File No. S360-20
Order No. GC28-6670-4

OS

IBM Systems Reference Library

IBM System/360 Operating System:

Programmer's Guide to Debugging

This publication describes the major debugging facilities provided with the System/360 Operating System for the assembler language programmer:

- Abnormal termination and snapshot dumps.
- Indicative dumps.
- Storage image dumps.
- Stand-alone hexadecimal dumps.

The text explains those aspects of system control pertinent to debugging, tells what information each debugging facility offers, and outlines procedures for invoking and interpreting dumps issued at the three operating system levels: PCP, MFT, and MVT.

Debugging facilities inherent in higher languages and additional aids open to the assembler language programmer are discussed in other SRL publications.

Information in this publication for TSO is for planning purposes until that item is available.

Fifth Edition (January, 1971)

This is a major revision of, and obsoletes C28-6670-3. For a description of the major changes see page 7. All changes to the text, and small changes to illustrations, are indicated by a vertical line to the left of the change. New figures have been added. Changed and added illustrations are denoted by the symbol • to the left of the caption.

This edition, with Technical Newsletter GN28-2457, applies to release 20.1 of IBM System/360 Operating System and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/360 Newsletter, Order No. GN20-0360, for the editions that are applicable and current.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of the publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Systems Publications, Department D58, P.O. Box 390, Poughkeepsie, N. Y. 12602

Preface

This publication is intended to help you use the debugging facilities provided with the IBM System/360 Operating System. To fulfill this purpose, the publication is divided into two sections: "Section 1: Operating System Concepts," and "Section 2: Interpreting Dumps." You should read the introduction to familiarize yourself with the debugging facilities before proceeding to Section 1.

Section 1 deals with internal aspects of the operating system that you should know to use the debugging facilities efficiently. A working knowledge of this information will provide you with the means of determining the status of the system at the time of the failure, and the course of events which led up to that failure. It includes information from other System Reference Library publications, Program Logic Manuals, and Installation Guides. You should be familiar with the information covered in Section 1 before attempting to use Section 2.

Section 2 includes instructions for invoking, reading, and interpreting dumps issued by systems with PCP, MFT, and MVT. It presents an after-the-fact look at a dump. You've put in a run, it failed, and you now have a dump before you. Where do you start; what do you look at; what does it all mean? The section begins with a general debugging procedure, followed by topics dealing with each type of dump. Each topic tells how to invoke a particular dump, what information the dump contains, and how to use this information in following the debugging procedure. The material in Section 2 is intended to aid you in interpreting dumps and isolating errors.

Before reading this publication, you should have a general knowledge of operating system features and concepts as presented in the prerequisite publications. Occasionally, the text refers you to other publications for detailed discussions beyond the scope of this book.

For information on debugging facilities provided within higher languages, consult the programmers' guides associated with the respective languages. Other System/360 Operating System publications, such as TESTRAN and Messages and Codes, describe additional debugging aids provided for the assembler language programmer.

Prerequisite Publications

IBM System/360: Principles of Operation, GA22-6821

IBM System/360 Operating System:

Concepts and Facilities, GC28-6535

Supervisor and Data Management Services, GC28-6646

Reference Publications

IBM System/360 Operating System:

System Control Blocks, GC28-6628

Messages and Codes, GC28-6631

Supervisor and Data Management Macro Instructions, GC28-6647

System Programmer's Guide, GC28-6550

Service Aids, GC28-7619

TCAM Programmer's Guide and Reference, GC30-2024.

TCAM Serviceability Aids, GY30-2027.

TCAM, GY30-2029.

Contents

Illustrations

Figures

Tables

Diagram

Summary of Major Changes--Release 20.1

Item	Description	Areas Affected
TCAM	A brief description of TCAM debugging aids and a new SVC.	SECTION 2: ABEND/SNAP Dump (PCP and MFT) ABEND/SNAP Dump (MVT) APPENDIX A APPENDIX H
TSO	The addition of new SVCs and a summary of the control blocks formatted by IMDPRDMP.	SECTION 2: TSO Control Blocks APPENDIX A

Summary of Major Changes--Release 20

Item	Description	Areas Affected
IMDPRDMP	IMDPRDMP is used instead of IEAPRINT to print MFT and MVT dumps.	"Guide to Using a Storage Image Dump"
TSO	New SVCs in Appendix A. This information is for planning purposes only.	Appendix A.

Introduction

Debugging is possibly the most important aspect of programming. Few programmers, especially those involved in control program modification, ever produce a perfect solution in one run; abnormal termination is inevitable and must be prepared for.

Program debugging in an operating system environment is made more difficult by the large volume of control information, the presence of control program routines, and the changing contents of main storage. Frequently, a large part of debugging lies in determining what state the system was in when the error occurred and which essential information was obscured.

To debug problem programs efficiently, you should be familiar with the system control information reflected in dumps. This control information, in the form of control blocks and traces, tells you what has happened up to the point of error and where key information related to the program is located.

This book is therefore designed to:

- Help you prepare proper dump data set definitions.
- Provide an insight into the IBM System/360 Operating System and its complex aspects of task management, storage supervisor, control blocks, and debugging aids.
- Give you a starting point, an approach, and a method of debugging.

The IBM System/360 Operating System provides extensive degugging facilities to aid you in locating errors and determining the system state quickly. Some debugging aids, such as console messages, provide limited information that may not always help you identify the error. This manual discusses those debugging facilities that provide you with the most extensive information:

a. Abnormal termination (ABEND) and snapshot (SNAP) dumps.
b. Indicative dumps.
c. Storage image dumps.
d. Stand-alone hexadecimal dumps.

ABEND and SNAP Dumps are invoked by ABEND and SNAP macro instructions, respectively. They are grouped in a single category because they provide identical information. In addition to a hexadecimal dump of main storage, they can contain conveniently edited control information and displays of the operating system nucleus and trace table.

Indicative dumps contain control information useful in isolating the instruction that caused an abnormal end of task situation. The information is similar to that given in an ABEND/SNAP dump, but does not include a dump of main storage.

Storage image dumps are taken by the system dump facility at the time of a system failure. The dump is written to the SYS1.DUMP data set. For a PCP dump, use the IEAPRINT print dump program to print the SYS1.DUMP data set. The dump consists of a first page, containing edited control information, followed by a dump of the printable contents of main-storage, beginning at location 00. Each line contains the hexadecimal address of the first byte in the line, eight main-storage words in hexadecimal, and the same eight words in EBCDIC.

For MFT, MVT, and M65MP dumps, use the IMDPRDMP print dump program to print the SYS1.DUMP data set. The output of IMDPRDMP is described in the publication, IBM System/360 Operating System: Service Aids, GC28-6719.

Stand-alone dumps, invoked by the dump program you have produced from the IMDSADMP macro instruction (see Appendix G), offer a complete picture of main storage at a given time. They are, for the most part, unedited. Each line contains the hexadecimal address of the first byte in the line, eight main-storage words in hexadecimal, and the same eight words in EBCDIC.

General Notes:

- Displacements and addresses shown in the text and illustrations of this publication are given in decimal numbers, followed by the corresponding hexadecimal number in parentheses, e.g., TCB+14(E); location 28(1C); SVC 42(2A). All other numbers in the text are decimal, e.g., the seventeenth word of the TCB; a 4-word control block; 15 job steps.
- Control block field names referred to are those used in the IBM System/360 Operating System: System Control Blocks manual, GC28-6628.
- Wherever possible, diagrams, and reproductions of dumps have been included to aid you during the debugging process.

Section 1: Operating System Concepts

To effectively use the debugging aids provided by the IBM System/360 Operating System, you should be familiar with those control blocks, traces, and other control information that can lead you quickly to the source of error. This section of the manual introduces you to the control information that you must know to interpret dumps. It is divided into four topics:

- TASK MANAGEMENT
- MAIN STORAGE SUPERVISION
- SYSTEM CONTROL BLOCKS AND TABLES
- TRACES

The first two topics deal with those aspects of task management and main storage management, respectively, that are represented in dumps. The third topic describes the remaining system control blocks and tables helpful in pinpointing errors. The last topic covers tracing features that are useful in re-creating the events that led to an error condition.

Note: The descriptions of system control blocks and tables in this section emphasize function rather than byte-by-byte contents. Appendix J summarizes the contents of those control blocks most useful in debugging.

For a more detailed description of system control blocks and tables, please see the System Control Blocks publication, GC28-6628.

Task Management

The task management control information most useful in debugging with a dump includes the task control block and its associated request blocks and elements. These items have the same basic functions at each of the three control program levels. Their functions, interactions, and relationships to other system features are discussed in this topic. A summary of how task supervision differs at each system level concludes the topic.

Task Control Block

The operating system keeps pointers to all information related to a task in a task control block (TCB). For the most part, the TCB contains pointers to other system control blocks. By using these pointers, you can learn such facts as what I/O devices were allocated to the task, which data sets were open, and which load modules were requested.

Figure 1 shows some of the control information that can be located by using the pointers in the TCB. Later, in the discussion of system control blocks and tables, Figure 1 is expanded to show the actual block names and pointer addresses.

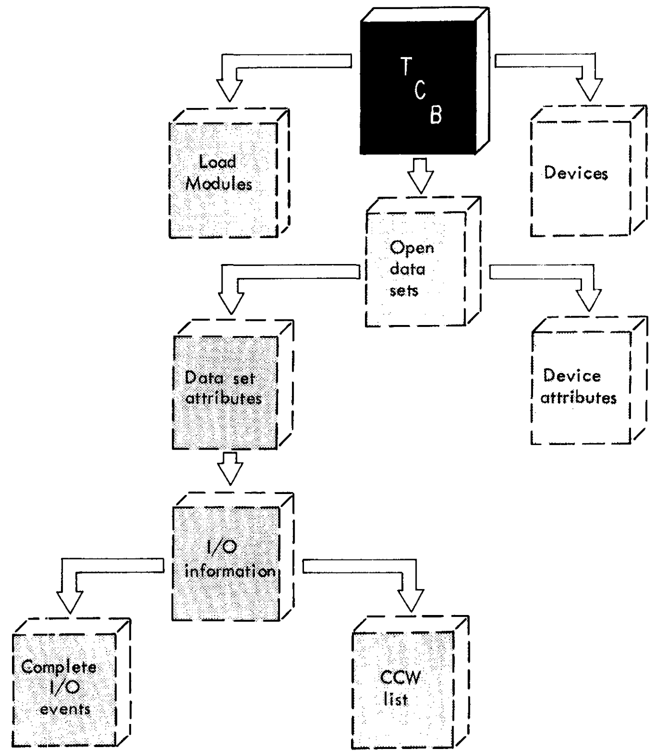


Figure 1. Control Information Available Through the TCB

Request Blocks

Frequently, the routines that comprise a task are not all brought into main storage with the first load module. Instead, they are requested by the task as it requires them. This dynamic loading capability necessitates another type of control block to describe each load module associated with a task -- a request block (RB). An RB is created by the control program when it receives a request from the system or from a problem program to fetch a load module for execution, and at other times, such as when a type II supervisor call (SVC) is issued. By looking at RBs, you can determine which load modules have been executed, why each lost control, and, in most cases, which one was the source of an error condition.

There are seven types of RBs created by the control program:

- Program request block (PRB)
- Supervisor request block (SVRB)
- Interrupt request block (IRB)
- Supervisor interrupt request block (SIRB)
- Loaded program request block (LPRB)
- Loaded request block (LRB)
- Finch request block (FRB)

Of these, you will most often encounter the PRB and SVRB in dumps. The type of RB created depends on the routine or load module with which it is associated.

PRB (Systems with PCP and MFT): A PRB is created whenever an XCTL, LINK, or ATTACH macro instruction is issued. It is located immediately before the load module with which it is associated.

PRB (Systems with MVT): A PRB is created whenever an XCTL or LINK macro instruction is issued. It is located in a fixed area of the operating system.

SVRB: An SVRB is created each time a type II, III, or IV supervisor call is issued. (Type I SVC routines are resident, but run disabled; they do not require a request block.) This block is used to store information if an interruption occurs during execution of these SVC routines. A list of SVCs, including their numbers and types, appears in Appendix A.

IRB: An IRB is created each time an asynchronous exit routine is executed. It is associated with an event that can occur at an unpredictable time during program execution, such as a timing routine initiated by an STIMER macro instruction. The IRB is filled at the time the event occurs, just before control is given to the exit routine.

SIRB: An SIRB is similar to an IRB, except that it is associated only with IBM-supplied input/output error routines. Its associated error routine is fetched from the SYS1.SVCLIB data set.

LPRB: (PCP and MFT only): An LPRB is created when a LOAD macro instruction is issued unless the LOAD macro instruction specifies:

- A routine that has already been loaded.
- A routine that is being loaded in response to a LOAD macro instruction previously issued by a task in the partition (MFT with subtasking).
- A routine that is "only loadable" (see LRB).

An LPRB is located immediately before the load module with which it is associated. Routines for which an LPRB is created can also be invoked by XCTL, LINK, and ATTACH macro instructions.

LRB: (PCP and MFT only): The LRB is a shortened form of an LPRB. Routines associated with LRBs can be invoked only by a LOAD macro instruction. This attribute is assigned to a routine through the OL (only loadable) subparameter in the PARM parameter of the EXEC statement that executes the linkage editor. The most common reason for assigning this attribute is that linkage conventions for XCTL, LINK, and ATTACH are not followed. This request block is located immediately before the load module with which it is associated.

FRB (MFT with subtasking only): An FRB is created and attached to the job pack area queue, during LOAD macro instruction processing, if the requested module is not already in the job pack area. The FRB describes a module being loaded in response to a LOAD macro instruction. Any subsequent requests for the same module, received while it is still being loaded, are deferred by means of wait list elements (WLEs) queued to the FRB. When the module is fully loaded, an LRB or an LPRB is created, the FRB is removed from the job pack area queue, and any requests, represented by wait list elements, are reinitiated.

Figure 2 shows the relative size of the seven types of RBs and the significant fields in each.

In Figure 2, the "size" field tells the number of doublewords in both the RB and its associated load module. The PSW contained in the "resume PSW" field reflects the reason that the associated load module lost control. Other fields are discussed in succeeding topics.

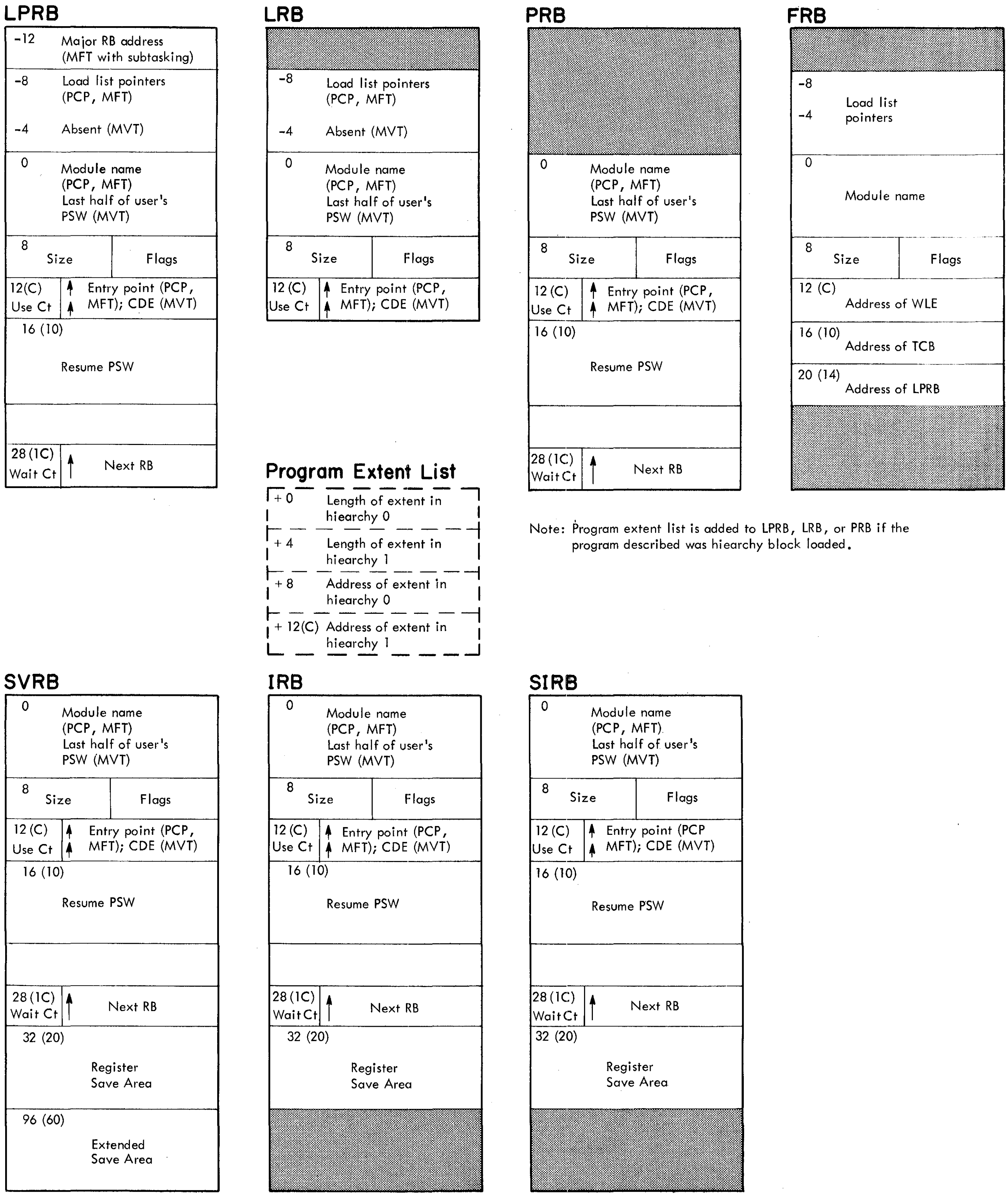


Figure 2. RB Formats

Thus far, the characteristics of the TCB and its associated RBs have been discussed. With the possibility of many RBs subordinate to one task, it is necessary that queues of RBs be maintained. In systems with PCP and MFT without subtasking, two queues are maintained by the system -- the active RB queue and the load list. In MFT systems with subtasking, a job pack area queue, containing FRBs, and LRBs and LPRBs that represent reenterable modules is also maintained. MVT systems maintain an active RB queue and a contents directory. The contents directory is made up of three separate queues: the link pack area control queue (LPAQ); the job pack area control queue (JPAQ); and the load list.

Active RB Queue

The active RB queue is a chain of request blocks associated with active load modules and SVC routines. This queue can contain PRBs, SVRBs, IRBs, SIRBs, and under certain circumstances, LPRBs. Figure 3 illustrates how the active RB queue links together the TCB and its associated RBs.

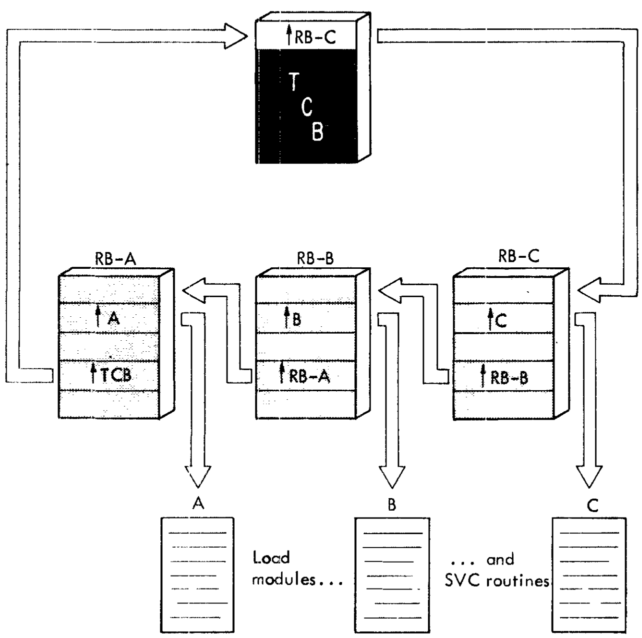


Figure 3. Active RB Queue

The request blocks in the active RB queue in Figure 3 represent three load modules. Load module A invokes load module B, and B, in turn, invokes C. When execution of A began, only one RB existed. When the first invoking request was encountered, a second RB was created, the TCB field that points to the most recent RB was changed, and A's status information was stored in RB-A. A similar set of actions occurred when the second invoking request was encountered. As each load module is executed and control is returned to the next higher level load module, its RB is removed from the chain and pointers are updated accordingly.

Load List

The load list is a chain of request blocks or elements associated with load modules invoked by a LOAD macro instruction. The load list differs from the active RB queue in that RBs and associated load modules are not deleted automatically. They remain intact until they are deleted with a DELETE macro instruction or job step termination occurs. By looking at the load list, you can determine which system and problem program routines were loaded before the dump was taken. The format of the load list differs with control program levels.

Systems with PCP and MFT (without subtasking): At these control program levels, the load list associated with a TCB contains LRBs and LPRBs. RBs on the load list are linked together somewhat differently from those on the active RB queue because of the characteristics of the LOAD macro instruction. Because RBs may be deleted from a load list in a different order than they were created (depending on the order of DELETE macro instructions), they must have both forward and backward pointers. Figure 4 illustrates how a load list links together a TCB and three RBs.

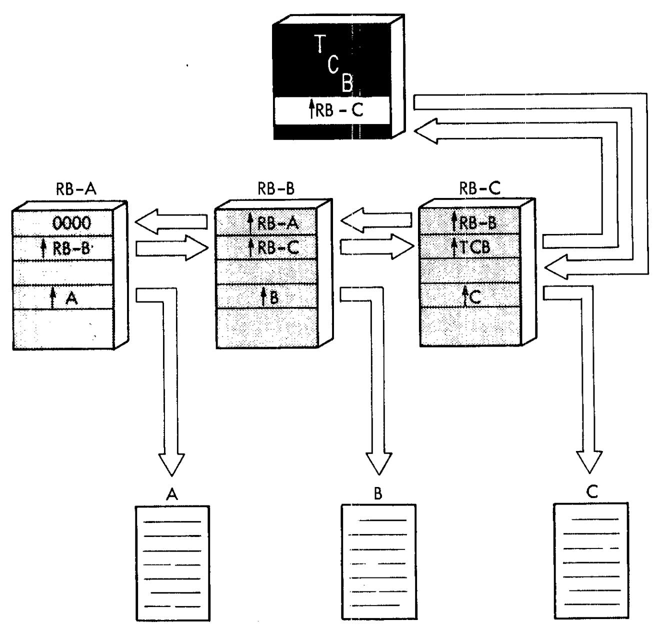


Figure 4. Load List (PCP, MFT)

Here, each RB contains a pointer both to the previous RB and the next most recent RB in the list. If there is no previous or more recent RB, these fields contain zeros and a pointer to the TCB, respectively.

Another field of a load list RB that merits consideration is the use count. Whenever a LOAD macro instruction is issued, the load list is searched to see if the routine is already loaded. If it is loaded, the system increments the use count by one and passes the entry point address to the requesting routine.

Each time a DELETE macro instruction is issued for the routine, the use count is decremented by one. When it reaches zero, the RB is removed from the load list and storage occupied by the associated routine is freed.

Systems with MFT (with subtasking): At this control program level, the load list is used as described for PCP and MFT without subtasking, with the following exceptions:

1. The LRBs and LPRBs queued on the load list represent modules that are not reenterable. LRBs and LPRBs representing reenterable modules are queued on the job pack area queue.
2. When a LOAD macro instruction is issued, the system searches the job pack area queue before searching the load list.

Systems with MVT: Instead of LRBs and LPRBs created as a result of LOAD macro instructions, the load list maintained by a system with MVT contains elements representing load modules. Load list elements (LLEs) are associated with load modules through another control medium called the contents directory.

The contents directory is made up of three separate queues: the link pack area control queue (LPAQ), the job pack area control queue (JPAQ), and the load list.

The LPAQ is a record of every program in the system link pack area. This area contains reenterable routines specified by the control program or by the user. The routines in the system link pack area can be used repeatedly to perform any task of any job step in the system. The entries in the LPAQ are contents directory entries (CDEs).

There is a JPAQ for each job step in the system that uses a program not in the link pack area. The JPAQ, like the LPAQ, is made up of CDEs. It describes routines in a job step region. The routines in the job pack area can be either reenterable or not reenterable. These routines however, cannot be used to perform a task that is not part of the job step.

The load list represents routines that are brought into a job pack area or found in the link pack area by the routines that perform the Load function. The entries in the load list are load list elements, not CDEs. Each load list element is associated with a CDE in the JPAQ or the LPAQ; the programs represented in the load list are thus also represented in one of the other contents directory queues.

Load list elements also contain a count field that corresponds to the use count in a LPRB or LRB. Each time a LOAD macro instruction is issued for a load module already represented on the load list, the count is incremented by one. As corresponding DELETE macro instructions are issued, the count is decremented until it reaches zero. An LLE has the following format:

Res	↑ Next LLE	Count	↑ CDE
0	1	4	5

Byte 0: Reserved (RES).

Bytes 1-3: Pointer to the next more recent LLE on the load list.

Byte 4: Count.

Bytes 5-7: Pointer to the corresponding CDE.

More will be said about CDEs in the next topic of Section 1, titled "Main Storage Supervision."

Job Pack Area Queue (MFT with Subtasking only)

In an MFT system with subtasking, the job pack area queue is a chain of request blocks associated with load modules invoked by a LOAD macro instruction. The queue contains FRBs, and those LRBs and LPRBs that represent reenterable modules. FRBs are queued on the job pack area queue until the requested module is completely loaded. When the module is completely loaded into main storage, the FRB is removed from the job pack area queue and replaced with an LBR or an LPR queue on the job pack area queue if the loaded module is reenterable, and on the load list if it is not.

In the MFT with subtasking configuration, the load list represents non-reenterable modules, while the job pack area queue represents only reenterable modules within the partition. These RBs on the job pack area queue are not deleted automatically, but remain intact until they are deleted by a DELETE macro instruction, or until job step termination occurs. Reenterable load modules are therefore retained in the partition for use by the job step task or any subtasks which may be created.

Whenever a LOAD macro instruction is issued, the job pack area queue is searched. If the routine is already fully loaded and represented by an LRB or an LPRB on the JPAQ (the routine is reenterable), the system increments the use count by one and passes the module entry point address to the requesting routine. If an FRB for the requested module is found, a wait list element (WLE) representing the deferred request is queued to the FRB, and the request is placed in a wait. When the requested routine is fully loaded, the system releases the request from the wait condition, and the request is re-initiated. If no RB for the requested routine is found, an FRB is created and queued on the JPAQ. The system then searches the load list of the requesting task for an RB for the requested routine. If an RB for that routine is found on the load list (the routine is not reenterable), the use count is incremented by one, the entry point address of the module is passed to the requesting routine, and the FRB is dequeued from the JPAQ. If no RB is found on the load list, the FRB remains on the JPAQ and the system begins loading the requested module.

Each time a DELETE macro instruction is issued for the routine, the use count is decremented by one (the DELETE routine ignores FRBs). When the use count reaches zero, the RB is removed from the queue.

Figure 5 illustrates how the job pack area queue is chained to a TCB.

In Figure 5, each RB contains a pointer to the previous RB and a pointer to the next RB on the queue. If there is no previous RB on the queue, that pointer will contain zero; if there is no next RB on the queue (this RB is the most recent on the JPAQ), the next RB pointer will point back to the job pack area queue pointer in the PIB.

Two wait list elements (WLEs) are queued to FRB-C representing deferred requests waiting until the initial loading of the module is completed. The last WLE contains zero in its forward pointer, indicating that it is the last element on the WLE queue.

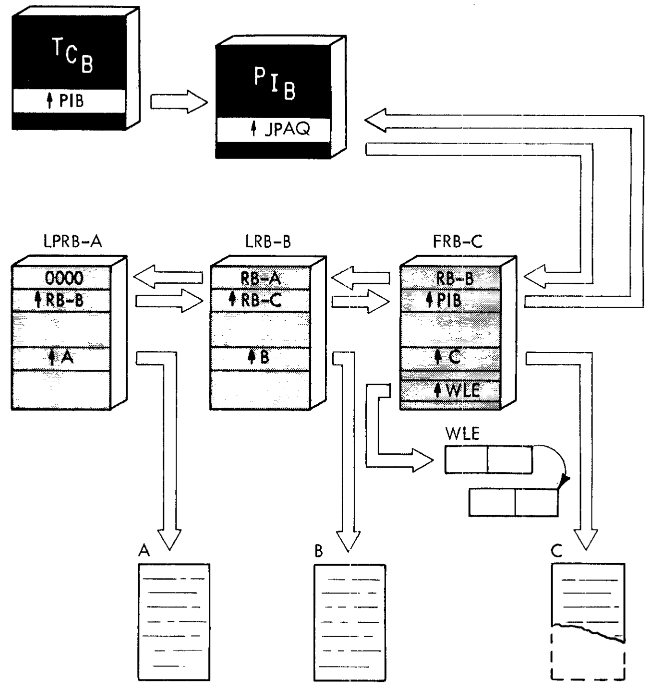


Figure 5. Job Pack Area queue

<u>Effects of LINK, ATTACH, XCTL, and LOAD</u>

In the previous paragraphs we have mentioned the LINK, ATTACH, XCTL, and LOAD macro instructions. A brief description of each will be helpful at this point. LINK, ATTACH, XCTL, and LOAD, though similar, have some distinguishing characteristics and system dependencies worth mentioning. By knowing what happens when these macro instructions are issued, you can make more effective use of the active RB queue and the load list.

<u>LINK</u>: A LINK results in the creation of a PRB chained to the active RB queue. Upon completion of the invoked routine, control is returned to the invoking routine. In systems with PCP and MFT, the RB is removed from the queue. The storage occupied by the invoked routine is freed unless the routine is also represented on the load list, or on the job pack area queue in MFT systems with subtasking. In systems with MVT, the use count in the RB is decremented by one; if it is then zero, the RB and the storage occupied by the routine are marked for deletion. A LINK macro instruction generates an SVC 6.

<u>ATTACH</u>: An ATTACH is similar to the other three macro instructions in systems with PCP or with MFT without subtasking. In systems with MFT (with subtasking) or MVT,

ATTACH is the means for dynamically creating a separate but related task -- a subtask. At the PCP and MFT (without subtasking) levels, tasks cannot create subtasks. ATTACH effectively performs the same functions as LINK at these control program levels, with two notable additions:

1. You can request an exit routine to be given control upon normal completion of the attached routine.

2. You can request the posting of an event control block upon the routine's completion.

Exit routines are represented by additional RBs on the active RB queue. The ATTACH macro instruction generates an SVC 42(2A).

XCTL: An XCTL also results in the creation of a PRB and immediate transfer of control to the invoked routine. However, XCTL differs from the other macro instructions in that, upon completion of the invoked routine, control is passed to a routine other than the invoking routine. In fact, an XCTL does not result in the creation of a lower level RB. Instead, the invoking routine and its associated RBs are deleted when the XCTL is issued. In effect, the RB for the invoked routine replaces the invoking routine's RB. The XCTL macro instruction generates an SVC 7.

LOAD: The LOAD macro instruction was treated previously in the discussion of the load list. To summarize: the system responds to a LOAD by fetching the routine into main storage and passing the entry point address to the requesting routine in register 0. Because the system does not have an indication of when the routine is no longer needed, a LOAD must be accompanied by a corresponding DELETE macro instruction. If not, the routine and its RB remain intact until the job step is terminated. The LOAD macro instruction generates an SVC 8.

System Task Control Differences

Thus far, this topic has dealt with the aspects of task supervision that are similar at the three control program levels. There are, however, some major areas of difference, namely:

1. The number of tasks that can be known to the system concurrently.

2. The layout of main storage.

3. The additional main storage control information in systems with MVT.

The first two subjects are discussed here, by system. The third subject, because of its volume, is discussed in the next topic of Section 1.

Systems with PCP: The distinguishing characteristic of an operating system with the primary control program is that it handles a single task. It has one TCB at any given time, which resides in the system nucleus. Jobs are processed sequentially, one step at a time. ATTACH macro instructions are treated similarly to LINKs; that is, they do not create subtasks.

Figure 6 is a snapshot of main storage in a system with PCP. The fixed area contains those routines, control blocks, and tables that are brought into main storage at IPL, and never overlaid. It also may contain optional access method and SVC routines which are normally nonresident, and an optional list of absolute addresses for routines which reside on direct access devices. These options can be selected during system generation.

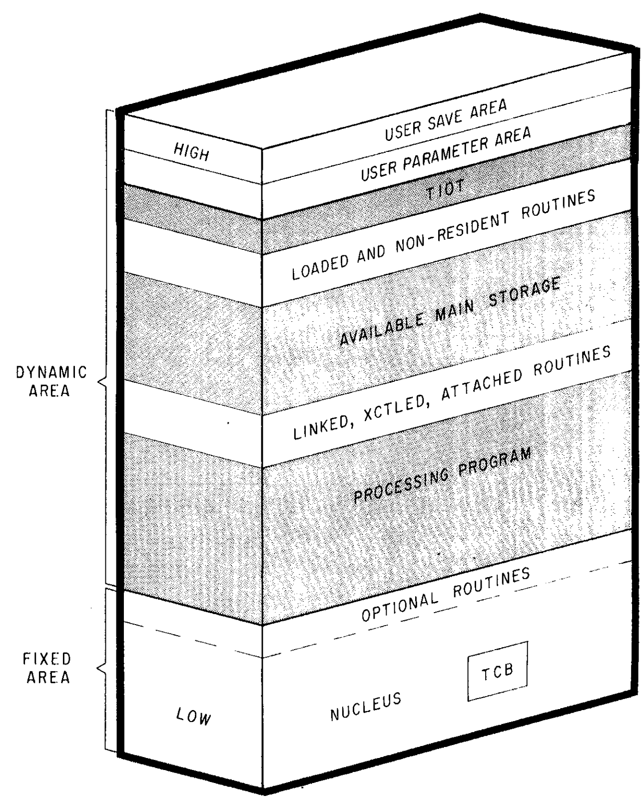


Figure 6. Main Storage Snapshot (PCP)

The dynamic area contains, in lower main storage adjacent to the fixed area, the processing program and routines invoked by

LINK, XCTL, and ATTACH macro instructions. At some points in the job processing flow, the processing program may be a job management routine. Upper main storage contains the user save area, user parameter area, task input/output table, routines requested by LOAD macro instructions, and non-resident routines, such as access method routines.

<u>Systems with MFT (Without Subtasking)</u>: Operating Systems that provide multiprogramming with a fixed number of tasks without the subtasking option (MFT without subtasking), resemble systems with PCP except that the dynamic area may be divided into as many as 52 partitions. Partitions sizes and attributes are defined during system generation. These sizes and attributes remain fixed unless redefined by the operator during or after system initialization. Each partition contains one task. Three additional tasks, the transient area loading task, the communication task, and the master scheduler task, reside in the fixed area. One TCB exists for each task. All TCBs are linked by dispatching priority in a TCB queue, beginning with the TCBs for the three resident tasks.

The <u>dynamic area</u> may contain as many as 3 reading tasks, as many as 36 writing tasks, and as many as 15 job step tasks, so long as the total number of tasks does not exceed 52. Jobs are processed sequentially in a partition, one job step at a time. An ATTACH macro instruction, as in systems with PCP, is treated similarly to a LINK.

Because more than one task exists at any given time, systems with MFT introduce the concept of <u>task switching</u>. The relative dispatching priority of tasks is determined by the TCB queue. Control of the CPU must often be relinquished by one task and given to another of higher priority. MFT dumps contain task switching information often important in reconstructing the environment at the time of task failure.

Figure 7 is a snapshot of main storage in a system with MFT (without subtasking), having n partitions. The fixed area contains the nucleus (including the TCB queue, transient area loading task, communications task, and master scheduler task), and the system queue area. The <u>fixed area</u> may also contain the same system generation options discussed under the heading "Systems with PCP," and a reenterable load module area, which is optional in MFT. Each partition in the dynamic area is similar to the entire dynamic area of PCP.

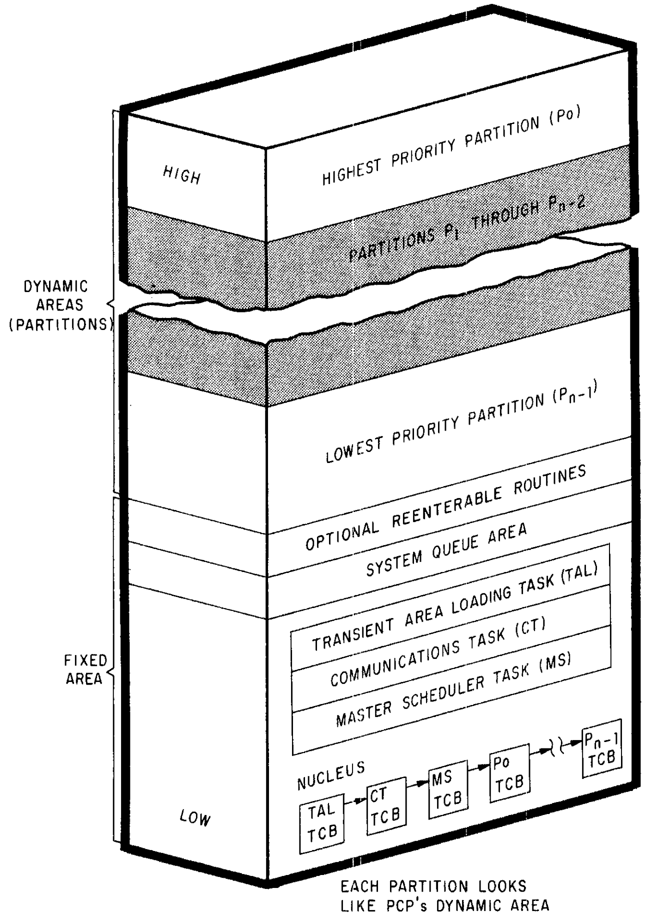


Figure 7. Main Storage Snapshot (MFT Without Subtasking)

<u>Systems with MFT (With Subtasking)</u>: Operating Systems that provide multiprogramming with a fixed number of tasks with the subtasking option (MFT with subtasking) more closely resemble systems with MVT, and differ from MFT systems without subtasking in the following major areas:

1. MFT with subtasking has an ATTACH facility similar to the ATTACH facility in MVT. While the number of job step TCBs still may not exceed 15, the number of tasks in any partition, and therefore the total number of tasks in the system, is now variable. Job step task TCBs reside in the nucleus. They are queued, following the system task TCBs, in the same manner as in MFT without subtasking. When subtasks are created, however, the subtask TCBs are placed in the system queue area and queued to the job step TCBs according to dispatching priority (TCBTCB field), and according to subtask relationships (TCBNTC, TCBOTC, TCBLTC fields).

2. MFT with subtasking provides the ability to change the dispatching priority of any task within a partition through the use of the CHAP macro instruction. For information regarding the use of the CHAP macro instruction, refer to the publication IBM System/360 Operating System: Supervisor and Data Management Services, GC28-6646.

Figure 8 is a snapshot of main storage in an MFT system with subtasking having n partitions. Note here that the TCBs in the nucleus are all job step TCBs, while those residing in the sytem queue area are the subtask TCBs.

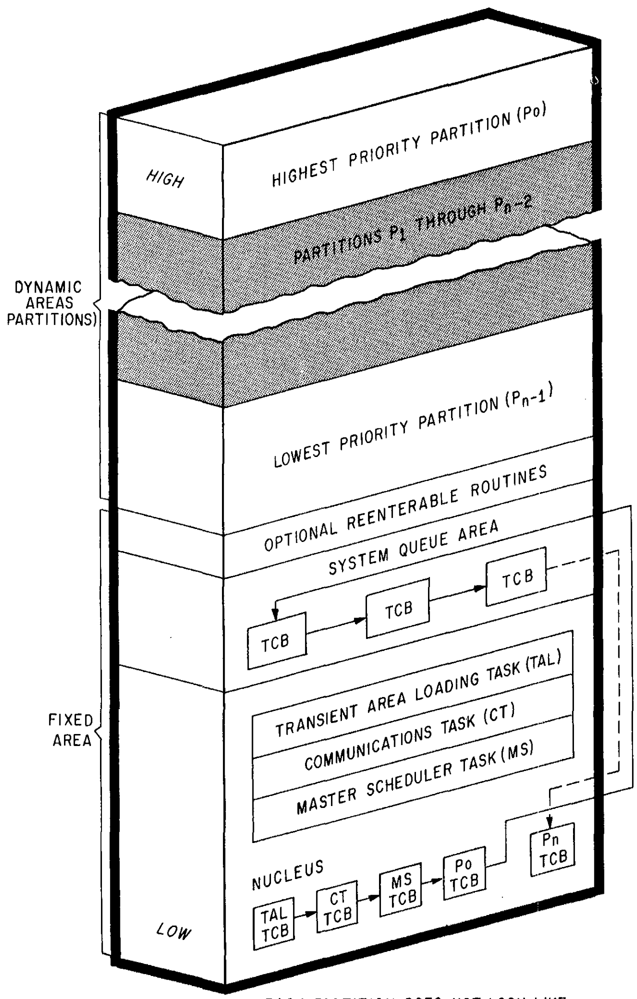


Figure 8. Main Storage Snapshot (MFT With Subtasking)

Systems with MVT: In Operating Systems that provide multiprogramming with a variable number of tasks (MVT), as many as 15 job steps can be executed concurrently. Each job step requests an area of main storage called a region and is executed as a job step task. In addition, system tasks request regions and can be executed concurrently with job step tasks.

Regions are assigned automatically from the dynamic area when tasks are initiated. Regions are constantly redefined according to the main storage requirements of each new task.

With the facility of attaching subtasks available to each task through the ATTACH macro instruction, the number of TCBs in the system is variable. Tasks gain control of the CPU by priority. To keep track of the priority and status of each task in the system, TCBs are linked together in a TCB queue.

Figure 9 is a snapshot of main storage in a system with MVT. The fixed area is occupied by the resident portion of the control program loaded at IPL. The system queue space is reserved for control blocks and tables built by the control program. The dynamic area is divided into variable-sized regions, each of which is allocated to a job step task or a system task. Finally, the link pack area contains selected reenterable routines, loaded at IPL. If an IBM 2361 Core Storage device and Main Storage Hierarchy Support are included in the system, a secondary link pack area may be created in hierarchy 1 to contain other reenterable routines.

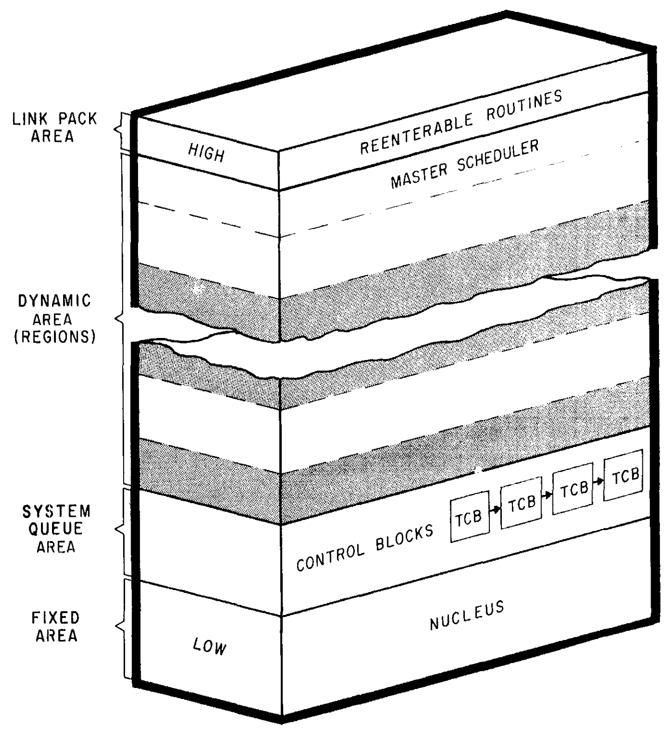


Figure 9. Main Storage Snapshot (MVT)

Main Storage Supervision

Because main storage is allocated dynamically in an operating system, current storage control information must be kept. Such information is contained in a series of control blocks called queue elements. In systems with PCP and MFT without subtasking, queue elements reflect areas of main storage that are unassigned. In MFT systems with subtasking, a gotten subtask area queue element (GQE) is introduced to record storage obtained for a subtask by a supervisor issued GETMAIN macro instruction. In systems with MVT, more elaborate storage control is maintained; at any given time, queue elements reflect the distribution of main storage in regions, subpools, and load modules. A familiarity with storage control information is necessary to understand the main storage picture provided in dumps.

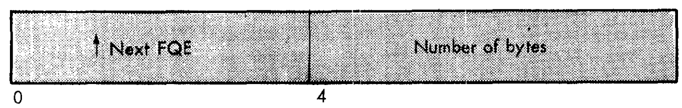
The dynamic area may be significantly expanded by including IBM 2361 Core Storage in the system. Main Storage Hierarchy Support for IBM 2361 Models 1 and 2 permits selective access to either processor storage (hierarchy 0) or 2361 Core Storage (hierarchy 1). If IBM 2361 Core Storage is not included, requests for storage from hierarchy 1 are obtained from hierarchy 0. If 2361 Core Storage is not present in an MVT system and a region is defined to exist in two hierarchies, a two-part region is established within processor storage. The two parts are not necessarily contiguous.

Storage Control in Systems With PCP

The chain of storage control information in a system with PCP begins at a table called the main storage supervisor (MSS) boundary box, located in the system nucleus. This table, pointed to by the TCBMSS field of the TCB, contains three words. The first word points to a free queue element (FQE) associated with the highest free area in processor storage. The second word points to the first doubleword outside the nucleus. The third word contains the highest address in processor storage plus one.

If Main Storage Hierarchy Support for IBM 2361 Models 1 and 2 is included in the system, the boundary box is expanded to six words. The first byte of the expanded boundary box contains a "1" in bit 7 to indicate that hierarchy support is included. The second set of three words describes storage in hierarchy 1. The first word of this second set points to an FQE associated with the highest free area in hierarchy 1. The second word points to the first doubleword in hierarchy 1. The third word points to the highest position in hierarchy 1 plus one. If 2361 Core Storage is not included in the system, the hierarchy 1 pointers are set to zero.

FQE: Each free area in main storage is described by an FQE. FQEs are chained, beginning with the FQE associated with the highest address. If Main Storage Hierarchy Support is present, one FQE chain exists for each hierarchy specified. Each FQE occupies the first 8 bytes of the area it describes. It has the following format:



Bytes 0-3: Pointer to FQE associated with next lower free area or, if this is the last FQE, zeros.

Bytes 4-7: Number of bytes in the free area.

Storage control in systems with PCP is summarized in Figure 10.

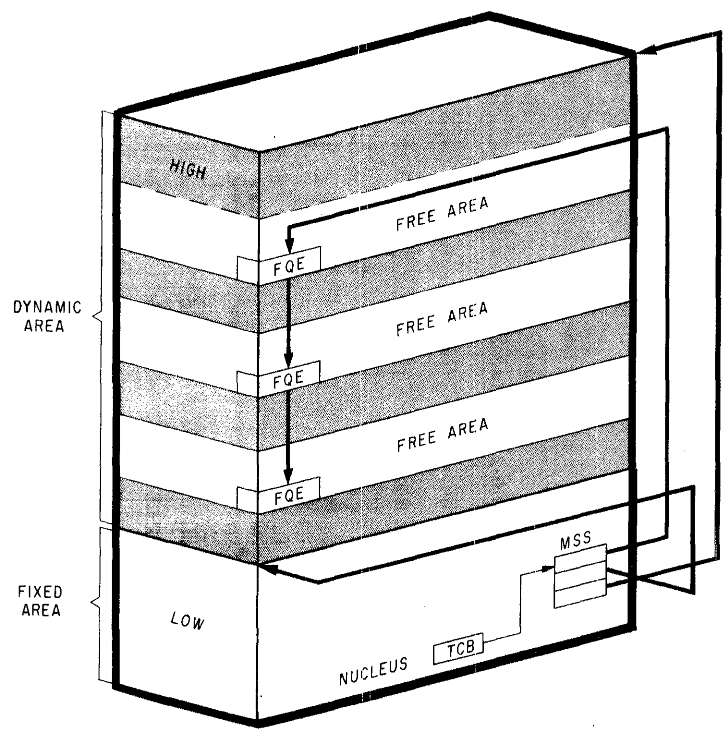


Figure 10. Storage Control (PCP)

Storage Control in Systems with MFT (Without Subtasking)

Storage control information in systems with MFT without subtasking is similar to that in systems with PCP, except that one MSS boundary box is maintained for each partition. The TCB associated with the partition contains a pointer (TCBMSS) to the boundary box.

If Main Storage Hierarchy Support is included, the first half of each expanded boundary box describes the processor storage (hierarchy 0) partition segment, and the second half describes the 2361 Core Storage (hierarchy 1) partition segment. Any partition segment not currently assigned storage in the system has the applicable boundary box pointers set to zero. If the partition is established entirely within hierarchy 0, or if 2361 Core Storage is not included in the system, the hierarchy 1 pointers in the second half of the expanded boundary box are set to zero. If a partition is established entirely within hierarchy 1, the hierarchy 0 pointers in the first half of the expanded boundary box are set to zero.

The boundary box format for MFT is identical to the format for PCP. The pointers, however, point to the boundaries of the partition and to the partition FQEs rather than to the boundaries of storage. Figure 11 summarizes storage control in systems with MFT.

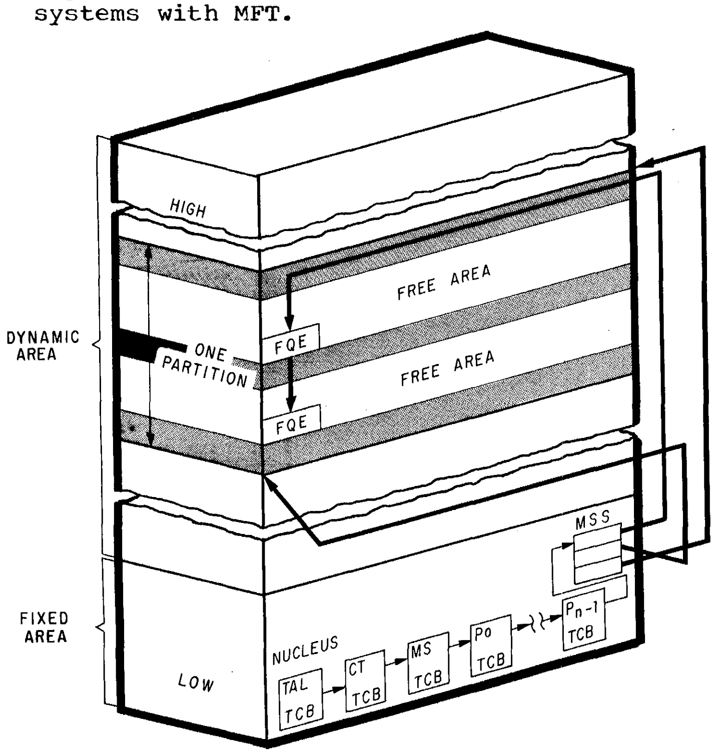


Figure 11. Storage Control for a Partition (MFT Without Subtasking)

Storage Control in Systems with MFT (With Subtasking)

Storage control information for the job step or partition TCB in MFT systems with subtasking is handled in the same way as in MFT systems without subtasking. However, when subtasks are created, the supervisor builds another control block, the gotten subtask area queue element (GQE). The GQEs associated with each subtask originate from a one word pointer addressed by the TCBMSS field of the subtask TCB.

GQE: Each area in main storage belonging to a subtask, and obtained by a supervisor issued GETMAIN macro instruction, is described by a gotten subtask area queue element (GQE). GQEs are chained in the order they are created. The TCBMSS field of the subtask TCB contains the address of a word which points to the most recently created GQE.

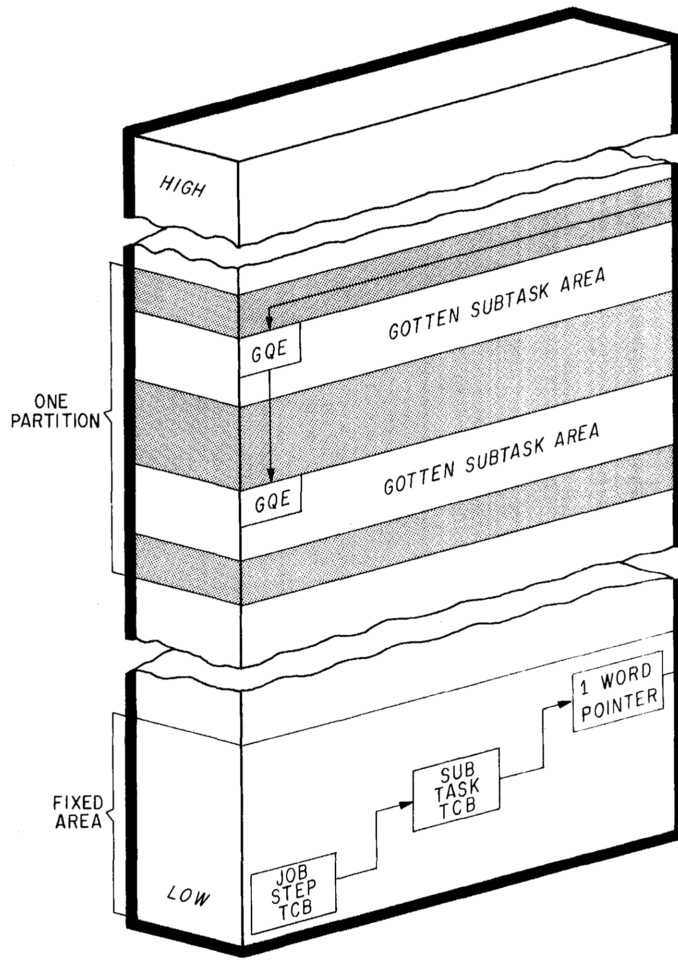


Figure 12. Storage Control for Subtask Storage (MFT with Subtasking)

Bytes 1-3: Pointer to the FQE associated with the first free area.

Bytes 5-7: Pointer to the next DQE or, if this is the last DQE, zeros.

Bytes 9-11(B): Pointer to first 2K block described by this DQE.

Bytes 13-15(D-F): Length in bytes of area described by this DQE.

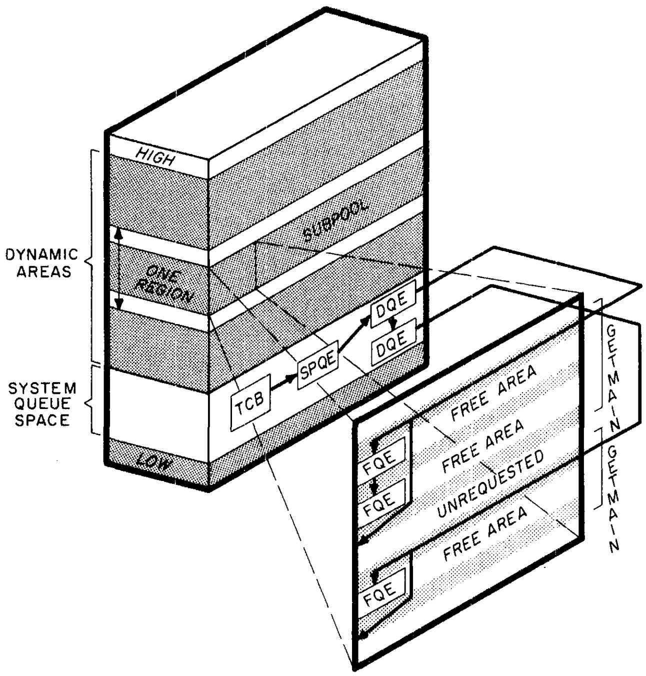


Figure 14. Storage Control for a Subpool (MVT)

FQE: The FQE describes a free area within a set of 2K blocks described by a DQE. It occupies the first eight bytes of that free area. Since the FQE is within the subpool, it has the same protect key as the task active within that subpool. Extreme care should be exercised to see that FQEs are not destroyed by the problem program. If an FQE is destroyed, the free space that it describes is lost to the system and cannot be assigned through a GETMAIN. As area distribution within the set of blocks changes, FQEs are added to and deleted from the free queue. An FQE has the following format:

Res	↑Next FQE	Res	Number of bytes
0	1	4	5

Bytes 1-3: Pointer to the next lower FQE or, if this is the last FQE, zeros.

Bytes 5-7: Number of bytes in the free area.

A subpool is summarized in Figure 14.

Storage Control for a Load Module in Systems with MVT

Each load module in main storage is described by a contents directory entry (CDE) and an extent list (XL) that tells how much space it occupies.

CDE: The contents directory is a group of queues, each of which is associated with an area of main storage. The CDEs in each queue represent the load modules residing in the associated area. There is a CDE queue for the link pack area and one for each region, or job pack area. The TCB for the job step task that requested the region points to the first CDE for that region. Contents directory queues reside in the system queue space. A CDE has the following format:

Flags	↑Next CDE	Res	↑RB
0	1	4	5

EBCDIC program name
8

Count	↑Entry point	Flags	↑XL
16(10)	17(11)	20(14)	21(15)

Byte 0: Flag bits, when set to one, indicate:
- Bit 0 - Module was loaded by NIP.
- Bit 1 - Module is in process of being loaded.
- Bit 2 - Module is reenterable.
- Bit 3 - Module is serially reusable.
- Bit 4 - Module may not be reused.
- Bit 5 - This CDE reflects an alias name (a minor CDE).
- Bit 6 - Module is in job pack area.
- Bit 7 - Module is not only-loadable.

Bytes 1-3: Pointer to next CDE.

Bytes 5-7: Pointer to the RB.

Bytes 8-15(F): EBCDIC name of load module.

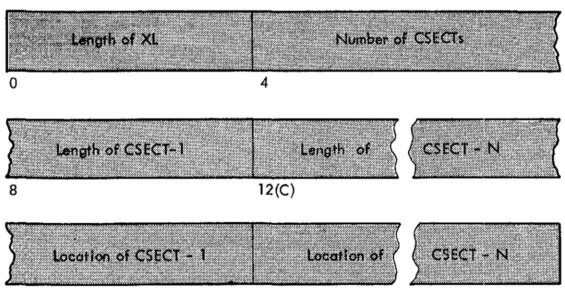
Byte 16(10): Use count.

Bytes 17-19(11-13): Entry point address of load module.

Byte 20: Flag bits, when set to one, indicate:
- Bit 0 - Reserved.
- Bit 1 - Module is inactive.
- Bit 2 - An extent list has been built for the module.
- Bit 3 - This CDE contains a relocated alias entry point address.
- Bit 4 - The module is refreshable.
- Bits 5, 6, 7 - Reserved.

Bytes 21-23(15-17): Pointer to the XL for this module or, if this is a minor CDE, pointer to the major CDE.

XL: The total amount of main storage occupied by a load module is reflected in an extent list (XL). XLs are located in the system queue space. An XL has the following format:



Bytes 0-3: Length of XL in bytes.

Bytes 4-7: Number of scattered control sections. If the control sections are block-loaded, 1.

Remaining bytes: Length in bytes of each control section in the module (4 bytes for each control section) and starting location of each control section (4 bytes for each control section).

Storage control elements and queues for load modules are summarized in Figure 15.

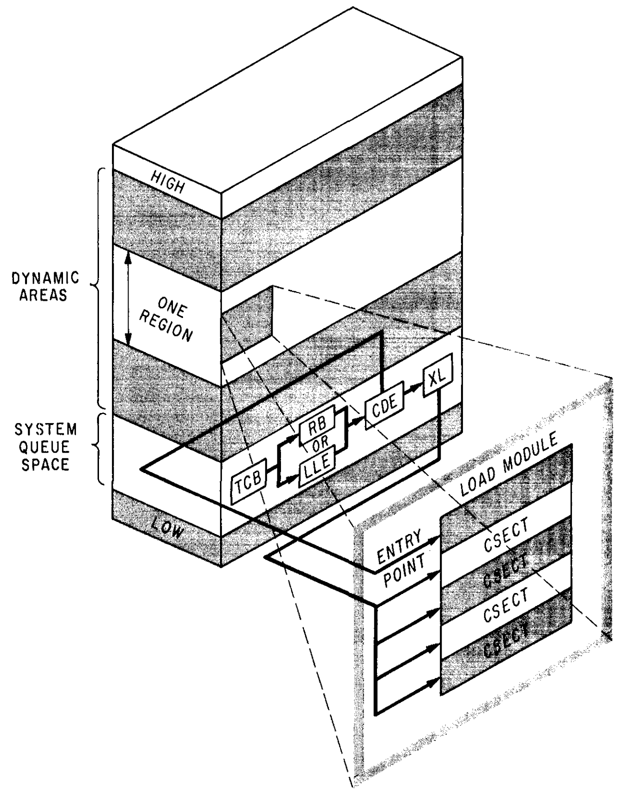


Figure 15. Storage Control for a Load Module (MVT)

System Control Blocks and Tables

In addition to the key task management control blocks (TCB and RB), several other control blocks containing essential debugging information are built and maintained by data management and job management routines. Although some of these blocks are not readily identifiable on a storage dump, they can be located by following chains of pointers that begin at the TCB.

The control blocks discussed here have the same basic functions at each control program level. The precise byte-by-byte contents of the blocks can be found in the publication System Control Blocks. Block contents useful in debugging are listed in Appendix J.

Communications Vector Table (CVT)

The CVT provides a means of communication between nonresident routines and the control program nucleus. Its most important role in debugging is its pointer to two words of TCB addresses. These words enable you to locate the TCB of the active task, and from there to find other essential control information. Storage location 16(10) contains a pointer to the CVT.

Task Input/Output Table (TIOT)

A TIOT is constructed by job management for each task in the system. It contains primarily pointers to control blocks used by I/O support routines. It is usually located in the highest part of the main storage area occupied by the associated task (in systems with MVT, TIOTs are in the system queue space.) Through the TIOT, you can obtain addresses of unit control blocks allocated to the task, the job and step name, the ddnames associated with the step, and the status of each device and volume used by the data sets.

Unit Control Block (UCB)

The UCB describes the characteristics of an I/O device. One UCB is associated with each I/O device configured into a system. The UCB's most useful debugging aid is the sense information returned by the last sense command issued to the associated device.

Event Control Block (ECB)

The ECB is a 1-word control block created when a READ or WRITE macro instruction is issued, initiating an asynchronous I/O operation. At the completion of the I/O operation, the access method routine posts the ECB. By checking this ECB, the completion status of an I/O operation can be determined. In all access methods but QTAM, the ECB is the first word of a larger block, the data event control block.

Input/Output Block (IOB)

The IOB is the source of information required by the I/O supervisor. It is filled in with information taken from an I/O operation request. In debugging, it is useful as a source of pointers to the DCB associated with the I/O operation and the channel commands associated with a particular device.

Data Control Block (DCB)

The DCB is the place where the operating system and the problem program store all pertinent information about a data set. It may be completely filled by operands in the DCB macro instruction, or partially filled in and completed when the data set is opened, with subparameters in a DD statement and/or information from the data set label. The format of DCBs differs slightly for each of the various access methods and device types. The DCB's primary debugging aids are its pointers to the DEB and current IOB associated with its data set, and the offset value of the ddname in the TIOT.

Data Extent Block (DEB)

A DEB describes a data set's auxiliary storage assignments and contains pointers to some other control blocks. The DEB is created and queued to the TCB at the time a data set is opened. Each TCB contains a pointer to the first DEB on its chain. Through this pointer you can find out which data sets are opened for the task at a given time, what extents are occupied by open data sets, and where the DCB and UCB are located.

Summary of Control Block Relationships

Figure 16, an expansion of Figure 1, shows the relationships among the principal control blocks and tables in the System/360 Operating System.

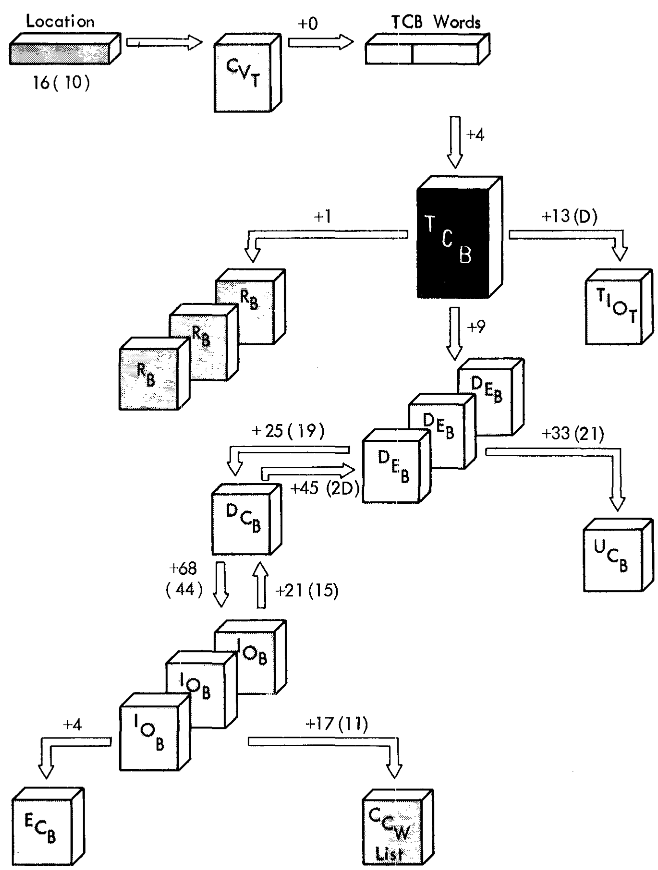


Figure 16. Control Block Relationships

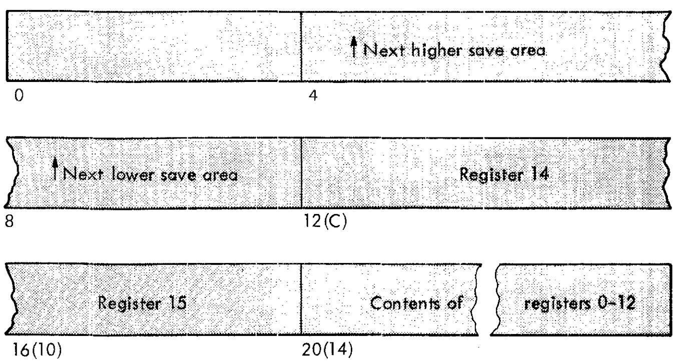
Traces

Two features that assist you in tracing the flow of your program are the save area chain and the trace table (the trace table is optional at system generation.) Both these features are edited and clearly identified on ABEND/SNAP dumps, and can be located easily on storage image and stand-alone dumps.

Save Area Chain

When control is passed from one load module to another, the requested module is responsible for storing the contents of general registers. This necessitates the use of separate save areas for each level of load module in a task. With the different types of linkages that can occur, save areas must be chained so that each one points to both its predecessor and successor.

A save area is a block of 72 bytes containing chain pointers and register contents. It has the following format:



Bytes 4-7: Pointer to the next higher level save area or, if this is the highest level save area, zeros.

Bytes 8-11(B): Pointer to the next lower level save area or, if this is the lowest level save area, unused.

Bytes 12-15(C-F): Contents of register 14 (optional)

Bytes 16-19(10-13): Contents of register 15 (optional)

Bytes 20-71(14-3F): Contents of registers 0 to 12

The save area for the first or highest level load module in a task (save area 1) is provided by the control program. The address of this area is contained in register 13 when the load module is first entered. It is the responsibility of the highest level module to:

1. Save registers 0-12 in bytes 20-71(14-3F) of save area 1 when it is entered.

2. Establish a new save area (save area 2).

3. Place the contents of register 13 into bytes 4-7 of save area 2.

4. Place the address of save area 2 into register 13.

5. Place the address of save area 2 into bytes 8-11(B) of save area 1.

At this point, the save areas appear as shown in Figure 17.

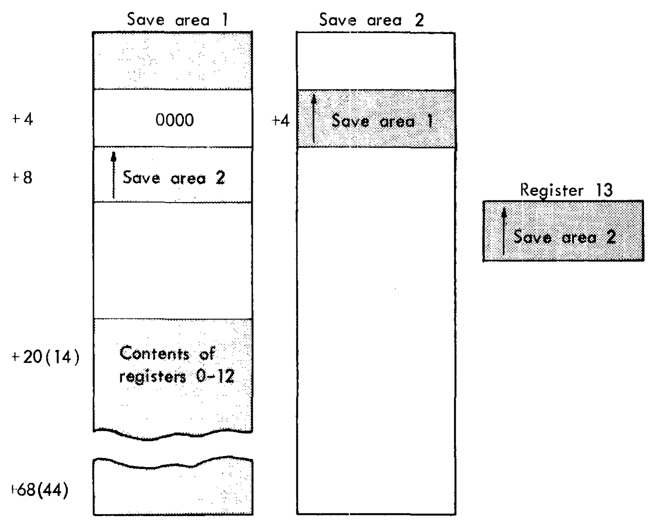


Figure 17. Save Area Trace

If a module requests a lower level module, it must perform actions 1 through 4 to ensure proper restoration of registers when it regains control. (Action 5 is not required, but must be performed if the dump printout of the field is desired.) A module that does not request a lower level module need only perform the first action.

ABEND and SNAP dumps include edited information from all save areas associated with the dumped task under the heading "SAVE AREA TRACE". In a stand-alone dump, the highest level save area can be located through a field of the TCB. Subsequent save areas can be located through the save area chain.

Trace Table

The tracing routine is an optional feature specified during system generation. This routine places entries, each of which is associated with a certain type of event, into a trace table. The size of the table is also a system generation option; when the table is filled, the routine overlays old entries with new entries, beginning at the top of the table (the entry having the lowest storage address). The contents and size of a trace table are highly system-dependent.

Systems with PCP: Trace table entries for systems with PCP are 4 words long and represent occurrences of SIO, I/O, and SVC interruptions. Figure 18 shows the word contents of each type of entry.

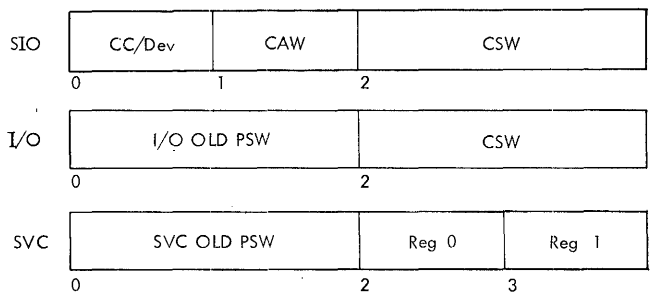


Figure 18. Trace Table Entries (PCP)

Systems with MFT: Systems with MFT have the same type of trace table entries as PCP, plus an additional type representing task switches, as shown in Figure 19.

Systems with MVT: The trace table in a system with MVT is expanded to include more entries and more information in each entry. Trace table printouts occur only on SNAP dumps and stand-alone dumps. Entries are eight words long and represent occurences of SIO, external, SVC, program, and I/O interruptions, and dispatcher loaded PSWs.

Figure 20 shows the word contents of trace table entries for SNAP dumps and stand-alone dumps. Figure 21 shows the contents of trace table entries as filled by MVT with Model 65 multiprocessing. (SSM -- set system mask -- entries are optional.)

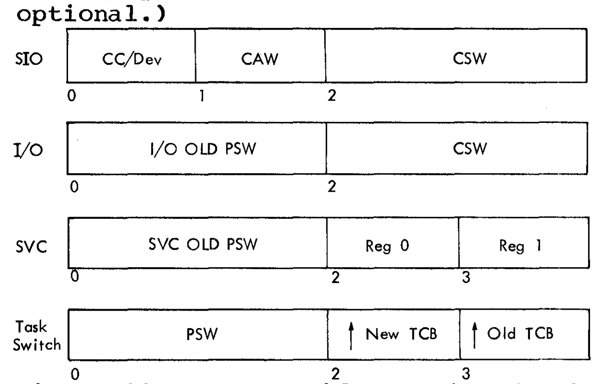


Figure 19. Trace Table Entries (MFT)

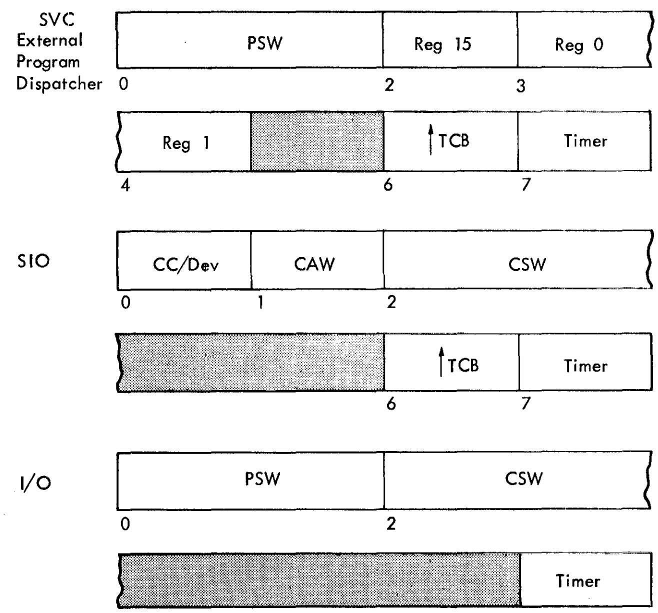


Figure 20. Trace Table Entries (MVT)

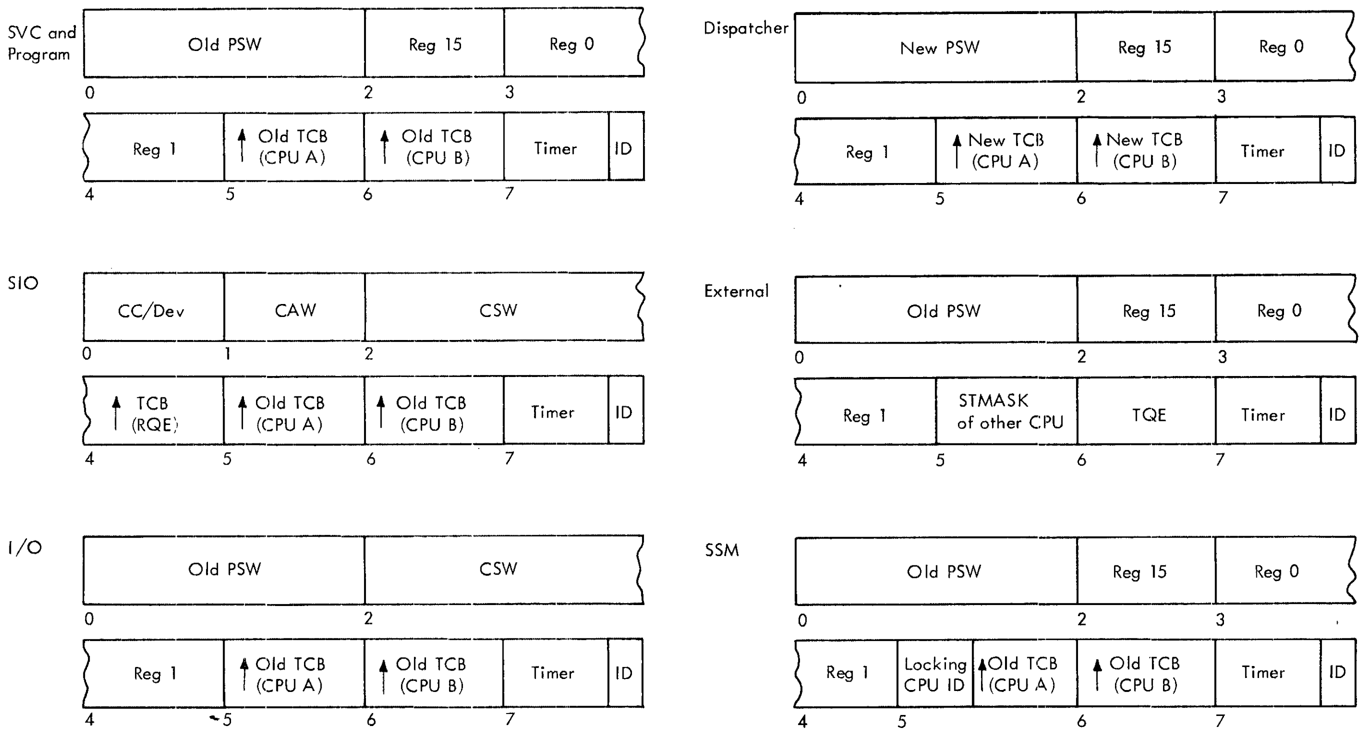


Figure 21. Trace Table Entries (MVT with Model 65 multiprocessing)

NOTES

Section 2: Interpreting Dumps

How are ABEND dumps invoked? What does information in a SNAP dump mean? What useful facts can be gleaned from an indicative dump? Where are key tables and control blocks in a stand-alone dump?

These and similar debugging questions are answered in this section of the manual. Topics comprising Section 2 describe each of the debugging facilities introduced earlier -- what information they provide, where to find this information , and how to apply it.

The introduction to this section describes a general procedure for debugging with a dump. Subsequent topics deal with

- ABEND/SNAP dumps issued by systems with PCP and MFT.
- ABEND/SNAP dumps issued by systems with MVT.
- Indicative dumps.
- Storage Image dumps.
- Stand-alone dumps.

Each topic includes instructions for invoking the dump, a detailed description of the dump's contents, and a guide to using the dump, with specific instructions for following the general debugging procedure.

General Debugging Procedure

The first facts you must determine in debugging with an operating system dump are the cause of the abnormal termination and whether it occurred in a system routine or a problem program. To aid you in making these determinations, ABEND, SNAP, and indicative dumps provide two vital pieces of information -- the completion code and the active RB queue. Similar information can be obtained from a storage image dump or a stand-alone dump by analyzing PSWs and re-creating an active RB queue.

A completion code is printed at the top of ABEND, SNAP, and indicative dumps. It consists of a system code and a user code. The system code is supplied by the control program and is printed as a 3-digit hexadecimal number. The user code is the code you supplied when you issued your own ABEND macro instruction; it is printed as a 4-digit decimal number. If the dump shows a user code, the error is in your program, and the completion code should lead you directly to the source of error. Normally, however, a system code will be listed; this indicates that the operating system issued the ABEND. Often the system completion code gives enough information for you to determine the cause of the error. The explanations of system completion codes, along with a short explanation of the action to be taken by the programmer to correct the error, are contained in the publication IBM System/360 Operating System: Messages and Codes, GC28-6631.

To locate the load module that had control at the time the dump was issued, find the RB associated with the module. If the dump resulted from an ABEND or SNAP macro instruction, the third most recent RB on the queue represents the load module that had control. The most recent and second most recent RBs represent the ABDUMP and ABEND routines, respectively. Storage image dumps and stand-alone dumps contain PSW information that can be used to identify the load module in control.

Once you have located the RB or load module, look at its name. If it does not have a name, it is probably an SVRB for an SVC routine, such as one resulting from a LINK, ATTACH, XCTL or LOAD macro instruction. To find the SVC number, look at the last three digits of the resume PSW in the previous RB on the queue. If a previous RB does not exist, the RB in question is an SVRB for a routine invoked by an XCTL macro instruction. Register 15 in the extended save area of the RB gives a pointer to a parameter list containing the name of the routine that issued the XCTL.

If the RB does not bear the name of one of your load modules, either an RB was overlaid or termination occurred during execution of a system routine. The first three characters of the name identify the system component; Appendix C contains a list of component names to aid you in determining which load module was being executed.

If the RB bears the name of one of your load modules, you can be reasonably certain that the source of the abnormal termination lies in your object code. However, an access method routine may be at fault. This possibility arises because your program branches to access method routines through a supervisor-assisted linkage, instead of invoking them. Thus, an access method routine is not represented on the active RB queue. To ascertain whether an access method routine was the source of the abnormal termination, you must examine the resume PSW field in the RB. If the last 3 bytes in this field point to a main storage address outside your program, check the load list to see if an access method routine is loaded at that address. If it is, you can assume that it, and not your program, was the source of abnormal termination.

Abnormal Termination in System Routines: By analyzing the RB's name field or the SVC number in the previous RB, you can determine which system load module requested the termination. If the RB has a system module name, the first three characters tell you the name of the system component. The remaining characters in the name identify the load module in error.

Remember, although a system routine had control when the dump was taken, a problem program error may indirectly have been at fault. Such a situation might result from an incorrectly specified macro instruction,an FQE modified inadvertently, a request for too much storage space, a branch to an invalid storage address, etc. To determine the function of the load module that had control, consult Appendix C. With its function in mind, the completion code together with an examination of the trace table may help you to uncover which instruction in the problem program incorrectly requested a system function.

Program Check Interruptions in Problem Programs: If you have determined from the completion code or PSWs and evaluation of the RB queue that the dump resulted from a program check in your problem program, examine the status of your program in main storage. (If you have received only an indicative dump, you must obtain either an ABEND/SNAP dump or a stand-alone dump at this point.) Locate your program using pointers in the RB. If its entry point does not coincide with the lower boundary of the program, you can find the lower boundary by adding 32(20) to the address of the RB (systems with PCP and MFT). The RB's size field gives the number of doublewords occupied by the RB, the program, and associated supervisor work areas. ABEND/SNAP dumps with PCP and MFT have the storage boundaries of the problem program calculated and printed.

Next, locate the area within your program that was executed immediately prior to the dump. To do this, you must examine the program check old PSW. Pertinent information in this PSW includes:

Bits 12-15:	AMWP bits
Bits 32,33:	Instruction length in halfwords.
Bits 40-63:	Instruction address

A useful item of information in the PSW is the P bit of the AMWP bits (bits 12-15). If the P bit is on, the PSW was stored while the CPU was operating in the problem program state. If it is off, the CPU was operating in the supervisor state.

Find the last instruction executed before the dump was taken by subtracting the instruction length from the instruction address. This gives you the address of the instruction that caused the termination. If the source program was written in a higher level language, you must evaluate the instructions that precede and follow the instruction at fault to determine their function. You can then relate the function to a statement in the source program.

Other Interruptions in Problem Programs: If the completion code or PSWs and the active RB queue indicate a machine check interruption, a hardware error has occurred. Call your IBM Field Engineering representative and show him the dump.

If an external interruption is indicated, with no other type of interruption, the dump probably was taken by the operator. Check with him to find out why the dump was taken at this point. The most likely reasons are an unexpected wait or a program loop. If a trace table exists, examine it for the events preceding the trouble or, if the trace table was made ineffectual by a program loop, resubmit the job and take a dump at an earlier point in the program. You may want to consider using the TESTRAN facility to find where the program loop occurred.

The remaining causes of a dump are an error during either execution of an SVC or an I/O interruption. In either case, examine the trace table. Entries in the table tell you what events occurred leading up to termination. From the sequence of events, you should be able to determine what caused a dump to be taken. From here, you can turn to system control blocks and save areas to get specific information. For example, you can find the sense information issued as a result of a unit check in the UCB, a list of the open data sets from the DEB chain, the CCW list from the IOB, the reason for an I/O interrupt in the status portion of the CSW, etc.

Debugging Procedure Summary

1. Look at the completion code or PSW printouts to find out what type of error occurred. Common completion codes and causes are explained in Appendix B.

2. Check the name of the load module that had control at the time the dump was taken by looking at the active RB's.

3. If the name identifies a system routine, proceed to step 4. If the name identifies a problem program and the completion code or PSW indicates a program check, proceed to step 6. If the name identifies a problem program, and the completion code or PSW indicates other than a program check, proceed to step 10.

----

4. Find the function of the system routine using Appendix C.

5. If the dump contains a trace table, begin at the most recent entry and proceed backward to locate the most recent SVC entry indicating the problem state. From this entry, proceed forward in the table, examining each entry for an error that could have caused the system routine to be terminated.

----

6. If the name identifies one of your load modules, check the instruction address and the load list to see if an access method routine last had control. If so, return to step 4.

7. Locate your program in the dump.

8. Locate the last instruction executed before the dump.

9. Examine the instruction and, if the program was written in a high-level language, the instructions around it for a possible error in object code.

----

10. If a machine check interruption is indicated, call your IBM Field Engineering representative.

11. If only an external interruption is indicated, ask the operator why he took the dump. Resubmit the job and take a dump at the point where trouble first occurred.

12. Examine the trace table, if one is present, for events leading up to the termination. Use trace table entries and/or information in system control blocks and save areas to isolate the cause of the error.

ABEND/SNAP Dump (Systems With PCP and MFT)

ABEND/SNAP dumps for systems with PCP and MFT are discussed together because they are nearly identical in format. System differences in the contents of the dumps are shaded for easy recognition. Debugging instructions for the dumps are discussed later, in the guide to using the dump.

ABEND/SNAP storage dumps are issued whenever the control program or problem program issues an ABEND or SNAP macro instruction, or the operator issues a CANCEL command requesting a dump, and proper dump data sets have been defined. However, in the event of a system failure, if a SYS1.DUMP data set has been defined and is available, a full storage image dump will be provided, as explained in the section headed "Storage Image Dump."

Since, in an MFT with subtasking system, subtasks may be created, you may receive one or more partial dumps in addition to the complete dump of the task that caused the abnormal termination. A complete dump includes a printout of all control information related to the terminating task, and the nucleus and all allocated storage within the partition in which the abending task resided. A partial dump of a task related to the terminating task includes only control information. The partial dump is identified by either ID=001 or ID=002 printed in the first line of the dump. Figure 22 is a copy of the first few pages of a complete ABEND dump of an MFT system with subtasking. It illustrates some of the key areas on an ABEND dump, as issued by systems with PCP and MFT. Those portions of the dump that would only appear on a dump of a subtasking system are noted in the later discussions as appearing only in a dump of an MFT with subtasking system.

For a discussion of a formatted ABEND dump using the telecommunications access method (TCAM) in an MFT environment, see IBM System/360 Operating System: TCAM Program Logic Manual, GY30-2029. References to other TCAM debugging aids are found in Appendix H.

Invoking an ABEND/SNAP Dump (PCP,MFT)

ABEND dumps are produced as a result of an ABEND macro instruction, issued either by a processing program or an operating system routine. The macro instruction requires a DD statement in the input stream for each job step that is subject to abnormal termination. This DD statement must be identified by one of the special ddnames SYSABEND or SYSUDUMP. SYSABEND results in edited control information, the system nucleus, the trace table, and a dump of main storage; SYSUDUMP excludes the nucleus and the trace table. In the event of a system failure, the Damage Assessment routine (DAR) attempts to write a storage image dump to the SYS1.DUMP data set. A full explanation of storage image dumps may be found in the section headed "Storage Image Dump."

SNAP Dumps result from a problem program issuing a SNAP macro instruction. The contents of a SNAP dump vary according to the operands specified in the SNAP macro instruction. SNAP dumps also require a DD statement in the input stream. This DD statement has no special characteristics except that its ddname must not be SYSABEND or SYSUDUMP. The processing program must define a DCB for the snapshot data set. The DCB macro instruction must contain, in addition to the usual DCB requirements, the operands DSORG=PS, RECFM=VBA, MACRF=(W), BLKSIZE=882 or 1632, and LRECL=125. In addition, the DCB must be opened before the first SNAP macro instruction is issued.

Reference: The SNAP and DCB macro instructions are discussed in the publication Supervisor and Data Management Macro Instructions.

Device and Space Considerations: DD statements for ABEND/SNAP dumps, must contain parameters appropriate for a basic sequential (BSAM) data set. Data sets can be allocated to any device supported by the basic sequential access method. There are several ways to code these DD statements depending on what type of device you choose and when you want the dump printed.

If you wish to have the dump printed immediately, code a DD statement defining a printer data set.

```
//SYSABEND DD UNIT=1443,DCB=(...
```

If your installation operates under a system with PCP or MFT, and a printer is associated with the SYSOUT class, you can also obtain immediate printing by routing the data set through the output stream.

```
//SNAPDUMP DD SYSOUT=A,DCB=(...
```

This type of request is the easiest, most economical way to provide for a dump. All other DD statements result in the tying up of an output unit or delayed printing of the dump.

If you wish to retain the dump, you can keep or catalog it on a direct access or tape unit. The last step in the pertinent job can serve several functions: to print out key data sets in steps that have been

abnormally terminated, to print an ABEND or SNAP dump stored in an earlier step, or to release a tape volume or direct access space acquired for dump data sets. Conditional execution of the last step can be established through proper use of the COND parameter and its subparameters, EVEN and ONLY, on the EXEC statement.

Direct access space should be requested in units of average block size rather than in cylinders (CYL) or tracks (TRK). If abnormal termination occurs and the data set is retained, the tape volume or direct access space should be released (DELETE in the DISP parameter) at the time the data set is printed.

```
 * ABDUMP REQUESTED *


JOB ATHEOT24          STEP STEP            TIME 000737   DATE 99366                                    PAGE 0001

COMPLETION CODE       USER = 0123

INTERRUPT AT C6EF5A

PSW AT ENTRY TO ABEND  0015000D 4006EF5A

TCB     01CB20  RB   0007FC58   PIE     00000000   DEB  0007F78C   TIOT 0007FDB0   CMP    8000007B   TRN  00000000
                MSS  0001CC58   PK/FLG 10B10408    FLG  000001FB   LLS  00000000   JLB    0007FF78   JST  000055D8
                FSA  1506EBF8   TCB     0001D0A0   TME  0001CBD8   PIB  E0012420   NTC    00000000   OTC  0001C0E0
                LTC  00000000   IQE     00000000   ECB  0006EE1C   XTCB 00000000   LP/FL  FB050000   RESV 00000000
                STAE 00000000   TCT     00000000   USER 00000000   DAR  00000000   RESV   00000000   JSCB 00000000


ACTIVE RBS

PRB     06EE28  NM TATHB10G    SZ/STAB 003D20D0    USE/EP 0106EE48    PSW 0015000D 4006EF5A    Q 000000    WT/LNK 0001CB20

SVRB    07FD20  NM SVC-601C    SZ/STAB 0012D062    USE/EP 00007B78    PSW FF040033 50007D20    Q 900390    WT/LNK 0006EE28
                RG 0-7     000002A0     80000078     00000000     00080000     0007FE48     00000098     000055D8     0007FC30
                8-15-7     0006EE60     0007FF78     0007FFB0     0007FFF8     4006EE4E     0006EE60     00009848     00000000

SVRB    07FC58  NM SVC-A05A    SZ/STAB 000CD062    USE/EP 00007B78    PSW FF04000E 8001E7EC    Q F803F8    WT/LNK 0007FD20
                RG 0-7     0007F7E8     0007FD80     40007B7A     000097F8     0001CB20     0007FD20     0006F230     000055D8
                8-15-7     0007F7E8     0006F296     0001CC56     0000225C     0001CB20     0006F230     90007CBC     0001E7C8

JOB PACK AREA QUEUE

LPRB 06ECA8  NM TATHA10G    SZ/STAB 002F20D0    USE/EP 0106ECC8    PSW FF15000E 8006ED9C    Q 000000    WT/LNK 0101CDE0

LPRB 06EE28  NM TATHB10G    SZ/STAB 003D20D0    USE/EP 0106EE48    PSW 0015000D 4006EF5A    Q 000000    WT/LNK 0001CB20

LPRB 06F018  NM TATHC10G    SZ/STAB 00122090    USE/EP 0106F038    PSW 0004000D 40006AE4    Q 000000    WT/LNK 0001CC80

LPRB 06F0B0  NM TATHD10G    SZ/STAB 001B20D0    USE/EP 0106F0D0    PSW FF150001 4006F16C    Q 000000    WT/LNK 0101D0A0

LPRB 06F190  NM TATHE10G    SZ/STAB 001320D0    USE/EP 0106F1B0    PSW FF150001 4006F21E    Q 000000    WT/LNK 0101CF40

P/P STORAGE BOUNDARIES 0006E800 TO 00080000

FREE AREAS     SIZE

  06EB90       00000060
  06EC50       00000050
  06F5B8       0000FC58
  07F668       C0000098
  07F7D8       00000010
  07F840       C0000228
  07FB90       000000C0
  C7FEE8       C000C018

GOTTEN CORE    SIZE

  07F210       00000380
  06F310       C00002A8
  07FC50       0000C068
  06F228       000000E8
  07F590       00000008
  07F5F0       C0000008
  07FD18       00000098
  07F700       0000C060
  07F760       00000078
  07FA68       C0000060
  07FAC8       00000078
```

Figure 22A. Sample of an ABEND Dump (PCP, MFT)

```
  * * * A B D U M P   R E Q U E S T E D * * *

 *ccccccc...

JOB cccccccc          STEP cccccccc          TIME dddddd   DATE ddddd                                                    PAGE dddd

COMPLETION CODE       SYSTEM = hhh (or USER = dddd)

cccccc...

INTERRUPT AT hhhhhh

PSW AT ENTRY TO ABEND (SNAP) hhhhhhhh hhhhhhhh
```

* * * A B D U M P   R E Q U E S T E D * * *
identifies the dump as an ABEND or SNAP dump.

*ccccccc.....
is omitted or is one or more of the following:

*CORE NOT AVAILABLE, LOC. hhhhhhhhhhhh TAKEN...
indicates that the ABDUMP routine confiscated storage locations hhhhhh through hhhhhh because not enough storage was available. This area is printed under P/P STORAGE, but can be ignored because the problem program originally in it was overlaid during the dumping process.

*MODIFIED, /SIRB/DEB/LLS/ARB/MSS...
indicates that the one or more queues listed were destroyed or their elements dequeued during abnormal termination:

- SIRB -- system interruption request block queue. One or more SIRB elements were found in the active RB queue: these elements are always dequeued during dumping.
- DEB -- DEB queue. If the first message also appeared, either a DEB or an associated DCB was overlaid.
- LLS -- load list. If the first message also appeared, one or more loaded RBs were overlaid.
- ARB -- active RB queue. If the first message also appeared, one or more RBs were overlaid.
- MSS -- boundary box queue. One or more MSS elements were dequeued, but an otherwise valid control block was found in the free area specified by an MSS element.

*FOUND ERROR IN /DEB/LLS/ARB/MSS...
indicates that one or more of the following contained an error:

- DEB: data extent block
- LLS: load list
- ARB: active RB
- MSS: boundary box

This message appears with either the first or second message above. The error could be: improper boundary alignment, control block not within storage assigned to the program being dumped, or an infinite loop (300 times is the maximum for this test). For an MSS block, 4 other errors could also be found: incorrect descending sequence (omitting loop count), overlapping free areas, free area not entirely within the storage assigned to the program being dumped, or count in count field not a multiple of 8.

JOB cccccccc
is the job name specified in the JOB statement.

STEP cccccccc
is the step name specified in the EXEC statement for the problem program being dumped.

TIME dddddd
is the hour (first 2 digits), minute (second 2 digits), and second (last 2 digits) when the ABDUMP routine began processing.

DATE ddddd
is the year (first 2 digits) and day of the year (last 3 digits). For example, 67352 would be December 18, 1967.

PAGE dddd
is the page number. Appears at the top of each page.

COMPLETION CODE SYSTEM=hhh or COMPLETION CODE USER=dddd
is the completion code supplied by the control program (SYSTEM=hhh) or the problem program (USER=dddd). Either SYSTEM=hhh or USER=dddd is printed, but not both. Common completion codes are explained in Appendix B.

cccccc...
explains the completion code or, if a program interruption occurred:
PROGRAM INTERRUPTION ccccc... AT LOCATION hhhhhh,

where ccccc is the program interruption cause -- OPERATION, PRIVILEGED OPERATION, EXECUTE, PROTECTION, ADDRESSING, SPECIFICATION, DATE, FIXED-POINT OVERFLOW, FIXED-POINT DIVIDE, DECIMAL OVERFLOW, DECIMAL DIVIDE, EXPONENT OVERFLOW,EXPONENT UNDERFLOW, SIGNIFICANCE, or FLOATING-POINT DIVIDE; and hhhhhh is the starting address of the instruction being executed when the interruption occurred.

INTERRUPT AT hhhhhh
is the address of next instruction to be executed in the problem program. It is obtained from the resume PSW of the PRB or LPRB in the active RB queue at the time abnormal termination was requested.

PSW AT ENTRY TO ABEND hhhhhhhh hhhhhhhh or
PSW AT ENTRY TO SNAP hhhhhhhh hhhhhhhh
is the PSW for the problem or control program that had control when abnormal termination was requested or when the SNAP macro instruction was executed.

```
TCB     hhhhhh RB     hhhhhhhh   PIE     hhhhhhhh   DEB  hhhhhhhh   TIOT hhhhhhhh   CMP   hhhhhhhh   TRN  hhhhhhhh
               MSS    hhhhhhhh   PK/FLG  hhhhhhhh   FLG  hhhhhhhh   LLS  hhhhhhhh   JLB   hhhhhhhh   JST  hhhhhhhh
               RG 0-7     hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh
               RG 8-15    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh
               FSA    hhhhhhhh   TCB     hhhhhhhh   TME  hhhhhhhh   PIB  hhhhhhhh   NTC   hhhhhhhh   OTC  hhhhhhhh
               LTC    hhhhhhhh   IQE     hhhhhhhh   ECB  hhhhhhhh   XTCB hhhhhhhh   LP/FL hhhhhhhh   RESV hhhhhhhh
               STAE   hhhhhhhh   TCT     hhhhhhhh   USER hhhhhhhh   DAR  hhhhhhhh   RESV  hhhhhhhh   JSCB hhhhhhhh
```

TCB hhhhhh
is the starting address of the TCB.

RB hhhhhhhh
is the TCBRBP field (bytes 0 through 3): starting address of the active RB queue and, consequently, the most recent RB on the queue (usually ABEND's RB).

PIE hhhhhhhh
is the TCBPIE field (bytes 4 through 7): starting address of the program interruption element (PIE) for the task.

DEB hhhhhhhh
is the TCBDEB field (bytes 8 through 11): starting address of the DEB queue.

TIOT hhhhhhhh
is the TCBTIO field (bytes 12 through 15): starting address of the TIOT.

CMP hhhhhhhh
is the TCBCMP field (bytes 16 through 19): task completion code in hexadecimal. System codes are shown in the third through fifth digits and user codes in the sixth through eighth.

TRN hhhhhhhh
is the TCBTRN field (bytes 20 through 23): starting address of control core (table) for controlling testing of the task by TESTRAN.

MSS hhhhhhhh
is the TCBMSS field (bytes 24 through 27): starting address of the main storage supervisor's boundary box.

PK/FLG hhhhhhhh
contains, in the first 2 digits, the TCBPKF field (byte 28): protection key.

FLG hhhhhhhh
contains, in the first 4 digits, the last 2 bytes of the TCBFLGS field (bytes 32 and 33): last 2 flag bytes.

contains, in the next 2 digits, the TCBLMP field (byte 34): in systems the timer is not being used, contains no meaningful information; in SVRB for a type 2 SVC routine, the first 4 bytes contain the TTR of the load module in the SVC library, and the last 4 bytes contain the SVC number in signed, unpacked decimal.

SZ/STAB hhhhhhhh
: contains in the first 4 digits, the XRBSZ field (bytes 8 and 9): number of contiguous doublewords in the RB, the program (if applicable), and associated supervisor work areas.

  contains in the last 4 digits, the XSTAB field (bytes 10 and 11): flag bytes.

USE/EP hhhhhhhh
: contains, in the first 2 digits, the XRBUSE field (byte 12): use count.

  contains, in the last 6 digits, the XRBEP field (bytes 13 through 15): address of entry point in the associated program.

PSW hhhhhhhh hhhhhhhh
: is the XRBPSW field (bytes 16 through 23): resume PSW.

Q hhhhhh
: is the last 3 bytes of the XRBQ field (bytes 25 through 27): in PRB and LPRB, starting address of an LPRB for an entry identified by an IDENTIFY macro instruction; in IRB, starting address of a request element; in SVRB for a type 3 or 4 SVC, size of the program in bytes.

WT/LNK hhhhhhhh
: contains, in the first 2 digits, the XRBWT field (byte 28): wait count.

  contains, in the last 6 digits, the XRBLNK field (bytes 29 through 31): primary queuing field. It is the starting address of the previous RB for the task or, in the first RB to be placed on the queue, the starting address of the TCB.

RG 0-7 and RG 8-15
: is the XRBREB field (bytes 32 through 95 in IRBs and SVRBs): contents of general registers 0 through 15 stored in the RB. These 2 lines do not appear for PRBs, LPRBs, and LRBs.

```
LOAD LIST
cccc hhhhhh  NM cccccccc   SZ/STAB hhhhhhhh  USE/EP hhhhhhhh  PSW hhhhhhhh hhhhhhhh   Q hhhhhh   WT/LNK hhhhhhhh
```

LOAD LIST
: identifies the next lines as the contents of the load list queued to the TCB.

cccc hhhhhh
: indicates the RB type and its starting address.

  The RB types are:

  LRB — Loaded request block
  LPRB — Loaded program request block
  D-LPRB — Dummy loaded program request block. (Present if the resident reenterable load module option was selected in MFT).

NM cccccccc
: is the XRBNM field (bytes 0 through 7): program name.

SZ/STAB hhhhhhhh
: contains, in the first 4 digits, the XRBSZ field (bytes 8 and 9): number of contiguous doublewords for the RB, the program (if applicable), and associated supervisor work areas.

  contains, in the last 4 digits, the XSTAB field (bytes 10 and 11): flag bytes.

USE/EP hhhhhhhh
: contains, in the first 2 digits, the XRBUSE field (byte 12): use count.

  contains, in the last 6 digits, the XRBEP field (bytes 12 through 15): address of entry point in the program.

PSW hhhhhhhh hhhhhhhh
is the XRBPSW field (bytes 16 through 23): resume PSW.

Q hhhhhh
is the last 3 bytes of the XRBQ field (bytes 25 through 27): in LPRB, starting address of an LPRB for an entry identified by an IDENTIFY macro instruction; in LRB, unused.

WT/LNK hhhhhhhh
contains, in the first 2 digits, the XRBWT field (byte 28): wait count.

contains, in the last 6 digits, the XRBLNK field (bytes 29 through 31): primary queuing field for LRBs and LPRBs also on the active RB queue. It points to the previous RB for the task or, in the oldest RB in the queue, back to the TCB.

```
JOB PACK AREA QUEUE

cccc hhhhhh  NM cccccccc   SZ/STAB hhhhhhhh  USE/EP hhhhhhhh  PSW hhhhhhhh hhhhhhhh  Q hhhhhh  WT/LNK hhhhhhhh
cccc hhhhhh  NM cccccccc   SZ/STAB hhhhhhhh  WTL    hhhhhhhh  REQ hhhhhhhh  TLPRB hhhhhhhh
cccc hhhhhh  NM cccccccc   SZ/STAB hhhhhhhh  USE/EP hhhhhhhh  PSW hhhhhhhh hhhhhhhh  Q hhhhhh  WT/LNK hhhhhhhh
```

JOB PACK AREA QUEUE (MFT with subtasking only)
identifies the next lines as the contents of the job pack area queue originating in the partition information block (PIB).

cccc hhhhhh
indicates the RB type and its starting address.

The RB types are:

FRB   Finch request block
LRB   Loaded request block
LPRB Loaded program request block

NM cccccccc
is the XRBNM field (bytes 0 through 7): Program name.

SZ/STAB hhhhhhhh
contains, in the first 4 digits, the XRBSZ field (bytes 8 and 9): number of contiguous doublewords for the RB, the program (if applicable), and associated supervisor work areas.

contains, in the last 4 digits, the XSTAB field (bytes 10 and 11): flag bytes.

USE/EP hhhhhhhh (LPRB, LRB Only)
contains, in the first 2 digits, the XRBUSE field (byte 12): use count.

contains, in the last 6 digits, the XRBEP field (bytes 13 through 15): address of entry point in the program.

WTL hhhhhhhh (FRB Only)
is the XRWTL field of the FRB (bytes 12 through 15): address of the most recent wait list element (WLE) on the WLE queue.

PSW hhhhhhhh hhhhhhhh (LPRB, LRB Only)
is the XRBPSW field (bytes 16 through 23): resume PSW.

REQ hhhhhhhh (FRB Only)
is the XRREQ field of the FRB (bytes 16 through 19): address of the TCB of the requesting task.

TLPRB hhhhhhhh (FRB Only)
is the XRTLPRB field of the FRB (bytes 20 through 23): address of the LPRB built by the Finch routine for the requested program.

Q hhhhhh (LRB, LPRB Only)
is the last 3 bytes of the XRBQ field (bytes 25 through 27):

- in an LPRB, the starting address of an LPRB for an entry identified by an IDENTIFY macro instruction.
- in an LRB, unused.

WT/LNK hhhhhhhh (LRB, LPRB Only)
contains, in the first 2 digits, the XRBWT field (byte 28): wait count.

contains, in the last 6 digits (bytes 29 through 31): primary queuing field for RBs. These RBs may be queued either on the job pack area queue or on the active RB queue. It points to the previous RB for the task or, in the oldest RB on the queue, back to the TCB.

```
P/P STORAGE BOUNDARIES hhhhhhhh TO hhhhhhhh

FREE AREAS     SIZE

  hhhhhh      hhhhhhhh

GOTTEN CORE    SIZE

  hhhhhh      hhhhhhhh

SAVE AREA TRACE

cccccccc WAS ENTERED VIA LINK (CALL) ddddd AT EP ccccc...

SA    hhhhhh  WD1 hhhhhhhh   HSA hhhhhhhh   LSA hhhhhhhh   RET hhhhhhhh   EPA hhhhhhhh   R0  hhhhhhhh
              R1  hhhhhhhh   R2  hhhhhhhh   R3  hhhhhhhh   R4  hhhhhhhh   R5  hhhhhhhh   R6  hhhhhhhh
              R7  hhhhhhhh   R8  hhhhhhhh   R9  hhhhhhhh   R10 hhhhhhhh   R11 hhhhhhhh   R12 hhhhhhhh

INCORRECT BACK CHAIN

PROCEEDING BACK VIA REG 13
```

P/P STORAGE BOUNDARIES hhhhhhhh TO hhhhhhhh
: gives the addresses of the lower and upper boundaries of a main storage area assigned to the task. This heading is repeated for every noncontiguous block of storage owned by the task.

FREE AREAS SIZE

hhhhhh hhhhhh
.
.
.
hhhhhh hhhhhh
: are the starting addresses of free areas and the size, in bytes, of each area contained within the P/P STORAGE BOUNDARIES field listed above.

GOTTEN CORE SIZE

hhhhhh hhhhhhhh
.
.
.
hhhhhh hhhhhhhh
: (Printed only in a dump of a system with the MFT with subtasking option). These figures represent the starting addresses of the gotten areas (those areas obtained for a subtask through a supervisor issued GETMAIN macro instruction), and the size, in bytes, of each area contained within the P/P STORAGE BOUNDARIES field listed above. If main storage hierarchy support is included in the system, the values in this field can address storage in either hierarchy 0 or hierarchy 1, or both.

SAVE AREA TRACE
: identifies the next lines as a trace of the save areas for the program.

cccccccc WAS ENTERED
: is the name of the program that stored register contents in the save area. This name is obtained from the RB.

VIA LINK (CALL) ddddd
: indicates the macro instruction (LINK or CALL) used to give control to the next lower level module, and is the ID operand, if it was specified, of the LINK or CALL macro instruction.

AT EP ccccc...
: is the entry point identified, which appears only if it was specified in the SAVE macro instruction that filled the save area.

SA hhhhhh
: is the starting address of the save area.

WD1 hhhhhhhh
: is the first word of the save area: use of this word is optional.

HSA hhhhhhhh
: is the second word of the save area: starting address of the save area in the next higher level module. In the first save area in a job step, this word contains zeros. In all other save areas, this word must be filled.

LSA hhhhhhhh
: is the third word of the save area (register 13): starting address of the save area in the next lower level module.

RET hhhhhhhh
: is the fourth word of the save area (register 14): return address. Optional.

EPA hhhhhhhh
: is the fifth word of the save area (register 15): entry point to the invoked module. Optional.

R0 hhhhhhhh R1 hhhhhhhh ... R12 hhhhhhhh
: are words 6 through 18 of the save area (registers 0 through 12): contents of registers 0 through 12 immediately after the linkage for the module containing the save area.

INCORRECT BACK CHAIN
: indicates that the following lines may not be a save area because the second word in this area does not point back to the previous save area in the chain.

PROCEEDING BACK VIA REG 13
: indicates that the next 2 save areas are (1) the save area in the lowest level module, followed by (2) the save area in the next higher level module. The lowest save area is assumed to be the save area pointed to by register 13. These 2 save areas appear only if register 13 points to a full word boundary and does not contain zeros.

```
DATA SETS

***** N O T   F O R M A T T E D *****

cccccccc      UCB   ddd   hhhhhh      DEB hhhhhh      DCB hhhhhh

**D/S FORMATTING TERMINATED**
```

DATA SETS
: indicates that the next lines present information about the data sets for the task. For unopened data sets, only the ddname and UCB information are printed.

N O T   F O R M A T T E D
: indicates that the abnormal termination dump routine confiscated storage (indicated by *CORE NOT AVAILABLE, LOC. hhhhhh-hhhhhh TAKEN); because DCBs may have been overlaid, data set information is not presented.

cccccccc
: is the name field (ddname) of the DD statement.

UCB ddd hhhhhh
: is the unit to which the data set was assigned, and the starting address of the UCB for that unit. If the data set was assigned to several units, the additional units are identified on following lines.

DEB hhhhhh
: is the starting address of the DEB for the data set. Appears only for open data sets.

DCB hhhhhh
: is the starting address of the DCB for the data set. Appears only for open data sets.

D/S FORMATTING TERMINATED
: indicates that no more data set information is presented because a DCB is incorrect, possibly because a program incorrectly modified it.

```
TRACE TABLE - STARTING WITH OLDEST ENTRY

dddd     I/O ddd     PSW  hhhhhhhh hhhhhhhh                      CSW        hhhhhhhh hhhhhhhh
dddd     SIO ddd     CC = d                   CAW  hhhhhhhh      OLD CSW    hhhhhhhh hhhhhhhh (or CSW STATUS hhhh)
dddd     SVC ddd     PSW  hhhhhhhh hhhhhhhh   RG 0 hhhhhhhh      RG 1       hhhhhhhh
```

TRACE TABLE -- STARTING WITH OLDEST ENTRY
: identifies the next lines as the contents of the trace table. Each entry is presented on one line. The types of entries are:

I/O Input/output interruption entry

SIO Start input/output (SIO) entry

SVC Supervisor call (SVC) interruption entry

dddd
: is the number assigned to each entry. The oldest entry receives the number 0001.

I/O ddd
: is the channel and unit that caused the input/output interruption.

PSW hhhhhhhh hhhhhhhh
: is the program status word that was stored when the input/output interruption occurred.

CSW hhhhhhhh hhhhhhhh
: is the channel status word that was stored when the input/output interruption occurred.

SIO ddd
: is the device specified in the SIO instruction.

CC=d
: is the condition code resulting from execution of the SIO instruction. Zero indicates a successful start.

CAW hhhhhhhh
: is the channel address word used by the SIO instruction.

OLD CSW hhhhhhhh hhhhhhhh
: is the channel status word stored during execution of an SIO operation. It appears when CC is nct equal to 1.

CSW STATUS hhhh
: is the status portion of the channel status word stored during execution of an SIO instruction. Appears when CC is equal to 1.

SVC ddd
: is the SVC instruction's operand.

PSW hhhhhhhh hhhhhhhh
: is the PSW stored during the SVC interruption. (After release 11, an F in the fifth digit of the first word identifies the entry as representing a task switch.)

RG 0 hhhhhhhh
: is the contents of register 0 as passed to the SVC routine.

RG 1 hhhhhhhh
: is the contents of register 1 as passed to the SVC routine.

```
REGS AT ENTRY TO ABEND (SNAP)

FLTR 0-6          hhhhhhhhhhhhhhhh       hhhhhhhhhhhhhhhh            hhhhhhhhhhhhhhhh       hhhhhhhhhhhhhhhh

REGS 0-7          hhhhhhhh   hhhhhhhh    hhhhhhhh   hhhhhhhh         hhhhhhhh   hhhhhhhh    hhhhhhhh   hhhhhhhh
REGS 8-15         hhhhhhhh   hhhhhhhh    hhhhhhhh   hhhhhhhh         hhhhhhhh   hhhhhhhh    hhhhhhhh   hhhhhhhh
```

REGS AT ENTRY TO ABEND or REGS AT ENTRY TO SNAP
: identifies the next 3 lines as the contents of the floating point and general registers when the abnormal termination routine received control in response to an ABEND macro instruction or when the SNAP routine received control in response to a SNAP macro instruction.

FLTR 0-6
: is the contents of floating point registers 0, 2, 4, and 6.

REGS 0-7
: is the contents of general registers 0 through 7.

REGS 8-15
: is the contents of general registers 8 through 15.

```
NUCLEUS

hhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    *cccccccccccccccccccccccccccccccc*
hhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    *cccccccccccccccccccccccccccccccc*
   LINE      hhhhhh      SAME AS ABOVE
hhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    *cccccccccccccccccccccccccccccccc*
   LINES     hhhhhh-hhhhhh      SAME AS ABOVE
hhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    *cccccccccccccccccccccccccccccccc*
P/P STORAGE

hhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    *cccccccccccccccccccccccccccccccc*
hhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    *cccccccccccccccccccccccccccccccc*
hhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    *cccccccccccccccccccccccccccccccc*
   LINES     hhhhhh-hhhhhh      SAME AS ABOVE
hhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    *cccccccccccccccccccccccccccccccc*
          END OF DUMP
```

The content of main storage is given under 2 headings: NUCLEUS and P/P STORAGE. Under these headings, the lines have the following format:

- First entry: the address of the initial byte of main storage contents presented on the line.
- Next 8 entries: 8 full words (32 bytes) of main storage in hexadecimal.
- Last entry (surrounded by asterisks): the same 8 full words of main storage in EBCDIC. Only A through Z, 0 through 9, and blanks are printed; a period is printed for anything else. An exception occurs in the printed lines representing the ABDUMP work area. The contents of the ABDUMP work area during the printing of EBCDIC characters differs from the contents during printing of the hexadecimal characters because a portion of the work area is used to write lines to the printer. This exception should not create any problems since the contents of the ABDUMP work area is of little use in debugging.

The following lines may also appear:

LINES hhhhhhhh-hhhhhhhh SAME AS ABOVE
: are the starting addresses of the first and last line of a group of lines that are identical to the line immediately preceding.

LINE hhhhhh SAME AS ABOVE
: is the starting address of a line that is identical to the line immediately preceding.

NUCLEUS
: identifies the next lines as the contents of the control program nucleus.

P/P STORAGE
: identifies the next lines as the contents of the main storage area assigned to the task (problem program).

END OF DUMP
: indicates that the dump or snapshot is completed.

Guide to Using an ABEND/SNAP Dump (PCP, MFT)

Cause of Abnormal Termination: Evaluate the user (USER Decimal code) or system (SYSTEM=hex code) completion code using Appendix B or the publication Messages and Codes.

Active RB Queue: The first RB shown on the dump represents the oldest RB on the queue. The RB representing the load module that had control when the dump was taken is third from the bottom. The last RB represents the ABDUMP routine, and the second from last, the ABEND routine. The names of load modules represented in the active RB queue are given in the RB field labeled NM in the dump. Names of load modules in SVC routines are presented in the format:

```
NM     SVC-mnnn
```

where m is the load module number (minus 1) in the routine and nnn is the signed decimal SVC number. The last two RBs on an ABEND/SNAP dump will always be SVRBs with edited names SVC-105A (ABDUMP--SVC 51) and SVC-401C (ABEND--SVC 13).

Resume PSW: The resume PSW field is the fourth entry in the first line of each RB printout. It is identified by the subheading PSW. For debugging purposes, the resume PSW of the third RB from the bottom, on the dump, is most useful. The last three characters of the first word give the SVC number or the I/O device address, depending on which type of interruption caused the associated routine to lose control. It also provides the CPU state at the time of the interruption (bit 15), the length of the last instruction executed in the program (bits 32,33), and the address of the next instruction to be executed (bytes 5-8).

Load List and Job Pack Area Queue: The load module that had control at the time of abnormal termination may not contain the instruction address pointed to by the resume PSW. In that case, look at the RBs on the load list and on the job pack area queue (MFT with subtasking). Compare the instruction address with the entry points of each load module (shown in the last 3 bytes of the field labeled USE/EP). The module which contains the instruction pointed to by the resume PSW is the one in which abnormal termination occurred. The name of the load module is indicated in the field labeled NM.

Trace Table: Entries in the trace table reflect SIO, I/O, and SVC interruptions. SIO entries can be used to locate the CCW (through the CAW), which reflects the operation initiated by an SIO instruction. If the SIO operation was not successful, the CSW STATUS portion of the entry will show you why it failed.

I/O entries reflect the I/O old PSW and the CSW that was stored when the interruption occurred. From the PSW, you can learn the address of the device on which the interruption occurred (bytes 2 and 3), the CPU state at the time of interruption (bit 15), and the instruction address where the interruption occurred (bytes 5-8). The CSW provides you with the unit status (byte 4), the channel status (byte 5), and the address of the previous CCW plus 8 (bytes 0-3).

SVC entries provide the SVC old PSW and the contents of registers 0 and 1. The PSW offers you the hexadecimal SVC number (bits 20-31), the CPU mode (bit 15), and the address of the SVC instruction (bytes 5-8). The contents of registers 0 and 1 are useful in that many system macro instructions use these registers for parameter information. Contents of registers 0 and 1 for each SVC interruption are given in Appendix A.

Note: If an ABEND macro instruction is issued by the system when a program check interruption causes abnormal termination, an SVC entry does not appear in the trace table, but is reflected in the PSW at entry to ABEND.

Free Areas: ABEND/SNAP dumps do not print out areas of main storage that are available for allocation. Since the ABEND routine uses some available main storage, the only way you can determine the amount of free storage available when abnormal termination occurred is to re-create the situation and take a stand-alone dump.

MFT Considerations: Dumps issued by systems with MFT include an additional trace table entry for task switches. This entry looks similar to an SVC entry, except that words 3 and 4 of the entry contain the address of the TCBs for the "new" and "old" tasks being performed, respectively. The trace table entries for one particular task are contained between sets of two task switch entries. Word 3 of the beginning task switch entry and word 4 of the ending task switch entry point to the TCB for that task. With release 11 and following releases, task switch entries are identified by a fifth digit of 'F'.

Note: To find all the entries for the terminated task, on a dump issued prior to release 11, obtain the TCB addresses under the TCB heading of the dump and scan the trace table under words 3 and 4 for these addresses. Then enclose the areas that begin with an entry having the TCB address in word 3, and end with an entry having the same TCB address in word 4. If words 3 and 4 contain the same address, disregard the task switch entry.

ABEND/SNAP Dump (Systems With MVT)

MVT dumps differ from PCP and MFT dumps in the addition of detailed main storage control information, the omission of a complete main storage dump, and the omission of a trace table in ABEND dumps. MVT dumps occur immediately after an abnormal termination, provided an ABEND or SNAP macro instruction was issued and proper dump data sets were defined. However, if a system failure has occurred and a SYS1.DUMP data set has been defined and is available, a full storage image dump is provided, as explained in the section headed "Storage Image Dump."

With MVT's subtask creating capability, you may receive one or more partial dumps in addition to a complete dump of the task that caused abnormal termination. A complete dump includes all control information associated with the terminating task and a printout of the load modules and subpools used by the task. A partial dump of a task related to the terminating task includes only control information. A partial dump is identified by either ID=001 or ID=002 printed in the first line of the dump. Figure 24 shows the key areas of a complete dump.

In systems with MVT, you can effect termination of a job step task upon abnormal termination of a lower level task. To do this, you must either terminate each task upon finding an abnormal termination completion code issued by its subtask or pass the completion code on to the next higher level task.

For a discussion of a formatted ABEND dump using the telecommunications access method (TCAM) in an MVT environment, see IBM System/360 Operating System: TCAM Program Logic Manual, GY30-2029. References to other TCAM debugging aids are found in Appendix H.

Invoking an ABEND/SNAP Dump (MVT)

ABEND/SNAP dumps issued by systems with MVT are invoked in the same manner as those under systems with PCP and MFT. They result from an ABEND or SNAP macro instruction in a system or user program, accompanied by a properly defined data set. In the case of a system failure, the damage assessment routine (DAR) attempts to write a storage image dump to the SYS1.DUMP data set. A full explanation of storage image dumps may be found in the section headed "Storage Image Dump." The instructions that invoke an ABEND/SNAP dump in MVT environment are the same as those given in the preceding topic for systems with PCP and MFT. However, some additional considerations must be made in requesting main storage and direct access space.

MVT Considerations: In specifying a region size for a job step subject to abnormal termination, you must consider the space requirements for opening a SYSABEND or SYSUDUMP data set (if there is one), and loading the ABDUMP routine and required data management routines. This space requirement can run as high as 6000 bytes.

Direct access devices are used frequently for intermediate storage of dump data sets in systems with MVT. To use direct access space efficiently, the space for the dump data set should be varied, depending on whether or not abnormal termination is likely. A small quantity should be requested if normal termination is expected. To prevent termination of the dump due to a lack of direct access space, always specify an incremental (secondary) quantity when coding a SPACE parameter for a dump data set. You can obtain a reasonable estimate of the direct access space required for an ABEND/SNAP dump by adding, (1) the number of bytes in the nucleus, (2) the part of the system queue space required by the task (9150 bytes is a sufficient estimate), and (3) the amount of region space occupied by the task. Multiply the sum by 4, and request this amount of space in 1024-byte blocks.

This formula gives the space requirements for one task. Request additional space if partial dumps of subtasks and invoking tasks will be included.

Contents of an ABEND/SNAP Dump (MVT)

This explanation of the contents of ABEND/SNAP dumps issued by systems with MVT is interspersed with sample sections from an ABEND dump. Capital letters represent the headings found in all dumps, and lowercase letters, information that varies with each dump. The lowercase letter used indicates the mode of the information and the number of letters indicates its length:

- h represents 1/2 byte of hexadecimal information
- d represents 1 byte of decimal information
- c represents a 1-byte character

You may prefer to follow the explanation on your own ABEND or SNAP dump.

```
JOB IPCT41             STEP EXSTEP           TIME 002409   DATE 99366                                          PAGE 0001

COMPLETION CODE        SYSTEM = B37

PSW AT ENTRY TO ABEND  FF04000D 5000C408

TCB   02F028   RBP    0002EC78   PIE      00000000   DEB  0002ED34   TIO 000302F0   CMP    80B37000   TRN  00000000
               MSS    01031738   PK-FLG F0850409    FLG  00000000   LLS 000309B0   JLB    00000000   JPQ  000301E8
               FSA    01060768   TCB      00000000   TME  00000000   JST 0002F028   NTC    00000000   OTC  00030508
               LTC    00000000   IQE      00000000   ECB  000304B4   STA 00000000   D-PQE 00032668    SQS  0002EAA0
               NSTAE 00000000    TCT      00030268   USER 00000000   DAR 00000000   RESV   00000000   JSCB 0003146C

ACTIVE RBS

PRB   030DE8   RESV    00000000    APSW   00000000   WC-SZ-STAB 00040082    FL-CDE 00031290    PSW FFF50006 7003553E
               Q/TTR   00000000    WT-LNK 0002F028

PRB   030988   RESV    00000000    APSW   00000000   WC-SZ-STAB 00040002    FL-CDE 00030E80    PSW FFF50037 5207EC4A
               Q/TTR   00000000    WT-LNK 00030DE8

SVRB 02F0F0    TAB-LN 00980400     APSW   F5F5F0F2   WC-SZ-STAB 00120002    TQN     00000000   PSW FF04000D 5000C408
               Q/TTR   00003C0F    WT-LNK 00030988
               RG 0-7     00000FD9     000396F4     00000003     00000006     00000073     0003BC00     00036F88     0003CC33
               RG 8-15    00039100     000396F4     00060630     0003A158     0003ACE1     000395C0     5207F434     0007FC10
               EXTSA      F2F8F2F5     F3D6C340     0006DDE0     0002EFF4     0002EFC4     0006DF88     00000837     0003036C
                          80002648     00000001     0006DFE0     C3C45D04

SVRB 02F170    TAB-LN 008803C8     APSW   F2F0F1C3   WC-SZ-STAB 00120002    TQN     00000000   PSW 00040033 5000C0CE
               Q/TTR   00006109    WT-LNK 0002F0F0
               RG 0-7     80000000     80B37000     000396F4     4000C182     0006DDE0     0002EFD4     0002EFC4     0006DF88
               RG 8-15    00000837     0003036C     80002648     00000001     0006DFE0     00002648     0000DB68     00000001
               EXTSA      0000298F     0006DD88     2000FFFF     0006DBE0     FF030000     0002F1FC     0002F1F4     F2E8F2C9
                          C5C1F0F1     C9C5C128     C1C2C505     C407B386

SVRB 02EC78    TAB-LN 00C803C8     APSW   F1F0F5C1   WC-SZ-STAB 00120002    TQN     00000000   PSW FF040001 4007E8A4
               Q/TTR   00006201    WT-LNK 0002F170
               RG 0-7     00000000     0002F100     8000BDC8     0000DB68     0002F028     0002F170     00031290     00000000
               RG 8-15    0002F028     4000BD3A     0002F028     0006DD88     00030320     0002F1F4     4000D594     00000000
               EXTSA      00620300     00090040     0008000A     18002648     00000040     00090041     00028460     00000018
                          00120002     00000000     00000000     00000000

LOAD LIST

         NE 00030BF8    RSP-CDE 020301E8          NE 00030DF0    RSP-CDE 01032390          NE 00031078    RSP-CDE 01032290
         NE 00031080    RSP-CDE 01032260          NE 000310C8    RSP-CDE 01032390          NE 00031170    RSP-CDE 01032200
         NE 000311C0    RSP-CDE 010323C0          NE 00000000    RSP-CDE 01030BF0

CDE

      031290       ATR1 C8   NCDE 000000   ROC-RB 00030DE8   NM GO         USE 01   EPA 035508   ATR2 20   XL/MJ 031280
      030E80       ATR1 C8   NCDE 031290   ROC-RB 00030988   NM IEKAA00    USE 01   EPA 036240   ATR2 20   XL/MJ 02F398
      0301E8       ATR1 31   NCDE 030BF0   ROC-RB 00000000   NM IGC0A05A   USE 02   EPA 06C980   ATR2 28   XL/MJ 030AB0
      032390       ATR1 B8   NCDE 0323C0   ROC-RB 00000000   NM IGG019CD   USE 06   EPA 07FA00   ATR2 20   XL/MJ 032380
      032290       ATR1 B8   NCDE 0322C0   ROC-RB 00000000   NM IGG019BA   USE 05   EPA 07F4A0   ATR2 20   XL/MJ 032280
      032260       ATR1 B8   NCDE 032290   ROC-RB 00000000   NM IGG019BB   USE 05   EPA 07F880   ATR2 20   XL/MJ 032250
      032390       ATR1 B8   NCDE 0323C0   ROC-RB 00000000   NM IGG019CD   USE 06   EPA 07FA00   ATR2 20   XL/MJ 032380
      032200       ATR1 B8   NCDE 032230   ROC-RB 00000000   NM IGG019AJ   USE 03   EPA 07F3A0   ATR2 20   XL/MJ 0321F0
      0323C0       ATR1 B8   NCDE 0323F0   ROC-RB 00000000   NM IGG019AR   USE 04   EPA 07FC10   ATR2 20   XL/MJ 0323B0
      0303F0       ATR1 39   NCDE 030E80   ROC-RB 00000000   NM IEWSZOVR   USE 01   EPA 06C480   ATR2 20   XL/MJ 030B88

XL                                                  LN          ADR          LN          ADR          LN          ADR

      031280    SZ 00000010    NO 00000001        800002F8    00035508
      02F398    SZ 0000004C    NO 00000001        80016F38    000359C8    000359C8    00030B00    010A0400    010D0500
                                                  011C0300    011D0300    011E0200    01290400    012F0500    01300500
                                                  01320300    013A0100    01460600    01480400    014D0500
      030AB0    SZ 00000010    NO 00000001        80000680    0006C980
      032380    SZ 00000010    NO 00000001        80000210    0007FA00
      032280    SZ 00000010    NO 00000001        80000180    0007F4A0
      032250    SZ 00000010    NO 00000001        80000058    0007F880
      032380    SZ 00000010    NO 00000001        80000210    0007FA00
      0321F0    SZ 00000010    NO 00000001        80000100    0007F3A0
      0323B0    SZ 00000010    NO 00000001        80000090    0007FC10
      030B88    SZ 00000010    NO 00000001       .80000350    0006C480

DEB

02ED00                                                 00000D50 00000D50 00000D50 00000D50    *................................*
02ED20     00000D50 00000000 0000020A 00002BEC         0F000000 0002F028 0402FED4 98000000    *.....................0....M.....*
02ED40     8F000000 01000000 00000000 FF06D088         0402ED10 18002648 00000031 00010032    *................................*
02ED60     0001000B 00010001 C2C2C2C1 C3C40000         00000000 00000000 00000000 C3C40000    *........BBBACD..............CD..*
```

Figure 24A. Sample of Complete ABEND Dump (MVT)

DEB hhhhhhhh
is the TCBDEB field (bytes 8 through 11): starting address of the DEB queue. Under the heading DEB in the dump, the prefix section for the first DEB in the queue is presented in the first 8-digit entry on the first line. The 6-digit entry at the left of each line under DEB is the address of the second column on the line, whether or not the column is filled.

TIO hhhhhhhh
is the TCBTIO field (bytes 12 through 15): starting address of the TIOT.

CMP hhhhhhhh
is the TCBCMP field (bytes 16 through 19): task completion code or contents of register 1 when the dump was requested. System codes are given in the third through fifth digits and user codes in the sixth through eight digits.

TRN hhhhhhhh
is the TCBTRN field (bytes 20 through 23): starting address of the control core (table) for controlling testing of the task by TESTRAN.

MSS hhhhhhhh
is the TCBMSS field (bytes 24 through 27): starting address of SPQE most recently added to the SPQE queue.

PK-FLG hhhhhhhh
contains, in the first 2 digits, the TCBPKF field (byte 28): protection key.

contains, in the last 6 digits, the first 3 bytes of the TCBFLGS field (bytes 29 through 31): first 3 flag bytes.

FLG hhhhhhhh
contains, in the first 4 digits, the last 2 bytes of the TCBFLGS (bytes 32 and 33): last 2 flag bytes.

contains, in the next 2 digits, the

TCBLMP field (byte 34): limit priority (converted to an internal priority, 0 to 255).

contains, in the last 2 digits, the TCBDSP field (byte 35): dispatching priority (converted to an internal priority, 0 to 255).

LLS hhhhhhhh
is the TCBLLS field (bytes 36 through 39): starting address of the load list element most recently added to the load list.

JLB hhhhhhhh
is the TCBJLB field (bytes 40 through 43): starting address of the DCB for the JOBLIB data set.

JPQ hhhhhhhh
is the TCBJPQ field (bytes 41 through 47): when translated into binary bits:

- Bit 0 is the purge flag.
- Bits 1 through 7 are reserved for future use and are zeros.
- Bits 8 through 31 are the starting address of the queue of CDEs for the job pack area control queue, which is for programs acquired by the job step.

The TCBJPQ field is used only in the first TCB in the job step; it is zeros for all other TCBs.

RG 0-7 and RG 8-15
is the TCBGRS field (bytes 48 through 111): contents of general registers 0 through 7 and 8 through 15, as stored in the save area of the TCB when a task switch occurred. These 2 lines appear only in dumps of tasks other than the task in control when the dump was requested.

FSA hhhhhhhh
contains, in the first 2 digits, the TCBQEL field (byte 112): count of enqueue elements.

contains, in the last 6 digits, the TCBFSA field (bytes 113 through 115): starting address of the first problem program save area. This save area was set up by the control program when the job step was initiated.

TCB hhhhhhhh
is the TCBTCB field (bytes 116 through 119): starting address of the next lower priority TCB on the TCB queue or, if this is the lowest priority TCB, zeros.

TME hhhhhhhh
is the TCBTME field (bytes 120 through 123): starting address of the timer element created when an STIMER macro instruction is issued by the task.

JST hhhhhhhh
is the TCBJSTCB field (bytes 124 through 127): starting address of the TCB for the job step task. For tasks with a protection key of zero, this field contains the starting address of the TCB.

NTC hhhhhhhh
is the TCBNTC field (bytes 128 through 131): the starting address of the TCB for the previous subtask on this subtask queue. This field is zero in the job step task, and in the TCB for the first subtask created by a parent task.

OTC hhhhhhhh
is the TCBOTC field (bytes 132 through 135): starting address of TCB for the parent task. In the TCB for the job step task, this field contains the address of the initiator.

LTC hhhhhhhh
is the TCBLTC field (bytes 136 through 139): starting address of the TCB for the most recent subtask created by this task. This field is zero in the TCB for the last subtask of a job step, or in a TCB for a task that does not create subtasks.

IQE hhhhhhhh
is the TCBIQE field (bytes 140 through 143): starting address of the interruption queue element (IQE) for the ETXR exit routine. This routine is specified by the ETXR operand of the ATTACH macro instruction that created the TCB being dumped. The routine is to be entered when the task terminates.

ECB hhhhhhhh
is the TCBECB field (bytes 144 through 147): starting address of the ECB to be posted by the control program at task termination. This field is zero if the task was attached without an ECB operand.

STA hhhhhhhh
contains zeros, reserved for future use.

D-PQE hhhhhhhh
is the TCBPQE field (bytes 152 through 155): starting address minus 8 bytes of the dummy PQE. This field is passed by the ATTACH macro instruction to each TCB in a job step.

SQS hhhhhhhh
is the TCBAQE field (bytes 156 through 159): starting address of the allocation queue element (AQE).

NSTAE hhhhhhhh
contains, in the first 2 digits, STAE flags (byte 160).

contains, in the last 6 digits, the TCBNSTAE field (bytes 161 through 163): starting address of the current STAE control block for the task. This field is zero if STAE has not been issued.

TCT hhhhhhhh
is the TCBTCT field (bytes 164 through 167): address of the Timing Control Table (TCT).

USER hhhhhhhh
is the TCBUSER field (bytes 168 through 171): to be used as the user chooses.

DAR hhhhhhhh
contains, in the first two digits, Damage Assessment Routine (DAR) flags (byte 172);

MFT only, contains, in the last 6 digits, the secondary non-dispatchability bits (bytes 173 through 175).

RESV hhhhhhhh
reserved for future use.

JSCB hhhhhhhh
is the TCBJSCB field (bytes 180 through 183): the last three bytes contain the address of the Job Step Control Block.

```
ACTIVE RBS

cccc hhhhhh  cccccc hhhhhhhh   APSW    hhhhhhhh    WC-SZ-STAB hhhhhhhh    cccccc hhhhhhhh    PSW hhhhhhhh hhhhhhhh
             Q/TTR  hhhhhhhh   WT-LNK hhhhhhhh
             RG 0-7    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh
             RG 8-15   hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh
             EXTSA     hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh
                       hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh
```

ACTIVE RBS
identifies the next lines as the contents of the active RBs queued to the TCB, beginning with the oldest RB first.

cccc hhhhhh
indicates the RB type (cccc) and starting address (hhhhhh).

The RB types are:

PRB program request block
IRB interruption request block
SVRB supervisor request block

cccccc hhhhhhhh
indicates the RB's function (cccccc) and bytes 0 through 3 of the RB (hhhhhhhh):

- RESV hhhhhhhh indicates PRB or SVRB for resident routines. Bytes 0 through 3 are reserved for later use and contain zeros.
- TAB-LN hhhhhhhh indicates SVRB for transient routines. The first 4 digits contain the RBTABNO field (bytes 0 and 1): displacement from the beginning of the transient area control table (TACT) to the entry for the module represented by the RB. The last 4 digits contain the RBRTLNTH field (bytes 2 and 3): length of the SVC routine.
- FL-PSA hhhhhhhh indicates IRB. The first 2 digits contain the RBTMFLD field (byte 0): indicators for the timer routines. This byte contains zeros when the IRB does not represent a timer routine. The last 6 digits contain the RBPSAV field (bytes 1 through 3): starting address of the problem program register save area (PSA).

APSW hhhhhhhh
is the RBABOPSW field (bytes 4 through 7):

- In PRB, right half of the problem program's PSW when the interruption occurred.
- In IRB or SVRB for type II SVC routines, right half of routine's PSW during execution of ABEND or ABTERM, or zeros.
- In SVRB for type III or IV SVC routines, right half of routine's PSW during execution of ABEND or ABTERM, or the last four characters of the name of the requested routine. (The last two characters give the SVC number.)

WC-SZ-STAB hhhhhhhh
contains, in the first 2 digits, the RBWCSA field (byte 8): wait count in effect at time of abnormal termination of the program.

contains, in the second 2 digits, the RBSIZE field (byte 9): size of the RB in doublewords.

contains, in the last 4 digits, the RBSTAB field (bytes 10 and 11): status and attribute bits.

cccccc hhhhhhhh
indicates the RB's function (cccccc) and bytes 12 through 15 of the RB (hhhhhhhh):

- FL-CDE hhhhhhhh indicates SVRB for resident routines, or PRB. The first 2 digits contain the RBCDFLGS field (byte 12): control flags.

The last 6 digits contain the RBCDE field (bytes 13 through 15): starting address of the CDE for the module associated with this RB.

- EPA hhhhhhhh is the RBEP field of an IRB (bytes 12 through 15): entry-point address of asynchronously executed routine.

- TQN hhhhhhhh indicates SVRB for transient routines. Is the RBSVTQN field (bytes 12 through 15): address of the next RB in the transient control queue.

PSW hhhhhhhh hhhhhhhh
is the RBOPSW field (bytes 16 through 23): resume PSW.

Q/TTR hhhhhhhh
- In PRBs and SVRBs for resident routines, contains zeros in the first 2 digits. The last 6 digits contain the RBPGMQ field (bytes 25 through 27): queue field for serially reusable programs (also called the secondary queue).

- In IRBs, contains the RBUSE field in the first 2 digits (byte 24): count of requests for the same exit (ETXR). The RBIQE field in last 6 digits (bytes 25 through 27): starting address of the queue of interruption queue elements (IQE), or zeros in the first 4 digits and the RBIQE field in the last 4 digits (bytes 26 and 27): starting address of the request queue elements.

- In SVRBs for transient routines the first 2 digits contain the RBTAWCSA field (byte 24): number of requests (used if transient routine is overlaid) and the last 6 digits, the RBSVTTR field (bytes 25 through 27): relative track address for the SVC routine.

WT-LNK hhhhhhhh
contains, in the first 2 digits, the RBWCF field (byte 28): wait count.

contains, in the last 6 digits, the RBLINK field (bytes 29 through 31): starting address of the previous RB on the active RB queue (primary queuing field) or, if this is the first or only RB, the starting address of the TCB.

RG 0-7 and RG 8-15
is the RBGRSAVE field (bytes 32 through 95): in SVRBs and IRBs, contents of registers 0 through 15.

EXTSA
- In IRBs, contains the RBNEXAV field in the first 8 digits (bytes 96 through 99): address of next available interruption queue element (IQE), and in the remaining digits, the interruption queue element work space (up to 1948 bytes).

- In SVRBs, contains the RBEXSAVE field (bytes 96 through 143): extended save area for SVC routine.

```
LOAD LIST
        NE hhhhhhhh   RSP-CDE hhhhhhhh        NE hhhhhhhh   RSP-CDE hhhhhhhh        NE hhhhhhhh   RSP-CDE hhhhhhhh
```

LOAD LIST
identifies the next lines as the contents of the load list elements (LLEs) queued to the TCB by its TCBLLS field. The contents of 3 load list elements are presented per line until all elements in the queue are shown.

NE hhhhhhhh
contains, in the first 2 digits, LLE byte 0: zeros.

contains, in the last 6 digits, LLE bytes 1 through 3: starting address of the next element in the load list.

RSP-CDE hhhhhhhh
contains, in the first 2 digits, LLE byte 4: the count of the number of requests made by LOAD macro instructions for the indicated load module. This count is decremented by DELETE macro instructions.

contains, in the last 6 digits, LLE bytes 5 through 7: starting address of the CDE for the load module.

```
CDE
      hhhhhhhh    ATR1 hh    NCDE hhhhhh    ROC-RB hhhhhhhh    NM cccccccc    USE hh    EPA hhhhhh    ATR2 hh    XL/MJ hhhhhh
```

CDE
identifies the next lines as the contents directory addressed by an LLE or RB. One entry is presented per line.

hhhhhhhh
is the starting address of the entry given on the line.

ATR1 hh
is the attribute flags.

NCDE hhhhhh
is the starting address of the next entry in the contents directory.

ROC-RB hhhhhhhh
contains, in the first 2 digits, zeros.

contains, in the last 6 digits, the starting address of the RB for the load module represented by this entry.

NM cccccccc
is the name of the entry point to the load module represented by this entry.

USE hh
is the count of the uses (through ATTACH, LINK, and XCTL macro instructions) of the load module, and of the number of LOAD macro instructions executed for the module.

EPA hhhhhh
is the entry point address associated with the name in the NM field.

ATR2 hh
is the attribute flags.

XL/MJ hhhhhh
is the starting address of the extent list (XL) for a major CDE, or the starting address of the major CDE for a minor CDE. (Minor CDEs are for aliases.)

```
XL                                           LN        ADR        LN        ADR        LN        ADR
      hhhhhh   SZ hhhhhhhh   NO hhhhhhhh    hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh
```

XL
indicates the next lines are entries in the extent list, which is queued to the major contents directory entry. Each extent list entry is given in one or more lines. Only the first line for an entry contains the left 3 columns; additional lines for an entry contain information only in the right 6 columns.

hhhhhh
is the starting address of the entry.

SZ hhhhhhhh
is the total length, in bytes, of the entry.

NO hhhhhhhh
is the number of scattered control sections in the load module described by this entry. If this number is 1, the load module was loaded as one block.

LN hhhhhhhh
gives the length, in bytes, of the control sections in the load module described by this entry. Bit 0 is set to 1 in the last, or only, LN field to signal the end of the list of lengths.

ADR hhhhhhhh
gives the starting addresses of the control sections. Each ADR field is paired with the LN field to its left.

```
DEB

     hhhhhh                                   hhhhhhhh   hhhhhhhh         hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh
     hhhhhh         hhhhhhhh    hhhhhhhh    hhhhhhhh   hhhhhhhh         hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh
     hhhhhh         hhhhhhhh    hhhhhhhh    hhhhhhhh   hhhhhhhh         hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh
     hhhhhh         hhhhhhhh    hhhhhhhh

TIOT    JOB  cccccccc   STEP  cccccccc   PROC  cccccccc
     DD             hhhhhhhh     cccccccc    hhhhhhhh     hhhhhhhh
```

DEB
: identifies the next lines as the contents of the DEBs and their prefix sections. The first 6 digits in each line give the address of the DEB contents shown on the line, beginning with the second column. The first six digits of the first line contains the prefix section for the first DEB on the queue.

TIOT
: identifies the next lines as the contents of the TIOT.

JOB cccccccc
: is the name of the job whose task is being dumped.

STEP cccccccc
: is the name of the step whose task is being dumped.

PROC cccccccc
: is the name for the job step that called the cataloged procedure. This field appears if the job step whose task is being dumped was part of a cataloged procedure.

DD
: identifies the line as the contents of the DD entry in the TIOT.

```
MSS             ************ SPQE ************      *************** DQE ***************     ******* FQE ********
                FLGS  NSPQE      SPID      DQE           BLK        FQE         LN      NDQE           NFQE          LN


     hhhhhh     hh    hhhhhh     ddd      hhhhhh         hhhhhh hhhhhh      hhhhhh  hhhhhh       hhhhhhhh     hhhhhhhh

D-PQE hhhhhh   FIRST  hhhhhhhh   LAST   hhhhhhhh

PQE   hhhhhh    FFB   hhhhhhhh   LFB    hhhhhhhh  NPQ   hhhhhhhh  PPQ  hhhhhhhh
                TCB   hhhhhhhh   RSI    hhhhhhhh  RAD   hhhhhhhh  FLG  hhhhhhhh

FBQE  hhhhhh    NFB   hhhhhhhh   PFB    hhhhhhhh  SZ    hhhhhhhh
 .               .                .                .
 .               .                .                .
 .               .                .                .
PQE   hhhhhh    FFB   hhhhhhhh   LFB    hhhhhhhh  NPQ   hhhhhhhh  PPQ  hhhhhhhh
                TCB   hhhhhhhh   RSI    hhhhhhhh  RAD   hhhhhhhh  FLG  hhhhhhhh

FBQE  hhhhhh    NFB   hhhhhhhh   PFB    hhhhhhhh  SZ    hhhhhhhh
```

MSS
: identifies the next lines as the contents of the main storage supervisor queue. This queue includes subpool queue elements (SPQE), descriptor queue elements (DQE), and free queue elements (FQE).

hhhhhh
: is the starting address of the first element shown on the line.

SPQE
: identifies the 4 columns beneath it as the contents of SPQEs.

FLGS hh
: is the SPQE flag byte.

NSPQE hhhhhh
: is the starting address of the next SPQE in the queue.

SPID ddd
is the subpool number.

DQE hhhhhh
for a subpool owned by the task being dumped: the starting address of the first DQE for the subpool.

for a subpool that is shared: the starting address of the SPQE for the task that owns the subpool.

DQE
identifies the 4 columns beneath it as the contents of DQEs.

BLK hhhhhh
is the starting address of the allocated 2K block of main storage or set of 2K blocks.

FQE hhhhhh
is the starting address of the first FQE within the allocated blocks.

LN hhhhhh
is the length, in bytes, of the allocated blocks.

NDQE hhhhhh
is the starting address of the next DQE.

FQE
identifies the 2 columns beneath it as the contents of FQEs.

NFQE hhhhhhhh
is the starting address of the next FQE.

LN hhhhhhhh
indicates the number of bytes in the free area.

D-PQE hhhhhh
is the TCBPQE field (bytes 152 through 155): starting address minus 8 bytes of the dummy PQE shown on the line.

FIRST hhhhhhhh
is the starting address of the first PQE.

LAST hhhhhhhh
is the starting address of the last PQE.

PQE hhhhhh
is the starting address of the PQE shown on the line.

FFB hhhhhhhh
is bytes 0 through 3 of the PQE: starting address of the first FBQE. If no FBQEs exist, this field is the starting address of this PQE

LFB hhhhhhhh
is bytes 4 through 7 of the PQE: starting address of the last FBQE. If no FBQEs exist, this field is the starting address of this PQE.

NPQ hhhhhhhh
is bytes 8 through 11 of the element: starting address of the next PQE or, if this is the last PQE, zeros.

PPQ hhhhhhhh
is bytes 12 through 15 of the element: starting address of the preceding PQE or, if this is the first PQE, zeros.

TCB hhhhhhhh
is bytes 16 through 19 of the element: starting address of the TCB for the job step to which the space belongs or, if the space was obtained from unassigned free space, zeros.

RSI hhhhhhhh
is bytes 20 through 23 of the element: size of the region described by this PQE (a multiple of 2048).

RAD hhhhhhhh
is bytes 24 through 27 of the element: starting address of the region described by this PQE.

FLG hhhhhhhh
is byte 28 of the element:

bit 0 when 0, indicates space described by this PQE is owned;

when 1, indicates space is borrowed.

bit 1 when 1, indicates region has been rolled out (meaningful only when bit 0 is 0).
bit 2 when 1, indicates region has been borrowed.
bit 3-7, reserved for future use.

Note: PQE information is contained in two lines on the dump. When the rollout/rollin feature or Main Storage Hierarchy Support is included in the system, PQE information (with associated FBQEs) appears once in the dump for each region segment of the job step. (Each PQE on the partition queue defines a region segment. A job step's region contains more than one segment only when the step has rolled out another step or steps, or Main Storage Hierarchy Support is present.)

FBQE hhhhhh
: is the starting address of the FBQE shown on the line.

NFB hhhhhhhh
: is bytes 0 through 3 of the element: starting address of the next FBQE. In the highest or only FBQE, this field contains the address of the PQE.

PFB hhhhhhhh
: is bytes 4 through 7 of the element: starting address of the previous FBQE. In the lowest or only FBQE, the field contains the address of the PQE.

SZ hhhhhhhh
: is bytes 8 through 11 of the element: size, in bytes, of the free area.

QCB TRACE				
MAJ hhhhhh	NMAJ hhhhhhhh	PMAJ hhhhhhhh	FMIN hhhhhhhh	NM cccccccc
MIN hhhhhh	FQEL hhhhhhhh	PMIN hhhhhhhh	NMIN hhhhhhhh	NM xx xxxxxxxx
	NQEL hhhhhhhh	PQEL hhhhhhhh	TCB hhhhhhhh	SVRB hhhhhhhh

QCB TRACE
: identifies the next lines as a trace of the queue control blocks (QCB) associated with the job step. Lines beginning with MAJ show major QCBs, lines beginning with MIN show minor QCBs, and lines beginning with NQEL show queue elements (QEL).

MAJ hhhhhh
: is the starting address of the major QCB whose contents are given on the line.

NMAJ hhhhhhhh
: is the starting address of the next major QCB for the job step.

PMAJ hhhhhhhh
: is the starting address of the previous major QCB for the job step.

FMIN hhhhhhhh
: is the starting address of the first minor QCB associated with the major QCB given on the line.

NM cccccccc
: is the name of the serially reusable resource represented by the major QCB.

MIN hhhhhh
: is the starting address of the minor QCB whose contents are given on the line.

FQEL hhhhhhhh
: is the starting address of the first queue element (QEL), which represents a request to gain access to a serially reusable resource or set of resources.

PMIN hhhhhhhh
: is the starting address of the previous minor QCB.

NMIN hhhhhhhh
: is the starting address of the next minor QCB.

NM xx xxxxxxxx
: indicates, in the first 2 digits, the scope of the name or address of the minor QCB being dumped. If the scope is hexadecimal FF, the name is known to the entire operating system. If the scope is hexadecimal 00 or 10 through FO, the name is known only to the job step; in this case, the scope is the protection key of the TCB enqueuing the minor QCB.

  Also contains, in the last 8 digits, the name or the starting address of the minor QCB.

NQEL hhhhhhhh
: indicates, by hexadecimal 10 in the first 2 digits, that the queue element on the line represents a request for step-must-complete; by 00, ordinary request; and by 20, a set-must-complete request.

  Also contains, in the last 6 digits, the starting address of the next queue element in the queue, or for the last queue element in the queue, zeros.

PQEL hhhhhhhh
: indicates, by hexadecimal 80 in the first 2 digits, that the queue element represents a shared request or, by hexadecimal 00, that the element represents an exclusive request. (If

the shared DASD option was selected, hexadecimal 40 in the first 2 digits indicates an exclusive RESERVE request and 00 indicates a shared RESERVE request.)

TCB hhhhhhhh
: is the starting address of the TCB under which the ENQ macro instruction was issued.

SVRB hhhhhhhh
: is the starting address of the SVRB under which the routine for the ENQ macro instruction is executed, or, after the requesting task receives control of the resource, the UCB address of a device being reserved through a RESERVE macro instruction (the latter value occurs only when the shared DASD option was selected).

```
SAVE AREA TRACE

cccccccc WAS ENTERED VIA LINK (CALL) ddddd AT EP ccccc...

SA    hhhhhh  WD1 hhhhhhhh   HSA hhhhhhhh   LSA hhhhhhhh  RET hhhhhhhh  EPA hhhhhhhh  R0  hhhhhhhh
              R1  hhhhhhhh   R2  hhhhhhhh   R3  hhhhhhhh  R4  hhhhhhhh  R5  hhhhhhhh  R6  hhhhhhhh
              R7  hhhhhhhh   R8  hhhhhhhh   R9  hhhhhhhh  R10 hhhhhhhh  R11 hhhhhhhh  R12 hhhhhhhh


INCORRECT BACK CHAIN

INTERRUPT AT hhhhhh

PROCEEDING BACK VIA REG 13
```

SAVE AREA TRACE
: identifies the next lines as a trace of the save areas for the program. Each save area is presented in 3 or 4 lines. The first line gives information about the linkage that last used the save area. This line will not appear when the RB for the linkage cannot be found. The second line gives the contents of words 0 through 5 of the save area. The third and fourth lines give the contents of words 6 through 18 of the save area; these words are the contents of registers 0 through 12. Save areas are presented in the following order:

1. The save area pointed to in the TCBFSA field of the TCB. This save area is the first one for the problem program; it was set up by the control program when the job step was initiated.

2. If the third word of the first save area was filled by the problem program, then the second save area shown is that of the next lower level module of the task. However, if the third word of the first area points to a location whose second word does not point back to the first area, the message INCORRECT BACK CHAIN appears, followed by possible contents of the second save area.

3. The third, fourth, etc. save areas are then shown, provided the third word in each higher save area was filled and the second word of each lower save area points back to the next higher save area. This process is continued until the end of the chain is reached (the third word in a save area contains zeros) or INCORRECT BACK CHAIN appears.

Following the forward trace, the message INTERRUPT AT hhhhhh appears, followed by the message PROCEEDING BACK VIA REG 13. Then, the save area in the lowest level module is presented, followed by the save area in the next higher level. The lowest save area is assumed to be the 76 bytes beginning with the byte addressed by register 13. These two save areas appear only if register 13 points to a full word boundary and does not contain zeros.

cccccccc WAS ENTERED
: is the name of the module that stored register contents in the save area. This name is obtained from the RB.

VIA LINK ddddd or VIA CALL ddddd
: indicates the macro instruction (LINK or CALL) used to give control to the next lower level module, and is the ID

operand, if it was specified, of the LINK or CALL macro instruction.

AT EP ccccc...
: is the entry point identifier, which appears only if it was specified in the SAVE macro instruction that filled the save area.

SA hhhhhh
: is the starting address of the save area.

WD1 hhhhhhhh
: is the first word of the save area (optional).

HSA hhhhhhhh
: is the second word of the save area: starting address of the save area in the next higher level module. In the first save area in a job step, this word contains zeros. In all other save areas, this word must be filled.

LSA hhhhhhhh
: is the third word of the save area (register 13): starting address of the save area in the next lower level (called) module. If the module containing this save area did not fill the word, it contains zeros.

RET hhhhhhhh
: is the fourth word of the save area (register 14): return address (optional); if the called module did not fill the word, it contains zeros.

EPA hhhhhhhh
: is the fifth word of the save area (register 15): entry point to the called module. Use of this word is optional; if the called module did not fill the word, it contains zeros.

RO hhhhhhhh R1 hhhhhhhh ... R12 hhhhhhhh
: are words 6 through 18 of the save area (registers 0 through 12): contents of registers 0 through 12 for the module containing the save area immediately after the linkage. Use of these words is optional; if the called module did not fill these words, they contain zeros.

INCORRECT BACK CHAIN
: indicates that the following lines may not be a save area because the second word in this area does not point back to the previous save area in the trace.

INTERRUPT AT hhhhhh
: is the address of the next instruction to be executed in the problem program. It is obtained from the resume PSW word of the last PRB or LPRB in the active RB queue.

PROCEEDING BACK VIA REG 13
: indicates that the next 2 save areas are (1) the save area in the lowest level module, followed by (2) the save area in the next higher level module. The lowest save area is the save area pointed to by register 13. These 2 save areas appear only if register 13 points to a fullword boundary and does not contain zero.

```
CPUx PSA

hhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   *cccccccccccccccccccccccccccccccc*
hhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   *cccccccccccccccccccccccccccccccc*

NUCLEUS

hhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   *cccccccccccccccccccccccccccccccc*
hhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhhhhhhhhhhh hhhhhhhh   *cccccccccccccccccccccccccccccccc*

NUCLEUS CONT.

hhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   *cccccccccccccccccccccccccccccccc*
hhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   *cccccccccccccccccccccccccccccccc*

REGS AT ENTRY TO ABEND (SNAP)

     FLTR 0-6     hhhhhhhhhhhhhhhh     hhhhhhhhhhhhhhhh          hhhhhhhhhhhhhhhh     hhhhhhhhhhhhhhhh

     REGS 0-7     hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh          hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh
     REGS 8-15    hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh          hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh

LOAD MODULE cccccccc

hhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   *cccccccccccccccccccccccccccccccc*
hhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   *cccccccccccccccccccccccccccccccc*
   LINES    hhhhhh-hhhhhh    SAME AS ABOVE
hhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   *cccccccccccccccccccccccccccccccc*
hhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   *cccccccccccccccccccccccccccccccc*
   LINE     hhhhhh     SAME AS ABOVE

CSECT dd OF cccccccc

hhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   *cccccccccccccccccccccccccccccccc*
hhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   *cccccccccccccccccccccccccccccccc*
```

The contents of main storage are given under 6 headings: CPUx PSA, NUCLEUS, NUCLEUS CONT., LOAD MODULE cccccccc, CSECT dd OF cccccccc, and in the trace table, SP ddd BLK hh. Under these headings, the lines have the following format:

- First entry: the address of the initial bytes of the main storage presented on the line.
- Next 8 entries: 8 full words (32 bytes) of main storage in hexadecimal.
- Last entry (surrounded by asterisks): the same 8 full words of main storage in EBCDIC. Only A through Z, 0 through 9, and blanks are printed; a period is printed for anything else.

The following lines may also appear:

LINES hhhhhh-hhhhhh SAME AS ABOVE
: are the starting addresses of the first and last lines for a group of lines that are identical to the line immediately preceding.

LINE hhhhhh SAME AS ABOVE
: is the starting address of a line that is identical to the line immediately preceding.

CPUx PSA (Model 65 Multiprocessing dumps only)
: identifies the next lines as the contents of the prefixed storage area (PSA) -- 0 through 4095 (FFF). If the system is operating in partitioned mode (1 CPU), x is the CPU identification. If the system is operating in a 2 CPU multisystem mode, both PSAs are printed, the first under the heading CPUA PSA and the second under CPUB PSA.

NUCLEUS
: identifies the next lines as the contents of the nucleus of the control program.

NUCLEUS CONT.
: identifies the next lines as the contents of the part of the nucleus that lies above the trace table.

REGS AT ENTRY TO ABEND or REGS AT ENTRY TO SNAP
: identifies the next 3 lines as the contents of the floating point and general registers when the abnormal termination routine received control in response to an ABEND macro instruction or when the SNAP routine received control in response to a SNAP

macro instruction. These are not the registers for the problem program when the error occurred.

FLTR 0-6
indicates the contents of floating point registers 0, 2, 4, and 6.

REGS 0-7
indicates the contents of general registers 0 through 7.

REGS 8-15
indicates the contents of general registers 8 through 15.

LOAD MODULE cccccccc
identifies the next lines as the contents of the main storage area occupied by the load module cccccccc addressed by an LLE or RB. All the modules for the job step are dumped under this type of heading. Partial dumps do not contain this information.

CSECT hhhh OF cccccccc
identifies the next lines as the contents of the main storage area occupied by the control section (CSECT) indicated by hhhh. This control section belongs to the scatter-loaded load module cccccccc.

```
TRACE TABLE

DSP  NEW PSW     hhhhhhhh hhhhhhhh   R15/RO hhhhhhhh hhhhhhhh   R1  hhhhhhhh   SW  hhhhhhhh   TCB hhhhhhhh   TME hhhhhhhh
I/O  OLD PSW     hhhhhhhh hhhhhhhh   R15/RO hhhhhhhh hhhhhhhh   R1  hhhhhhhh   RES hhhhhhhh   TCB hhhhhhhh   TME hhhhhhhh
SIO  CC/DEV/CAW  hhhhhhhh hhhhhhhh   CSW    hhhhhhhh hhhhhhhh   RES hhhhhhhh   RES hhhhhhhh   TCB hhhhhhhh   TME hhhhhhhh
SVC  OLD PSW     hhhhhhhh hhhhhhhh   R15/RO hhhhhhhh hhhhhhhh   R1  hhhhhhhh   RES hhhhhhhh   TCB hhhhhhhh   TME hhhhhhhh
PGM  OLD PSW     hhhhhhhh hhhhhhhh   R15/RO hhhhhhhh hhhhhhhh   R1  hhhhhhhh   RES hhhhhhhh   TCB hhhhhhhh   TME hhhhhhhh
EXT  OLD PSW     hhhhhhhh hhhhhhhh   R15/RO hhhhhhhh hhhhhhhh   R1  hhhhhhhh   RES hhhhhhhh   TCB hhhhhhhh   TME hhhhhhhh
```

TRACE TABLE (SNAP dumps only)
identifies the next lines as the contents of the trace table. Each trace table entry is presented on one line; the name at the beginning of each line identifies the type of entry on the line:

- DSP Dispatcher entry
- I/O Input/output interruption entry
- SIO Start input-output (SIO) entry
- SVC Supervisor call (SVC) interruption entry
- PGM Program interruption entry
- EXT External interruption entry

OLD PSW hhhhhhhh hhhhhhhh
is the PSW stored when the interruption represented by the entry occurred.

NEW PSW hhhhhhhh hhhhhhhh
is the new PSW stored in the entry.

CC/DEV/CAW hhhhhhhh hhhhhhhh
contains, in the first 2 digits: completion code.

contains, in the next 6 digits: device type.

contains, in the last 8 digits: address of the channel address word (CAW) stored in the entry.

R15/RO hhhhhhhh hhhhhhhh
contains, in the first 8 digits: contents of register 15 stored in the entry.

contains, in the last 8 digits: contents of register 0 stored in the entry.

CSW hhhhhhhh hhhhhhhh
is the channel status word (CSW) stored in the entry.

R1 hhhhhhhh
is the contents of register 1 stored in the entry.

RES hhhhhhhh
is reserved for future use; all digits are zeros.

SW hhhhhhhh
is reserved for future use; all digits are zeros.

TCB hhhhhhhh
is the starting address of the TCB associated with the entry.

TME hhhhhhhh
is a representation of the timer element associated with the entry.

```
TRT

X DSP  NEW PSW     hhhhhhhh hhhhhhhh  R15/R0 hhhhhhhh hhhhhhhh  R1  hhhhhhhh  NUA hhhhhhhh  NUB hhhhhhhh  TME hhhhhh
X I/O  OLD PSW     hhhhhhhh hhhhhhhh  CSW    hhhhhhhh hhhhhhhh  R1  hhhhhhhh  OLA hhhhhhhh  OLB hhhhhhhh  TME hhhhhh
X SIO  CC/DEV/CAW hhhhhhhh hhhhhhhh  CSW    hhhhhhhh hhhhhhhh  TCB hhhhhhhh  OLA hhhhhhhh  OLB hhhhhhhh  TME hhhhhh
X SVC  OLD PSW     hhhhhhhh hhhhhhhh  R15/R0 hhhhhhhh hhhhhhhh  R1  hhhhhhhh  OLA hhhhhhhh  OLB hhhhhhhh  TME hhhhhh
X PGM  OLD PSW     hhhhhhhh hhhhhhhh  R15/R0 hhhhhhhh hhhhhhhh  R1  hhhhhhhh  OLA hhhhhhhh  OLB hhhhhhhh  TME hhhhhh
X EXT  OLD PSW     hhhhhhhh hhhhhhhh  R15/R0 hhhhhhhh hhhhhhhh  R1  hhhhhhhh  MSK hhhhhhhh  TQE hhhhhhhh  TME hhhhhh
X SSM  OLD PSW     hhhhhhhh hhhhhhhh  R15/R0 hhhhhhhh hhhhhhhh  R1  hhhhhhhh  AFF yyhhhhhh  OLB hhhhhhhh  TME hhhhhh
```

TRT (MVT with Model 65 multiprocessing dumps only)
identifies the next lines as the contents of the trace table. Each trace table entry is presented on one line; the letter and name at the beginning of each line identify the CPU and the type of entry, respectively:

- **DSP** Dispatcher entry.
- **I/O** Input/output interruption entry.
- **SIO** Start input/output entry.
- **SVC** Supervisor call interruption entry.
- **PGM** Program interruption entry.
- **EXT** External interruption entry.
- **SSM** Set system mask entry.

OLD PSW hhhhhhhh hhhhhhhh
is the PSW stored when the interruption represented by the entry occurred.

NEW PSW hhhhhhhh hhhhhhhh
is the new PSW stored in the entry.

CC/DEV/CAW hhhhhhhh hhhhhhhh
contains, in the first 2 digits: completion code; in the next 6 digits: device type; in the last 8 digits: address of the channel address word stored in the entry.

R15/R0 hhhhhhhh hhhhhhhh
contains, in the first 8 digits: contents of register 15; in the last 8 digits: contents of register 0, both as stored in the entry.

CSW hhhhhhhh hhhhhhhh
is the channel status word stored in the entry.

R1 hhhhhhhh
is the contents of register 1 as stored in the entry.

TCB hhhhhhhh
is the starting address of the TCB associated with the entry.

NUA hhhhhhhh
is the starting address of the new TCB for CPU A, as stored in the entry.

OLA hhhhhhhh
is the starting address of the old TCB for CPU A, as stored in the entry.

MSK hhhhhhhh
is the STMASK of the other CPU as stored in the entry.

NUB hhhhhhhh
is the starting address of the new TCB for CPU B, as stored in the entry.

OLB hhhhhhhh
is the starting address of the old TCB for CPU B, as stored in the entry.

TQE hhhhhhhh
is the first word of the timer queue element stored in the entry, provided a timer interrupt occurred.

TME hhhhhhhh
is a representation of the timer element associated with the entry.

AFF yyhhhhhh
contains, in the first 2 digits: the ID of the locking CPU at the time of the interrupt; in the last 6 digits: starting address of the old TCB for CPU A, as stored in the entry.

```
SP ddd

hhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    *cccccccccccccccccccccccccccccccc*
hhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    *cccccccccccccccccccccccccccccccc*

END OF DUMP
```

SP ddd
: identifies the next lines as the contents of a block of main storage obtained through a GETMAIN macro instruction, and indicates the subpool number (ddd). The part of subpool 252 that is the supervisor work area is presented first, followed by the entire contents of any problem program subpools (0 through 127) in existence during the dumping.

END OF DUMP
: indicates that the dump or snapshot is completed. If this line does not appear, the ABDUMP routine was abnormally terminated before the dump was completed, possibly because enough space was not allocated for the dump data set.

Guide to Using an ABEND/SNAP Dump (MVT)

Cause of Abnormal Termination: Evaluate the user (USER=decimal code) or system (SYSTEM=hex code) completion code using Appendix B or the publication Messages and Codes.

Dumped Task: Check the ID field for an indication of which task is being dumped in relation to the task that was abnormally terminated:

- 001 indicates a partial dump of a subtask
- 002 indicates a partial dump of the invoking task

If the ID field is absent, the dump contains a full dump of the task that was abnormally terminated.

Active RB Queue: The first RB shown on the dump represents the oldest RB on the queue. The RB representing the load module that had control when the dump was taken is third from the bottom. The last RB represents the ABDUMP routine and the second from last, the ABEND routine. The load module name and entry point (for a PRB) are given in a contents directory entry, the address of which is shown in the last 3 bytes of the FL/CDE field.

Program Check PSW: The program check old PSW is the fifth entry in the first line of each RB printout. It is identified by the subheading APSW. For debugging purposes, the APSW of the third RB from the bottom of the dump is most useful. It provides the length of the last instruction executed in the program (bits 32,33), and the address of the next instruction to be executed (bytes 5-8).

Load List: Does the resume PSW indicate an instruction address outside the limits of the load module that had control at the time of abnormal termination? If so, look at the LLEs on the load list. Each LLE contains the CDE address in the dump field labeled RSP-CDE.

CDEs: The entries in the contents directory for the region are listed under the dump heading CDE. The printouts for each CDE include the load module and its entry point. If you have a complete dump, each load module represented in a CDE is printed in its entirety following the NUCLEUS section of the dump.

Trace Table (SNAP dumps only): Entries on an MVT SNAP dump, if valid, represent occurrences of SIO, external, SVC, program, I/O, and dispatcher interruptions. SIO entries can be used to locate the CCW (through the CAW), which reflects the operation initiated by an SIO instruction. If the SIO operation was not successful, the CSW STATUS portion of the entry will show you why it failed. EXT and PGM entries are useful for locating the instruction where the interruption occurred (bytes 5-8 of the PSW).

SVC trace table entries provide the SVC old PSW and the contents of registers 0, 1, and 15. The PSW offers you the hexadecimal SVC number (bits 20-31), the CPU mode (bit 15), and the address of the SVC instruction (bytes 5-8). The contents of registers 0 and 1 are especially useful in that many system macro instructions pass key information in these registers. (See Appendix A.)

I/O entries reflect the I/O old PSW and the CSW that was stored when the interruption occurred. From the PSW, you can learn the address of the device that caused the interruption (bytes 2 and 3), the CPU state at the time of interruption (bit 15), and the instruction address where the interruption occurred (bytes 5-8). The CSW provides you with the unit status (byte 4), the channel status (byte 5), and the address of the previous CCW plus 8 (bytes 0-3).

You can use the DSP entry to delimit the entries in the trace table. To find all entries for the terminated task, scan word 7 of each trace table entry for the TCB address in a DSP entry. The lines between this and the next DSP entry represent interruptions that occurred in the task.

Region Contents: Free areas for the region occupied by the dumped task are identified under headings PQE and FBQE. The field labeled SZ gives the number of bytes in the free area represented by the FBQE.

Subpool Contents: Free and requested areas of the subpools used by the dumped task are described under the dump heading MSS. Subpool numbers are given under the SPID column in the list of SPQEs. If a GETMAIN macro instruction was issued without a subpool specification, space is assigned from subpool 0. Thus, two SPQEs may exist for subpool 0. The sizes of the requested areas and free areas are given under the LN column in the lists of DQEs and FQEs, respectively.

Load Module Contents: The contents of each load module used by the job step are given under the heading XL. Each entry includes the sizes (LN) and starting addresses (ADR) of the control sections in the load module.

Indicative Dump

An indicative dump is issued when a task is abnormally terminated by an ABEND macro instruction, and a dump is requested, but a dump data set is not defined, due either to omission or incorrect specification of a SYSABEND or SYSUDUMP DD statement. In systems with PCP or MFT, an indicative dump is issued automatically on the system output (SYSOUT) device. Indicative dumps issued by these two systems are identical in format. Systems with MVT do not issue indicative dumps.

Contents of an Indicative Dump

This explanation of indicative dumps utilizes capital letters for the headings found in all dumps, and lowercase letters for information that varies with each dump. The lowercase letter used indicates the mode of the information, and the number of letters indicates its length:

- h represents 1/2 byte of hexadecimal information
- d represents 1 byte of decimal information
- c represents a 1-byte character

Figure 25 shows the contents of an indicative dump. You may prefer to follow the explanation on your own indicative dump.

CONTROL BYTE=hh
describes the contents of the indicative dump.

First digit:

Bit	Setting	Meaning
0	0	Instruction image not present
	1	Instruction image present
1	0	Floating-point registers not present
	1	Floating-point registers present
2	0	One general register set present
	1	Two general register sets present
3	0	All active RBs present
	1	All active RBs not present

Last digit:

Digit in Hexadecimal	Meaning
0	All loaded RBs present
8	All loaded RBs not present

TCB FLAGS=hh
is the first byte of TCBFLGS field (byte 29 in the TCB for the program being dumped): task end flag byte:

Bit	Setting	Meaning
0	1	Abnormal termination in process
1	1	Normal termination in process
2	1	Abnormal termination was initiated by the resident ABTERM routine

```
CONTROL BYTE=hh  TCB FLAGS=hh  NO. ACTIVE RB=dd  NC. LOAD RB=dd
COMPLETION CODE - SYSTEM=hhh USER=dddd
cccccc...
REGISTER SET 1
hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh
hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh
REGISTER SET 2
hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh
hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh  hhhhhhhh
INSTRUCTION IMAGE=hhhhhhhhhhhhhhhhhhhhhhhh
hhhhhhhhhhhhhhhh    hhhhhhhhhhhhhhhh    hhhhhhhhhhhhhhhh    hhhhhhhhhhhhhhhh
PROGRAM ID=cccccccc  RB TYPE=hh  ENTRY POINT=hhhhhh
RESUME PSW  SM=hh  K=h  AMWP=h  IC=hhhh  IL.CC=h  PM=h  IA=hhhhhh
PROGRAM ID=cccccccc  RB TYPE=hh  ENTRY POINT=hhhhhh
```

Figure 25. Contents of an Indicative Dump

3	1	ABTERM routine entered because of program interruption
4	1	Reserved for future use
5	1	Data set closing initiated by the ABTERM routine
6	1	The ABTERM routine overlaid some or all of the problem program
7	1	The system prohibited queuing of asynchronous exit routines for this task

NO. ACTIVE RB=dd
is the number of active RBs presented in the dump.

NO. LOAD RB=dd
is the number of RBs in the load list presented in the dump.

COMPLETION CODE SYSTEM=hhh USER=dddd
is the completion code supplied by the control program (SYSTEM=hhh) or the problem program (USER=dddd). Both SYSTEM=hhh and USER=dddd are printed; however, one of them is always zero.

cccccc...
explains the completion code or, if a program interruption occurred:

PROGRAM INTERRUPTION ccccc... AT LOCATION hhhhhh
where ccccc is the program interruption cause: OPERATION, PRIVILEGED OPERATION, EXECUTE, PROTECTION, ADDRESSING, SPECIFICATION, DATE, FIXED-POINT OVERFLOW, FIXED-POINT DIVIDE, DECIMAL OVERFLOW, DECIMAL DIVIDE, EXPONENT OVERFLOW, DECIMAL DIVIDE, EXPONENT OVERFLOW, EXPONENT UNDERFLOW, SIGNIFICANCE, or FLOATING-POINT DIVIDE; and hhhhhh is the address of the instruction being executed when the interruption occurred.

REGISTER SET 1
indicates that the next 2 lines give the contents of general registers 0 through 7 and 8 through 15 for a program being executed under control of an RB when it:

- Passed control to a type I SVC routine through an SVC instruction and the routine terminated abnormally.
- Lost control to the input/output interruption handler, which subsequently terminated abnormally.
- Was abnormally terminated by the control program because of a program interruption.
- Issued an ABEND macro instruction to request an abnormal termination.

If REGISTER SET 2 also appears in the dump, the lines under REGISTER SET 1 give the general register contents for a type II, III, or IV SVC routine operating under an SVRB.

REGISTER SET 2
indicates that the next 2 lines give the contents of general registers 0 through 7 and 8 through 15 for a program being executed under control of an RB other than an SVRB when the program last passed control to a type II, III, or IV SVC routine.

INSTRUCTION IMAGE=hhhhhhhhhhhhhhhhhhhhhhhh
is 12 bytes of main storage, with the instruction that caused a program interruption in the right part of the printout. This field appears only if a program interruption occurred and is also valid when the instruction length in the resume PSW is 0.

hhhhhhhhhhhhhhhh hhhhhhhhhhhhhhhh
hhhhhhhhhhhhhhhh hhhhhhhhhhhhhhhh
are the contents of floating-point registers 0, 2, 4, and 6 when the abnormal termination occurred. This field appears only if the floating point option is present. The first 2 digits of each register are the characteristic of the floating point number. The last 14 digits are the mantissa.

PROGRAM ID=cccccccc
is the XRBNM field (bytes 0 through 7): in PRB, LRBs, and LPRBs, the program name; in IRBs, the first character contains flags for the timer or, if the timer is not being used, contains no meaningful information; in SVRBs for a type II SVC routine, contains no meaningful information; in SVRBs for a type III or IV SVC routine, the first 4 bytes contain the relative track address (TTR) of the load module in the SVC library and the last 4 bytes contain the SVC number in signed, unpacked decimal; in SIRBs, the name of the error routine currently occupying the 400-byte input/output supervisor transient area.

RB TYPE=hh
    indicates the type of active RB

hh	Type of RB
00	PRB that does not contain entry points identified by IDENTIFY macro instructions
10	PRB that contains one or more entry points identified by IDENTIFY macro instructions
20	LPRB that does not contain entry points identified by IDENTIFY macro instructions
30	LPRB that contains one or more entry points identified by IDENTIFY macro instructions
40	IRB
80	SIRB
C0	SVRB for a type II SVC routine
D0	SVRB for a type III or IV SVC routine
E0	LPRB for an entry point identified by an IDENTIFY macro instruction
F0	LRB

ENTRY POINT=hhhhhh
    is the XRBEP field (bytes 13 through 15): address of entry point in the program.

RESUME PSW
    XRBPSW field (bytes 16 through 23): is the contents of the resume PSW.

SM=hh
    is bits 0 through 7 of PSW: system mask.

K=h
    is bits 8 through 11 of PSW: protection key.

AMWP=h
    is bits 12 through 15 of PSW: indicators.

IC=hhhh
    is bits 16 through 31 of PSW: interruption code.

IL.CC=h
    is bits 32 through 35 of PSW: instruction length code (bits 32 and 33) and condition code (bits 34 and 35).

PM=h
    is bits 36 through 39 of PSW: program mask.

IA=hhhhhh
    is bits 40 through 63 of PSW: instruction address.

PROGRAM ID=cccccccc
    is the XRBNM field (bytes 0 through 7): program name.

RB TYPE=hh
    indicates the type of RB:

hh	Type of RB
20	LPRB that does not contain entry points identified by IDENTIFY macro instructions.
30	LPRB that contains one or more entry points identified by IDENTIFY macro instructions.
E0	LPRB for an entry point identified by an IDENTIFY macro instruction.
F0	LRB.

ENTRY POINT=hhhhhh
    is the XRBEP field (bytes 13 through 15): address of entry point in the program.

Guide to Using an Indicative Dump

Completion Code: Evaluate the user (USER=decimal code) or system (SYSTEM=hex code) completion code using either Appendix B of this publication or the publication Messages and Codes. The line under the completion code gives a capsule explanation of the code or the type of program interruption that occurred.

Instruction Address: If a program interruption occurred, get the address of the erroneous instruction in the last 3 bytes of the field labeled INSTRUCTION IMAGE.

Active RB Queue: RBs are shown in the first group of two-line printouts labeled PROGRAM ID and RESUME PSW, with the most recent RB shown first. There are two lines for as many RBs indicated by NO. ACTIVE RB=dd.

Register Contents: General register contents at the time a program last had control are given under the heading REGISTER SET 2 or, if this heading is not present, under REGISTER SET 1. Register contents, particularly those of register 14, may aid you in locating the last instruction executed in your program.

Storage Image Dump

A storage image dump writes to an external data set all of main storage from location 00 through the end of printable storage. The damage assesment routine (DAR) will produce a storage image dump when a system task fails if the SYS1.DUMP data set is properly defined and available to accept the dump. In MFT, MVT, and M65MP the print dump service aid program (IMDPRDMP) is used to print from the SYS1.DUMP data set; in PCP the print dump program (IEAPRINT) is used.

Note: IEAPRINT or IMDPRDMP is placed in SYS1.LINKLIB at system generation, depending on the system option. For PCP, IEAPRINT is placed in SYS1.LINKLIB. For MFT, MVT, and M65MP, the IMDPRDMP program is placed in SYS1.LINKLIB. IEAPRINT may be invoked with the JCL statements shown in Figure 27 and IMDPRDMP with those shown in Figure 26.

DAMAGE ASSESSMENT ROUTINE (DAR)

The damage assessment routine (DAR) is designed to provide increased system availability in the event of a system failure, and to provide more meaningful diagnostic information by means of a storage image dump taken at the time of the system failure. This storage image dump is written to the SYS1.DUMP data set, which you may print by means of the IMDPRDMP service aid program or, in the case of PCP, the IEAPRINT print dump program.

If a system routine fails, DAR attempts to reinitialize the failing task, thereby permitting the system to continue operation without interruption. DAR permits the system to continue processing in a degraded condition if it encounters a system failure that does not permit total reinstatement of the affected task or region. The operator will be informed, via a WTO, that the system is in an unpredictable state; he then must decide whether or not already-scheduled jobs should be allowed to attempt completion.

SYSTEM FAILURE

If a system failure occurs, the damage assessment routine immediately attempts to write a storage image dump to the SYS1.DUMP data set. A system failure may be caused by a failure in any of the following system tasks:

PCP and MFT:

- Communications Task
- Master Scheduler Task
- Log Task (MFT only)

MVT:

- System Error Task
- Rollout/Rollin Task
- Communications Task
- Master Scheduler Task
- Transient Area Fetch Task

A system failure is also caused by an ABEND recursion in other than OPEN, CLOSE, ABDUMP, or STAE; by a failure of a task in 'must complete' status; or, in MFT only, by a failure in the scheduler if no SYSABEND or SYSUDUMP DD card is provided.

THE SYS1.DUMP DATA SET

One of the primary functions of the damage assessment routine is to provide a storage image dump at the time of a system failure. Secondary storage space must be available to receive this dump. The SYS1.DUMP data set provides this space.

The SYS1.DUMP data set may reside on tape or on a direct access device.

Tape

If you wish to have the SYS1.DUMP data set reside on tape, you may specify the tape drive during IPL. If the drive has not been made ready prior to IPL, a MOUNT message is issued to the console, specifying the selected device. The device should be mounted with an unlabeled tape.

After writing a storage image dump, the damage assessment routine writes a tape mark and will position the tape to the next file. The tape drive will remain in a ready state to receive another storage image dump.

Direct Access

If you wish to have the SYS1.DUMP data set placed on a direct access device, you may preallocate the data set at system generation or prior to any IPL of the system. The following restrictions apply:

- The data set name must be SYS1.DUMP.
- The data set must be cataloged on the IPL volume.
- The data set may be preallocated on any volume that will be online during system operation.

* The data set must be sequential.

* Sufficient space must be allocated to receive a storage image dump for all of main storage.

When a direct access device is used for the SYS1.DUMP data set, the data set can hold only one storage image dump. If additional failures occur, and if the SYS1.DUMP data set is occupied, DAR does not attempt to write another storage image dump.

You may execute the print dump service aid program (IMDPRDMP) or, in the case of PCP, the print dump program (IEAPRINT), to produce hard copy of the dump.

THE PRINT DUMP SERVICE AID (IMDPRDMP) FOR MFT, MVT AND M65MP

For MFT, MVT, and M65MP you must use the print dump service aid program to print out the storage image dump contained on the SYS1.DUMP data set. The print dump service aid is placed in SYS1.LINKLIB at system generation; it is invoked in the same manner as any other problem program.

Figure 26 shows how to use the PRMP catalogued procedure to process a SYS1.DUMP data set that contains a storage image dump from a 512K machine or less.

The following explanation is for the job control language statements in Figure 26; for information about the job control language statements in the cataloged procedure and the IMDPRDMP control statements, see the Service Aids SRL, GC28-6719.

JOB STATEMENT: This statement marks the beginning of the job.

EXEC STATEMENT: This execute statement invokes the cataloged procedure called PRDMP. The PRDMP procedure causes the IMDPRDMP program to be executed. When the cataloged procedure is invoked, the user's job control language statements are merged with the job control language statements in the procedure. PARM.DMP=T causes the IMDPRDMP program to request the title of the dump from the console operator before formatting and printing the dump data set; this permits the operator to assign a distinct name to each dump.

DMP.SYSIN DD STATEMENT: This data definition statement defines the data set where the IMDPRDMP control statements are located. In this case, the control statements follow this DD statement in the input job stream. If this statement is omitted, IMDPRDMP requests control statement information from the console operator.

GO FUNCITON CONTROL STATEMENT: The GO statement causes the IMDPRDMP program to format and print the SYS1.DUMP data set described in the cataloged procedure by the TAPE data definition statement. The SYS1.DUMP data set is cataloged. The absence of the ONGO statement in this procedure causes IMDPRDMP to format and print this data set using the default GO format parameters: QCBTRACE (Q), LPAMAP (L), FORMAT (F), and PRINT ALL (PA).

END FUNCTION CONTROL STATEMENT: The END statement terminates IMDPRDMP processing. Had this statement been omitted, IMDPRDMP would issue a write to operator with reply (WTOR) asking the console operator to enter additional control statements; by using this IMDPRDMP feature, an operator can format and print several dumps during the same execution of IMDPRDMP.

For additional examples of the various uses and output of IMDPRDMP, see the Service Aids SRL, GC28-6719.

THE PRINT DUMP PROGRAM (IEAPRINT)

For PCP dumps, you must use the IEAPRINT print dump program to print the storage image dump contained on the SYS1.DUMP data set. The IEAPRINT print dump program is placed in SYS1.LINKLIB at system generation time only if PCP is the chosen option; it may be invoked in the same manner as any other problem program.

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

Figure 26. Sample JCL Statement Required for IMDPRDMP

You must supply the job control statements for the print dump program; the following statements are required:

JOB — This is a standard statement.

EXEC — This statement specifies the program name (PGM=IEAPRINT) or, if the job control statements reside on the procedure library, the procedure name.

SYSPRINT DD — This statement defines an output data set. The data set may be written cnto a system output device, a magnetic tape volume, or a direct access device.

SYSUT1 DD — This statement defines the input data set. The DSNAME SYS1.DUMP must be used.

(See Figure 27 for the JCL statements required to execute the IEAPRINT print dump program.)

```
//URJOB    JOB    ,URNAME,MSGLEVEL=(1,1)
//         EXEC   PGM=IEAPRINT
//SYSPRINT DD     SYSOUT=A,SPACE=(CYL,(20,5))
//SYSUT1      DD  DSNAME=SYS1.DUMP,UNIT=2400,LABEL=(,NL),DISP=(OLD,KEEP),  X
//                VOLUME=SER=1234
```

Figure 27. Sample JCL Statements Required for IEAPRINT

Input to the Print Dump Program

Input to the IEAPRINT program is the sequential data set SYS1.DUMP, which may reside on either a direct access device or on magnetic tape. The first byte of the first record on the SYS1.DUMP data set will be the contents of storage location 00, and the data set will contain the full storage image up to the last writable byte. The input devices supported are:

IBM 2301 Drum Storage Unit

IBM 2302 Disk Storage Drive

IBM 2303 Drum Storage Unit

IBM 2311 Disk Storage Drive

IBM 2314 Storage Facility

IBM 2400 Magnetic Tape Drive

Output From the Print Dump Program

The output from the print dump program is a formatted storage image dump of the printable contents of main storage, beginning at location 00. The dump may be written onto a system output device, a magnetic tape volume, or a direct access device. You must define the device, upon which the dump is to be written, on the SYSPRINT DD card of the JCL statements that invoke the print dump program. (See Figure 27.)

CONTENTS OF A STORAGE IMAGE DUMP

The storage image dump is formatted into two distinct sections: low storage and register contents are displayed on the first page, and a printout of the contents of main storage begins on the second page. The main storage contents are unedited and are displayed beginning from location 00 through the end of printable storage. (See Figure 28.)

Low Storage and Registers

The initial section of a storage image dump (the first page) consists of information of immediate use to the programmer who must determine the cause of the failure.

The first printed line displays the control program option of the operating system, i.e. PCP, MFT, MVT, or M65MP; the timer contents at the time of the failure; and the date of the failure.

The remainder of the first page consists of a printout of register contents and hardware control words as they appeared at the time of the failure. The contents of floating point registers 0, 2, 4, and 6 are displayed; if the floating point feature is not present in the system, these register printouts contain zeros. The two lines beginning with REG 0-7 and REG 8-15 show the contents of general registers 0 through 7 and 8 through 15, respectively.

Storage below location 128(80 hex) is permanently assigned and can be used to determine the status of a program. The line beginning 40-CSW (following the register printout) gives, in unedited form, the CSW and CAW. The next five lines contain the new and old PSWs for the five types of interruptions.

The last line in this portion of the dump, beginning 4C-UNUSED-, gives the contents of locations 76(4C hex) through 87 (57 hex), which include unused bytes and the timer. This line contains pointers useful in locating key debugging information, such as the CVT and the trace table. The use of these locations will be explained under the sections headed "Guide to Using...".

Main Storage

The main section of the dump is printed starting with location zero and continuing to the end of printable storage. Each line contains, from left to right:

- The hexadecimal storage address of the first byte on the line.
- Eight words of storage in hexadecimal.
- The same eight words in EBCDIC, enclosed in asterisks (*).

If one or more consecutive lines contain the same word throughout the line, the first line will be printed, followed by the message,

hhhhhh TO THE NEXT LINE ADDRESS - SAME AS ABOVE

where:

hhhhhh
: is the address of the first omitted line.

```
CORE IMAGE DUMP OF  MVT  SYSTEM                                   TIMER= 084082                      DATE =  00099366

FLOATING POINT REGISTERS     0                  2                  4                  6
                             C9D5C9E3C9C1E3D6   D9407DC1D3D37D40   E6C1C9E3C9D5C740   0000000000000000

REG 0-7        C0020C00  8CC00C0B  00021B98  000000F0     00000010  400586EC  00020C00  00021748
REG 8-15       00021B8C  00FFFFF8  00000068  400586EE     6007EAB2  000587AD  00008904  0000000B

40-CSW  000CD5C00C000000                                  48-CAW  00004408

EXTERNAL INTERRUPT PSWS      NEW=0004000000007628   OLD=01040080800388F6
SUPERVISOR CALL PSWS         NEW=000400CC000C80B0   OLD=FF040C015000D8C4
PROGRAM CHECK PSWS           NEW=000400000000785C   OLD=000C0CCCCC0CC000
MACHINE CHECK PSWS           NEW=0C0000000C0184C0   OLD=000CFF0000000000
INPUT/OUTPUT PSWS            NEW=0CC4C0CC000077E0   OLD=FF06C2918C0C0000

4C-UNUSED-0000DE48       5C-TIMER-C8408262       54-UNUSED-0000EE70
```

```
000000   0CCCCCCC 0CC00000 00000000 0000000C   0000DE48 C0000000 C1040C8C 80038BF6   *...............................6*
00C02C   FF0400C1 5000D8C4 C000000C 00000000   0000FF0C 000000C0 FF060291 80000000   *....&.QD........................*
00C04C   0CCCD5CC CCCCCCCC 0CC044C8 0CC0DE48   0840B262 0000EE70 00040000 00007628   *..N.............................*
000060   0C0400CC 0000B0BC 0CC40C00 0000785C   C0000000 000184CC CCC40000 000077E0   *..............*.................*
000080   A7ABCCFF F2F39FFF 0F003FFF F23F9FFF   00000000 C00000CC FFFFFFFF 0CC880CC   *....23......2...................*
000CA0   0000400C 4C0C00CC 30008CCB 005FBF81   001AC200 FFC40000 02010000 00000000   *.. ...............B..D..........*
0C0CCC   00000429 0CC008CC CC000429 03831600   D207830C C3DAE8C1 0070CF8A D2070030   *..............K.....Y.....K....*
000CE0   FF6FFFFF CC000000 D02CF9FF 0003830A   00000000 00000000 00000000 00021000   *..........9.....................*
0CC10C   0CCC0CCC 001001CA CCCC0CCC CC80E800   CC0C0000 46CC82B9 CCCCC000 5010CF03   *..............Y.............&...*
00C12C   0CCC0CCC 0C000000 00000000 02010000   00000000 00000CC0 00000000 000000C0   *................................*
0CC14C   CCCCCCCC C00000CC CCCCCCCC 0C000000   00000000 000000C0 00000000 00000000   *................................*
00016C   000000C0 0CC000CC 0CC00000 8200017C   0CCCC000 00038280 CC000000 00000000   *................................*
0CC18C   0CCC0CCC 0C0000C0 CC000000 00000000   00000000 00000000 00000000 00000000   *................................*
CCC1AC TC THE NEXT LINE ADDRESS  -  SAME AS ABOVE
0C0200   FF060291 800000CC 0CC00001 0001D344   CCCCDE48 0000DE48 000D380 4000D5CA    *..............L...........L. .N.*
0CC220   00C2CCD0 8C00D68C 90C0D832 000215D8   0001D310 F30024F8 0C020CD0 000006CA   *......0...Q....Q..L.3..8........*
000240   8C00D644 000C24F4 CCCCCC00 0C000000   CC0C0C0C 00000000 00000000 00000000   *..0....4........................*
0CC260   0CCC000C C0000000 00000CCC 0C000000   000C0000 0C0C0CCC C0000CCC 00000000   *................................*
0CC28C   0CCC0CCC C000C000 CCC00000 00000000   000CA820 00000000 0000DE38 00019CC8   *...............................H*
0C02AC   00CC010C 8200C300 34CC350C 36CC0001   0C07F8CC CCCCCCCC C200C0C0 00000000   *.................8.....B........*
0C02C0   000C8340 00008340 00000000 0000001F   112C1BE4 24F802DA 02DA02DA 02DA7FFF   *... ... ............U.8........".*
0CC2EC   C000000C C00184C0 CCC20CC0 0C000A21   CCC4C0CC 00007628 0C040000 00000472   *................................*
000300   00000000 00000000 000000CC 00000000   0000C000 00000000 00000000 00000000   *................................*
0CC32C TC THE NEXT LINE ADDRESS  -  SAME AS ABOVE
000340   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   *................................*
0CC360 TC THE NEXT LINE ADDRESS  -  SAME AS ABOVE
0CC4CC   CCCC0CCC CCC000CC CCC0CC00 0C0C000C   00000000 00000000 C000C0CC 0C0000C0   *................................*
000420 TC THE NEXT LINE ADDRESS  -  SAME AS ABOVE
CCC460   CCCC0C00 0CC0C0CC 00000000 00000000   0C0090EF C4DC5EEC C2B09120 001B4710   *................................*
0CC480   05109110 E2BC471C 050C9121 E2BC4770   06209120 04714710 04BA41F0 02F09107   *....S.......S...............0.0..*
0CC4A0   04714770 04BE58EC E2BC54E0 C508477C   C4BE91CF C01B477C C4BE41FC 0018D207   *........S.................0..K.*
0CC4C0   04D8FC0C 98EF04D0 820004D8 00000000   0C000000 0000C000 0C000000 00000000   *.Q0........Q....................*
0CC4EC   00000000 0CCC00CC CCCC0000 0C00000C   0C020CC0 4000D5CA 00020CC0 7000D622   *.......................N.......0.*
00C50C   00020F10 0000D5F2 CEFFFFFF 94EFE2BC   968002CC 918CE2CC 477CC53A 9120001B   *......N2......S........S.........*
0CC52C   478C056C 58F0029C 9300F001 47800542   C500F000 02B84770 C54241F0 02E847FC   *...-.0....0.....N.0........0.Y.0*
000540   04BED20C F000E288 C207EC0C 02E058E0   06140700 07000700 46EC0558 848005F0   *..K.0.S.K.......................0*
000560   94F702CC 58F00618 58E0061C 07000700   46EC057C 910802CC 47700594 58E002B0   *.7...0..........................*
0C0580   9121E2BC 477CC62C 46FC0568 41F005F0   47F0048E 58F00608 9102F001 478005A8   *..S......0...0.0.0...0....0.....*
0005A0   91FFF00C 478004BE C20704D8 0018D207   CC58C5EC 82000058 82CC05E8 947F02CC   *..0.....K..Q..K............Y."..*
0C05C0   D60004DB 001BD207 001804D8 94DF001B   D207005E C2F841F0 02F847F0 048E0000   *O.....K....Q....K....8.0.8.0....*
0CC5EC   C1C4CCCC CCCC05B8 CCC40CC0 0C0005BC   0C02CCC0 00000A22 0104C0C0 000006A0   *................................*
000600   0004000C 000006A4 00019948 00000000   0C8C0000 0003512F CCC01F40 000006C8   *...........................  ....*
CCC62C   9CEFC4E0 58E002B0 9120E28C 4710C888   91010471 4780063E D207C4C0 C04091C1   *..........S.............K....  ..*
000640   E2BC4710 074091C1 04714780 0654D207   00400400 98EF04E0 918002CC 47800492   *S.... .........K.. ..............*
0CC660   47F00588 910302AF 077E91A1 02BC078E   90EF04E8 58FC061C 5CFCC60C 841005FC   *.C.......=.........Y.0..&0.....0*
000680   912102BC 478CC6C4 58F0060C 58E002B0   91FFE032 47100686 96200471 820005F8   *.......D.0......................8*
0CC6AC   82C00600 94DFC471 D6CCC470 00184780   06B69602 C47146F0 0678D207 04D004E8   *........C..............0..K....Y*
0006C0   47FC058C 98EF04E8 07FE9048 04F050E0   05045860 02B04170 06F25840 62C44580   *.0.....Y.....0&....-.....2. .D..*
0CC6EC   C718968C 02BC45EC C66491C2 64714780   06884180 07305840 02C44570 07185840   *......................... .D..... *
000700   62C04570 07185840 02C04570 07189848   04FC58EC 050407FE 12440788 48504020   *....... ..........0............& .*
0CC720   1255077E 585040C0 91FF501C 077807F7   D70302C0 02C0D703 62C062C0 47F007CE   *.....& ...&....7P.....P......0..*
0CC74C   900F0810 910302AF 478C075C 58E002B0   94FEE2BC 980F0810 47F00646 58700864   *...........*......S.......0......*
000760   48807000 417070C2 5480087C 47800760   55800870 4780074C 5810086C 1A184320   *..............-............%....*
0CC78C   1CCC882C CC04542C C874442C 087C4780   07604320 10005420 08785920 02844770   *..............@...-.............*
0007A0   07AA58C0 C86845F0 D0064320 02B8542C   08748920 00045720 C8704420 088447F0   *.......0.........................0*
```

Figure 28. Sample of a Storage Image Dump

Guide to Using a Storage Image or a Stand-Alone Dump

The purpose of this section is to suggest debugging procedures that you may use with a storage dump or a stand-alone dump. This discussion applies to the output of the following programs:

- IMDSADMP- The low speed version that formats and dumps main storage.
- IEAPRINT- Formats and prints storage dumps for PCP.
- IMDPRDMP- Reads, formats, and prints storage dumps from MFT or Mvt systems and the high speed version of IMDSADMP.

All of these programs produce hexadecimal dumps of the contents of main storage from location zero to the highest machine address.

The IMDPRDMP program provides formatting capabilities which can be used to display the important system control blocks for easy examination. The IMDPRDMP program does most of the procedures described in this section automatically. The cases in which the IMDPRDMP program does not provide formatting are identified. A complete description of the services provided by the IMDPRDMP program is found in the publication, IBM System/360 Operating System: Service Aids, GC28-6719.

Since the formatting for the IMDPRDMP program depends on the contents of the dump, it is not always possible to provide complete formatting. For example, if the CVT of the system to be dump has been overlaid, the IMDPRDMP program can provide only a hexadecimal dump of main storage.

DETERMINING THE CAUSE OF THE DUMP

Main storage dumps are invokes by system routines and these routines can be identified by module names appearing in the most recent request block (RB) for the failing task. (With PCP, there is only one task at any given time and that task will invoke the dump. This can be verified in the system.) With MVT and MFT, the main storage dump is invoked by SVC 51. This SVC PSW appears as the resume PSW in the second most recent RB of some task in the system. The module name in the current RB for that task must be 201C.

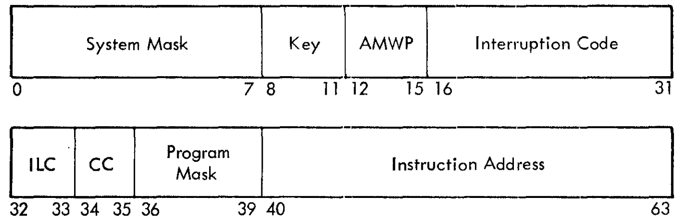
Main storage locations from zero to 128 (hexadecimal 80) are permanently assigned and contain hardware control words. Table 1 shows these fields, their location, their length, and their purpose.

• Table 1. Permanently Assigned Hardware Control Words

Address Dec	Address Hex	Length In Bytes	Purpose
0	0	8	IPL PSW
8	8	8	IPL CCW1
16	10	8	IPL CCW2
24	18	8	External old PSW
32	20	8	Supervisor call old PSW
40	28	8	Program old PSW
48	30	8	Machine check old PSW
56	38	8	I/O old PSW
64	40	8	Channel Status Word
72	48	4	Channel Address Word
76	4C	4	Unused
80	50	4	Timer
84	54	4	Unused
88	58	8	External new PSW
96	60	8	Supervisor call new PSW
104	68	8	Program new PSW
112	70	8	Machine check new PSW
120	78	8	I/O new PSW

Cause of the Dump: Evaluate the PSWs that appear in the formatted section of the dump (the first four lines) to find the cause of the dump. (For PCP, the IEAPRINT program places the PSWs in the dump header; they are appropriately labeled.) The PSW has the following format:

Program Status Word



- Does the instruction address field of the old machine check PSW show either the value E2 or E02? If so, a hardware error has occurred.
- Does the instruction address field of the old program check PSW have a value other than zero? If so, a program check at the instruction preceding that address caused the interruption.

Task Structure

PCP: Since there is only one task in the system, there is only one TCB. This TCB is always at location 384 (180 hexadecimal) in the main storage dump.

MFT (Without Subtasking): There is a TCB associated with each partition of main storage there are also TCBs for critical system tasks such as the master scheduler task and the transient area loading task. Table 1 shows location 76 (4C) unused for hardware control words. The control program uses this word to contain a pointer to the CVT. Use this CVT pointer to locate the first byte of the CVT, then the CVTIXAVL field (offset 124) in the CVT. The address contained at CVTIXAVL is a pointer to the IOS freelist. At offset 4 in the IOS freelist is a pointer to the first address in a list of TCB addresses. You can look through this list of TCB addresses, and, keeping your system options in mind, find the TCBs for each partition. The TCB addresses are listed in the following order:

- Transient area loading task.
- System error task (MFT with subtasking).
- Multiple console support write-to-log task (optional).
- I/O recovery management support task (optional).
- Communications task.
- Master scheduler task.
- System management facilities task (optional).

- Partition 0 task.
- Partition 1 task.
- •
- •
- Partition n task.

Figure 29 shows how to locate the partition TCBs in sample output from the IMDPRDMP program.

<u>MFT With Subtasking</u>: For MFT subtasking (and for MVT), a task may create a subtask. The partition TCBs for MFT with subtasking are referred to as <u>job step TCBs</u>. The task structure for a job step may be reconstructed in a main storage dump by using the information in Diagram 1.

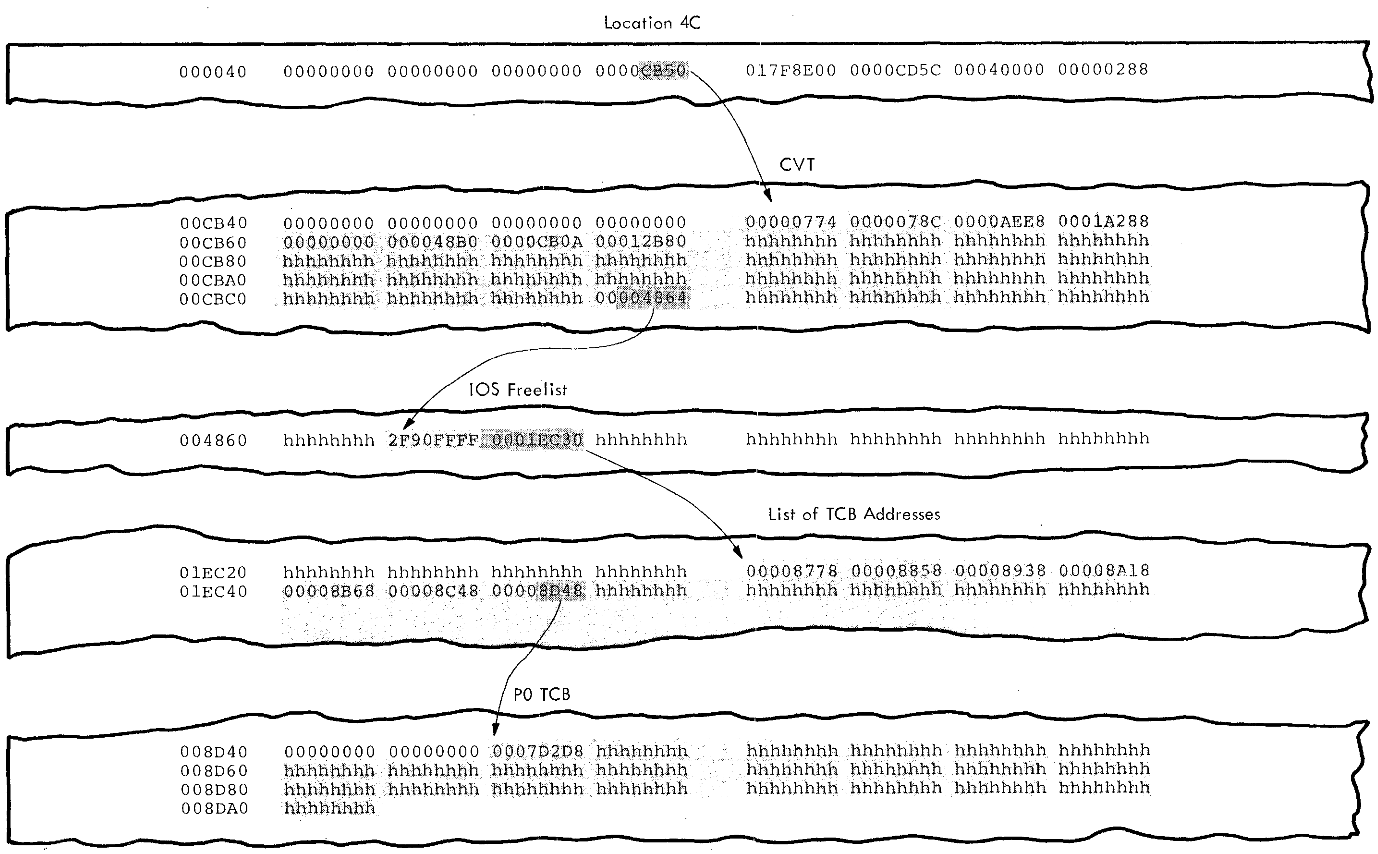
For MFT with subtasking, the job step TCB may be found using the method described for MFT without subtasking or by a more direct method. CVT offset 245 (F5) contains a pointer to the partition 0 job step TCB address in this address table.

To recreate the task structure within any partition, simply locate the job step TCB, and follow the TCB pointers - as explained in the previous section.

<u>MVT</u>: To find the current TCB, look at location 76 (4C) for a pointer to the CVT. The first word of the CVT contains a pointer to a doubleword of TCB addresses, which contains pointers to the next TCB to be dispatched (first word) and the current TCB (second word). Beginning with the current TCB, you can recreate the task structure for the job step using the methods in Diagram 1.

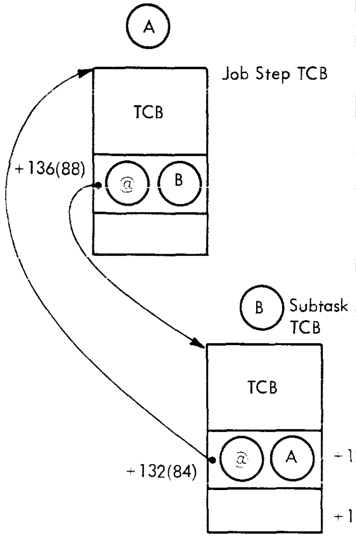
If the first word of the current TCB points to itself, there are no ready tasks to be dispatched, and the system has been placed in an enabled wait state. This TCB, now in control, is called the <u>system wait TCB</u>.

All TCBs in the system are maintained in a queue called the CVT ready queue. These TCBs are queued according to their dispatching priority. The CVTHEAD field, offset +160 (A0) in the CVT, contains the address of the highest priority TCB in the system. Offset +116 (74) in the TCB points to the TCB with the next lowest priority. Diagram 1 shows how to locate all of the TCBs in the system.

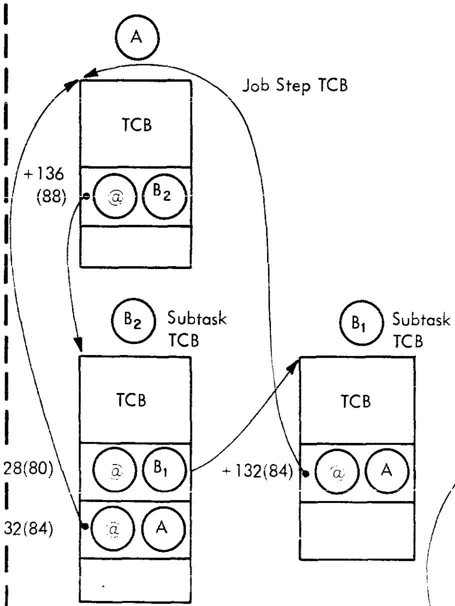


• Figure 29. Finding the Partition TCBs in MFT

• Diagram 1. Finding the TCB



(A) is a job step TCB and (B) is the TCB of the subtask created by (A). Offset +136(88) in (A) points to its subtask TCB ((B)). Offset +132(84) in the subtask TCB ((B)) points back to the job step TCB ((A)).



(A) is a job step TCB. (B_1) is the TCB for the first subtask created by (A). (B_2) is the TCB for the second and most recent subtask created by (A). Offset +136(88) in (A) points to the TCB of its most recently created subtask. Offset +136(84) in (B_2) points back to the creating task ((A)). Offset +128(80) in (B_2) points to (B_1) the next most recently created subtask TCB. Offset +132(84) in (B_1) points back to the originating TCB ((A)).

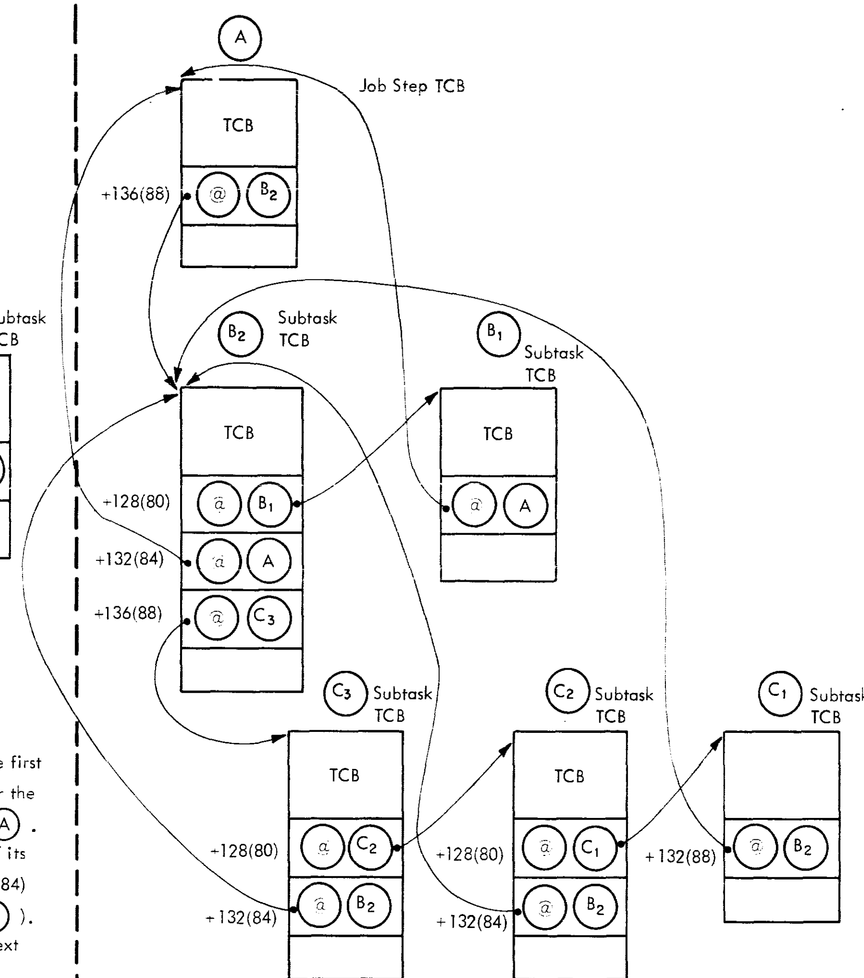
In each TCB:

Offset

+128(80) points to the TCB of the next most recently created subtask. If none exists, this field is zero.

+132(84) points to the TCB of the task that created it. If none exists, this field is zero.

+136(88) points to the TCB of the most recent subtask created by this task. If none exists, this field is zero.



(A) is the job step TCB. (B_1) is the TCB for the first subtask created by (A). (B_2) is the TCB for the second and most recent subtask created by (A). Offset +136(88) in (A) points to the TCB of its most recently created subtask. Offset +132(84) in (B_2) points to the TCB of the creating task. Offset -128 in (B_2) points to the next most recently created subtask TCB. Offset +132(84) in (B_2) points back to the job step TCB ((A)). Offset +136(88) in (B_2) points to the TCB of its most recently created subtask ((C_3)).

(C_3) points to the TCB of its creating task ((B_2)) and to the TCB of the subtask most recently created by (B_2). (C_2) contains pointers to the TCB of the originating task ((B_2)) and to the TCB of the task most recently created by (B_2). (C_2) contains only a pointer to the TCB of the invoking task ((B_2)).

Keep in mind that all TCBs in the system appear on this queue. Therefore, not only does a particular job step TCB appear on the ready queue, but all of its subtask also appear.

You can find the job step TCB associated with any TCB by using the TCBJSTCB field of the TCB, offset +124 (7C). This field contains the address of the job step TCB for the TCB you are examining.

In response to the FORMAT control statement, the IMDPRDMP program will do most of this work for you. It will recreate the task structure, format all TCBs in the system, and provide a TCB summary. The TCB summary shows the task structure. Figure 30 shows a portion of the TCB summary information from an MVT system. TCBs associated with a particular job are grouped together under the job name and step name. The TCB summary contains the TCB address, the completion code, and, when applicable, the address of the originating TCB and the addresses of created TCBs.

<u>Task Status - Active RB Queue</u>

The first word of the TCB contains a one-word pointer to the first word of the most recent RB added to the queue. In its eighth word, RB+28(1C), each RB contains a pointer to the next most recent RB. The last RB points pack to the TCB.

You can determine the idenity of the load module by looking either in the first and/or second words of the RB for its EBCDIC name or in the last 3 digits of the resume PSW in the previous RB for its SVC number. The entry point to the module is in the last 3 bytes of the fourth word in the RB, RB-13(D).

In MVT system, the name and entry point of the associated load module are not always contained in the RB associated with the module. Instead, they are found in a contents directory entry.

The address of the contents directory entry for a particular load module is given in the fourth word of the RB, RB+12(C). The CDE gives the address of the next entry in the directory (bytes 1-3), the name of the load module, bytes 8-15(F); the entry points of the module, bytes 17-19(11-13).

Figure 31 shows the formatting that the IMDPRDMP program does for a task in an MVT system. Notice the connection between the RB and the CDE. The IMDPRDMP program extracts the CDE information and displays this information with the RB.

The wait-count field of the RB is particularly important when locating the TCB by using the CVT ready queue (CVTHEAD). The high-order byte of the RB link field, RB-28(1C), of the most recent RB for a TCB contains a count of the number of events for which the task is waiting. Tasks that have a zero wait count are ready to be dispatched. Such a task will be dispatched or become the current task when all TCBs of higher priority are waiting for the completion of an event. To determine the events for which a task is waiting, use the

* * * * T C B S U M M A R Y * * * *

JOB	MASTER	STEP SCHEDULER				
	TCBhhhhhh	CMPhhhhhhhh	NTChhhhhhhh	OTChhhhhhhh	LTChhhhhhhh	PAGE hhhh
JOB	MASTER	STEP SCHEDULER				
	TCBhhhhhh	CMPhhhhhhhh	NTChhhhhhhh	OTChhhhhhhh	LTChhhhhhhh	PAGE hhhh
JOB	WTR	STEP 00E				
	TCBhhhhhh	CMPhhhhhhhh	NTChhhhhhhh	OTChhhhhhhh	LTChhhhhhhh	PAGE hhhh
	TCBhhhhhh	CMPhhhhhhhh	NTChhhhhhhh	OTChhhhhhhh	LTChhhhhhhh	PAGE hhhh
JOB	JOB11	STEP GO				
	TCBhhhhhh	CMPhhhhhhhh	NTChhhhhhhh	OTChhhhhhhh	LTChhhhhhhh	PAGE hhhh
	TCBhhhhhh	CMPhhhhhhhh	NTChhhhhhhh	OTChhhhhhhh	LTChhhhhhhh	PAGE hhhh
	TCBhhhhhh	CMPhhhhhhhh	NTChhhhhhhh	OTChhhhhhhh	LTChhhhhhhh	PAGE hhhh
JOB	JOB12	STEP GO				
	TCBhhhhhh	CMPhhhhhhhh	NTChhhhhhhh	OTChhhhhhhh	LTChhhhhhhh	PAGE hhhh

• Figure 30. IMDPRDMP TCB Summary

instruction address field in the resume PSW to locate the WAIT macro instruction in the source program. This will point you to the operation being executed at the time of the dump.

Main Storage Contents

Load List (PCP and MFT): The load list is a chain of request blocks associated with load modules invoked by a LOAD macro instruction. By looking at the load list, and at the job pack area queue described below, you can determine which system and problem program routines were loaded before the dump was taken. To construct the load list associated with the task in control, look at the tenth word in the TCB, TCB+36(24), for a pointer to the most recent RB entry on the load list, minus 8 bytes (RB-8). This word, in turn, points to the next most recent entry (minus 8), and so on. If this is the last RB, RB-8 will contain zeroes. The word preceding the most recent RB on the list (RB-4) points back to the TCB's load list pointer.

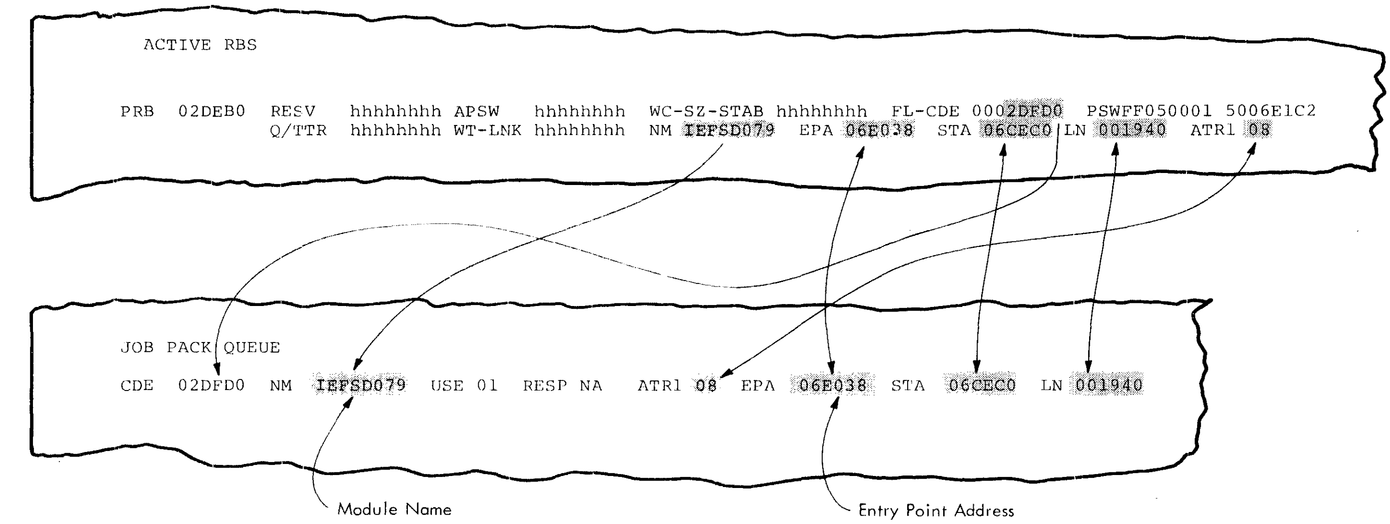
Load List (MVT): To construct the load list associated with the task in control, look at the tenth word in the TCB, TCB+36(24), for a pointer to the most recent load list entry (LLE). Each LLE contains the address of the next most recent entry (bytes 0-3), the count (byte 4), and the address of the CDE for the associated load module (bytes 5-7). If this is the last LLE in the list, TCB+36(24) will contain zeroes.

Job Pack Area Queue: In systems with MFT with subtasking and with MVT system, the job pack area queue is used to maintain reenterable modules within a partition or region. The complete description of this queue is found under the topic "Task Structure-Active RB Queue".

To reconstruct the job pack area queue in an MFT system with subtasking, look at TCB+125(7D) for a three byte pointer to the partition information block (PIB). The twelfth word of the PIB, PIB+44(2C), points to the most recent RB on the job pack area queue minus 8 bytes (RB-8). This word in turn points to the next most recent RB minus 8, and so on. The last RB will have zero in this field. The word preceding the most recent RB on the queue (RB-4) points back to the job pack area queue pointer in the PIB. You can determine the identity of the load module by looking either in the first and/or second word of the RB for its EBCDIC name, or in the last three digits of the resume PSW in the previous RB for the SVC number. The entry point of the module is given in the last three bytes of the fourth word in the RB, RB+29(1D), unless it is an FRB.

The first five words of an FRB (beginning at offset minus 8) are identical in content to those of other RBs. The XRWTL field, offset 12(C), contains the address of a wait list element. The first word of the WLE points to the next WLE, or contains zeros if the WLE is the last one. The second word to the waiting SVRB. You can determine the number of deferred requests for the module by tracing the chain of WLEs.

The XRREQ field of an FRB, offset 16(10), contains a pointer to the TCB of the requesting task. The next word, CRTLPRB, offset 20(14), points to an LPRB



• Figure 31. Determining Module From CDE in MVT built by the Finch routine for the requested program. The FRB for the requested program is removed from the job pack area queue by the Finch routine when the program is fully loaded.

In MVT, the job pack area queue is maintained in the same manner as the load list. The distinction between the two queues is that the job pack area queue contains reenterable programs. There are no FRBs in MVT.

Main Storage Supervision

Free Areas in Non-MVT Systems: Areas of main storage that are available for allocation at the time the dump was taken are described by the MSS boundary box and a series of free queue elements (FQEs). The seventh word of the TCB for the task, TCB+24(18), points to a six-word MSS boundary box. The first word of the MSS boundary box points to the FQE with the highest processor storage address in the partition (hierarchy 0), and the fourth word, to the highest 2361 Core Storage address in the partition (hierarchy 1). The first word of each FQE points to the next lower FQE; the second word of the FQE gives the length of the area it describes. FQEs occupy the first 8 bytes of the area they describe.

Gotten Subtask Areas: In MFT with subtasking, areas of a partition allocated by the system to a subtask within the partition are described by gotten subtask area queue elements (GQEs). The seventh word of the subtask TCB, TCB+24(18), points to a one word pointer to the most recently created GQE on the GQE queue. Bytes 0 through 3 of the GQE contain a pointer to the previous GQE or, if zero, indicate that the GQE is the last one on the queue. Bytes 4 through 7 of the GQE contain the length of the gotten subtask area. Each GQE occupies the first eight bytes of the gotten subtask area it describes.

Region Structure in MVT System: The region associated with a particular task in an MVT system is described by partition queue elements (PQEs). The thirty-ninth word of the TCB, offset +152 (98) contains a pointer to the dummy PQE (D-PQE) for the region. The first word of the dummy PQE points to the first PQE and the second word, to the last PQE. The first and second words of each PQE point to the first and last free block queue elements (FBQEs), respectively, associated with the PQE. Separate PQEs are used to describe parts of a region in different storage hierarchies or part of a region that was obtained by another task which has been rolled out.

FBQEs describe free areas in the region that have a a length which is a multiple of 2048 bytes. These free areas are available for allocation to a specific subpool.

Subpool Descriptions (SPQEs) The seventh word of the TCB, TCB+24(18), points to the SPQE representing the first subpool used by the task. Each SPQE contains the address of the next SPQE (bytes 1-3), the subpool number (byte 4), and the address of the first descriptor queue element (DQE) for the subpool (bytes 5-7) or, if the subpool is owned by another task (bit 0 is 1), the address of the SPQE that describes it (bytes 5-7).

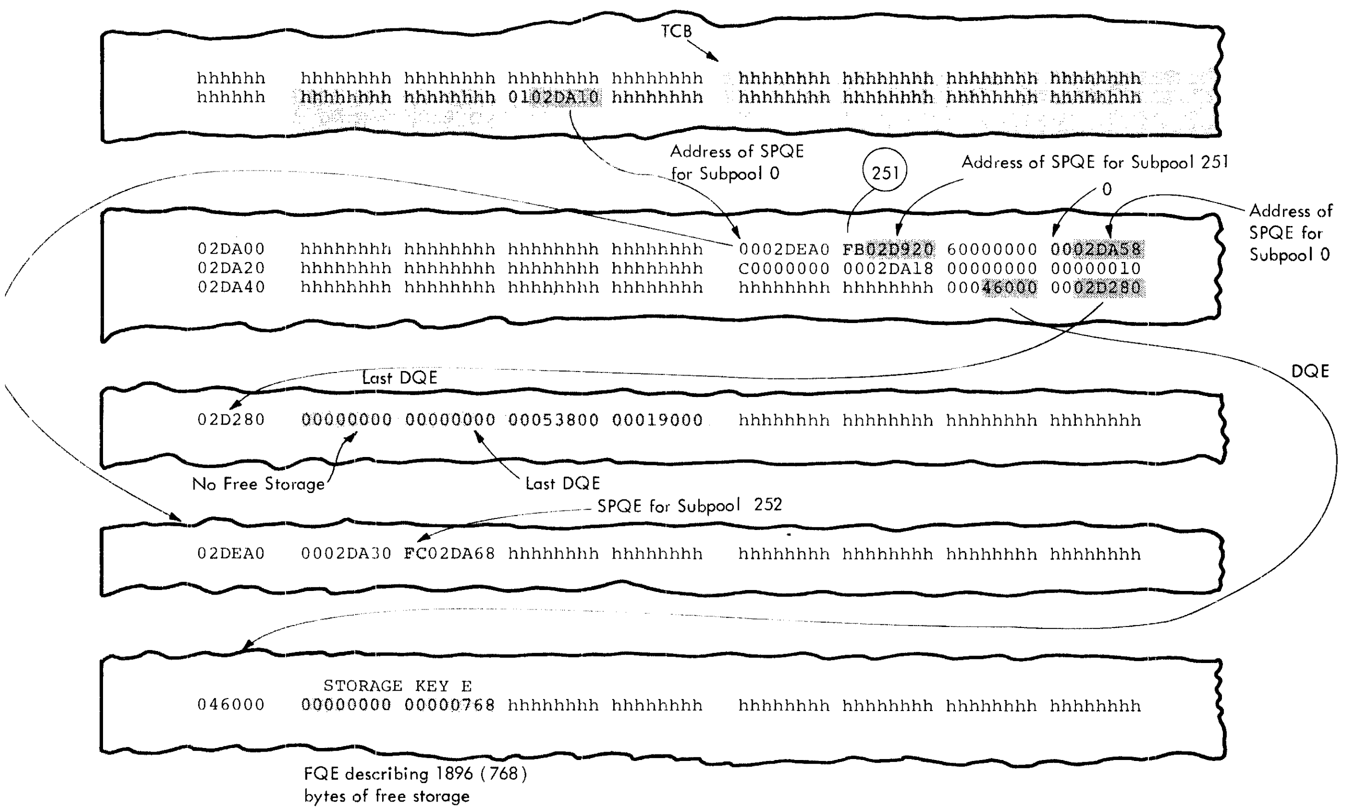
Storage within a subpool is described by a descriptor queue element. Each DQE contains the number of bytes of main storage in the subpool. This count is always a multiple of 2048 bytes. If a request for space from a subpool cannot be satisfied with the space described by an existing DQE the GETMAIN routine builds another DQE and links the new DQE to the chain of existing DQE's. Each DQE contains a pointer to the FQE that represents the free area with the highest main storage address in the subpool (bytes 1-3), a pointer to the next DQE (bytes 5-7), and the length of the area described by the DQE, bytes 13-15(D-F).

Figure 32 shows the control blocks used to describe the subpools for a task in an MVT system.

I/O Control Blocks

Queue of DEBs: To find the queue of DEBs for the task, look at the third word in the TCB (TCB+8). The address given here points to the first word of the most recent entry on the DEB queue. There is a DEB on this queue for each data set opened to the task at the time of the dump. DEBs are enqueued in the same order as the data sets are opened. The last three bytes of the second word in each DEB (DEB+5) points to the next most recent DEB on the queue. The queue contains one DEB for each open data set.

UCBs: You can find unit information for each device in your system in the unit control block (UCB) for that device. The address of the UCB is contained in the last 3 bytes of the ninth word of the DEB, DEB+33(21). If the DEB queue is empty, scan the dump around location 4096(1000) for words whose fifth and sixth digits are FF. These are the first words of the UCBs for the system; UCBs are arranged in numerical order by device address. (You may find it easier to locate UCBs by looking for the device address in the EBCDIC printout to the right of each page.) The first two bytes of the second word of



• Figure 32. Subpool Descriptions in MVT - IMDPRDMP Storage Print

each UCB give the device address. The device type and class are given in the third and fourth bytes of the fifth word, UCB+18(12), respectively. The sense bytes are given in the last two bytes of the sixth UCB word, UCB-22(16), and extend for from 1 to 6 bytes, depending on the device type. Sense bytes are explained in Appendix F.

DCB and TIOT: The address of the DCB, a control block that describes the attributes of an open data set, is located in the last 3 bytes of the seventh DEB word, DEB+25(19). The first two bytes of the ninth word of the DCB, offset 40(28), contains the offset in the task input/output table (TIOT) of the DD name entered for the data set. Therefore, the address of the DD name for a particular data set may be found by adding the TIOT offset in the DCB to the TIOT address in the TCB (TCB+12), plus 24(16) bytes for the TIOT header.

IOB: If a data set is being accessed by a sequential access method with normal scheduling, the address of the input/output block (IOB) prefix (IOB-8) is located in the seventeenth word of the DCB, DCB-68(44). The first word of the IOB prefix points to the next IOB (if more than one IOB exits for the data set). Each IOB for an open data set contains a pointer to the CCW list in the last three bytes of the fifth word, IOB+17(11).

ECB: The completion code for an I/O operation is posted in the first byte of the event control block (ECB). ECB completion codes are explained in Appendix E. If the I/O event is not complete and an SVC I (WAIT) has been issued, the high-order bit of the ECB is on, and bytes one through three contain the address of the associated RB. For the sequential and basic partition access methods the second word of an IOB points to its associated ECB.

Figure 33 shows the DEB, UCB, DCB, and IOB for a BSAM data set.

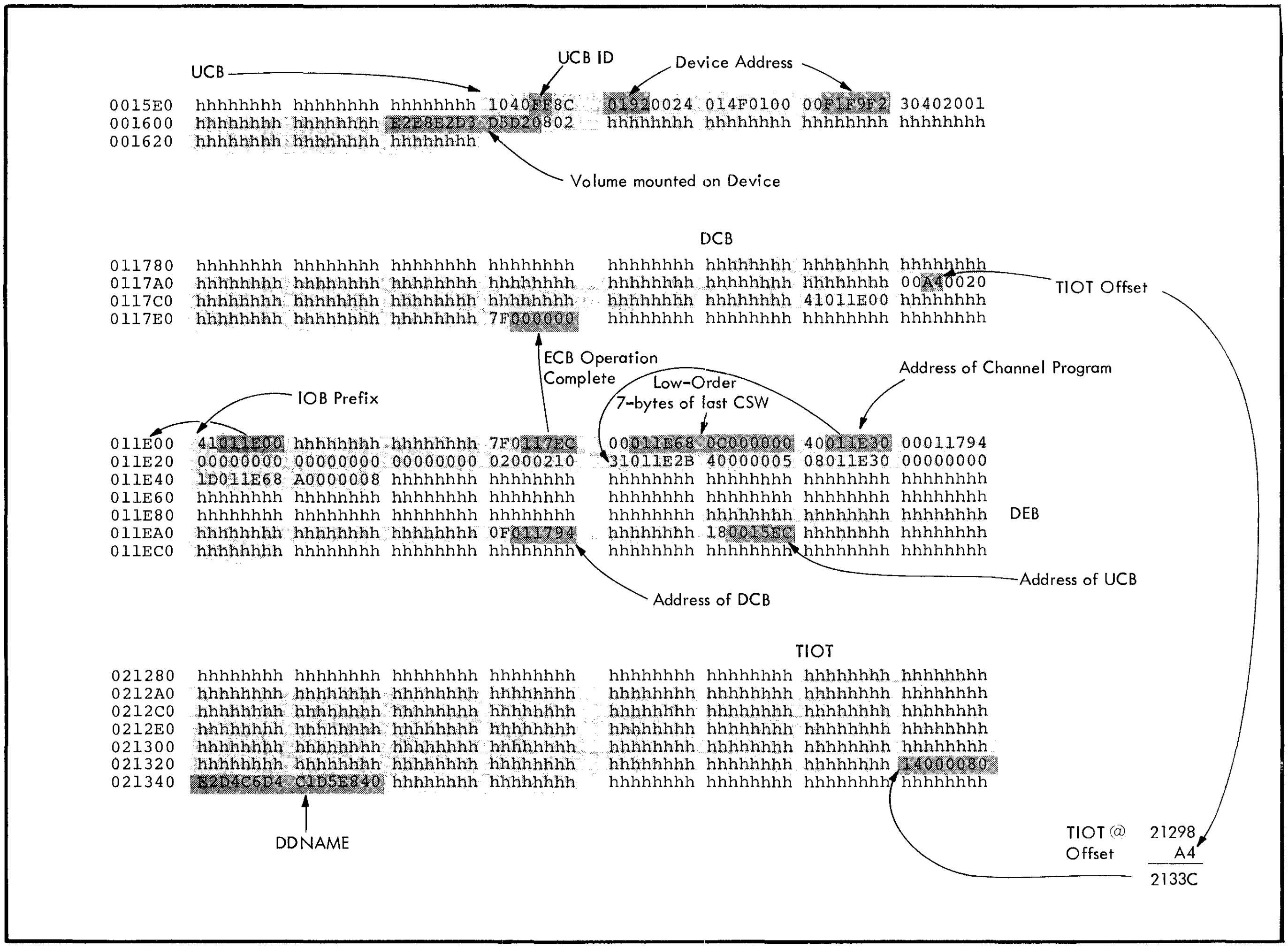


Figure 33. I/O Control Blocks

TSO Control Blocks

The time sharing (TSO) control blocks are obtained from the IMDPRDMP service aid program by specifying the TSO control statement in the input stream. The first part of the TSO dump is the same as the normal MVT dump. The control blocks that IMDPRDMP formats are divided into two group: system and user.

TSCVT: The time sharing communications vector table (TSCVT) is a secondary CVT for the MVT CVT. The time sharing CVT resides in the time sharing region; therefore, it exists only while the time sharing region is active. When time sharing does not exist in the system, the MVT CVT pointer to the TSCVT (CVT+229) is zero.

RCB: A region control block (RCB) contains information that is unique to a time sharing region. There is one RCB for each time sharing region. The RCBs reside in the time sharing controller's region, they are contiguous, and they are created during initialization of the time sharing controller.

The TSCVT points to a region control block table. The RCB table is an indexed table containing one RCB address for each possible time sharing region, therefore, the table contains the maximum number of RCBs that may be used by time sharing. The first RCB is for region one, the second for region two, etc. The time sharing job block (TJB) of a job points to the RCB associated with that job.

UMSM: One user main storage map (UMSM) exists for each possible time sharing user. The UMSM contains a series of consecutive one-word extent fields (ADDR-LN). Each one-word extent contains a halfword address field (ADDR) and a halfword length field

(LN) that describes the main storage allocated to the time sharing user. The UMSM contains the address and length of a storage block (a multiple of 2K bytes) that has been allocated to the user; only this allocated storage will be swapped out for the user. The time sharing job block (TJB) points to the UMSM.

SWAP DCB: The swap data control block (SWAP DCB) is used whenever a time sharing user's region is swapped into or out of main storage. It describes a swap data set that contains an IOB, area for channel programs, and the track map queue. The TJB points to the swap DCB.

TJB: The time sharing job block (TJB) contains status information about a time sharing user. The TJB is retained in main storage while the user is swapped out. One time sharing job block exists for each possible simultaneous time sharing user. The space for the TJB is obtained from the time sharing control task (TSC) region during time sharing initialization. Status information about the terminal related to this TJB is contained in the terminal status block (TSB). The address of the terminal status block is the first word of the TJB. The first word of the TSCVT points to the TJB.

TSB: Each terminal status block (TSB) contains status information about one terminal. The terminal input/output coordinator (TIOC) uses this information. During system initialization, one TSB is created for each possible user. The main storage space is obtained in one contiguous block for all of the TSBs in the region of the time sharing control task (TSC); this contiguous string of TSBs is called the TSB table. The origin pointer to the TSB table is the TIOCTSB field of the TIOCRPT.

TJBX: The time sharing job block extension (TJBX) contains user job information that can be rolled out to the swap data set with the user's job. The TJBX resides in the local system queue space (LSQS) for the region. The TJBX location is pointed to by the third word of the time sharing job block (TJB). The space for the TJBX is obtained by the region control task (RCT) during initialization.

PSCB: The protected step control block (PSCB) contains accounting information related to a single user. All timing information is in software timer units. A software timer unit is equal to 26.04166 micro seconds. The job step control block (JSCB), offset 268, points to the PSCB.

TAXE: The TSO terminal attention exit element (TAXE) is a physical addendum to a regular 24 word interrupt request block (IRB). It is used to schedule an attention exit resulting from a terminal attention interruption. It is created, queued, and dequeued by the specify terminal attention exit (STAX) macro instruction. The main storage space for the TAXE is obtained in the local system queue space (LSQS) of the terminal user's region.

For a more detailed description of the TSO control blocks formatted by the IMDPRDMP program, see the publication IBM System/360 Operating System: Service Aids, GC28-7619.

Trace Table

Find the Trace Table: Location 84(54) in main storage contains the address of the first word of the three word trace table control block. The trace table control block immediately preceeds the table. The trace table control block describes the bounds of the table and the most recent entry at the time of the dump.

Current Entry	First Entry	Last Entry
0	4	8

You can locate the trace table by scanning the contents of main storage between locations 16,384(4000) and 32,768(8000) for trace table entries. Entries are four words long and begin at addresses ending with zero. To find the table boundaries and current entry, scan the table in reverse until you reach the trace table control block.

Trace Table Entries in PCP and MFT: Trace table entries for systems with PCP and MFT are 4 words long and represent occurrences of SIO, I/O, and SVC interruptions. Figure 34 gives some sample entries and their contents.

SIO entries can be used to locate the CCW (through the CAW), which reflects the operation initiated by an SIO instruction. If the SIO operation was not successful, the CSW STATUS portion of the entry will show you why it failed.

I/O entries reflect the I/O old PSW and the CSW that was stored when the interruption occurred. From the PSW, you can learn the address of the device on which the interruption occurred (bytes 2 and 3), the CPU state at the time of interruption (bit 15), and the instruction address where the interruption occurred (bytes 5-8). The CSW provides you with the unit status (byte 4), the channel status (byte 5), and the address of the previous CCW plus 8 (bytes 0-3).

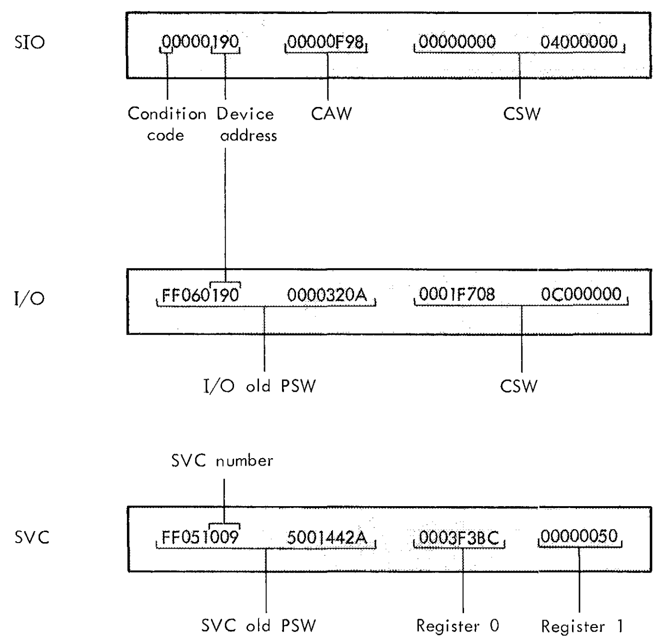


Figure 34. Sample Trace Table Entries (PCP and MFT)

SVC entries provide the SVC old PSW and the contents of registers 0 and 1. The PSW offers you the hexadecimal SVC number (bits 20-31), the CPU mode (bit 15), and the address of the SVC instruction (bytes 5-8(. The contents of registers 0 and 1 are useful in that many system macro instructions use these registers for parameter information. Contents of registers 0 and 1 for each SVC interruption are given in Appendix A.

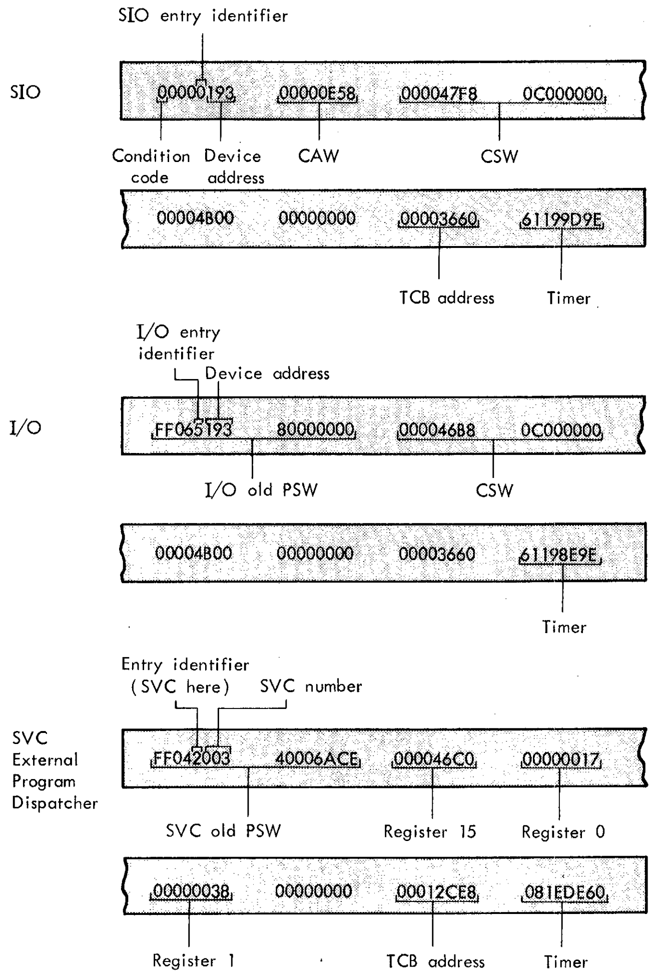
Trace Table Entries in MVT and M65MP: Entries in an MVT trace table are 8 words long and represent occurrences of SIO, external, SVC, program, I/O, and dispatcher interruptions. You can identify what type of interruption caused an entry by looking at the fifth digit:

0 = SIO
1 = External
2 = SVC
3 = Program
5 = I/O
D = Dispatcher

Figure 35 gives some sample entries and their contents.

In dumps of Model 65 Multiprocessing system, trace table entries differ as follows:

SIO	5th word	address of TCB.
	6th word:	address of old TCB for CPU A.
	7th word:	address of old TCB for CPU B.
	8th word	CPU identification (last byte).
I/O	3rd word:	contents of register 15.
	4th word	contents of register 0.
	8th word	CPU identification (last byte).
SVC and Program	6th word:	address of old TCB for APU A.
	7th word:	address of old TCB for CPU B.
	8th word	CPU identification (last byte).
Dispatcher	6th word:	address of new TCB for CPU A.
	7th word:	address of new TCB for CPU B.
	8th word:	CPU identification (last byte).
External	6th word:	STMASK of other CPU.
	7th word:	TQE if timer interrupt occurred.
	8th word:	CPU identification (last byte). If so, a program check at the instruction preceding that address caused the interruption.



• **Figure 35. Sample Trace Table Entries (MVT)**

Appendix A: SVCs

Register contents at entry to an SVC routine are often helpful in finding pointers and control information. The table below lists SVC numbers in decimal and hexadecimal, and gives the type, associated macro instruction, and significant contents of registers 0 and 1 at entry to each SVC routine.

Macro instructions followed by an asterisk (*) are documented in the System Programmers Guide. Expanded descriptions of remaining macro instructions listed here may be found in the publication Supervisor and Data Management Macro Instructions. Graphics and telecommunications macro instructions are discussed in the Program Logic Manuals associated with these access methods.

Decimal No.	Hex. No.	Type	Macro	Register 0	Register 1
0	0	I	EXCP *		IOB address
0	0	I	XDAP *		
1	1	I	WAIT	Event count	ECB address
1	1	I	WAITR	Event count	2's complement of ECB address
1	1	I	PRTOV		
2	2	I	POST	Completion code	ECB address
3	3	I			
4	4	I	GETMAIN		Parameter list address
5	5	I	FREEMAIN		Parameter list address
6	6	II	LINK		Parameter list address
7	7	II	XCTL		Parameter list address
8	8	II	LOAD	Address of entry point address	DCB address
9	9	I, II	DELETE	Address of program name	
10	A	I	GETMAIN or FREEMAIN (R Operand)	Subpool number (byte 0) Length (bytes 1-3)	Address of area to be freed
10	A	I	FREEPOOL		
11	B	I, III	TIME		Time units code
12	C	II	SYNCH *		
13	D	IV	ABEND		Completion code
14	E	II, III	SPIE		PICA address
15	F	I			Address of request queue element

(Part 1 of 5)

Decimal No.	Hex. No.	Type	Macro	Register 0	Register 1
16	10	III	PURGE *		
17	11	III	RESTORE *		IOB chain address
18	12	II	BLDL	Address of build list	DCB address
18	12	II	FIND		
19	13	IV	OPEN		Address of parameter list of DCB addresses
20	14	IV	CLOSE		Address of parameter list of DCB addresses
21	15	III	STOW	Parameter list address	DCB address
22	16	IV	OPEN TYPE=J*		Address of parameter list of DCB addresses
23	17	IV	CLOSE TYPE=T		Address of parameter list of DCB addresses
24	18	III	DEVTYPE *		ddname address
25	19	III			DCB address
26	1A	IV	CATALOG *		Parameter list address
26	1A	IV	INDEX *		Parameter list address
26	1A	III	LOCATE *		Parameter list address
27	1B	III	OBTAIN *		Parameter list address
28	1C	IV			
29	1D	IV	SCRATCH *	UCB address	Parameter list address
30	1E	IV	RENAME *	UCB address	Parameter list address
31	1F	IV	FEOV		DCB address
32	20	IV			Address of UCB list
33	21	III	IOHALT		UCB address
34	22	IV	MGCR (MAST CMD EXCP)		
35	23	IV	WTO		Message address
35	23	IV	WTOR		Message address
36	24	IV	WTL		Address of message
37	25	II	SEGLD		Segment name address
37	25	II	SEGWT		Segment name address
38	26	II			
39	27	III,IV	LABEL		Parameter list address

(Part 2 of 5)

Decimal No.	Hex. No.	Type	Macro	Register 0	Register 1
40	28	I, II, III	EXTRACT		Parameter list address
41	29	II, III	IDENTIFY	Entry point name address	Entry point address
42	2A	II, III	ATTACH		
43	2B	II, III	CIRB *	Entry point address	Size of work area in doublewords
44	2C	I	CHAP	+ Increase priority - Decrease priority	TCB address
45	2D	II			
46	2E	I	TTIMER		1: Cancel
47	2F	II	STIMER	Exit address	Timer interval address
48	30	I, II	DEQ		QCB address
49	31	III	TEST		
50	32	IV			
51	33	IV	SNAP		Parameter list address
52	34	IV			DCB address
53	35	III	RELEX	Key address	DCB address
54	36	II			
55	37	IV	EOV *	EOB address	DCB address
56	38	I, II	ENQ	QEL address	QCB address
56	38	I, II	RESERVE *		
57	39	III	FREEDBUF	DECB address	DCB address
58	3A	I	RELBUF		DCB address
58	3A	I	REQBUF		DCB address
59	3B	III			
60	3C	III	STAE	0 Create SCB 4 Cancel SCB 8 0	Parameter list address
61	3D	III			Parameter list address
62	3E	II	DETACH		TCB address
63	3F	IV	CHKPT		DCB address
64	40	III	RDJFCB *		Address of parameter list of DCB addresses
65	41	II			Parameter list address
66	42	IV			

(Part 3 of 5)

Decimal No.	Hex. No.	Type	Macro	Register 0	Register 1
67	43	II	ENDREADY		QPOST
68	44	IV	SYNADAF	Same as register 0 on entry to SYNAD	Same as register 1 on entry to SYNAD
68	44	IV	SYNADRLS		
69	45	III	BSP		DCB address
70	46	II	GSERV		Parameter list address
71	47	III	RLSEBFR		Parameter list address
71	47	III	ASGNBFR		Parameter list address
71	47	III	BUFINQ		Parameter list address
72	48	IV			Parameter list address
73	49	III	SPAR		Parameter list address
74	4A	III	DAR		Parameter list address
75	4B	III			Parameter list address
76	4C	IV			
77	4D	IV			
78	4E	III			
79	4F	I	STATUS		
80	50	III			
81	51	IV	SETPRT		
82	52	IV			
83	53	III	SMFWTM *		Message address
84	54	I		UCB address and buffer restart address	
85	55	IV			
86	56	IV	ATLAS		Parameter list address
87	57	III	DOM	If zero If negative	A DOM message I.D. A pointer to a list of DOM message I.Ds
88	58	III	MOD88	Routine code	DCB address
89	59	III	EMSRV		Parameter list address
90	5A	IV	XQMNGR	Address of list of ECB/IOB pointers (optional)	QMPA address
91	5B	III	VOLSTAT	DCB address	zero: issued by CLOSE Non-zero: issued by EOV

(Part 4 of 5)

Decimal No.	Hex. No.	Type	Macro	Register 0	Register 1
92	5C	I			
93	5D	IV	TGET/TPUT	TJID & buffer Size	Address of User's Buffer
94	5E	IV	STERMINAL STATUS	Entry code	
95	5F	I	TSEVENT	TJID/Entry Code or 0	Not Always Applicable
96	60	III	STAX		Parameter List Address
97	61	III			
98	62	IV	PROTECT		Parameter List Address
99	63	IV	none		
100	64	III	FIB		
101	65	I	QTIP	Entry code	Parameter List Address
102	66	I	AQCTL		Parameter List Address

(Part 5 of 5)

Appendix B: Completion Codes

Completion codes issued by operating system routines are often caused by problem program errors. This appendix includes the most common system completion codes, their probable causes, and how to correct the error or locate related information using a dump. For a more comprehensive coverage of completion codes, see the publication Messages and Codes.

0Cx A program check occurred without a recovery routine. If bit 15 of the old program PSW (PSW at entry to ABEND) is on, the problem program had control when the interruption occurred; "x" reflects the type of error that causes the interruption:

x	Cause
1	Operation
2	Privileged operation
3	Execute
4	Protection
5	Addressing
6	Specification
7	Data
8	Fixed-point overflow
9	Fixed-point divide
A	Decimal overflow
B	Decimal divide
C	Exponent overflow
D	Exponent underflow
E	Significance
F	Floating-point

The correct register contents are reflected under the heading "REGS AT ENTRY TO ABEND" in an ABEND/SNAP dump. In a stand-alone dump, register contents can be found in the register save area for ABEND'S SVRB.

0F1 A program check occurred in the interruption handling part of the input/output supervisor. The applicable program check PSW can be found at location 40(28). (In systems with MFT, this PSW is valid only if the first four digits are 0004).

The problem program can be responsible for this code if:

1. An access method routine in the problem program storage area has been overlaid.
2. An IOB, DCB, or DEB has been modified after an EXCP has been issued, but prior to the completion of an event.

If a trace table exists (trace option was specified at system generation), the instruction address in the new program check PSW, location 104(68), contains the address of a field of register contents. This field includes registers 10 through 1 (PCP) or 10 through 9 (MFT) on an ABEND/SNAP dump, or 10 through 1 (both systems) on a stand-alone dump.

If no trace table exists, the above field contains registers 10 through 1 on both ABEND/SNAP (MFT only) and stand-alone dumps.

0F2 Most frequently caused by incorrect parameters passed to a type I SVC routine.

100 A device has been taken off-line without informing the system, or a device is not operational.

If a trace table exists, the most current entry is an SIO entry beginning with 30. The last 3 digits of the first word give the device address.

If a trace table does not exist, register 1 (in the SVRB for the ABEND routine) contains a pointer to the IOB associated with the device.

101 The wait count, contained in register 0 when a WAIT macro instruction was issued, is greater than the number of ECBs being waited upon.

102 An invalid ECB address has been given in a POST macro instruction.

If a POST macro instruction has been issued by the problem program, the ECB address is given in register 1 of either the trace table entry or the SVRB for the ABEND routine.

If the POST was issued by an I/O interruption handler, the ECB address can be found in the IOB associated with the event.

106 During a transient area load or a dynamic load resulting from a LINK, LOAD, XCTL, or ATTACH macro instruction, the fetch routine found an error. A description of the error is contained in register 15 of ABEND's SVRB register save area:

0D The control program found an invalid record type.

0E The control program found an invalid address. The problem program may contain a relocatable expression that specifies a location outside the partition boundaries.

0F A permanent I/O error has occurred. This error can probably be found in the trace table prior to the ABEND entry.

Register 6 of ABEND's SVRB register save area points to the work area used by the fetch routine. This area contains the IOB, channel program, RLD buffer, and the BLDL directory entry associated with the program being loaded.

122 The operator cancelled the job and requested a dump.

155 An unauthorized user (a user other than dynamic device reconfiguration) has issued SVC 85. The user's task has been abnormally terminated by dynamic device recognition.

201 This completion code is identical to 102, but applies to the WAIT macro instruction instead of POST.

202 An invalid RB address was found in an ECB. The RB address is placed in the ECB when a WAIT macro instruction is issued.

213 The error occurred during execution of an OPEN macro instruction for a data set on a direct-access device. Either:

1. The data set control block (DSCB) could not be found on the direct access device.
2. An uncorrectable input/output error occurred in reading or writing the data set control block.

Register 4 contains the address of a combined work and control block area. This address plus x'64' is the address of the data set name in the JFCBDSNM field of the job file control block (JFCB).

222 The operator cancelled the job without requesting a dump. The cancellation was probably the result of a wait state or loop.

301 A WAIT macro instruction was issued, specifying an ECB which has not been posted complete from a previous event. Either:

1. The ECB has been reinitialized by the problem program prior to a second WAIT on the same ECB, or
2. The high order bit of the ECB has been inadvertently turned on.

308 The problem program requested the loading of a module using an entry point given to the control program by an IDENTIFY macro instruction.

Register 0 of LOAD's SVRB register save area contains the address (or its complement) of the name of the module being loaded.

400 The control program found an invalid IOB, DCB, or DEB. Check the following blocks for the indicated information:

- IOB - a valid DCB address.
- DCB - a valid DEB address.
- DEB - ID of 0F and a valid UCB address.
- UCB - a valid identification of FF.

Note: In systems with MVT, this code may appear instead of a 200 code, for the reasons given under 200.

406 A program has the "only loadable" attribute or has an entry point given to the control program by an IDENTIFY macro instruction. In either case, the program was invoked by a LINK, XCTL, or ATTACH macro instruction.

Register 15 of the LINK, XCTL, or ATTACH SVRB register save area contains the address of the name of the program being loaded.

506 The error occurred during execution of a LINK, XCTL, ATTACH, or LOAD macro instruction in an overlay program or in a program that was being tested using the TESTRAN interpreter.

The program name can be found as follows:

1. If a LOAD macro instruction was issued, register 0 in the trace table SVC entry or in the SVRB register save area contains the address (or its complement) of the program name.

2. If a LINK, XCTL, or ATTACH was issued, register 15 of the associated SVRB register save area contains the address of a pointer to the program name.

Note: Programs written in an overlay structure or using TESTRAN should not reside in the SVC library.

604 During execution of a GETMAIN macro instruction, the control program found one of the following:

1. A free area exceeds the boundaries of the main storage assigned to the task. This can result from a modified FQE.

2. The A-operand of the macro instruction specified an address outside the main storage boundaries assigned to the task.

605 During execution of a FREEMAIN macro instruction, the control program found that part of the area to be freed is outside the main storage boundaries assigned to the task, possibly resulting from a modified FQE.

Item 1 under the 604 completion code is also applicable to 605.

606 During execution of a LINK, XCTL, ATTACH, or LOAD macro instruction, a conditional GETMAIN request was not satisfied because of a lack of available main storage for a fetch routine work area. Consequently, the request was not satisfied.

The name of the load module can be found as described under completion code 506.

60A Results from the same situations described under 604 and 605 for R-form GETMAIN and FREEMAIN macro instructions.

613 The error occurred during execution of an OPEN macro instruction for a data set on magnetic tape. An uncorrectable input/output error occurred in tape positioning or in label processing.

700 A unit check resulted from an SIO issued to initiate a sense command.

The defective device can be determined from the SIO trace table entry that reflects a unit check in the CSW status.

704 A GETMAIN macro instruction requested a list of areas to be allocated. This type of request is valid only for systems with MVT.

The applicable SVC can be found in a trace table entry or in the PSW at entry to ABEND.

705 Results from the same situations described under 704 for FREEMAIN macro instructions.

706 During execution of a LINK, LOAD, XCTL, or ATTACH macro instruction, the requested load module was found to be not executable.

The name of the module can be found as described under the completion code 506.

804 The error occurred during execution of a GETMAIN macro instruction with a mode operand of EU or VU. More main storage was requested than was available.

806 The error occurred during execution of a LINK, XCTL, ATTACH, or LOAD macro instruction.

An error was detected by the control program routing for the BLDL macro instruction. This routine is executed as a result of these macro instructions if the problem program names the requested program in an EP or EPLOC operand. The contents of register 15 indicate the nature of the error:

X'04' The requested program was not found in the indicated source.

X'08' An uncorrectable input/output error occurred when the control program attempted to search the directory of the library indicated as containing the requested program.

Register 12 contains the address of the BLDL list used by the routine. This address plus 4 is the location of the 8-byte name of the requested program that could not be loaded.

80A The error occurred during execution of an R-form GETMAIN macro instruction. More main storage was requested than was available.

905 The address of the area to be freed (given in a FREEMAIN macro instruction) is not a multiple of eight. The contents of register one in either the trace table entry or ABEND's SVRB register save area reflect the invalid address.

90A Results from the same situations described under 905 for R-forms of GETMAIN and FREEMAIN macro instructions.

A05 The error occurred during execution of a FREEMAIN macro instruction. The area to be freed overlaps an already existing free area. This error can occur if the address or the size of the area to be freed were incorrect or modified.

The contents of registers 0 and 1 in either the SVC trace table entry or ABEND's SVRB register save area reflect the size and address.

A0A Results from the same situations described under A05 for R-form of GETMAIN and FREEMAIN macro instructions.

B04 This error occurred during execution of a GETMAIN macro instruction. A subpool number greater than 127 was specified. The problem program is restricted to using subpools 0-127. This error can occur if the subpool number was either incorrectly specified or modified.

A displacement of nine bytes from the list address passed to GETMAIN in register 1 contains the subpool number. Register 1 can be found in either the SVC trace table entry or ABEND's SVRB register save area.

B05 Results from the same situation described under B04 for a FREEMAIN macro instruction.

B0A Results from the same situations described under B04 and B05 for R-form of GETMAIN and FREEMAIN macro instructions.

The subpool number can be found in the high order bytes of register 0 in either the SVC trace table entry or ABEND's SVRB register save area.

B37 The error occurred at an end of volume. The control program found that all space on the currently mounted volumes was allocated, that more space was required, and that no volume was available for demounting.

Either allocate more devices or change the program so that a device will be free when a volume must be mounted.

Fnn An SVC instruction contained an invalid operand; nn is the hexadecimal value of the SVC.

This error can occur if either an invalid instruction was issued by the problem program or an operand referring to an optional function was not included during system generation.

Appendix C: System Module Name Prefixes

All load modules associated with a specific operating system component have a common prefix on their module names. This appendix lists the module name prefixes and the associated system component(s).

Prefix	Component
IBC	Independent utility programs
IEA	Supervisor, I/O supervisor, and NIP
IEB	Data set utility programs
IEC	Input/output supervisor
IEE	Master scheduler
IEF	Job scheduler
IEG	TESTRAN
IEH	System utility programs
IEI	Assembler program during system generation
IEJ	FORTRAN IV E compiler
IEK	FORTRAN IV H compiler
IEM	PL/I F compiler
IEP	COBOL E compiler
IEQ	COBOL F compiler
IER	Sort/Merge program
IES	Report program generator
IET	Assembler E
IEU	Assembler F
IEW	Linkage editor/overlay supervisor/program fetch
IEX	ALGOL compiler
IEY	FORTRAN IV G compiler
IFB	Environment recording routines
IFC	Environment recording and print routines
IFD	On line test executive program
IFF	Graphic programming support
IGC	Transient SVC routines
IGE	I/O error routines
IGF	Machine check handler program
IGG	Close, open, and related routines
IHA	System control blocks
IHB	Assembler during expansion of supervisor and data management macro instructions
IHC	FORTRAN library subroutines
IHD	COBOL library subroutines
IHE	PL/I library subroutines
IHF	PL/I library subroutines
IHG	Update analysis program
IHI	Object program originally coded in ALGOL language
IHJ	Checkpoint/restart
IHK	Remote job entry
IIN	7094 emulator program for the Model 85
IKA	Graphic Job Processor
IKD	Satellite graphic job processor messages
IKF	USAS COBOL compiler
ILB	USAS COBOL subroutines

Appendix D: List of Abbreviations

ABEND	abnormal end-of-task
APR	alternate path retry
CCW	channel command word
CDE	contents directory entry
CPU	central processing unit
CSW	channel status word
CVT	communications vector table
DAR	damage assessment routine
DCB	data control block
DD	data definition
DDR	dynamic device reconfiguration
DEB	data extent block
DPQE	dummy partition queue element
DQE	descriptor queue element
ECB	event control block
FBQE	free block queue element
FQE	free queue element
FRB	finch request block
GQE	gotten subtask area queue element
IOB	input/output block
IPL	initial program loading
IRB	interrupt request block
LLE	load list element
LPRB	loaded program request block
LRB	loaded request block
MFT	multiprogramming with a fixed number of tasks
MVT	multiprogramming with a variable number of tasks
NIP	nucleus initialization program
PCP	primary control program
PIB	partition information block
PQE	partition queue element
PRB	program request block
PSA	prefixed storage area
PSW	program status word
QCB	queue control block
QEL	queue element
RB	request block
SCB	STAE control block
SIO	start input/output
SIRB	supervisor interrupt request block
SPQE	subpool queue element
SVC	supervisor call
SVRB	supervisor request block
SYSOUT	system output
TCB	task control block
TIOT	task input/output table
UCB	unit control block
WLE	wait list element
XCTL	transfer control
XL	extent list

Appendix E: ECB Completion Codes

Hexadecimal Code	Meaning
7F000000	Channel program has terminated without error. (CSW contents can be useful.)
41000000	Channel program has terminated with permanent error. (CSW contents can be useful.)
42000000	Channel program has terminated because a direct access extent address has been violated. (CSW contents do not apply.)
44000000	Channel program has been intercepted because of permanent error associated with device end of previous request. You may reissue the intercepted request. (CSW contents do not apply.)
48000000	Request element for channel program has been made available after it has been purged. (CSW contents do not apply.)
4F000000	Error recovery routines have been entered because of direct access error but are unable to read home address of record 0. (CSW contents do not apply.)

Appendix F: UCB Sense Bytes

BYTE 0

DEVICE \ BIT	0	1	2	3	4	5	6	7
2400	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	OVER-RUN	WRT CNT ZERO	DATA CNVTT CHK
2311, 2841	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	OVER-RUN	TRK CCND CHK	SEEK CHK
2301, 2302, 2303, 2314, 2820	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	OVER-RUN		INVAL ADDR
2250	CMD REJ	SHOULD NOT OCCUR	BUS OUT	SHOULD NOT OCCUR	DATA CHK	SHOULD NOT OCCUR	BUFFER RUN-NING	SHOULD NOT OCCUR
2280	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	SHOULD NOT OCCUR	SHOULD NOT OCCUR	ILLGL SEG
2282	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	SHOULD NOT OCCUR	SHOULD NOT OCCUR	ILLGL SEGN
1052, 2150	CMD REJ	INT REQ	BUS OUT	EQ CHK				
1285	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	OVER-RUN	NON RCVY	KYBD CORR
1287	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	OVER-RUN	NON RCVY	KYBD CORR
1288	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	OVER-RUN	NON RCVY	SHOULD NOT OCCUR
2495	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	SHOULD NOT OCCUR	POSN CHK	SHOULD NOT OCCUR
2540, 2021	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK		UN-USUAL CMD	
1403, 1443	CMD REJ	INT REQ	BUS OUT	EQ CHK	TYPE BAR	TYPE BAR		CH 9
1442, 2501, 2520	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	OVER-RUN		
2671, 2822	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK			
2260	CMD REJ	INT REQ	BUS OUT	EQ CHK	SHOULD NOT OCCUR	SHOULD NOT OCCUR	SHOULD NOT OCCUR	SHOULD NOT OCCUR
2701, 2702	CMD RES	INT REQ	BUS OUT	EQ CHK	DATA CHK	OVER-RUN	LOST DATA	TIME OUT
1419/1275 PCU	CMD REJ	INT REQ	BUS OUT	NOT USED	DATA CHK	OVER-RUN	AUTO SELECT	NOT USED
1419/1275 SCU	CMD REJ	INT REQ	BUS OUT CHK	NOT USED	NOT USED	LATE STKR SELECT	AUTO SELECT	OP ATT

BYTE 1

DEVICE \ BIT	0	1	2	3	4	5	6	7
2400	NOISE	00-NON-XST TU 01-NOT READY 10-RDY & NO RWD 11-RDY & RWDNG		7 TRK	AT LOAD POINT	WRT STATUS	FILE PROTECT	TAPE IND
2311, 2841	DATA CHK FLD	TRK OVER-RUN	END OF CYL	IN-VALID SEQ	NO REC FOUND	FILE PROT	MISSING ADR MRKR	OVER-FLOW INL
2301, 2302, 2303, 2314, 2820	DATA CHK IN COUNT	TRK OVER-RUN	END OF CYL	INVAL SEQ	NO REC FOUND	FILE PROT	SERVICE OVER-RUN	OVER-FLOW INL
2250	LIGHT PEN DETECT	END ORDER SEQ	CHAR MODE					
2280	READ COUNT CHK	FILM LOW	RECRDR FORCED GAP	SHOULD NOT OCCUR	SHOULD NOT OCCUR	2840 OUTPUT CHK	2840 INPUT CHK	GRAPH-IC CHK
2282	READ COUNT CHK	FILM LOW	RECRDR FORCED GAP	FILM MOTION LIMIT	SHOULD NOT OCCUR	2840 OUTPUT CHK	2840 INPUT CHK	GRAPH-IC CHK
1052, 2150								
1285								
1287	TAPE MODE	LATE STKR SELECT	NO DOC FOUND	SHOULD NOT OCCUR	INVAL OP	SHOULD NOT OCCUR	SHOULD NOT OCCUR	SHOULD NOT OCCUR
1288	SHOULD NOT OCCUR	END OF PAGE	NO DOC FOUND	SHOULD NOT OCCUR	INVAL OP	SHOULD NOT OCCUR	SHOULD NOT OCCUR	SHOULD NOT OCCUR
2495								
2540, 2021								
1403, 1443								
1442, 2501, 2520								
2671, 2822								
2260								
2701, 2702								
1419/1275 PCU	NOT USED	NOT USED	DOC UNDER READ HEAD	AMT FIELD VALID	PROCESS CNTRL FIELD VALID	ACCT # FIELD VALID	TRANSIT FIELD VALID	SERIAL # FIELD VALID
1419/1275 SCU								

BYTE 2

DEVICE \ BIT	0	1	2	3	4	5	6	7
2400	BITS 0-7 INDICATE A TRACK IS IN ERROR						6 & 7 INDICATE NO ERROR OR MULTI-ERROR	
2311, 2841	UN-SAFE		SERIAL-IZER CHK	TAG LINE CHK	ALU CHK	UNSEL STATUS		
2301, 2302, 2303, 2314, 2820	UN-SAFE	SHIFT REG CHK	SKEW FAIL	CTR CHK	COMP CHK			
2250		BUFFER ADDRESS REGISTER: BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9
2280		BUFFER ADDRESS REGISTER: BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9
2282		BUFFER ADDRESS REGISTER: BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9

BYTE 3

DEVICE \ BIT	0	1	2	3	4	5	6	7
2400	R/W VRC	LRCR	SKEW	CRC	SKEW REQ VRC	0-1600 1-800	BKWD STATUS	COM-PARE
2311, 2841	READY	ON LINE	READ SAFETY	WRITE SAFETY		END OF CYL		SEEK INCMPL
2301, 2302, 2303, 2314, 2820	LRC BIT 0	LRC BIT 1	LRC BIT 2	LRC BIT 3				
2250	BUFFER ADDRESS REGISTER: BIT 8	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1
2280	BUFFER ADDRESS REGISTER: BIT 8	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1
2282	BUFFER ADDRESS REGISTER: BIT 8	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1

BYTE 4

DEVICE \ BIT	0	1	2	3	4	5	6	7
2400	ECHO ERR	RES TAPE UNIT	READ CLOCK ERR	WRITE CLOCK ERR	DELAY CNTR ERR	SEQ IND C	SEQ IND B	SEQ IND A
2311, 2841								
2301, 2302, 2303, 2314, 2820	SEQ IND 0	SEQ IND 1	SEQ IND 2	SEQ IND 3	SEQ IND 4	SEQ IND 5	SEQ IND 6	SEQ IND 7
2250								
2280								
2282								

BYTE 5

DEVICE \ BIT	0	1	2	3	4	5	6	7
2400	COMMAND IN PROGRESS WHEN OVERFLOW INCOMPLETE OCCURS OR ZERO							
2311, 2841	COMMAND IN PROGRESS WHEN OVERFLOW INCOMPLETE OCCURS WRITE = X'05' READ = X'06' OR ZERO							
2301, 2302, 2303, 2314, 2820								
2250								
2280								
2282								

Appendix G: Service Aids

In addition to the debugging facilities discussed in this manual, IBM provides the following service aid programs to aid you in debugging. A complete description of each of these service aids and instructions for their use are found in the publication IBM System/360 Operating System Service Aids, GC28-6719.

Program Name	Functional Description
IMDSADMP	A stand-alone program, assembled with user-selected options, that dumps the contents of main storage onto a tape or a printer. The program has two versions: • A high speed version that dumps the contents of main storage to a tape. • A low speed version that formats and dumps the contents of main storage either to a tape or directly to a printer.
IMDPRDMP	A problem program that reads, formats according to user supplied parameters, and prints the tape produced by execution of the stand-alone dump program assembled from the service aid IMDSADMP. The format of the printed output is similar to that produced by ABEND.
IMCJQDMP	A stand-alone program that reads, formats, and prints either the entire operating system data set SYS1.SYSJOBQE, or selects and prints information related to a specific job in that data set. Because it operates independently of the operating system, IMCJQDMP can print the contents of the job queue as it appeared at the time of abnormal termination.
IMBMDMAP	A problem that produces a map of the system nucleus, any load module, the resident reenterable load module area of an MFT system, or the link pack area of an MVT system. The listing produced by this program shows the locations of CSECTS, external references, and entry points within a load module.
IMASPZAP	A problem program that can inspect and modify either data records or load modules located on a direct access storage device.
IMAPTFLS	A problem program that identifies program temporary fixes (PTFs) and local fixes that have been applied to libraries.
IMAPTFLE	A problem program that produces the job control language (JCL) statements necessary to apply PTFs to a system; these JCL statements are tailored to the user's individual system.

Appendix H: TCAM Debugging Aids

In addition to the debugging facilities described in this publication, the telecommunications access method provides the following aids to debugging:

- I/O error recording procedures.
- I/O interrupt trace table (line trace).
- A dispatcher subtask trace table (STCB trace).
- Sequential listings of buffers and message queue data sets.

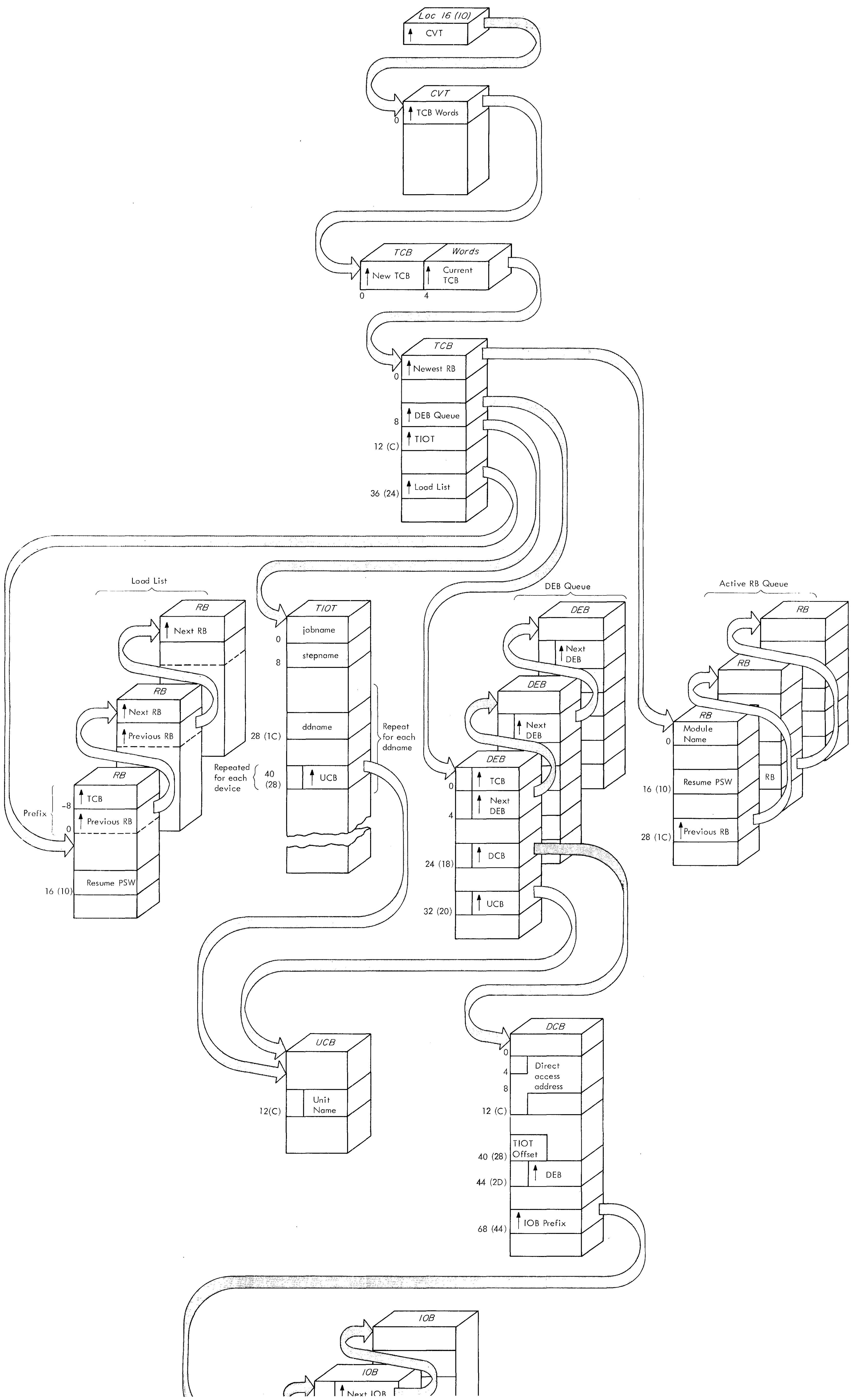
Optional formatted listing of the line and STCB traces are available with TCAM. These debugging aids are described in the publications IBM System/360 Operating System: TCAM Programmer's Guide and Reference Manual, GC30-2024, and IBM System/360 Operating System: TCAM Serviceability Aids Program Logic Manual, GY30-2027. A discussion of the TCAM formatted ABEND dump is given in the publication IBM System/360 Operating System: TCAM Program Logic Manual, GY30-2029.

TCB - Task Control Block (MVT)

+1	Address of most recent RB
+9	Address of most recent DEB
+13(D)	TIOT address
+16(10)	Completion code
+25(19)	Address of most recent SPQE
+33(21)	Bit 7 -- Non-dispatchability bit
+37(25)	Address of most recent LLE
+113(71)	Address of first save area
+125(7D)	Address of TCB for job step task
+129(81)	Address of TCB for next subtask attached by same parent task
+133(85)	Address of TCB for parent task
+137(89)	Address of TCB for most recent subtask
+145(91)	Address of ECB to be posted at task completion
+153(99)	Address of dummy PQE minus 8 bytes
+161(A1)	Address of STAE control block
+181(B5)	Address of the job step control block

UCB - Unit Control Block

-4	CPU ID (used only with Model 65 Multiprocessing systems)
+2	FF (UCB identification)
+4	Device address
+13(D)	Unit name
+18(12)	Device class
+19(13)	Device type
+22(16)	Sense bytes
+40(28)	Number of outstanding RESERVE requests (shared DASD only)



Loc 16 (10)
CVT
CVT
TCB Words
0
TCB
Words
New TCB
Current TCB
0
4
TCB
Newest RB
0
DEB Queue
8
TIOT
12 (C)
Load List
36 (24)
Load List
RB
Next RB
RB
Next RB
Previous RB
RB
TCB
-8
Prefix
Previous RB
0
Resume PSW
16 (10)
TIOT
jobname
0
stepname
8
ddname
28 (1C)
Repeat for each ddname
Repeated for each device
40 (28)
UCB
DEB Queue
DEB
Next DEB
DEB
Next DEB
DEB
TCB
0
Next DEB
4
DCB
24 (18)
UCB
32 (20)
Active RB Queue
RB
RB
RB
Module Name
0
Resume PSW
16 (10)
RB
Previous RB
28 (1C)
UCB
Unit Name
12(C)
DCB
0
4
Direct access address
8
12 (C)
TIOT Offset
40 (28)
DEB
44 (2D)
IOB Prefix
68 (44)
IOB
IOB
Next IOB

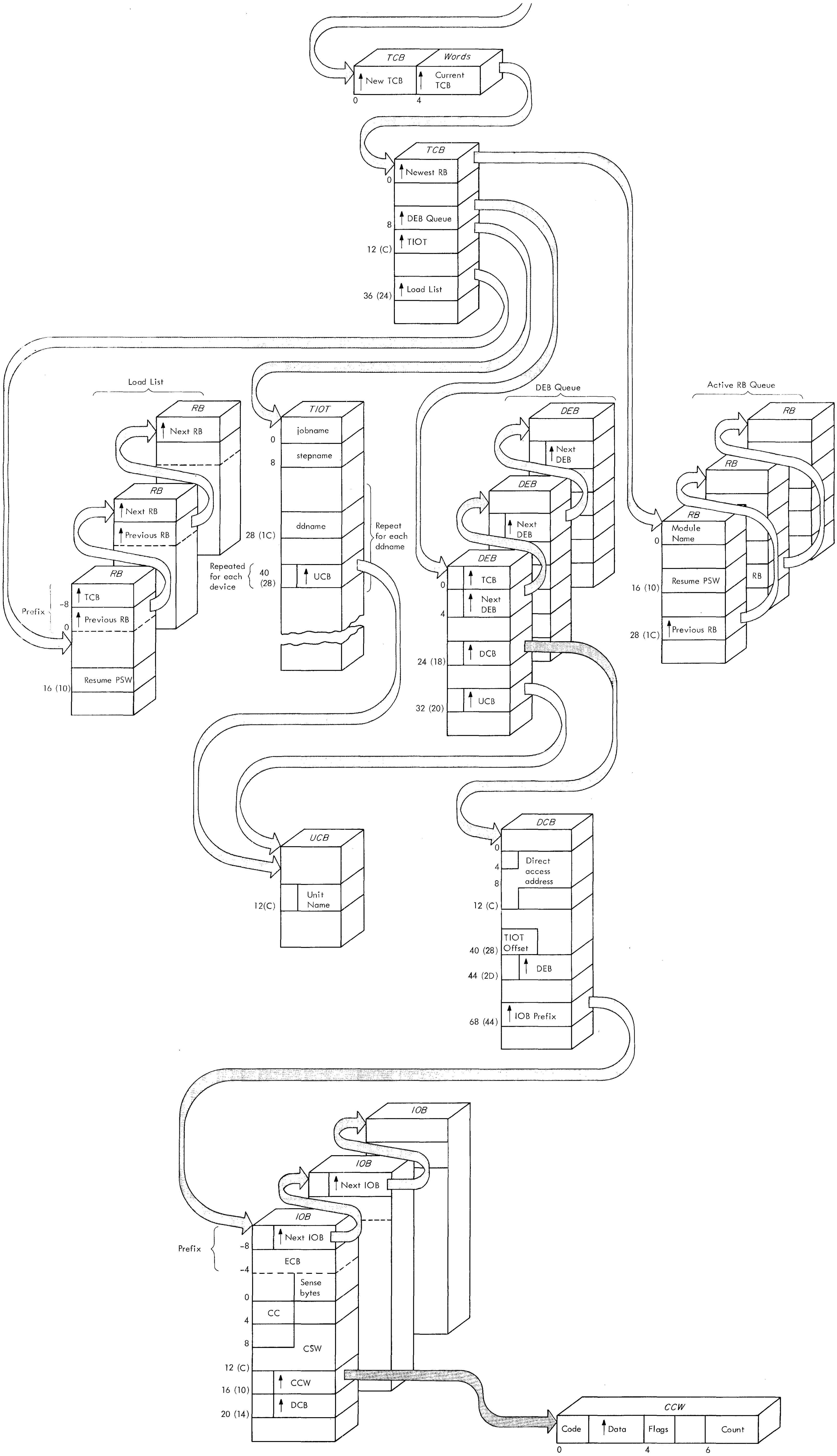


Figure 36. Control Block Flow

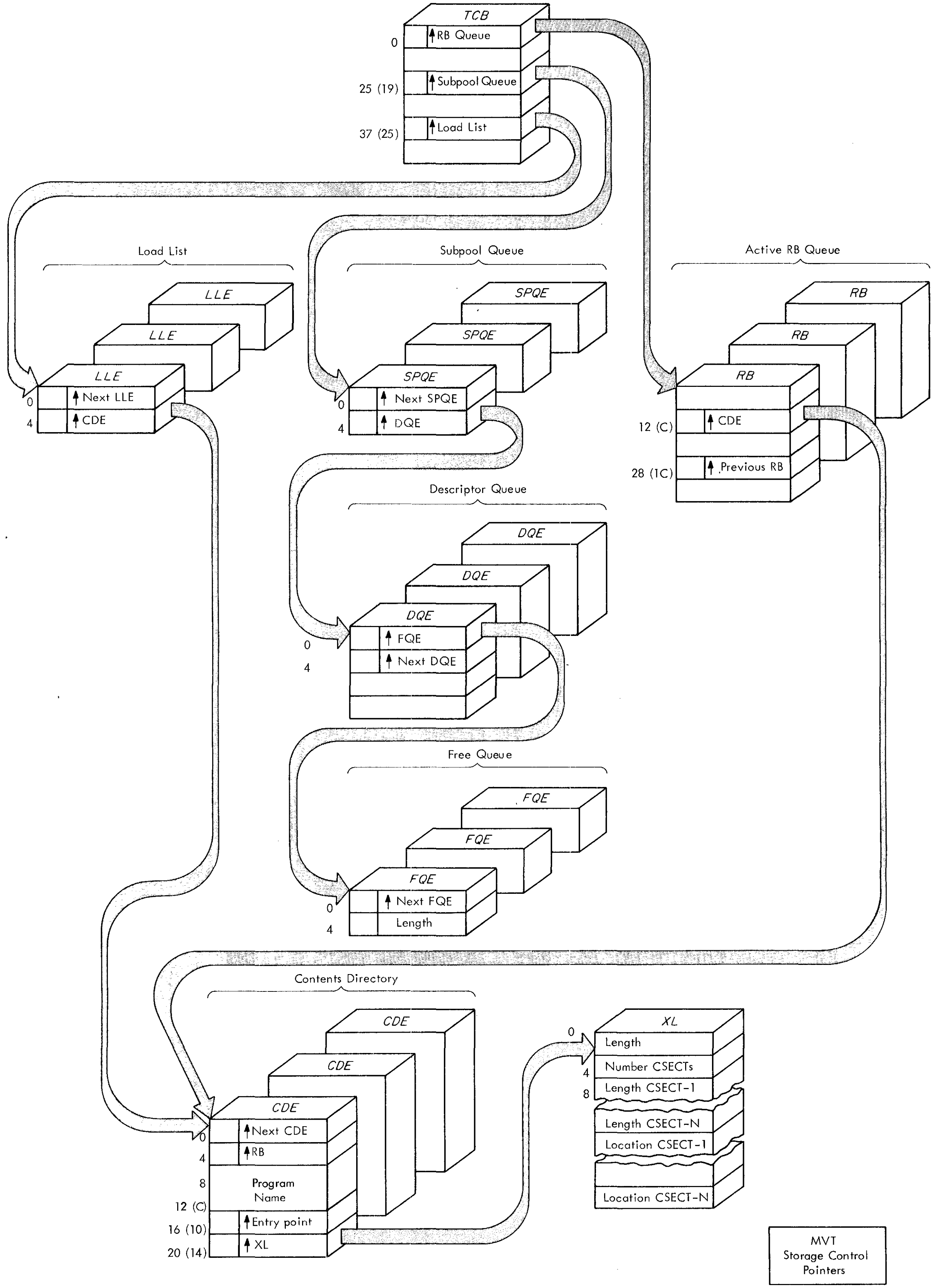


Figure 37. MVT Storage Control Flow

Index

Indexes to systems reference library manuals are consolidated in the publication IBM System/360 Operating System: Systems Reference Library Master Index, GC28-6644. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

When more than one page reference is given, the major reference is first.

GC28-6670-4

System/360 OS Debugging Guide (S360-20) Printed in U.S.A. GC28-6670-4

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENT FORM

IBM System/360 Operating System
Programmer's Guide to Debugging

Order No. GC28-6670-4

Please use this form to express your opinion of this publication. We are interested in your comments about its technical accuracy, organization, and completeness. All suggestions and comments become the property of IBM.

Please do not use this form to request technical information or additional copies of publications. All such requests should be directed to your IBM representative or to the IBM Branch Office serving your locality.

- Please indicate your occupation: ______________________________

- How did you use this publication?
  - ☐ Frequently for reference in my work.
  - ☐ As an introduction to the subject.
  - ☐ As a textbook in a course.
  - ☐ For specific information on one or two subjects.

- Comments (Please include page numbers and give examples.):

- Thank you for your comments. No postage necessary if mailed in the U.S.A.

YOUR COMMENTS, PLEASE . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Note: Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Cut Along Line

Fold Fold

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY ...

IBM Corporation
P.O. Box 390
Poughkeepsie, N.Y. 12602

Attention: Programming Systems Publications
Department D58

Fold Fold

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENT FORM

IBM System/360 Operating System
Programmer's Guide to Debugging

Order No. GC28-6670-4

Please use this form to express your opinion of this publication. We are interested in your comments about its technical accuracy, organization, and completeness. All suggestions and comments become the property of IBM.

Please do not use this form to request technical information or additional copies of publications. All such requests should be directed to your IBM representative or to the IBM Branch Office serving your locality.

- Please indicate your occupation: ______________________________

- How did you use this publication?
  - ☐ Frequently for reference in my work.
  - ☐ As an introduction to the subject.
  - ☐ As a textbook in a course.
  - ☐ For specific information on one or two subjects.

- Comments (Please include page numbers and give examples.):

- Thank you for your comments. No postage necessary if mailed in the U.S.A.

YOUR COMMENTS, PLEASE . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Note: Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Cut Along Line

Fold Fold

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY ...

IBM Corporation
P.O. Box 390
Poughkeepsie, N.Y. 12602

**Attention: Programming Systems Publications
Department D58**

Fold Fold

IBM®

**International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
(USA Only)**

**IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)**

System/360 OS Debugging Guide (S360-20) Printed in U.S.A. GC28-6670-4

READER'S COMMENT FORM

IBM System/360 Operating System
Programmer's Guide to Debugging

Order No. GC28-6670-4

Please use this form to express your opinion of this publication. We are interested in your comments about its technical accuracy, organization, and completeness. All suggestions and comments become the property of IBM.

Please do not use this form to request technical information or additional copies of publications. All such requests should be directed to your IBM representative or to the IBM Branch Office serving your locality.

- Please indicate your occupation: ______________________________

- How did you use this publication?
  - ☐ Frequently for reference in my work.
  - ☐ As an introduction to the subject.
  - ☐ As a textbook in a course.
  - ☐ For specific information on one or two subjects.

- Comments (Please include page numbers and give examples.):

- Thank you for your comments. No postage necessary if mailed in the U.S.A.

GC28-6670-4

YOUR COMMENTS, PLEASE . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Note: Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Cut Along Line

Fold **Fold**

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
P.O. Box 390
Poughkeepsie, N.Y. 12602

Attention: Programming Systems Publications
Department D58

Fold **Fold**

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENT FORM

IBM System/360 Operating System
Programmer's Guide to Debugging

Order No. GC28-6670-4

Please use this form to express your opinion of this publication. We are interested in your comments about its technical accuracy, organization, and completeness. All suggestions and comments become the property of IBM.

Please do not use this form to request technical information or additional copies of publications. All such requests should be directed to your IBM representative or to the IBM Branch Office serving your locality.

- Please indicate your occupation: ______________________________

- How did you use this publication?
  - ☐ Frequently for reference in my work.
  - ☐ As an introduction to the subject.
  - ☐ As a textbook in a course.
  - ☐ For specific information on one or two subjects.

- Comments (Please include page numbers and give examples.):

- Thank you for your comments. No postage necessary if mailed in the U.S.A.

YOUR COMMENTS, PLEASE . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Note: Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Cut Along Line

Fold Fold

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY ...

IBM Corporation
P.O. Box 390
Poughkeepsie, N.Y. 12602

Attention: Programming Systems Publications
Department D58

Fold Fold

System/360 OS Debugging Guide (S360-20) Printed in U.S.A. GC28-6670-4

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

Storage Control in Systems with MFT (Without Subtasking)

Storage control information in systems with MFT without subtasking is similar to that in systems with PCP, except that one MSS boundary box is maintained for each partition. The TCB associated with the partition contains a pointer (TCBMSS) to the boundary box.

If Main Storage Hierarchy Support is included, the first half of each expanded boundary box describes the processor storage (hierarchy 0) partition segment, and the second half describes the 2361 Core Storage (hierarchy 1) partition segment. Any partition segment not currently assigned storage in the system has the applicable boundary box pointers set to zero. If the partition is established entirely within hierarchy 0, or if 2361 Core Storage is not included in the system, the hierarchy 1 pointers in the second half of the expanded boundary box are set to zero. If a partition is established entirely within hierarchy 1, the hierarchy 0 pointers in the first half of the expanded boundary box are set to zero.

The boundary box format for MFT is identical to the format for PCP. The pointers, however, point to the boundaries of the partition and to the partition FQEs rather than to the boundaries of storage. Figure 11 summarizes storage control in systems with MFT.

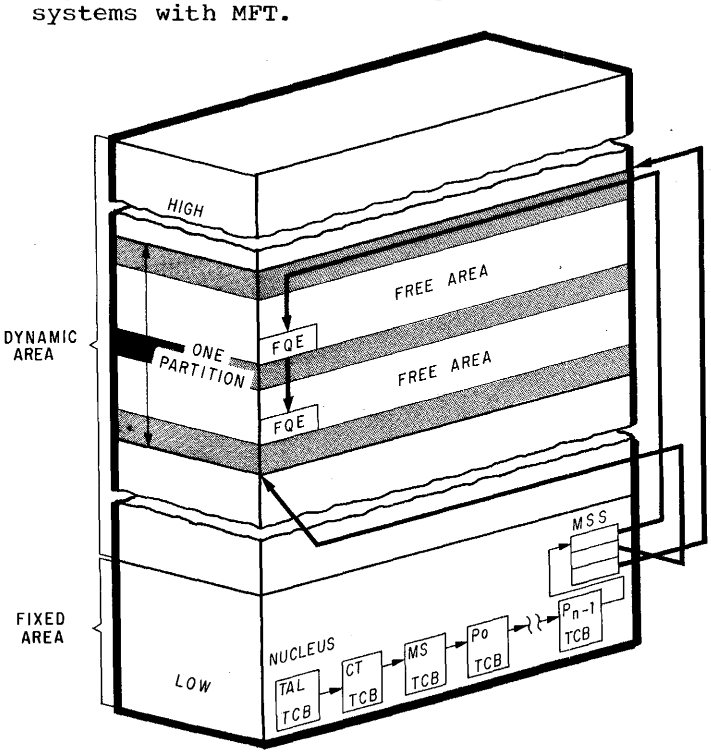


Figure 11. Storage Control for a Partition (MFT Without Subtasking)

Storage Control in Systems with MFT (With Subtasking)

Storage control information for the job step or partition TCB in MFT systems with subtasking is handled in the same way as in MFT systems without subtasking. However, when subtasks are created, the supervisor builds another control block, the <u>gotten subtask area queue element</u> (GQE). The GQEs associated with each subtask originate from a one word pointer addressed by the TCBMSS field of the subtask TCB.

<u>GQE</u>: Each area in main storage belonging to a subtask, and obtained by a supervisor issued GETMAIN macro instruction, is described by a <u>gotten subtask area queue element</u> (GQE). GQEs are chained in the order they are created. The TCBMSS field of the subtask TCB contains the address of a word which points to the most recently created GQE.

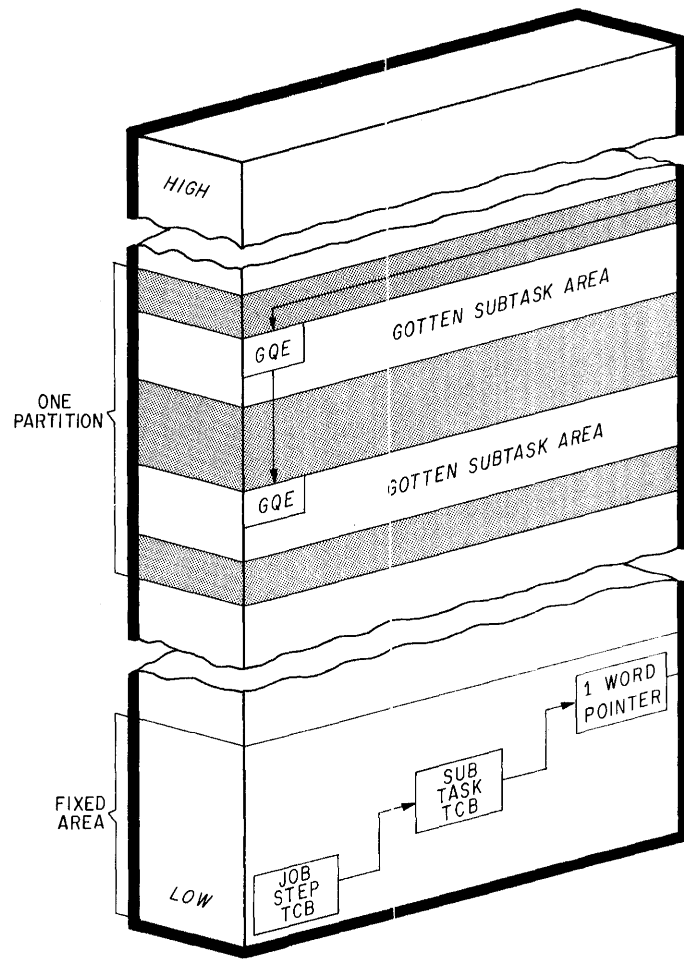


Figure 12. Storage Control for Subtask Storage (MFT with Subtasking)

If Main Storage Hierarchy Support is present in the system, the GQE chain can span from hierarchy 0 to hierarchy 1 and back in any order. Each GQE occupies the first eight bytes of the area it describes, and has the following format:

↑ Previous GQE	Number of Bytes
0	4

Bytes 0-3: Pointer to the Previous GQE or, if zero, this is the last GQE on the chain.

Bytes 4-7: Number of bytes in the gotten subtask area.

Figure 12 summarizes the chaining of GQEs to a subtask TCB.

Storage Control for a Region in Systems with MVT

Unassigned areas of main storage within each region of a system with MVT are reflected in a queue of partition queue elements (PQEs) and a series of free block queue elements (FBQEs).

PQE: The partition queue associated with a region resides in the system queue space. It is connected to the TCBs for all tasks in the job step through a dummy PQE located in the system queue space. A dummy PQE has the following format:

↑ First PQE	↑ Last PQE
0	4

Bytes 0-3: Pointer to the first PQE in the partition queue.

Bytes 4-7: Pointer to the last PQE in the partition queue.

In systems that do not include the rollout/rollin feature or Main Storage Hierarchy Support for IBM 2361 Models 1 and 2, there is one PQE for each job step. If the rollout feature is used, additional PQEs are added each time a job step borrows storage space from existing steps or acquires unassigned free space to satisfy an unconditional GETMAIN request. These additional PQEs are removed from the queue as the rollin feature is used. If Main Storage Hierarchy Support is present, one PQE exists for each hierarchy used by the job step. A PQE has the following format:

Res	↑ First FBQE	Res	↑ Last FBQE
0	1	4	5
Res	↑ Next PQE	Res	↑ Previous PQE
8	9	12 (C)	13 (D)
Res	↑ Owning TCB	Res	Region Size
16 (10)	17 (11)	20 (14)	21 (15)
Res	↑ Region	Flags	Res
24 (18)	25 (19)	28 (1C)	29 (1D)

Bytes 1-3: Pointer to the first FBQE or, if there are no FBQEs, a pointer to the PQE itself.

Bytes 5-7: Pointer to the last FBQE or, if there are no FBQEs, a pointer to the PQE itself.

Bytes 9-11(B): Pointer to the next PQE or, if this is the last PQE, zeros.

Bytes 13-15(D-F): Pointer to the previous PQE or, if this is the first PQE, zeros.

Bytes 17-19(11-13): Pointer to the TCB of the owning job step.

Bytes 21-23(15-17): Size of the region, in 2K (2048) bytes.

Bytes 25-27(19-1B): Pointer to the first byte of the region.

Byte 28(1C): Rollout flags.

FBQE: The FBQEs chained to a PQE reflect the total amount of free space in a region. Each FBQE is associated with one or more contiguous 2K blocks of free storage area. FBQEs reside in the lowest part of their associated area. As area distribution within the region changes, FBQEs are added to and deleted from the free block queue.

An FBQE has the following format:

Res	↑ Next FBQE	Res	↑ Previous FBQE
0	1	4	5

Res	Number of bytes
8	9

Bytes 1-3: Pointer to the next lower FBQE or, if this is the last FBQE, a pointer to the PQE.

Bytes 5-7: Pointer to the preceding FBQE, or, if this is the first FBQE, a pointer to the PQE.

Bytes 9-12(C): Number of bytes in the free block.

The remaining main storage in a region is used by problem programs and system programs. For convenience in referring to storage areas, the total amount of space assigned to a task represents one or more numbered subpools. (Subpools can also be shared by tasks.) Subpools are designated by a number assigned to the area through a GETMAIN macro instruction. Subpool numbers available for problem program use range from 0 through 127. Subpool numbers 128 through 255 are either unavailable or used by system programs.

Storage control elements and queues for a region are summarized in Figure 13.

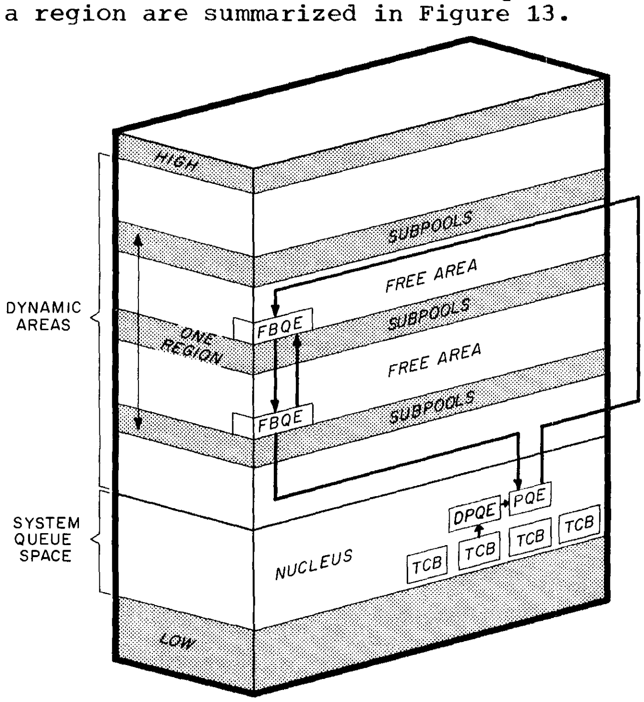


Figure 13. Storage Control for a Region (MVT)

Storage Control for a Subpool in Systems with MVT

Main storage distribution within each subpool is reflected in a subpool queue element (SPQE) and queues of descriptor queue elements (DQEs) and free queue elements (FQEs).

SPQE: SPQEs are associated with the subpools created for a task. SPQEs reside in the system queue space and are chained to the TCB(s) that use the subpool. They serve as a link between the TCB and the descriptor queue, and may be part of a subpool queue if the task uses more than one subpool. If a subpool is used by more than one task, only one SPQE is created. An SPQE has the following format:

Flags	↑ Next SPQE	SP #	↑ First DQE
0	1	4	5

Byte 0:
- Bit 0 - Subpool is owned by this task if zero; shared, and owned by another task, if one.
- Bit 1 - This SPQE is the last on the queue, if one.
- Bit 2 - Subpool is shared and owned by this task, if one.
- Bits 3-7 - Reserved.

Bytes 1-3: Pointer to next SPQE or, in last SPQE, zero.

Byte 4: Subpool number.

Bytes 5-7: Pointer to first DQE or, if the subpool is shared, a pointer to the "owning" SPQE.

DQE: DQEs associated with each SPQE reflect the total amount of space assigned to a subpool. Each DQE is associated with one or more 2K blocks of main storage set aside as a result of a GETMAIN macro instruction. Each DQE is also the starting point for the free queue. A DQE has the following format:

Res	↑ First FQE	Res	↑ Next DQE
0	1	4	5

Res	↑ First 2K block	Res	Length of area
8	9	12(C)	13(D)

Bytes 1-3: Pointer to the FQE associated with the first free area.

Bytes 5-7: Pointer to the next DQE or, if this is the last DQE, zeros.

Bytes 9-11(B): Pointer to first 2K block described by this DQE.

Bytes 13-15(D-F): Length in bytes of area described by this DQE.

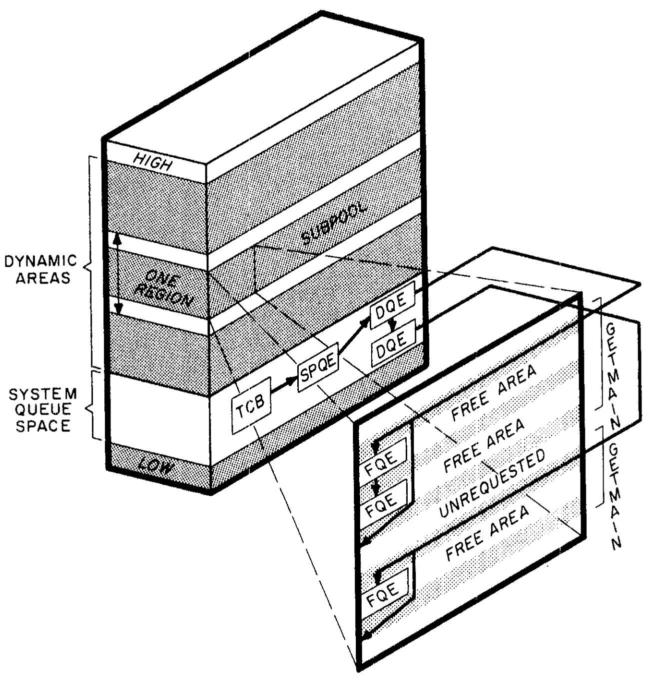


Figure 14. Storage Control for a Subpool (MVT)

FQE: The FQE describes a free area within a set of 2K blocks described by a DQE. It occupies the first eight bytes of that free area. Since the FQE is within the subpool, it has the same protect key as the task active within that subpool. Extreme care should be exercised to see that FQEs are not destroyed by the problem program. If an FQE is destroyed, the free space that it describes is lost to the system and cannot be assigned through a GETMAIN. As area distribution within the set of blocks changes, FQEs are added to and deleted from the free queue. An FQE has the following format:

Res	↑ Next FQE	Res	Number of bytes
0	1	4	5

Bytes 1-3: Pointer to the next lower FQE or, if this is the last FQE, zeros.

Bytes 5-7: Number of bytes in the free area.

A subpool is summarized in Figure 14.

Storage Control for a Load Module in Systems with MVT

Each load module in main storage is described by a contents directory entry (CDE) and an extent list (XL) that tells how much space it occupies.

CDE: The contents directory is a group of queues, each of which is associated with an area of main storage. The CDEs in each queue represent the load modules residing in the associated area. There is a CDE queue for the link pack area and one for each region, or job pack area. The TCB for the job step task that requested the region points to the first CDE for that region. Contents directory queues reside in the system queue space. A CDE has the following format:

Flags	↑ Next CDE	Res	↑ RB
0	1	4	5

EBCDIC program name
8

Count	↑ Entry point	Flags	↑ XL
16(10)	17(11)	20(14)	21(15)

Byte 0: Flag bits, when set to one, indicate:
- Bit 0 - Module was loaded by NIP.
- Bit 1 - Module is in process of being loaded.
- Bit 2 - Module is reenterable.
- Bit 3 - Module is serially reusable.
- Bit 4 - Module may not be reused.
- Bit 5 - This CDE reflects an alias name (a minor CDE).
- Bit 6 - Module is in job pack area.
- Bit 7 - Module is not only-loadable.

Bytes 1-3: Pointer to next CDE.

Bytes 5-7: Pointer to the RB.

Bytes 8-15(F): EBCDIC name of load module.

Byte 16(10): Use count.

abnormally terminated, to print an ABEND or SNAP dump stored in an earlier step, or to release a tape volume or direct access space acquired for dump data sets. Conditional execution of the last step can be established through proper use of the COND parameter and its subparameters, EVEN and ONLY, on the EXEC statement.

Direct access space should be requested in units of average block size rather than in cylinders (CYL) or tracks (TRK). If abnormal termination occurs and the data set is retained, the tape volume or direct access space should be released (DELETE in the DISP parameter) at the time the data set is printed.

```
 * ABDUMP REQUESTED *


JOB ATHEOT24         STEP STEP            TIME 000737    DATE 99366                                                PAGE 0001

COMPLETION CODE      USER = 0123

INTERRUPT AT C6EF5A

PSW AT ENTRY TO ABEND  0015000D 4006EF5A

TCB      01CB20  RB   0007FC58   PIE      00000000   DEB  0007F78C   TIOT 0007FD80   CMP   8000007B   TRN  00000000
                 MSS  0001CC58   PK/FLG 10B10408     FLG  000001F8   LLS  00000000   JLB   0007FF78   JST  000055D8
                 FSA  1506EBF8   TCB      0001D0A0   TME  0001CBD8   PIB  E0012420   NTC   00000000   OTC  0001CDE0
                 LTC  00000000   IQE      00000000   ECB  0006EE1C   XTCB 00000000   LP/FL FB050000   RESV 00000000
                 STAE 00000000   TCT      00000000   USER 00000000   DAR  00000000   RESV  00000000   JSCB 00000000


ACTIVE RBS

PRB      06EE28  NM TATHB10G    SZ/STAB 003D20D0    USE/EP 0106EE48    PSW 0015000D 4006EF5A    Q 000000    WT/LNK 0001CB20

SVRB     07FD20  NM SVC-601C    SZ/STAB 0012D062    USE/EP 00007B78    PSW FF040033 50007D20    Q 900390    WT/LNK 0006EE28
                 RG 0-7     000002A0    8000007B    00000000    00080000    0007FE48    00000098    000055D8    0007FC30
                 8-15-7     0006EE60    0007FF78    0007FFB0    0007FFF8    4006EE4E    0006EE60    00009848    00000000

SVRB     07FC58  NM SVC-A05A    SZ/STAB 000CD062    USE/EP 00007B78    PSW FF04000E 8001E7EC    Q F803F8    WT/LNK 0007FD20
                 RG 0-7     0007F7E8    0007FD80    40007B7A    000097F8    0001CB20    0007FD20    0006F230    000055D8
                 8-15-7     0007F7E8    0006F296    0001CC56    0000225C    0001CB20    0006F230    90007CBC    0001F7C8

JOB PACK AREA QUEUE

LPRB 06ECA8  NM TATHA10G    SZ/STAB 002F20D0    USE/EP 0106ECC8    PSW FF15000E 8006ED9C    Q 000000    WT/LNK 0101CDE0

LPRB 06EE28  NM TATHB10G    SZ/STAB 003D20D0    USE/EP 0106EE48    PSW 0015000D 4006EF5A    Q 000000    WT/LNK 0001CB20

LPRB 06F018  NM TATHC10G    SZ/STAB 00122090    USE/EP 0106F038    PSW 0004000D 40006AE4    Q 000000    WT/LNK 0001CC80

LPRB 06F080  NM TATHD10G    SZ/STAB 001820D0    USE/EP 0106F0D0    PSW FF150001 4006F16C    Q 000000    WT/LNK 0101D0A0

LPRB 06F190  NM TATHE10G    SZ/STAB 001320D0    USE/EP 0106F180    PSW FF150001 4006F21E    Q 000000    WT/LNK 0101CF40

P/P STORAGE BOUNDARIES 0006E800 TO 00080000

FREE AREAS     SIZE

  06EB90     00000060
  06EC50     00000050
  06F5B8     0000FC58
  07F668     C0000098
  07F7D8     00000010
  07F840     C0000228
  07FB90     000000C0
  C7FEE8     C000C018

GOTTEN CORE    SIZE

  07F210     00000380
  06F310     C00002A8
  07FC50     C0000068
  06F228     000000E8
  07F590     00000008
  07F5F0     C0000008
  07FD18     00000098
  07F700     00000060
  07F760     00000078
  07FA68     C0000060
  07FAC8     00000078
```

Figure 22A. Sample of an ABEND Dump (PCP, MFT)

```
SAVE AREA TRACE                                                                                                PAGE 0002

TATHB10G WAS ENTERED

SA    06EBF8  WD1 0606EAC8   HSA 00000100   LSA 0006EE60   RET 00009848   EPA 4006EE48   R0  000098CE
              R1  0001CC80   R2  00000000   R3  00080000   R4  0007FE48   R5  00000098   R6  000055D8
              R7  0007FC30   R8  0006ECE0   R9  0007FF78   R10 0007FFB0   R11 0007FFF8   R12 4006ECCE

SA    06EE60  WD1 00000000   HSA 0006EBF8   LSA 00000000   RET 00000000   EPA 00000000   R0  00000000
              R1  00000000   R2  00000000   R3  00000000   R4  00000000   R5  00000000   R6  00000000
              R7  00000000   R8  00000000   R9  00000000   R10 00000000   R11 00000000   R12 00000000

PROCEEDING BACK VIA REG 13

SA    06EE60  WD1 00000000   HSA 0006EBF8   LSA 00000000   RET 00000000   EPA 00000000   R0  00000000
              R1  00000000   R2  00000000   R3  00000000   R4  00000000   R5  00000000   R6  00000000
              R7  00000000   R8  00000000   R9  00000000   R10 00000000   R11 00000000   R12 00000000

TATHB10G WAS ENTERED

SA    06EBF8  WD1 0606EAC8   HSA 00000100   LSA 0006EE60   RET 00009848   EPA 4006EE48   R0  000098CE
              R1  0001CC80   R2  00000000   R3  00080000   R4  0007FE48   R5  00000098   R6  000055D8
              R7  0007FC30   R8  0006ECE0   R9  0007FF78   R10 0007FFB0   R11 0007FFF8   R12 4006ECCE

DATA SETS

SNAP2        UCB   192    00225C       DEB 07F78C       DCB 06EFB4

DUMDCB       UCB   192    00225C       DEB 07FAF4       DCB 06EF5C

JOBLIB       UCB   190    00218C

SYSPRINT     UCB   192    00225C

SYSABEND     UCB   192    00225C

SNAP1        UCB   190    00218C


REGS AT ENTRY TO ABEND

FL.PT.REGS 0-6        00.000000 00000000       00.000000 00000000       00.000000 00000000       00.000000 00000000

REGS 0-7            000002A0   8000007B   00000000   00080000          0007FE48   00000098   000055D8   0007FC30
REGS 8-15           0006EE60   0007FF78   0007FFB0   0007FFF8          4006EE4E   0006EE60   00009848   00000000


NUCLEUS

000000   000C0000 0000051C F0F0F5C1 00000000    000097F8 00013440 01040080 8003ACD4   *........005A.......8... .......M*
000020   0004000A 50006B46 00000000 00000000    0000FF00 00000000 FF04000E A0007E2A   *................................*
000040   1007F5E8 50000000 00001480 000097F8    60C85DC0 00000000 00040000 00000282   *..5Y...........8.H..............*
000060   00040000 0000033A 00040000 000002DE    00000000 0000B278 00040000 00000226   *................................*
000080   000153B0 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *................................*
0000A0   0000C000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *................................*
   LINES    0000C0-000140    SAME AS ABOVE
000160   00000000 00000000 00000000 82000170    00040000 0003A7A0 00000000 00000000   *................................*
000180   0001C820 00007E91 0006F465 80007D16    00000080 0006F491 00000001 0006F4A8   *.........4...........4.......4.*
0001A0   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *................................*
   LINE     0001C0     SAME AS ABOVE
0001E0   000079FC 000068B8 0000A43A 00000001    40007720 0000AD42 90001520 00000000   *...0........... ................*
000200   0000846C 000083E4 00006780 00006942    00001000 00000F28 00009730 0001335C   *......U.........................*
000220   00013340 00234700 024C96F0 02279029    01805830 06C45840 30004700 025CD207   *... .......0.........D. ......K.*
000240   40100038 94FD4011 90A13030 5890021C    05B95850 02105890 021407F9 90A101E0   * ..... ..................9......*
000260   02070440 003847F0 024C940F 02279829    018091F0 023B4780 044898A1 01E08200   *K.. ...0...........0............*
000280   04409C29 018091F0 023B4780 029C90A1    01E0D207 04400018 47F002B2 589006C4   *. .....0..........K.. ...0.....D*
0002A0   90A1903C 58990000 D2079010 001894FD    90119140 001B4780 02C05820 02D40522   *.......K.........  ........M..*
0002C0   9180001B 478002CE 582002D8 052247F0    026A0000 000153B8 000087DA 0A0390A9   *..........Q...0.................*
0002E0   01A098CD 002858B0 02189101 0029078B    58A006C4 58A0A004 12AA07CB 18BA58AA   *.................D...........*
000300   000012AA 47C00332 90C2B004 181B58B0    02189280 100098F0 A0008900 C0001200   *.........B............0.........*
000320   07B850F0 002C41E0 020C98AD 01A08200    0028181B 58B00218 07FB900F 04005890   *...0............................*
```

Figure 22B. Sample of an ABEND Dump (PCP, MFT)

Sample DD Statements: Figure 23 shows a set of job steps that include DD statements for ABEND dump data sets.

The SYSABEND DD statement in STEP2 takes advantage of the direct access space acquired in STEP1 by indicating MOD in the DISP parameter. Note that the space request in STEP1 is large so that the dumping operation is not inhibited due to insufficient space. The final SYSABEND DD statement in the job should indicate a disposition of DELETE to free the space acquired for dumping.

Contents of an ABEND/SNAP Dump (PCP,MFT)

This explanation of the contents of ABEND/SNAP dumps for systems with PCP and MFT is interspersed with sample sections taken from an ABEND dump. Capital letters represent the headings found in all dumps, and lowercase letters, information that varies with each dump. The lowercase letter used indicates the mode of the information, and the number of letters indicates its length:

- h represents 1/2 byte of hexadecimal information
- d represents 1 byte of decimal information
- c represents a 1-byte character

You may prefer to follow the explanation on your own ABEND or SNAP dump.

```
//STEP1    EXEC    PGM=PROGRAM1
//SYSABEND DD      DSNAME=DUMP,UNIT=2311,DISP=(,KEEP,KEEP),            X
//                 VOLUME=SER=1234,SPACE=(TRK,(110,10))

        other DD statements

//STEP2    EXEC    PGM=PROGRAM2
//SYSABEND DD      DSNAME=*.STEP1.SYSABEND,DISP=(MOD,DELETE,KEEP),     X
//                 VOL=REF=*.STEP1.SYSABEND
```

Figure 23. SYSABEND DD Statements

```
* * * A B D U M P   R E Q U E S T E D * * *

 *ccccccc...

JOB cccccccc          STEP cccccccc         TIME dddddd   DATE ddddd                                      PAGE dddd

COMPLETION CODE       SYSTEM = hhh (or USER = dddd)

cccccc...

INTERRUPT AT hhhhhh

PSW AT ENTRY TO ABEND (SNAP) hhhhhhhh hhhhhhhh
```

* * * A B D U M P   R E Q U E S T E D * * *
identifies the dump as an ABEND or SNAP dump.

*ccccccc.....
is omitted or is one or more of the following:

*CORE NOT AVAILABLE, LOC. hhhhhhhhhhhh TAKEN...
indicates that the ABDUMP routine confiscated storage locations hhhhhh through hhhhhh because not enough storage was available. This area is printed under P/P STORAGE, but can be ignored because the problem program originally in it was overlaid during the dumping process.

*MODIFIED, /SIRB/DEB/LLS/ARB/MSS...
indicates that the one or more queues listed were destroyed or their elements dequeued during abnormal termination:

- SIRB -- system interruption request block queue. One or more SIRB elements were found in the active RB queue: these elements are always dequeued during dumping.
- DEB -- DEB queue. If the first message also appeared, either a DEB or an associated DCB was overlaid.
- LLS -- load list. If the first message also appeared, one or more loaded RBs were overlaid.
- ARB -- active RB queue. If the first message also appeared, one or more RBs were overlaid.
- MSS -- boundary box queue. One or more MSS elements were dequeued, but an otherwise valid control block was found in the free area specified by an MSS element.

*FOUND ERROR IN /DEB/LLS/ARB/MSS...
indicates that one or more of the following contained an error:

- DEB: data extent block
- LLS: load list
- ARB: active RB
- MSS: boundary box

This message appears with either the first or second message above. The error could be: improper boundary alignment, control block not within storage assigned to the program being dumped, or an infinite loop (300 times is the maximum for this test). For an MSS block, 4 other errors could also be found: incorrect descending sequence (omitting loop count), overlapping free areas, free area not entirely within the storage assigned to the program being dumped, or count in count field not a multiple of 8.

JOB cccccccc
is the job name specified in the JOB statement.

STEP cccccccc
is the step name specified in the EXEC statement for the problem program being dumped.

TIME dddddd
is the hour (first 2 digits), minute (second 2 digits), and second (last 2 digits) when the ABDUMP routine began processing.

DATE ddddd
is the year (first 2 digits) and day of the year (last 3 digits). For example, 67352 would be December 18, 1967.

PAGE dddd
is the page number. Appears at the top of each page.

COMPLETION CODE SYSTEM=hhh or COMPLETION CODE USER=dddd
is the completion code supplied by the control program (SYSTEM=hhh) or the problem program (USER=dddd). Either SYSTEM=hhh or USER=dddd is printed, but not both. Common completion codes are explained in Appendix B.

cccccc...
explains the completion code or, if a program interruption occurred: PROGRAM INTERRUPTION ccccc... AT LOCATION hhhhhh,

where ccccc is the program interruption cause -- OPERATION, PRIVILEGED OPERATION, EXECUTE, PROTECTION, ADDRESSING, SPECIFICATION, DATE, FIXED-POINT OVERFLOW, FIXED-POINT DIVIDE, DECIMAL OVERFLOW, DECIMAL DIVIDE, EXPONENT OVERFLOW,EXPONENT UNDERFLOW, SIGNIFICANCE, or FLOATING-POINT DIVIDE; and hhhhhh is the starting address of the instruction being executed when the interruption occurred.

INTERRUPT AT hhhhhh
is the address of next instruction to be executed in the problem program. It is obtained from the resume PSW of the PRB or LPRB in the active RB queue at the time abnormal termination was requested.

PSW AT ENTRY TO ABEND hhhhhhhh hhhhhhhh or PSW AT ENTRY TO SNAP hhhhhhhh hhhhhhhh
is the PSW for the problem or control program that had control when abnormal termination was requested or when the SNAP macro instruction was executed.

```
TCB    hhhhhh RB      hhhhhhhh   PIE     hhhhhhhh   DEB  hhhhhhhh   TIOT hhhhhhhh   CMP    hhhhhhhh   TRN  hhhhhhhh
              MSS     hhhhhhhh   PK/FLG  hhhhhhhh   FLG  hhhhhhhh   LLS  hhhhhhhh   JLB    hhhhhhhh   JST  hhhhhhhh
              RG 0-7    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh
              RG 8-15   hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh
              FSA     hhhhhhhh   TCB     hhhhhhhh   TME  hhhhhhhh   PIB  hhhhhhhh   NTC    hhhhhhhh   OTC  hhhhhhhh
              LTC     hhhhhhhh   IQE     hhhhhhhh   ECB  hhhhhhhh   XTCB hhhhhhhh   LP/FL  hhhhhhhh   RESV hhhhhhhh
              STAE    hhhhhhhh   TCT     hhhhhhhh   USER hhhhhhhh   DAR  hhhhhhhh   RESV   hhhhhhhh   JSCB hhhhhhhh
```

TCB hhhhhh
is the starting address of the TCB.

RB hhhhhhhh
is the TCBRBP field (bytes 0 through 3): starting address of the active RB queue and, consequently, the most recent RB on the queue (usually ABEND's RB).

PIE hhhhhhhh
is the TCBPIE field (bytes 4 through 7): starting address of the program interruption element (PIE) for the task.

DEB hhhhhhhh
is the TCBDEB field (bytes 8 through 11): starting address of the DEB queue.

TIOT hhhhhhhh
is the TCBTIO field (bytes 12 through 15): starting address of the TIOT.

CMP hhhhhhhh
is the TCBCMP field (bytes 16 through 19): task completion code in hexadecimal. System codes are shown in the third through fifth digits and user codes in the sixth through eighth.

TRN hhhhhhhh
is the TCBTRN field (bytes 20 through 23): starting address of control core (table) for controlling testing of the task by TESTRAN.

MSS hhhhhhhh
is the TCBMSS field (bytes 24 through 27): starting address of the main storage supervisor's boundary box.

PK/FLG hhhhhhhh
contains, in the first 2 digits, the TCBPKF field (byte 28): protection key.

FLG hhhhhhhh
contains, in the first 4 digits, the last 2 bytes of the TCBFLGS field (bytes 32 and 33): last 2 flag bytes.

contains, in the next 2 digits, the TCBLMP field (byte 34): in systems with PCP, both digits are zeros; in systems with MFT, number of resources on which the task is queued.

contains, in the last 2 digits, the TCBDSP field (byte 35):

- Reserved in PCP and MFT without subtasking; both digits are zero.
- In MFT with subtasking, this field contains the dispatching priority of the TCB.

LLS hhhhhhhh
is the TCBLLS field (bytes 36 through 39): starting address of the RB most recently added to the load list.

JLB hhhhhhhh
is the TCBJLB field (bytes 40 through 43): starting address of the DCB for the JOBLIB data set.

JST hhhhhhh
is the TCBJST field (bytes 44 through 47). Not currently used in PCP or MFT without subtasking. In MFT with subtasking - the starting address of the TCB for the job step task.

RG 0-7 and RG 8-15
is the TCBGRS field (bytes 48 through 111): contents of general registers 0 through 7 and 8 through 15, as stored in the save area of the TCB when a task switch occurred. These 2 lines appear only in TCBs of tasks other than the task in control when the dump was requested.

FSA hhhhhhhh
contains, in the first 2 digits, the TCBIDF field (byte 112): TCB identifier field.

contains, in the last 6 digits, the TCBFSA field (bytes 113 through 115): starting address of the first problem program save area. This save area was set up by the control program when the job step was initiated.

TCB hhhhhhhh
is the TCBTCB field (bytes 116 through 119): in systems with PCP, all digits are zeros; in systems with MFT, starting address of the next TCB of lower priority or, if this is the last TCB, zeros.

TME hhhhhhhh
is the TCBTME field (bytes 120 through 123): starting address of the timer element created when an STIMER macro instruction is issued by the task. This field is not printed if the computer does not contain the timer option.

PIB hhhhhhhh
is the TCBPIB field (bytes 124 through 127): starting address of the program information block (MFT) or zeros (PCP).

NTC hhhhhhhh (printed only in MFT)
is the TCBNTC field (bytes 128 through 131):

MFT without subtasking: zeros.

MFT with subtasking: the starting address of the TCB for the previous subtask on this subtask TCB queue. This field is zero both in the job step task, and in the TCB for the first subtask created by a parent task.

OTC hhhhhhhh (printed only in MFT)
is the TCBOTC field (bytes 132 through 135): starting address of the TCB for the parent task. Both in the TCB for the job step task, and in MFT systems without subtasking this field is zero.

LTC hhhhhhhh (printed only in MFT)
is the TCBLTC field (bytes 136 through 139): starting address of the TCB for the most recent subtask created by this task. This field is zero in the TCB for the last subtask of a job step, or in the TCB for a task that does not create subtasks. This field is always zero in an MFT system without subtasking.

IQE hhhhhhhh (printed only in MFT)
is the TCBIQE field (bytes 140 through 143).

MFT without subtasking: zero.

MFT with subtasking: starting address of the interruption queue element (IQE) for the ETXR exit routine. This routine is specified by the ETXR operand of the ATTACH macro instruction that created the TCB being dumped. The routine is to be entered when the task terminates.

ECB hhhhhhhh (printed only in MFT)
is the TCBECB field (bytes 144 through 147).

MFT without subtasking: zero.

MFT with subtasking: starting address of the ECB field to be posted by the control program at task termination. This field is zero if the task was attached without an ECB operand.

XTCB hhhhhhhh (printed only in MFT)
reserved for future use.

LP/FL hhhhhhhh (printed only in MFT)
MFT without subtasking: reserved.

MFT with subtasking: contains in the first byte, the limit priority of the task (byte 152). contains, in the last three bytes the field TCBFTFLG (bytes 153 through 155) - flag bytes.

RESV hhhhhhhh (printed only in MFT)
reserved for future use.

STAE hhhhhhhh
contains, in the first 2 digits, STAE flags (byte 160).

contains, in the last 6 digits, the TCBNSTAE field (bytes 161 through 163): starting address of the current STAE control block for the task. This field is zero if STAE has not been issued.

TCT hhhhhhhh
is the TCBTCT field (bytes 164 through 167):

PCP: Zeros.

MFT: Address of the Timing Control Table (TCT) Zeros of the System Management Facilities option is not present in the system.

USER hhhhhhhh
is the TCBUSER field (bytes 168 through 171): to be used as the user chooses.

DAR hhhhhhhh
contains, in the first 2 digits, Damage Assessment Routine (DAR) flags (byte 172);

MFT only, contains, in the last 6 digits, the secondary non-dispatchability bits (bytes 173 through 175).

RESV hhhhhhhh
reserved for future use.

JSCB hhhhhhhh
is the TCBJSCB field (bytes 180 through 183): the last three bytes contain the address of the Job Step Control Block.

```
ACTIVE RBS

cccc hhhhhh  NM cccccccc    SZ/STAB hhhhhhhh   USE/EP hhhhhhhh   PSW hhhhhhhh hhhhhhhh   Q hhhhhh   WT/LNK hhhhhhhh
             RG 0-7    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh
             RG 8-15   hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh    hhhhhhhh
```

ACTIVE RBS
identifies the next lines as the contents of the active RBs queued to the TCB.

cccc hhhhhh
indicates the RB type and its starting address.

The RB types are:

PRB Program request block

SIRB Supervisor interrupt request block

LPRB Loaded program request block

IRB Interruption request block

SVRB Supervisor request block

NM xxxxxxxx
is the XRBNM field (bytes 0 through 7): in PRB, LRB, and LPRB, the program name; in IRB, the first byte contains flags for the timer or, if

the timer is not being used, contains no meaningful information; in SVRB for a type 2 SVC routine, the first 4 bytes contain the TTR of the load module in the SVC library, and the last 4 bytes contain the SVC number in signed, unpacked decimal.

SZ/STAB hhhhhhhh
: contains in the first 4 digits, the XRBSZ field (bytes 8 and 9): number of contiguous doublewords in the RB, the program (if applicable), and associated supervisor work areas.

  contains in the last 4 digits, the XSTAB field (bytes 10 and 11): flag bytes.

USE/EP hhhhhhhh
: contains, in the first 2 digits, the XRBUSE field (byte 12): use count.

  contains, in the last 6 digits, the XRBEP field (bytes 13 through 15): address of entry point in the associated program.

PSW hhhhhhhh hhhhhhhh
: is the XRBPSW field (bytes 16 through 23): resume PSW.

Q hhhhhh
: is the last 3 bytes of the XRBQ field (bytes 25 through 27): in PRB and LPRB, starting address of an LPRB for an entry identified by an IDENTIFY macro instruction; in IRB, starting address of a request element; in SVRB for a type 3 or 4 SVC, size of the program in bytes.

WT/LNK hhhhhhhh
: contains, in the first 2 digits, the XRBWT field (byte 28): wait count.

  contains, in the last 6 digits, the XRBLNK field (bytes 29 through 31): primary queuing field. It is the starting address of the previous RB for the task or, in the first RB to be placed on the queue, the starting address of the TCB.

RG 0-7 and RG 8-15
: is the XRBREB field (bytes 32 through 95 in IRBs and SVRBs): contents of general registers 0 through 15 stored in the RB. These 2 lines do not appear for PRBs, LPRBs, and LRBs.

```
LOAD LIST
cccc hhhhhh  NM cccccccc   SZ/STAB hhhhhhhh  USE/EP hhhhhhhh  PSW hhhhhhhh hhhhhhhh   Q hhhhhh   WT/LNK hhhhhhhh
```

LOAD LIST
: identifies the next lines as the contents of the load list queued to the TCB.

cccc hhhhhh
: indicates the RB type and its starting address.

  The RB types are:

LRB	Loaded request block
LPRB	Loaded program request block
D-LPRB	Dummy loaded program request block. (Present if the resident reenterable load module option was selected in MFT).

NM cccccccc
: is the XRBNM field (bytes 0 through 7): program name.

SZ/STAB hhhhhhhh
: contains, in the first 4 digits, the XRBSZ field (bytes 8 and 9): number of contiguous doublewords for the RB, the program (if applicable), and associated supervisor work areas.

  contains, in the last 4 digits, the XSTAB field (bytes 10 and 11): flag bytes.

USE/EP hhhhhhhh
: contains, in the first 2 digits, the XRBUSE field (byte 12): use count.

  contains, in the last 6 digits, the XRBEP field (bytes 12 through 15): address of entry point in the program.

```
JOB IPCT41            STEP EXSTEP          TIME 002409    DATE 99366                                          PAGE 0001

COMPLETION CODE       SYSTEM = 837

PSW AT ENTRY TO ABEND  FF04000D 5000C408

TCB   02F028  RBP    0002EC78   PIE     00000000   DEB   0002ED34   TIO 000302F0   CMP    80837000   TRN  00000000
              MSS    01031738   PK-FLG  F0850409   FLG   00000000   LLS 000309B0   JLB    00000000   JPQ  000301E8
              FSA    01069768   TCB     00000000   TME   00000000   JST 0002F028   NTC    00000000   OTC  00030508
              LTC    00000000   IQE     00000000   ECB   000304B4   STA 00000000   D-PQE  00032668   SQS  0002EAA0
              NSTAE  00000000   TCT     00030268   USER  00000000   DAR 00000000   RESV   00000000   JSCB 0003146C


ACTIVE RBS

PRB   030DE8  RESV    00000000   APSW    00000000    WC-SZ-STAB 00040082    FL-CDE 00031290    PSW FFF50006 7003553E
              Q/TTR   00000000   WT-LNK  0002F028

PRB   0309B8  RESV    00000000   APSW    00000000    WC-SZ-STAB 00040002    FL-CDE 00030E80    PSW FFF50037 5207EC4A
              Q/TTR   00000000   WT-LNK  00030DE8

SVRB  02F0F0  TAB-LN  00980400   APSW    F5F5F0F2    WC-SZ-STAB 0012D002    TQN     00000000   PSW FF04000D 5000C408
              Q/TTR   00003C0F   WT-LNK  000309B8
              RG 0-7    00000FD9    000396F4    00000003    00000006    00000073    0003BC00    00036F88    0003CC33
              RG 8-15   00039100    000396F4    00069620    0003A158    0003ACE1    000395C0    5207F434    0007FC10
              EXTSA     F2F8F2F5    F3D6C340    0006DDE0    0002EFF4    0002EFC4    0006DE88    00000837    0003036C
                        80002648    00000001    0006DEE0    C3C45D04

SVRB  02F170  TAB-LN  008803C8   APSW    F2F0F1C3    WC-SZ-STAB 0012D002    TQN     00000000   PSW 00040033 5000C0CE
              Q/TTR   00006109   WT-LNK  0002F0F0
              RG 0-7    80000000    80837000    000396F4    4000C182    0006DDE0    0002EED4    0002EEC4    0006DE88
              RG 8-15   00000837    0003036C    80002648    00000001    0006DEE0    00002648    0000DB68    00000001
              EXTSA     000029BF    0006DD88    2000EEEE    0006DBE0    FE030000    0002F1EC    0002F1F4    F2F8F2C9
                        C5C1F0F1    C9C5C128    C1C2C5D5    C4078386

SVRB  02EC78  TAB-LN  00C803C8   APSW    F1F0F5C1    WC-SZ-STAB 0012D002    TQN     00000000   PSW FF040001 4007E8A4
              Q/TTR   00006201   WT-LNK  0002F170
              RG 0-7    00000000    0002F1D0    8000BDC8    0000DB68    0002F028    0002F170    00031290    00000000
              RG 8-15   0002F028    4000BD3A    0002F028    0006DD88    00030320    0002F1F4    4000D594    00000000
              EXTSA     00620300    00090040    0008000A    18002648    00000040    00090041    00028460    00000018
                        0012C002    00000000    00000000    00000000


LOAD LIST

         NE 000308F8   RSP-CDE 020301E8         NE 00030DE0   RSP-CDE 01032390        NE 00031078   RSP-CDE 01032290
         NE 00031080   RSP-CDE 01032260         NE 000310C8   RSP-CDE 01032390        NE 00031170   RSP-CDE 01032200
         NE 000311C0   RSP-CDE 010323C0         NE 00000000   RSP-CDE 01030BF0


CDE

      031290        ATR1 C8   NCDE 000000   ROC-RB 00030DE8   NM GO          USE 01    EPA 035508   ATR2 20    XL/MJ 031280
      030E80        ATR1 C8   NCDE 031290   ROC-RB 000309B8   NM IEKAA00     USE 01    EPA 036240   ATR2 20    XL/MJ 02F398
      0301E8        ATR1 31   NCDE 030BF0   ROC-RB 00000000   NM IGC0A05A    USE 02    EPA 06C980   ATR2 28    XL/MJ 0304B0
      032390        ATR1 B8   NCDE 0323C0   ROC-RB 00000000   NM IGG019CD    USE 06    EPA 07FA00   ATR2 20    XL/MJ 032380
      032290        ATR1 B8   NCDE 0322C0   ROC-RB 00000000   NM IGG019BA    USE 05    EPA 07F4A0   ATR2 20    XL/MJ 032280
      032260        ATR1 B8   NCDE 032290   ROC-RB 00000000   NM IGG019BB    USE 05    EPA 07F880   ATR2 20    XL/MJ 032250
      032390        ATR1 B8   NCDE 0323C0   ROC-RB 00000000   NM IGG019CD    USE 06    EPA 07FA00   ATR2 20    XL/MJ 032380
      032200        ATR1 B8   NCDE 032230   ROC-RB 00000000   NM IGG019AJ    USE 03    EPA 07F3A0   ATR2 20    XL/MJ 0321F0
      0323C0        ATR1 B8   NCDE 0323F0   ROC-RB 00000000   NM IGG019AR    USE 04    EPA 07FC10   ATR2 20    XL/MJ 032380
      0303F0        ATR1 39   NCDE 030E80   ROC-RB 00000000   NM IEWSZOVR    USE 01    EPA 06C480   ATR2 20    XL/MJ 030B88


XL                                                LN          ADR          LN          ADR          LN          ADR

      031280    SZ 00000010    NO 00000001       800002E8    00035508
      02F398    SZ 0000004C    NO 00000001       80016E38    000359C8    000359C8    00030800    010A0400    010D0500
                                                 011C0300    011D0300    011E0200    01290400    012F0500    01300500
                                                 01320300    013A0100    01460600    01480400    014D0500
      0304B0    SZ 00000010    NO 00000001       80000680    0006C980
      032380    SZ 00000010    NO 00000001       80000210    0007FA00
      032280    SZ 00000010    NO 00000001       80000180    0007F4A0
      032250    SZ 00000010    NO 00000001       80000058    0007F880
      032380    SZ 00000010    NO 00000001       80000210    0007FA00
      0321F0    SZ 00000010    NO 00000001       80000100    0007F3A0
      032380    SZ 00000010    NO 00000001       80000090    0007FC10
      030B88    SZ 00000010    NO 00000001      .80000350    0006C480


DEB

02ED00                                               00000D50 00000D50 00000D50 00000D50    *................................*
02ED20    00000D50 00000000 0000020A 00002BEC        0F000000 0002F028 0402EED4 98000000    *......................O....M....*
02ED40    8F000000 01000000 00000000 FE06DD88        0402ED10 18002648 00000031 00010032    *................................*
02ED60    0001000B 00010001 C2C2C2C1 C3C40000        00000000 00000000 00000000 C3C40000    *........BBBACD..............CD..*
```

Figure 24A. Sample of Complete ABEND Dump (MVT)

```
DEB                                                                                                  PAGE 0002

02EEA0                                                 00000050 00000050 00000D50 00000050   *..P.............................*
02EEC0   00000D50 00000000 0000020F 00011AEC           2A003000 0302E028 04000000 88000000   *......................0.........*
02EEE0   0E000000 10000000 00000000 FE0396E4           0402EEB0 18002648 00000039 0009003E   *.............4..................*
02EF00   00030032 18002648 0000003F 0009003F           0008000A 18002648 0000003F 00090040   *...............................*
02EF20   0008000A 18002648 00000040 00090041           0008000A 18002648 00000041 00090042   *..........  ....................*
02EF40   0008000A 18002648 00000042 00090043           0008000A 18002648 00000043 00090044   *................................*
02EF60   0008000A 18002648 00000044 00090045           0008000A 18002648 00000045 00090046   *................................*
02EF80   0008000A 18002648 00000046 00090047           0008000A 18002648 00000047 00090048   *................................*
02EFA0   0008000A 18002648 00000048 00090049           0008000A 18002648 00000049 0009004A   *................................*
02EFC0   0008000A 18002648 0000004A 0009004B           0008000A 18002648 0000004B 0009004C   *................................*
02EFE0   0008000A 18002648 0000004C 0009004D           0008000A 00010001 C1D9C1D1 C3C4F6C0   *.........................ARAJCD6.*

TIOT    JOB   IPCT41       STEP EXSTEP
      DD             14040101     PGM=*.DD     00230E00      80002648
      DD             14040100     SYSABEND     00240900      80002648
      DD             14040180     FT06F001     00240C00      80002648
      DD             14040100     FTNLIN       00250100      80003984
      DD             14000000     SYSPUNCH     00250800      0000000C
      DD             14040100     SYSPRINT     00240E00      80002648
      DD             14040101     SYSIN        00250A00      80002648

MSS             ************ SPQE ************     *************** DQE ***************     ******* FQE ********
                FLGS   NSPQE      SPID     DQE         BLK       FQE       LN        NDQE         NFQE        LN

      031728    00     031740     251      031250    00035000 00035000 00000800 000310F0     00000000    00000508
                                                     00035800 00035800 00017000 00000000     00000000    000001C8
      031740    00     031488     252      0314C0    0006D800 0006D800 00000800 00030878     00000000    00000598
                                                     0006C000 0006C000 00000800 000303D8     00000000    00000480
                                                     0006C800 0006C800 00000800 0002F388     00000000    00000180
                                                     00068800 00068800 00000800 00000000     00000000    000001A0
      031488    C0     000000     000      031400
      031400    60     000000     000      031488    00060000 0006D748 00000800 00000000     0006D000    00000020
                                                                                             00000000    00000518

D-PQE   00032660    FIRST 00031460    LAST 00031460
PQE   031460    FFB 0004C800    LFB 0004C800    NPQ 00000000    PPQ 00000000
                TCB 00030E08    RSI 00030000    RAD 00035000    FLG 0000

FBQE 04C800    NFB 00031460    PFB 00031460    SZ 0001F000

QCB TRACE

MAJ 0311C8    NMAJ 000301D0    PMAJ 0001C6A0    FMIN 00031088    NM   SYSDSN

MIN 031088    FQEL 00031698    PMIN 000311C8    NMIN 00000000    NM FF   SYS1.MACLIB

              NQEL 00000000    PQEL 80031088    TCB   00030508    SVRB 000301D0

MAJ 0301D0    NMAJ 00000000    PMAJ 000311C8    FMIN 000301A0    NM   SYSIEA01

MIN 0301A0    FQEL 00030190    PMIN 000301D0    NMIN 00000000    NM F0   IEA

              NQEL 00000000    PQEL 000301A0    TCB   0002E028    SVRB 0002EBE8

SAVE AREA TRACE

SA    060768   WD1 00000000    HSA 00000000    LSA 00000000    RET 00000000    EPA 00000000    R0   00000000
               R1  00000000    R2  00000000    R3  00000000    R4  00000000    R5  00000000    R6   00000000
               R7  00000000    R8  00000000    R9  00000000    R10 00000000    R11 00000000    R12  00000000

INTERRUPT AT 07EC4A

PROCEEDING BACK VIA REG 13

SA    0395C0   WD1 957035FF    HSA 70004780    LSA 95789180    RET 80064710    EPA 958C1811    R0   5203936E
               R1  9207E3A0    R2  0006D570    R3  000396E4    R4  000396E4    R5  0006D570    R6   7E06D5CC
               R7  0006060B    R8  0006D78C    R9  00000ED8    R10 0007EC10    R11 5207E434    R12  0007EC10

SA    004780   WD1 47000000    HSA FF000000    LSA 00000000    RET 00000000    EPA 47A00000    R0   FF000000
               R1  00000000    R2  00000000    R3  47800000    R4  FF000000    R5  00000000    R6   00000000
               R7  47C00000    R8  FF000000    R9  00000000    R10 00000000    R11 47D00000    R12  FF000000

NUCLEUS

000000   00000000 00000000 00000000 00000000           00000368 00000000 FF040080 8003B724   *................................*
000020   FF050001 4007EC3C FFF50001 02036CF2           0000FE00 00000C00 FF060336 80000000   *.... ....5.....2................*
000040   0003A7C8 0C000000 00072540 00000868           0836F88C 0001389C 00040000 0000F678   *...H..............Y...........6.*
```

Figure 24B. Sample of Complete ABEND Dump (MVT)

```
JOB cccccccc        STEP cccccccc        TIME dddddd    DATE ddddd    ID = ddd                               PAGE dddd

COMPLETION CODE     SYSTEM = hhh (or USER = dddd)

PSW AT ENTRY TO ABEND (SNAP)   hhhhhhhh hhhhhhhh
```

JOB cccccccc
    is the job name specified in the JOB statement.

STEP cccccccc
    is the step name specified in the EXEC statement for the problem program associated with the task being dumped.

TIME dddddd
    is the hour (first 2 digits), minute (next 2 digits), and second (last 2 digits) when the abnormal termination dump routine began processing.

DATE ddddd
    is the year (first 2 digits) and day of the year (last 3 digits). For example, 67352 would be December 18, 1967.

ID=ddd
    is an identification of the dump. For dumps requested by an ABEND macro instruction, this identification is:

- Absent if the dump is of the task being abnormally terminated.
- 001 if the dump is of a subtask of the task being abnormally terminated. (Note that, when a task is abnormally terminated, its subtasks are also abnormally terminated.)
- 002 if the dump is of a task that directly or indirectly created the task being abnormally terminated, up to and including the job step task.

PAGE dddd
    is the page number. Appears at the top of each page. Page numbers begin at 0001 for each task or subtask dumped.

COMPLETION CODE SYSTEM=hhh or COMPLETION CODE USER=dddd
    is the completion code supplied by the control program (SYSTEM=hhh) or the problem program (USER=dddd).

PSW AT ENTRY TO ABEND hhhhhhhh hhhhhhhh or PSW AT ENTRY TO SNAP hhhhhhhh hhhhhhhh
    is the PSW for the problem program or control program routine that had control when abnormal termination was requested, or when the SNAP macro instruction was executed. It is not necessarily the PSW at the time the error condition occurred.

```
TCB  hhhhhh  RBP    hhhhhhhh   PIE      hhhhhhhh   DEB  hhhhhhhh   TIO hhhhhhhh   CMP    hhhhhhhh   TRN  hhhhhhhh
             MSS    hhhhhhhh   PK-FLG hhhhhhhh     FLG  hhhhhhhh   LLS hhhhhhhh   JLB    hhhhhhhh   JPQ  hhhhhhhh
             RG 0-7      hhhhhhhh     hhhhhhhh     hhhhhhhh     hhhhhhhh     hhhhhhhh     hhhhhhhh     hhhhhhhh     hhhhhhhh
             RG 8-15     hhhhhhhh     hhhhhhhh     hhhhhhhh     hhhhhhhh     hhhhhhhh     hhhhhhhh     hhhhhhhh     hhhhhhhh
             FSA    hhhhhhhh   TCB      hhhhhhhh   TME  hhhhhhhh   JST hhhhhhhh   NTC    hhhhhhhh   OTC  hhhhhhhh
             LTC    hhhhhhhh   IQE      hhhhhhhh   ECB  hhhhhhhh   STA hhhhhhhh   D-PQE  hhhhhhhh   SQS  hhhhhhhh
             NSTAE  hhhhhhhh   TCT      hhhhhhhh   USER hhhhhhhh   DAR hhhhhhhh   RESV   hhhhhhhh   JSCB hhhhhhhh
```

TCB hhhhhh
    is the starting address of the TCB.

RBP hhhhhhhh
    is the TCBRBP field (bytes 0 through 3): starting address of the active RB queue and, consequently, the most recent RB on the queue.

PIE hhhhhhhh
    is the TCBPIE field (bytes 4 through 7): starting address of the program interruption element (PIE) for the task; however, in an abnormal termination dump for the task causing the abnormal termination, zeros. The field is zeroed by the ABEND routine to prevent interruptions during dumping.

DEB hhhhhhhh
is the TCBDEB field (bytes 8 through 11): starting address of the DEB queue. Under the heading DEB in the dump, the prefix section for the first DEB in the queue is presented in the first 8-digit entry on the first line. The 6-digit entry at the left of each line under DEB is the address of the second column on the line, whether or not the column is filled.

TIO hhhhhhhh
is the TCBTIO field (bytes 12 through 15): starting address of the TIOT.

CMP hhhhhhhh
is the TCBCMP field (bytes 16 through 19): task completion code or contents of register 1 when the dump was requested. System codes are given in the third through fifth digits and user codes in the sixth through eight digits.

TRN hhhhhhhh
is the TCBTRN field (bytes 20 through 23): starting address of the control core (table) for controlling testing of the task by TESTRAN.

MSS hhhhhhhh
is the TCBMSS field (bytes 24 through 27): starting address of SPQE most recently added to the SPQE queue.

PK-FLG hhhhhhhh
contains, in the first 2 digits, the TCBPKF field (byte 28): protection key.

contains, in the last 6 digits, the first 3 bytes of the TCBFLGS field (bytes 29 through 31): first 3 flag bytes.

FLG hhhhhhhh
contains, in the first 4 digits, the last 2 bytes of the TCBFLGS (bytes 32 and 33): last 2 flag bytes.

contains, in the next 2 digits, the

TCBLMP field (byte 34): limit priority (converted to an internal priority, 0 to 255).

contains, in the last 2 digits, the TCBDSP field (byte 35): dispatching priority (converted to an internal priority, 0 to 255).

LLS hhhhhhhh
is the TCBLLS field (bytes 36 through 39): starting address of the load list element most recently added to the load list.

JLB hhhhhhhh
is the TCBJLB field (bytes 40 through 43): starting address of the DCB for the JOBLIB data set.

JPQ hhhhhhhh
is the TCBJPQ field (bytes 41 through 47): when translated into binary bits:

- Bit 0 is the purge flag.
- Bits 1 through 7 are reserved for future use and are zeros.
- Bits 8 through 31 are the starting address of the queue of CDEs for the job pack area control queue, which is for programs acquired by the job step.

The TCBJPQ field is used only in the first TCB in the job step; it is zeros for all other TCBs.

RG 0-7 and RG 8-15
is the TCBGRS field (bytes 48 through 111): contents of general registers 0 through 7 and 8 through 15, as stored in the save area of the TCB when a task switch occurred. These 2 lines appear only in dumps of tasks other than the task in control when the dump was requested.

FSA hhhhhhhh
contains, in the first 2 digits, the TCBQEL field (byte 112): count of enqueue elements.

contains, in the last 6 digits, the TCBFSA field (bytes 113 through 115): starting address of the first problem program save area. This save area was set up by the control program when the job step was initiated.

TCB hhhhhhhh
is the TCBTCB field (bytes 116 through 119): starting address of the next lower priority TCB on the TCB queue or, if this is the lowest priority TCB, zeros.

TME hhhhhhhh
is the TCBTME field (bytes 120 through 123): starting address of the timer element created when an STIMER macro instruction is issued by the task.

Appendix J: Control Block Pointers

This appendix summarizes the contents of the control blocks that are most useful in debugging. Control blocks are presented in alphabetical order, with displacements in decimal, followed by the hexadecimal counterpart in parentheses. Figure 38 illustrates control block relationships in the System/360 Operating System. Figure 39 shows relationships between storage control elements in a system with MVT.

CVT - Communications Vector Table

+0	Address of TCB control words
+53(35)	Address of entry point of ABTERM
+193(C1)	Address of secondary CVT (used only with Model 65 Multiprocessing systems)

DCB - Data Control Block

+40(28)	ddname (before open); offset to ddname in TIOT (after open)
+45(2D)	DEB address
+69(45)	IOB address

DEB - Data Extent Block

+1	TCB address
+5	Address of next DEB
+25(19)	DCB address
+33(21)	UCB address
+38(26)	Address of start of extent
+42(2A)	Address of end of extent

ECB - Event Control Block

+1	RB address or completion code

RB - Request Block (MVT)

+4	Last half of user's PSW
+13(D)	CDE address
+16(10)	Resume PSW
+29(1D)	Address of previous RB

TIOT - Task Input/Output Table

+0	Job name
+8	Step name
+24(18)	DD entries begin (one variable-length entry for each DD statement)
+0	Length of DD entry
+4	ddname
+16(10)	Device entries begin (one 4-byte entry for each device)
+20(14)	Next device entry (if there is one)
.	
.	
	(Next DD entry begins at 24(18) plus length of first DD entry)

TCB - Task Control Block (PCP and MFT)

+1	Address of most recent RB
+9	Address of most recent DEB
+13(D)	TIOT address