAEDATA verb name), can be used to format information related to data-in-virtual.

COMP=DATA IN VIRTUAL, COMPID=SCDIV, ISSUER=ITVRG, DATA IN VIRTUAL GENERAL FRR RECOVERY FAILURE

Component: Data-in-Virtual (5752-SCDIV)

Issuing Module: ITVRGFRR - FRR

PLM: *System Logic Library*

Explanation: An error has occurred during data-in-virtual general FRR recovery processing. The areas dumped are SUMDUMP, LSQA, SQA, and NUC.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - CSECT in error
SDWAREXN - ITVRG (recovery routine CSECT)
SDWACID - SCDIV
SDWARRL - ITVRGFRR (recovery routine label)
SDWASC - DIV FRR RECOVERY
SDWAVRA - Variable data in key-length-data format; includes the DRA.

Note: IPCS subcommands, DIVDATA and VERBEXIT (with the DAEDATA verb name), can be used to format information related to data-in-virtual.

COMP=DATA IN VIRTUAL, COMPID=SCDIV, ISSUER=ITVRK, TRACE TABLE, SEQUENCE NUMBER = xxxxxxxxxx

Component: Data-in-Virtual (5752-SCDIV)

Issuing Module: ITVRKTR - Trace

PLM: *System Logic Library*

Explanation: The data-in-virtual trace table was filled during data-in-virtual processing. Sequence number xxxxxxxxxx in the dump title indicates the number of times that the first entry in the trace table was used. The sequence number starts at zero and is increased by one each time the trace table fills and wraps around. When a new table replaces the trace table, the sequence number starts again at zero. The dumped area is SUMDUMP.

Problem Determination: The dump includes the following information in the summary dump:

- DIB, DIBX
- Data-in-virtual component trace table control area (CTC)
- Data-in-virtual trace table

Note: IPCS subcommands, DIVDATA and VERBEXIT (with the DAEDATA verb name), can be used to format information related to data-in-virtual.

COMP=DATA IN VIRTUAL, COMPID=SCDIV, ISSUER=ITVRM, WITH INVALID DRA

Component: Data-in-Virtual (5752-SCDIV)

Issuing Module: ITVRMDMP - FRR

PLM: *System Logic Library*

Explanation: An error has occurred during data-in-virtual disabled processing and the DRA is damaged. The areas dumped are SUMDUMP, LSQA, SQA, and NUC.

Problem Determination: The dump includes the following information in the summary dump:

- DIB, DIBX at the time of the error.
- Data-in-virtual component trace table control area (CTC), if applicable.
- Data-in-virtual trace table, if applicable.
- Data-in-virtual CPU-related work/save area.
- LSQA used by data-in-virtual, if applicable.

Also, the dump includes the 4K SQA buffer in description-length-data format, if applicable. (The CVTSDBF field in the CVT contains the address of the buffer.)

A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - CSECT in error
SDWAREXN - ITVRF (recovery routine CSECT)
SDWACID - SCDIV
SDWARRL - ITVRFRR (recovery routine label)
SDWASC - DATA IN VIRTUAL DISABLE
SDWAVRA - Variable data in key-length-data format; includes the time-of-error information.

Notes:

1. *For an explanation of data-in-virtual's VRA keys and error reason codes, see System Logic Library.*

2. *IPCS subcommands, DIVDATA and VERBEXIT (with the DAEDATA verb name), can be used to format information related to data-in-virtual.*

COMP=DATA IN VIRTUAL, COMPID=SCDIV, ISSUER=ITVRM, WITH VALID DRA

Component: Data-in-Virtual (5752-SCDIV)

Issuing Module: ITVRMDMP - FRR

PLM: *System Logic Library*

Explanation: An error has occurred during data-in-virtual disabled processing. The areas that are dumped are SUMDUMP, LSQA, SQA, and NUC.

Problem Determination: The dump includes the following information in the summary dump:

- DIB, refreshed DIBX.
- Data-in-virtual component trace table control area (CTC), if applicable.
- Data-in-virtual trace table, if applicable.
- Data-in-virtual CPU-related work/save area.
- LSQA used by data-in-virtual, if applicable.

Also, the dump includes the 4K SQA buffer in description-length-data format. If applicable, the buffer will contain the DIBX at the time of the error and any queue error information. (The CVTSDBF field in the CVT contains the address of the buffer.)

A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - CSECT in error
SDWAREXN - ITVRF (recovery routine CSECT)
SDWACID - SCDIV
SDWARRL - ITVRFRR (recovery routine label)
SDWASC - DATA IN VIRTUAL DISABLE
SDWAVRA - Variable data in key-length-data format; includes the symptom strings and time-of-error information.

Notes:

1. *For an explanation of data-in-virtual's VRA keys and error reason codes, see System Logic Library.*

2. *IPCS subcommands, DIVDATA and VERBEXIT (with the DAEDATA verb name), can be used to format information related to data-in-virtual.*

COMP=DATA IN VIRTUAL, COMPID=SCDIV, ISSUER=ITVRR, WITH INVALID DRA

Component: Data-in-Virtual (5752-SCDIV)

Issuing Module: ITVRRDMP - ESTAE

PLM: *System Logic Library*

Explanation: An error has occurred during data-in-virtual enabled processing and the DRA is damaged. The areas that are dumped are SUMDUMP, LSQA, and SQA.

Problem Determination: The dump includes the following information:

- DIB, DIBX at the time of the error, DRA (which are in the summary dump).

- 4K SQA buffer in description-length-data format, if applicable. (The CVTSDBF field in the CVT contains the address of the buffer.)

A software record is written to SYS1.LOGREC (if the SDWA is available and this is an unexpected error) and includes:

SDWAMODN - ITVDIV
SDWACSCT - CSECT in error
SDWAREXN - ITVRE (recovery routine CSECT)
SDWACID - SCDIV
SDWARRL - ITVRESTA (recovery routine label)
SDWASC - DATA IN VIRTUAL ENABLED
SDWAVRA - Variable data in key-length-data format; includes the time-of-error information.

Notes:

1. *For an explanation of data-in-virtual's VRA keys and error reason codes, see System Logic Library.*

2. *IPCS subcommands, DIVDATA and VERBEXIT (with the DAEDATA verb name), can be used to format information related to data-in-virtual.*

COMP=DATA IN VIRTUAL, COMPID=SCDIV, ISSUER=ITVRR, WITH VALID DRA

Component: Data-in-Virtual (5752-SCDIV)

Issuing Module: ITVRRDMP - ESTAE

PLM: *System Logic Library*

Explanation: An error has occurred during data-in-virtual enabled processing. The areas that are dumped are SUMDUMP, LSQA, and SQA.

Problem Determination: The dump includes the following information:

- DIB, refreshed DIBX, DRA (which are in the summary dump).
- 4K SQA buffer in description-length-data format. If applicable, the buffer will contain the DIBX at the time of the error and any queue error information. (The CVTSDBF field in the CVT contains the address of the buffer.)

A software record is written to SYS1.LOGREC (if the SDWA is available and this is an unexpected error) and includes:

SDWAMODN	- ITVDIV
SDWACSCT	- CSECT in error
SDWAREXN	- ITVRE (recovery routine CSECT)
SDWACID	- SCDIV
SDWARRL	- ITVRESTA (recovery routine label)
SDWASC	- DATA IN VIRTUAL ENABLED
SDWAVRA	- Variable data in key-length-data format; includes the symptom strings and time-of-error information.

Notes:

1. *For an explanation of data-in-virtual's VRA keys and error reason codes, see System Logic Library.*

2. *IPCS subcommands, DIVDATA and VERBEXIT (with the DAEDATA verb name), can be used to format information related to data-in-virtual.*

COMPID=SC1B8, xxx ABEND IN MASTER TR modname

Component: Master Scheduler Commands (5752-SC1B8)

Issuing Module: IEEMB816

PLM: *System Logic Library*

Explanation: An abend has occurred during (1) processing of the TRACE command, or (2) execution of the IEETRACE macro, xxx indicates the abend code. Modname indicates the module in control at the time of the error and is one of the following:

IEEMB808 - the error occurred while adding an entry to the master trace function during system initialization or in response to a TRACE command.

IEEMB809 - the error occurred while activating or deactivating the master trace function during system initialization or in response to a TRACE command.

IEEMB816 - the abend occurred while processing some other error in the master trace facility.

UNKNOWN - the recovery routine could not determine the module that was in control at the time of the error.

The areas dumped are SUMDUMP, TRT, FRR work area, FRR parameter area, UCM extension, master trace caller's parameter list, and load module IEEMB808 with its dynamically acquired storage. Message IEE480I or IEE581I is also issued.

Problem Determination: A software record is written to SYS1.LOGREC. Refer to message IEE480I or IEE481I in *System Messages,* and to the appropriate code (indicated by xxx in the title) in *System Codes.*

COMPID=SC1CJ,COMPON=CONTENTS
SUPERVISOR,ISSUER=CSVFRR,
DUMP PRIOR TO QUEUE VERIFICATION

Component: Contents Supervisor (5752-SC1CJ)

Issuing Module: CSVFRR

PLM: *System Logic Library*

Explanation: An error has occurred during processing by the contents supervisor. The error is probably a system error because errors that occur during the validation of user-specified parameter lists result in abend codes 206. The FRR routine CSVFRR issues the SDUMP prior to performing queue validation for the load list (LLE queue) and the job pack area queue (CDE queue) for the failing task, both of which reside in the LSQA. The areas dumped are SUM, TRT, and LSQA.

Problem Determination: A software record is written to SYS1.LOGREC. The following fields in the SDWA are filled in: SDWAMODN, SDWACID, SDWASC, SDWARRL, SDWAREXN, SDWAMLVL, and SDWACSCT.

The queue verify routine, IEAVEQV0, records errors in the SDWAVRA that are detected in the LLE queue or the CDE queue. The error recording fields contain the EBCDIC labels "JPQ ERROR" and "LLS ERROR," either of which are followed by "NONE" if no errors were detected.

While the contents supervisor is active, register 5 points to the contents supervisor SVRB (except when the recovery module CSVFRR is in control, or when the contents supervisor calls other services). The extended save area RBEXSAVE in the SVRB contains data that is specific to the contents

supervisor and includes the name of the requested module, pointers to the CDEs and other resources, and various flags.

COMPID=SC1CJ,COMPON=CONTENTS
SUPERVISOR,ISSUER=CSVFRR2,
FAILURE DURING QUEUE VERIFICATION

Component: Contents Supervisor (5752-SC1CJ)

Issuing Module: CSVFRR (CSVFRR2 routine)

PLM: *System Logic Library*

Explanation: During recovery processing, an error has occurred while the contents supervisor was attempting to perform queue validation as a result of a previous error. This error caused the second level FRR, CSVFRR2, to gain control. The areas dumped are SUM, TRT, LSQA, SQA, and NUC.

Problem Determination: A software record is written to SYS1.LOGREC. The following fields in the SDWA are filled in: SDWAMODN, SDWACID, SDWASC, SDWARRL, SDWAREXN, SDWAMLVL, and SDWACSCT.

The SDWAVRA contains the FRR parameter list that was initialized by CSVFRR before the queue validation began. The parameter list is preceded by the EBCDIC header "CSVFRR ABEND, CSVFRR DATA IS: QVPL, SDWA, QVCSAREA, TCB, ASCB, NSI" and contains the following:

- Address of the queue verification parameter list (QVPL) that is used by the queue verify routine IEAVEQV0.
- Address of the SDWA.
- Address of the 200-byte FRR work area for CSVFRR.
- Address of the TCB (PSATOLD).
- Address of the ASCB (PSAAOLD).
- Return address for the FRR.

COMPID=SC1CZ,MODULE IEECB927 FAILED,ABEND(xxx)

Component: Reconfiguration (5752-SC1CZ)

Issuing Module: IEECB927

PLM: *System Logic Library*

Explanation: An abend has occurred in the command processor for a CONFIG (CF) operator's request. The command and the main parameter area (RDPMPARM) is put into the SDWA.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IEECB927
SDWACSCT - IEECB927
SDWAREXN - IEECB927
SDWARRL - ESTAERTN
SDWACID - SC1CZ
SDWACF - CF COMMAND PROCESSOR
SDWAMLVL - module level

COMPID=SC1CZ,MODULE IEEVCONF FAILED,ABEND(xxx)

Component: Reconfiguration (5752-SC1CZ)

Issuing Module: IEEVCONF

PLM: *System Logic Library*

Explanation: An abend has occurred during CONFIG (CF) command processing. The retry point index and main parameter area are put into the SDWA. A retry attempt is made to continue the next request. Processing for the current request is terminated.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IEEVCONF
SDWACSCT - IEEVCONF
SDWAREXN - IEEVCONF
SDWARRL - ESTARTN
SDWACID - SC1CZ
SDWACF - CONFIG COMMAND
SDWAMLVL - module level

COMPID=SC1CZ,MODULE IEEVRDPM FAILED,ABEND(xxx)

Component: Reconfiguration (5752-SC1CZ)

Issuing Module: IEEVRDPM

PLM: *System Logic Library*

Explanation: An abend has occurred while IEEVRDPM was trying to read a CONFIGxx member from SYS1.PARMLIB as a result of the DISPLAY M=CONFIG(xx) or CONFIG MEMBER(xx) operator request. The main parameter area (RDPMPARM) for the module is put into the SDWA.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IEEVRDPM
SDWACSCT - IEEVRDPM
SDWAREXN - IEEVRDPM
SDWARRL - RDPMSTA
SDWACID - SC1CZ
SDWACF - D M OR CF PARMLIB READ
SDWAMLVL - module level

COMPID=SC1CZ,MODULE IEEVRSCN FAILED,ABEND(xxx)

Component: Reconfiguration (5752-SC1CZ)

Issuing Module: IEEVRSCN

PLM: *System Logic Library*

Explanation: An abend has occurred while IEEVRSCN was trying to execute a configuration display during a CONFIG (CF) ON/OFF request. The command and the main parameter area (RDPMPARM) are put into the SDWA.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IEEVRSCN
SDWACSCT - IEEVRSCN
SDWAREXN - IEEVRSCN
SDWARRL - ESTAERTN
SDWACID - SC1CZ
SDWACF - CF CMND-DISPLAY MODULE
SDWAMLVL - module level

COMPID=5752-SC1CM,ISSUER=IEAVTDSV (IN LINKLIB), FAILURE IN DUMPSRV ADDRESS SPACE

Component: Dumping Services (5752-SC1CM)

Issuing Module: IEAVTDSV

PLM: *System Logic Library*

Explanation: An error has occurred during processing in the job step task of the DUMPSRV address space. The problem could occur during initialization of the DUMPSRV address space, or during post exit processing for a SVC dump or a SYSMDUMP. The areas dumped are SUM, TRT, LSQA, subpools 231 and 0, and the GRSQ data (if there was an enqueue error).

Problem Determination: Print the summary dump and PRINT CURRENT. Check the DSVCB to determine the state of the address space. The SDWAVRA contains:

- The ESTAE parameter area
- The DSVCB control block

COMPID=5752-SC1CM,ISSUER=IEECB910 - DISPLAY DUMP COMMAND PROCESSOR

Component: Dumping Services (5752-SC1CM)

Issuing Module: IEECB910

PLM: *System Logic Library*

Explanation: An error has occurred during DISPLAY DUMP command processing. The areas dumped are SUM, TRT, LSQA, subpools 245 and 0, and a storage list containing the command input buffer. Module IEECB910 allows duplicate dumps to be suppressed by DAE by specifying the VRADAE key.

Problem Determination Print the summary dump and PRINT CURRENT. Check the DISPLAY DUMP command to determine the type of processing requested. The SDWAVRA contains the following:

- The ESTAE parameter area
- The DISPLAY DUMP command from the CSCB

COMPID=5752-SC1CM,ISSUER=IEECB926 - DUMPDS PROCESSOR

Component: Dumping Services (5752-SC1CM)

Issuing Module: IEECB926

PLM: *System Logic Library*

Explanation: An error has occurred while processing the dump data sets for a DUMPDS command in the DUMPSRV address space. The error also could have occurred while initializing the dump data set queue (IHASDDSQ). The areas dumped are SUM, TRT, LSQA, subpools 245 and 15, and a storage list containing the DSVCB, the DSPA (DUMPDS parameter area), and the DSPAOUT area pointed to by the DSPA.

Problem Determination: Print the summary dump and PRINT CURRENT. Check the DSPA to determine which DUMPDS command was requested. Check the LOGREC entry for this dump. If the SDWARRL field contains ESTATASK, then the problem probably occurred during initialization of the DUMPSRV address space. If the field contains ESTADDS, then the error occurred during DUMPDS command processing. The SDWAVRA contains the following:

- The ESTAE parameter area
- The DSPA (IHADSPA)

COMPID=5752-SC1CM,ISSUER=IEECB923 - DUMPDS COMMAND FAILED

Component: Dumping Services (5752-SC1CM)

Issuing Module: IEECB923

PLM: *System Logic Library*

Explanation: An error has occurred while a DUMPDS command was processing. The areas dumped are SUM, TRT, LSQA, subpool 245, and a storage list containing the DSPA (DUMPDS parameter area). Module IEECB923 allows duplicate dumps to be suppressed by DAE by specifying the VRADAE key.

Problem Determination: Print the summary dump, PRINT CSA, and PRINT CURRENT. Check the DSPA to determine which DUMPDS command was issued. The SDWAVRA contains the following:

- The ESTAE parameter area
- The DSPA (IHADSPA)
- The command input buffer for the DUMPDS command.

COMPID=5752-SC143,ISSUER=ADYPSTD, FAILURE IN THE DUMP ANALYSIS AND ELIMINATION POST DUMP EXIT

Component: Dump Analysis and Elimination (5752-SC143)

Issuing Module: ADYPSTD

PLM: *System Logic Library*

Explanation: An abend has occurred during ADYPSTD processing. Diagnostic data is placed in the SDWA. A retry attempt is performed when possible. All resources are cleaned up if the ESTAE routine percolates the error.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - ADYPSTD
SDWACSCT - ADYPSTD
SDWAMLVL - Module level
SDWASC - "FAILURE IN (Failing subroutine name)"
SDWAREXN - ADYPSTD
SDWARRL - ADYPSTDR
SDWACID - SC143
SDWACIDB - 5752

The SDWAVRA contains the ESTAE parameter list, the SDUMP exit parameter list (SDEPL), and the DAE predump/postdump parameter list (DSPD).

COMPID=5752-SC143,ISSUER=ADYSETP, FAILURE IN DAE SET PROCESSING

Component: Dump Analysis and Elimination (5752-SC143)

Issuing Module: ADYSETP

PLM: *System Logic Library*

Explanation: An abend has occurred during ADYSETP, ADYPARS, or ADYMSG processing. Diagnostic data is placed in the SDWA. A retry attempt is performed when possible. The GETMAINed area for the temporary transaction queue is freed if the ESTAE routine percolates the error.

ADYSETP allows duplicate dumps to be suppressed by DAE by specifying the VRADAE key.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - ADYSETP, ADYPARS, or ADYMSG
SDWACSCT - ADYSETP, ADYPARS, or ADYMSG
SDWAMLVL - Module level
SDWASC - "FAILURE IN (Failing subroutine name)," or DAE PARSING ROUTINE
SDWAREXN - ADYSETP
SDWARRL - ADYSETPR
SDWACID - SC143
SDWACIDB - 5752

The SDWAVRA contains the ESTAE parameter list, the name of the SYS1.PARMLIB member at the time of the error, and the DAE key to specify dump suppression.

COMPID=5752-SC143,ISSUER=ADYTRNS, FAILURE IN THE TRANSACTION PROCESSOR FOR DAE

Component: Dump Analysis and Elimination (5752-SC143)

Issuing Module: ADYTRNS

PLM: *System Logic Library*

Explanation: An abend has occurred during ADYTRNS, ADYIO, or ADYMSG processing. Diagnostic data is placed in the SDWA. A retry attempt is performed when possible. All resources are cleaned up if the ESTAE routine percolates the error.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - ADYTRNS, ADYIO, or ADYMSG
SDWACSCT - ADYTRNS, ADYIO, or ADYMSG
SDWAMLVL - Module level
SDWASC - "FAILURE IN (Failing subroutine name)"
SDWAREXN - ADYTRNS
SDWARRL - ADYTRNSR
SDWACID - SC143
SDWACIDB - 5752

The SDWAVRA contains the ESTAE parameter list, and the first 200 bytes of the current DAE transaction.

COMPON=AVM,COMPID=SCAVM,ISSUER=modname(s),descriptive name

Component: Availability Manager (5752-SCAVM)

Issuing Module: AVFES, AVFEP, AVFFR, or AVFFS

PLM: *System Logic Library*

Explanation: Availability manager recovery routines have intercepted an abend in the availability manager. Retry may or may not be attempted. The areas dumped are all protect key 3 storage in CSA subpools 227, 231, and 241. If the private area of the failing routine's address space is

accessible, the dump will contain key 3 storage from private area subpools 230 and 251.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=CMND-ESTAE,COMPID=SC1B8,ISSUER=IEECB860
FAILURE IN COMMAND xxxx

Component: Master Scheduler Commands (5752-SC1B8)

Issuing Module: IEECB860

PLM: *System Logic Library*

Explanation: An error has occurred in the command processor while processing command xxxx. The areas dumped are PSA, ALLNUC, LSQA, RGN, LPA, TRT, CSA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=COMMTASK,COMPID=SC1CK,ISSUER=IEAVG600,FAILURE
IN COMMTASK ENF ROUTINE

Component: COMMUNICATIONS TASK (5752-SC1CK)

Issuing Module: IEAVG600

PLM: *System Logic Library*

Explanation: An error has occurred during event notification facility (ENF) signal processing. The areas dumped are SQA, TRT and SUMDUMP. The summary dump contains the UCM prefix.

Problem Determination: A software record is written to SYS1.LOGREC. The SDWAVRA, which is in key-length-data format, contains the following information:

- Code and data registers
- Save area register
- Event code
- Qualifier code

COMPON=COMMTASK,COMPID=SC1CK,ISSUER=IEAVMFRR=FRR
or ESTAE, COMMUNICATIONS TASK DUMP

Component: Communications Task (5752-SC1CK)

Issuing Module: IEAVMFRR

PLM: *System Logic Library*

Explanation: IEAVMFRR is entered when an error has occurred during processing of a communications task function. The areas dumped are LSQA, LPA, ALLPSA, ALLNUC, SUMDUMP, and TRT. The SUMLSTA contains the UCM, RDCMs, SACBs, and TDCMs.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=COMMTASK,COMPID=SC1CK,ISSUER=IEAVSTAA, FAILURE IN COMMUNICATIONS TASK

Component: Communication Task (5752-SC1CK)

Issuing Module: IEAVSTAA

PLM: *System Logic Library*

Explanation: IEAVSTAA is entered when both (1) an error has occurred during communications task processing, and (2) after unsuccessful recovery processing by ESTAE or FRR routines in the communications task. The areas dumped are ALLNUC, SUMDUMP, LSQA, RGN, LPA, SWA, ALLPSA, and TRT.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=COMMTASK,COMPID=SC1CK,ISSUER=IEAVN700,FAILURE IN COMM TASK ADDRESS SPACE CREATE ROUTINE

Component: Communications Task (5752-SC1CK)

Issuing Module: IEAVN700

PLM: *System Initialization Logic*

Explanation: An error has occurred while IEAVN700 was creating the communications task address space. The areas dumped are ALLPSA, RGN, LSQA, SQA, and SUMDUMP. SUMDUMP contains the trace table, registers, and storage near the register values at the time of the error.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=COMMTASK,COMPID=SC1CK,ISSUER=IEAVN701, FAILURE IN COMM TASK ADDRESS SPACE INITIALIZATION

Component Communications Task (5752-SC1CK)

Issuing Module: IEAVN701

PLM: *System Initialization Logic*

Explanation: An error has occurred while IEAVN701 was initializing the communications task address space. The areas dumped are ALLPSA, NUC, RGN, LSQA, SQA, CSA, TRT, and SUMDUMP. SUMDUMP contains the trace table, registers, and storage near the register values at the time of the error.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=DDR,COMPID=BB1CS,ISSUER=IGFDE1

Component: DDR (5752-BB1CS)

Issuing Module: IGFDE1

PLM: *System Logic Library*

Explanation: An error has occurred during DDR (dynamic device reconfiguration) processing. The areas dumped are SQA, PSA, and TRT. Generally, register 10 points to the DDRCOM control block (mapped by IHADDR).

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IGFDIO or IGC0003D
SDWACSCT - Currently active DDR module
SDWAREXN - IGFDE1
SDWACID - BB1CS
SDWARRL - IGFDE1
SDWAVRA - Variable data in key-length-data format - including the DERPLIST and exit data (if any).

COMPON=DEVSERV PATHS COMMAND,ISSUER=IGUDSP02 or IGUDSP03
COMPID=28463

Component: DEVSERV (5665-28463)

Issuing Module: IGUDSP02 or IGUDSP03

PLM: None

Explanation: During DEVSERV command processing, either an abend occurred or a dump was requested. The areas dumped are PSA, NUC, RGN, LPA, TRT, SQA, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IGUDS001
SDWACSCT - DEVSERV CSECT
SDWACID - 28463
SDWAMDAT - Assembly date
SDWAMVRS - Version/PTF level
SDWARRL - DSESTA or DVS3ESTA
SDWAVRA - Variable data in key-length-data format.

COMPON=DIDOCS-D U,,ALLOC PROC,COMPID=SC1C4,
ISSUER=IEE24110-DUESTAE

Component: DIDOCS (5752-SC1C4)

Issuing Module: IEE24110 - ESTAE

PLM: *System Logic Library*

Explanation: An error has occurred during D U,ALLOC (DISPLAY) processing. Any storage areas obtained are freed. The following areas are dumped for both the master and the allocation address space: LPA, TRT, and SUMDUMP. The ESTAE routine percolates to IEECB860.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=DISPLAY SMF COMMAND, ISSUER=IFADSMF, COMPID=SC100

Component: SMF (5752-SC100)

Issuing Module: IFADSMF

PLM: *System Logic Library*

Explanation: An abend has occurred during SMF DISPLAY command processing. The areas dumped are PSA, NUC, RGN, LPA, TRT, SQA, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - Load module in error
SDWACSCT - CSECT in error
SDWACID - Component ID
SDWASC - Failing function name
SDWARRL - ESTAE routine label
SDWAREXN - ESTAE module name

COMPON=GRS-COMMANDS,COMPID=SCSDS,ISSUER=ISGCRET0, POST OF GVTCECB FAILED

Component: Global Resource Serialization (5752-SCSDS)

Issuing Module: ISGCRET0

PLM: *System Logic Library*

Explanation: An error has occurred while one of the following modules was attempting to cross-memory post the command ECB being used by ISGCMDR.

ISGBSM - RSA send/receive routine
ISGCMDR - Command router
ISGGFRR0 - FRR for ENQ/DEQ;RESERVE
ISGGTRM1 - ENQ/DEQ/RESERVE termination resource manager

ISGCMDR was waiting for a command request or a message request. The areas dumped are PSA, SQA, and LSQA of the global resource serialization address space, and the GVT.

Problem Determination: Either the ECB address provided on the cross-memory post is in error, or the RB address in the ECB is in error.

COMPON=GRS-COMMANDS,COMPID=SCSDS,ISSUER=ISGCRET1,
POST OF ECB OF COMMAND REQUESTOR FAILED

Component: Global Resource Serialization (5752-SCSDS)

Issuing Module: ISGCRET1

PLM: *System Logic Library*

Explanation: An error has occurred while ISGCMDR (command router) was attempting to cross-memory post the ECB that was being used by a command requestor to wait for a command request to be processed by ISGCMDR. The areas dumped are PSA, SQA, and LSQA of the command requestor's address space, and the command requestor's ECB.

Problem Determination: Either the ECB address provided on the cross-memory post is in error, or the RB address in the ECB is in error.

COMPON=GRS-CTC-DRIVER,COMPID=SCSDS,ISSUER=ISGJRCV

Component: Global Resource Serialization (5752-SCSDS)

Issuing Module: ISGJRCV

PLM: *System Logic Library*

Explanation: An error has occurred while ISGJDI (CTC driver DIE) was processing. The FRR ISGJRCV (for ISGJDI) uses the branch entry to SVC dump. If the GCL is valid, a summary dump is requested that contains the GCV.

Problem Determination: A software record is written to SYS1.LOGREC. It includes the failing CSECT name plus the following in the variable recording area (SDWAVRA) of the SDWA:

- If the IOSB is valid:
  - UCBNAME field of the UCB.
  - IOSB identifier.
  - IOSB address.
  - IOSB Fields: IOSFLA, IOSFLB, IOSFLC, IOSCOD, IOSUCB, IOSCSW, IOSUSE, IOSSNS.
- If the GCQ is valid:
  - GCQ address.
  - GCQDISEC field.
- If the GCL is valid:
  - GCL address.
  - GCL (without IOSBs)
- If the IOSB, GCQ, and/or GCL is in error, then the IOSB, GCQ, and/or GCL address is recorded.

COMPON=GRS-CTC DRIVER ENF EXITS,
COMPID=SCSDS,ISSUER=ISGJENF0

Component: Global Resource Serialization (5752-SCSDS)

Issuing Module: ISGJENF0 - ESTAE

PLM: *System Logic Library*

Explanation: An error has occurred while ISGJENF0 (event notification facility exits routine) was processing. The ESTAE routine ISGJENFR (in ISGJENF0) issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC. It includes the ESTAE parameter list and the VARY parameter list in the variable recording area (SDWAVRA) of the SDWA.

COMPON=GRS-QUEUE SCANNING SERVICES,COMPID=SCSDS,
ISSUER=ISGQSCNR

Component: Global Resource Serialization (5752-SCSDS)

Issuing Module: ISGQSCNR - FRR

PLM: *System Logic Library*

Explanation: An error has occurred while ISGQSCAN (queue scanning services) was processing. The FRR routine ISGQSCNR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=GRS-RING-PROCESSING,COMPID=SCSDS,
ISSUER=ISGBERCV

Component: Global Resource Serialization (5752-SCSDS)

Issuing Module: ISGBERCV - ESTAE

PLM: *System Logic Library*

Explanation: An error has occurred while ISGBTC (ring processing task mode routine) or ISGBCI (ring processing command interface routine) was processing. ESTAE routine ISGBERCV issues the SDUMP macro. If the basic control blocks are valid, a summary dump is requested that includes the GVT, SQA (obtained by ISGBTC), and the private area (obtained by (ISGBTC) for ring processing. An asynchronous dump of the current address space is always included in the dump request.

Problem Determination: A software record is written to SYS1.LOGREC. It includes the failing CSECT name plus the following in the variable recording area (SDWAVRA) of the SDWA:

- Address of ISGREPL (input parameter list to ISGBERCV).
- The ISGREPL.
- Address of ISGRSC (input parameter list to ISGBCI).
- The ISGRSC if ISGBTC failed.

COMPON=GRS-RING-PROC,COMPID=SCSDS,ISSUER=ISGBFRCV

Component: Global Resource Serialization (5752-SCSDS)

Issuing Module: ISGBFRCV - FRR

PLM: *System Logic Library*

Explanation: An error has occurred while ISGBSR (RSA send/receive routine) was processing. The FRR ISGBFRCV uses the branch entry to SVC dump. If the basic control blocks are valid, a summary dump is requested that includes the GVT, SQA (obtained by ISGBTC), and the private area (obtained by ISGBTC) for ring processing. An asynchronous dump of the current address space is always included in the dump request.

Problem Determination: A software record is written to SYS1.LOGREC. It includes the failing CSECT name plus the following in the variable recording area SDWAVRA) of the SDWA:

- RVRPARM address.
- The RVRPARM (input parameter area to ISGBFRCV).
- RSV ID and address.
- The following fields in the RSV control block:

RSVCRSAT	RSVTRSL	RSVIBFOR (input buffer)
RSVRSLR	RSVERR	RSAMRPFX (input buffer)
RSVRSLS	RSVWLOCK	RSVOBFOR (output buffer)
RSVRSLRF	RSVRSASC	RSAMRPFX (output buffer)
RSVRSLSF	RSVCPHNO	

- RSL ID and address.
- The following fields in the RSL control block (if the error occurred in ISGBSRRI):

RSLWLOCK	RSLLKIF
RSLLNKI	RSLBFCT
RSLNMSC	RSLERR
RSLLKSF	

COMPON=GRS,COMPID=SCSDS,ISSUER=ISGDSNRV

Component: Global Resource Serialization (5752-SCSDS)

Issuing Module: ISGDSNAP

PLM: *System Logic Library*

Explanation: An error has occurred while ISGDSNAP (snap dump exit) was processing. ESTAE routine ISGDSNRV (in ISGDSNAP) issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=GRS,COMPID=SCSDS,ISSUER=ISGGFRR0

Component: Global Resource Serialization (5752-SCSDS)

Issuing Module: ISGGFRR0 - FRR

PLM: *System Logic Library*

Explanation: An error has occurred while one of the following modules was processing.

ISGGDEQP	ISGGREX0	ISGLNQDQ
ISGGNQDQ	ISGGRP00	ISGSALC
ISGGQWBC	ISGGTRM0	ISGSDAL
ISGGQWBI	ISGGTRM1	ISGSHASH

The FRR ISGGFRR0 uses the branch entry to SVC dump. A summary dump is requested that includes the GVT and GVTX control blocks. An asynchronous dump of the current address space is also included in the dump request.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=GRS,COMPID=SCSDS,ISSUER=ISGSMIFR

Component: Global Resource Serialization (5752-SCSDS)

Issuing Module: ISGSMI

PLM: *System Logic Library*

Explanation: Either a program check has occurred while ISGSMI, ISGSALC, or ISGSDAL was processing, or an abend has occurred while ISGSALC was processing. The FRR routine ISGSMIFR (in ISGSMI) uses the branch entry to SVC dump. The areas dumped are PSA, SQA, and GRSQ. The dump also contains a summary dump.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=IOS,COMPID=SC1C3,ISSUER=IECVPST,PSTFRRTN

Component: IOS (5752-SC1C3)

Issuing Module: IECVPST

PLM: *System Logic Library*

Explanation: The IOS post status FRR has received control because of a program check. The error might have occurred in IECVPST or in an exit (such as an ABEND or PCI). The areas dumped are ALLPSA, SQA, LSQA, SUMDUMP, TRT, and NUC.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IECVPST
SDWAREXN - IECVPST
SDWACID - SC1C3
SDWARRL - PSTFRRTN
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSPURGA,IOSPGRVR

Component: IOS (5752-SC1C3)

Issuing Module: IOSPURGA

PLM: *System Logic Library*

Explanation: An error has occurred in purge or prevention mainline processing. Recovery routine IOSPGRVR issues the SDUMP macro. The areas dumped are purge's dynamic workarea, PSA, SQA, TRT, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSPURGA
SDWAREXN - IOSPURGA
SDWACID - SC1C3
SDWARRL - IOSPGRVR
SDWASC - IOS PURGE FUNCTION
SDWAVRA - UCB information if UCB lock held at time of error.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSRACRW,ACRWFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSRACRW

PLM: *System Logic Library*

Explanation: An error has occurred while IOS was processing a channel report word (CRW). The areas dumped are NUC, SQA, ALLPSA, TRT, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSRACRW
SDWAREXN - IOSRACRW
SDWACID - SC1C3
SDWARRL - ACRWFRR
SDWASC - IOS- CRW ASYNCH PROC or
- IOSRACRW - FRR RECURSION
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSRCHPR,CHPRFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSRCHPR

PLM: *System Logic Library*

Explanation: An error has occurred during channel path recovery processing. Routine CHPRFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSRCHPR
SDWAREXN - IOSRCHPR
SDWACID - SC1C3
SDWARRL - CHPRFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSRDBOX,BOXFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSRDBOX

PLM: *System Logic Library*

Explanation: An error has occurred while a device was being boxed. The areas dumped are SQA, PSA, TRT, and SUM.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSRDBOX
SDWAREXN - IOSRDBOX
SDWACID - SC1C3
SDWARRL - BOXFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSRHDET

Component: IOS (5752-SC1C3)

Issuing Module: IOSRHDET

PLM: *System Logic Library*

Explanation: An error has occurred while IOS was checking for a hot I/O condition. Routine HDETFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSRHDET
SDWAREXN - IOSRHDET
SDWACID - SC1C3
SDWARRL - HDETFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSRHREC

Component: IOS (5752-SC1C3)

Issuing Module: IOSRHREC

PLM: *System Logic Library*

Explanation: An error has occurred while IOS was performing hot I/O recovery. Routine HRECFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSRHREC
SDWAREXN - IOSRHREC
SDWACID - SC1C3
SDWARRL - HRECFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSRMIHP,MIHPFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSRMIHP

PLM: *System Logic Library*

Explanation: An error has occurred during processing in the missing interruption handler. Routine MIHPFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSRMIHP
SDWAREXN - IOSRMIHP
SDWACID - SC1C3
SDWARRL - MIHPFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSRMIHR,MIHRFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSRMIHR

PLM: *System Logic Library*

Explanation: An error has occurred during processing in the missing interruption handler. Routine MIHRFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IEANUC01
SDWACSCT - IOSRMIHR
SDWAREXN - IOSRMIHR
SDWACID - SC1C3
SDWARRL - MIHRFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSRMIHT-
MISSING INTERRUPT HANDLER ROUTINE

Component: IOS (5752-SC1C3)

Issuing Module: IOSRMIHT

PLM: *System Logic Library*

Explanation: An error has occurred during processing in the missing interruption handler (module IOSRMIHL). Routine ESTIENTL issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IOSRMIHL
SDWACSCT - IOSRMIHT
SDWAREXN - IOSRMIHT
SDWACID - SC1C3
SDWARRL - ESTIENTL
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSRMIHT-
MISSING INTERRUPT HANDLER ROUTINE

Component: IOS (5752-SC1C3)

Issuing Module: IOSRMIHT

PLM: *System Logic Library*

Explanation: An error has occurred during processing in the missing interruption handler (module IOSRMIHM). Routine ESTIENTM issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IOSRMIHM
SDWACSCT - IOSRMIHT
SDWAREXN - IOSRMIHT
SDWACID - SC1C3
SDWARRL - ESTIENTM
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSRMIHT-
MISSING INTERRUPT HANDLER ROUTINE

Component: IOS (5752-SC1C3)

Issuing Module: IOSRMIHT

PLM: *System Logic Library*

Explanation: An error has occurred during processing in the missing interruption handler (module IOSRMIHT). Routine MIHESTAE issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IOSRMIHT
SDWACSCT - IOSRMIHT
SDWAREXN - IOSRMIHT
SDWACID - SC1C3
SDWARRL - MIHESTAE
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSRRRSV

Component: IOS (5752-SC1C3)

Issuing Module: IOSRRRSV

PLM: *System Logic Library*

Explanation: An error has occurred during recovery processing for a device or channel path. Routine RRSVFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSRRRSV
SDWAREXN - IOSRRRSV
SDWACID - SC1C3
SDWARRL - RRSVFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSRSCH,RSCHFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSRSCH

PLM: *System Logic Library*

Explanation: An error has occurred during recovery processing for the subchannel CRW (channel report word). The areas dumped are NUC, SQA, ALLPSA, TRT, SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSRSCH
SDWAREXN - IOSRSCH
SDWACID - SC1C3
SDWARRL - RSCHFRR
SDWASC - IOS - SUBCHANNEL RECOVERY or
- IOSRSCH - SECOND FRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVFCHP,FCHPESTA

Component: IOS (5752-SC1C3)

Issuing Module: IOSVFCHP

PLM: *System Logic Library*

Explanation: An error has occurred during IOSVFCHP (force channel path offline) processing. ESTAE routine FCHPESTA issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IOSVFCHP
SDWACSCT - IOSVFCHP
SDWAREXN - IOSVFCHP
SDWACID - SC1C3
SDWARRL - FCHPESTA
SDWAVRA - Variable data in key-length-data format - including the channel path identifier to be forced offline, the input parameter list, module work area, and CRWQ control block.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVHSCH,HSCHFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSVHSCH

PLM: *System Logic Library*

Explanation: An error has occurred during HSCH (halt) or CSCH (clear) subchannel processing. The areas dumped are SQA, PSA, TRT, and SUM.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVHSCH
SDWAREXN - IOSVHSCH
SDWACID - SC1C3
SDWARRL - HSCHFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVIPID,VIPIDFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSVIPID

PLM: *System Logic Library*

Explanation: An error has occurred while IOS was processing a caller's request to obtain or release an I/O prevention identifier. The areas dumped are NUC, SQA, ALLPSA, TRT, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVIPID
SDWAREXN - IOSVIPID
SDWACID - SC1C3
SDWARRL - VIPIDFRR
SDWASC - IOS - IOPID SERVICE ROUTINE or
- IOSVIPID - FRR RECURSION
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVIRBA,IRBAFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSVIRBA

PLM: *System Logic Library*

Explanation: An error has occurred while subchannel status, probably signaled by an I/O interruption, was being processed. Routine IRBAFRR issues the SDUMP macro.

The address space dumped is the address space associated with the I/O request being processed. This address space might not match the current ASID in the associated LOGREC entry.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVIRBA
SDWAREXN - IOSVIRBA
SDWACID - SC1C3
SDWARRL - IRBAFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVIRBD,IRBDFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSVIRBD

PLM: *System Logic Library*

Explanation: An error has occurred during IRB device status processing. The areas dumped are NUC, SQA, ALLPSA, TRT, and SUMDUMP.

The address space dumped is the address space associated with the I/O request being processed. This address space might not match the current ASID in the associated LOGREC entry.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVIRBD
SDWAREXN - IOSVIRBD
SDWACID - SC1C3
SDWARRL - VIRBFRR
SDWASC - IOS - IRB DEVICE STATUS or
- IOSVIRBD - SECOND FRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVIRBH,IRBHFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSVIRBH

PLM: *System Logic Library*

Explanation: An error has occurred during IRB halt (HSCH) or clear (CSCH) status processing. The areas dumped are NUC, SQA, ALLPSA, TRT, and SUMDUMP.

The address space dumped is the address space associated with the I/O request being processed. This address space might not match the current ASID in the associated LOGREC entry.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVIRBH
SDWAREXN - IOSVIRBH
SDWACID - SC1C3
SDWARRL - VIRBHFRR
SDWASC - IOS - IRB HALT/CLEAR or
- IOSVIRBH - SECOND FRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVIRBN,IRBNFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSVIRBN

PLM: *System Logic Library*

Explanation: An error has occurred during IRB N-bit or deferred CC3 processing. The areas dumped are NUC, SQA, ALLPSA, TRT, and SUMDUMP.

The address space dumped is the address space associated with the I/O request being processed. This address space might not match the current ASID in the associated LOGREC entry.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVIRBN
SDWAREXN - IOSVIRBN
SDWACID - SC1C3
SDWARRL - VIRBNFRR
SDWASC - IOS - IRB N/DCC3 STATUS or
- IOSVIRBN - SECOND FRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVIRBU,UNSOLFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSVIRBU

PLM: *System Logic Library*

Explanation: An error has occurred while unsolicited subchannel status, probably signaled by an I/O interruption, was being processed. Routine UNSOLFRR issues the SDUMP macro.

The address space dumped is the address space associated with the I/O request being processed. This address space might not match the current ASID in the associated LOGREC entry.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVIRBU
SDWAREXN - IOSVIRBU
SDWACID - SC1C3
SDWARRL - UNSOLFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVLEVL

Component: IOS (5752-SC1C3)

Issuing Module: IOSVLEVL

PLM: *System Logic Library*

Explanation: An error has occurred while IOS was managing the serialization (LEVEL) for a UCB. Routine LVLFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVLEVL
SDWAREXN - IOSVLEVL
SDWACID - SC1C3
SDWARRL - LVLFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVMSCH,IOSMSCHF,
ERROR DURING MODIFY SUBCHANNEL INIT

Component: IOS (5752-SC1C3)

Issuing Module: IOSVMSCH

PLM: *System Logic Library*

Explanation: An error has occurred during modify subchannel (MSCH) initialization. The areas dumped are SQA, PSA, TRT, and SUM.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVMSCH
SDWAREXN - IOSVMSCH
SDWACID - SC1C3
SDWARRL - IOSMSCHF
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVMSCQ,IOSMSCQF

Component: IOS (5752-SC1C3)

Issuing Module: IOSVMSCQ

PLM: *System Logic Library*

Explanation: An error has occurred during modify subchannel (MSCH) queue processing. The areas dumped are SQA, PSA, TRT, and SUM.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVMSCQ
SDWAREXN - IOSVMSCQ
SDWACID - SC1C3
SDWARRL - IOSMSCQF
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVPRVT,VPRVTFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSVPRVT

PLM: *System Logic Library*

Explanation: An error has occurred while IOS was processing a caller's request to perform I/O prevention. The areas dumped are NUC, SQA, ALLPSA, TRT, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVPRVT
SDWAREXN - IOSVPRVT
SDWACID - SC1C3
SDWARRL - VPRVTFRR
SDWASC - IOS - IOS PREVENTION ROUTINE or
- IOSVPRVT - FRR RECURSION
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVRSUM-
RESUME SERVICE ROUTINE

Component: IOS (5752-SC1C3)

Issuing Module: IOSVRSUM

PLM: *System Logic Library*

Explanation: An error has occurred while the resume service routine (IOSVRSUM) was processing. Routine RSUMFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVRSUM
SDWAREXN - IOSVRSUM
SDWACID - SC1C3
SDWARRL - RSUMFRR
SDWAVRA - Variable data in key-length-data format - including the UCB and IOSB.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVSLIH,SLIHFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSVSLIH

PLM: *System Logic Library*

Explanation: An error has occurred while the IOS second level interruption handler (SLIH) was processing. The areas dumped are SQA, PSA, TRT, and SUM.

The address space dumped is the address space associated with the I/O request being processed. This address space might not match the current ASID in the associated LOGREC entry.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVSLIH
SDWAREXN - IOSVSLIH
SDWACID - SC1C3
SDWARRL - SLIHFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVSSCH,IOSSSCHF

Component: IOS (5752-SC1C3)

Issuing Module: IOSVSSCH

PLM: *System Logic Library*

Explanation: An error has occurred during start subchannel (SSCH) processing. The areas dumped are SQA, PSA, TRT, and SUM.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVSSCH
SDWAREXN - IOSVSSCH
SDWACID - SC1C3
SDWARRL - IOSSSCHF
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVSSCQ,SSCQFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSVSSCQ

PLM: *System Logic Library*

Explanation: An error has occurred while routine IOSVSSCQ was processing. Routine SSCQFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVSSCQ
SDWAREXN - IOSVSSCQ
SDWACID - SC1C3
SDWARRL - SSCQFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVSTSC,STSCFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSVSTSC

PLM: *System Logic Library*

Explanation: An error has occurred during IOSVSTSC (IOS store subchannel routine) processing. FRR routine STSCFRR issues the SDUMP macro. The areas dumped are SQA, ALLPSA, SUMDUMP, and TRT.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IEANUC01
SDWACSCT - IOSVSTSC
SDWAREXN - IOSVSTSC
SDWACID - SC1C3
SDWARRL - STSCFRR
SDWAVRA - Variable data in key-length-data format - including the 24-byte FRR work area, and IOSB and UCB fields.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVSTSQ,STSQFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSVSTSQ

PLM: *System Logic Library*

Explanation: An error has occurred during IOSVSTSQ (STSCH queue routine) processing. FRR routine STSQFRR issues the SDUMP macro. The areas dumped are SQA, ALLPSA, SUMDUMP, and TRT.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IEANUC01
SDWACSCT - IOSVSTSQ
SDWAREXN - IOSVSTSQ
SDWACID - SC1C3
SDWARRL - STSQFRR
SDWAVRA - Variable data in key-length-data format - including the 24-byte FRR work area, and the IOSB and UCB.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVSWAP,SWAPFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSVSWAP

PLM: *System Logic Library*

Explanation: An error has occurred while IOS was executing a swap between UCBs. Routine SWAPFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVSWAP
SDWAREXN - IOSVSWAP
SDWACID - SC1C3
SDWARRL - SWAPFRR
SDWAVRA - Variable data in key-length-data format - including from-UCB and to-UCB data.

COMPON=IOS,COMPID=SC1C3,ISSUER=IOSVVARY

Component: IOS (5752-SC1C3)

Issuing Module: IOSVVARY

PLM: *System Logic Library*

Explanation: An error has occurred while a path to a device was being varied online or offline. Routine VARYFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVVARY
SDWAREXN - IOSVVARY
SDWACID - SC1C3
SDWARRL - VARYFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS-DASD VOLUME VERIFICATION,
COMPID=SC1C3,ISSUER=IOSVDAVV

Component: IOS (5752-SC1C3)

Issuing Module: IOSVDAVV

PLM: *System Logic Library*

Explanation: An error has occurred while IOS was attempting to verify the volume label for a DASD device. Routine DAVVFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVDAVV
SDWAREXN - IOSVDAVV
SDWACID - SC1C3
SDWARRL - DAVVFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS-DYNAMIC PATHING,
COMPID=SC1C3,ISSUER=IECVDPTH

Component: IOS (5752-SC1C3)

Issuing Module: IECVDPTH

PLM: *System Logic Library*

Explanation: An error has occurred during IECVDPTH (dynamic pathing) processing. ESTAE routine DPTHESTA issues the SDUMP macro. The areas dumped are SQA, TRT, and SUM.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IECVDPTH
SDWAREXN - IECVDPTH
SDWACID - SC1C3
SDWARRL - DPTHESTA
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS-DYNAMIC PATHING,
COMPID=SC1C3,ISSUER=IECVDPTH

Component: IOS (5752-SC1C3)

Issuing Module: IECVDPTH

PLM: *System Logic Library*

Explanation: An error has occurred during IECVDPTH (dynamic pathing) processing. FRR routine DPTHFRR issues the SDUMP macro. The areas dumped are SQA, TRT, and SUM.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IECVDPTH
SDWAREXN - IECVDPTH
SDWACID - SC1C3
SDWARRL - DPTHFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS-DYNAMIC PATHING DRIVER,
COMPID=SC1C3,ISSUER=IOSVDPDR

Component: IOS (5752-SC1C3)

Issuing Module: IOSVDPDR

PLM: *System Logic Library*

Explanation: An error has occurred during IOSVDPDR (dynamic pathing driver routine) processing. FRR routine DPDRFRR issues the SDUMP macro. The areas dumped are SQA, TRT, and SUM.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVDPDR
SDWAREXN - IOSVDPDR
SDWACID - SC1C3
SDWARRL - DPDRFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS-DYNAMIC PATHING INIT,COMPID=SC1C3,
ISSUER=IECVIOSI

Component: IOS (5752-SC1C3)

Issuing Module: IECVIOSI

PLM: *System Logic Library*

Explanation: An error has occurred during IECVIOSI (IOS initialization) processing. ESTAE routine IOSIRECV issues the SDUMP macro. The module work area is dumped.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IECVIOSI
SDWACSCT - IECVIOSI
SDWAREXN - IECVIOSI
SDWACID - SC1C3
SDWARRL - IOSIRECV
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS-IOS CHANNEL PATH DEVICE RECOVERY,
COMPID=SC1C3,ISSUER=IOSRCHDR,CHDRFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSRCHDR

PLM: *System Logic Library*

Explanation: An error has occurred while IOS was performing channel path device recovery. FRR routine CHDRFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSRCHDR
SDWAREXN - IOSRCHDR
SDWACID - SC1C3
SDWARRL - CHDRFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS-IOS CHANNEL PATH DEVICE SCAN,
COMPID=SC1C3,ISSUER=IOSRCHDS,CHDSFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSRCHDS

PLM: *System Logic Library*

Explanation: An error has occurred while IOS was processing devices after a channel path error. Routine CHDSFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSRCHDS
SDWAREXN - IOSRCHDS
SDWACID - SC1C3
SDWARRL - CHDSFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS-IOS CLEAR DEVICE SUBCHANNEL ROUTINE,
COMPID=SC1C3,ISSUER=IOSRCDEV,CDEVFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSRCDEV

PLM: *System Logic Library*

Explanation: An error occurred while IOS was attempting to clear a subchannel. FRR routine CDEVFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSRCDEV
SDWAREXN - IOSRCDEV
SDWACID - SC1C3
SDWARRL - CDEVFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS-IOS FORCE DEVICE ROUTINE,
COMPID=SC1C3,ISSUER=IOSRFDEV,FDEVFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSRFDEV

PLM: *System Logic Library*

Explanation: An error has occurred while IOS was attempting to force a device offline. FRR routine FDEVFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSRFDEV
SDWAREXN - IOSRFDEV
SDWACID - SC1C3
SDWARRL - FDEVFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS-IOS STORAGE MANAGER,COMPID=SC1C3,
ISSUER=IOSVSMGR,IOSVSMFR,error

Component: IOS (5752-SC1C3)

Issuing Module: IOSVSMGR

PLM: *System Logic Library*

Explanation: An error (GETMAIN FAILURE, PROGRAM ERROR, or ABEND=C0D) has occurred while the IOS storage manager was processing a caller's request. The areas dumped are NUC, SQA, TRT, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVSMGR
SDWAREXN - IOSVSMGR
SDWACID - SC1C3
SDWARRL - IOSVSMFR
SDWASC - IOS - STORAGE MANAGER
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS-PATH VALIDATION,COMPID=SC1C3,
ISSUER=IECVIOPM,PMSKESTE

Component: IOS (5752-SC1C3)

Issuing Module: IECVIOPM

PLM: *System Logic Library*

Explanation: An error has occurred during IECVIOPM (I/O path mask update routine) processing. The areas dumped are NUC, SQA, LSQA, TRT, and PSA.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IECVIOPM
SDWACSCT - IECVIOPM
SDWAREXN - IECVIOPM
SDWACID - SC1C3
SDWARRL - PMSKESTE
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS-RESTART SUPPORT,COMPID=SC1C3,
ISSUER=IOSVRSTS,RSTSFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSVRSTS

PLM: *System Logic Library*

Explanation: An error has occurred while IOS was processing a restart request. FRR routine RSTSFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVRSTS
SDWAREXN - IOSVRSTS
SDWACID - SC1C3
SDWARRL - RSTSFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS (SC1C3), STAND-ALONE I/O RTN,
ISSUER=IOSRSAIO(SAIOFRR)

Component: IOS (5752-SC1C3)

Issuing Module: IOSRSAIO

PLM: *System Logic Library*

Explanation: An error has occurred while IOS was attempting to initiate a stand-alone I/O operation. FRR routine SAIOFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSRSAIO
SDWAREXN - SAIOFRR
SDWACID - SC1C3
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS-SHARED UP SERVICE, COMPID=SC1C3,
ISSUER=IOSVSHUP

Component: IOS (5752-SC1C3)

Issuing Module: IOSVSHUP

PLM: *System Logic Library*

Explanation: An error has occurred while IOSVSHUP was processing. The FRR routine SHUPFRR issues the SDUMP macro. The areas dumped are SQA, TRT, and SUM.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVSHUP
SDWAREXN - IOSVSHUP
SDWACID - SC1C3
SDWARRL - SHUPFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS (SC1C3), STAND-ALONE I/O RTN,
ISSUER=IOSRSUBC(SAIOFRR)

Component: IOS (5752-SC1C3)

Issuing Module: IOSRSAIO

PLM: *System Logic Library*

Explanation: An error has occurred while IOS was attempting to set or reset the stand-alone I/O interruption subclass for a subchannel. FRR routine SAIOFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSRSAIO
SDWAREXN - SAIOFRR
SDWACID - SC1C3
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS-SIMULATED INTERRUPT,COMPID=SC1C3,
ISSUER=IECVGENA

Component: IOS (5752-SC1C3)

Issuing Module: IECVGENA

PLM: *System Logic Library*

Explanation: An error has occurred while IECVGENA was simulating an interruption. FRR routine GENAFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IECVGENA
SDWAREXN - IECVGENA
SDWACID - SC1C3
SDWARRL - GENAFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS-SUBCHANNEL LOGOUT,COMPID=SC1C3,
ISSUER=IOSRSLH,SLHFRR

Component: IOS (5752-SC1C3)

Issuing Module: IOSRSLH

PLM: *System Logic Library*

Explanation: An error has occurred while IOS was processing a subchannel logout. FRR routine SLHFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSRSLH
SDWAREXN - IOSRSLH
SDWACID - SC1C3
SDWARRL - SLHFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS-SUBCHANNEL REDRIVE,COMPID=SC1C3,
ISSUER=IOSVSCHR

Component: IOS (5752-SC1C3)

Issuing Module: IOSVSCHR

PLM: *System Logic Library*

Explanation: An error has occurred during subchannel redrive processing. The areas dumped are SQA, PSA, TRT, and SUM.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVSCHR
SDWAREXN - IOSVSCHR
SDWACID - SC1C3
SDWARRL - SCHRFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS-UCBFLG FUNCTION,COMPID=SC1C3,
ISSUER=IECVGENA

Component: IOS (5752-SC1C3)

Issuing Module: IECVGENA

PLM: *System Logic Library*

Explanation: An error has occurred while IECVGENA was modifying a flag in the UCB. FRR routine GENAFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IECVGENA
SDWAREXN - IECVGENA
SDWACID - SC1C3
SDWARRL - GENAFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS-UNCONDITIONAL RESERVE,COMPID=SC1C3,
ISSUER=IOSVURDT

Component: IOS (5752-SC1C3)

Issuing Module: IOSVURDT

PLM: *System Logic Library*

Explanation: An error has occurred while IOSVURDT, IECVDURP, or IOSVURSV (unconditional reserve back-end routines) was processing. The areas dumped are SQA, TRT, and SUM.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVURDT
SDWAREXN - IOSVURDT
SDWACID - SC1C3
SDWARRL - URDTFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=IOS-UNCONDITIONAL RESERVE,COMPID=SC1C3,
ISSUER=IOSVURVL

Component: IOS (5752-SC1C3)

Issuing Module: IOSVURVL

PLM: *System Logic Library*

Explanation: An error has occurred during IOSVURVL (unconditional reserve front-end routine) processing. The areas dumped are SQA, TRT, and SUM.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - IOSVURVL
SDWAREXN - IOSVURVL
SDWACID - SC1C3
SDWARRL - URVLFRR
SDWAVRA - Variable data in key-length-data format.

COMPON=JES3 SUBSYS COMMUNIC,COMPID=SC1BA,
ISSUER=IATSSRE(SSREFRR)

Component: JES3 (5752-SC1BA)

Issuing Module: IATSSRE

PLM: *JES3 Logic*

Explanation: An error has occurred during read end processing of subsystem communication. Recovery routine SSREFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=JES3 SUBSYS COMMUNIC,COMPID=SC1BA,
ISSUER=IATSSXM(SXMFRR)

Component: JES3 (5752-SC1BA)

Issuing Module: IATSSXM

PLM: *JES3 Logic*

Explanation: An error has occurred during cross memory processing of subsystem communication. Recovery routine SXMFRR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=JSS-REC,COMPID=SC1B8,ISSUER=IEESB670,
JOB SCHEDULING SUBROUTINE RECOVERY EXIT ROUTINE

Component: Master Scheduler Commands (5752-SC1B8)

Issuing Module: IEESB670

PLM: *System Logic Library*

Explanation: The recovery exit routine IEESB670 schedules a retry of the job scheduling subroutine (IEESB605). If an SDWA is provided, IEESB670 issues the SDUMP macro. The areas dumped are SQA, PSA, LSQA, RGN, LPA, TRT, CSA, and NUC.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=MSTR-BASE,COMPID=SC1B8,ISSUER=IEEVIPL
ERROR IN MASTER SCHEDULER INITIALIZATION

Component: Master Scheduler Commands (5752-SC1B8)

Issuing Module: IEEVIPL - Master Scheduler Base Initialization

PLM: *System Logic Library*

Explanation: During error recovery processing, SDUMP is issued if (1) STAE processing was unsuccessful, (2) a program check occurred, (3) the system restart key was pressed, or (4) control was returned because system initialization terminated. The areas dumped are PSA, LSQA, RGN, LPA, TRT, CSA, ALLNUC, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=MSTR-REGION,COMPID=SC1B8,ISSUER=IEEMB860,
MASTER SCHEDULER REGION INITIALIZATION DUMP

Component: Master Scheduler Commands (5752-SC1B8)

Issuing Module: IEEMB860 - Master Scheduler Region Initialization

PLM: *System Initialization Logic*

Explanation: Either ESTAE or recovery termination setup failed. The error occurs if the LOAD macro (SVC 8) was unsuccessful, or master scheduler initialization failed. The areas dumped are PSA, ALLNUC, LSQA, RGN, LPA, TRT, CSA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=MSTR-WAIT,COMPID=SC1B8,ISSUER=IEEVWAIT,reason

Component: Master Scheduler Commands (5752-SC1B8)

Issuing Module: IEEVWAIT

PLM: *System Logic Library*

Explanation: An error has occurred during command processing. The 'reason' field is one of the following:

BAD ESTAE RETURN CODE
ERROR IN MASTER ADDR SPACE
ERROR IN CONSOLE ADDR SPACE
IEEVWAIT RESTART FAILED IN CONSOLE ADDR SPACE

IEEVWAIT issues SDUMP for all but percolation and machine check entries. The areas dumped are PSA, NUC, LSQA, RGN, LPA, TRT, CSA, GRSQ, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=M S CMNDS,COMPID=SC1B8,ISSUER=IEECB800,
FAILURE IN TRACK COMMAND

Component: Master Scheduler Commands (5752-SC1B8)

Issuing Module: IEECB800 - DISPLAY TRACK Common Processor

PLM: *System Logic Library*

Explanation: An error has occurred during TRACK command processing. Routine STAEXIT in IEECB800 issues the SDUMP macro. The areas dumped are NUC, LSQA, LPA, TRT, ALLPSA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=M S CMNDS,COMPID=SC1B8,ISSUER=IEECB862,
FAILURE IN VARY ONLINE/OFFLINE/CONSOLE PROCESSOR

Component: Master Scheduler Commands (5752-SC1B8)

Issuing Module: IEECB862

PLM: *System Logic Library*

Explanation: An error has occurred in the VARY device command. The areas dumped are SQA, ALLPSA, LSQA, LPA, TRT, and GRSQ. In addition, the UCM and UCMEs are dumped using a storage list.

Problem Determination: A software record is written to SYSl.LOGREC. The SDWAVRA contains the following:

- Pointer to the VARY service routine interface list (VSRI)
- The vary footprints
- Pointer to the XSA
- Pointer to the CSCB
- The command operand from CHBUF
- The command verb code
- The caller's token

COMPON=M S CMNDS,COMPID=SC1B8,ISSUER=IEEMB881, FAILURE IN SYSTEM ADDR SPACE CREATE ROUTINE

Component: Master Scheduler Commands (5752-SC1B8)

Issuing Module: IEEMB881 - System address space create routine

PLM: *System Initialization Logic*

Explanation: An error occurred, after master scheduler initialization, while IEEMB881 was attempting to start a system address space. Routine EAESTAE issues the SDUMP macro. The areas dumped are SQA, ALLPSA, SUMDUMP, LSQA, LPA, TRT, GRSQ, and the master scheduler's ASCB.

Problem Determination: A software record is written to SYS1.LOGREC. The SDWAVRA contains the following:

- Return and reason codes
- Footprints
- Input attribute list
- Name of the initialization routine specified by the caller
- Start parameters specified by the caller
- Code and data registers
- Pointers to the CSCB, ASCB, JSCB, TCB, and BASEA

COMPON=M S CMDS,COMPID=SC1B8,ISSUER=IEEMB883, FAILURE IN SYSTEM ADDR SPACE INIT WAIT/POST ROUTINE

Component: Master Scheduler Commands (5752-SC1B8)

Issuing Module: IEEMB883 - System address space initialization WAIT/POST routine

PLM: *System Initialization Logic*

Explanation: An error occurred, after master scheduler initialization, during WAIT/POST processing. Routine WPESTAE issues the SDUMP macro. The areas dumped are SQA, ALLPSA, LSQA, LPA, and TRT.

Problem Determination: A software record is written to SYS1.LOGREC. The SDWAVRA contains the following:

- Return and reason codes
- Input event code
- Footprints
- Code and data registers
- Pointer to TCB in error
- Pointers to the CSCB, ASCB, JSCB, and BASEA

COMPON=MS CMNDS,COMPID=SC1B8,ISSUER=IEEMB887,
GENERALIZED PARSER,ABEND=xxx,
RSN=xxxxxxxx|UNKNOWN
or
COMPON=MS CMNDS,COMPID=SC1B8,ISSUER=IEEMB887,
GENERALIZED PARSER-EXIT ABENDED,ABEND=xxx,
RSN=xxxxxxxx|UNKNOWN

Component: Master Scheduler (SC1B8)

Issuing Module: IEEMB887 - Generalized Parser

PLM: *System Logic Library*

Explanation: An error has occurred: (1) in module IEEMB887, or (2) in an exit routine that was called by IEEMB887.

Recovery routine PRSESTAE issued a summary SVC dump with the following areas included:

- IEEMB887
- IEEMB887's data area
- SCL (IEEMB887's parameter list)
- First parse description
- Current parse description
- Input being processed

Problem Determination: A software record is written to SYS1.LOGREC. The SDWAVRA contains:

- 'ENABLING DAE'
- If 'ROUT' exit routine abended, exit routine address with the address of the keyword used to call the routine
- If I/O exit abended, exit routine address
- Footprints
- Base registers
- Data register
- Address of SCL
- Address of current parse description
- Current value of input record pointer

COMPON=PROGRAM-MANAGER-LNKLST-LOOKASIDE,
COMPID=SC1CJ, ISSUER=CSVLLCES-CSVLLCRE

Component: Contents Supervisor (5752-SC1CJ)

Issuing Module: CSVLLCRE - issued by ESTAE CSVLLCES

PLM: *System Logic Library*

Explanation: An abend (other than code 222, 322, or 522) occurred while (1) LNKLST lookaside (LLA) was building or refreshing the LLA directory, or (2) the LLA directory was being searched and the caller of LLA determined that LLA caused the error. The caller terminates LLA with a 312 abend code.

Up to six dump ranges are dumped and include:

- The LLA control block in the nucleus pointed to by CVTLLCB.
- The oldest hash table and its overflow area.
- The replacement hash table and its overflow area.
- The temporary table of PDS directory entries (INFOTAB).
- The LNKLST table (LLT) pointed to by CVTLLTA.
- The LPALST table (LPAT) pointed to by CVTEPLPS.

Problem Determination: Except for operator cancel abends (codes 222 and 122), a software record is written to SYS1.LOGREC.

Variable SDWAPTR in module CSVLLCRE contains the address of the SDWA. The fields in the SDWA filled in are: SDWAMODN, SDWACSCT, SDWAREXN, SDWASC, SDWAMLVL, SDWARRL, and SDWACID.

The variable area in the SDWA (SDWAVRA) contains CSVLLCRE's processing status footprints (field FPCRE in CSVLLCRE), and data from the LLCB (field FPCES in CSVLLCRE).

Field CVTLLCB points to the LLA control block (LLCB) in nucleus module CSVLLCB1. LLCBASCB contains the address of the ASCB of the current LLA address space. The LLCB contains processing status flags and LLA-related data.

Field FOOTPRTS in CSVLLCRE contains footprints indicating the processing status and the resources that were owned by CSVLLCRE at the time of the error.

COMPON=PROGRAM-MANAGER-VIRTUAL-FETCH,COMPID=SC1CJ,
ISSUER=CSVVFCES-CSVVFCRE

Component: Contents Supervisor (5752-SC1CJ)

Issuing Module: CSVVFCRE - issued by ESTAE CSVVFCES

PLM: *System Logic Library*

Explanation: An abend (other than code 222, 322, or 522) occurred during virtual fetch initialization processing in CSVVFCRE. ESTAE routine CSVVFCES requests an SVC dump. Up to nine storage ranges are dumped and include:

- The VFCB pointed to by CVTVFCB.
- The VFCB obtained by this task and pointed to by VFCBPTR.
- The hash table, if a hash table was activated.
- The new hash table.
- The temporary table of PDS directory entries (INFOTAB).
- The ACA area.
- The VCBs area.
- The area used as a VIO window for reformatted modules.
- The area used as an input buffer for load module TXT records.

Problem Determination: A software record is written to SYS1.LOGREC. Register 1 contains the address of the SDWA. The following fields in the SDWA are filled in: SDWAMODN, SDWACSCT, SDWAREXN, SDWASC, SDWAMLVL, SDWARRL, and SDWACID.

Field CVTVFCB points to the VFCB. Field VFCBASCB points to the ASCB of the address space that owns the VFCB. CSVVFCRE sets VFCBASCB early in its processing. The VFCB also indicates whether CSVVFCRE was doing an initial build (VFCBECB = 0) or a refresh (VFCBECB ≠ 0).

In CSECT CSVVFCR1, local variable IOERTEXT contains a text description of the most recent I/O error detected by BSAM while reading modules. It is filled in by BSAM whenever the SYNAD exit is entered for the DCB named DCBLIBS.

In CSECT CSVVFCRE, field USERPRMS contains processing footprints that indicate the resources owned and the stage of processing at the time of the dump. USERPRMS is mapped by the structure FOOTPRTS in CSVVFCRE. Field RETRY in CSVVFCRE indicates whether CSVVFCES was retrying (RETRY = ON) or percolating the error. The intended retry address is in field RETRYADR in CSVVFCRE.

In the case when the ESTAE routine is percolating the error, no clean up has been performed and the footprints will describe the stage of processing at the time of the error.

COMPON = PROGRAM-MANAGER-VIRTUAL-FETCH,COMPID = SC1CJ,
ISSUER = CSVVFCFR-CSVVFCRE

Component: Contents Supervisor (5752-SC1CJ)

Issuing Module CSVVFCRE - issued by FRR CSVVFCFR

PLM: *System Logic Library*

Explanation: During refresh processing of the virtual fetch service address space, an abend occurred in CSVVFCES (CSVVFCRE's ESTAE routine) while it was attempting to release the logical group number (LGN) of the new VIO data set obtained by CSVVFCRE. Refresh processing failed after

the new LGN was obtained, and before the VFCB had been updated for the new generation. CSVVFCFR requests an SVC dump only when flag SDWACLUP is on, or when the FRR was reentered after attempting a retry.

Up to nine storage areas are dumped and include:

- The VFCB pointed to by CVTVFCB.
- The VFCB obtained by this task and pointed to by VFCBPTR.
- The old hash table.
- The new hash table.
- The temporary table of PDS directory entries (INFOTAB).
- The ACA area.
- The VCBs area.
- The area used as a VIO window for reformatted modules.
- The area used as an input buffer for load module TXT records.

Problem Determination: A software record is written to SYS1.LOGREC. Register 1 contains the address of the SDWA. The following fields in the SDWA are filled in: SDWAMODN, SDWACSCT, SDWAREXN, SDWASC, SDWAMLVL, SDWARRL, and SDWACID.

In CSECT CSVVFCR1, local variable IOERTEXT contains a text description of the most recent I/O error detected by BSAM while reading modules. It is filled in by BSAM whenever the SYNAD exit is entered for the DCB named DCBLIBS.

The VRA area in the SDWA contains a copy of variable USERPRMS as it was on entry to CSVVFCFR. Variable USERPRMS (in CSVVFCRE) contains processing footprints that indicate the resources owned and the stage of processing at the time of the dump. USERPRMS is mapped by the structure FOOTPRTS in CSVVFCRE. If this is a recursion, flag FPFRRCUR in USERPRMS is set on.

COMPON=PROGRAM-MANAGER-VIRTUAL-FETCH,COMPID=SC1CJ,
ISSUER=CSVVFGTE-CSVVFGES, CONTROL BLOCK FAILURE

Component: Contents Supervisor (5752-SC1CJ)

Issuing Module: CSVVFGTE - ESTAE

PLM: *System Logic Library*

Explanation: An abend occurred while CSVVFGTE (get processing) was searching a queue of VFWKs. ESTAE routine CSVVFGES in CSVVFGTE requests an SVC dump.

The areas dumped are LWK, VFPM, VFCB (if it exists), and LSQA (which contains the ASXB, VFVT, VFWKs, and job pack queue).

Problem Determination: A software record is written to SYS1.LOGREC. The following fields in the SDWA are filled in: SDWAMODN, SDWACSCT, SDWAREXN, SDWAMDAT, SDWAMVRS, SDWACID, and SDWARRL.

Field ASXBVFVT points to the VFVT. VFVTHASH is the start of a 31-way hash table, which contains pointers to the VFWK collision queues. The VFWKs are chained in single-threaded queues using field VFWKSYNP as the chain pointer.

If the error occurred while searching a queue of VFWKs, a probable cause might be that a chain pointer in the VFWK was overwritten with invalid data. In this case, flag VFVTVFUP is set off after requesting a SVC dump to prevent virtual fetch from being used again in this address space.

If the error occurred while freeing module and VCBs storage, the VFWK might have been overwritten causing CSVVFRM to pass invalid addresses to FREEMAIN. In this case, flag VFWKBADV is set on after requesting an SVC dump to prevent the VFWK from being used again.

COMPON=PROGRAM-MANAGER-VIRTUAL-FETCH,COMPID=SC1CJ,
ISSUER=CSVVFGTE-CSVVFRR, CONTROL BLOCK FAILURE

Component: Contents Supervisor (5752-SC1CJ)

Issuing Module: CSVVFGTE - FRR

PLM: *System Logic Library*

Explanation: An abend occurred while CSVVFGES (ESTAE for CSVVFGTE, get processing) was attempting to free storage (by calling CSVVFRM) related to a VFWK. FRR routine CSVVFRR in CSVVFGTE requests an SVC dump.

The areas dumped are LWK, VFPM, VFCB (if it exists), and LSQA (which contains the ASXB, VFVT, VFWKs, and job pack queue).

Problem Determination: A software record is written to SYS1.LOGREC. The following fields in the SDWA are filled in: SDWAMODN, SDWACSCT, SDWAREXN, SDWAMDAT, SDWAMVRS, SDWACID, and SDWARRL.

Field ASXBVFVT points to the VFVT. VFVTHASH is the start of a 31-way hash table, which contains pointers to the VFWK collision queues. The VFWKs are chained in single-threaded queues using field VFWKSYNP as the chain pointer.

If the error occurred while freeing module and VCBs storage, the VFWK might have been overwritten causing CSVVFRM to pass invalid addresses to FREEMAIN. In this case, flag VFWKBADV is set on after requesting an SVC dump to prevent the VFWK from being used again.

COMPON=PROGRAM-MANAGER-VIRTUAL-FETCH,COMPID=SC1CJ,
ISSUER=CSVVFNDE-CSVVFFES, CONTROL BLOCK FAILURE

Component: Contents Supervisor (5752-SC1CJ)

Issuing Module: CSVVFNDE - ESTAE

PLM: *System Logic Library*

Explanation: An abend occurred while CSVVFNDE (build or find processing) was searching a queue of VFWKs. ESTAE routine CSVVFFES in CSVVFNDE requests an SVC dump.

The areas dumped are LWK, VFPM, VFCB (if it exists), and LSQA (which contains the ASXB, VFVT, VFWKs, and job pack queue).

Problem Determination: A software record is written to SYS1.LOGREC. The following fields in the SDWA are filled in: SDWAMODN, SDWACSCT, SDWAREXN, SDWAMDAT, SDWAMVRS, SDWACID, and SDWARRL.

Field ASXBVFVT points to the VFVT. VFVTHASH is the start of a 31-way hash table, which contains pointers to the VFWK collision queues. The VFWKs are chained in single-threaded queues using field VFWKSYNP as the chain pointer.

If the error occurred while searching a queue of VFWKs, a probable cause might be that a chain pointer in the VFWK was overwritten with invalid data. In this case, flag VFVTVFUP is set off after requesting a SVC dump to prevent virtual fetch from being used again in this address space.

COMPON=PROGRAM-MANAGER-VIRTUAL-FETCH,COMPID=SC1CJ, ISSUER=CSVVFGTE-CSVVFRR, JPQ FAILURE

Component: Contents Supervisor (5752-SC1CJ)

Issuing Module: CSVVFGTE - FRR

PLM: *System Logic Library*

Explanation: During get processing, an abend occurred while the ESTAE routine for CSVVFGTE was attempting to remove a CDE from the job pack queue. FRR routine CSVVFRR in CSVVFGTE requests an SVC dump. Field LWKVFWK of the local work area (LWK) points to the VFWK containing the CDE at the time of the error.

The areas dumped are LWK, VFPM, VFCB, and LSQA (which contains the ASXB, VFVT, VFWKs, and job pack queue).

Problem Determination: A software record is written to SYS1.LOGREC. The following fields in the SDWA are filled in: SDWAMODN, SDWACSCT, SDWAREXN, SDWAMDAT, SDWAMVRS, SDWACID, and SDWARRL.

COMPON=PROGRAM-MANAGER-VIRTUAL-FETCH,COMPID=SC1CJ, ISSUER=CSVVFNDE-CSVVFFES, JPQ FAILURE

Component: Contents Supervisor (5752-SC1CJ)

Issuing Module: CSVVFNDE - ESTAE

PLM: *System Logic Library*

Explanation: During build or find processing, an abend occurred while CSVVFNDE was attempting to remove a CDE from the job pack queue for a module that previously abended under the current TCB. ESTAE routine CSVVFFES in CSVVFNDE requests an SVC dump. Register 8 points to the VFWK containing the CDE at the time of the error.

The areas dumped are LWK, VFPM, VFCB, and LSQA (which contains the ASXB, VFVT, VFWKs, and job pack queue).

Problem Determination: A software record is written to SYS1.LOGREC. The following fields in the SDWA are filled in: SDWAMODN, SDWACSCT, SDWAREXN, SDWAMDAT, SDWAMVRS, SDWACID, and SDWARRL.

COMPON=PROGRAM-MANAGER-VIRTUAL-FETCH,COMPID=SC1CJ, ISSUER=CSVVFGTE-CSVVFGES, RETRY INHIBITED

Component: Contents Supervisor (5752-SC1CJ)

Issuing Module: CSVVFGTE - ESTAE

PLM: *System Logic Library*

Explanation: Recovery routine CSVVFGES attempted to retry, but was reentered before retry processing was established. Retry is not attempted again. CSVVFGES sets flag LWKRECUR on in the LWK.

The areas dumped are LWK, VFPM, VFCB (if it exists), and LSQA (which contains the ASXB, VFVT, VFWKs, and job pack queue).

Problem Determination: A software record is written to SYS1.LOGREC. The following fields are filled in: SDWAMODN, SDWACSCT, SDWAREXN, SDWAMDAT, SDWAMVRS, SDWACID, and SDWARRL.

COMPON=PROGRAM-MANAGER-VIRTUAL-FETCH,COMPID=SC1CJ, ISSUER=CSVVFGTE-CSVVFRR, RETRY INHIBITED

Component: Contents Supervisor (5752-SC1CJ)

Issuing Modules: CSVVFGTE - FRR

PLM: *System Logic Library*

Explanation: The FRR CSVVFRR (in CSVVFGTE) was entered with flag SDWACLUP on and the abend was not a cancel abend (X'x22').

The areas dumped are LWK, VFPM, VFCB (if it exists), and LSQA (which contains the ASXB, VFVT, VFWKs, and job pack queue).

Problem Determination: A software record is written to SYS1.LOGREC. The following fields are filled in: SDWAMODN, SDWACSCT, SDWAREXN, SDWAMDAT, SDWAMVRS, SDWACID, and SDWARRL.

COMPON=PROGRAM-MANAGER-VIRTUAL-FETCH,COMPID=SC1CJ,
ISSUER=CSVVFNDE-CSVVFFES, RETRY INHIBITED

Component: Contents Supervisor (5752-SC1CJ)

Issuing Module: CSVVFNDE - ESTAE

PLM: *System Logic Library*

Explanation: Recovery routine CSVVFFES attempted to retry, but was reentered before retry processing was established. Retry is not attempted again. CSVVFFES sets flag LWKRECUR on in the LWK.

The areas dumped are LWK, VFPM, VFCB (if it exists), and LSQA (which contains the ASXB, VFVT, VFWKs, and job pack queue).

Problem Determination: A software record is written to SYS1.LOGREC. The following fields are filled in: SDWAMODN, SDWACSCT, SDWAREXN, SDWAMDAT, SDWAMVRS, SDWACID, and SDWARRL.

COMPON=REAL STORAGE MANAGEMENT,COMPID=SC1CR,
ISSUER=IARRRCV

Component: RSM (5752-SC1CR)

Issuing Module: IARRRCV

PLM: *System Logic Library*

Explanation: An abend has occurred during RSM processing. The areas dumped are SQA, LSQA, and TRT. A SUMDUMP is taken of the following:

- PFT and RPBs
- RSM's processor-related work/save area that was active when the error occurred
- RAB for the currently addressable address space
- RAB for the common area
- RAB for the address space being swapped (if applicable)
- RSM trace table
- SGT and PGTs for the currently addressable address space
- SGT and PGTs for the common area
- SFT for the currently addressable address space (if one exists)
- SFT for the address space being swapped (if applicable)

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS
SDWACSCT - CSECT in error
SDWAREXN - IARRR (recovery routine CSECT)
SDWARRL - IARRRCV (recovery routine label)
SDWAVRA - RSM's recovery termination area (RCA)

The RCA contains CSECT identifiers, footprint bits, and lockword addresses.

COMPON=RECONFIGURATION- DISPLAY M,COMPID=SC1CZ

Component: Reconfiguration (5752-SC1CZ)

Issuing Module: IEEMPDM

PLM: *System Logic Library*

Explanation: An abend has occurred during DISPLAY M processing. The main work area of the command processor is dumped.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IEEMPDM (module in error)
SDWACSCT - IEEMPDM (CSECT in error)
SDWAREXN - ESTAERTN (recovery routine)

COMPON=RECONFIG,COMPID=SC1CZ,CONFIG CPU ONLINE/OFFLINE (IEEVCPR),ABEND=xxx,ISSUER=IEEVCPR(ESTAECPR)

Component: Reconfiguration (CONFIG CPU processor)

Issuing Module: IEEVCPR

PLM: *System Logic Library*

Explanation: An error (ABEND=xxx) has occurred during CONFIG CPU processing. The areas dumped are PSA, SQA, TRT, LPA, LSQA, and the dynamic area for module IEEVCPR.

Problem Determination: The SDWAVRA contains:

- Label of the last retry point passed in IEEVCPR (See note)
- Reason code for the ABEND (REG15CDE)
- Caller's input to IEEVCPR (INPARMS)
- IEEVCPR's work area (WORKAREA)
- IEEVCPR's save area (SAVEAR)
- IEEVCPR's ESTAE area (ESTAEPRM)

Note: IEEVCPR has 21 labels that are used for returns after an ABEND. As each retry point is passed, the label name is saved so you can determine the section of code that was in control when the error occurred.

COMPON=RECONFIG,COMPID=SC1CZ,ISSUER=IEEVCHPF

Component: Reconfiguration (5752-SC1CZ)

Issuing Module: IEEVCHPF

PLM: *System Logic Library*

Explanation: An abend has occurred during reconfiguration processing of a force channel path offline request. The areas dumped are the FRR tracking area, the main work area for module IEEVCHPF, and, if there is a work area, the parameters passed to the MSSF.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS (load module in error)
SDWACSCT - IEEVCHPF (CSECT in error)
SDWAREXN - IEEVCHPF (CSECT containing recovery routine)
SDWARRL - FRRVCHPF (recovery routine)

COMPON=RECONFIG,COMPID=SC1CZ,ISSUER=IEEVIOSD

Component: Reconfiguration (5752-SC1CZ)

Issuing Module: IEEVIOSD

PLM: *System Logic Library*

Explanation: An abend has occurred during I/O processing. The areas dumped are the FRR tracking area, the pointer to the main work area for module IEEVCHPF, and, if there is a work area, the parameters passed to or received from the MSSF.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - NUCLEUS (load module in error)
SDWACSCT - IEEVIOSD (CSECT in error)
SDWAREXN - IEEVIOSD (CSECT containing recovery routine)
SDWARRL - VIOSDFRR (recovery routine)

COMPON=RECONFIG,COMPID=SC1CZ,ISSUER=IEEVSTEE

Component: Reconfiguration (5752-SC1CZ)

Issuing Module: IEEVSTEE - ESTAE

PLM: *System Logic Library*

Explanation: An abend has occurred during CONFIG STOR reconfiguration processing for a storage element request. The error occurred in module IEEVSTEL (storage element reconfiguration) or module IEEVSTFA (storage element alternate reconfiguration). The areas dumped are the MSSF data (for an offline request, both offline command INFO and OFFLINE command data are included; for an online request, only the

ONLINE command data is included), the storage address increment (SAI) array, NUC, LSQA, SQA, TRT, and PSA.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IEEMPVST (load module in error)
SDWACSCT - IEEVSTEL or IEEVSTFA (CSECT in error)
SDWAREXN - IEEVSTEE (CSECT containing recovery routine)

COMPON=RECONFIG,COMPID=SC1CZ,ISSUER=IEEVSTPE,IEEVSTGP FAILED

Component: Reconfiguration (5752-SC1CZ)

Issuing Module: IEEVSTPE - ESTAE

PLM: *System Logic Library*

Explanation: An abend has occurred during reconfiguration processing of a CONFIG STOR physical request in module IEEVSTGP. The areas dumped are the MSSF data, the storage address increment (SAI) array, NUC, LSQA, SQA, TRT, and PSA.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IEEMPVST (load module in error)
SDWACSCT - IEEVSTGP (CSECT in error)
SDWAREXN - IEEVSTPE (CSECT containing recovery routine)

COMPON=RECONFIG(SC1CZ),MODULE=IEEVPTH (VARY PATH) FAILED,ABEND(xxx)

Component: Reconfiguration (5752-SC1CZ)

Issuing Module: IEEVPTH

PLM: *System Logic Library*

Explanation: An abend has occurred during VARY PATH command processing. The areas dumped are the command image buffer (CHBUF), the current VARY request block (if any), and the main work area of module IEEVPTH.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IEEVPTH (load module in error)
SDWACSCT - IEEVPTH (CSECT in error)
SDWAREXN - IEEVPTH (CSECT containing recovery routine)
SDWARRL - ESTAPTH (recovery routine)

COMPON=RECONFIG(SC1CZ),MODULE=IEEVPTHR
FAILED,ABEND(xxx)

Component: Reconfiguration (5752-SC1CZ)

Issuing Module: IEEVPTHR

PLM: *System Logic Library*

Explanation: An abend has occurred during VARY PATH reconfiguration processing. The areas dumped are the main work area for module IEEVPTHR, the first request block in the chain passed to IEEVPTHR, the current request block (if any) that represents the path being processed, and, if there is a current request block, the device number and the channel path identifier for the path.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IEEVPTHR (load module in error)
SDWACSCT - IEEVPTHR (CSECT in error)
SDWAREXN - IEEVPTHR (CSECT containing recovery routine)
SDWARRL - VPTHESTA (recovery routine)

COMPON=RMF,COMPID=27404,ISSUER=ERBCNFGC,
I/O CONFIG.TAB. CREATE

Component: RMF (5665-27404)

Issuing Module: ERBCNFGC

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An abend occurred while the RMF Monitor I I/O configuration table create module (ERBCNFGC) was processing. ERBCNFGC is called by ERBMFMFC during RMF initialization. The ESTAE recovery routine CNFGABND issues the SDUMP macro. The areas dumped are LSQA, SWA, TRT, PSA, and SUMDUMP. The LIST option specifies the STGST, IOCHT, and IODNT.

Problem Determination: A software record is written to SYS1.LOGREC. The RMF control block STGST (pointed to by field CVTMFCTL in the CVT) and the trace table are helpful in determining the cause of the error. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=RMF,COMPID=27404,ISSUER=ERBCNFGF,
I/O CONFIG.TAB. BUILD

Component: RMF (5665-27404)

Issuing Module: ERBCNFGF

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An abend occurred while the RMF Monitor I I/O configuration table build module (ERBCNFGF) was processing. ERBCNFGF is called by ERBMFMFC during RMF initialization. The ESTAE recovery routine CNFGABND issues the SDUMP macro. The areas dumped are LSQA, SWA, TRT, PSA, and SUMDUMP. The LIST option specifies the STGST, IOCHT, IODNT, and LCUT.

Problem Determination: A software record is written to SYS1.LOGREC. The RMF control block STGST (pointed to by field CVTMFCTL in the CVT) and the trace table are helpful in determining the cause of the error. The SDWAVRA contains module trace information and pointers to the STGST, IOCHT, IODNT, and LCUT. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=RMF,COMPID=27404,ISSUER=ERBCNFGG,
I/O CONFIG.TAB. CREATE

Component: RMF (5665-27404)

Issuing Module: ERBCNFGG

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An abend occurred while the RMF Monitor I I/O configuration table build for 4381 processors (module ERBCNFGG) was processing. ERBCNFGG is called by ERBMFMFC during RMF initialization. The internal ESTAE recovery routine CNFGGESA issues the SDUMP macro. The areas dumped are LSQA, SWA, TRT, PSA, and SUMDUMP. The LIST option specifies the STGST, IOCHT, and IODNT.

Problem Determination: A software record is written to SYS1.LOGREC. The RMF control block STGST (pointed to by field CVTMFCTL in the CVT) and the trace table are helpful in determining the cause of the error. The SDWAVRA contains module trace information and pointers to the STGST, IOCHT, IODNT, LCUT, HSARB, SCHIB, and IOSB. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=RMF,COMPID=27404,ISSUER=ERBMFDEA,
RMF MON.I CONTROL

Component: RMF (5665-27404)

Issuing Module: ERBMFDEA - ESTAE

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An error has occurred during RMF processing. The data control ESTAE routine ERBMFDEA issues the SDUMP macro. The areas dumped are LSQA, SWA, TRT, PSA, and SUMDUMP. The LIST option specifies the STGST, IOCHT, STMMV, RMCT, CMCT, CPMT, ICHPT, RCE, RMPT, CMB, and ICSC. The entries in the RMF storage resource table (STSGT) are also specified depending on whether there is sufficient space in the LIST pool.

Problem Determination: A software record is written to SYS1.LOGREC. The RMF control block STGST (pointed to by field CVTMFCTL in the CVT) and the trace table are helpful in determining the cause of the error. The SDWAVRA contains module trace information and the problem control table (ERBMFPCT). The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=RMF,COMPID=27404,ISSUER=ERBMFEAR,
RMF LISTEN EXITS

Component: RMF (5665-27404)

Issuing Module: ERBMFEAR

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An abend occurred while the RMF Monitor I event arrival routine (ERBMFEAR) was processing. ERBMFEAR receives control when a change occurs for device state, reconfiguration (DDR) activity, CMB data state, channel facility recovery, and channel path state. The internal ESTAE recovery routine ERBLXERV issues the SDUMP macro. The areas dumped are SQA, LSQA, TRT, PSA, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC. The RMF control block STGST (pointed to by field CVTMFCTL in the CVT) and the trace table are helpful in determining the cause of the error. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=RMF,COMPID=27404,ISSUER=ERBMFEVT,
RMF MON.I SAMPLER

Component: RMF (5665-27404)

Issuing Module: ERBMFEVT

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An abend occurred while the RMF Monitor I MFROUTER service module (ERBMFEVT) was processing. ERBMFEVT receives control as a timer DIE from the timer second level interruption handler. Control is passed consecutively to the list of event measurement gathering routines associated with the MFROUTER. The internal FRR recovery routine EVFRR recovers from errors occurring in the MFROUTER service module or in any of the RMF samplers. Routine EVSFRR issues the SDUMP macro. The areas dumped are SQA, CSA, TRT, PSA, RGN, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC. The RMF control block STGST (pointed to by field CVTMFCTL in the CVT) and the trace table are helpful in determining the cause of the error. The SDWAVRA contains module trace information, the FRR parameter area, STMMV entry, and lock names. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=RMF,COMPID=27404,ISSUER=ERBMFFUR,
RMF MON.I CONTROL

Component: RMF (5665-27404)

Issuing Module: ERBMFFUR

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An error has occurred during RMF processing. The FRR lock release failure recovery routine ERBMFFUR issues the SDUMP macro. The areas dumped are SQA, TRT, PSA, RGN, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC. The RMF control block STGST (pointed to by field CVTMFCTL in the CVT) and the trace table are helpful in determining the cause of the error. The SDWAVRA contains module trace information, the address of the failing routine, timer queue element, and RMF TQE from the timer supervisor work area. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=RMF,COMPID=27404,ISSUER=ERBMFIDX,
RMF MSCH COMPLETION

Component: RMF (5665-27404)

Issuing Module: ERBMFIDX

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An abend occurred while the asynchronous MSCH (modify subchannel) completion module (ERBMFIDX) was processing. ERBMFIDX is scheduled as an SRB routine upon completion of an asynchronous MSCH request. The internal FRR recovery routine ERBMFIDF issues the SDUMP macro. The areas dumped are SQA, LSQA, TRT, PSA, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC. The RMF control block STGST (pointed to by field CVTMFCTL in the CVT) and the trace table are helpful in determining the cause of the error. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=RMF,COMPID=27404,ISSUER=ERBMFIQA,
RMF I/O QUEUING

Component: RMF (5665-27404)

Issuing Module: ERBMFIQA

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An abend occurred while the start/stop hardware measurements for I/O queuing for 4381 processors (ERBMFIQA) was processing. The internal ESTAE recovery routine ERBIQERV issues the SDUMP macro. The areas dumped are SQA, LSQA, TRT, PSA, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC. The RMF control block STGST (pointed to by field CVTMFCTL in the CVT) and the trace table are helpful in determining the cause of the error. The SDWAVRA contains module trace information and pointers to the STGST, IOCHT, IODNT, LCUT, and HSARB. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=RMF,COMPID=27404,ISSUER=ERBMFMFC,
RMF SESSION CONTROL

Component: RMF (5665-27404)

Issuing Module: ERBMFMFC

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An abend occurred while the measurement facility control module (ERBMFMFC) was processing. The internal ESTAE recovery routine ABNDEXIT issues the SDUMP macro. The areas dumped are LSQA, SWA, TRT, PSA, and SUMDUMP. The LIST option specifies the STGST, GSTC3, IOCHT, and IODNT.

Problem Determination: A software record is written to SYS1.LOGREC. The RMF control block STGST (pointed to by field CVTMFCTL in the CVT) and the trace table are helpful in determining the cause of the error. The SDWAVRA contains module trace information, the ACT control block, and ESTAE parameter area. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=RMF,COMPID=27404,ISSUER=ERBMFMLN,
ERROR RMF MON I INIT

Component: RMF (5665-27404)

Issuing Module: ERBMFMLN

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An error has occurred during RMF processing. ERBMFMLN, the ESTAE for ERBMFIZZ, receives control after any error that occurs after issuing the MFSTART SVC. ERBMFMLN is the highest level ESTAE error recovery routine for the RMF Monitor I session. The areas dumped are LSQA, SWA, TRT, PSA, and SUMDUMP. The LIST option specifies the STGST and IOCHT.

Problem Determination: A software record is written to SYS1.LOGREC. The RMF control block STGST (pointed to by field CVTMFCTL in the CVT) and the trace table are helpful in determining the cause of the error. The SDWAVRA contains module trace information, the PCT control block, session name, and ESTAE parameter area. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=RMF,COMPID=27404,ISSUER=ERBMFPVS,
RMF VSTOR PVT SAMPLER

Component: RMF (5665-27404)

Issuing Module: ERBMFPVS

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An abend occurred while the virtual storage private area sampling module (ERBMFPVS) was processing. ERBMFPVS receives control from ERBMFEVS via an SRB schedule at the end of each cycle. The internal FRR recovery routine PVSFRR issues the SDUMP macro. The areas dumped are TRT, PSA, RGN, and SUMDUMP. The SUMLIST option specifies the EDTVS, virtual storage private data tables, and the SRB.

Problem Determination: A software record is written to SYS1.LOGREC. The RMF control block STGST (pointed to by field CVTMFCTL in the CVT) and the trace table are helpful in determining the cause of the error. The SDWAVRA contains module trace information, the FRR parameter area, pointers to the EDTVS, and current job sampler block. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=RMF,COMPID=27404,ISSUER=ERBMFRES,
MEMTERM RESOURCE MANAGER

Component: RMF (5665-27404)

Issuing Module: ERBMFRES

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An abend occurred while the RMF memory termination resource manager (ERBMFRES) was processing. The internal ESTAE recovery routine RESESTAE issues the SDUMP macro. The areas dumped are RGN, SQA, TRT, PSA, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC. The RMF control block STGST (pointed to by field CVTMFCTL in the CVT) and the trace table are helpful in determining the cause of the error. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=RMF,COMPID=27404,ISSUER=ERBMFSDE,
RMF MON.I CONTROL

Component: RMF (5665-27404)

Issuing Module: ERBMFSDE - ESTAE

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An error has occurred during RMF processing. The MFSTART ESTAE routine ERBMFSDE issues the SDUMP macro. The areas dumped are LSQA, SWA, TRT, PSA, and SUMDUMP. The LIST option specifies the STGST, IOCHT, STMMV, RMCT, CMCT, CPMT, ICHPT, RCE, RMPT, CMB, and ICSC. The entries in the RMF storage resource table (STSGT) are also specified depending on whether there is sufficient space in the LIST pool.

Problem Determination: A software record is written to SYS1.LOGREC. The RMF control block STGST (pointed to by field CVTMFCTL in the CVT) and the trace table are helpful in determining the cause of the error. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=RMF,COMPID=27404,ISSUER=ERBMFTMA,
RMF MON.I TERMINATION

Component: RMF (5665-27404)

Issuing Module: ERBMFTMA

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An abend occurred while the RMF termination mainline module (ERBMFTMA) was processing. ERBMFTMA receives control from either ERBMFSDE (abnormal termination) or IGX00007 (normal termination). The internal ESTAE recovery routine ERBMFTXR issues the SDUMP macro. The areas dumped are LSQA, SWA, TRT, PSA, and SUMDUMP. The LIST option specifies the STGST, IOCHT, STMMV, RMCT, CMCT, CPMT, ICHPT, RCE, RMPT, CMB, and ICSC. The entries in the RMF storage resource table (STSGT) are also specified depending on whether there is sufficient space in the LIST pool.

Problem Determination: A software record is written to SYS1.LOGREC. The RMF control block STGST (pointed to by field CVTMFCTL in the CVT) and the trace table are helpful in determining the cause of the error. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=RMF,COMPID=27404,ISSUER=ERBMFTRM,
RMF MON.I TERMINATION

Component: RMF (5665-27404)

Issuing Module: ERBMFTRM

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An abend occurred while the RMF general resource release module (ERBMFTRM) was processing. ERBMFTRM receives control from ERBMFTMA. The internal ESTAE recovery routine ERBMFTGR issues the SDUMP macro. The areas dumped are SQA, LSQA, SWA, TRT, PSA, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC. The RMF control block STGST (pointed to by field CVTMFCTL in the CVT) and the trace table are helpful in determining the cause of the error. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=RMF,COMPID=27404,ISSUER=ERB3GEEH,
RMF ENQ EVENT HANDLER

Component: RMF (5665-27404)

Issuing Module: ERB3GEEH

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An abend occurred while the Monitor III data gatherer enqueue event handler module (ERB3GEEH) was processing. ERB3GEEH receives control from ERB3GLUE. ERB3GLUE is invoked when enqueue contention in the system changes. The internal FRR recovery routine GEEHFRR issues the SDUMP macro. The areas dumped are TRT and SUMDUMP. The SUMLIST option specifies the ERB3GEEH module work area and the enqueue event table entries.

Problem Determination: A software record is written to SYS1.LOGREC. The RMF control block STGST (pointed to by field CVTMFCTL in the CVT) and the trace table are helpful in determining the cause of the error. The SDWAVRA contains module trace information and pointers to the STGST and GSTC3. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=RMF,COMPID=27404,ISSUER=ERB3GESA,
and,
MONIII GATHERER CANCEL FAILING CSECT NAME ccccccc
or,
MONIII GATH RECURSION FAILING CSECT NAME ccccccc
or,
FAILURE MONIII GATHERER FAILING CSECT NAME ccccccc

Component: RMF (5665-27404)

Issuing Module: ERB3GESA - ESTAE

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An error has occurred during RMF Monitor III data gathering. ccccccc is an 8-character CSECT name. The MONITOR III gatherer ESTAE routine ERB3GESA issues the SDUMP macro. The areas dumped are LSQA, SWA, TRT, PSA, SQA, and SUMDUMP. The LIST option specifies the STGST, GSTC3, and WSHG3.

Problem Determination: A software record is written to SYS1.LOGREC. The SDWA contains the module slot of the failing module, the current stack entry of the RETG3, and pointers to the STGST, GSTC3, GGDG3, WSHG3, and RETG3. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=RMF,COMPID=27404,ISSUER=ERB3GXMV,
TSO RMFWDM | sid SESSION

Component: RMF (5665-27404)

Issuing Module: ERB3GXMV - ESTAE

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An abend occurred while the RMF Monitor III gatherer cross memory move module (ERB3GXMV) was processing. A TSO session or local session (where sid is the session-id) was active.

The SDUMP macro is issued by ERB3GXFR (1) when requested by a MON III reporter module, or (2) from the internal FRR recovery routine itself. If the dump is requested by a reporter module, a SUMDUMP, all local areas, and the wrap-around buffers are dumped. If the dump is issued from the recovery routine, a SUMDUMP and all local areas except the wrap-around buffers are dumped.

Problem Determination: If the dump is requested by a reporter module, SDWA and VRA information is not available; the wrap-around buffer area contains the set of samples that caused the problem in the reporter module. If the dump is issued from the FRR routine, a software record is written to SYS1.LOGREC. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=RMF,COMPID=27404,ISSUER=ERB3RMFC,
M3 LOCAL SESSION INIT

Component: RMF (5665-27404)

Issuing Module: ERB3RMFC

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An abend occurred while the Monitor III reporter local session initialization module (ERB3RMFC) was processing. ERB3RMFC receives control from ERB3CREP. The internal ESTAE recovery routine RMFCABND issues the SDUMP macro. The areas dumped are RGN, TRT, PSA, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC. The RMF control block STGST (pointed to by field CVTMFCTL in the CVT) and the trace table are helpful in determining the cause of the error. The SDWAVRA contains module trace information and pointers to the STGST and GSTC3. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=RMF-ENQ EVENT HANDLER,COMPID=27404,
ISSUER=ERBMFEEQ

Component: RMF (5665-27404)

Issuing Module: ERBMFEEQ

PLM: *Resource Measurement Facility (RMF) Version 3 Program Logic Manual*

Explanation: An abend occurred while the RMF Monitor I ENQ event handler (ERBMFEEQ) was processing. ERBMFEEQ receives control when an increase or decrease in enqueue contention occurs. Recovery routine ERBMFFRQ issues the SDUMP macro. The areas dumped are TRT and SUMDUMP. The SUMLIST option specifies the ERBMFEEQ module work area and the ENQ data collection area (ERBEQEDT and ERBEQRES).

Problem Determination: A software record is written to SYS1.LOGREC. The RMF control block STGST (pointed to by field CVTMFCTL in the CVT) and the trace table are helpful in determining the cause of the error. The SDWAVRA contains module trace information and pointers to the module work area and ERBEQEDT. The failing CSECT name and the error condition can be determined from RTM2WA and SDWA.

COMPON=SAM,COMPID=27405,ISSUER=AMSACT,ERROR IN SAM TERMINATION EXIT

Component: RMF SAM (5665-27405)

Issuing Module: AMSACT

PLM: None

Explanation: The AMSCOL collector module was tracking an application program that terminated. While processing the application program's termination, the AMSACT module abnormally terminated.

Problem Determination: A software record is written to SYS1.LOGREC. The failing CSECT name and the error condition can be determined from the RTM2WA and SDWA. If you are not able to determine the cause of the problem from the dump provided, follow your installation's procedures for contacting IBM for service.

COMPON=SAM,COMPID=27405,ISSUER=AMSACT,ERROR IN SAM USER AMSACU EXIT

Component: RMF SAM (5665-27405)

Issuing Module: AMSACT

PLM: None

Explanation: The AMSCOL collector module was tracking an application program that terminated. While processing the application program's termination, the AMSACT module called an AMSACU user exit. During execution of AMSACU, an abnormal termination occurred that was not covered by a user ESTAE routine.

Problem Determination: A software record is written to SYS1.LOGREC. The failing CSECT name and the error condition can be determined from the RTM2WA and SDWA. If you are not able to determine the cause of the problem from the dump provided, you might want to code an ESTAE exit for AMSACU to capture the error.

COMPON=SAM,COMPID=27405,ISSUER=AMSCOL,ABEND

Component: RMF SAM (5665-27405)

Issuing Module: AMSCOL

PLM: None

Explanation: The AMSCOL collector module (or one of its subtasks) abnormally terminated. AMSACT automatically restarts the collector for the first occurrence of the ABEND.

Problem Determination: The cause of the ABEND might be obvious from the abend code. If not, follow your installation's procedures for contacting IBM for service.

COMPON=SAM,COMPID=27405,ISSUER=AMSCOL,AMSCFREE
OVERLAID - RECOVERED

Component: RMF SAM (5665-27405)

Issuing Module: AMSCOL

PLM: None

Explanation: The AMSCOL collector module detected that the AMSCFREE pointer in the AMSCNTL control block (in the ECSA) was overlaid with some other data. AMSCOL corrects the pointer's value and continues processing.

Problem Determination: Because the SDUMP was taken before AMSCOL corrected the data, the overlaying data appears in the dump. Therefore, by examining the dump data, you might be able to determine the program that caused the overlay.

COMPON=SAM,COMPID=27405,ISSUER=AMSCOL,AMSCNTL HEADER
OVERLAID - RECOVERED

Component: RMF SAM (5665-27405)

Issuing Module: AMSCOL

PLM: None

Explanation: The AMSCOL collector module detected that the header information for its AMSCNTL control block (in the ECSA) was overlaid with some other data. AMSCOL corrects the header information and continues processing.

Problem Determination: Because the SDUMP was taken before AMSCOL corrected the data, the overlaying data appears in the dump. Therefore, by examining the dump data, you might be able to determine the program that caused the overlay.

COMPON=SAM,COMPID=27405,ISSUER=AMSCOL,AMSCPREV OVERLAID - RECOVERED

Component: RMF SAM (5665-27405)

Issuing Module: AMSCOL

PLM: None

Explanation: The AMSCOL collector module detected that the AMSCPREV pointer in the AMSCNTL control block (in the ECSA) was overlaid with some other data. AMSCOL corrects the pointer's value and continues processing.

Problem Determination: Because the SDUMP was taken before AMSCOL corrected the data, the overlaying data appears in the dump. Therefore, by examining the dump data, you might be able to determine the program that caused the overlay.

COMPON=SAM,COMPID=27405,ISSUER=AMSCOL,BAD ADDRESS IN AMSCNTL - RECOVERED

Component: RMF SAM (5665-27405)

Issuing Module: AMSCOL

PLM: None

Explanation: The AMSCOL collector module was posted by either AMSUJI or AMSACT, indicating that there was data to be passed. However, the pointer in the AMSCNTL control block (in the ECSA) did not point to a valid AMSP data block. AMSCOL ignores the data and continues processing.

Problem Determination: The problem could be due to one or more of the following conditions:

- An overlay of the pointer to the AMSP data block
- An overlay of the AMSP data block
- An internal error in AMSUJI, AMSACT, or AMSCOL

If an overlay occurred, by examining the data, you might be able to determine the program that caused the overlay.

COMPON=SAM,COMPID=27405,ISSUER=AMSCOL,POINTER OVERLAID IN AMSCNTL RECOVERED

Component: RMF SAM (5665-27405)

Issuing Module: AMSCOL

PLM: None

Explanation: The AMSCOL collector module was posted by either AMSUJI or AMSACT, indicating that there was data to be passed. However, the AMSCPREV pointer in the AMSCNTL control block did not point to a valid field. AMSCOL corrects the pointer's value and continues processing, but no data is passed.

Problem Determination: Because the SDUMP was taken before AMSCOL corrected the data, the overlaying data appears in the dump. The overlay of data could have been caused by an internal error within AMSUJI, AMSACT, or AMSCOL, or by another program overlaying the correct data. By examining the dump data, you might be able to determine the program that caused the overlay.

COMPON=SAM,COMPID=27405,ISSUER=AMSCOL,WDS RECORD MISMATCH - RECOVERED

Component: RMF SAM (5665-27405)

Issuing Module: AMSCOL

PLM: None

Explanation: The AMSCOL collector module was tracking an application program that terminated. When the AMSDISK subtask attempted to update the work data set (WDS), it found that the WDS record did not match the record in storage.

Problem Determination: The WDS cannot be shared between systems. If it was not being shared, it is most probable that an internal error occurred in AMSCOL. Follow your installation's procedures for contacting IBM for service.

COMPON=SAM,COMPID=27405,ISSUER=AMSUJI,ERROR IN SAM INITIATION EXIT

Component: RMF SAM (5665-27405)

Issuing Module: AMSUJI

PLM: None

Explanation: An error occurred in the SAM job initiation module.

Problem Determination: A software record is written to SYS1.LOGREC. The failing CSECT name and the error condition can be determined from the RTM2WA and SDWA. If you are not able to determine the cause of the problem from the dump provided, follow your installation's procedures for contacting IBM for service.

COMPON=SAM,COMPID=27405,ISSUER=AMSUJI,ERROR IN SAM USER AMSUJU EXIT

Component: RMF SAM (5665-27405)

Issuing Module: AMSUJI

PLM: None

Explanation: An application program was initiated and control passed from AMSUJI to the AMSUJU user exit. During execution of AMSACU, an abnormal termination occurred that was not covered by a user ESTAE routine.

Problem Determination: A software record is written to SYS1.LOGREC. The failing CSECT name and the error condition can be determined from the RTM2WA and SDWA. If you are not able to determine the cause of the problem from the dump provided, you might want to code an ESTAE exit for AMSUJU to capture the error.

COMPON=SC,COMPID=SC1C5,ISSUER=IEAVEGR,TRQ GLOBAL RECOVERY -- UNEXPECTED ERROR

Component: Supervisor Control (5752-SC1C5)

Issuing Module: IEAVEGR

PLM: *System Logic Library*

Explanation: An error occurred while IEAVEGR was processing (repairing the true ready queue). As a result, IEAVEGR's FRR gains control and issues the SDUMP macro. The areas dumped are TRT, SUMDUMP, and a SUMLIST that includes the module work area, the SVT (with its true ready queue counts and pointer), the actual true ready queue header, and the pseudo-header.

Problem Determination: You can determine the caller of IEAVEGR from the register save area in the SCWA (SCWEGRSV) or from the module work area.

COMPON=SC,ISSUER=IEAVEGR,TRQ GLOBAL RECOVERY INVOKED BY xxxxxxxx -- NORMAL PROCESSING

Component: Supervisor Control (5752-SC1C5)

Issuing Module: IEAVEGR

PLM: *System Logic Library*

Explanation: An error in the true ready queue was detected by xxxxxxxx, which then called IEAVEGR to repair the queue. IEAVEGR issues the SDUMP macro before it repairs the queue. The areas dumped are TRT and SUM. The SUMLIST option includes the ASCBs that are found to be in error.

IEAVEGR uses the queue verifier (IEAVEQV0) to validate both the swapped-in queue and the true ready queue. The dump will contain any information that the queue verifier gathers pertaining to invalid ASCBs.

Problem Determination: IEAVEGR may or may not be called in a recovery environment. If called by IEAVEDSR, which is the dispatcher recovery routine, a software record is written to SYS1.LOGREC Otherwise, no software record is written to correspond to the SDUMP.

COMPON=SDUMP,COMPID=SC1CM,ISSUER=IEAVTSEP, FAILURE IN POST DUMP EXIT PROCESSOR

Component: Dumping Services (5752-SC1CM)

Issuing Module: IEAVTSEP

PLM: *System Logic Library*

Explanation: An error has occurred while processing post dump exits in the DUMPSRV address space. The areas dumped are SUM, TRT, LSQA, CSA, NOSQA, and subpools 231 and 0.

Problem Determination: Print the summary dump and CURRENT. The SDWAVRA contains the following:

- The ESTAE parameter area
- The list of post dump exits
- Field DSVEXPRC of the DSVCB

COMPON=SETSMF COMMAND, ISSUER=IEECB913, COMPID=SC100

Component: SMF (5752-SC100)

Issuing Module: IEECB913

PLM: *System Logic Library*

Explanation: An abend has occurred during SETSMF command processing. The areas dumped are PSA, NUC, RGN, LPA, TRT, SQA, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=SET SMF COMMAND, ISSUER=IEEMB835, COMPID=SC100

Component: SMF (5752-SC100)

Issuing Module: IEEMB835 - ESTAE

PLM: *System Logic Library*

Explanation: An abend has occurred during SET SMF command processing. The areas dumped are PSA, NUC, RGN, SQA, LPA, TRT, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=SETSMF COMMAND, ISSUER=IFASSMF, COMPID=SC100

Component: SMF (5752-SC100)

Issuing Module: IFASSMF

PLM: *System Logic Library*

Explanation: An abend has occurred during SETSMF command processing. The areas dumped are PSA, NUC, RGN, LPA, TRT, SQA, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - Load module in error
SDWACSCT - CSECT in error
SDWACID - Component ID
SDWASC - Failing function name
SDWARRL - ESTAE routine label
SDWAREXN - ESTAE module name

COMPON=SET SMF COMMAND, ISSUER=IFATSMF, COMPID=SC100

Component: SMF (5752-SC100)

Issuing Module: IFATSMF

PLM: *System Logic Library*

Explanation: An abend has occurred during SET SMF command processing. The areas dumped are PSA, NUC, RGN, LPA, TRT, SQA, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - Load module in error
SDWACSCT - CSECT in error
SDWACID - Component ID
SDWASC - Failing function name
SDWARRL - ESTAE routine label
SDWAREXN - ESTAE module name

COMPON=SMF CONTROL TASK, ISSUER=IEEMB825,
ERRORMOD=IFASMF/IEFU29, COMPID=SC100

Component: SMF (5752-SC100)

Issuing Module: IEEMB825

PLM: *System Logic Library*

Explanation: An error has occurred during SMF control task (IFASMF) processing, or some other function issued a CALLRTM against IFASMF with a 253 abend code. If ERRORMOD=IEFU29, the user exit was in control at the time of the error. The areas dumped are PSA, NUC, RGN, SQA, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - Load module in error
SDWACSCT - CSECT in error
SDWACID - Component ID
SDWASC - Failing function name
SDWAREXN - ESTAE module name

For a description of abend code 253, refer to *System Codes.*

COMPON=SMF INITIALIZATION, ISSUER=IEEMB827, COMPID=SC100

Component: SMF (5752-SC100)

Issuing Module: IEEMB827

PLM: *System Logic Library*

Explanation: An error has occurred during SMF address space initialization. The areas dumped are PSA, NUC, RGN, SQA, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - Load module in error
SDWACSCT - CSECT in error
SDWACID - Component ID
SDWASC - Failing function name
SDWAREXN - ESTAE module name

COMPON=STC-REC,COMPID=SC1B8,ISSUER=IEESB665,
STARTED TASK CONTROL RECOVERY EXIT ROUTINE

Component: Master Scheduler Commands (5752-SC1B8)

Issuing Module: IEESB665

PLM: *System Logic Library*

Explanation: The recovery exit routine IEESB665 schedules a retry for STC in the event of an error (if information is available for a retry). If an SDWA is provided, IEESB665 issues the SDUMP macro. The areas dumped are SQA, PSA, LSQA, RGN, LPA, TRT, CSA, and NUC.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=SUPERVISOR CONTROL,COMPID=SC1C5,
ISSUER=IEAVESAR,UNEXPECTED ERROR

Component: Supervisor Control (5752-SC1C5)

Issuing Module: IEAVESAR - Supervisor Analysis Router

PLM: *System Logic Library*

Explanation: An error has occurred during processing by the supervisor analysis router (IEAVESAR) or one of the analysis routines called by the router. The areas dumped are SUM, PSA, SQA, and NUC.

Problem Determination: A software record is written to SYS1.LOGREC. The SDWAVRA contains a copy of the FRR parameter area, which includes:

- The caller of the supervisor analysis router
- The routine in control at the time of the error

Refer to label FRRPRM in module IEAVESAR for a detailed description of the FRR parameter area.

COMPON=SUPV CNTL,COMPID=SC1C5,FUNCTION=RESUME,
MODULE=IEAVETCL,ISSUER=IEAVETCL(IEAVETCR)

Component: Supervisor Control (5752-SC1C5)

Issuing Module: IEAVETCL

PLM: *System Logic Library*

Explanation: An error has occurred during RESUME or TCTL processing. Recovery routine IEAVETCR issues the SDUMP macro.

Problem Determination: A software record is written to SYS1.LOGREC. The SDWAVRA contains diagnostic information indicating the function in control at the time of the error, and debugging data related to that function.

COMPON=SVC34,COMPID=SC1B8,ISSUER=IEE5103D,
FAILURE IN SVC34/COMMAND xxxx

Component: Master Scheduler Commands (5752-SC1B8)

Issuing Module: IEE5103D - STAE

PLM: *System Logic Library*

Explanation: The SVC 34 STAE routine IEE5103D issues SDUMP when (1) a system error or program check has occurred, or (2) the system restart key was pressed. The areas dumped are PSA, NUC, LSQA, RGN, LPA, TRT, CSA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=SYMREC,COMPID=SCASR,ISSUER=ASRSERVR,
LOGIC ERROR IN SYMREC SERVICE

Component: Symptom Record (5752-SCASR)

Issuing Module: ASRSERVR - FRR entry point in ASRSERVP

PLM: *System Logic Library*

Explanation: An abend has occurred during the processing of a symptom record request. The FRR routine ASRSERVR issues the SDUMP macro. The areas dumped are SUMDUMP and SUMLIST.

Problem Determination: An SDWA software record is written to SYS1.LOGREC, with the SDWAVRA in key-length-data format. A text description (in key-length-data format) precedes each data area included in the VRA. The VRA contains a required dump analysis and elimination (DAE) symptom identified by key X'E1'. The data associated with this key is the one-byte hexadecimal footprint, which indicates where the error occurred in ASRSERVP. The footprint is an index into a table that defines the symbolic name of the footprint. The cross-reference listing in module ASRSERVP indicates where the symbolic name is used.

The SUMLIST data is the input symptom record and the dynamic area or workarea allocated for symptom record processing. A text description precedes the dumped SUMLIST data.

COMPON=SYSLOG-ESTAE,COMPID=SC1B8,ISSUER=IEEMB806,
SYSTEM LOG ESTAE PROCESSOR

Component: Master Scheduler Commands (5752-SC1B8)

Issuing Module: IEEMB806 - ESTAE

PLM: *System Logic Library*

Explanation: An abend has occurred during system log task processing. The areas dumped are PSA, NUC, LSQA, SQA, and subpool 231.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=SYSLOG-INIT,COMPID=SC1B8,ISSUER=IEEMB803,
SYSTEM LOG INITIALIZATION

Component: Master Scheduler Commands (5752-SC1B8)

Issuing Module: IEEMB803

PLM: *System Logic Library*

Explanation: An error has occurred during IEEMB803 (system log initialization/writer) processing. The areas dumped are PSA, NUC, LSQA, and subpool 231.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=SYSLOG-SVC,COMPID=SC1B8,ISSUER=IEEMB804
SYSTEM LOG SVC DUMP

Component: Master Scheduler Commands (5752-SC1B8)

Issuing Module: IEEMB804

PLM: *System Logic Library*

Explanation: The data of a non-authorized user does not exist in his own storage area. The user has failed the authorization check and FETCH/PROTECT validity test, or an abend occurred during processing. Procedure DUMPRTN issues the SDUMP macro. The areas dumped are LSQA, and subpool 231.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=SYSTEM TRACE - A.S. CREATE,COMPID=SC142,
ISSUER=IEAVETAC

Component: System Trace (5752-SC142)

Issuing Module: IEAVETAC

PLM: *System Logic Library*

Explanation: An error has occurred during IEAVETAC processing while creating the TRACE address space. Routine ETACRECV issues the SDUMP macro. The areas dumped are SUM, ALLPSA, SQA, LSQA, NUC, TRT, and GRSQ.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=SYSTEM TRACE - A.S. INIT,COMPID=SC142,
ISSUER=IEAVETAI

Component: System Trace (5752-SC142)

Issuing Module: IEAVETAI

PLM: *System Logic Library*

Explanation: An error has occurred during IEAVETAC processing while initializing the TRACE address space. Routine ETAICRECV issues the SDUMP macro. The areas dumped are SUM, ALLPSA, SQA, LSQA, NUC, TRT, and GRSQ.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=SYSTEM TRACE-FORMATTER,COMPID=SC142,
ISSUER=IEAVETFC

Component: System Trace (5752-SC142)

Issuing Module: IEAVETFC

PLM: *System Logic Library*

Explanation: An error has occurred during IEAVETAC processing while formatting the system trace table for a SNAP request. Module IEAVETFC issues the SDUMP macro. The areas dumped are:

- The trace table snapshot copy header (TTCH) that is being formatted
- The dynamic work area of module IEAVETFC that contains the TFWA and the BY-TIME and DEVICES tables
- SUMDUMP, TRT, and LSQA

Problem Determination: A software record is written to SYS1.LOGREC. The SDWA contains the following:

- The address of the caller of IEAVETFC.
- The address and length of the TFWA.
- The TFWAFP footprint field that contains flags and execution trace footprints designed to help screen duplicate problems.
- The significant portion of the BY-TIME table. The entries in this table indicate where in the data for each processor the formatter is.
- SDWAMODN - Module name in control.
- SDWAMLVL - CSECT level in control.
- SDWAREXN - IEAVETFC recovery routine name.

- SDWARRL - ETFCRECV recovery routine label.
- SDWACID - SC142 component ID.
- SDWASC - SYSTEM TRACE FORMATTER subcomponent.

COMPON=SYSTEM TRACE - xxxxxxxxxx,COMPID=SC142
ISSUER=IEAVETRR

Component: System Trace (5752-SC142)

Issuing Module: IEAVETRR

PLM: *System Logic Library*

Explanation: An error has occurred during IEAVETRR processing while performing a system trace service. Field xxxxxxxxxx in the title indicates one of the following services that was in control:

- ALTRTRC
- SUSPEND/R/P
- SNAPTRC
- COPYTRC
- ASIDTRC
- VERFYTRC

Module IEAVETRR issues the SDUMP macro. If the SNAPTRC service was in control, the areas dumped are ALLPSA, SQA, NOSUMDUMP, and LSQA for the home, primary and secondary address spaces at the time of the error. If any other service was in control, the areas dumped are ALLPSA, SQA, SUMDUMP, TRT, and LSQA for the home, primary and secondary address spaces at the time of the error.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

- SDWACID - SC142 component ID.
- SDWASC - SYSTEM TRACE - xxxxxxxxxx subcomponent.
- SDWAMODN - Module name in control.
- SDWACSCT - CSECT name in control.
- SDWAMLVL - CSECT level in control.
- SDWAREXN - IEAVETRR recovery routine module name.
- SDWARRL - IEAVETFR recovery routine label name.

The following areas, if available, are included in the SDWAVRA:

- FRR Parameter area - see TRFP for the mapping.
- Module footprint word - see the mapping of TRRVMFPA in the particular module.
- Return address of the invoker.

- Variable module data - see the mapping of TRRVRCDM in the particular module.

COMPON=TASK MGMT,COMPID=SC1CL,ISSUER=IEAVEPST,
POST EXIT ACTIVE

Component: Task Management (5752-SC1CL)

Issuing Module: IEAVEPST

PLM: *System Logic Library*

Explanation: An error has occurred while a POST exit ECB was being posted. The areas dumped are TRT, SQA, LSQA, and ALLNUC.

Problem Determination: A software record is written to SYS1.LOGREC. The SDWAVRA includes POST's local save area (LSA).

COMPON=TASK MGMT,COMPID=SC1CL,ISSUER=IEAVEPST,
POST FAILED -- UNEXPECTED ERROR

Component: Task Management (5752-SC1CL)

Issuing Module: IEAVEPST

PLM: *System Logic Library*

Explanation: An error has occurred while a POST request was processing. The areas dumped are TRT, SQA, LSQA, and ALLNUC.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=TASK MGMT,COMPID=SC1CL,ISSUER=IEAVEPST,
XMPOST FAILED -- NO ERRET

Component: Task Management (5752-SC1CL)

Issuing Module: IEAVEPST

PLM: *System Logic Library*

Explanation: An error has occurred while a cross memory POST was processing. The ERRET routine was not invoked because the originating address space has terminated. The areas dumped are TRT, SQA, LSQA, and ALLNUC.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=TASK MGMT,COMPID=SC1CL,ISSUER=IEAVEPST,
XMPOST FAILED, UNABLE TO RETRY

Component: Task Management (5752-SC1CL)

Issuing Module: IEAVEPST

PLM: *System Logic Library*

Explanation: An error has occurred while a cross memory POST was processing. The ERRET routine was not invoked because the address of the SRB was zero. The areas dumped are TRT, SQA, LSQA, and ALLNUC.

Problem Determination: A software record is written to SYS1.LOGREC.

COMPON=TASK MGMT,COMPID=SC1CL,ISSUER=IEAVEWAT,
WAIT FAILED -- UNEXPECTED ERROR

Component: Task Management (5752-SC1CL)

Issuing Module: IEAVEWAT

PLM: *System Logic Library*

Explanation: An error has occurred while an SVC WAIT request was processing. The areas dumped are TRT, SQA, LSQA, and ALLNUC.

Problem Determination: A software record is written to SYS1.LOGREC. The SDWAVRA includes the values passed to WAIT in registers 0 and 1.

COMPON=VSM,COMPID=SC1CH,ISSUER=IGVGCAS,ABEND=xxx

Component: VSM (5752-SC1CH)

Issuing Module: IGVGCAS - FRR

PLM: *System Logic Library*

Explanation: Abend xxx has occurred during memory create processing in IGVGCAS. The areas dumped are SQA, PSA, SUMDUMP, LSQA, ALLNUC, and TRT.

Problem Determination: A software record is written to SYS1.LOGREC with the SDWAVRA written in key-length-data format. VSM uses keys 16 and 200 to record information in the VRA. Refer to *System Logic Library* for an explanation of these keys.

COMPON=VSM,COMPID=SC1CH,ISSUER=IGVGRRGN,ABEND=xxx

Component: VSM (5752-SC1CH)

Issuing Module: IGVGRRGN - ESTAE

PLM: *System Logic Library*

Explanation: Abend xxx has occurred during get real region processing. The areas dumped are SQA, PSA, SUMDUMP, and LSQA.

Problem Determination: A software record is written to SYS1.LOGREC with the SDWAVRA written in key-length-data format. VSM uses key 16 to record information in the VRA. Refer to *System Logic Library* for an explanation of this key.

COMPON=VSM,COMPID=SC1CH,ISSUER=IGVGVRGN,ABEND=xxx

Component: VSM (5752-SC1CH)

Issuing Module: IGVGVRGN - ESTAE

PLM: *System Logic Library*

Explanation: Abend xxx has occurred during get virtual region processing. The areas dumped are SQA, PSA, SUMDUMP, and LSQA.

Problem Determination: A software record is written to SYS1.LOGREC with the SDWAVRA written in key-length-data format. VSM uses key 16 to record information in the VRA. Refer to *System Logic Library* for an explanation of this key.

COMPON=VSM,COMPID=SC1CH,ISSUER=IGVRVSM

Component: VSM (5752-SC1CH)

Issuing Module: IGVRVSM - FRR

PLM: *System Logic Library*

Explanation: An error has occurred during GETMAIN or FREEMAIN processing. The abend code can be found in field SDWACMPC. While attempting to recover from this error, module IGVRVSM encountered an uncorrectable error in one of VSM's major control blocks (such as VSWK or GDA). Module IGVRVSM forces percolation of the abend.

Problem Determination: A software record is written to SYS1.LOGREC with the SDWAVRA written in key-length-data format. VSM uses keys 16, 206, 211, 215, 216, 218, 219, 222, and 223 to record information in the VRA. Refer to *System Logic Library* for an explanation of these keys.

COMPON=VSM,COMPID=SC1CH,ISSUER=IGVQSPET,ABEND=xxx

Component: VSM (5752-SC1CH)

Issuing Module: IGVSTSKT - FRR

PLM: *System Logic Library*

Explanation: Abend xxx has occurred during task termination processing in IGVSTSKT. The areas dumped are SQA, PSA, SUMDUMP, and LSQA.

Problem Determination: A software record is written to SYS1.LOGREC with the SDWAVRA written in key-length-data format. VSM uses keys 16, 200, 201, and 202 to record information in the VRA. Refer to *System Logic Library* for an explanation of these keys.

COMPON=VSM,COMPID=SC1CH,ISSUER=IGVSTSKI,ABEND=xxx

Component: VSM (5752-SC1CH)

Issuing Module: IGVSTSKI - FRR

PLM: *System Logic Library*

Explanation: Abend xxx has occurred during attach processing in IGVSTSKI. The areas dumped are SQA, PSA, SUMDUMP, LSQA, ALLNUC, and TRT.

Problem Determination: A software record is written to SYS1.LOGREC with the SDWAVRA written in key-length-data format. VSM uses keys 16 and 33 to record information in the VRA. Refer to *System Logic Library* for an explanation of these keys.

COMPON=VSM-CELLPOOL BUILD|DELETE|EXTEND|RECOVERY,
COMPID=SC1CH,ISSUER=IGVRCP,ABEND=xxx

Component: VSM (5752-SC1CH)

Issuing Module: IGVRCP - FRR

PLM: *System Logic Library*

Explanation: Abend xxx has occurred during CPOOL processing. The areas dumped are SQA, PSA, SUMDUMP, LSQA, ALLNUC, and TRT.

Problem Determination: A software record is written to SYS1.LOGREC with the SDWAVRA written in key-length-data format. VSM uses keys 16, 17, 18, 32, and 33 to record information in the VRA. Refer to *System Logic Library* for an explanation of these keys.

COMPON=VSM-GETMAIN|FREEMAIN,COMPID=SC1CH,
ISSUER=IGVSRTN,ABEND=xxx

Component: VSM (5752-SC1CH)

Issuing Module: IGVRSRTN - FRR

PLM: *System Logic Library*

Explanation: Abend xxx has occurred during GETMAIN or FREEMAIN processing. The areas dumped are SQA, PSA, LSQA, ALLNUC, TRT, and SUMDUMP. The areas dumped using the LIST option are the VSM work area (VSWK), the global cell pools, the global data area (GDA), the VSM table module (IGVSTBL), the address space control block (ASCB), the local data area (LDA), and the task control block (TCB).

Problem Determination: A software record is written to SYS1.LOGREC with the SDWAVRA written in key-length-data format. VSM uses keys 16, and 200 through 235 to record information in the VRA. Refer to *System Logic Library* for an explanation of these keys.

COMPON=VSM-IGVFVIRT,COMPID=SC1CH,
ISSUER=IGVFVIRT,ABEND=xxx

Component: VSM (5752-SC1CH)

Issuing Module: IGVFVIRT - FRR

PLM: *System Logic Library*

Explanation: Abend xxx has occurred during CSA deferred release processing in IGVFVIRT. The areas dumped are PSA, SUMDUMP, ALLNUC, TRT, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC with the SDWAVRA written in key-length-data format. VSM uses keys 16, 215, and 218 to record information in the VRA. Refer to *System Logic Library* for an explanation of these keys.

COMPON=VSM-VSMLIST,COMPID=SC1CH,
ISSUER=IGVSLIST,ABEND=xxx

Component: VSM (5752-SC1CH)

Issuing Module: IGVSLIST - FRR

PLM: *System Logic Library*

Explanation: Abend xxx has occurred during VSMLIST processing. The areas dumped are PSA, SUMDUMP, ALLNUC, TRT, SQA, LSQA, and the caller's work area.

Problem Determination: A software record is written to SYS1.LOGREC with the SDWAVRA written in key-length-data format. VSM uses keys 16 and 40 to record information in the VRA. Refer to *System Logic Library* for an explanation of these keys.

COMPON=VSM-VSMLOC,COMPID=SC1CH,
ISSUER=IGVLOCP,ABEND=xxx

Component: VSM (5752-SC1CH)

Issuing Module: IGVLOCP - FRR

PLM: *System Logic Library*

Explanation: Abend xxx has occurred during VSMLOC processing. The areas dumped are PSA, SUMDUMP, ALLNUC, TRT, SQA, and LSQA.

Problem Determination: A software record is written to SYS1.LOGREC with the SDWAVRA written in key-length-data format. VSM uses key 16 to record information in the VRA. Refer to *System Logic Library* for an explanation of this key.

DUMP BY/(OF) MODULE xxxxxxxx

Component: GTF (5752-SC111)

Issuing Module: AHLWTO

PLM: *Service Aids Logic*

Explanation: Entry point AHLDMPMD in AHLWTO provides a dumping service for the GTF FGBRs (filter, gather, and build routines). xxxxxxxx indicates the FGBR affected: AHLTSLIP, AHLTSYSM, AHLTUSR, AHLTSIO, AHLTSVC, AHLTPID, AHLTSYFL, AHLTEXT, AHLTFOR, or AHLTXSYS. The GTF control blocks dumped are MCHEAD, MCRWSA, MCAWSA, MCCE, MCQE, and GTFPCT. The SQA, SDWA, and the failing FGBR module are also dumped.

Problem Determination: The error is probably a page fault that occurred when the FGBR referenced a data area that should be fixed but was not. Message AHL118I is issued. For additional information, refer to message AHL118I in *System Messages*. For most errors, the GTF FGBR also writes a software record to SYS1.LOGREC.

DUMP OF AHLREADR

Component: GTF (5752-SC111)

Issuing Module: AHLREADR

PLM: *Service Aids Logic*

Explanation: An error has occurred while AHLREADR was attempting to pass GTF buffers to SDUMP or SNAP for inclusion in an outstanding dump request. The dump taken by AHLREADR includes a dump of itself plus a dump of the failing address space. The AHLREAD macro request is cleaned up, which includes posting the original requestor, releasing locks, dequeueing on the MC (monitor call) control blocks, and releasing allocated storage.

Problem Determination: A software record is written to SYS1.LOGREC.

DUMP OF GTF MODULE AHLSBLOK

Component: GTF (5752-SC111)

Issuing Module: AHLSBUF

PLM: *Service Aids Logic*

Explanation: An error has occurred while moving the GTF global trace buffer to a page in the GTF address space. The failing address space is dumped. The error is percolated to the FRR for the active data gathering routine. The FRR in the router routine (AHLMCER) disables and terminates GTF.

Problem Determination: A software record is written to SYS1.LOGREC.

DUMP OF GTF MODULE AHLWTASK

Component: GTF (5752-SC111)

Issuing Module: AHLWTASK

PLM: *Service Aids Logic*

Explanation: An error has occurred when either (1) entry point AHLWPOST in AHLWTASK was attempting to post AHLWTASK in order to schedule message AHL118I, or (2) entry point AHLWTASK was attempting to schedule an SRB for a WTO of message AHL118I. The areas dumped are the SDUMP buffer, failing module, and failing address space.

Problem Determination: Message AHL119I is issued. The SDUMP buffer contains message AHL118I (which would have been issued if the error had not occurred), the SRB that did not complete, and the SDWA.

DUMP OF JES2 CHECKPOINT DATA. SYSTEM=id, $ERROR CODE=code

Component: JES2 (5752-SC1BH)

Issuing Module: HASPCKPT

PLM: *JES2 Logic*

Explanation: JES2 has detected a major error during I/O processing to the checkpoint data set. Fields in the dump title are:

id - system ID on which the error was detected
code - JES2 abend code

The JES2 actual checkpoint master record, job queue, and JOT storage are dumped.

Problem Determination: For additional information on JES2 error codes, refer to message HASP095 in *System Codes*. For additional information on the action to be taken, refer to *SPL: JES2*.

ENF ABEND ERRORMOD=IEFENFFX

Component: Scheduler Services (5752-BB131)

Issuing Module: IEFENFFX

PLM: *System Logic Library*

Explanation: An abend has occurred while IEFENFFX (ENF request router routine) was processing an event notification request. The areas dumped are NUC and SQA.

Problem Determination: A software record is written to SYS1.LOGREC. The SDWAVRA contains the ESTAE or FRR parameter list and footprint bits that indicate the execution path of IEFENFFX.

ENF ABEND ERRORMOD=IEFENFNM

Component: Scheduler Services (5752-BB131)

Issuing Module: IEFENFNM

PLM: *System Logic Library*

Explanation: An abend has occurred while IEFENFNM (ENF mainline routine) was processing an event notification request. The areas dumped are NUC, RGN, CSA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC. The SDWAVRA contains the ESTAE or FRR parameter list and footprint bits that indicate the execution path of IEFENFNM.

ENQUEUE FAILURE IN ABDUMP,COMPON=ABDUMP,
COMPID=SC1CM,ISSUER=IEAVTABD

Component: RTM - ABDUMP Processing (5752-SC1CM)

Issuing Module: IEAVTABD

PLM: *System Logic Library*

Explanation: An error has occurred when ABDUMP attempted to enqueue (ENQ) on the dump data set in order to process a dump request specified on a user's SYSABEND, SYSMDUMP, or SYSUDUMP DD statement. The areas dumped are LPA, LSQA, SQA, TRT, GRSQ, and subpools 230 and 250.

Problem Determination: Because ABDUMP was unable to take the dump requested by the user, use the SVC dump information to identify the user's problem.

ERROR DURING RESTART PROCESSING - IEFXB609

Component: Scheduler Restart (5752-SC1B3)

Issuing Module: IEFXB609 - Data Set Descriptor Record Processor

PLM: *System Logic Library*

Explanation: An error has occurred during restart processing. The areas dumped are NUC, LSQA, RGN, LPA, TRT, SWA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC.

ERROR DURING SNAP,COMPON=SNAP,
COMPID=SC1CM,ISSUER=IEAVAD01

Component: RTM - Dump Services (5752-SC1CM)

Issuing Module: IEAVAD01 - ESTAE

PLM: *System Logic Library*

Explanation: An error has occurred during SNAP dump processing when SNAP was attempting to take a dump for the user. An I/O error or invalid control block field can cause this error. The areas dumped are LPA, SQA, TRT, GRSQ, and subpools 230 and 250.

Problem Determination: A software record is written to SYS1.LOGREC. It includes the failing CSECT name that identifies the formatter in control at the time of the error.

ERROR IN AHLSETEV

Component: GTF (5752-SC111)

Issuing Module: AHLSETEV

PLM: *Service Aids Logic*

Explanation: A program check has occurred when referencing the MC (monitor call) tables that are built during GTF initialization by the SETEVENT macro. GTF applications are terminated and acquired resources are freed. Message AHL132I is issued. The area dumped is SQA, which contains the MC tables.

Problem Determination: Validate the MC tables that are located in the SQA. For additional information, refer to message AHL132I in *System Messages.*

ERROR IN IATSIDMO FOR SYSOUT DATA SET

Component: JES3 (5752-SC1BA)

Issuing Module: IATDMFR - FRR

PLM: *JES3 Logic*

Explanation: An error has occurred while module IATSIDM (USAM subsystem interface routine) was attempting to open a SYSOUT data set. The FRR routine IATDMFR issues the SDUMP macro. IATDMFR returns to IATSIDM via the retry address (RETADDR parameter) on the SETRP macro. IATSIDM terminates the job with a 1FB system abend code. The areas dumped are SQA, CSA, and LPA.

Problem Determination: A software record is written to SYS1.LOGREC. For a description of the 1FB abend code, refer to *System Codes.*

ERROR IN IEAVTSLP

Component: RTM - SLIP Processor (5752-SC1CM)

Issuing Module: IEAVTSLR

PLM: *System Logic Library*

Explanation: An error has occurred during SLIP processing. The FRR in IEAVTSLR issues SDUMP. The areas dumped are ALLPSA, NUC, LSQA, and SQA. The summary part of the dump requested by IEAVTSLR contains information relevant to the error.

Problem Determination: A software record is written to SYS1.LOGREC. The FRR parameter list is put in the SDWAVRA.

ERROR IN INITIATOR,ABEND=,COMPON=INIT,
COMPID=SC1B6,ISSUER=IEFIB620

Component: Initiator (5752-SC1B6)

Issuing Module: IEFIB620 - ESTAE

PLM: *System Logic Library*

Explanation: During initiator processing, the ESTAE exit routine IEFIB620 issues SDUMP when (1) a system error or program check has occurred, or (2) the system restart key is pressed. The areas dumped are RGN, LPA, TRT, ALLPSA, SWA, LSQA, and ALLNUC.

Problem Determination: A software record is written to SYS1.LOGREC.

ERROR IN MASTER SUBSYSTEM BROADCAST FUNCTION,
ABEND=aaa,COMPON=INIT-SSI,COMPID=SC1B6,ISSUER=IEFJRASP

Component: Initiator - Subsystem Interface (5752-SC1B6)

Issuing Module: IEFJRASP

PLM: *System Logic Library*

Explanation: An abend has occurred while IEFJRASP was routing a subsystem interface request to all active subsystems, via the subsystem interface. The areas dumped are NUC, CSA, LPA, TRT, and LSQA.

Problem Determination: A software record is written to SYS1.LOGREC. The SDWAVRA contains footprint bits that indicate the execution path of IEFJRASP.

ERROR IN MODULE AHLMCER

Component: GTF (5752-SC111)

Issuing Module: AHLMCER

PLM: *Service Aids Logic*

Explanation: An error has occurred during GTF processing when AHLMCER attempted to route the MC (monitor call) interruption to its affiliated FGBR (filter, gather, and build routine). The FRR routine (AHLMCFRR) requests the dump prior to attempting retry. The MCRWSA and SDWA are moved into the SDUMP buffer. AHLMCER is included in the dump as part of the storage dumped. GTF is terminated. The areas dumped are SQA, SDUMP buffer, failing module, and failing address space.

Problem Determination: Message AHL007I is issued. A software record is written to SYS1.LOGREC. This error is usually an inability to pass control to an FGBR because of changes to the FGBR in SYS1.LPALIB. Field MCREID in the MCRWSA contains the event identifier of the HOOK that GTF was processing.

ERROR IN QMNGRIO PROCESSING,COMPON=SNAP,
COMPID=SC1CM,ISSUER=IEAVAD01

Component: RTM - Dump Services (5752-SC1CM)

Issuing Module: IEAVAD01 - ESTAE

PLM: *System Logic Library*

Explanation: An error has occurred during SNAP dump processing when the QMNGRIO macro attempted to read the JFCB in order to obtain an output line and the page capacity. The areas dumped are LPA, SWA, SQA, TRT, and subpools 250 and 253.

Problem Determination: The JFCB might be in error.

ERROR IN SUBSYSTEM SERVICE RTN, COMPON=INIT-SSI, COMPID=SC1B6,ISSUER=IEFJSBLD,ABEND=aaa

Component: Initiator - Subsystem Interface (5752-SC1B6)

Issuing Module: IEFJSBLD

PLM: *System Initialization Logic*

Explanation: An abend (aaa) has occurred while IEFJSBLD was either building an SSCVT, SSVT, SHAS, or SAST, or was preparing to link to a subsystem's initialization routine. The areas dumped are ALLPSA, LSQA, RGN, CSA, and TRT.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IEEVIPL (module in error)
SDWACSCT - IEFJSBLD (CSECT in error)
SDWAREXN - IEFJSBLD (recovery routine)

The SDWAVRA contains the input parameter list and footprint bits that indicate the execution path of IEFJSBLD.

ERROR IN SUBSYSTEM INITIALIZATION,COMPON=INIT-SSI, COMPID=SC1B6,ISSUER=IEFJSIN2,ABEND=aaa

Component: Initiator - Subsystem Interface (5752-SC1B6)

Issuing Module: IEFJSIN2

PLM: *System Initialization Logic*

Explanation: An abend (aaa) occurred during initialization processing of the subsystems. The error occurred in IEFJSIN2 or in service routines IEEMB878 or IEEMB882. The areas dumped are ALLPSA, LSQA, RGN, and TRT.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IEEMB860 (module in error)
SDWACSCT - IEFJSIN2 (CSECT in error)
SDWAREXN - IEFJSIN2 (recovery routine

The SDWAVRA contains footprint bits to indicate the execution path of IEFJSIN2.

EVENT NOTIFICATION FACILITY ERROR,ABEND=xxx,
COMPON=SCHR-ENF,COMPID=BB131,ISSUER=IEFENFWT

Component: Scheduler Services (5752-BB131)

Issuing Module: IEFENFWT

PLM: *System Logic Library*

Explanation: An abend has occurred while IEFENFWT (ENF wait routine) was processing. The areas dumped are NUC, CSA, SQA, and RGN.

Problem Determination: A software record is written to SYS1.LOGREC.

FAILURE DURING SNAP RECOVERY,COMPON=SNAP,
COMPID=SC1CM,ISSUER=IEAVAD01

Component: RTM - Dump Services (5752-SC1CM)

Issuing Module: IEAVAD01 - ESTAE

PLM: *System Logic Library*

Explanation: An error has occurred while the SNAP dump ESTAE routine was attempting to cleanup after an error had occurred during SNAP mainline processing. No further cleanup is attempted. The areas dumped are LPA, SQA, TRT, GRSQ, and subpools 250 and 253.

Problem Determination: The SNAP storage buffers are probably incorrect. Use the previous RTM2WA to identify the error that occurred during SNAP mainline processing. The SNAP mainline error might have affected this error.

FIOD:IDA019S2 - ABEND FROM FIOD FRR

Component: VSAM - Record Management (5665-28418)

Issuing Module: IDA019S2 - FRR

PLM: *VSAM Logic*

Explanation: An abnormal termination has occurred during VSAM record management processing. The FRR routine IDA019S2 (at entry point IDAF19S2), issues the SDUMP macro. The areas dumped are PSA, NUC, RGN, TRT, CSA, and SQA.

Problem Determination: A VSAM ICIP (improved control interval processing) request was executing in supervisor state or SRB mode and encountered a program check while the I/O manager was processing the request. Register 3 points to the IOMB for the request. For information on how the IOMB relates to the VSAM control block structure, refer to *VSAM Logic*.

GTF TERMINATING ON ERROR CONDITION

Component: GTF (5752-SC111)

Issuing Module: AHLTMON

PLM: *Service Aids Logic*

Explanation: An error has occurred during GTF initialization before the initialization was successfully completed. The retry routine AHLTERM2 (in AHLTMON) issues the SDUMP macro. GTF is terminated. The areas dumped are RGN, LPA, SQA, and MCHEAD control block.

Problem Determination: A software record is written to SYS1.LOGREC.

HASPDUMP SUBSYS=ssss vvvvvvvv MODULE=mmmmmmmm CODE=cccc

Component: JES2 (5752-SC1BH)

Issuing Module: HASPTERM or HASPRAS

PLM: *JES2 Logic*

Explanation: An error has occurred during JES2 processing. ssss is the subsystem identification, normally JES2 (ssss is obtained from the TIOT); vvvvvvvv is the JES2 version identification; mmmmmmmm is the name of the primary JES2 load module, normally HASJES20; cccc is either the system completion code, Shhh (such as S0C1) or JES2 catastrophic error code, $ccc (such as $K01).

Problem Determination: See message $HASP095 in *JES2 Messages* for an explanation of JES2 error codes, and *System Codes* for an explanation of system codes.

A software record is written to SYS1.LOGREC. Refer to the JES2 LGRR mapping macro in module HASPDOC for a description of SDWAVRA information.

IATSIJS JSESEXIT

Component: JES3 (5752-SC1BA)

Issuing Module: IATSIJS

PLM: *JES3 Logic*

Explanation: An abend has occurred during IATSIJS (job processing subsystem interface) processing. The ESTAE routine established by IATSIJS is given control to examine the function control table (FCT) active at the time of termination to determine which function or DSP has failed. The areas dumped are PSA, NUC, SQA, RGN, LPA, TRT, and CSA.

Problem Determination: A software record is written to SYS1.LOGREC.

IATSNLS - ESTAE EXIT

Component: JES3 (5752-SC1BA)

Issuing Module: IATSNLS

PLM: *JES3 Logic*

Explanation: A subtask was terminated because an abend has occurred in one of the following: (1) OPNDST processing, (2) CLSDST exit, (3) CLSDST error exit, (4) SETLOGON exit, (5) SIMLOGON exit, (6) LOGON IRB, (7) TPEND processing, (8) LOSTERM exit, (9) RESPONSE IRB exit, (10) DFSAY exit or (11) OPEN or CLOSE processing (in which case, no retry is attempted). IATSNLS issues the SDUMP macro. The areas dumped are SQA, ALLPSA, NUC, LSQA, RGN, LPA, TRT, and CSA.

Problem Determination: A software record is written to SYS1.LOGREC.

IATSSCM READ-END FAILURE

Component: JES3 (5752-SC1BA)

Issuing Module: IATSSCM

PLM: *JES3 Logic*

Explanation: An error has occurred during IATSSCM (subsystem communication scheduler) read-end processing. The areas dumped are PSA, NUC, RGN, LPA, TRT, CSA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC.

IAT1081 ERROR IN IATDMDKT - IATYISR POSSIBLY LOST

Component: JES3 (5752-SC1BA)

Issuing Module: IATDMFR - FRR

PLM: *JES3 Logic*

Explanation: A software or hardware error has occurred and caused the JES3 channel end termination routine (IATDMDKT) to abnormally terminate. The FRR routine IATDMFR was not able to recover from the error. Either the input/output service block (IOSB) or service request block (SRB) in IATYISR might be invalid. The areas dumped are SQA, LPA, and CSA.

Problem Determination: A software record is written to SYS1.LOGREC. Message IAT1801 is issued. For a description of message IAT1801, refer to *JES3 Messages.*

IAT3702 dspname (ddd) ABENDED/FAILED ABEND code/DMxxx - JES3 FAILURE NO.nnn

Component: JES3 (5752-SC1BA)

Issuing Module: IATABN0

PLM: *JES3 Logic*

Explanation: A DSP has abended or failed. The text in the SVC dump title identifies the failing DSP (dspname), and the unit address (ddd), if available. The system abend code (code) or the DM type (xxx) is given along with the unique JES3 failsoft identifier (nnn). Message IAT3702 is issued. IATABN0 (online format driver) issues the SDUMP macro. The areas dumped are PSA, NUC, SQA, LSQA, RGN, LPA, TRT, and CSA.

Problem Determination: For additional information: refer to abend codes in *System Codes*, DM codes in *JES3 Diagnosis*, and message IAT3702 in *JES3 Messages*.

IAT4830 IATIISB MASTERTASK ABEND

Component: JES3 (5752-SC1BA)

Issuing Module: IATIISB

PLM: *JES3 Logic*

Explanation: An abend has occurred during IATIISB (interpreter master subtask) processing. The areas dumped are NUC, PSA, RGN, LPA, TRT, and CSA.

Problem Determination: A software record is written to SYS1.LOGREC. Also check the SYSMSG data set for error indications.

IAT4831 IATIISB SUBTASK ABEND

Component: JES3 (5752-SC1BA)

Issuing Module: IATIISB (IATYICT work area)

PLM: *JES3 Logic*

Explanation: An abend has occurred while an interpreter subtask was processing. Message IAT4211 is issued. IATIISB issues the SDUMP macro. The areas dumped are SQA, PSA, NUC, RGN, LPA, TRT, and CSA.

Problem Determination: A software record is written to SYS1.LOGREC.

ICBRECRD RECORDING FAILED

Component: MSS - MSSC (5752-SC1DP)

Issuing Module: ICBRECRD - ESTAE

PLM: *Mass Storage System Communicator Logic*

Explanation: An abend has occurred in the 3850 message journaling function of MSS processing. The RESTAEXT routine in module ICBRECRD issues the SDUMP macro. The areas dumped are CSA, LSQA, and TRT.

Problem Determination: A software record is written to SYS1.LOGREC. Message ICB328E is sent to the system console indicating that 3850 message journaling is disabled due to a system failure. The MJX (message journaling exit control block), located in the CSA, indicates the function that ICBRECRD was performing at the time of error (such as, enabling the message journaling function, disabling the message journaling function, or writing a record to the message journaling data set). Also, the MJX contains the queue header for the messages to be written to the message journaling data set.

ICB425I ABEND IN PROCESS - MSVC TASK (nnnnnnnn)

Component: MSS - MSVC (5752-SC1DR)

Issuing Module: ICBVPR00 - ESTAE

PLM: *Mass Storage System Communicator Logic*

Explanation: An abend has occurred in the MSVC (mass storage volume control) function of MSS processing. The VPRSDUMP routine in module ICBVPR00 issues the SDUMP macro. The areas dumped are PSA, TRT, and CSA. Address ranges are included to dump the following control blocks: SDWA, MSVC control block (IEZVVICB), and the MSSC control block (IEZSSC).

Problem Determination: In the dump title, nnnnnnnn indicates the module in error, ICBVPR00 is a generalized recovery module, therefore, nnnnnnnn can be any module that is part of the MSVC function. Refer to *Mass Storage System Communicator Logic* for a complete list of MSVC modules. If 'UNKNOWN' appears in the title, ICBVPR00 was unable to determine the module in error. Message ICB425I is sent to the system operator. This message indicates the failing module and contains the MSS order that was being performed at the time of the error. When 'UNKNOWN' appears in the message and SDUMP title, the MSS order can be used to determine which MSVC module was involved in the error. Refer to *Mass Storage System Communicator Logic* for a description of the MSVC modules and their relationship to MSS orders.

ICHRST00 - RACF SVCS, ABEND CODE=xxx, SVC=svcname, USER=user, GROUP=group, EXIT=installation exit name

Component: RACF - SVC Processing (5752-XXH00)

Issuing Module: ICHRST00 - ESTAE

PLM: *Resource Access Control Facility (RACF): Program Logic Manual*

Explanation: An abend has occurred during processing of one of the RACF SVCs (ICHRIN00, ICHRCK00, or ICHRDF00). The executing task is terminated. The areas dumped are PSA, RGN, LPA, TRT, CSA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - main CSECT name of SVC
SDWACSCT - blanks or installation exit name
SDWAREXN - ICHRST00 (recovery routine)
SDWAGR15 - reason code if the abend is a RACF abend
SDWACRC - completion code
SDWACID - XXH00
SDWAEAS - 1 if SDUMP is taken by ICHRST00
SDWAREQ - 0 if SDUMP is taken by ICHRST00

ICTMCS01, CRYPTOGRAPHY INITIALIZATION

Component: Programmed Cryptographic Facility (5752-XY500)

Issuing Module: ICTMCS01 - ESTAE

PLM: *Programmed Cryptographic Facility: Program Logic Manual*

Explanation: An abend has occurred during initialization of the Programmed Cryptographic Facility. The areas dumped are PSA, NUC, LSQA, RGN, LPA, TRT, CSA, SWA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - ICTMCS01 (module in error)
SDWACSCT - ICTMCS01 (CSECT in error)
SDWAREXN - ESTAEXIT (recovery routine)

ICTMKG00, KEY GENERATOR PROGRAM

Component: Programmed Cryptographic Facility (5752-XY500)

Issuing Module: ICTMKG00 - ESTAE

PLM: *Programmed Cryptographic Facility: Program Logic Manual*

Explanation: An abend has occurred during key generator program processing in ICTMKG00. The areas dumped are PSA, NUC, LSQA, RGN, TRT, CSA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - ICTMKG00 (module in error)
SDWACSCT - ICTMKG00 (CSECT in error)
SDWAREXN - ESTAEXIT (recovery routine)

ICTMKG01 HANDLE SYSIN MODULE

Component: Programmed Cryptographic Facility (5752-XY500)

Issuing Module: ICTMKG01 - ESTAE

PLM: *Programmed Cryptographic Facility: Program Logic Manual*

Explanation: An abend has occurred during key generator control statement processing in ICTMKG01. The areas dumped are PSA, NUC, LSQA, RGN, TRT, CSA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - ICTMKG00 (module in error)
SDWACSCT - ICTMKG01 (CSECT in error)
SDWAREXN - RECVRTN (recovery routine)

ICTMKM01, START CRYPTOGRAPHY COMMAND

Component: Programmed Cryptographic Facility (5752-XY500)

Issuing Module: ICTMKM01 - ESTAE

PLM: *Programmed Cryptographic Facility: Program Logic Manual*

Explanation: An abend has occurred during start cryptography command processing in ICTMKM01. The areas dumped are PSA, NUC, LSQA, RGN, LPA, TRT, CSA, SWA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - ICTMKM01 (module in error)
SDWACSCT - ICTMKM01 (CSECT in error)
SDWAREXN - ESTAEXIT (recovery routine)

ICTMKM04 - KEY MANAGER

Component: Programmed Cryptographic Facility (5752-XY500)

Issuing Module: ICTMKM04 - FESTAE

PLM: *Programmed Cryptographic Facility: Program Logic Manual*

Explanation: An abend has occurred during GENKEY or RETKEY macro processing in ICTMKM04. The areas dumped are PSA, NUC, LSQA, RGN, LPA, TRT, CSA, SWA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - ICTMKM04 (module in error)
SDWACSCT - ICTMKM04 (CSECT in error)
SDWAREXN - ICTMKM04 (recovery routine)

Message ICT922I is issued to the master console and identifies the requested function and abend code.

ICTMSM07 - ICTMSM07 - CIPHER DUMP

Component: Programmed Cryptographic Facility (5752-XY500)

Issuing Module: ICTMSM07 - FESTAE or FRR

PLM: *Programmed Cryptographic Facility: Program Logic Manual*

Explanation: An abend has occurred during processing of a request to encipher or decipher data (CIPHER macro) in ICTMSM07. If the CIPHER macro was branch-entered, an FRR was established and a branch entry to SDUMP was used. The areas dumped are NUC, LSQA, RGN, LPA, TRT, CSA, SWA, ALLPSA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - ICTMSM07 (module in error)
SDWACSCT - ICTMSM07 (CSECT in error)
SDWAREXN - ERRFES or ERRFRR (recovery routine)

ICTMSM07 - ICTMSM08 TRNSKEY DUMP

Component: Programmed Cryptographic Facility (5752-XY500)

Issuing Module: ICTMSM07 - FESTAE

PLM: *Programmed Cryptographic Facility: Program Logic Manual*

Explanation: An abend has occurred during the processing of the translate key (TRNSKEY macro) function. The areas dumped are NUC, LSQA, RGN, LPA, TRT, CSA, SWA, ALLPSA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - ICTMSM07 (module in error)
SDWACSCT - ICTMSM08 (CSECT in error)
SDWAREXN - ERRFES (recovery routine)

ICTMSM07 - ICTMSM09 EMK DUMP

Component: Programmed Cryptographic Facility (5752-XY500)

Issuing Module: ICTMSM09 - FESTAE

PLM: *Programmed Cryptographic Facility: Program Logic Manual*

Explanation: An abend has occurred during the processing of the encipher under master key (EMK macro) function. The areas dumped are NUC, LSQA, RGN, LPA, TRT, CSA, SWA, ALLPSA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - ICTMSM07 (module in error)
SDWACSCT - ICTMSM09 (CSECT in error)
SDWAREXN - ERRFES (recovery routine)

IDA019SB:IDA121F7 - ABEND FROM BUILD IDACPA

Component: VSAM - Record Management (5752-SC1DE)

Issuing Module: IDA019SB - FRR

PLM: *VSAM Logic*

Explanation: An abnormal termination has occurred during VSAM record management processing. The FRR in IDA019SB issues the SDUMP macro. This FRR allows termination to continue. The areas dumped are PSA, NUC, RGN, TRT, CSA, and SQA.

Problem Determination: A channel program was being constructed for a VSAM global shared resources (GSR) request. Register 3 points to the IOMB for the request. For information on how the IOMB relates to the VSAM control block structure, refer to *VSAM Logic*.

IEAVEMCR - MEMORY CREATE ABNORMAL TERMINATION

Component: Supervisor Control (5752-SC1C5)

Issuing Module: IEAVEMCR - Memory Create

PLM: *System Logic Library*

Explanation: An error has occurred during memory create processing in IEAVEMCR. The ESTAE routine in IEAVEMCR issues the SDUMP macro. The areas dumped are NUC, LPA, TRT, ALLPSA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IEAVEMCR (module in error)
SDWACSCT - IEAVEMCR (CSECT in error)
SDWAREXN - MCRESTAE (recovery routine)

IEAVEMDL - MEMORY DELETE ABNORMAL TERMINATION

Component: Supervisor Control (5752-SC1C5)

Issuing Module: IEAVEMDL - Memory Delete

PLM: *System Logic Library*

Explanation: An error has occurred during memory delete processing in IEAVEMDL. The ESTAE routine in IEAVEMDL issues the SDUMP macro. The areas dumped are NUC, LPA, TRT, ALLPSA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IEAVEMDL (module in error)
SDWACSCT - IEAVEMDL (CSECT in error)
SDWAREXN - MDLESTAE (recovery routine)

IEAVEMRQ UNEXPECTED ABEND

Component: Supervisor Control (5752-SC1C5)

Issuing Module: IEAVEMRQ - Memory Request

PLM: *System Logic Library*

Explanation: An error has occurred during memory request processing in IEAVEMRQ while the global dispatcher lock was not held. The ESTAE routine in IEAVEMRQ issues the SDUMP macro. The areas dumped are NUC, LPA, TRT, ALLPSA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IEAVEMRQ (module in error)
SDWACSCT - IEAVEMRQ (CSECT in error)
SDWAREXN - MRQESTAE (recovery routine)

IEAVEMRQ - UNEXPECTED ABEND WITH DISPATCHER LOCK

Component: Supervisor Control (5752-SC1C5)

Issuing Module: IEAVEMRQ - Memory Request

PLM: *System Logic Library*

Explanation: An error has occurred during memory create (ASID or ASCB) processing while the global dispatcher lock was held. The FRR in IEAVEMRQ issues the SDUMP macro. The areas dumped are NUC, LPA, TRT, ALLPSA, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC. It contains information concerning the associated ASCB and ASCR.

IEAVTRT2 - UNRECOVERABLE ABEND FAILURE

Component: RTM - RTM2 Processing (5752-SC1CM)

Issuing Module: IEAVTRT2

PLM: *System Logic Library*

Explanation: An unrecoverable error has occurred during RTM2 processing. On completion of IEAVTRT2 processing, the current task tree is set nondispatchable and the failing address space is terminated. The areas dumped are PSA, NUC, RGN, and SQA.

Problem Determination: The most recent RTM2WA addressed by the TCB contains the most pertinent information. However, if a RTM2WA does not exist, not enough storage was available in LSQA or SQA.

IEC251I, VSAM GSR FORCE DLVRP DUMP DATA

Component: VSAM - CLOSE Processing (5665-28418)

Issuing Module: IDA0200T

PLM: *Virtual Storage Access Method (VSAM) Logic*

Explanation: VSAM was closing the last data set opened against the resource pool, and the ASCB originating the pool had already terminated. A force delete of the pool was done to release resources and storages.

Problem Determination: This is an informational dump (with associated message IEC251I). It indicates that a FORCE DLVRP was done to free storage used by a GSR (global shared resources) pool, with an attempt to dump control blocks to the SYS1.DUMP data set. For additional information, refer to message IEC251I in *System Messages.*

IEC999I IFG0RR0A,IFG0RR0F,jobname,stepname,
WORKAREA=address

Component: Open/Close/EOV (5665-28413)

Issuing Module: IGF0RR0F - ESTAE

PLM: *Open/Close/EOV Logic*

Explanation: An error has occurred during open, close, or EOV processing. If the TIOT is available, jobname and stepname indicate the name of the affected job. WORKAREA=address indicates the address of the task recovery routine (TRR) work area. The areas dumped are NUC and RGN.

Problem Determination: For additional information, refer to message IEC999I in *System Messages.*

IEC999I IFG0RR0A,error-module,jobname,stepname,
WORKAREA=address

Component: Open/Close/EOV (5665-28413)

Issuing Module: IFG0RR0A - ESTAE

PLM: *Open/Close/EOV Logic*

Explanation: An error has occurred during open, close, EOV, or DADSM processing. error-module indicates the name of the module in which the error occurred. If the TIOT is available, jobname and stepname indicate the name of the affected job. WORKAREA=address indicates the address of the task recovery routine (TRR) work area. The area dumped is RGN.

Problem Determination: For additional information, refer to message IEC999I in *System Messages.*

IEC999I IFG0RR0A,error-module,jobname,stepname,
WORKAREA=address

Component: Open/Close/EOV (5665-28413)

Issuing Module: IFG0RR0E - ESTAE

PLM: *Open/Close/EOV Logic*

Explanation: An error has occurred during open, close, EOV, or DADSM processing. error-module indicates the name of the module in which the error occurred. If the TIOT is available, jobname and stepname indicate the name of the affected job. WORKAREA=address indicates the address of the task recovery routine (TRR) work area. The areas dumped are NUC and RGN.

Problem Determination: For additional information, refer to message IEC999I in *System Messages.*

IEC999I IFG0TC0A,subroutine,jobname,stepname,DEB ADDR=address
IEC999I IFG0TC4A,subroutine,jobname,stepname,DEB ADDR=address
IEC999I IFG0TC5A,subroutine,jobname,stepname,DEB ADDR=address

Component: Open/Close/EOV (5665-28413)

Issuing Module: IFG0TC0A (Task Close) or IFG0TC4A (ESTAE)

PLM: *Open/Close/EOV Logic*

Explanation: An error has occurred during task close processing. If the abend occurs in one of the subroutines called by task close, the task close ESTAE routine IFG0TC4A issues SDUMP. If the error occurs during mainline task close processing, IFG0TC0A issues SDUMP. More than one SDUMP may be issued when errors are encountered in the called subroutines. The failing subroutine is indicated by subroutine in the title. If the TIOT is available, jobname and stepname indicate the name of the

affected job. DEB ADDR=address identifies the associated DEB control block. The areas dumped are NUC, RGN, CSA, and SQA.

Problem Determination: If a program check has occurred within task close processing, a software record is written to SYS1.LOGREC that includes the error module/routine information in the format of message IEC999I. For additional information, refer to message IEC999I in *System Messages.*

IEECB906 SLIP ESTAE DUMP

Component: RTM - SLIP Command (5752-SC1CM)

Issuing Module: IEECB906 - ESTAE

PLM: *System Logic Library*

Explanation: An error has occurred during SLIP or DISPLAY SLIP command processing.

Problem Determination: The SDWAVRA field in the SDWA contains the ESTAE parameter list. A software record is written to SYS1.LOGREC.

IEECB914 SLIP TSO COMM RTN ESTAE DUMP

Component: RTM - SLIP TSO Communication (5752-SC1CM)

Issuing Module: IEECB914

PLM: *System Logic Library*

Explanation: An error has occurred while a SLIP command was being entered from a TSO terminal. The area dumped is SQA.

Problem Determination: A software record is written to SYS1.LOGREC. The SDWAVRA contains a copy of the ESTAE parameter list and a copy of the SLIP TSO element (STE) associated with the SLIP command.

IEEMPS03 - DUMP OF MAIN WORKAREA

Component: Reconfiguration (5752-SC1CZ)

Issuing Module: IEEMPS03

PLM: *System Logic Library*

Explanation: An abend has occurred during QUIESCE command processing. The main work area for IEEMPS03 is dumped.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IEEMPS03 (module in error)
SDWACSCT - IEEMPS03 (CSECT in error)
SDWAREXN - ESTAERTN (recovery routine)

IEEVLDWT ERROR

Component: Reconfiguration (5752-SC1CZ)

Issuing Module: IEEVLDWT

PLM: *System Logic Library*

Explanation: An error has occurred during IEEVLDWT (load-wait) processing. The FRR routine in IEEVLDWT issues the SDUMP macro.

Problem Determination: The SDWAVRA field in the SDWA contains the FRR parameter list.

IGCT0018,jobname,stepname

Component: SAM (5665-28414)

Issuing Module: IGCT0018 - ESTAE

PLM: *SAM Logic*

Explanation: During SVC 18 (BLDL or FIND) processing, the ESTAE routine IGCT0018 issues SDUMP if either (1) an abend has occurred, (2) a previous error recovery routine has failed, or (3) a system error has occurred. For a system error, message IEC909I is issued. The areas dumped are PSA, NUC, SQA, and RGN.

Problem Determination: A software record is written to SYS1.LOGREC. Also refer to message IEC909I in *System Messages.*

IGCT002D,jobname,stepname

Component: SAM (5665-28414)

Issuing Module: IGCT002D - ESTAE

PLM: *SAM Logic*

Explanation: During SVC 24 (DEVTYPE) processing, the ESTAE routine IGCT002D issues SDUMP if either (1) an abend has occurred, (2) a previous error recovery routine has failed, or (3) a system error has occurred. For a system error, message IEC912I is issued. The areas dumped are PSA, NUC, RGN, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC. Also refer to message IEC912I in *System Messages.*

IGCT002E,jobname,stepname

Component: SAM (5665-28414)

Issuing Module: IGCT002E - ESTAE

PLM: *SAM Logic*

Explanation: During SVC 25 (track balance/overflow) processing, the ESTAE routine IGCT002E issues SDUMP if either (1) an abend has occurred, (2) a previous error recovery routine has failed, or (3) a system error has occurred. For a system error, message IEC915I is issued. The areas dumped are PSA, NUC, RGN, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC. Also refer to message IEC915I in *System Messages.*

IGCT0021,jobname,stepname

Component: SAM (5665-28414)

Issuing Module: IGCT0021 - ESTAE

PLM: *SAM Logic*

Explanation: During SVC 21 (STOW) processing, the ESTAE routine IGCT002I issues SDUMP if either (1) an abend has occurred, (2) a previous error recovery routine has failed, or (3) a system error has occurred. For a system error, message IEC911I is issued. The areas dumped are PSA, NUC, RGN, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC. Also refer to message IEC911I in *System Messages.*

IGCT005C,jobname,stepname

Component: DAM (5665-28416)

Issuing Module: IGCT005C - ESTAE

PLM: *BDAM Logic*

Explanation: During SVC 53 (exclusive control) processing, the ESTAE routine IGCT005C issues SDUMP if either (1) a previous error recovery routine has failed, or (2) a system error has occurred. Message IEC903I is issued. The areas dumped are PSA, NUC, RGN, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC. Also refer to message IEC903I in *System Messages.*

IGCT005G,jobname,stepname

Component: DAM (5665-28416)

Issuing Module: IGCT005G - ESTAE

PLM: *BDAM Logic*

Explanation: During SVC 57 (FREEDBUF) processing, the ESTAE routine IGCT005G issues SDUMP if either (1) an error other than a program check occurred in the cleanup routine, (2) a previous error recovery has failed, or

(3) a system error has occurred. For a system error, message IEC905I is issued. The areas dumped are PSA, NUC, RGN, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC. Also refer to message IEC905I in *System Messages.*

IGCT006H,jobname,stepname,procstepname,744

Component: SAM (5665-28414)

Issuing Module: IGCT006H - ESTAE

PLM: *SAM Logic*

Explanation: During SVC 68 (SYNADAF/SYNADRLS) processing, the ESTAE routine IGCT006H issues SDUMP if either (1) an abend has occurred, (2) a previous error recovery routine has failed, or (3) a system error has occurred. For a system error, message IEC906I is issued. The areas dumped are PSA, NUC, RGN, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC. Also refer to message IEC906I in *System Messages.*

IGCT0069,jobname,stepname

Component: SAM (5665-28414)

Issuing Module: IGCT0069 - ESTAE

PLM: *SAM Logic*

Explanation: During SVC 69 (BSP) processing, the ESTAE routine IGCT0069 issues SDUMP if either (1) an abend has occurred, (2) a previous error recovery routine has failed, or (3) a system error has occurred. For a system error, message IEC917I is issued. The areas dumped are PSA, NUC, RGN, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC. Also refer to message IEC917I in *System Messages.*

IGCT010E,jobname,stepname

Component: SAM (5665-28414)

Issuing Module: IGCT010E - ESTAE

PLM: *SAM Logic*

Explanation: During SVC 105 (IMGLIB) processing, the ESTAE routine IGCT010E issues SDUMP if either (1) an abend has occurred, or (2) a previous error recovery routine has failed, or (3) a system error has occurred. For a system error, message IEC920I is issued. The areas dumped are PSA, NUC, RGN, and SQA.

Problem Determination: A software record written to SYS1.LOGREC. Also refer to message IEC920I in *System Messages.*

IGCT105C jobname,stepname

Component: DAM (5665-28416)

Issuing Module: IGCT105C - ESTAE

PLM: *BDAM Logic*

Explanation: During SVC 53 (exclusive control) processing, the ESTAE routine IGCT105C issues SDUMP if either (1) an abend has occurred, or (2) an error other than a program check has occurred in the cleanup routine for the first-level ESTAE routine. Message IEC903I is issued. The areas dumped are PSA, NUC, RGN, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC. Also refer to message IEC903I in *System Messages.*

IGCT1081,jobname,stepname

Component: SAM (5665-28414)

Issuing Module: IGCT1081 - ESTAE

PLM: *SAM Logic*

Explanation: During SVC 81 (SETPRT) processing, the ESTAE routine IGCT1081 issues SDUMP if either (1) the DEB is invalid, (2) the FCB image is invalid, or (3) a system error has occurred. If the ESTAE routine was not entered directly from RTM, then message IEC918 is issued. The areas dumped are PSA, NUC, RGN, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC. Also refer to message IEC918I in *System Messages.*

IGC0002F CATLG CTLR 3

Component: Catalog Controller 3 - CVOL Processor (5665-28420)

Issuing Module: IGC0002F - ESTAE

PLM: *CVOL Processor Logic*

Explanation: During SVC 26 (CATALOG/INDEX/LOCATE) processing, the catalog controller ESTAE routine IGC0002F issues SDUMP if any OCx abend occurs. The ESTAE routine frees storage resources so they are not lost to the system. The areas dumped are PSA, LSQA, and RGN.

IKJEFLGM REQUEST

Component: TSO Scheduler (5752-SC1T4)

Issuing Module: IKJEFLGM (LOGON Message Module)

PLM: *System Logic Library*

Explanation: An error has occurred during LOGON processing. SDUMP is taken if one of the following messages is issued:

IKJ56451 - results from an installation-exit error.
IKJ56452 - results from a system error.
IKJ600I - results from an I/O, OBTAIN, or OPEN error.
IKJ603I - results from an installation-exit abend.
IKJ608I - results from a TSO service routine error.

The areas dumped are NUC, RGN, SQA, and LPA if TSO dump is requested.

Problem Determination: Refer to message IKJ600I, IKJ603I, and IKJ608I in *System Messages.*

IKTLTERM - I/O ERROR

Component: TSO/VTAM (5665-28002)

Issuing Module: IKTLTERM

PLM: *VTIOC and TCAS Logic*

Explanation: TSO/VTAM has issued an abend due to an unrecoverable I/O error. The installation had requested the SVC dump by specifying the RPL sense code for the I/O error via the RCFBDUMP keyword in the TSOKEYxx member of SYS1.PARMLIB.

Problem Determination: Excessive line or hardware errors might be occurring. A software record is written to SYS1.LOGREC.

IOS - IECVERPL ERROR

Component: IOS (5752-SC1C3)

Issuing Module: IECVERPL

PLM: *System Logic Library*

Explanation: An error has occurred while either IECVERPL was in control or an ERP that does not have a recovery routine was in control. The areas dumped are PSA, SQA, LSQA, and TRT.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IECVERPL
SDWACSCT - IECVERPL
SDWAREXN - ERPLESTA

ISAM INTRFC,OPEN,IDA0192I,IDAICIA1,**AUDIT NOT STARTED**
ISAM INTRFC,OPEN,IDA0192I,IDAICIA1,**IDA0192I IN CONTROL**
ISAM INTRFC,CLOSE,IDA0200S,IDAICIA1,**AUDIT UNAVAILABLE**
ISAM INTRFC,CLOSE,IDA0200S,IDAICIA1,**IDAIIPM1 IN CONTROL**
ISAM INTRFC,CLOSE,IDA0200S,IDAICIA1,**IDA0200S IN CONTROL**

Component: VSAM - ISAM-Interface (5665-28418)

Issuing Module: IDAICIA1 - ESTAE

PLM: *Virtual Storage Access Method (VSAM) Logic*

Explanation: An error has occurred during the opening or closing of a DCB via the ISAM interface. Module IDAICIA1 (ISAM-interface data-set management recovery routine) issues SDUMP. One of the five titles appears depending on the error and whether open or close was in control at the time of error.

Problem Determination: Depending on the error, some or all of the following areas are dumped: the dump list itself, user's DCB, protected copy of the user's DCB, OPEN/CLOSE work area, recovery work area, IICB, ACB, EXLST, buffers and message area.

ISSUER=IEFAB4ED,ERRCSECT=csect,COMPID=5752-SC1B4,
COMPON=DEVICE ALLOCATION-sss...sss

Component: Allocation (5752-SC1B4)

Issuing Module: IEFAB4ED - Allocation Common ESTAE Exit

PLM: *System Logic Library* and *System Initialization Logic*

Explanation: Fields in the dump title are:

csect — Name of the failing CSECT. If the name of the failing CSECT is not available, csect contains "SEE VRA." In addition, a message is put in the VRA that states: "THE CSECT IN THE SDWACSCT FIELD IS THE FIRST CSECT IN THE FAILING SUBCOMPONENT, NOT NECESSARILY THE FAILING CSECT."

sss...sss — Name of the subcomponent. The following shows the names of the subcomponents and the name of the first CSECT in each subcomponent:

IEFAB4F5 - Alloc catalog control
IEFAB4I0 - Alloc initialization
IEFAB4E5 - Alloc resource manager
IEEAB401 - Alloc/unalloc put rtn
IEFAB421 - Common allocation

IEFAB4A0	- Common unallocation
IEFGB4DC	- Data set reserve/release
IEFDB400	- Dynamic allocation
IEFAB4EC	- Group lock/unlock
IEFAB451	- JFCB housekeeping
IEFBB401	- Job step allocation
IEFBB410	- Job step unallocation
IEFAB4F4	- Unalloc catalog control
IEFAB493	- Volume mount & verify

An error has occurred during allocation processing. The ESTAE routine IEFAB4ED performs general recovery processing and issues the SDUMP macro (if no SDWA exists). If an SDWA exists, additional checks on the error are made. The SDUMP macro is then issued if the error is not a user error and one of the following occurred:

- A program check
- The restart key was pressed
- A dump was not previously taken
- An abend occurred and there was no percolation or if there was percolation, it was via FRR recovery processing.

The areas dumped are LPA, ALLPSA, SQA, TRT, SUM, SWA, and LSQA.

Problem Determination: A software record is written to SYS1.LOGREC. Key control blocks used by allocation are included in the summary list in the SVC dump.

ISSUER=IEFAB4E6,ERRCSECT=csect,COMPID=5752-SC1B4,
COMPON=DEVICE ALLOCATION-sss...sss

Component: Allocation (5752-SC1B4)

Issuing Module: IEFAB4E6 - Recovery Routine

PLM: *System Logic Library* and *System Initialization Logic*

Explanation: Fields in the dump title are:

csect	Name of the failing CSECT.
sss...sss	Name of the subcomponent.

For csect and sss...sss, see SVC dump title "ISSUER=IEFAB4ED,...."

An error has occurred during allocation processing. The areas dumped are LPA, ALLPSA, SQA, TRT, SUM, and LSQA.

If the error occurred during processing related to the allocation address space (ALLOCAS), message IEF100I is issued, the allocation address space might be terminated, and allocation processing continues. For other errors, all units allocated to the failing address space are unallocated and the job is abnormally terminated.

Problem Determination: A software record is written to SYS1.LOGREC. If the recovery routine was entered due to system completion code 05C, register 0 contains a reason code. See *System Codes* for an explanation of system code 05C and reason codes. If the recovery routine was entered due to an error related to allocation address space processing, message IEF100I is also issued. See *System Messages* for an explanation of message IEF100I.

ISSUER=IEFAB4GA,ERRCSECT=csect,COMPID=5752-SC1B4,
COMPON=DEVICE ALLOCATION-sss...sss

Component: Allocation (5752-SC1B4)

Issuing Module: IEFAB4GA - DDR/Swap Allocation Interface Routine

PLM: *System Logic Library* and *System Initialization Logic*

Explanation: Fields in the dump title are:

csect	Name of the failing CSECT.
sss...sss	Name of the subcomponent.

For csect and sss...sss, see SVC dump title "ISSUER=IEFAB4ED,...."

An error has occurred while allocation was scanning the UCB pointer list. IEFAB4GA issues the SDUMP macro if a dump was not previously taken. A retry is done to exit IEFAB4GA normally. The areas dumped are LPA, ALLPSA, SQA, TRT, SUM, and LSQA.

Problem Determination: A software record is written to SYS1.LOGREC. See *System Codes* for an explanation of system code 05C, which is related to this dump.

ISSUER=IEFAB4SF,ERRCSECT=csect,COMPID=5752-SC1B4,
COMPON=DEVICE ALLOCATION-sss...sss

Component: Allocation (5752-SC1B4)

Issuing Module: IEFAB4SF - Allocation Spool File Processor

PLM: *System Logic Library* and *System Initialization Logic*

Explanation: Fields in the dump title are:

csect	Name of the failing CSECT.
sss...sss	Name of the subcomponent.

For csect and sss...sss, see SVC dump title "ISSUER=IEFAB4ED,...."

An error has occurred while allocation was processing a request to segment a SYSOUT data set. IEFAB4SF issues the SDUMP macro if a dump was not previously taken. The areas dumped are LPA, ALLPSA, SQA, TRT, SUM, and LSQA.

Problem Determination: A software record is written to SYS1.LOGREC.

ISSUER=IEFDB440,ERRCSECT=csect,COMPID=5752-SC1B4,
COMPON=DEVICE ALLOCATION-sss...sss

Component: Allocation (5752-SC1B4)

Issuing Module: IEFDB440 - Unit Allocation/Unallocation Service

PLM: *System Logic Library* and *System Initialization Logic*

Explanation: Fields in the dump title are:

csect	Name of the failing CSECT.
sss...sss	Name of the subcomponent.

For csect and sss...sss, see SVC dump title "ISSUER=IEFAB4ED,...."

An error has occurred during allocation processing and RTM has passed control to routine ESTAERTN in module IEFDB440. ESTAERTN issues the SDUMP macro if a dump was not previously taken. The areas dumped are LPA, ALLPSA, SQA, TRT, SUM, SWA, and LSQA. Control is returned to RTM.

Problem Determination: A software record is written to SYS1.LOGREC.

ISTAPCES - ACF/VTAM PSS ESTAE ROUTINE

Component: ACF/VTAM (5665-28001)

Issuing Module: ISTAPCES - PSS ESTAE

PLM: *ACF/VTAM Diagnosis Reference*

Explanation: An abend has occurred while an ACF/VTAM task was processing and an ACF/VTAM IRB was active. The areas dumped are SQA, NUC, RGN, LPA, TRT, ALLPSA, and CSA.

Problem Determination: A software record is written to SYS1.LOGREC. For a description of the CRA fields recorded, refer to *ACF/VTAM Data Areas.*

ISTAPCFR - ACF/VTAM PSS FUNCTIONAL RECOVERY

Component: ACF/VTAM (5665-28001)

Issuing Module: ISTAPCFR - PSS FRR

PLM: *ACF/VTAM Diagnosis Reference*

Explanation: An abend has occurred while ACF/VTAM was processing and running under an SRB. The areas dumped are ALLPSA, CSA, NUC, SQA, TRT, LPA, and RGN.

Problem Determination: A software record is written to SYS1.LOGREC. For a description of the CRA fields recorded, refer to *ACF/VTAM Data Areas.*

ISTAPCMT - ACF/VTAM ABEND IN MEMORY TERMINATION

Component: ACF/VTAM (5665-28001)

Issuing Module: ISTAPCMT

PLM: *ACF/VTAM Diagnosis Reference*

Explanation: An abend has occurred while the ACF/VTAM memory termination resource manager was processing. ACF/VTAM attempts minimal cleanup so that ACF/VTAM can be restarted. However, CSA storage might not be usable until the next IPL. The areas dumped are SQA, NUC, RGN, LPA, LSQA, TRT, ALLPSA, and CSA.

Problem Determination: A software record is written to SYS1.LOGREC. For a description of the CRA fields recorded, refer to *ACF/VTAM Data Areas.*

ISTATM00 - ACF/VTAM TERMINATION TASK INIT|TERM|ESTAE

Component: ACF/VTAM (5665-28001)

Issuing Module: ISTATM00 - ESTAE

PLM: *ACF/VTAM Diagnosis Reference*

Explanation: An abend has occurred while the ACF/VTAM termination task was processing. The ESTAE routine ISTATM00 issues the SDUMP macro for abends that occur during ACF/VTAM processing (but not for abends that occur during application processing). The areas dumped are SQA, LSQA, TRT, ALLPSA, CSA, and RGN.

Problem Determination: A software record is written to SYS1.LOGREC. For a description of the CRA fields recorded, refer to *ACF/VTAM Data Areas.*

ISTINCST - ACF/VTAM STAE EXIT AND RECOVERY

Component: ACF/VTAM (5665-28001)

Issuing Module: ISTINCST - ESTAE

PLM: *ACF/VTAM Diagnosis Reference*

Explanation: An abend has occurred while the ACF/VTAM jobstep task was processing. The areas dumped are SQA, NUC, RGN, LPA, TRT, ALLPSA, and CSA.

Problem Determination: None.

ISTORMMG - ACF/VTAM FRR DUMP

Component: ACF/VTAM (5665-28001)

Issuing Module: ISTORMMG

PLM: *ACF/VTAM Diagnosis Reference*

Explanation: An abend has occurred while ISTORMMG was running in SRB mode. ISTORMMG frees CSA storage and recovery is attempted by zeroing the CSA to-be-freed queue (ATCORTBF). The areas dumped are SQA, NUC, RGN, LPA, ALLPSA, and CSA.

Problem Determination: A software record is written to SYS1.LOGREC. For a description of the CRA fields recorded, refer to *ACF/VTAM Data Areas.*

JES2 FSI ERROR. CODE=cde RC=rc (text)

Component: JES2 (5752-SC1BH)

Issuing Module: HASPFSSM

PLM: *JES2 Logic*

Explanation: A catastrophic error has occurred in the JES2 functional subsystem interface (FSI) support routines (HASPFSSM) for which JES2 issued a $ERROR macro. HASPFSSM was operating in a functional subsystem (FSS) address space. JES2 terminates the FSS address space.

The HASPFSSM error routine FSMCATER issued the SDUMP macro. The areas dumped are ALLPSA, RGN, TRT, SQA, CSA, LPA, SWA, and LSQA.

This dump is associated with JES2 message $HASP750 and system abend code 02C.

Problem Determination: See message $HASP750 in *System Messages* and abend code 02C in *System Codes* for information on this error.

JES3 LOCATE SUBTASK ABEND

Component: JES3 (5752-SC1BA)

Issuing Module: IATLVLC

PLM: *JES3 Logic*

Explanation: An abend has occurred during IATLVLC (locate subtask) processing. The ESTAE routine established by IATLVLC is given control to examine the function control table (FCT) active at the time of termination to determine which function or DSP has failed. The areas dumped are SQA, CSA, PSA, RGN, LPA, and TRT.

Problem Determination: A software record is written to SYS1.LOGREC.

JES3 SNA FRR IATSNDF

Component: JES3 (5752-SC1BA)

Issuing Module: IATSNDF - FRR

PLM: *JES3 Logic*

Explanation: The FRR routine (IATSNDF) handles abends that occur during SNA RJP processing under an SRB. Each time that the FRR is entered, an SVC dump is taken. Therefore, control of dumping is dependent on the FRR's recursion control to prevent more than two retry failures. (A dump is taken for every retry failure.) The areas dumped are SQA, ALLPSA, NUC, LSQA, RGN, TRT, CSA, and LPA.

Problem Determination: The SDWA is logged after it is updated with LCB data if available. A software record is written to SYS1.LOGREC.

JOB=jobname hh:mm:ss yy.ddd DUMP BY IGG0CLA9 - VSAM CATALOG MANAGEMENT

Component: VSAM - Catalog Management (5665-28418)

Issuing Module: IGG0CLA9 - ESTAE

PLM: *Catalog Diagnosis Guide*

Explanation: An abend has occurred during catalog management processing. The ESTAE routine IGG0CLA9 issues the SDUMP macro, frees storage resources, and backs-out partially defined catalog entries in the VSAM catalogs. Message IEC338I is also issued if a validity check failed on a user field parameter list (FPL) or a catalog parameter list (CPL).

Problem Determination: A software record is written to SYS1.LOGREC. The SDWAVRA includes the following:

Offset	Length	Meaning
0(0)	8	c'IGG0CLA9'
8(8)	3	Entry point address of IGG0CLA9.
11(B)	8	Name of the last routine called.
19(13)	3	Entry point address of the last routine called.
22(16)	8	Name of the calling routine.
30(1E)	3	Entry point address of the calling routine.
33(21)	4	c'CPL='
37(25)	28	User's CPL.

RACF INITIALIZATION FAILURE

Component: RACF (5752-XXH00)

Issuing Module: ICHSEC02 - ESTAE

PLM: *Resource Access Control Facility (RACF): Program Logic Manual*

Explanation: An abend has occurred during ICHSEC00 (RACF initialization) processing. The areas dumped are SQA, CSA, NUC, and RGN.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - ICHSEC00 (module in error)
SDWAREXN - ICHSEC02 (recovery routine)
SDWACID - XXH00
SDWACSCT - ICHSEC00
SDWAEAS - 1 if SDUMP is taken by ICHSEC00
SDWAREQ - 0 if SDUMP is taken by ICHSEC00

RCT DUMPING LSQA

Component: Region Control Task (5752-SC1CU)

Issuing Module: IEAVAR00 - ESTAE

PLM: *System Logic Library*

Explanation: The ESTAE routine in IEAVAR00 issues the SDUMP macro when a previous error recovery routine could not diagnose the error and either (1) RCT's RB was in control, (2) an error occurred in the previous recovery exit, (3) an RCT FRR routine requested the dump, or (4) retry recursion has occurred.

Problem Determination: A software record is written to SYS1.LOGREC. Pay particular attention to the cause of error flags and the RCT flags in the SDWAVRA. Additional footprints and data are available in the RCTD of the dumped storage.

RECORD PERMANENT ERROR,COMP=RTM,COMPID=SC1CM,
ISSUER=IEAVTRET

Component: RTM - RECORD Macro (5752-SC1CM)

Issuing Module: IEAVTRET - ESTAE

PLM: *System Logic Library*

Explanation: An operation exception (0C1) has occurred while IEAVTRET (RECORD macro processing) was in control or a second error occurred while processing a "temporary error" type. The recording function is turned off and message IEA896I is issued, stating that the recording function is not active. A return code of 20 is issued for following RECORD macro requests. The areas dumped are SQA, NUC, LPA, and RGN.

RECORD TEMPORARY ERROR,COMP=RTM,COMPID=SC1CM,
ISSUER=IEAVTRET

Component: RTM - RECORD Macro (5752-SC1CM)

Issuing Module: IEAVTRET

PLM: *System Logic Library*

Explanation: A protection exception (0C4) or priviledged operation (0C2) has occurred while IEAVTRER (RECORD macro processing) was in control and the RCB buffer was not being manipulated by the requesting routine, or the recording task (IEAVTRET) was in control and the error was not an operation exception (0C1). This abend is not a "permanent error" type. The areas dumped are SQA, NUC, LPA, and RGN.

REQUESTOR=xxxxxxxx,ISSUER=ISGCRCV,COMPID=SCSDS,
COMPON=GRS

Component: Global Resource Serialization

Issuing Module: ISGCRCV - ESTAE

PLM: *System Logic Library*

Explanation: An error has occurred while one of the following command processing modules was processing. The field xxxxxxxx in the dump title indicates the failing module.

ISGCMDI	ISGCQMRG	ISGNGRSP
ISGCMDR	ISGCQSC	
ISGCPRG	ISGCRST	

The ESTAE module ISGCRCV issues the SDUMP macro. If the error occurred in ISGCMDI, the master address space is dumped. For errors in the other modules, the global resource serialization address space is dumped.

Problem Determination: A software record is written to SYS1.LOGREC.

RESOURCE MANAGER

Component: Initiator (5752-SC1B6)

Issuing Module: IEFISEXR - ESTAE

PLM: *System Logic Library*

Explanation: A program check or a restart interruption has occurred in the initiator or a subsystem interface resource manager. The ESTAE routine IEFISEXR issues the SDUMP macro. The areas dumped are SQA, PSA, LSQA, RGN, LPA, TRT, CSA, and NUC.

Problem Determination: A software record is written to SYS1.LOGREC.

RESTART INTERRUPT IN CONVERTER**IEFNB9CR**

Component: Converter (5752-SC1B9)

Issuing Module: IEFNB9CR - Converter Recovery Routine

PLM: None (refer to microfiche)

Explanation: A restart interruption has occurred during converter processing. The ESTAE routine IEFNB9CR issues the SDUMP macro. The areas dumped are LSQA, SWA, RGN, and LPA.

Problem Determination: A software record is written to SYS1.LOGREC.

RESTART INTERRUPT IN INTERPRETER**IEFNB9IR**

Component: Interpreter (5752-SC1B9)

Issuing Module: IEFNB9IR - Interpreter Recovery Routine

PLM: None (refer to microfiche)

Explanation: A restart interruption has occurred during interpreter processing. The recovery routine IEFBN9IR issues the SDUMP macro. The areas dumped are LSQA, SWA, RGN, and NUC.

Problem Determination: A software record is written to SYS1.LOGREC.

SCHEDULER JCL FACILITY FAILURE,ABEND=nnn,
COMPON=SCHR-SJF,COMPID=BB131,ISSUER=IEFSJCNL

Component: Scheduler JCL Facility (5752-BB131)

Issuing Module: IEFSJCNL

PLM: *System Logic Library*

Explanation: An abend occurred while the scheduler JCL facility (SJF) was processing. The areas dumped are SWA, CSA, LPA, and LSQA.

Problem Determination: A software record is written to SYS1.LOGREC. The SDWAVRA field (in the SDWA) contains:

- SJF module footprints that are useful in determining the execution path taken during processing.
- Addresses of the SJF local storage area, control work area, and parameter list.

The SDWA also includes:

SDWAMODN - IEFSJCNL (load module)
SDWACSCT - Name of the SJF module that failed
SDWAREXN - IEFSJCNL (recovery routine)
SDWACID - BB131
SDWACIDB - 5752
SDWARRL - RECOVERY (recovery routine label)
SDWAMLVL - Product level of failing module
SDWASC - SCHR-SJF, name of module that failed

SDUMP - IGG0CLCA CVOL CATALOG MANAGEMENT

Component: Catalog Controller 3 - CVOL Processor (5665-28420)

Issuing Module: IGG0CLCA - ESTAE

PLM: *CVOL Processor Logic*

Explanation: An abend has occurred in the first CSECT of the CVOL processor mapper. The ESTAE routine IGG0CLCA issues the SDUMP macro, and dequeues the PCCB and DSNAME resources. The areas dumped are PSA, LSQA, LPA, and RGN.

SDUMP - IGG0CLCD - CVOL CATALOG MANAGEMENT

Component: Catalog Controller 3 - CVOL Processor (5665-28420)

Issuing Module: IGG0CLCD - ESTAE

PLM: *CVOL Processor Logic*

Explanation: An abend has occurred while IGG0CLCD was building catalog entries for CVOLs. The ESTAE routine IGG0CLCD issues the SDUMP macro and frees resources. The areas dumped are PSA, LSQA, LPA, and RGN.

SLIP ID=xxxx

Component: RTM - SLIP Processor (5752-SC1CM)

Issuing Module: IEAVTSLP

PLM: *System Logic Library*

Explanation: A SLIP trap has matched and the action specified on the trap definition is ACTION=SVCD. IEAVTSLP issues the SDUMP macro. The areas dumped are defaulted or specified on the SDATA, LIST, SUMLIST, and ASIDLST keywords of the SLIP command. ID=xxxx is the SLIP trap identifier.

Problem Determination: None - the system is functioning as requested.

SMF ABEND, ERRMOD=IFAPCWTR, RECVMOD=IFAPCWTR

Component: SMF (5752-SC100)

Issuing Module: IFAPCWTR - FRR

PLM: *System Logic Library*

Explanation: An abend has occurred while moving SMF records from the user area into buffers in the SMF address space. The areas dumped are PSA, NUC, RGN, LPA, SQA, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - Load module in error
SDWACSCT - CSECT in error
SDWAREXN - ESTAE module name

SMF ABEND, ERRMOD=xxxxxxxx, RECVMOD=IEEMB830

Component: SMF (5752-SC100)

Issuing Module: IEEMB830

PLM: *System Logic Library*

Explanation: An abend has occurred during SMF record processing. If xxxxxxxx is IEFU83 or IEFU84, the error occurred during processing by the installation exit. Otherwise, xxxxxxxx is IEEMB830. The areas dumped are PSA, NUC, RGN, SQA, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - Load module in error
SDWACSCT - CSECT in error
SDWAREXN - ESTAE module name

SMF ABENDED, ERRMOD=IEEMB834, RECVMOD=IEEMB834

Component: SMF (5752-SC100)

Issuing Module: IEEMB834 - FRR

PLM: *System Logic Library*

Explanation: An abend has occurred during the SRB mode processing which writes to the SMF recording data set. The areas dumped are PSA, NUC, RGN, LPA, SQA, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - Load module in error
SDWACSCT - CSECT in error
SDWAREXN - ESTAE module name

The FRR parameter area contains footprints and is mapped by the structure FRRPARM in the IHAFRRS control block.

SMF TIMER - IEEMB839

Component: SMF (5752-SC102)

Issuing Module: IEEMB839 - FRR

PLM: *System Logic Library*

Explanation: An error has occurred in the SMF timer module while the dispatcher lock was held. The areas dumped are PSA, NUC, RGN, SQA, LPA, TRT, and SUMDUMP.

Problem Determination: A software record is written to SYS1.LOGREC.

SRM - IRARMSRV 55F ABEND DURING XMPOST

Component: SRM (5752-SC1CX)

Issuing Module: IRARMSRV

PLM: *System Logic Library*

Explanation: An error has occurred during the cross-address-space post function. The post was requested by module IRARMEVT to notify the issuer of a REQSWAP or TRANSWAP that the swap is complete or that the address space became non-swappable before the swap could be initiated. The address space being posted is terminated with a 55F completion code. The areas dumped are PSA, SQA, and TRT.

Problem Determination: The terminating address space's ASCB and OUCB are copied into the SDUMP buffer pointed to be CVTSDBF. The buffer fields are mapped by SDMPBUFF in module IRARMSRV.

SRM RECOVERY ENTERED,COMPON=SRM,COMPID=SC1CX, ISSUER=IRARMERR

Component: SRM (5752-SC1CX)

Issuing Module: IRARMERR - FRR

PLM: *System Logic Library*

Explanation: An error has occurred during SRM processing. Depending on the error, retry of the failing function is attempted or the error is percolated. The current address space is dumped.

Problem Determination: A software record is written to SYS1.LOGREC. The variable recording area in the SDWA contains the abending module name, module level, entry point address, recovery routine name, and the six-word recovery parameter area (RRPA).

SSICS ABEND 6FB

Component: JES3 (5752-SC1BA)

Issuing Module: IATSSCM

PLM: *JES3 Logic*

Explanation: A system error has occurred while IATSSCM (subsystem communication scheduler) was processing in an address space other than JES3's address space. Abend 6FB is issued. The areas dumped are PSA, RGN, LPA, TRT, CSA, NUC, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC. For a description of code 6FB, refer to *System Codes*.

SSICS ESTAE-IATSSCM

Component: JES3 (5752-SC1BA)

Issuing Module: IATSSCM

PLM: *JES3 Logic*

Explanation: IATSSCM (subsystem communication scheduler) was not able to reduce the system impact caused by communication failures for the second time. JES3 is put in the IATSSCM quiesce condition. The areas dumped are PSA, RGN, LPA, TRT, CSA, NUC, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC.

STORAGE DUMP TAKEN AT ENTRY TO IEEMB812 ESTAE EXIT

Component: SRM (5752-SC1CX)

Issuing Module: IEEMB812 - SRM SET Processor

PLM: *System Logic Library*

Explanation: An error has occurred during SRM processing of a SET command. The new tables are freed and the old controls remain in effect. The SET command is retried. If the error recurs, IEEMB812 percolates the error.

Problem Determination: A software record is written to SYS1.LOGREC.

STORAGE DUMP TAKEN AT ENTRY TO IRARMERR

Component: SRM (5752-SC1CX)

Issuing Module: IRARMERR - FRR

PLM: *System Logic Library*

Explanation: An error has occurred during SRM processing. Depending on the error, retry of the failing function is attempted or the error is percolated. The current address space is dumped.

Problem Determination: A software record is written to SYS1.LOGREC. The variable recording area in the SDWA contains a message that gives an offset into the data module IRARMCNS. This offset is the location of the control block for the SRM routine in control when the error occurred.

SWA CREATE

Component: SWA Manager (5752-SC1B5)

Issuing Module: IEFIB645

PLM: *System Logic Library*

Explanation: A program check or a restart interruption has occurred during interpreter, restart, warmstart, or SWA create processing. The recovery routine IEFIB645 issues the SDUMP macro. The areas dumped are SQA, PSA, LSQA, RGN, LPA, TRT, CSA, and NUC.

Problem Determination: A software record is written to SYS1.LOGREC.

TCAS DUMP

Component: TSO/VTAM (5665-28002)

Issuing Module: IKTCAS52

PLM: *VTIOC and TCAS Logic*

Explanation: TCAS (terminal control address space) is terminating because (1) the operator requested termination via the STOP command, or (2) a program check has occurred. The dump is taken as a result of the operator responding DUMP to message IKT012D.

TIMER FRR DUMP

Component: Timer Supervisor (5752-SC1CV)

Issuing Module: IEAVRTI1 - FRR

PLM: *System Logic Library*

Explanation: An error has occurred during timer supervision processing. In module IEAVRTI1, subroutine IEAVRSWR (entry point IEAVRFRR) issues the SDUMP macro. The areas dumped are PSA, NUC, SQA, TRT, and LSQA for the current address space.

Problem Determination: A software record is written to SYS1.LOGREC. The data area TFRRPARM (mapped in IEAVRTI1) is saved in the SDWAVRA. TFRRPARM contains indicators that tell the type of processing taking place and the locks held at the time of the error, as well as the results of the TQE validation process. The SDWA also includes:

SDWAMODN - Load module name
SDWACSCT - Module name
SDWASC - CSECT name
SDWAREXN - IEAVRFRR (recovery routine)

TSO OUTPUT CP ESTAE

Component: TSO Scheduler (5752-SC1T4)

Issuing Module: IKJCT460 - ESTAE

PLM: *TSO Command Processor Logic, Volume IV*

Explanation: An abend error or a DETACH with STAE has occurred during TSO command processing. The ESTAE exit routine IKJCT460 receives control from the supervisor and issues the SDUMP macro for (1) x0A abends (except 80A), and (2) all other abends except for a DETACH with STAE, the abends B37, D37, E37, 913, 622, and 222. The areas dumped are RGN, NUC, SQA, and LPA.

Problem Determination: For some errors, a software record is written to SYS1.LOGREC. If a workarea was obtained, SDWAREXN='IKJCT460' and the SDWA is based on register 1 for the SETRP macro.

TSO SDUMP FROM IKJEFT05 - THE TMP ESTAE ROUTINE

Component: TSO Scheduler (5665-28502)

Issuing Module: IKJEFT05

PLM: *TSO/E Terminal Monitor Program and Service Routines Logic*

Explanation: The TMP ESTAE exit routine, IKJEFT05, issues the SDUMP macro on the first occurrence of an error in a TMP module. The areas dumped are NUC, LSQA, RGN, TRT, and SQA.

Problem Determination: A software record is written to SYS1.LOGREC.

TSOLOGON ESTAE

Component: TSO Scheduler (5752-SC1T4)

Issuing Module: IKJEFLS - ESTAE

PLM: *System Logic Library*

Explanation: A program check or PSW restart interruption has occurred during TSO logon initialization or scheduling. The ESTAE routine IKJEFLS issues the SDUMP macro. The areas dumped are RGN, NUC, SQA, and LPA.

Problem Determination: A software record is written to SYS1.LOGREC. Register 1 points to the SDWA (if one exists) and includes:

SDWAMODN - IKJEFLA - Name of the abending load module.
SDWAREXN - IKJEFLS - Name of the ESTAE routine.

TSOLOGON ESTAI

Component: TSO Scheduler (5752-SC1T4)

Issuing Module: IKJEFLGB - Prompter's ESTAI

PLM: *System Logic Library*

Explanation: During logon processing, the ESTAI routine IKJEFLGB issued the SDUMP macro for (1) a program check, (2) a PSW restart condition, or (3) an abend in IKJEFLD (logon pre-prompt exit). The areas dumped are RGN, NUC, SQA, and LPA.

Problem Determination: A software record is written to SYS1.LOGREC. If a SDWA exists:

- Register 1 contains the address of the STAE work area.
- Register 14 contains the return address.

If a SDWA does not exist:

- Register 1 contains the abend code.
- Register 2 contains a pointer to the LWA.
- Register 14 contains the return address.

VSAM CHECKPOINT (IDA0xxxx) or VSAM RESTART (IDA0xxxx)

with one of the following:

MACHINE CHECK
PROGRAM CHECK LOCATION=xxxxxx
RESTART KEY DEPRESSED
PAGING ERROR
ABEND Sxxx, Uxxxx, REGISTER 15=xxxxxxxx

Component: VSAM - Checkpoint/Restart (5665-28418)

Issuing Module: IDACKRA1 - ESTAE

PLM: *Virtual Storage Access Method (VSAM) Logic*

Explanation: An error has occurred during VSAM checkpoint or restart processing. The ESTAE routine issues the SDUMP macro. The title on the dump depends on the type of error and whether checkpoint or restart was in control at the time of error. The areas dumped are SQA, LPA, and user's region.

WTO USER EXIT xxxxxxxx ABENDED,COMPON=COMMTASK,
COMPID=SC1CK,ISSUER=IEAVX600,ABEND=xxx,RSN=xxxxxxxx

Component: Communications Task (SC1CK)

Issuing Module: IEAVX600 - WTO User Exit Interface Routine

PLM: *System Logic Library*

Explanation: An error has occurred in a WTO user exit routine that was called by IEAVX600. Recovery routine XITABEND issues a summary SVC dump with the following areas included:

- IEAVX600
- IEAVX600's dynamic area
- UXIR (IEAVX600's parameter list)
- WQE for a single line or the first line of a multi-line message
- The user exit in error
- Minor WQE if minor currently being processed

Problem Determination: A software record is written to SYS1.LOGREC. The SDWAVRA contains:

- Base registers
- Data register
- Address of the CTXT (user exit interface)
- Message ID of the message currently being processed
- Address of the major WQE
- Address of the exit routine in error
- Address of the minor WQE, if minor line currently being processed

Operator- and Caller-Defined SVC Dump Titles

This topic provides diagnostic information for the modules that initiate SVC dumps via the SDUMP macro but where the dump title is defined by the system operator or the caller of SVC dump.

variable title - supplied by the system operator

Component: Master Scheduler Commands (5752-SC1B8)

Issuing Module: IEECB866 - Console Dump

PLM: *System Logic Library*

Explanation: The system operator has issued the DUMP command and specified the title of the SVC dump on the command.

Problem Determination: None.

variable title - supplied by the system operator

Component: JES2 (5752-SC1BH)

Issuing Module: HASPTERM or HASPRAS

PLM: *JES2 Logic*

Explanation: The system operator has entered an SVC dump title in response to the $HASP098 message. This title overrides the default dump title. The areas dumped are PSA, NUC, RGN, TRT, SQA, CSA, LPA, and SWA.

Problem Determination: For information on the error, refer to messages $HASP098 and $HASP095 in *JES2 Messages.*

variable title - supplied by the caller

Component: Task Manager (5752-SC1CL)

Issuing Module: IEAVECH0 - CHAP Service Routine

PLM: *System Logic Library*

Explanation: An error has been detected in the TCB dispatching queue for the current address space by the TCB queue verification routine. The FRR (IGC044R1) issues the SDUMP macro. The areas dumped are LSQA, SQA, and TRT.

Problem Determination: A software record is written to SYS1.LOGREC. The SDWAVRA contains the associated TCB-queue-verification output.

SVC Dumps Without Titles

This topic provides diagnostic information for the modules that initiate SVC dumps but where no titles are supplied on the SDUMP macro.

no title

Component: Catalog Controller 3 - CVOL Processor (5665-28420)

Issuing Module: IGG0CLCB - ESTAE

PLM: *CVOL Processor Logic*

Explanation: An abend has occurred during the processing of a GENERIC LOCATE request for a CVOL. All storage resources are freed and the CVOL processor SDUMP routine issues the SDUMP macro. The area dumped is LPA.

no title

Component: IOS (5752-SC1C3)

Issuing Module: IGC0001F

PLM: *System Logic Library*

Explanation: An error has occurred while IGC0001F was processing and holding a lock.

Problem Determination: A software record is written to SYS1.LOGREC and includes:

SDWAMODN - IGC0001F
SDWACSCT - IGC016
SDWAREXN - PURGEFRR
SDWACID - SC1C3

no title

Component: JES3 (5752-SC1BA)

Issuing Module: IATIIII (IATYIIW work area)

PLM: *JES3 Logic*

Explanation: An abend has occurred during interpreter/initiator (IATIIII) processing. The ESTAE routine established by IATIIII is given control to examine the function control table (FCT) active at the time of termination to determine which function or DSP failed. The areas dumped are PSA, RGN, LPA, TRT, and CSA.

Problem Determination: A software record is written to SYS1.LOGREC. Register 9 points to a work area containing formatted messages.

no title

Component: RTM (5752-SC1CM)

Issuing Module: IEAVTRTE

PLM: *System Logic Library*

Explanation: The SDUMP macro is issued by the RTM2 exit processor (IEAVTRTE) as a result of an error during abnormal termination of a subtask of the jobstep task. The jobstep task is terminated with a D0D completion code. The current task is set non-dispatchable in order to wait for the jobstep task to detach it. If another failure occurs, the address space is terminated. The areas dumped are SQA, LSQA, LPA, TRT, CSA, and SWA.

Problem Determination: The RTM2WAs addressed by the current TCB contain the most pertinent information.

Module to SVC Dump Title Cross-Reference

This topic lists, in alphameric order, the MVS modules that issue the SDUMP macro and provides the titles of the SVC dumps specified by the modules on the SDUMP macro.

Issuing Module	Dump Title
ADYPSTD	COMPID=5752-SC143,ISSUER=ADYPSTD...
ADYSETP	COMPID=5752-SC143,ISSUER=ADYSETP...
ADYTRNS	COMPID=5752-SC143,ISSUER=ADYTRNS...
AHLGTFI	AHL007I GTF TERMINATING ON ERROR CONDITION
AHLMCER	ERROR IN MODULE AHLMCER
AHLREADR	DUMP OF AHLREADR
AHLSBUF	DUMP OF GTF MODULE AHLSBLOK
AHLSETEV	ERROR IN AHLSETEV
AHLTMON	GTF TERMINATING ON ERROR CONDITION
AHLWTASK	DUMP OF GTF MODULE AHLWTASK
AHLWTO	DUMP BY/(OF) MODULE xxxxxxxx
AMSACT	COMPON=SAM,COMPID=27405,ISSUER=AMSACT,ERROR IN - SAM TERMINATION EXIT COMPON=SAM,COMPID=27405,ISSUER=AMSACT,ERROR IN - SAM USER AMSACU EXIT
AMSCOL	COMPON=SAM,COMPID-27405,ISSUER=AMSCOL,ABEND COMPON=SAM,COMPID-27405,ISSUER=AMSCOL,AMSCFREE... COMPON=SAM,COMPID-27405,ISSUER=AMSCOL,AMSCNTL... COMPON=SAM,COMPID-27405,ISSUER=AMSCOL,AMSCPREV... COMPON=SAM,COMPID-27405,ISSUER=AMSCOL,BAD ADDRESS... COMPON=SAM,COMPID-27405,ISSUER=AMSCOL,POINTER... COMPON=SAM,COMPID-27405,ISSUER=AMSCOL,WDS RECORD...
AMSUJI	COMPON=SAM,COMPID-27405,ISSUER=AMSUJI,ERROR IN -SAM INITIATION EXIT COMPON=SAM,COMPID-27405,ISSUER=AMSUJI,ERROR IN -SAM USER AMSUJU EXIT
ASRSERVR	COMPON=SYMREC,COMPID=SCASR,ISSUER=ASRSERVR...
AVFxx	COMPON=AVM,COMPID=SCAVM,ISSUER=...
CSVFRR	COMPID=SC1CJ,COMPON=CONTENTS SUPERVISOR.. - DUMP PRIOR TO QUEUE VERIFICATION
CSVFRR	COMPID=SC1CJ,COMPON=CONTENTS SUPERVISOR... - FAILURE DURING QUEUE VERIFICATION
CSVLLCRE	COMPON=PROGRAM-MANAGER-LNKLST...
CSVVFCRE	COMPON=PROGRAM-MANAGER...CSVVFCES... COMPON=PROGRAM-MANAGER...CSVVFCFR...
CSVVFGTE	COMPON=PROGRAM-MANAGER-VIRTUAL-FETCH...CSVVFGES... - CONTROL BLOCK FAILURE COMPON=PROGRAM-MANAGER-VIRTUAL-FETCH...CSVVFRR... - CONTROL BLOCK FAILURE COMPON=PROGRAM-MANAGER-VIRTUAL-FETCH...CVSSFRR... - JPQ FAILURE COMPON=PROGRAM-MANAGER-VIRTUAL-FETCH...CSVVFGES... - RETRY INHIBITED COMPON=PROGRAM-MANAGER-VIRTUAL-FETCH...CSVVFRR... - RETRY INHIBITED
CSVVFNDE	COMPON=PROGRAM-MANAGER-VIRTUAL-FETCH...CSVVFFES... - CONTROL BLOCK FAILURE COMPON=PROGRAM-MANAGER-VIRTUAL-FETCH...CSVVFFES... - JPQ FAILURE COMPON=PROGRAM-MANAGER-VIRTUAL-FETCH...CSVVFFES - RETRY INHIBITED

ERBCNFGC	COMPON=RMF,COMPID=27404,ISSUER=ERBCNFGC...
ERBCNFGF	COMPON=RMF,COMPID=27404,ISSUER=ERBCNFGF...
ERBCNFGG	COMPON=RMF,COMPID=27404,ISSUER=ERBCNFGG...
ERBMFDEA	COMPON=RMF,COMPID=27404,ISSUER=ERBMFDEA...
ERBMFEAR	COMPON=RMF,COMPID=27404,ISSUER=ERBMFEAR...
ERBMFEEQ	COMPON=RMF-ENQ EVENT...,COMPID=27404, ISSUER=ERBMFEEQ
ERBMFEVT	COMPON=RMF,COMPID=27404,ISSUER=ERBMFEVT...
ERBMFFUR	COMPON=RMF,COMPID=27404,ISSUER=ERBMFFUR...
ERBMFIDX	COMPON=RMF,COMPID=27404,ISSUER=ERBMFIDX...
ERBMFIQA	COMPON=RMF,COMPID=27404,ISSUER=ERBMFIQA...
ERBMFMFC	COMPON=RMF,COMPID=27404,ISSUER=ERBMFMFC...
ERBMFMLN	COMPON=RMF,COMPID=27404,ISSUER=ERBMFMLN...
ERBMFPVS	COMPON=RMF,COMPID=27404,ISSUER=ERBMFPVS...
ERBMFRES	COMPON=RMF,COMPID=27404,ISSUER=ERBMFRES...
ERBMFSDE	COMPON=RMF,COMPID=27404,ISSUER=ERBMFSDE...
ERBMFTMA	COMPON=RMF,COMPID=27404,ISSUER=ERBMFTMA...
ERBMFTRM	COMPON=RMF,COMPID=27404,ISSUER=ERBMFTRM...
ERB3GEEH	COMPON=RMF,COMPID=27404,ISSUER=ERB3GEEH...
ERB3GESA	COMPON=RMF,COMPID=27404,ISSUER=ERB3GESA...
ERB3GXMV	COMPON=RMF,COMPID=27404,ISSUER=ERB3GXMV...
ERB3RMFC	COMPON=RMF,COMPID=27404,ISSUER=ERB3RMFC...
HASPCKPT	DUMP OF JES2 DATA...
HASPFSSM	JES2 FSI ERROR...
HASPTERM	ABEND code AT hhhhhhhh (nnnnnn) + X'nnnn' cc-
or	HASPDUMP SUBSYS=ssss...
HASPRAS	title - supplied by the system operator
IARRRCV	COMPON=REAL STORAGE MANAGEMENT...
IATABN0	IAT3702 dspname (ddd) ABENDED/FAILED...
IATDMFR	ERROR IN IATSIDMO FOR SYSOUT DATA SET
	IAT1801 ERROR IN IADMDKT - IATYISR POSSIBLY LOST
IATIIII	none
IATIISB	IAT4830 IATIISB MASTERTASK ABEND
	IAT4831 IATIISB SUBTASK ABEND
IATLVLC	JES3 LOCATE SUBTASK ABEND
IATSIJS	IATSIJS JSESEXIT
IATSNDF	JES3 SNA FRR IATSNDF
IATSNLS	IATSNLS - ESTAE EXIT
IATSSCM	IATSSCM READ-END FAILURE
	SSICS ABEND 6FB
	SSICS ESTAE-IATSSCM
IATSSRE	COMPON=JES3 SUBSYS COMMUNIC...
IATSSXM	COMPON=JES3 SUBSYS COMMUNIC...
ICBRECRD	ICBRECRD RECORDING FAILED
ICBVPR00	ICB425I ABEND IN PROCESS - MSVC TASK (nnnnnnnn)
ICHRST00	ICHRST00 - RACF SVCS, ABEND CODE=xxx,...
ICHSEC02	RACF INITIALIZATION FAILURE
ICTMCS01	ICTMCS01, CRYPTOGRAPHY INITIALIZATION
ICTMKG00	ICTMKG00, KEY GENERATOR PROGRAM
ICTMKG01	ICTMKG01 HANDLE SYSIN MODULE
ICTMKM01	ICTMKM01, START CRYPTOGRAPHY COMMAND
ICTMKM04	ICTMKM04 - KEY MANAGER
ICTMSM07	ICTMSM07 - ICTMSM07 - CIPHER DUMP
ICTMSM08	ICTMSM07 - ICTMSM08 TRNSKEY DUMP
ICTMSM09	ICTMSM07 - ICTMSM09 EMK DUMP
IDACKRA1	VSAM CHECKPOINT (IDA0xxxx)...
	VSAM RESTART (IDA0xxxx)...
IDAICIA1	ISAM INTRFC, OPEN/CLOSE, IDA0192I/IDA0200S...
IDA019SB	IDA019SB:IDA121F7 - ABEND FROM BUILD IDACPA

IDA019S2	FIOD:IDA019S2 - ABEND FROM FIOD FRR
IDA0200T	IEC251I, VSAM GSR FORCE DLVRP DUMP DATA
IDA121A2	ABP:IDA121A2 - ABEND FROM ABP FRR
IDA121A3	ABP:IDA121A3 - ABEND FROM NORMAL END FRR
IDA121A4	ABP:IDA121A4 - ABEND FROM ABNORMAL END FRR
IEASMFSP	SMF ERRMOD=IEASMFSP, RECVRMOD=IEASMFSP
IEAVAD01	ERROR DURING SNAP,COMPON=SNAP...
	ERROR IN QMNGRIO PROCESSING,COMPON=SNAP...
	FAILURE DURING SNAP RECOVERY,COMPON=SNAP...
IEAVAR00	RCT DUMPING LSQA
IEAVECH0	variable title - supplied by caller
IEAVEGR	COMPON=SC,COMPID=SC1C5,ISSUER=IEAVEGR,TRQ
	-GLOBAL RECOVERY -- UNEXPECTED ERROR
	COMPON=SC,ISSUER=IEAVEGR,TRQ GLOBAL RECOVERY...
	-NORMAL PROCESSING
IEAVEMCR	IEAVEMCR - MEMORY CREATE ABNORMAL TERMINATION
IEAVEMDL	IEAVEMDL - MEMORY DELETE ABNORMAL TERMINATION
IEAVEMRQ	IEAVEMRQ UNEXPECTED ABEND
	IEAVEMRQ - UNEXPECTED ABEND WITH DISPATCHER LOCK
IEAVEPST	COMPON=TASK MGMT...POST EXIT ACTIVE...
	COMPON=TASK MGMT...POST FAILED...
	COMPON=TASK MGMT...XMPOST - NO ERRET...
	COMPON=TASK MGMT...XMPOST FAILED...
IEAVESAR	COMPON=SUPERVISOR CONTROL,COMPID=SC1C5...
IEAVETAC	COMPON=SYSTEM TRACE...A. S. CREATE
IEAVETAI	COMPON=SYSTEM TRACE...A. S. INIT
IEAVETCL	COMPON=SUPV CNTL,COMPID=SC1C5
IEAVETFC	COMPON=SYSTEM TRACE - FORMATTER...
IEAVETRR	COMPON=SYSTEM TRACE xxxxxxxx...
IEAVEWAT	COMPON=TASK MGMT...WAIT FAILED...
IEAVG600	COMPON=COMMTASK,COMPID=SC1CK,ISSUER=IEAVG600...
IEAVMFRR	COMPON=COMMTASK...ISSUER=IEAVMFRR...
IEAVN700	COMPON=COMMTASK...ISSUER=IEAVN700...
IEAVN701	COMPON=COMMTASK...ISSUER=IEAVN701...
IEAVRTI1	TIMER FRR DUMP
IEAVSPDM	ABEND=xxx,...COMPON=RECONFIG-SPDM...
IEAVSTAA	COMPON=COMMTASK...ISSUER=IEAVSTAA...
IEAVTABD	ABDUMP ERROR,COMPON=ABDUMP...
	ENQUEUE FAILURE IN ABDUMP,COMPON=ABDUMP...
IEAVTDSV	COMPON=5752-SC1CM...DUMPSRV...
	COMPON=5752-SC1CM...DISPLAY DUMP...
IEAVTGLB	ABEND IN IEAVTGLB
IEAVTJBN	ABEND IN IEAVTJBN
IEAVTLCL	ABEND IN IEAVTLCL
IEAVTRET	RECORD PERMANENT ERROR,COMP=RTM,
	-COMPID=SC1CM,ISSUER=IEAVTRET
	RECORD TEMPORARY ERROR,COMP=RTM,
	-COMPID=SC1CM,ISSUER=IEAVTRET
IEAVTRTE	no title
IEAVTRT2	IEAVTRT2 - UNRECOVERABLE ABEND FAILURE
IEAVTSEP	COMPON=SDUMP...POST DUMP EXIT...
IEAVTSLP	SLIP ID=xxxx
IEAVTSLR	ERROR IN IEAVTSLP
IEAVXPCR	ABEND=aaa,REASON=xxyy,GPRrr=zzzzzzzz,...
IEAVXSTK	ABEND=aaa,COMPON=PC/AUTH-PCLINK UNSTACK...
IEAVX600	WTO USER EXIT xxxxxxxx ABENDED...
IECVDPTH	COMPON=IOS-DYNAMIC PATHING...(estae)
	COMPON=IOS-DYNAMIC PATHING...(frr)
IECVERPL	IOS - IECVERPL ERROR
IECVGENA	COMPON=IOS-SIMULATED INTERRUPT...
	COMPON=IOS-UCBFLG FUNCTION...
IECVIOPM	COMPON=IOS-PATH VALIDATION...
IECVIOSI	COMPON=IOS-DYNAMIC PATHING INIT...
IECVPST	COMPON=IOS,COMID=SC1C3,ISSUER=IECVPST

IEECB800	COMPON=M S COMMANDS,COMPID=SC1B8...
IEECB860	COMPON=CMND-ESTAE,...DUMPDS...
IEECB862	COMPON=M S CMNDS,COMPID=SC1B8,...
IEECB866	variable title - supplied by the system operator
IEECB906	IEECB906 SLIP ESTAE DUMP
IEECB910	COMPID=5752-SC1CM...DISPLAY DUMP...
IEECB913	COMPON=SETSMF COMMAND...
IEECB914	IEECB914 SLIP TSO COMM RTN ESTAE DUMP
IEECB923	COMPID=5752-SC1CM...DUMPDS FAILED...
IEECB926	COMPID=5752-SC1CM...DUMPDS PROCESSOR...
IEECB927	COMPID=SC1CZ...ABEND=(xxx)...
IEEMB803	COMPON=SYSLOG-INIT,COMPID=SC1B8...
IEEMB804	COMPON=SYSLOG-SVC,COMPID=SC1B8...
IEEMB806	COMPON=SYSLOG-ESTAE,COMPID=SC1B8...
IEEMB812	STORAGE DUMP TAKEN AT ENTRY TO IEEMB812 ESTAE
IEEMB816	COMPID=SC1B8,xxx ABEND IN MASTER TR...
IEEMB825	COMPON=SMF CONTROL TASK...
IEEMB827	COMPON=SMF INITIALIZATION...
IEEMB830	SMF ABEND, ERROR=xxxxxxxx,...
IEEMB834	SMF ABENDED, ERRORMOD=IEEMB834,...
IEEMB835	COMPON=SET SMF COMMAND...
IEEMB836	ABEND IN SMF INTERVAL PROCESSING - ROUTINE...
IEEMB839	SMF TIMER - IEEMB839
IEEMB860	COMPON=MSTR-REGION,COMPID=SC1B8...
IEEMB881	COMPON=M S CMNDS,COMPID=SC1B8,...
IEEMB883	COMPON=M S CMNDS,COMPID=SC1B8,...
IEEMB887	COMPON=MS CMNDS...PARSER...
IEEMPDM	COMPON=RECONFIGURATION-DISPLAY M...
IEEMPS03	IEEMPS03 - DUMP OF MAIN WORKAREA
IEESB670	COMPON=STS-REC,COMPID=SC1B8...
IEESB670	COMPON=JSS-REC,COMPID=SC1B8...
IEEVCHPF	COMPON=RECONFIG,...ISSUER=IEEVCHPF...
IEEVCONF	COMPID=SC1CZ,...ABEND=(xxx)...
IEEVCPR	COMPON=RECONFIG,...CONFIG CPU...
IEEVIOSD	COMPON=RECONFIG,...ISSUER=IEEVIOSD...
IEEVIPL	COMPON=MSTR-BASE,COMPID=SC1B8...
IEEVLDWT	IEEVLDWT ERROR
IEEVPTH	COMPON=RECONFIG(SC1CZ)...(VARY PATH)...
IEEVPTHR	COMPON=RECONFIG(SC1CZ)...ABEND(xxx)...
IEEVRDPM	COMPID=SC1CZ,...ABEND=(xxx)
IEEVRSCN	COMPID=SC1CZ,...ABEND=(xxx)
IEEVSTEE	COMPON=RECONFIG,...ISSUER=IEEVSTEE
IEEVSTPE	COMPON=RECONFIG,...ISSUER=IEEVSTPE
IEEVWAIT	COMPON=MSTR-WAIT...ISSUER=IEEVWAIT...
IEE24110	COMPON=DIDOCS-D U,,ALLOC PROC...
IEE5103D	COMPON=SVC34,COMPID=SC1B8...

IEFAB4ED	ISSUER=IEFAB4ED,ERRCSECT=csect,COMPID=5752-SC1B4,...
IEFAB4E6	ISSUER=IEFAB4E6,ERRCSECT=csect,COMPID=5752-SC1B4,...
IEFAB4GA	ISSUER=IEFAB4GA,ERRCSECT=csect,COMPID=5752-SC1B4,...
IEFAB4SF	ISSUER=IEFAB4SF,ERRCSECT=csect,COMPID=5752-SC1B4,...
IEFCMAUT	COMMON AUTHORIZATION CHECK ROUTINE ...
IEFDB440	ISSUER=IEFDB440,ERRCSECT=csect,COMPID=5752-SC1B4,...
IEFENFFX	ENF ABEND ERRORMOD=IEFENFFX
IEFENFNM	ENF ABEND ERRORMOD=IEFENFNM
IEFENFWT	EVENT NOTIFICATION FACILITY ERROR
IEFIB620	ERROR IN INITIATOR,...
IEFIB645	SWA CREATE
IEFISEXR	RESOURCE MANAGER
IEFJRASP	ERROR IN BROADCAST FUNCTION,ABEND=aaa,...
IEFJSBLD	ERROR IN SUBSYSTEM SERVICE RTN,COMPON=INIT-SSI
IEFJSIN2	ERROR IN SUBSYSTEM INITIALIZATION,COMPON=INIT-SSI...
IEFNB9CR	ABEND=aaa,COMPON=CONVERTER... RESTART INTERRUPT IN CONVERTER**IEFNB9CR**
IEFNB9IR	ABEND=aaa,COMPON=INTERPRETER... RESTART INTERRUPT IN INTERPRETER**IEFNB9IR**
IEFSJCNL	SCHEDULER JCL FACILITY...

IEFXB609	ERROR DURING RESTART PROCESSING - IEFXB609
IFADSMF	COMPON=DISPLAY SMF COMMAND...
IFAEASI	ABEND IN SMF INTERNAL PROCESSING...
IFAPCWTR	SMF ABEND,ERRMOD=IFAPCWTR...
IFASSMF	COMPON=SETSMF COMMAND...
IFATSMF	COMPON=SET SMF COMMAND...
IFG0RR0A	IEC999I IFG0RR0A,error-module,jobname,...
IFG0RR0E	IEC999I IFG0RR0A,error-module,jobname,...
IFG0RR0F	IEC999I IFG0RR0A,IFG0RR0F,jobname,stepname...
IFG0TC0A	IEC999I IFG0TC0A/4A/5A,subroutine,jobname...
IGCT0018	IGCT0018,jobname,stepname
IGCT002D	IGCT002D,jobname,stepname
IGCT002E	IGCT002E,jobname,stepname
IGCT0021	IGCT0021,jobname,stepname
IGCT005C	IGCT005C,jobname,stepname
IGCT005G	IGCT005G,jobname,stepname
IGCT006H	IGCT006H,jobname,stepname,procstepname,744
IGCT0069	IGCT0069,jobname,stepname,
IGCT010E	IGCT010E,jobname,stepname
IGCT105C	IGCT105C,jobname,stepname
IGCT1081	IGCT1081,jobname,stepname
IGC0001F	no title
IGC0002F	IGC0002F CATLG CTLR 3
IGC121	ABP:IGC121 - ABEND FROM SIOD FRR
IGFDE1	COMPON=DDR,COMPID=BB1CS...
IGG0CLA9	JOB=jobname hh:mm:ss yy.ddd DUMP BY IGG0CLA9...
IGG0CLCA	SDUMP - IGG0CLCA CVOL CATALOG MANAGEMENT
IGG0CLCB	no title
IGG0CLCD	SDUMP - IGG0CLCD - CVOL CATALOG MANAGEMENT
IGUDSP02 or IGUDSP03	COMPON=DEVSERV PATHS COMMAND,ISSUER=IGUDSP02 or IGUDSP03,COMPID=28463
IGVFVIRT	COMPON=VSM=IGVFVIRT,COMPID=SC1CH...
IGVGCAS	COMPON=VSM,COMPID=SC1CH,ISSUER=IGVGCAS...
IGVGRRGN	COMPON=VSM,COMPID=SC1CH,ISSUER=IGVGRRGN...
IGVGVRGN	COMPON=VSM,COMPID=SC1CH,ISSUER=IGVGVRGN...
IGVLOCP	COMPON=VSM-VSMLOC,COMPID=SC1CH...
IGVRCP	COMPON=VSM-CELLPOOL...
IGVRSRTN	COMPON=VSM-GETMAIN/FREEMAIN...
IGVRVSM	COMPON=VSM,COMPID=SC1CH,ISSUER=IGVRVSM...
IGVSLIST	COMPON=VSM-VSMLIST,COMPID=SC1CH...
IGVSTSKI	COMPON=VSM,COMPID=SC1CH,ISSUER=IGVSTSKI...
IGVSTSKT	COMPON=VSM,COMPID=SC1CH,ISSUER=IGVQSPET...
IKJCT460	TSO OUTPUT CP ESTAE
IKJEFLGB	TSOLOGON ESTAI
IKJEFLGM	IKJEFLGM REQUEST
IKJEFLS	TSOLOGON ESTAE
IKJEFT05	TSO SDUMP FROM IKJEFT05 ...
IKTCAS52	TCAS DUMP
IKTLTERM	IKTLTERM - I/O ERROR
IOSPURGA	COMPON=IOS,COMPID=SC1C3,...
IOSRACRW	COMPON=IOS,COMPID=SC1C3,...
IOSRCDEV	COMPON=IOS-IOS CLEAR DEVICE...
IOSRCHDR	COMPON=IOS-IOS CHANNEL PATH...
IOSRCHDS	COMPON=IOS-IOS CHANNEL PATH...
IOSRCHPR	COMPON=IOS,COMPID=SC1C3,...

IOSRDBOX	COMPON=IOS,COMPID=SC1C3,...
IOSRFDEV	COMPON=IOS-IOS FORCE DEVICE...
IOSRHDET	COMPON=IOS,COMPID=SC1C3,...
IOSRHREC	COMPON=IOS,COMPID=SC1C3,...
IOSRMIHP	COMPON=IOS,COMPID=SC1C3,...
IOSRMIHR	COMPON=IOS,COMPID=SC1C3,...
IOSRMIHT	COMPON=IOS...MISSING INTERRUPT HANDLER...
IOSRRSV	COMPON=IOS,COMPID=SC1C3,...
IOSRSAIO	COMPON=IOS(SC1C3),STAND-ALONE...
IOSRSCH	COMPON=IOS,COMPID=SC1C3,...
IOSRSLH	COMPON=IOS-SUBCHANNEL LOGOUT...
IOSVDAVV	COMPON=IOS-DASD VOLUME...
IOSVDPDR	COMPON=IOS-DYNAMIC PATHING DRIVER...
IOSVFCHP	COMPON=IOS,COMPID=SC1C3,...
IOSVHSCH	COMPON=IOS,COMPID=SC1C3,...
IOSVIPID	COMPON=IOS,COMPID=SC1C3,...
IOSVIRBA	COMPON=IOS,COMPID=SC1C3,...
IOSVIRBD	COMPON=IOS,COMPID=SC1C3,...
IOSVIRBH	COMPON=IOS,COMPID=SC1C3,...
IOSVIRBN	COMPON=IOS,COMPID=SC1C3,...
IOSVIRBU	COMPON=IOS,COMPID=SC1C3,...
IOSVLEVL	COMPON=IOS,COMPID=SC1C3,...
IOSVMSCH	COMPON=IOS,COMPID=SC1C3,...
IOSVMSCQ	COMPON=IOS,COMPID=SC1C3,...
IOSVPRVT	COMPON=IOS,COMPID=SC1C3,...
IOSVRSTS	COMPON=IOS-RESTART SUPPORT...
IOSVRSUM	COMPON=IOS,COMPID=SC1C3,...
IOSVSCHR	COMPON=IOS-SUBCHANNEL REDRIVE...
IOSVSHUP	COMPON=IOS-SHARED UP SERVICE...
IOSVSLIH	COMPON=IOS,COMPID=SC1C3,...
IOSVSMGR	COMPON=IOS-IOS STORAGE MANAGER...
IOSVSSCH	COMPON=IOS,COMPID=SC1C3,...
IOSVSSCQ	COMPON=IOS,COMPID=SC1C3,...
IOSVSTSC	COMPON=IOS,COMPID=SC1C3,...
IOSVSTSQ	COMPON=IOS,COMPID=SC1C3,...
IOSVSWAP	COMPON=IOS,COMPID=SC1C3,...
IOSVURDT	COMPON=IOS-UNCONDITIONAL RESERVE...
IOSVURVL	COMPON=IOS-UNCONDITIONAL RESERVE...
IOSVVARY	COMPON=IOS,COMPID=SC1C3,...
IRARMERR	SRM RECOVERY ENTERED,COMPON=SRM
IRARMSRV	SRM - IRARMSRV 55F ABEND DURING XMPOST
ISGBERCV	COMPON=GRS-RING-PROC...ISSUER=ISGBERCV
ISGBFRCV	COMPON=GRS-RING-PROC...ISSUER=ISGBFRCV
ISGCRCV	REQUESTOR=xxx,ISSUER=ISGCRV...
ISGCRET0	COMPON=GRS-COMMANDS...ISSUER=ISGCRET0...
ISGCRET1	COMPON=GRS-COMMANDS...ISSUER=ISGCRET1...
ISGDSNAP	COMPON=GRS...ISSUER=ISGDSNRV
ISGGFRR0	COMPON=GRS...ISSUER=ISGGFRR0
ISGJENF0	COMPON=GRS-CTC-DRIVER...ISSUER=ISGJENF0
ISGJRCV	COMPON=GRS-CTC-DRIVER...ISSUER=ISGJRCV
ISGQSCNR	COMPON=GRS-QUEUE SCANNING...ISSUER=ISGQSCNR
ISGSMI	COMPON=GRS,COMPID=SCSDS,ISSUER=ISGSMIFR
ISTAPCES	ISTAPCES - ACF/VTAM PSS ESTAE...
ISTAPCFR	ISTAPCFR - ACF/VTAM PSS FRR...
ISTAPCMT	ISTAPCMT - ACF/VTAM ABEND IN MEMORY TERM
ISTATM00	ISTATM00 - ACF/VTAM TERMINATION...
ISTINCST	ISTINCST - ACF/VTAM STAE EXIT...
ISTORMMG	ISTORMMG - ACF/VTAM FRR DUMP

```
ITVRD    COMPON=DATA IN VIRTUAL,COMPID=SCDIV,ISSUER=ITVRD...
ITVRG    COMPON=DATA IN VIRTUAL,COMPID=SCDIV,ISSUER=ITVRG...
ITVRK    COMPON=DATA IN VIRTUAL,COMPID=SCDIV,ISSUER=ITVRK...
ITVRM    COMPON=DATA IN VIRTUAL,COMPID=SCDIV,ISSUER=ITVRM,
         -WITH INVALID DRA
         COMPON=DATA IN VIRTUAL,COMPID=SCDIV,ISSUER=ITVRM,
         -WITH VALID DRA
ITVRR    COMPON=DATA IN VIRTUAL,COMPID=SCDIV,ISSUER=ITVRR,
         -WITH INVALID DRA
         COMPON=DATA IN VIRTUAL,COMPID=SCDIV,ISSUER=ITVRR,
         -WITH VALID DRA
```

Appendix C. Abbreviations

ABP	- Actual block processor
ACA	- ASM control area
ACB	- Access method control block
ACE	- ASM control element
ACF/TCAM	- Advanced Communications Function for TCAM
ACF/VTAM	- Advanced Communications Function for VTAM
ACP	- Automatic command processing
ACR	- Alternate CPU recovery
ACT	- Account control table
ADA	- Automatic data area
ADB	- Allocation descriptor block
AFQ	- Available frame queue
AIA	- ASM I/O request area
ALCWA	- Allocation work area
ALPAQ	- Active link pack area queue
AMB	- Access method block
AMBL	- AMB list
AMCBS	- Access method control block structure
AMDSB	- Access method data statistics block
AP	- Attached processor
APF	- Authorized program facility
APG	- Automatic priority group
AQAT	- Address queue anchor table
ASCB	- Address space control block
ASID	- Address space identification
ASM	- Auxiliary storage manager
ASMHD	- Auxiliary storage management header
ASMVT	- ASM vector table
ASPCT	- Auxiliary storage page correspondence table
ASST	- Address space sector table
ASTE	- Address space second table entry
ASVT	- Address space vector table
ASXB	- Address space extension block
AT	- Authorization table
ATA	- ASM tracking area
AVT	- TCAM address vector table
AX	- Authorization index
AXAT	- Authorization index allocation table
BASEA	- Master scheduler resident data area
BPCB	- Buffer pool control block
BUFC	- Buffer control area
CA	- Control area or channel adapter
CAXWA	- Catalog ACB extended work area
CCA	- Catalog communications area
CCW	- Channel command word
CDE	- Contents directory entry
CEPL	- Command ESTAE parameter list
CHAP	- Change priority
CI	- Control interval
CIDF	- Control interval definition field
CKB	- Checkpoint buffer
CMB	- Console message buffer
CML	- Cross memory lock
CMS	- Cross memory services or catalog management services

CMSWA - CMS work area
CP - Central processor
CPA - Channel program area
CPAB - Cell pool anchor block
CPB - Channel program block
CPPL - Command processor parameter list
CPUID - CPU identification
CQE - Console queue element
CRA - Component recovery area
CRB - Command request block
CRW - Channel report word
CRWA - Command recovery work area
CSA - Common service area
CSCB - Command scheduling control block
CSD - Common system data area
CTGPL - Catalog parameter list
CVT - Communications vector table
CXSA - Communications extended save area

DADSM - Direct access device space management
DAE - Dump analysis and elimination
DAIT - Display allocation index table
DALT - Display allocation lookup table
DAM - Direct access method
DAT - Dynamic address translation
DAVV - Direct access volume verification
DCB - Data control block
DCM - Display control module
DCT - Device control table
DDR - Dynamic device reconfiguration
DDRCOM - Dynamic device reconfiguration communication table
DE - Directory entry
DEB - Data extent block
DECB - Data event control block
DEPI - SDUMP ESTAE parameter list
DFE - Double free element
DIDOCS - Device independent display operators console support
DIE - Disable interrupt exit
DIR - Deferred incident record
DMDT - Domain descriptor table
DMVT - Domain vector table
DPL - DEQ purge list
DQE - Descriptor queue element
DRQ - Data ready queue
DSAB - Data set association block
DSCB - Data set control block
DSPCT - Data set page correspondence table
DSPL - Dump sort parameter list
DVT - Destination vector table or display allocation vector table

ECB - Event control block
ECC - Error checking and correction
ECT - Environment control table
EDB - Extent descriptor table
EDL - Eligible device list
EDT - Eligible device table
EED - Extended error descriptor
EIL - Event indication list
EIP - EXCP intercept processor
EMS - Emergency signal
ENF - Event notification facility
ENFPM - ENF event parameter list
EOA - End of address
EOE - End of extent
EOV - End of volume
EP - Emulator program
EPATH - Error path (recovery audit trail area)
EPS - External page storage
ERP - Error recovery procedures

ERPIB - Error recovery procedures interface block
ESTAE - Extended STAE
ESTAI - Extended STAI
ET - Entry table
ETE - Entry table entry
ETIB - Entry table information block
ETIX - ETIB extension
EVNT - Event table
EWA - Common ERP work area
EX - Entry table index

FBQE - Free block queue element
FCB - Function control block
FCT - Function control table
FDB - Feedback data block
FETWK - Fetch work area
FIFO - First in first out
FLIH - First level interrupt handler
FMCB - ACF/VTAM function management control block
FOE - Fixed ownership element
FOT - Fixed ownership table
FQE - Free queue element
FRR - Functional recovery routine
FRRS - FRR stack
FSB - Feedback status block
FVT - Field vector table

GCB - Global resource serialization CTC-driver request block
GCC - Global resource serialization CTC-driver control card table
GCL - Global resource serialization CTC-driver link control block
GCP - Global resource serialization CTC-driver buffer prefix
GCQ - Global resource serialization CTC-driver queueing element
GCT - Global resource serialization CTC-driver branch table
GCV - Global resource serialization CTC-driver vector table
GCX - Global resource serialization CTC-driver extract table
GDA - Global data area
GPR - General purpose register
GQHT - Global queue hash table
GRS - Global resource serialization
GSMQ - Global service manager queue
GSPL - Global system priority list
GSR - Global shared resource
GTF - Generalized trace facility
GVT - Global resource serialization vector table
GVTX - GVT extension

HIR - Hardware instruction retry

IC - Instruction counter
ICNCB - Intermediate controller node control block
IHSA - Interrupt handler save area
ILC/CC - Instruction length condition code
IOB - Input output block
IOMB - I/O management block
IOQE - I/O queue element
IORB - I/O request block
IOS - I/O supervisor
IOSB - I/O supervisor block
IOT - I/O table
IOWA - I/O work area
IPC - Inter-processor communication
IPCS - Interactive problem control system
IPL - Initial program load
IPS - Installation performance specifications
IPTE - Invalidate page table entry
IQE - Interrupt queue element
IRB - Interrupt request block
IRT - IOS recovery table
ISAM - Indexed sequential access method

JCL - Job control language
JCT - Job control table
JDT - JCL definition table
JDVT - JCL definition vector table
JES - Job Entry Subsystem
JES2 NJE - JES2 Network Job Entry (Program Product)
JESCT - JES control table
JFCB - Job file control block
JFCBX - Job file control block extension
JIX - Job queue index
JOE - Job output element
JOT - Job output table
JPQ - Job pack queue
JQE - Job queue element
JSCB - Job step control block
JSEL - Job scheduling entry list
JSXL - Job scheduling exit list

KSDS - Key sequence data set

LCB - TP line control block
LCCA - Logical configuration communication area
LCCAVT - LCCA vector table
LCPB - Logical channel program block
LCT - Linkage control table
LDA - Local data area
LG - Logical group
LGF - Line group block
LGCB - Logical group control block
LGE - Logical group entry
LGN - Logical group number
LGVT - Logical group vector table
LGVTE - Logical group vector table entry
LIFO - Last in first out
LIT - Lock interface table
LLA - LNKLST lookaside
LLCB - LLA control block
LLDE - LLA directory entry
LLE - Load list element
LLQ - Load list queue
LLT - LNKLST table
LNKLST - SYS1.LINKLIB (and concatenated data sets)
LPA - Link pack area or latent parameter area
LPALST - SYS1.LPALIB (and concatenated data sets)
LPAT - Link pack area (LPA) table
LPDE - Link pack directory entry
LPID - Logical page identifier
LPME - Logical to physical mapping entry (or) logical page mapping entry
LQHT - Local queue hash table
LRB - Logrec buffer
LSID - Logical slot ID
LSMQ - Local service manager queue
LSPL - Local service priority list
LSQA - Local system queue area
LT - Linkage table
LTE - Linkage table entry
LUB - NCP logical unit block
LWA - Logon work area
LWK - Local work area
LX - Linkage index
LXAT - Linkage index allocation table

MCH - Machine check handler
MCIC - Machine check interrupt code
MCP - Message control program
MCS - Multiple console support
MFA - Malfunction alert
MH - Message handler
MIH - Missing interrupt handler

MLPA - Modified link pack area
MP - Multiprocessor
MPF - Message processing facility
MPFT - MPF table
MPST - Memory process scheduling table
MRB - Message request block
MSFAB - MSSFCALL SVC attention block
MSFCB - MSSFCALL SVC control block
MSS - Mass storage subsystem
MSSC - Mass storage system communicator
MSSF - Monitoring and system support facility
MSVC - Mass storage volume control
MVS - Multiple Virtual Storage
MVS/XA - MVS/Extended Architecture
MWA - Module work area

NCB - ACF/VTAM node control block
NCP - Network Control Program
NIP - Nucleus initialization program

OCR - Output control record
OCT - Output control table
OPWA - Open work area
ORE - Operator reply element
OUCB - SRM-user control block
OUSB - SRM-user swappable block
OUXB - SRM-user extension block

PAB - Process anchor block
PART - Paging activity reference table
PARTE - PART entry
PAT - Page allocation table
PC/AUTH - Program call/authorization
PCB - Page control block
PCCA - Physical configuration communication area
PCCAVT - PCCA vector table
PCCB - Private catalog control block
PCCW - Paging channel command work area
PCE - Processor control element
PCRA - Program call recovery area
PDDB - Peripheral data definition block
PDS - Partitioned data set
PEL - Parameter element
PEP - Partitioned emulator program
PER - Program event recording
PEXB - Pool extent block
PFT - Page frame table
PFTE - Page frame table entry
PGT - Page table
PGTE - Page table entry
PICA - Program interrupt control area
PIE - Program interrupt element
PIT - Partition information table
PIU - Physical information unit
PLH - Place holder
PLPA - Pageable link pack area
PLPAD - PLPA directory
PQCB - Placeholder queue control block
PRB - Program request block
PSA - Prefixed save area
PSCB - Protected step control block
PSS - Process scheduling service
PST - Process scheduling table
PSW - Program status word
PTLB - Purge translation lookaside buffer
PVT - Paging vector table
PWKA - Paging work area

QAB - Queue anchor block
QCB - Queue control block
QEL - Queue element
QFPL - ENQ/DEQ FRR parameter list
QFPL1 - Queue scanning services FRR parameter list
QHT - Queue hash table
QWA - Queue work area
QWB - Queue work block
QXB - Queue extension block

RAB - RSM address space block
RACF - Resource Access Control Facility (Program Product)
RB - Request block
RBA - Relative byte address
RBCB - Recording buffers control block
RBN - Real block number
RCA - RSM recovery communication area
RCB - Recording control block
RCE - RSM control and enumeration area
RCT - Region control task
RD - Region descriptor
RDCM - Resident display control module
RDF - Record definition field
RDT - Resource definition table
RDTE - Resource definition table entry
REPL - Ring processing ESTAE parameter list
RIB - Resource information block
RIBE - Resource information block extent
RIM - Resource initialization module
RIT - RSM internal table
RJE - Remote job entry
RMCT - Resource manager control table
RMF - Resource Measurement Facility (Program Product)
RMS - Recovery management support
RNLE - Resource name list entry
RPB - RSM pool block
RPH - Request parameter header
RPL - Request parameter list
RPT - Request pool table
RQA - Resource queue area
RQE - Request queue element
RRPA - Recovery routine parameter area
RSA - Ring processing system authority message
RSAIRCD - Ring processing information record
RSC - Ring status change parameter list
RSL - Ring processing system link block
RSM - Real storage manager
RST - Ring processing status table
RSV - Ring processing system vector table
RTAM - Remote terminal access method
RTCA - Recovery termination control area
RTCT - Recovery termination control table
RTM - Recovery termination manager

S/A - Stand-alone (dump program)
SAHT - System/ASID hash table
SAM - Sequential access method
SART - Swap activity reference table
SAST - Subsystem allocation sequence table
SAT - Swap allocation table or system authorization table
SCCW - Swap channel control work area
SCSW - Subchannel status word
SCT - Step control table
SDWA - System diagnostic work area
SFT - System function table or swap function table
SGT - Segment table
SGTE - Segment table entry
SHAS - Subsystem hash table
SIC - System initiated cancel

SIGP - Signal processor instruction
SIOT - Step I/O table
SJF - Scheduler JCL facility
SLH - Subchannel logout handler
SLIH - Second level interrupt handler
SLIP - Serviceability level indication processing
SLT - System linkage table
SMF - System management facility
SMPL - Storage management parameter list entry
SMS - Storage management services
SNA - System Network Architecture
SPQA - Subpool queue anchor
SPQE - Subpool queue element
SPT - Subpool table
SPTT - Subpool translation table
SQA - System queue area
SQHT - Size queue anchor table
SRB - Service request block
SRM - System resources manager
SRR - Serially reusable resource
SRRA - Service routine recovery area
SSCP - System services control point
SSCVT - Subsystem communications vector table
SSI - Subsystem interface
SSIB - Subsystem identification block
SSOB - Subsystem options block
SSQ - SVRB suspend queue
SSRB - Suspended service request block
SSVT - Subsystem vector table
STAE - Specify task abnormal exit
STAI - Subtask abend intercept
STC - Start task control
STCB - Subtask control block
STKE - Stack element
STOR - Segment table origin register
SVC - Supervisor call
SVRB - Supervisor request block
SVT - Supervisor vector table
SWA - Scheduler work area
SWB - Scheduler work block

TCAM - Telecommunications Access Method
TCB - Task control block
TDCM - Pageable display control module
TEA - Translation exception address
TH - Transmission header
TIOC - Terminal I/O coordinator
TIOT - Task input/output table
TLB - Translation lookaside buffer
TMC - Task mode controller
TMP - Terminal monitor program
TOD - Time of day
TSB - Terminal status block
TSO - Time Sharing Option
TTE - Trace table entry

UADS - User attribute data sets
UCB - Unit control block
UCM - Unit control module
UCME - Unit control module entry
UIC - Unreferenced interval count
UPT - User profile table

VBN - Virtual block number
VBP - Virtual block processor
VDSCB - Virtual data set control block
VFCB - Virtual fetch control block
VFHE - Virtual fetch hash entry
VFPM - Virtual fetch parameter list
VFVT - Virtual fetch vector table

VFWK - Virtual fetch work area
VIO - Virtual I/O
VSAM - Virtual storage access method
VSM - Virtual storage management
VTAM - Virtual Telecommunications Access Method
VTOC - Volume table of contents
VUT - Volume unload table

WAST - Workload activity specification table
WMST - Workload manager specification table
WQE - Write queue element
WSC - Wait state code
WTQE - Wait queue element

XL - Extent list
XMD - Cross memory directory
XPTE - External page table entry
XRBN - Extended real block number
XSB - Extended status block

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Numerics

MVS/Extended Architecture Diagnostic Techniques

LY28-1199-4

S370-37

Printed in U.S.A.

MVS/Extended Architecture
Diagnostic Techniques

READER'S
COMMENT
FORM

LY28-1199-4

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note: ***Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.***

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

_____

_____

_____

_____

What is your occupation? _____

How do you use this publication? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

MVS/Extended Architecture Diagnostic Techniques

LY28-1199-4

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®

MVS/Extended Architecture
Diagnostic Techniques

READER'S
COMMENT
FORM

LY28-1199-4

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note: ***Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.***

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

_____

_____

_____

_____

What is your occupation? _____

How do you use this publication? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

MVS/Extended Architecture Diagnostic Techniques

LY28-1199-4

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®

MVS/Extended Architecture
Diagnostic Techniques

READER'S
COMMENT
FORM

LY28-1199-4

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note: ***Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.***

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

_____

_____

_____

_____

What is your occupation? _____

How do you use this publication? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

MVS/Extended Architecture Diagnostic Techniques

LY28-1199-4

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®

LY28-1199-04