File No. S360-20
Order No. GC28-6670-5

OS

IBM Systems Reference Library

IBM System/360 Operating System: Programmer's Guide to Debugging

OS Release 21

This publication describes, in assembler language terms, the major debugging facilities provided with the System/360 Operating System. It is written for the programmer who debugs system and application programs.

The text explains those aspects of system control pertinent to debugging, tells what information each debugging facility offers, and outlines procedures for obtaining and interpreting dumps.

The various types of storage dumps available under the MFT and MVT control programs and event tracing facilities are described.

Debugging facilities inherent in higher languages and additional aids are discussed in other SRL publications.

Sixth Edition (March, 1972)

This is a major revision of, and obsoletes GC28-6670-4 and Technical Newsletters GN28-2457 and GN28-2472. See Summary of Amendments following the Contents. Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

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

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

A form for readers' comments is provided at the back of 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. Comments become the property of IBM.

Preface

This publication is intended to help you use the debugging facilities provided with the IBM System/360 Operating System. It describes, in assembler language terms, the major debugging facilities provided with the System/360 Operating System, and is directed towards the programmer who deals with system and application program problems.

The publication is divided into three principal parts: "Section 1: Operating System Concepts;" "Section 2: Interpreting Dumps;" and "Section 3: Tracing Aids," plus an Introduction and a set of Appendixes that provide specific debugging information.

The Introduction provides a brief survey of the material presented in the balance of the publication.

Section 1 deals with internal aspects of the operating system that are pertinent to debugging. A working knowledge of this information will provide you with the means of determining the status of the system at the time of failure, and the course of events which led up to that failure. The general precedure for debugging with an operating system dump (Appendix A) assumes knowledge of this control flow.

Section 2 includes instructions for invoking, reading, and interpreting storage dumps of systems with MFT or MVT control programs. The material is intended to aid you in interpreting dumps and isolating errors.

Section 3 deals with the save area chain, the Trace Option, and the Generalized Trace Facility. Output from the Generalized Trace Facility is discussed.

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 Messages and Codes, describe additional debugging aids provided for the assembler language programmer.

Notice: Coding level information presented in this publication must not be used for coding purposes or exposure to changes in implementation may result. The information is presented for debugging purposes only.

PREREQUISITE PUBLICATIONS

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

IBM System/360 Operating System:

Supervisor Services and Macro Instructions, GC28-6646

Data Management Services, GC26-3746

REFERENCE PUBLICATIONS

IBM System/360 Operating System:

System Control Blocks, GC28-6628

Messages and Codes, GC28-6631

Data Management Macro Instructions, GC26-3794

Service Aids, GC28-6719

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

TCAM Serviceability Aids, GY30-2027.

TCAM, GY30-2029.

TSO Control Program, GY27-7199.

Figures

Summary of Amendments for GC28-6670-5 OS Release 21

PCP REMOVAL

References to the PCP version of Operating System/360 have been deleted from the publication.

TESTRAN REMOVAL

References to the TESTRAN testing facility of Operating System/360 have been deleted from the publication.

IMDPRDMP SERVICE AID OUTPUT

Storage dumps as formatted and displayed by the IMDPRDMP service aid are now discussed in this publication. This material was formerly in the Service Aids publication, GC28-6719.

GENERALIZED TRACE FACILITY (GTF) OUTPUT

GTF trace records, as processed by the EDIT function of the IMDPRDMP service aid are illustrated and discussed in Section 3 of the publication.

DEVICE SUPPORT

The sense byte information given in Appendix G is updated to include information for the:

IBM 3420 Magnetic Tape Unit and 3803 Tape Control
IBM 2596 Card Read Punch
IBM 3505 Card Reader
IBM 3525 Card Punch
IBM 3410 Magnetic Tape Unit
IBM 3411 Magnetic Tape Unit and Control

PROBLEM DETERMINATION

Addition of an Appendix discussing problem determination aids for OPEN/CLOSE/EOV processing.

Updating of the completion codes and service aids Appendixes to reflect release 21 changes.

The Console Dump facility, used to obtain a storage dump for later processing by IMDPRDMP, is briefly described in the storage dump and IMDPRDMP formatting section of the publication.

MISCELLANEOUS

Editorial improvements and corrections to existing material have been made throughout the publication.

Summary of Amendments for GC28-6670-4 as Updated by GN28-2457 and GN28-2472 OS Release 20.1

TCAM

Section 2: ABEND/SNAP Dump (PCP and MFT)
ABEND/SNAP Dump (MVT)
Appendix A
Appendix H
A brief description of TCAM debugging Aids and a new SVC.

TSO

Section 2: TSO Control Blocks
Appendix A
The addition of new SVCs and a summary of the control blocks formatted by IMDPRDMP.

3330, 2305, 2319

Appendix F
Additional of sense byte information fo new devices.

MISCELLANEOUS

Appendix C
1. Addition of module name prefixes for emulator programs.

Appendix G
2. New features of service aid program IMAPTFLE.

Summary of Amendments for GC28-6670-3 OS Release 20

IMDPRDMP

"Guide to Using a Storage Image Dump"
IMDPRDMP is used instead of IEAPRINT to print MFT and MVT dumps.

TSO

Appendix A
New SVCs in Appendix A. This information is for planning purposes only.

Introduction

To debug 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. To provide an insight into the IBM System/360 Operating System and its complex aspects of task management and storage supervision, Section 1 of this publication provides an orientation in the control functions of the operating system.

The IBM System/360 Operating System provides extensive debugging 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. Tracing facilities.

Dumps are discussed in Section 2 and tracing facilities in Section 3.

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 dumps are produced by either the system dump facility at the time of a system failure, or by a dump program created through use of the IMDSADMP service aid. IMDSADMP programs must be loaded into storage through use of the IPL facilities and are intended for use in situations in which the system is not operative, e.g., a disabled wait state or an unending system loop.

The system dump facility writes to the SYS1.DUMP data set. The IMDPRDMP service aid is used to format and print the SYS1.DUMP data set. IMDPRDMP output is described in this publication. The IMDSADMP programs write to tape (high-speed dump) or to tape or printer (low-speed dump). The output tape produced by the high-speed dump must be processed by the IMDPRDMP program; low-speed output to tape may be processed by IMDPRDMP, IEBPTPCH or the IEBGENER utility program.

Storage dumps taken by the system dump facility consist of control information followed by a display of printable storage from location 00 to the capacity of storage. Storage words are displayed in both hexadecimal and EBCDIC notation. Storage dumps taken by an IMDSADMP program consist of register contents followed by a display of storage from location 00 to the capacity of storage. Notation is in both hexadecimal and EBCDIC.

Tracing facilities consist of the save area chain trace, the Trace Option and the Generalized Trace Facility.

The save area chain enables tracing of the save areas for each level of load module in a task. The save area trace is displayed in ABEND/SNAP and storage dumps.

The Trace Option, if installed in the system, provides records of system interruptions (IO, SIO, etc.) that are displayed in ABEND/SNAP and storage dumps.

The Generalized Trace Facility (GTF) enables selective tracing of system and application program events and records the information internally, in a table which is displayed in printouts of ABEND dumps and storage dumps, or externally in a data set which is processed by the IMDPRDMP service aid to provide edited and formatted GTF trace records. (For complete information on GTF see the Service Aids publication.) The GTF output, as processed by IMDPRDMP, is discussed in Section 3 of this publication.

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

This section introduces you to the control information that you must know to interpret dumps. It is divided into three topics:

- task management
- main storage supervision
- system control blocks and tables

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.

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

For a more detailed description of system control blocks and tables, refer to 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. the functions, interactions, and relationships to other system features of these items 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.

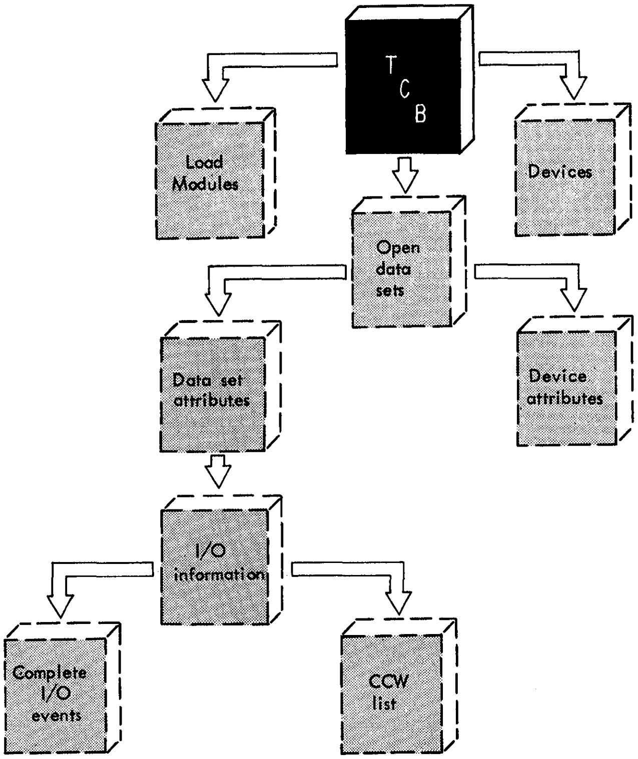


Figure 1. Control Information Available Through the TCB

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 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 (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.

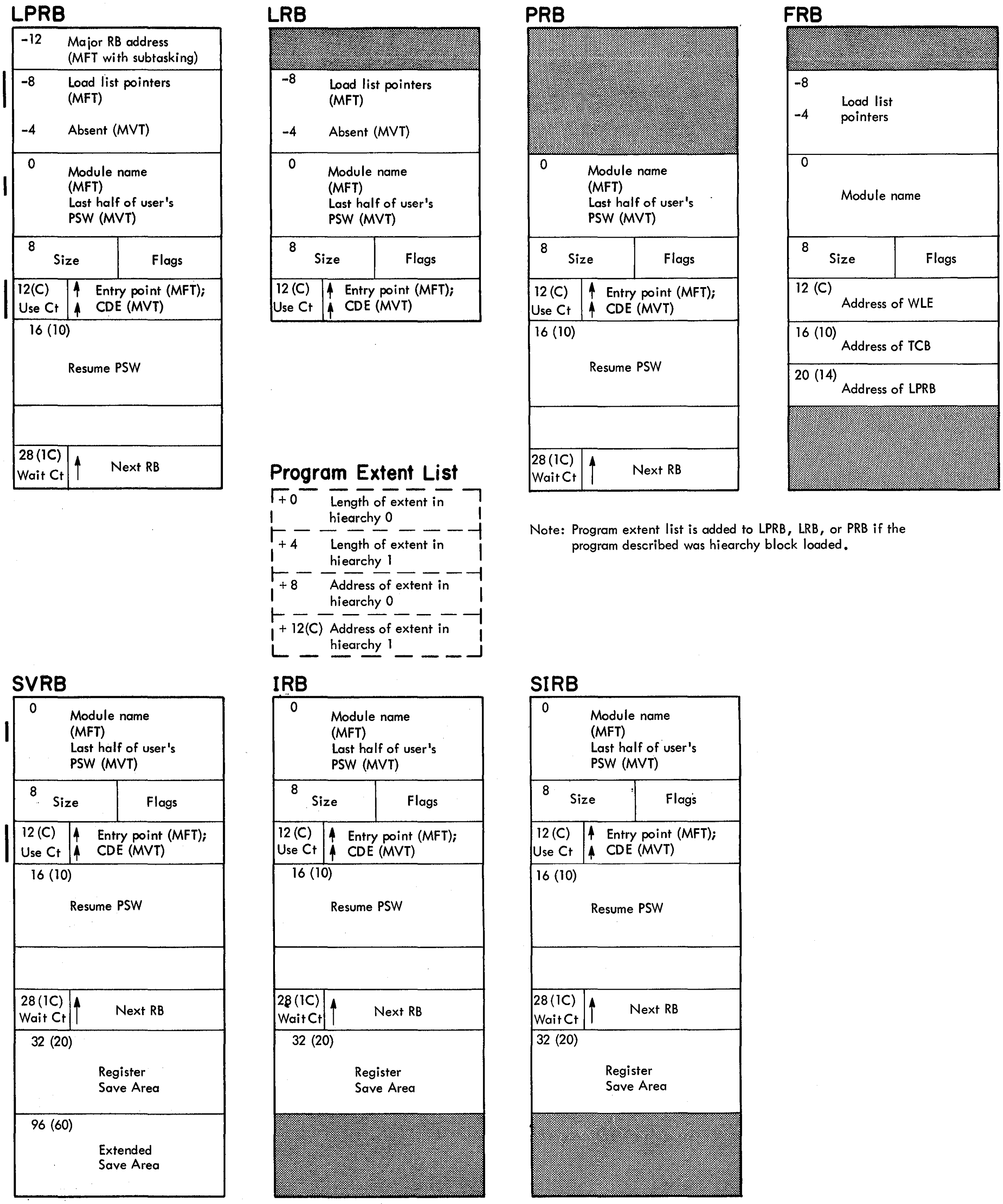
LRB (MFT only): The LRB is a shortened form of an LPRB. Routines associated with LRBs can be invoked only by a LOAD macro instruction. This attribute is assigned to a routine through the OL (only loadable) subparameter in the PARM parameter of the EXEC statement that executes the linkage editor. The 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.

This 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 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.


LPRB
-12 Major RB address (MFT with subtasking)
-8 Load list pointers (MFT)
-4 Absent (MVT)
0 Module name (MFT) Last half of user's PSW (MVT)
8 Size
Flags
12(C) Use Ct
Entry point (MFT); CDE (MVT)
16 (10)
Resume PSW
28 (1C) Wait Ct
Next RB
LRB
-8 Load list pointers (MFT)
-4 Absent (MVT)
0 Module name (MFT) Last half of user's PSW (MVT)
8 Size
Flags
12 (C) Use Ct
Entry point (MFT); CDE (MVT)
PRB
0 Module name (MFT) Last half of user's PSW (MVT)
8 Size
Flags
12 (C) Use Ct
Entry point (MFT); CDE (MVT)
16 (10)
Resume PSW
28 (1C) Wait Ct
Next RB
FRB
-8
-4
Load list pointers
0
Module name
8 Size
Flags
12 (C) Address of WLE
16 (10) Address of TCB
20 (14) Address of LPRB
Program Extent List
+0 Length of extent in hiearchy 0
+4 Length of extent in hiearchy 1
+8 Address of extent in hiearchy 0
+12(C) Address of extent in hiearchy 1
SVRB
0 Module name (MFT) Last half of user's PSW (MVT)
8 Size
Flags
12 (C) Use Ct
Entry point (MFT); CDE (MVT)
16 (10)
Resume PSW
28 (1C) Wait Ct
Next RB
32 (20)
Register Save Area
96 (60)
Extended Save Area
IRB
0 Module name (MFT) Last half of user's PSW (MVT)
8 Size
Flags
12 (C) Use Ct
Entry point (MFT); CDE (MVT)
16 (10)
Resume PSW
28 (1C) Wait Ct
Next RB
32 (20)
Register Save Area
SIRB
0 Module name (MFT) Last half of user's PSW (MVT)
8 Size
Flags
12 (C) Use Ct
Entry point (MFT); CDE (MVT)
16 (10)
Resume PSW
28 (1C) Wait Ct
Next RB
32 (20)
Register Save Area

Note: Program extent list is added to LPRB, LRB, or PRB if the program described was hiearchy block loaded.

Figure 2. RB Formats

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.

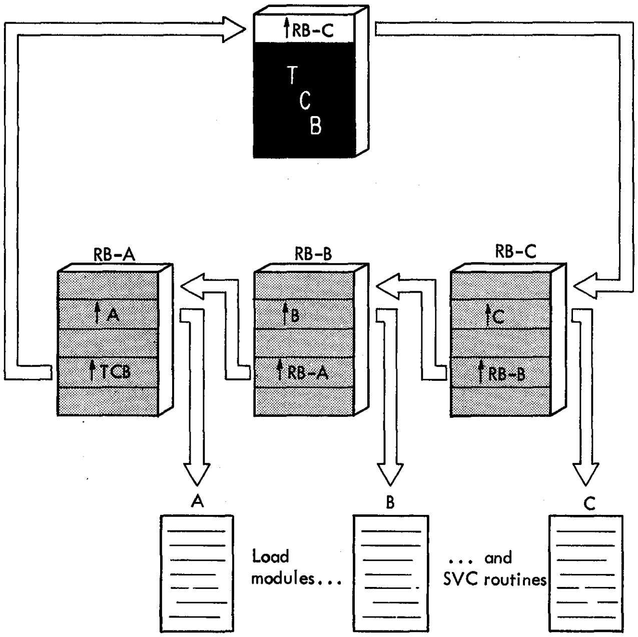


Figure 3. Active RB Queue

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 MFT (without subtasking): At this control program level, 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.

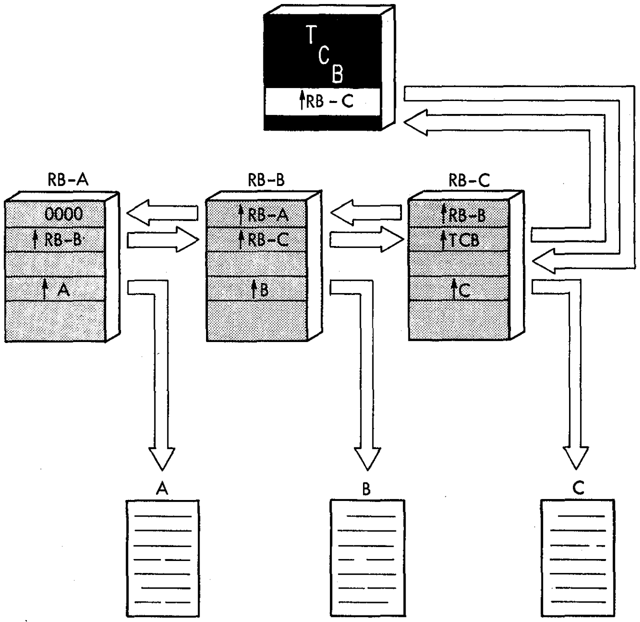


Figure 4. Load List (MFT)

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 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.

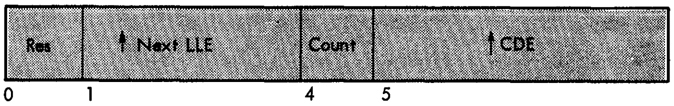
The contents directory is made up of three separate queues: the link pack area control queue (LPAQ), the job pack area control queue (JPAQ), and the load list.

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:



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 cnly 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.

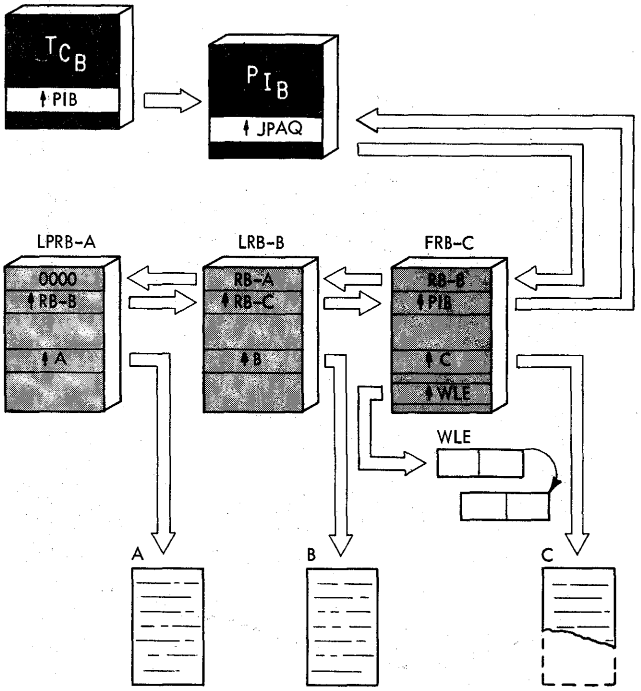


Figure 5. Job Pack Area Queue

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.

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

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 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 CDE 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 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 MFT without subtasking level, ATTACH effectively performs the same functions as LINK 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).

<u>XCTL</u>: 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 for MFT and MVT. There are, however, some major differences:

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 MFT (Without Subtasking)

Figure 6 is a snapshot of main storage in a system with MFT without subtasking.

The fixed area contains the nucleus (including TCB queue, transient area loading task, communications task, and master scheduler task), and the system queue area. Optionally it may contain access methods and SVC routine which are normally nonresident, a list of absolute addresses for routines which reside on direct access devices, and a reenterable load module area.

One TCB exists for each task. All TCBs are linked by dispatching priority in a TCB queue, beginning with the three resident tasks.

The dynamic area is divided into a maximum of 52 partitions. Each partition contains one task. The dynamic area can contain as many as 3 reading tasks, 36 writing tasks, and 15 job step tasks, providing that the total number of tasks does not exceed 52. Partition sizes and attributes are defined during system generation. Figure 7 shows the contents of an MFT partition.

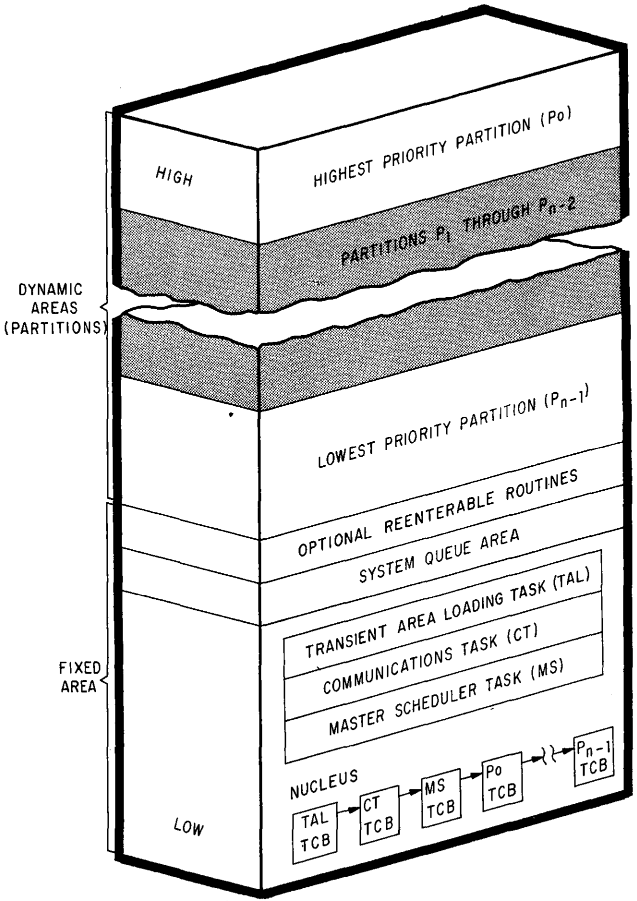

HIGH
HIGHEST PRIORITY PARTITION (Po)
PARTITIONS P1 THROUGH Pn-2
DYNAMIC AREAS (PARTITIONS)
LOWEST PRIORITY PARTITION (Pn-1)
OPTIONAL REENTERABLE ROUTINES
SYSTEM QUEUE AREA
TRANSIENT AREA LOADING TASK (TAL)
COMMUNICATIONS TASK (CT)
MASTER SCHEDULER TASK (MS)
FIXED AREA
NUCLEUS
TAL TCB
CT TCB
MS TCB
Po TCB
Pn-1 TCB
LOW

Figure 6. Main Storage Snapshot (MFT Without Subtasking)

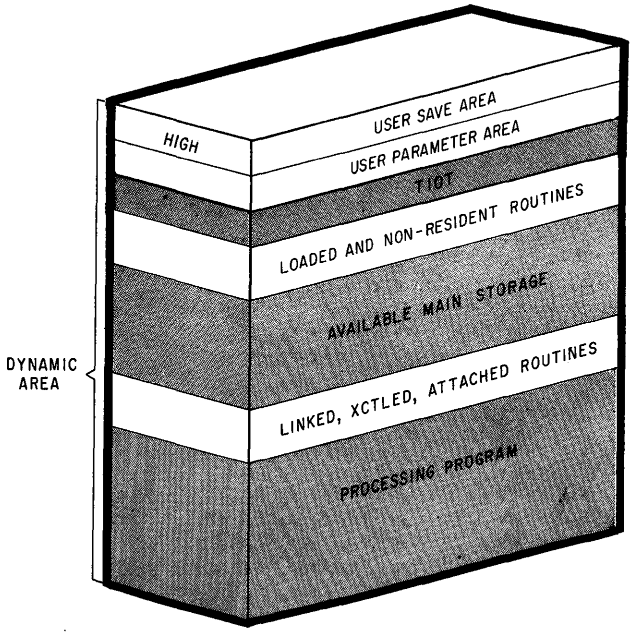


Figure 7. Partition (MFT Without Subtasking)

Jobs are processed sequentially in a partition, one job step at a time. An ATTACH macro instruction does not create a subtask.

Systems with MFT (With Subtasking): Operating Systems that provide multiprogramming with a fixed number of tasks with the subtasking option (MFT with subtasking) 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.

Figure 8 is a snapshot of main storage in an MFT system with subtasking. 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.

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.

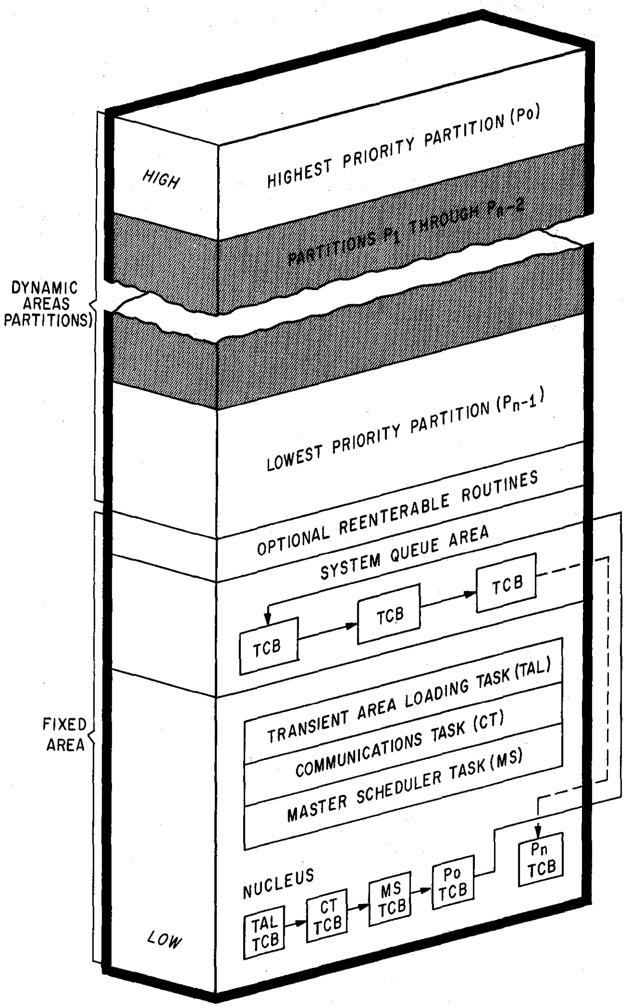

HIGHEST PRIORITY PARTITION (Po)
HIGH
PARTITIONS P1 THROUGH Pn-2
DYNAMIC AREAS PARTITIONS)
LOWEST PRIORITY PARTITION (Pn-1)
OPTIONAL REENTERABLE ROUTINES
SYSTEM QUEUE AREA
TCB
TCB
TCB
FIXED AREA
TRANSIENT AREA LOADING TASK (TAL)
COMMUNICATIONS TASK (CT)
MASTER SCHEDULER TASK (MS)
NUCLEUS
TAL TCB
CT TCB
MS TCB
Po TCB
Pn TCB
LOW

Figure 8. Main Storage Snapshot (MFT With Subtasking)

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.

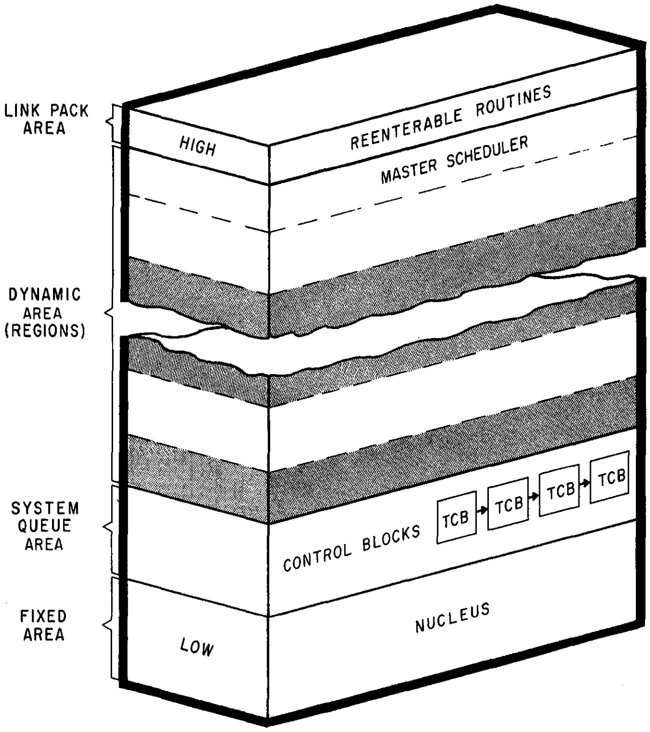


Figure 9. Main Storage Snapshot (MVT)

Main Storage Supervision

Storage control information is kept in a series of control blocks called queue elements. In systems with 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.

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 MFT (Without Subtasking)

The chain of storage control information in an MFT system without subtasking begins at a table called the main storage supervisor (MSS) boundary box, located in the system nucleus. There is one MSS boundary box for each partition. It is pointed to by the TCB (TCBMSS field) for the partition.

Each boundary box contains 3 words. The first word points to the Free Queue Element (FQE) associated with the highest free area in the partition. The second word points to the lowest limit of the partition. The third word contains the highest address in the partition plus 1.

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.

FQE: Each free area in a partition is described by an FQE. FQEs are chained beginning with the FQE associated with the free area having the highest address in the partition. 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:

↑ Next FQE	Number of bytes
0	4

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.

Figure 10 summarizes storage control in systems with MFT without subtasking.

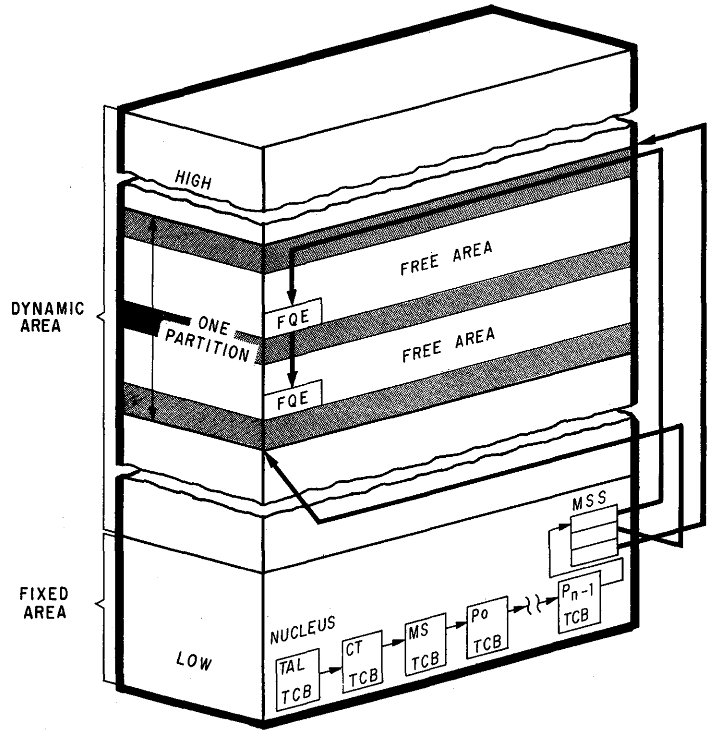


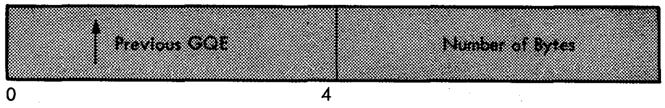
Figure 10. Storage Control for a Partition (MFT Without Subtasking)

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.

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:



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 11 summarizes the chaining of GQEs to a subtask TCB.

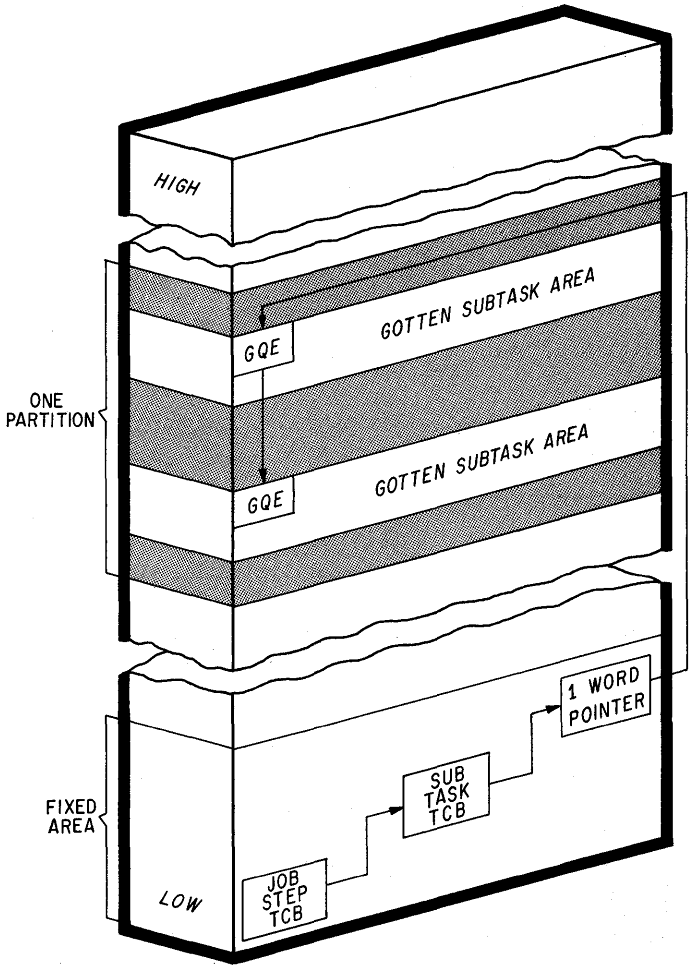


Figure 11. Storage Control for Subtask Storage (MFT With Subtasking)

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 8-11(B): 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 12.

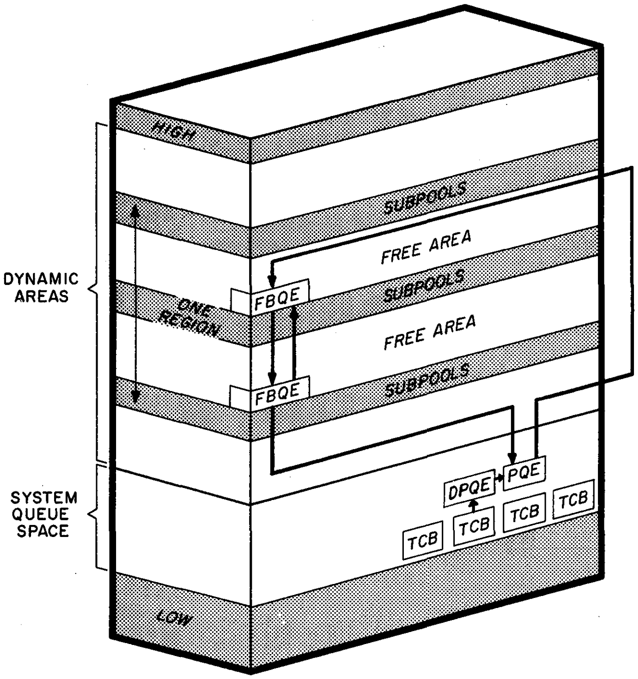


Figure 12. Storage Control for a Region (MVT)

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 anqther 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.

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.

Storage control for a subpool is summarized in Figure 13.

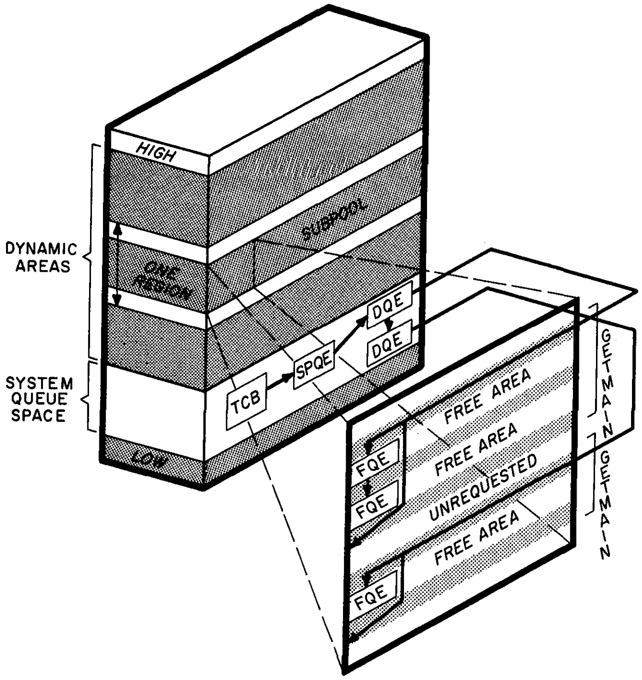
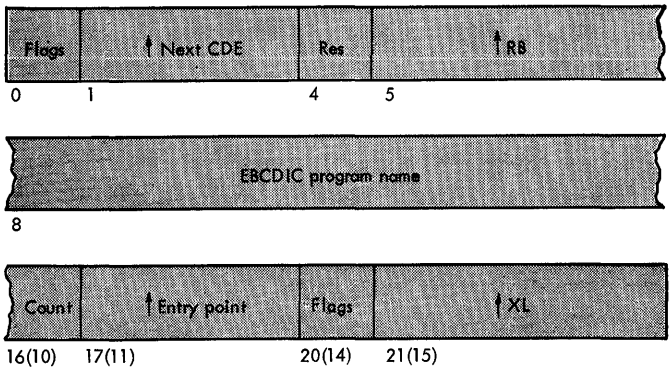


Figure 13. Storage Control for a Subpool (MVT)

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 contents directory queues reside in the system queue space. A CDE has the following format:



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.

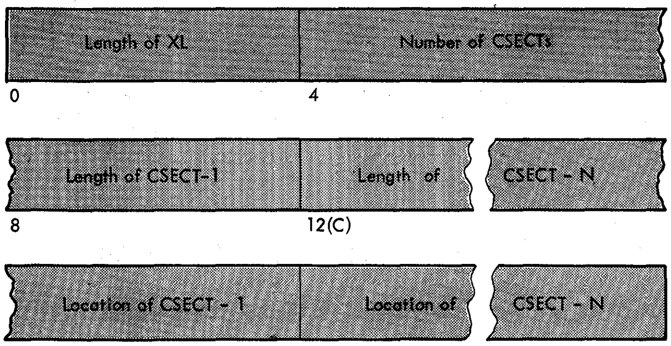
Byte 16(10): Use count.

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 14.

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 K.

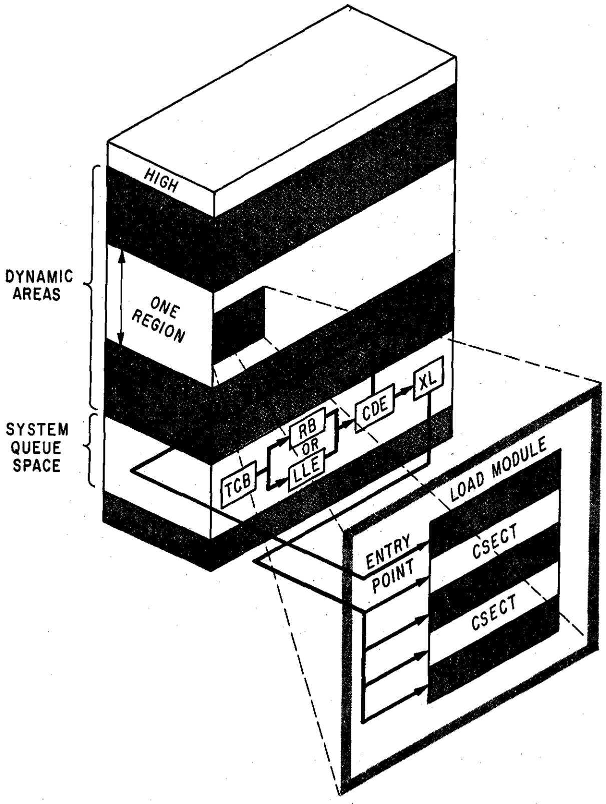


Figure 14. Storage Control for a Load Module (MVT)

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 locations 16(10) and 76(4c) contain 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 15, an expansion of Figure 1, shows the relationships among the principal control blocks and tables in the System/360 Operating System.

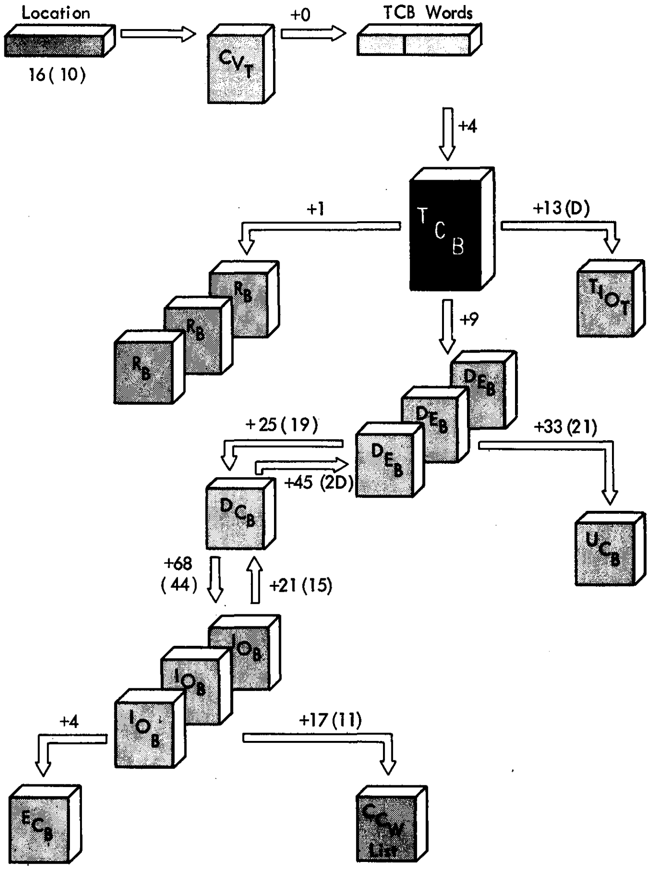


Figure 15. Control Block Relationships

Section 2: Interpreting Dumps

Topics composing Section 2 are:

- ABEND/SNAP dumps issued by systems with MFT.
- ABEND/SNAP dumps issued by systems with MVT.
- Indicative dumps.
- Storage dumps.

Each topic includes instructions for invoking the dump, a detailed description of the dump's contents, and a guide to using the dump.

ABEND/SNAP Dump (MFT)

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 dump will be provided, as explained in the section "Storage Dumps."

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 16 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 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 J.

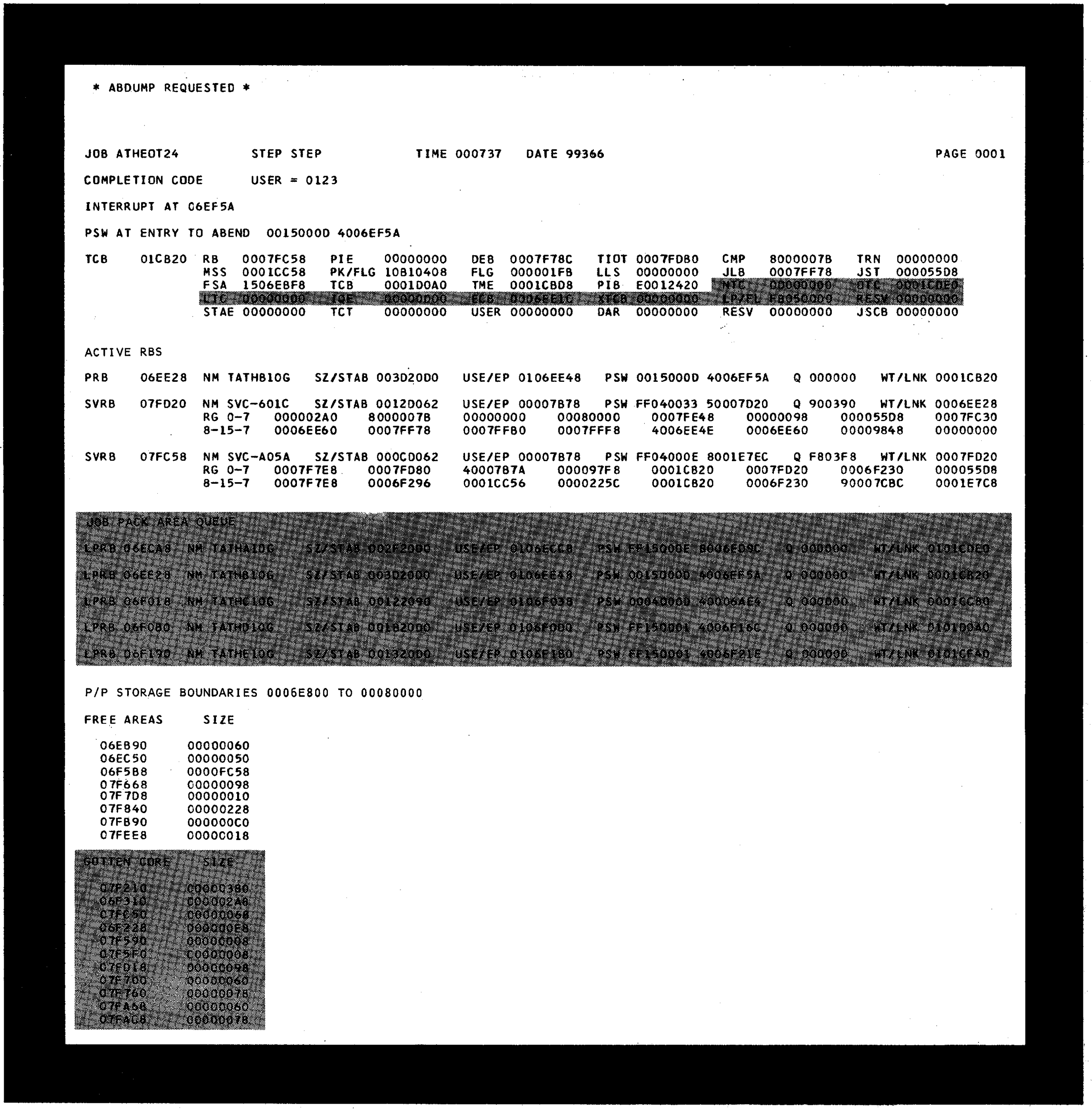
Invoking an ABEND/SNAP Dump (MFT)

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 dumps may be found in the section "Storage Dumps."

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.

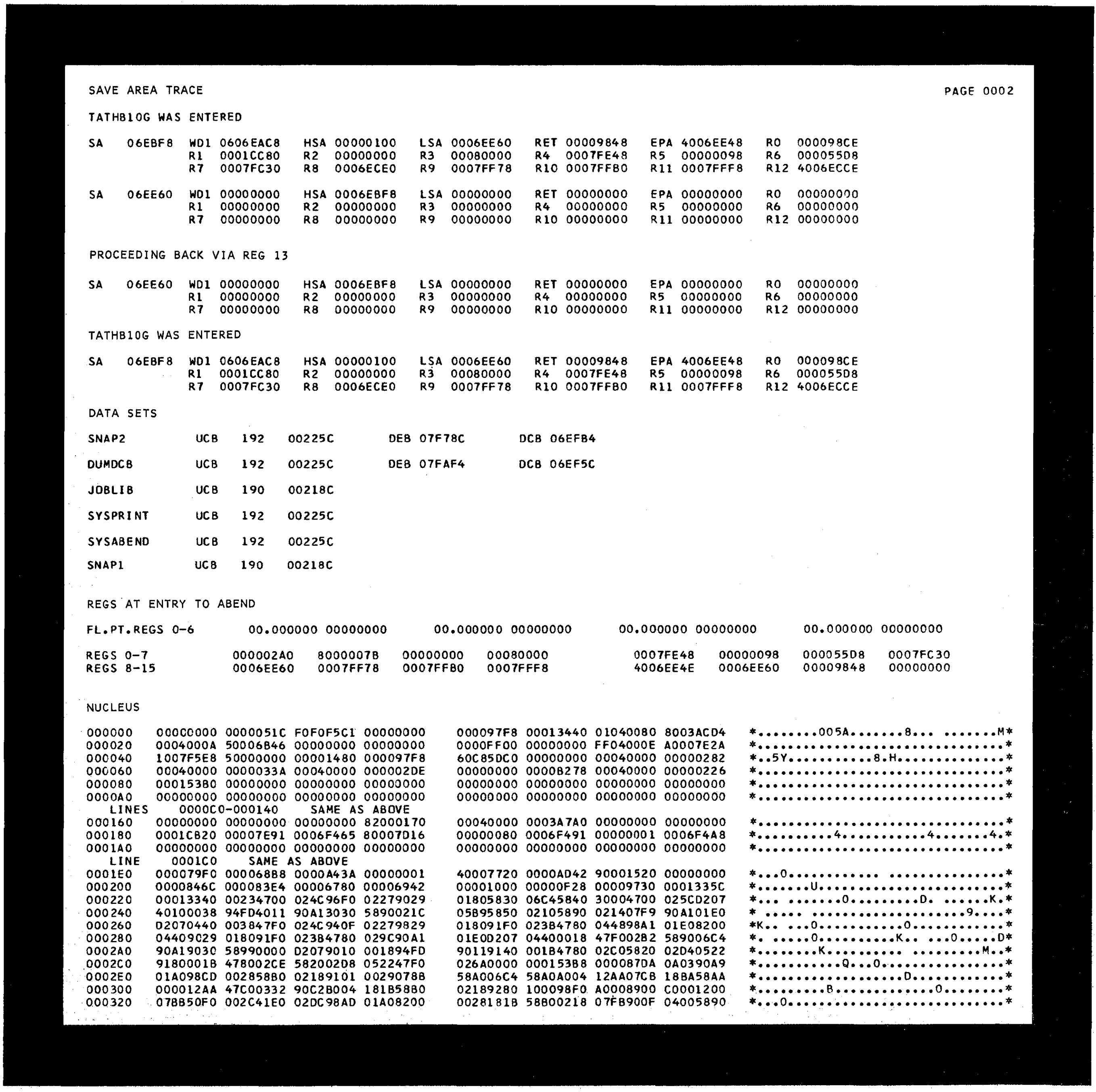
Main Storage Considerations: Three BSAM modules (IGG019BA, IGG019BB, and the device-dependent EOB module) are required to process dumps. These modules should be made resident in the Resident Access Method (RAM) area by specifying RESIDNT=ACSMETH in the SUPRVSOR macro instruction during system generation. If these modules are not resident, as much as 1352 bytes of main storage within the partition are required to contain them.

In addition to the area required for the BSAM modules, 2784 bytes must be available in the partition. 1344 of these bytes are required for EOV processing should the initial space specification for a direct access device be exceeded by the dump requirements.



```
 * 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  000001FB   LLS  00000000   JLB   0007FF78   JST  000055D8
                 FSA   1506EBF8   TCB      0001D0A0   TME  0001CBD8   PIB  E0012420   NTC   00000000   OTC  0001CDE0
                 LTC   00000000   IQE      00000D00   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      4000787A     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 06F080  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 0101CFA0

P/P STORAGE BOUNDARIES 0006E800 TO 00080000

FREE AREAS      SIZE

  06EB90     00000060
  06EC50     00000050
  06F5B8     0000FC58
  07F668     C0000098
  07F7D8     00000010
  07F840     00000228
  07FB90     000000C0
  07FEE8     00000018

GOTTEN CORE     SIZE

  07F210     C0000380
  06F310     000002A8
  07FC50     00000068
  06F228     000000E8
  07F590     00000008
  07F5F0     C0000008
  07FD18     00000098
  07F700     00000060
  07F760     00000078
  07FA68     00000060
  07FAC8     00000078
```

Figure 16. Sample of an ABEND Dump (MFT) (Part 1 of 2)



```
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    00000000 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    00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000    *................................*
   LINES     0000C0-000140     SAME AS ABOVE
000160    00000000 00000000 00000000 82000170    00040000 0003A7A0 00000000 00000000    *................................*
000180    0001CB20 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    000079F0 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    *... .......O.........D. ......K.*
000240    40100038 94FD4011 90A13030 5890021C    05B95850 02105890 021407F9 90A101E0    * .... .....................9....*
000260    D2070440 003847F0 024C940F 02279829    018091F0 023B4780 044898A1 01E08200    *K.. ...O...........O............*
000280    04409029 018091F0 023B4780 029C90A1    01E0D207 04400018 47F002B2 589006C4    *. .....O.........K.. ...O......D*
0002A0    90A19030 58990000 D2079010 001894FD    90119140 001B4780 02C05820 02D40522    *........K......... ..........M..*
0002C0    9180001B 478002CE 582002D8 052247F0    026A0000 000153B8 000087DA 0A0390A9    *...........Q...O................*
0002E0    01A098CD 002858B0 02189101 0029078B    58A006C4 58A0A004 12AA07CB 18BA58AA    *...................D............*
000300    000012AA 47C00332 90C2B004 181B5880    02189280 100098F0 A0008900 C0001200    *.........B.............O........*
000320    07B850F0 002C41E0 02DC98AD 01A08200    0028181B 58B00218 07FB900F 04005890    *...O............................*
```

Figure 16. Sample of an ABEND Dump (MFT) (Part 2 of 2)

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=(...
```

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.

Sample DD Statements: Figure 17 shows a set of job steps that include DD statements for ABEND dump data sets.

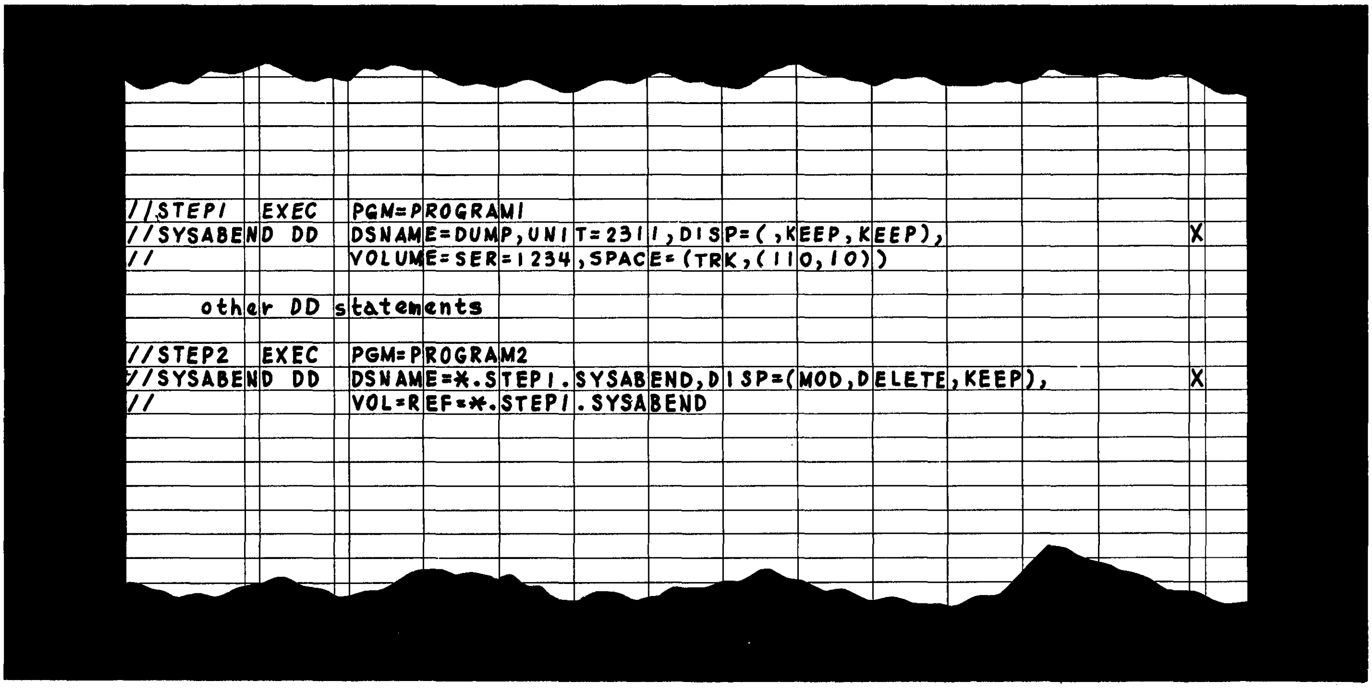
The SYSABEND DD statement in STEP2 takes advantage of the direct access space acquired in STEP1 by indicating MOD in the DISP parameter. Note that the space request in STEP1 is large so that the dumping operation is not inhibited due to insufficient space. The final SYSABEND DD statement in the job should indicate a disposition of DELETE to free the space acquired for dumping.

Contents of an ABEND/SNAP Dump (MFT)

This explanation of the contents of ABEND/SNAP dumps for systems with 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 17. 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
```

* * * ABDUMP REQUESTED * * *
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): number of resources on which the task is queued.

contains, in the last 2 digits, the TCBDSP field (byte 35):

- Reserved in 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 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): 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.

NTC hhhhhhhh
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
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
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
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
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
reserved for future use.

LP/FL hhhhhhhh
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
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):

Address of the Timing Control Table (TCT): zeros if 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);

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

<u>Note</u>: Three SVRBs for ABEND processing exist in the nucleus. They are used when there is insufficient space in the partition to create an SVRB.

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).

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.

NOT FORMATTED

indicates that the abnormal termination dump routine confiscated storage (indicated by *CORE NOT AVAILABLE, LOC. hhhhhh-hhhhhh TAKEN); because DCBs may have been overlaid, or that the dump is for an OLTEP task. 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 not 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. 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 (MFT)

Cause of Abnormal Termination: Evaluate the user (USER Decimal code) or system (SYSTEM=hex code) completion code using Appendix C 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 and task switching. 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.

A task switch entry is 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. Task switch entries are identified by a fifth digit of 'F'.

Notes: 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.

Dumps issued by systems with MFT contain only the last four characters of the module name in the RB APSW field. You cannot distinguish between IFG0xxxx and IGG0xxxx. After an SVC 19 has been issued, the OPEN where-to-go table should be checked for the module name.

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.

ABEND/SNAP Dump (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 18 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 <u>IBM System/360 Operating System: TCAM Program Logic Manual</u>, GY30-2029. References to other TCAM debugging aids are found in Appendix J.

<u>Invoking an ABEND/SNAP Dump (MVT)</u>

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 MFT. However, some additional considerations must be made in requesting main storage and direct access space.

<u>MVT Considerations</u>: 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.

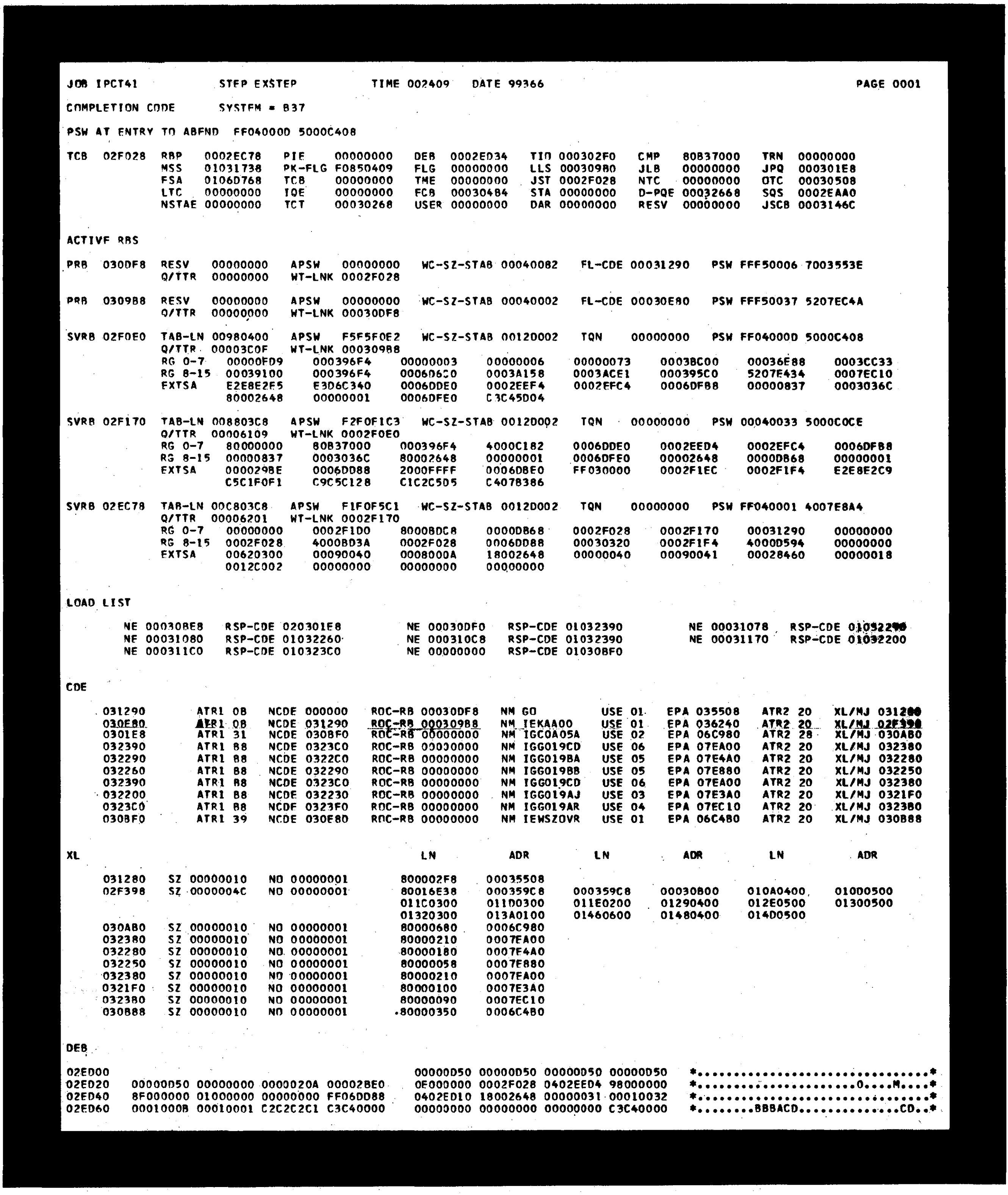
This formula gives the space requirements for one task. Request additional space if partial dumps of subtasks and invoking tasks will be included.

<u>Contents of an ABEND/SNAP Dump (MVT)</u>

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:

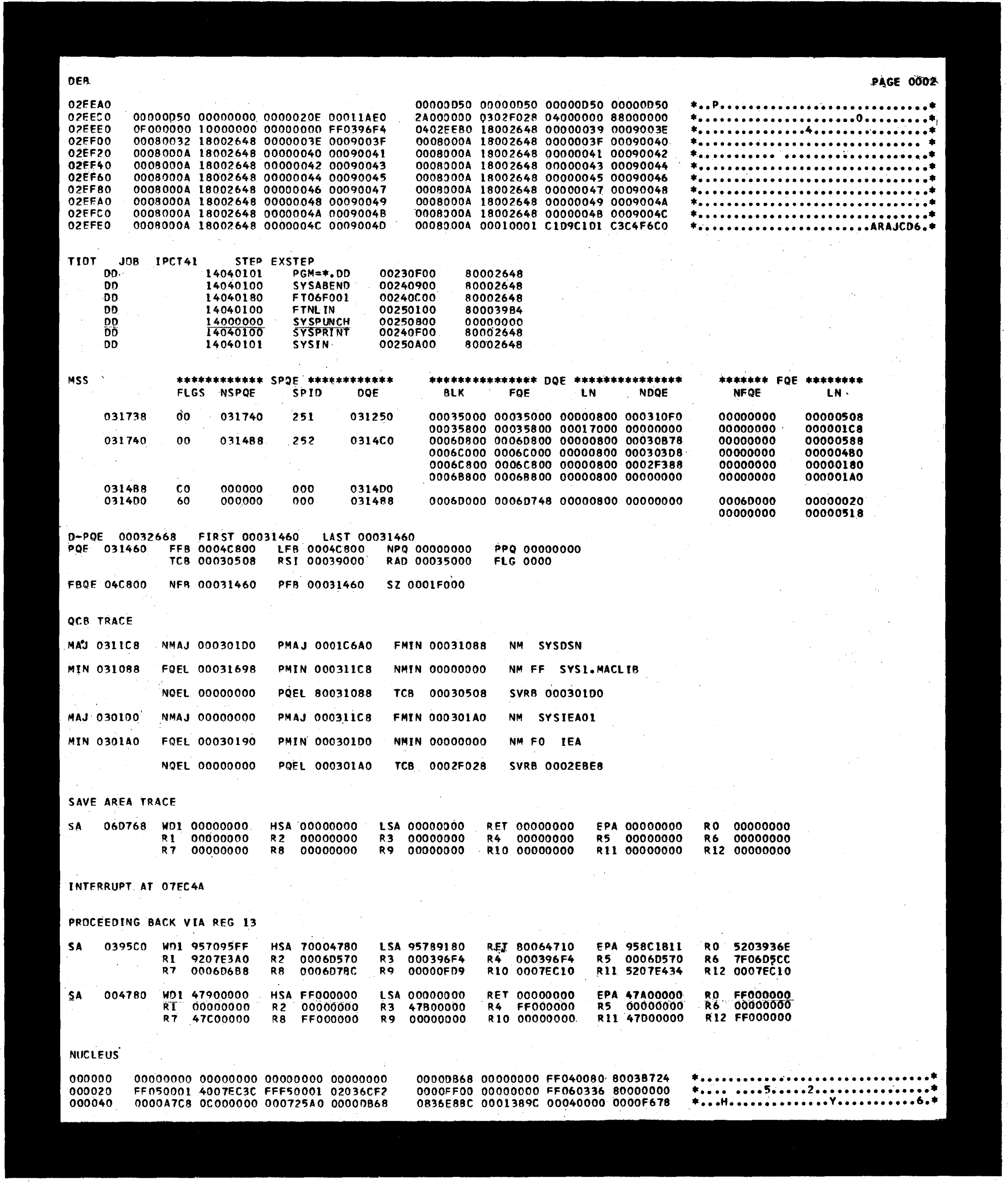
- h represents 1/2 byte of hexadecimal information
- d represents 1 byte of decimal information
- c represents a 1-byte character

You may prefer to follcw 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 00030980   JLB    00000000   JPQ  000301E8
             FSA    0106D768   TCB     00000000   TME  00000000   JST 0002F028   NTC    00000000   OTC  00030508
             LTC    00000000   IQE     00000000   FCB  00030484   STA 00000000   D-PQE 00032668   SQS  0002EAA0
             NSTAE 00000000    TCT     00030268   USER 00000000   DAR 00000000   RESV   00000000   JSCB 0003146C


ACTIVE RBS

PRB  030DF8  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 00030DF8

SVRB 02F0E0  TAB-LN 00980400   APSW    F5F5F0E2   WC-SZ-STAB 00120002   TQN     00000000   PSW FF04000D 5000C408
             Q/TTR  00003C0F   WT-LNK 000309B8
             RG 0-7    00000FD9    000396F4    00000003    00000006    00000073    0003BC00    00036E88    0003CC33
             RG 8-15   00039100    000396F4    0006D6C0    0003A158    0003ACE1    000395C0    5207E434    0007EC10
             EXTSA     E2E8E2E5    E3D6C340    0006DDE0    0002EEF4    0002EFC4    0006DFB8    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 0002F0E0
             RG 0-7    80000000    80B37000    000396F4    4000C182    0006DDE0    0002EED4    0002EFC4    0006DFB8
             RG 8-15   00000837    0003036C    80002648    00000001    0006DFE0    00002648    0000D868    00000001
             EXTSA     000029BE    0006DD88    2000FFFF    0006DBE0    FF030000    0002F1EC    0002F1F4    E2E8E2C9
                       C5C1F0F1    C9C5C128    C1C2C5D5    C4078386

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    0002F1D0    80008DC8    0000D868    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 00030BE8   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 08   NCDE 000000   ROC-RB 00030DF8   NM GO          USE 01   EPA 035508   ATR2 20   XL/MJ 031280
     030E80      ATR1 08   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 030AB0
     032390      ATR1 B8   NCDE 0323C0   ROC-RB 00000000   NM IGG019CD    USE 06   EPA 07EA00   ATR2 20   XL/MJ 032380
     032290      ATR1 B8   NCDE 0322C0   ROC-RB 00000000   NM IGG019BA    USE 05   EPA 07E4A0   ATR2 20   XL/MJ 032280
     032260      ATR1 B8   NCDE 032290   ROC-RB 00000000   NM IGG019BB    USE 05   EPA 07E880   ATR2 20   XL/MJ 032250
     032390      ATR1 B8   NCDE 0323C0   ROC-RB 00000000   NM IGG019CD    USE 06   EPA 07EA00   ATR2 20   XL/MJ 032380
     032200      ATR1 B8   NCDE 032230   ROC-RB 00000000   NM IGG019AJ    USE 03   EPA 07E3A0   ATR2 20   XL/MJ 0321F0
     0323C0      ATR1 B8   NCDF 0323F0   ROC-RB 00000000   NM IGG019AR    USE 04   EPA 07EC10   ATR2 20   XL/MJ 032380
     030BF0      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      80016E38    000359C8     000359C8    00030800     010A0400    010D0500
                                               011C0300    011D0300     011E0200    01290400     012E0500    01300500
                                               01320300    013A0100     01460600    01480400     014D0500
     030AB0    SZ 00000010    NO 00000001      80000680    0006C980
     032380    SZ 00000010    NO 00000001      80000210    0007EA00
     032280    SZ 00000010    NO 00000001      80000180    0007E4A0
     032250    SZ 00000010    NO 00000001      80000058    0007E880
     032380    SZ 00000010    NO 00000001      80000210    0007EA00
     0321F0    SZ 00000010    NO 00000001      80000100    0007E3A0
     032380    SZ 00000010    NO 00000001      80000090    0007EC10
     030B88    SZ 00000010    NO 00000001     .80000350    0006C480


DEB

02ED00                                            00000D50 00000D50 00000D50 00000D50    *................................*
02ED20    00000D50 00000000 0000020A 00002BE0     0E000000 0002F028 0402EED4 98000000    *........................O....M...*
02ED40    8F000000 01000000 00000000 FF06DD88     0402ED10 18002648 00000031 00010032    *................................*
02ED60    0001000B 00010001 C2C2C2C1 C3C40000     00000000 00000000 00000000 C3C40000    *........BBBACD..............CD..*
```

Figure 18. Sample of Complete ABEND Dump (MVT) (Part 1 of 2)



```
DEB                                                                                                          PAGE 0002

02EEA0                                                  00000D50 00000D50 00000D50 00000D50   *..P.............................*
02EEC0    00000D50 00000000 0000020E 00011AE0           2A000000 0302F028 04000000 88000000   *.....................0..........*
02EEE0    0F000000 10000000 00000000 FF0396F4           0402EE80 18002648 00000039 0009003E   *..............4.................*
02EF00    00080032 18002648 0000003E 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      00230F00      80002648
     DD              14040100      SYSABEND      00240900      80002648
     DD              14040180      FT06F001      00240C00      80002648
     DD              14040100      FTNLIN        00250100      800039B4
     DD              14000000      SYSPUNCH      00250800      00000000
     DD              14040100      SYSPRINT      00240F00      80002648
     DD              14040101      SYSIN         00250A00      80002648


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

     031738      00    031740     251     031250    00035000 00035000 00000800 000310F0      00000000     00000508
                                                    00035800 00035800 00017000 00000000      00000000     000001C8
     031740      00    031488     252     0314C0    0006D800 0006D800 00000800 00030B78      00000000     00000588
                                                    0006C000 0006C000 00000800 000303D8      00000000     00000480
                                                    0006C800 0006C800 00000800 0002F388      00000000     00000180
                                                    0006B800 0006B800 00000800 00000000      00000000     000001A0
     031488      C0    000000     000     0314D0
     0314D0      60    000000     000     031488    0006D000 0006D748 00000800 00000000      0006D000     00000020
                                                                                             00000000     00000518

D-PQE   00032668    FIRST 00031460    LAST 00031460
PQE   031460     FFB 0004C800    LFB 0004C800    NPQ 00000000    PPQ 00000000
                 TCB 00030508    RSI 00039000    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 FO   IEA

               NQEL 00000000     PQEL 000301A0     TCB  0002F028     SVRB 0002EBE8


SAVE AREA TRACE

SA    06D768   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 957095FF    HSA 70004780    LSA 95789180    RET 80064710    EPA 958C1811    R0  5203936E
               R1  9207E3A0    R2  0006D570    R3  000396F4    R4  000396F4    R5  0006D570    R6  7F06D5CC
               R7  0006D6B8    R8  0006D78C    R9  00000FD9    R10 0007EC10    R11 5207E434    R12 0007EC10

SA    004780   WD1 47900000    HSA FF000000    LSA 00000000    RET 00000000    EPA 47A00000    R0  FF000000
               R1  00000000    R2  00000000    R3  47B00000    R4  FF000000    R5  00000000    R6  00000000
               R7  47C00000    R8  FF000000    R9  00000000    R10 00000000    R11 47D00000    R12 FF000000


NUCLEUS

000000    00000000 00000000 00000000 00000000      0000DB68 00000000 FF040080 8003B724   *................................*
000020    FF050001 4007EC3C FFF50001 02036CF2      0000FF00 00000000 FF060336 80000000   *.... ....5.....2................*
000040    0000A7C8 0C000000 000725A0 0000DB68      0836E88C 0001389C 00040000 0000F678   *...H..............Y...........6.*
```

Figure 18. Sample of Complete ABEND Dump (MVT) (Part 2 of 2)

```
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. The contents of the TCBDEB field may differ in the main storage printout from what appears in the TCBDEB field of the formatted section. This occurs when the number of extents specified in the DEB for the dump data set is not sufficient to complete ABDUMP processing. When the dump of main storage is given, the END OF VOLUME routine may have built another DEB having additional extents for the dump data set and dequeued the original DEB. Therefore, the TCBDEB field in the main storage printout may contain the address of the new DEB built by END OF VOLUME.

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).

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.

Note: DEBs are not formatted if the dump is for an OLTEP task. If a dump of the DEB chain is desired, use a SYSABEND DD card so that the nucleus will be dumped.

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 hhhhhhh
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 F0, the name is known only to the job step; in this case, the scope is the protection key of the TCB enqueuing the mincr 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 cnly 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 available, due either to omission or incorrect specification of a SYSABEND or SYSUDUMP DD statement. An indicative dump is issued automatically on the system output (SYSOUT) device.

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 19 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:

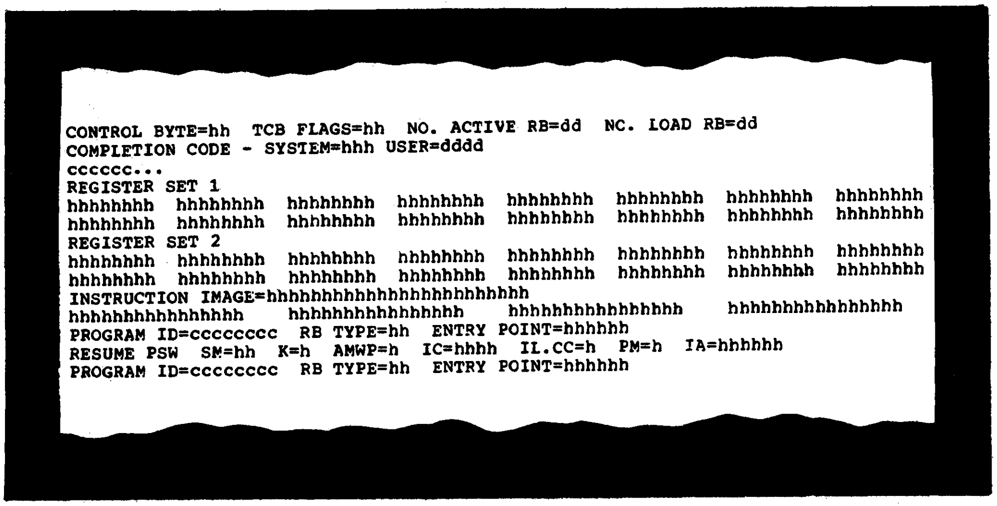
Bit	Setting	Meaning
0	0	Instruction image not present
	1	Instruction image present
1	0	Floating-point registers not present
	1	Floating-point registers present
2	0	One general register set present
	1	Two general register sets present
3	0	All active RBs present
	1	All active RBs not present

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 19. 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 C 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 Dumps

Storage dumps record the contents of main storage from location 00 to the end of printable storage.

Storage dumps are produced by the damage assessment routine (DAR) or other system recovery routines, the Console Dump facility, or the stand-alone service aid program IMDSADMP.

DAMAGE ASSESSMENT ROUTINE (DAR)

The damage assessement routine produces a storage dump when a system task fails and is designed to provide increased system availability in the event of system failure. The storage dump is written to the SYS1.DUMP data set.

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.

Note: If TSO is installed in the system and a failure occurs in the TSO subsystem or in the operating system the TSO SWAP data set must be recorded for use in diagnosis if needed. The system recovery routines do not do this. The IMDPRDMP service aid can be used as a high-performance dumping program for this purpose by directing its output to tape. Refer to the Service Aids publication for details of this usage of the IMDPRDMP program.

CONSOLE DUMP

The Console Dump function is designed to meet the requirements for a dynamic main storage dumping tool in the operating system. The operator initiates the Console Dump from the primary console via a DUMP command. Execution of the function allows a dump to be taken to the SYS1.DUMP data set of all or selective portions of main storage. The dump operation is performed during system operation and requires no IPL. The storage dump may then be formatted and printed by the IMDPRDMP Service Aid program. Refer to the Operator's Guide publication for details of the DUMP command.

IMDSADMP SERVICE AID

In situations where the system is not operative, an IMDSADMP program is loaded into storage through use of the IPL facilities. The storage dump taken may be written in a high-speed version to tape or disk, and in a low speed version to tape or printer. The high-speed IMDSADMP dump must be processed by the IMDPRDMP program. The low-speed tape output may be processed by a program such as the IEBGENER utility program. The format of the low-speed IMDSADMP output is similar to the general format listing produced by the IMDPRDMP program and therefore is not illustrated in this publication. A sample IMDSADMP listing and a discussion of the program are contained in the Service Aids publication.

SYSTEM FAILURE

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

MFT:

Communications Task
Master Scheduler Task
Log Task

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

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 dump. If additional failures occur, and if the SYS1.DUMP data set is occupied, DAR does not attempt to write another storage image dump.

Use the IMDPRDMP service aid to format and list the SYS1.DUMP data set.

IMDPRDMP Output

Main storage information processed by the IMDPRDMP program is presented in six different output formats. The output format used is determined by the function of the particular area of the dumped system's main storage that is being printed. Two of these formats, the queue control block trace and the link pack area map, are invoked by specific format statements. A third format is used to print the major system control blocks. Two formats are used for TSO; one for system control blocks and the other for user control blocks. Any areas of the dumped system's main storage that do not fall into any of the aforementioned functional categories are processed in the general format.

Dump List Headings: Each page of output listing contains a heading. This heading has the optional user specified title, the name of the module that invoked the dump, the date and the time the dump was taken except when processing Generalized Trace Facility output when the heading will be "EXTERNAL TRACE - DD ddname." Note: If the dump was produced by IMDSADMP on a system with the time-of-day (TOD) clock, IMDPRDMP can not determine the time at which the dump was taken; the time is replaced by "TOD CLK."

Dump Header: If the dump was produced by SVC DUMP, IMDPRDMP will print the title taken from the dump header record. A maximum of 100 characters are printed on the second line of the first page of the output listing.

Output Comments: While formatting the dump, the IMDPRDMP program occasionally is unable to locate, format and print a control block. On those occasions IMDPRDMP prints a comment explaining why the control block could not be formatted and printed. These comments are printed within the body of the formatted dump and are part of the IMDPRDMP output. A complete list of these output comments along with further explanations is contained at the end of this chapter.

Summary Information: In addition to formats, the following summary information is printed at the end of each execution of IMDPRDMP:

- The number of entries to the read routine;
- The number of times that the required address was not found in a buffer;
- The number of blocks read from the dump data set;
- The number of permanent I/O errors encountered during the execution;
- The average number of buffers used for each operation performed during this execution;
- The number of blocks read from the TSO swap data sets;
- The ratio of the number of times the read routine was called to the number of times the requested address was not in a buffer.
- When processing Generalized Trace Facility output, the number of trace records processed.

QUEUE CONTROL BLOCK TRACES

In a multiprogramming environment, requests for system resources are enqueued. This process is accomplished through the use of queue control blocks (QCBs).

Certain system failures, such as task contention deadlocks, become evident to the user upon examination of a queue control block trace. When requested through the use of the QCBTRACE statement, the QCB trace appears on a separate page of the IMDPRDMP program dump listing. The trace, a sample of which appears in Figure 20, contains a listing of all queue control blocks that were present in the dumped system, and is available to users who are processing main storage information gathered from an MVT or MFT system.

(For more information on system resource queuing, see IBM System/360 Operating System: MVT Supervisor, GY28-6659.)

The page of the IMDPRDMP listing containing the Queue Control Block trace is identified by two heading lines. The first line contains an optional title, the name of the module that invoked the dump, and the date and time that the information was gathered from the dumped system. The second line of the heading identifies the page as containing a Queue Control Block trace. The individual QCBs are then listed for each Task Control Block. Each Queue Control Block is formatted as follows:

MAJOR hhhhhh
    The starting address of a major queue control block, the contents of which are given, indented, on the line or lines below.

NAME cccccccc
    The name of the system resource represented by the major QCB.

```
   SAMPLE QCB TRACE    MODULE IMDSADMP    DATE   7/04/70    TIME    0.10    PAGE    2

* * * *   Q U E U E   C O N T R O L   B L O C K   T R A C E   * * * *


MAJOR 024100   NAME SYSDN

      MINOR 0239A0        NAME FF  SYS1.LINKLIB
            QEL 024068      TCB 023488   SHARED

      MINOR 023838        NAME FF  SYS1.MACLIB
            QEL 023ED8      TCB 023448   SHARED


MAJOR 0235E8   NAME SYSIEFSD

      MINOR 0235C8   NAME FF  Q5
            QEL 023208      TCB 023480   EXCLUSIVE
            QEL 023C10      TCB 0238E0   EXCLUSIVE
```

Figure 20. Queue Control Block Trace Sample

MINOR hhhhhh
    The starting address of the minor queue control block. Contents are given on this line or the lines below.

NAME hh cccccccc
    The first two characters appearing after the NAME field identifier indicate the scope of the minor QCB being dumped. If the scope is given as hexadecimal FF, the name of the QCB is known to the entire operating system. If the scope indicator is hexadecimal 00 or 10 through F0, the name of the QCB is known only to the job step. The scope indicator shows the storage protection key of the TCB that enqueued this minor QCB. The NAME field also contains the name of the specific system resource represented by the minor QCB.

QEL hhhhhh
    The address of a queue element (QEL) associated with the minor QCB described on the line above. A QEL line appears for each resource requested by the task associated with the minor QCB.

TCB hhhhhh
    The starting address of the task control block of the requesting task. This task requests a specific system resource through the use of the QEL indicated on this line.

SHARED or EXCLUSIVE
    This indicator tells whether the system resource is available to one task (EXCLUSIVE) or several tasks (SHARED).

LINK PACK AREA MAPS

Information on routines residing in either the MVT link pack area or the MFT resident reenterable load module area of the dump system is available to the user through use of the LPAMAP (link pack area map) format statement.

For users who are processing an MVT dump, the IMDPRDMP program produces a listing of all routines loaded into the link pack area by the nucleus initialization program (NIP). For MFT dumps, this list contains information pertaining to all resident reentrant routines loaded into the reenterable load module area by NIP.

The IMDPRDMP user will find the link pack area map, for MVT, or the reenterable load module area map, for MFT, to be a useful tool in isolating system failures that occurred in program modules that reside outside the user's partition or region. If requested, the applicable map appears on a separate page of the IMDPRDMP program dump listing. A sample Link Pack Area map is shown in Figure 21 .

The dump listing page containing the link pack area map is identified by two heading lines. The first line contains the optional title supplied by the user, the name of the module that invoked the dump, and the date and time that the information was gathered from the dumped system. The second line of the heading identifies the page as containing a link pack area map. Information on each module contained in the link pack area or reenterable load module area, is given in the following format:

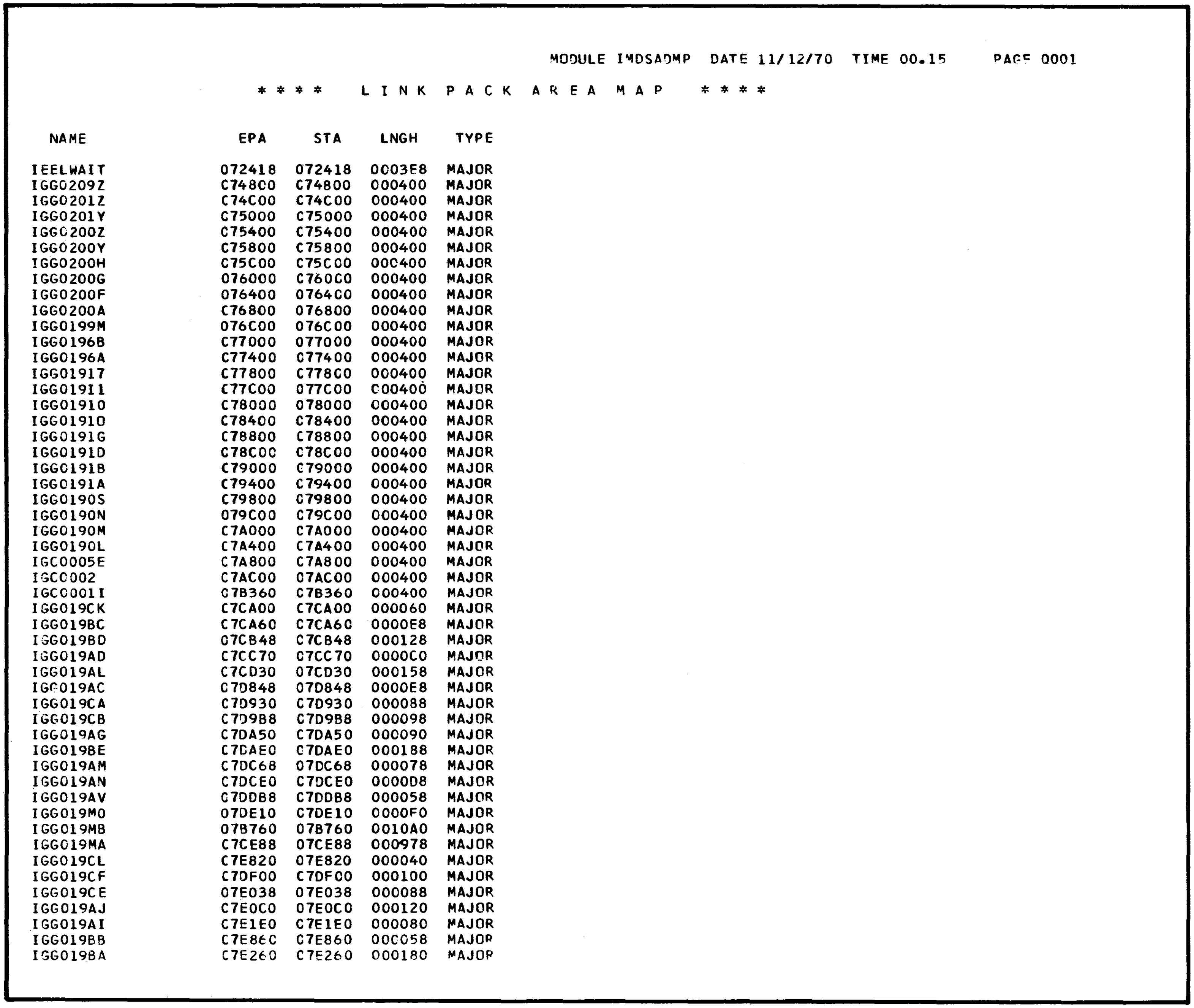
NAME cccccccc
The name of the load module represented by this entry.

EPA hhhhhh
The entry point address of the module identified on the corresponding line in the NAME column.

STA hhhhhh
The starting address of the named module's control section.

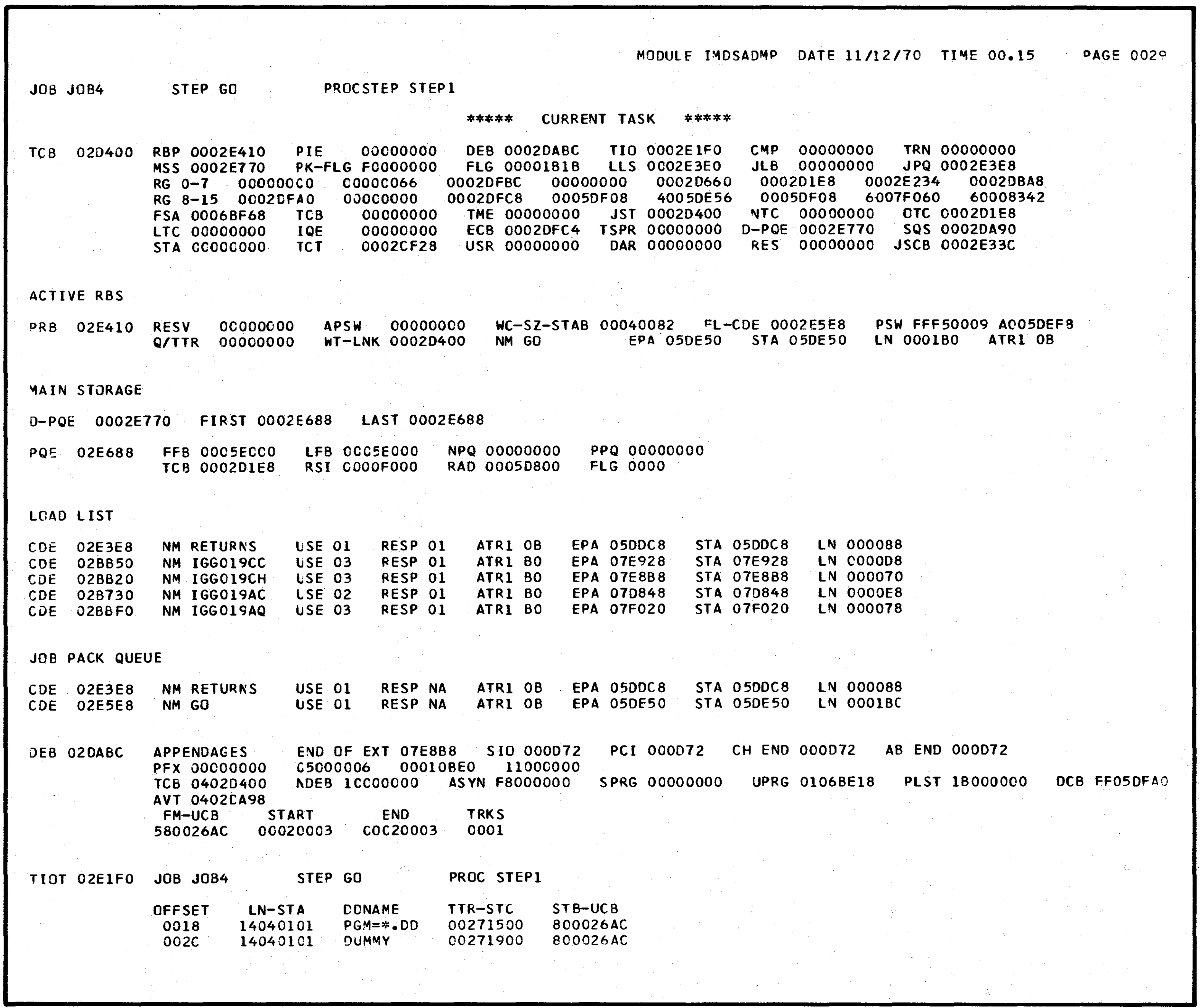
LNGH hhhhhh
The length, in bytes, of the control sections in the load module described on this line.

TYPE ccccc
The attributes of the control block associated with the module being described on this line. Under MVT, the type of the contents directory entry (CDE) associated with the module is given. The type may be either MAJOR or MINOR. Under MFT, the type is shown as either a loaded request block (LRB) or a loaded program request block (LPRB).



```
                                                        MODULE IMDSADMP  DATE 11/12/70  TIME 00.15      PAGE 0001

                         * * * *    L I N K   P A C K   A R E A   M A P    * * * *


  NAME                EPA      STA      LNGH     TYPE

IEELWAIT             072418   072418   0003E8   MAJOR
IGG0209Z             C74800   C74800   000400   MAJOR
IGG0201Z             C74C00   C74C00   000400   MAJOR
IGG0201Y             C75000   C75000   000400   MAJOR
IGGC200Z             C75400   C75400   000400   MAJOR
IGG0200Y             C75800   C75800   000400   MAJOR
IGG0200H             C75C00   C75C00   000400   MAJOR
IGG0200G             076000   C760C0   000400   MAJOR
IGG0200F             076400   0764C0   000400   MAJOR
IGG0200A             C76800   076800   000400   MAJOR
IGG0199M             076C00   076C00   000400   MAJOR
IGG0196B             C77000   077000   000400   MAJOR
IGG0196A             C77400   C77400   000400   MAJOR
IGG01917             C77800   C778C0   000400   MAJOR
IGG0191I             C77C00   077C00   C00400   MAJOR
IGG01910             C78000   078000   000400   MAJOR
IGG0191O             C78400   C78400   000400   MAJOR
IGG0191G             C78800   C78800   000400   MAJOR
IGG0191D             C78C0C   C78C00   000400   MAJOR
IGGC191B             C79000   C79000   000400   MAJOR
IGGC191A             C79400   C79400   000400   MAJOR
IGG0190S             C79800   C79800   000400   MAJOR
IGG0190N             079C00   C79C00   000400   MAJOR
IGG0190M             C7A000   C7A000   000400   MAJOR
IGG0190L             C7A400   C7A400   000400   MAJOR
IGC0005E             C7A800   C7A800   000400   MAJOR
IGCC002              C7AC00   07AC00   000400   MAJOR
IGCC001I             C7B360   C7B360   C00400   MAJOR
IGG019CK             C7CA00   C7CA00   000060   MAJOR
IGG019BC             C7CA60   C7CA6C   0000E8   MAJOR
IGG019BD             07CB48   C7CB48   000128   MAJOR
IGG019AD             C7CC70   C7CC70   0000C0   MAJOR
IGG019AL             C7CD30   07CD30   000158   MAJOR
IGG019AC             C7D848   07D848   0000E8   MAJOR
IGG019CA             C7D930   C7D930   000088   MAJOR
IGG019CB             C7D9B8   C7D9B8   000098   MAJOR
IGG019AG             C7DA50   C7DA50   000090   MAJOR
IGG019BE             C7DAE0   C7DAE0   000188   MAJOR
IGG019AM             C7DC68   07DC68   000078   MAJOR
IGG019AN             C7DCE0   C7DCE0   0000D8   MAJOR
IGG019AV             C7DDB8   C7DDB8   000058   MAJOR
IGG019MO             07DE10   C7DE10   0000F0   MAJOR
IGG019MB             07B760   07B760   0010A0   MAJOR
IGG019MA             C7CE88   07CE88   000978   MAJOR
IGG019CL             C7E820   07E820   000040   MAJOR
IGG019CF             C7DF00   C7DF00   000100   MAJOR
IGG019CE             07E038   07E038   000088   MAJOR
IGG019AJ             C7E0C0   07E0C0   000120   MAJOR
IGG019AI             C7E1E0   C7E1E0   000080   MAJOR
IGG019BB             C7E86C   C7E860   00C058   MAJOR
IGG019BA             C7E260   C7E260   000180   MAJOR
```

Figure 21. Link Pack Area Map Sample



```
                                                   MODULE IMDSADMP  DATE 11/12/70  TIME 00.15    PAGE 0029

JOB JOB4       STEP GO         PROCSTEP STEP1

                                        *****   CURRENT TASK   *****

TCB   02D400  RBP 0002E410   PIE     00000000   DEB 0002DABC   TIO 0002E1F0   CMP  00000000   TRN 00000000
              MSS 0002E770   PK-FLG F0000000   FLG 00001B1B   LLS 0002E3E0   JLB  00000000   JPQ 0002E3E8
              RG 0-7   000000C0   C0000066   0002DFBC   00000000   0002D660   0002D1E8   0002E234   0002DBA8
              RG 8-15  0002DFA0   00000000   0002DFC8   0005DF08   4005DE56   0005DF08   6007F060   60008342
              FSA 0006BF68   TCB     00000000   TME 00000000   JST 0002D400   NTC  00000000   OTC 0002D1E8
              LTC 00000000   IQE     00000000   ECB 0002DFC4  TSPR 00000000  D-PQE 0002E770   SQS 0002DA90
              STA 00000000   TCT     0002CF28   USR 00000000   DAR 00000000   RES  00000000  JSCB 0002E33C


ACTIVE RBS

PRB   02E410  RESV    00000000    APSW    00000000    WC-SZ-STAB 00040082    FL-CDE 0002E5E8    PSW FFF50009 A005DEF8
              Q/TTR   00000000    WT-LNK 0002D400     NM GO          EPA 05DE50    STA 05DE50    LN 0001B0    ATR1 0B


MAIN STORAGE

D-PQE   0002E770    FIRST 0002E688    LAST 0002E688

PQE   02E688    FFB 0005EC00    LFB 0005E000    NPQ 00000000    PPQ 00000000
                TCB 0002D1E8    RSI C000F000    RAD 0005D800    FLG 0000


LOAD LIST

CDE   02E3E8    NM RETURNS     USE 01    RESP 01    ATR1 0B    EPA 05DDC8    STA 05DDC8    LN 000088
CDE   02BB50    NM IGG019CC    USE 03    RESP 01    ATR1 B0    EPA 07E928    STA 07E928    LN 0000D8
CDE   02BB20    NM IGG019CH    USE 03    RESP 01    ATR1 B0    EPA 07E8B8    STA 07E8B8    LN 000070
CDE   02B730    NM IGG019AC    USE 02    RESP 01    ATR1 B0    EPA 07D848    STA 07D848    LN 0000E8
CDE   02BBF0    NM IGG019AQ    USE 03    RESP 01    ATR1 B0    EPA 07F020    STA 07F020    LN 000078


JOB PACK QUEUE

CDE   02E3E8    NM RETURNS     USE 01    RESP NA    ATR1 0B    EPA 05DDC8    STA 05DDC8    LN 000088
CDE   02E5E8    NM GO          USE 01    RESP NA    ATR1 0B    EPA 05DE50    STA 05DE50    LN 0001BC


DEB 02DABC    APPENDAGES      END OF EXT 07E8B8    SIO 000D72    PCI 000D72    CH END 000D72    AB END 000D72
              PFX 00000000    C5000006    00010BE0    11000000
              TCB 0402D400    NDEB 1C000000    ASYN F8000000    SPRG 00000000    UPRG 0106BE18    PLST 1B000000    DCB FF05DFA0
              AVT 0402CA98
               FM-UCB      START        END       TRKS
              580026AC    00020003    C0C20003    0001


TIOT 02E1F0   JOB JOB4        STEP GO          PROC STEP1

              OFFSET    LN-STA     DDNAME      TTR-STC     STB-UCB
               0018    14040101    PGM=*.DD    00271500    800026AC
               002C    14040101    DUMMY       00271900    800026AC
```

Figure 22. Sample of MVT Major Control Block Format

MAJOR SYSTEM CONTROL BLOCK FORMATS

Formatting of the major system control blocks associated with a task is a function of either a FORMAT control statement, or one of the several noted parameters associated with the PRINT control statement. The control blocks of several tasks may be printed during one execution of IMDPRDMP. When more than one task is printed, the associated task control blocks (TCBs) are grouped into a TCB summary, listed following the printing of all requested tasks. This summary provides an index to the formatted TCBs by jobname. See the discussion "Task Control Block Summaries."

For ease of identifying various dump printouts, specific headings are printed on each dump; such as FORMAT, DAR AND F03 TASKS, PRINT CURRENT, and PRINT JOBNAME.

Each task being printed begins on a new page, identified by two heading lines. The first heading line contains the optional title supplied by the user, the name of the module that invoked the dump, and the date and time that the information was gathered from the dumped system, and a page number. The second line of the heading identifies the particular task being printed. This task information is broken down into the following named fields:

JOB cccccccc
: The JOB field displays the eight-character name that was specified in the label field of the JOB statement.

STEP cccccccc
: The STEP field shows the eight-character step name of the problem program associated with the task being dumped. This name was supplied in the label field of the EXEC statement.

PROCSTEP cccccccc
: If the job step being displayed was invoked from a cataloged procedure, the step name of the cataloged procedure, as contained in the cataloged procedure's EXEC statement, is displayed in this field.

If the task being printed was in control at the time the dump was taken, a third heading line follows the two previously described. The line "**** CURRENT TASK ****" identifies the TCB associated with the task in control when the dump was taken.

While formatting the dumped control blocks, IMDPRDMP may issue various output comments to assist the person who analyzes the printout. The output comments are discussed following the control block discussion.

Specific formatting of the major system control blocks is dependent upon the operating system option under which the dumped system was operating. To allow the reader to concentrate on the particular operating system with which he is concerned, the discussion of control block formatting is divided into three parts: MVT, MFT, and the TSO option of MVT.

MVT Control Block Formatting

The formats described below are repeated for each requested task that is printed. A sample of the major system control blocks, as formatted from an MVT dump, is shown in Figure 22.

MVT TASK CONTROL BLOCK (TCB) FORMATTING: The task control block (TCB) contains information that pertains to the specific task named in the heading lines that appear at the top of the page. Each TCB is formatted as follows:

TCB hhhhhh
: The address of the task control block being displayed is given in this first field.

RBP hhhhhhhh
: The address of the request block (RB) that was currently associated with the task represented by this TCB.

PIE hhhhhhhh
: The address of the first program interrupt element (PIE) enqueued by this TCB.

DEB hhhhhhhh
: The address of the beginning of the data extent block (DEB) queue that was associated with this task. Information on the contents of each DEB in the queue is given in a separate portion of this MVT dump listing.

TIO hhhhhhhh

The address of the task input output table (TIOT) that was constructed during device allocation for the task represented by this TCB. The contents of this table are displayed in a later portion of this task's display.

CMP hhhhhhh

This word contains ABEND indicators and user and system completion codes. The usage of this field is as follows:

byte 0

1... .... Bit 0 indicates that a dump had been requested.

.1.. .... Bit 1 set indicates that a step ABEND had been requested.

..xx xxxx Bits 2 through 7 are reserved for future use.

bytes 1-3

The first 12 bits contain a system completion code. These codes and their meanings are explained in the publication IBM System/360 Operating System: Messages and Codes, GC28-6631 under the heading "System Completion Codes." A user completion code is contained in the last 12 bits.

TRN hhhhhhhh

Contains flags and TESTRAN indicators as follows:

byte 0

1... .... Bit 0 set indicates that both TESTRAN and decimal simulator programs were being used on a System/360 Model 91 machine.

.1.. .... Bit 1 set indicates that checkpoints were not taken for this step.

..1. .... Bit 2 set indicates that the TCB being displayed belonged to either a graphics foreground or the graphic job processor.

...1 .... Bit 3 set indicates that the TCB being displayed was associated with a 7094 emulator task that was being run on a System/360 Model 85 machine.

.... x... Bit 4 is reserved for future use.

.... .1.. Bit 5 set indicates that this is a time shared task under control of the TEST command processor.

.... ..1. Bit 6 set indicates that the OLTEP functions require cleanup before abnormal termination can be invoked.

....x Bit 7 is reserved for future use.

bytes 1-3

The address of the control core table that was used by TESTRAN.

MSS hhhhhhhh

Main storage supervision information as follows:

byte 0

This byte determined roll-out eligibility for the job step associated with this TCB.

00 in this byte indicated that the job step may be rolled out.

nz (nonzero) in this byte indicated that the job step may not be rolled out.

bytes 1-3

These bytes contain the starting address of the last subpool queue element (SPQE).

PK-FLG hhhhhhhh

The storage protection key of the task and a series of flags. This word is divided into several subfields. These are:

byte 0

xxxx .... The storage protection key of the task represented by this TCB.

.... 0000 Always contain zeros.

byte 1

1... .... Bit 0 set indicates thatan abnormal termination was in progress at the time the dump was taken.

.1.. .... Bit 1 set indicates that a normal termination was in progress at the time the dump was taken.

..1. .... Bit 2 set causes the Erase routine in ABEND to enter when ABEND is in control again.

...1 .... Bit 3 set causes the Purge routine in ABEND to enter when ABEND is in control again.

.... 1... Bit 4 set indicates that the Graphics Abnormal Termination routine was in control of the task associated with this TCB at the time the dump was taken. Bit 7 in byte 3 of this word must also be on.

.... .1.. Bit 5 set indicates that the top task in the TCB chain (usually the job step TCB) was in the process of being terminated when the dump was taken.

.... ..1. Bit 6 set indicates that an abnormal dump has been completed.

....1 Bit 7 indicates that asynchronous exits could not be scheduled.

byte 2

1... .... Bit 0 set indicates that the SYSABEND (or SYSUDUMP) data set for the job step is being opened. Operands of ABEND macro instruction have been saved in TCBCMP field.

.1.. .... Bit 1 set indicates that if this is an initiator TCB, the second job step interval has expired.

..1. .... Bit 2 set indicates that for a job step TCB, the job step can cause rollout.

...1 .... Bit 3 set indicates that the current task had a forced completion imposed upon it. Other tasks in the system could not have been performed until the current task had been completed.

.... 1... Bit 4 set indicates that the job step had a forced completion imposed upon it. Other tasks in the job step could not have been performed until the present job step had been completed.

.... .1.. Bit 5 set indicates that the SYSABEND (or SYSUDUMP) data set has been opened for the job step.

.... ..1. Bit 6 set indicates that an EXTR exit was requested by an attaching task.

....1 Bit 7 set indicates that the task associated with this TCB was a member of a time-sliced group.

byte 3

1... .... Bit 0 set indicates that a PSW associated with the task represented by this TCB was in the supervisor state.

.1.. .... Bit 1 set is applicable to job step TCBs. Setting of this bit indicates that the job step had invoked rollouts that were still in effect at the time the dump was taken.

..1. .... Bit 2 set indicates that ABEND was processing in such a manner as to prevent multiple ABENDS from occurring in the dumped system.

...1 ...x Bit 3 set indicates that the SYSABEND (or SYSUDUMP) data set is being opened by this task. (See also bit 7.)

.... 1..x Bit 4 set indicates that an ABDUMP was in process for the task associated with this TCB at the time the dump was taken. (See bit 7 of this byte.)

.... .1.. Bit 5 set is applicable only for job step TCBs. With this bit set, no abnormal termination dumps could have been provided within the job step represented by this TCB.

.... ..1x Bit 6 set indicates that a CLOSE had been issued during ABEND processing. (See bit 7 of this byte.)

...x x.x1 Bit 7 set, in conjunction with bits 3, 4, or 6 of this byte or bit 4 in byte 1 of this word indicates that, had the dumped system been allowed to continue processing without interruption by the IMDSADMP dump program, a valid reentry to ABEND would have been effected.

FLG hhhhhhhh

This field displays a further series of flags and certain priority indicators. This word is divided into subfields as follows:

byte 0
If any one of the flags comprising this byte were set at the time the dump was taken, the task represented by this TCB was considered to be non-dispatchable.

1... .... Bit 0 was set by ABDUMP

.x.. .... Bit 1 is reserved for future use.

..1. .... Bit 2 set indicates that the supply of I/O request queue elements (RQEs) had been exhausted.

...x xx.. Bits 3 through 5 are reserved for future use.

.... ..1. Bit 6 is applicable only to M65 multiprocessing situations. The setting of this bit indicates that the task represented by this TCB had been flagged non-dispatchable by one CPU to prevent any CPU from working on it.

....1 Bit 7 set indicates that the task associated with this TCB entered the ABEND routine while the data control block representing the SYSABEND data set was being opened for another task.

byte 1
If any one of the flags comprising this byte were set at the time the dump was taken, the task represented by this TCB was considered to be non-dispatchable.

1... .... Bit 0 set indicates that the task represented by this TCB was terminated prior to the time the dump was taken.

.1.. .... Bit 1 set indicates that the task represented by this TCB was a candidate for termination by ABEND.

..1. .... Bit 2 set indicates that a routine of the task represented by this TCB issued an unconditional GETMAIN that could only have been satisfied by the rolling out of another job step.

...1 .... Bit 3 indicates that the job step associated with this TCB was rolled out.

.... 1... Bit 4 set indicates that another task was in system-must-complete status.

.... .1.. Bit 5 set indicates that another task in this job step was in step-must-complete status at the time the dump was taken.

.... ..1. Bit 6 is applicable only for an initiator task. Setting of this bit indicates that a request for a region could not be satisfied.

....1 Bit 7 is the primary non-dispatchability indicator. Setting of this bit indicates that one or more of the secondary non-dispatchability bits (bytes 1-3 of the DAR field) was set at the time the dump was taken.

byte 2
The dispatching priority limit for the task represented by this TCB.

byte 3
The dispatching priority of the task represented by this TCB.

LLS hhhhhhhh
The load list element (LLE) for the program that was loaded by means of the LOAD macro instruction.

JLB hhhhhhhh
The address of the data control block associated with the JOBLIB associated with the task.

JPQ hhhhhhhh
Contains information pertaining to a job step TCB as follows:

byte 0

1... .... Bit 0 set indicates that if the associated job step had been allowed to continue processing without being interrupted by the dump program, the job step would have been purged.

.xxx xxxx Bits 1 through 7 are reserved for future use.

bytes 1-3
The address of the last contents directory entry for a job pack area (JPA) control queue.

RG 0-7 and RG 8-15
The register save area of the TCB being displayed. The general registers were stored in this area upon entry to the first routine invoked in the task. On entry to any

task, register 13 points to this TCB's register save area. This pointer is useful in locating the entry points of first routines and in tracing the save area chains.

FSA hhhhhhhh
The address of the first problem program save area.

TCB hhhhhhhh
The address of the TCB that had the next lowest priority on the ready queue at the time the dump was taken.

TME hhhhhhhh
The address of the timer element.

JST hhhhhhhh
The address of the first TCB for a job step. For tasks with a storage protection key of zero (as shown in the first byte of the PK-FLG field), this word contains the address of this TCB.

NTC hhhhhhhh
The address of the previous TCB that existed on the originating task's queue of subtask TCBs (sister). If this TCB was the first on the queue, this field contains zeros.

OTC hhhhhhhh
The address of the TCB representing the originating task (mother).

LTC hhhhhhhh
The address of the last TCB that existed on the originating task's queue of subtask TCBs at the time the dump was taken (daughter). If this TCB was the last on the queue, this field contains zeros.

IQE hhhhhhhh
The address of the interruption queue element (IQE) that was used in scheduling the EXTR routine on the originating task.

ECB hhhhhhhh
The address of the event control block (ECB) that would have been posted by the supervisor's task termination routines had either normal or abnormal task termination been allowed to occur.

TSPR hhhhhhhh

byte 0
This field contains flags that indicate the status of the time sharing (TSO). Without TSO or when TSO has not been started, this field contains zeros.

1... .... Bit 0 set indicates that this task is a time sharing task.

.1.. .... Bit 1 set indicates that the time sharing task should be set non-dispatchable. This bit was set by the TCBSTP routine while the routine was not executing as a privileged program.

..1. .... Bit 2 set indicates that the system is executing and requires that the time sharing task must not be interrupted by the attention exit or by the STATUS SVC.

...1 .... Bit 3 set indicates that a terminal I/O purge is required.

.... xxxx Bits 4 through 7 are reserved for future use.

byte 1
This field contains the number of SET STATUS starts required to make this time sharing task dispatchable.

byte 2
This field contains the limit priority of the time sharing task.

byte 3
This field contains the dispatching priority of the time sharing task.

D-PQE hhhhhhhh
The address of the region dummy partition queue element minus 8 (DPQE-8).

SQS hhhhhhhh
The address of an allocated queue element (AQE) which contains the amount of available bytes assigned to this task in the system queue area (SQA), and a pointer to the next AQE for this task.

STA hhhhhhhh
Internal STAE routine flags and the address of the STAE control block that was in effect at the time the dump was taken.

TCT hhhhhhhh
This word contains information pertaining to the dumped system's timing control table (TCT). The TCT field is divided into the following two subfields:

byte 0
Reserved for future use.

bytes 1-3
If the system management facilities option was present in the dumped system, these bytes contain the address of the dumped system's timing control table.

USR hhhhhhhh
This word is available to the user of the dumped system. It contains any information placed in it by the user.

DAR hhhhhhhh
The contents of this field were used by the damage assessment routines (DAR). Certain subfields displayed in this word were also used to control the dispatchability of the dumped task. The DAR field is divided into the following subfields:

byte 0
The first byte of the DAR field contains DAR flags. These flags are as follows:

1... .... Bit 0 set indicates that primary DAR recursion occurred in the dumped system. The damage assessment routine failed while writing a main storage image dump.

.1.. .... Bit 1 set indicates that secondary DAR recursion occurred in the dumped system. The damage assessment routine failed while attempting to reinstate a failing region or partition.

..xx .... Bits 2 and 3 are reserved for future use.

.... 1... Bit 4 set indicates that the system error task is failing. The DAR dump should not request any error recovery procedure (ERP) processing.

.... .xx. Bits 5 and 6 are reserved for future use.

....1 Bit 7 set indicates that an SVC dump is executing for this task.

byte 1
Bytes 1 through 3 of the DAR field are used to store secondary non-dispatchability flags. If any of the flag bits in this subfield were set, the primary non-dispatchability bit (the last bit in the FLG field) will also have been non-dispatchable. The bit settings that may appear in byte 1 are as follows:

xx.. .... Bits 0 and 1 are set by the damage assessment routines. Their meanings are:

1... .... Bit 0 set indicates that the task represented by this TCB is temporarily non-dispatchable.

.1.. .... Bit 1 set indicates that the task represented by this TCB is permanently non-dispatchable.

..xx .... Bits 2 and 3 are recovery management support and system error recovery flags. Their meanings are:

..1. .... Bit 2 set indicates that the task represented by this TCB is temporarily non-dispatchable.

...1 .... Bit 3 set indicates that the task represented by the TCB is permanently non-dispatchable.

.... x... Bit 4 is reserved for future use.

.... .1.. Bit 5 set indicates that this task is temporarily non-dispatchable. Timer services have been requested and the time-of-day clock is still inoperative.

.... ..xx Bits 6 and 7 are reserved for future use.

byte 2
The bit settings for byte 2 are as follows:

x... .... Bit 0 is reserved for future use.

.1.. .... Bit 1 set indicates that this task has been stopped by a STATUS stop.

..1. ... Bit 2 set indicates that task is non-dispatchable. An SVC dump is executing for another task.

...1 .... Bit 3 set indicates that this task is being swapped out by the time sharing (TSO).

.... 1... Bit 4 set indicates that this task is in an input wait state.

.... .1.. Bit 5 set indicates that this task is in an output wait state.

.... ..xx Bits 6 and 7 are reserved for future use.

byte 3
Reserved for future use.

RES hhhhhhhh
Reserved for future use.

JSCB hhhhhhhh

The address of the job step control block.

MVT ACTIVE REQUEST BLOCK (RB) FORMATTING: Request blocks (RBs) were used by the lines at the top of the dump page and in the preceding TCB display, are listed in the portion of the dump listing labeled "ACTIVE RBS." Information on each RB associated with the task is formatted as shown below:

PRB
IRB hhhhhh
SVRB
SIRB

Each RB display is preceded by a field that indicates the type and address of the RB being displayed. The four types of RBs that may be displayed under an MVT task are:

PRB

program request block

IRB

interruption request block

SVRB

supervisor request block (SVRBs may be divided into two categories; type 2 for resident routines and type 3 or 4 for transient routines)

SIRB

system interruption request block.

The type acronym for each RB is displayed in the first portion of the field. The starting address of the indicated request block appears in the last portion of the field. The contents of certain fields in the body of the formatted display are dependent upon the type of RB being displayed. Variations in display field usage are noted in the descriptions of the fields in which they occur.

RESV
TAB-LN hhhhhhhh
FL-PSA

This field shows both the function and the first word of the request block being displayed. The meanings of the function indicators and the values that follow them are:

RESV

indicates that the request block is either a PRB or an SVRB for resident routines. The first word of these particular RBs is reserved for future use and contains zeros.

TAB-LN

indicates that the request block being displayed is used as an SVRB for transient routines. The value field is divided into two subfields of two bytes each. The first two bytes show the displacement of the entry point of the module represented by this SVRB from the beginning of the transient area control table (TACT). The second subfield shows the length, in bytes, of the SVC routine.

FL-PSA

indicates that the RB being displayed is an IRB. The value portion of this field is divided into two subfields. The first subfield has a length of one byte and contains indicators for the timer routines. When there were no timer routines, this field contains zeros. The timer routine indicators set at the time the dump was taken are shown as:

1... .... indicates that the timer element was not on queue.
.1.. .... indicates that the local time-of-day option was used.
..00 .... indicates that the time interval was requested in timer units.
..01 .... indicates that the time interval was requested in binary units.
..11 .... indicates that the time interval was requested in decimal form.
.... 1... indicates that the time interval had expired.
.... .000 indicates a task request
.... .100 indicates a task request with an exit specified.
.... .001 indicates a wait request.
.... .011 indicates a real request.
.... .111 indicates a real request with an exit specified.

The second subfield is three bytes long and contains the starting address of the problem program register save area (PSA).

APSW hhhhhhhh

The APSW field displays information pertaining to the program status word that was active at the time the dump was taken. The functional variations associated with the usage of this field are:

- PRBs being formatted contain the right half (bytes 4 through 7) of the problem program's PSW when an ABTERM interruption occurred.
- IRBs, SIRBs, and SVRBs for resident routines use this field to display the right half (bytes 4 through 7) of the PSW that was active, in the dumped system, during the execution of an ABEND or ABTERM routine. If no ABEND or ABTERM routine was envoked in the dumped system, this field contains zeros.
- SVRBs for transient routines use this field in much the same way as SVRBs for resident routines. If an ABEND or ABTERM routine was invoked in the dumped system, bytes 4 through 7 of the associated PSW are displayed in this field. If an ABEND or ABTERM routine was not invoked, this field contains the last four characters of the name of the requested routine. (The last two characters of the name represent the SVC number.)

WC-SZ-STAB hhhhhhhh

This field contains information pertaining to wait conditions, request block sizes, and RB status and attribute characteristics. This field is divided into three subfields, as follows:

byte 0

The wait count that was in effect at the time of the dump.

byte 1

The size of this request block. This RB size is expressed as the number of doublewords comprising the block.

byte 2

The last two bytes of the WC-SZ-STAB field contain bit settings that reflect the status and attributes of the request block. The settings that may appear in byte 2 are:

xx.. .... Bits 0 and 1 indicate the type of RB being displayed. The possible settings for these two bits and their meanings are:

00.. .... This is a program request block (PRB).

01.. .... This is an interrupt request block (IRB).

10.. .... This is a system interrupt request block (SIRB).

11.. .... This is a supervisor request block (SVRB).

..x. x.xx Bits 2, 4, 6 and 7 are reserved for future use.

...1 .... Bit 3 set indicates that this request block is an SVRB for a transient routine.

.... .1.. Bit 5 is applicable only if the request block being displayed is an SVRB. If this bit is set, a checkpoint could have been taken in a user exit from the SVC routine associated with this RB.

byte 3

The last byte of the WC-SZ-STAB field contains more status and attribute flags. The possible settings for this subfield and their meanings are:

1... .... Bit 0 set indicates that the WT-LNK field in this RB display, contains in its last three bytes, the address of the TCB to which this request block is linked.

.1.. .... Bit 1 applies only to IRBs and SIRBs. If this bit is set, the indication is that at the time the dump was taken, the program associated with this RB was active.

..x. .... Bit 2 is reserved for future use.

...1 .... Bit 3 is applicable only to IRBs. The setting of this bit is an indication that the IRB was associated with an ETXR exit routine.

.... xx.. Bits 4 and 5 concern interruption queue elements (IQEs) and request queue elements (RQEs). This flag is used as follows:

.... 00.. This setting indicates that the request queue element was not to be returned to the free list when the exit was taken.

.... 01.. This setting indicates that the IRB had queue elements for asynchronously executed routines that were RQEs. This setting is applicable only if the RB being displayed is an IRB.

.... 10.. This setting indicates that the IQE was not to have been returned at EXIT.

.... 11.. This setting is applicable only to IRBs. If this setting appears, the indication is that the IRB had queue elements for asynchronously executed routines that were IQEs.

.... ..1. Bit 6 set indicates that request block storage could be freed at the time of exit.

....x Bit 7 indicates request wait conditions. The meanings of the two possible settings for this bit are:

....0 Bit 7 not set indicates that the request had to wait for a single event or all of a number of events.

....1 Bit 7 set indicates that the request had to wait for a number of events. This number of events was less than the total number of events that were waiting.

FL-CDE
EPA hhhhhhh
TQN

This field shows both the function and the fourth word of the request block being displayed. The meaning of the function indicator and the value following it is given below:

FL-CDE

the request block being displayed is either a PRB or an SVRB for a resident routine. The value field is divided into two elements. The first subfield has a length of one byte and contains control flag settings.

These control flags are as follows:

xxxx x... Bits 0 through 4 are reserved for future use.

.... .1.. indicates that a SYNC macro instruction was requested.

.... ..1. indicates that an XCTL macro instruction was requested.

....1 indicates that a LOAD macro instruction was requested.

The second subfield is three bytes long and contains the address of the contents directory entry (CDE) representing the module that this request block was associated with.

EPA

The request block being displayed is an IRB. The value field contains the entry point address of a routine that was asynchronously executed.

TQN

The request block being displayed represents a transient routine SVRB. The value field contains the address of the next request block that was on the queue of transient routines.

PSW hhhhhhhh hhhhhhhh

The resume program status word. This PSW represents the status of the program represented by the RB being displayed when a new RB was created. Had the dumped system been allowed to continue processing without being interrupted by the dump program, operation would have resumed on this PSW.

Q/TTR hhhhhhhh

This word is used to display various data, depending upon the type of request block being displayed. Usage of the Q/TTR value field is used by each type of request block as follows:

- PRBs and SVRBs that represented resident routines do not use this field; the first byte always contains zeros. Bytes 1 through 3 of the field show the address of a request block that requested the use of the same serially reusable program.

- IRBs utilize this field in one of two ways, to show either the three-byte link-field segment or the two-byte link-field segment, depending upon the IRB usage. The three-byte link-field segment appears in the Q/TTR value field as follows:

byte 0
Contains a count of the number of requests for the same exit (ETXR). This use count is utilized by the ATTACH macro instruction.

byte 1-3
Contains the starting address of the queue of interruption queue elements (IQEs).

Alternately, the Q/TTR value field may be formatted to show the two-byte link-field segment. In this instance, the field is used thusly:

byte 0-1
Reserved for future use.

bytes 2-3
The starting address of the queue of request queue elements (RQEs).

- SVRBs that represented transient routines display two data elements in this field. The first subfield has a length of one byte and shows the number of requests if the transient routine was overlaid. The last three bytes of the Q/TTR field contain the relative direct access device address for the associated supervisor routine in the form TTR.

WT-LNK hhhhhhhh
This field displays information pertaining to wait counts and request block linkages. In the case of a transient SVC, if this field contains x'FF', either the routine represented by the SVRB is currently being brought into the transient area, or this routine has been displaced in the transient area by a routine requested by a higher priority task. To tell what has happened, compare the APSW and NM field contents as described under NM below. This field is divided into two subfields, one with a length of one byte and the other with a length of three bytes. These subfields show the following:

byte 0
The number of requests that were pending at the time the dump was taken (wait count).

byte 1-3
The address of the next request block on the RB queue. In the last RB on the queue, this field contains the address of the task control block (TCB).

NM cccccccc
The eight character name of the load module represented by the request block being displayed with a possible exception for transient SVRBs.

If byte 0 of the WT-LNK field contains x'FF', it is possible that the module represented by this SVRB has been overlaid in the transient area by a module requested by a higher priority task. Compare the APSW field, (providing it contains the four low-order bytes of a module name) with the last four characters (the hexadecimal should be translated to EBCDIC) of the module name in the NM field. No match indicates the user of the transient area has been pre-empted by a higher priority task. NM therefore represents the module currently in the transient area, not the module represented by this SVRB.

If a match results, NM correctly identifies the module name requested by this SVRB.

EPA hhhhhh
The address of the entry point of the module named in the NM field of this RB display.

STA hhhhhh
The starting address of the module identified in the NM field of this RB's display.

LN hhhhhh
The length, in bytes, of the load module that is represented by this request block.

ATR1 hh
This one byte field displays the attributes of the described module. These attributes are taken from the contents directory entry associated with the module. The meanings of the attribute flag settings are given below:

1...	Bit 0 set indicates that the module was resident in the link pack area.
.1..	Bit 1 set indicates that at the time the dump was taken, the module represented by this request block was in the process of being fetched.
..1.	Bit 2 set indicates that the module was reenterable.
...1	Bit 3 set indicates that the module was serially reusable.

.... 1... Bit 4 set indicates that the module could not have been reused. This flag setting is not applicable if either bit 2 or 3 is set.

.... .1.. Bit 5 set indicates that the contents directory entry associated with this module reflects the use of an alias name. This information applies only to minor CDEs.

.... ..1. Bit 6 set indicates that the module was in the job pack area.

....1 Bit 7 set indicates that the module was considered not only-loadable.

MVT MAIN STORAGE INFORMATION: Each task operating under the MVT option of the operating system was dynamically assigned a region of main storage that consisted of one or more 2K-byte subpool areas. To keep track of main storage allocations, the MVT supervisor maintained a partition queue associated with each region. Composed of partition queue elements (PQEs), and residing in the system queue area, this partition queue was connected to the TCBs for each task in a job step through a dummy partition queue element (DPQE).

Information on the areas of main storage allocated to each task, is presented to the user in a separate portion of each task's dump listing headed "MAIN STORAGE." This main storage information is formatted as shown below:

D-PQE hhhhhhhh
The address minus eight bytes of the dummy partition queue element (DPQE-8) connecting the partition queue to this task's TCB.

FIRST hhhhhhhh
The starting address of the first partition queue element (PQE) on this region's partition queue.

LAST hhhhhhhh
The starting address of the last PQE on the partition queue.

PQE hhhhhh
The starting address of one of the partition queue elements on the partition queue bounded by the addresses given on the line above.

FFB hhhhhhhh
The starting address of the first free block queue element (FBQE) on the free block queue associated with this PQE. If no FBQEs exist, this field contains the address of the PQE being displayed

LFB hhhhhhhh
The starting address of the last free block queue element (FBQE) on the free block queue associated with this PQE. If no FBQEs exist, this field shows the starting address of this PQE.

NPQ hhhhhhhh
The starting address of the next partition queue element on the partition queue. If the PQE being displayed was the last PQE on the queue, this field contains zeros.

PPQ hhhhhhhh
The starting address of the partition queue element on the partition queue that preceded this PQE. If this PQE was the first on the queue, this field contains zeros.

TCB hhhhhhhh
The starting address of the TCB of the job step to which the described region is assigned. If this field contains zeros, the indication is that the area of main storage was obtained from unassigned free space.

RSI hhhhhhhh
The size of the region being described. This number is a multiple of 2K (2048).

RAD hhhhhhhh
The starting address of the region being described by this PQE.

FLG hhh
The FLG field shows the settings of several PQE flags whose meanings are given below:

x... .... Bit 0 indicates region ownership. The meanings of the settings are:

0... .... indicates that the space described by this PQE was owned by the associated task.

1... .... indicates that the space described by this PQE was borrowed.

.1.. .... The setting of bit 1 is meaningful only if bit 0 was not set. If this bit is set and bit 0 is not set, the indication is that the region had been rolled out.

..1. .... Bit 2 set indicates that the region described by this PQE was borrowed by another task.

...x xxxx Bits 3 through 7 are reserved for future use.

MVT LOAD LIST FORMATTING: A load list was maintained by the dumped system's supervisor in order to keep track of the load modules that were in main storage and the area of main storage each occupied. The load list maintained by a system operating under the MVT option of the operating system contained a series of load list elements (LLEs), each of which was associated with a particular load module through the use of a control block called a contents directory entry (CDE). A formatted listing of the dumped system's MVT load list appears as follows:

CDE hhhhhh
The starting address of the contents directory entry associated with this load list item.

NM cccccccc
The eight-character name of the entry point to the load module represented by this entry.

USE hh
The count of the number of uses (through the ATTACH, LINK and XCTL macro instructions) of the load module, and the number of times a LOAD macro instruction was issued for the module.

RESP hh
The responsibility count contained in the load list entry associated with the load module. This count indicates the number of requests made by the LOAD macro instruction for the indicated load module. This count was decremented by one for each occurrence of the DELETE macro instruction.

ATR1 hh
The attributes of the load module described in this load list entry. These attributes are taken from the contents directory entry associated with the module. The meanings of the attribute flag settings are given below:

1... .... Bit 0 set indicates that the module was resident in the link pack area.

.1.. .... Bit 1 set indicates that at the time the dump was taken, the load module represented by this load list element was in the process of being loaded.

..1. .... Bit 2 set indicates that the load module was reenterable.

...1 .... Bit 3 set indicates that the load module was serially reusable.

.... 1... Bit 4 set indicates that the load module could not have been reused. This flag setting is not applicable if either bit 2 or 3 is set.

.... .1.. Bit 5 set indicates that the contents directory entry associated with this load module reflects the use of an alias name. If this bit is set, this line of the load list display reflects information taken from a minor CDE.

.... ..1. Bit 6 set indicates that the load module was in the job pack area.

....1 Bit 7 set indicates that the load module was considered not only-loadable.

EPA hhhhhh
The address of the entry point of the load module named in the NM field of this load list display line.

STA hhhhhh
This field contains the starting address of the load module identified in the NM field of this load list display line.

LN hhhhhh
The LN field supplies the user with the length, in bytes, of the load module represented by this load list entry (LLE).

MVT JOB PACK QUEUE FORMAT: A job pack area control queue (JPACQ) exists for each job step in the dumped system that used a program not in the link pack area. The job pack queue, like the link pack area, is made up of contents directory entries (CDEs). This area describes routines in a job step region that were brought into main storage by contents supervision routines to perform a task in the job step. The IMDPRDMP program displays the contents of the dumped MVT system's job pack queue as follows:

CDE hhhhhh
The starting address of the contents directory entry associated with this job pack queue element.

NM cccccccc
The eight-character name of the entry point to the load module represented by this entry.

USE hh
The count of the number of uses (through the ATTACH, LINK and XCTL macro instructions) of the load module, and the number of times a LOAD macro instruction was issued for the module.

RESP NA
This responsibility count field is flagged 'NA' to indicate that the information is not applicable to modules displayed in the job pack queue.

ATR1 hh
The attributes of the load module described in this job pack queue entry. These attributes are taken from the contents directory entry associated with the module. The meanings of the attribute flag settings are:

1... .... Bit 0 set indicates that the module was resident in the link pack area.

.1.. .... Bit 1 set indicates that at the time the dump was taken, the load module represented by this job pack queue entry was in the process of being loaded.

..1. .... Bit 2 set indicates that the load module was reenterable.

...1 .... Bit 3 set indicates that the load module was serially reusable.

.... 1... Bit 4 set indicates that the load module could not have been reused. This flag setting is not applicable if either bit 2 or 3 is set.

.... .1.. Bit 5 set indicates that the contents directory entry associated with this load module reflects the use of an alias name. If this bit is set, this line of the job pack queue display reflects information taken from a minor CDE.

.... ..1. Bit 6 set indicates that the load module was in the job pack queue area.

....1 Bit 7 set indicates that the load module was considered not only-loadable.

EPA hhhhhh
The address of the entry point of the load module named in the NM field of this job pack queue entry display line.

STA hhhhhh
This field contains the starting address of the load module identified in the NM field of this job pack queue entry display line.

LN hhhhhh
The LN field supplies the user with the length, in bytes, of the load module represented by this job pack queue entry.

MVT DATA EXTENT BLOCK (DEB) FORMATTING: Data extent blocks (DEBs), describing a data set's external storage requirements, were queued to those task control blocks (TCBs) that represented tasks requiring auxiliary storage input/output processing. External storage information, taken from each DEB, is formatted as shown below:

DEB hhhhhh
The starting address of the basic section of the DEB being displayed.

APPENDAGES
The word "appendages" informs the user that the five named fields on this line contain information taken from the appendage vector table preceding the DEB being displayed. The named fields appearing on the rest of this line are:

END OF EXT hhhhhh
The entry point of the end-of-extent appendage routine.

SIO hhhhhh
The entry point of the start I/O appendage routine.

PCI hhhhhh
The entry point of the program-controlled-interruption appendage routine.

CH END hhhhhh
The entry point of the channel-end appendage routine.

AB END hhhhhh
The entry point of the abnormal-end appendage routine.

PFX hhhhhhhh hhhhhhhh hhhhhhhh
The second line of a DEB display contains information taken from the prefix section of the DEB being displayed. The area is subdivided as follows:

byte 0
The first byte of the prefix area contain the contents of the I/O support work area. This area is used only by DEBs dealing with direct access storage devices.

bytes 1-7
The next seven bytes of the DEB prefix section are used by DEBs associated with direct access storage device functions. This subfield displays the data set control block's (DSCB) address used by I/O support. The address is expressed in the following format:

bytes 1 and 2	the bin (cell) number.
bytes 3 and 4	the cylinder address.
bytes 5 and 6	the track address.
byte 7	the record number.

bytes 8-11
The third word of the PFX field contains the data control block (DCB) modification mask that was used by I/O support.

byte 12
The length of the DEB in doublewords .

bytes 13-15
The remainder of the DEB prefix section is reserved for future use.

TCB hhhhhh
This field marks the beginning of the basic section of the data extent block. The TCB field is divided into two subfields as follows:

byte 0
The number of subroutines for which a LOAD macro instruction was issued during the execution of the OPEN executor routines.

bytes 1-3
The starting address of the task control block to which this DEB was enqueued.

NDEB hhhhhh
The NDEB field is also used to display two data elements. It is subfielded as follows:

byte 0
The overall length of a data extent block includes the length of a variable length access method dependent section. The first byte of the NDEB field, expresses the length of the access method dependent section in bytes. If the access method was BDAM, this indicator is expressed as a number of fullwords.

bytes 1-3
The last portion of the NDEB field displays the starting address of the basic section of the next DEB on the task's queue. If this DEB was the last on the queue, the contents of this field are the starting address of the TCB that enqueued this DEB.

ASYN hhhhhhhh
This field contains data set status flags and the address of the associated IRB. This field is used as follows:

byte 0
The first byte of the ASYN field contains data set status flags. These flags have the following meanings:

xx..	Bits 0 and 1 indicate the data set's disposition. The possible settings are:
01..	This setting indicates that the disposition was OLD.
10..	This setting indicates that the disposition of the data set was MOD (modify).
11..	This setting indicates that the disposition was NEW.
..1.	Bit 2 set indicates that an end-of-volume (EOV) or end-of-file (EOF) condition had been encountered.
...1	The setting of bit 3 has one of two meanings depending upon the external storage medium. For disk this indicator reflects a release of unused external storage. For tape, the meaning of this indicator is that an emulator tape with second generation format was being used.
.... 1...	Bit 4 set is a data control block (DCB) modification indicator.
.... .1..	Bit 5 set has two meanings, depending upon the auxiliary storage recording medium. For disk, the setting of bit 5 indicates that a split cylinder was encountered. For tape, this flag indicates that an emulator tape with possible mixed parity records was used.
.... ..1.	Bit 6 set indicates the use of nonstandard labels.
....1	Bit 7 set indicates that reduced error recovery procedures were used on magnetic tapes containing the data set represented by this DEB.

bytes 1-3
The last portion of the ASYN field shows the starting address of the IRB that was associated with asynchronous appendage exit scheduling.

SPRG hhhhhhhh
This field contains information on I/O processing methods and the system PURGE routine. The usage of this field is as follows:

byte 0
The first byte of this field contain flags that indicate the method of input/output processing and the disposition of the data set that was to have been performed when an end-of-volume condition occurred. These flag settings are:

1...	Bit 0 was set by ABEND. The setting of this bit indicates that the data set associated with this DEB was a SYSABEND or SYSUDUMP data set.
.0..	Bit 1 is always zero.
..xx	Bits 2 and 3 show the end-of-volume disposition procedure. The values for this flag are:
..01	REREAD
..11	LEAVE
.... xxxx	The last half of this byte contains flags that indicate the type of input/output processing that was performed on the data set represented by this DEB. The values for this flag are:
.... 0000	INPUT
.... 1111	OUTPUT
.... 0011	INOUT
.... 0111	OUTIN
.... 0001	RDBACK
.... 0100	UPDAT

byte 1
The quiesce count. The byte is associated with the system PURGE routines (SVC 16) and indicates the number of auxiliary storage devices that were executing the user's channel programs.

bytes 2-3
Reserved for future use.

UPRG hhhhhhhh
The UPRG field contains extent information and data used by the user's purge routines. This field is divided into the following two subfields:

byte 0
The number of extents that were specified in the DSCBs associated with this DEB.

bytes 1-3
The address of the first input/output block (IOB) in the user's purge chain.

PLST hhhhhhhh
Task priority and supervisor purge information are contained in this field. This field is formatted as follows:

byte 0
The priority of the task under which this DEB was enqueued.

bytes 1-3
The starting address of a parameter list that was used to locate the purge event control block (ECB) for a supervisor purge request.

DCB hhhhhhhh
The DCB field contains three data elements. These are displayed in the format given below:

byte 0

xxxx	The storage protection key that was associated with the task under which this DEB was enqueued.
.... 1111	A hexadecimal 'F' in bits 4 through 7 of this field identify this control block as a data extent block (DEB).

bytes 1-3
The starting address of the data control block (DCB) that was associated with this DEB.

AVT hhhhhhhh
The AVT field displays two DEB data elements and is subfielded as follows:

byte 0
The DEB extent scale that is used to determine the size of the device dependent section of this DEB. For direct access devices, a 4 is displayed in this subfield. For a nondirect access device or a communication device, a 2 is displayed.

bytes 1-3
In most cases the last portion of the AVT field shows the starting address of the appendage vector table preceding this DEB. This table of appendage routine addresses appears on the first line of this DEB's display.

OP-UCB hhhhhhh
The contents of this field have meaning only when the DEB being displayed describes a data set that was assigned to a unit record or magnetic tape device. This information is formatted from the device dependent section of the DEB. The OP-UCB field is subfielded as follows:

byte 0
This first subfield is applicable only to data sets assigned to magnetic tape devices and shows the SET MODE operation code. For a data set that was assigned to a unit record device, this subfield is reserved.

bytes 1-3
The starting address of the unit control block (UCB) associated with the data set described by the DEB being displayed.

The following four fields are present only for data sets assigned to the IBM 3525 Card Punch for multi-function. The information is formatted as shown below:

UCB hhhhhhhh
byte 0
The device modifier field (not used for the 3525).
bytes 1-3
The starting address of the unit control block (UCB) associated with the data set described by the DEB being displayed.

RDRDCB hhhhhhhh
The starting address of the data control block (DCB) for the read associated data set.

PCHDCB hhhhhhhh
The starting address of the data control block (DCB) for the punch associated data set.

WTRDCB hhhhhhhh
The starting address of the data control block (DCB) for the print associated data set.

The final portion of a DEB display shows information pertaining to a data set that was assigned to a direct access device. This information, taken from the DEB's device dependent section, is arranged in columnar format with a line for each extent. The information is formatted as shown below:

FM-UCB hhhhhhhh
The first column displays two data elements and is formatted as follows:

byte 0
The device modifier showing the file mask.

bytes 1-3
The starting address of the unit control block (UCB) that was associated with the data extent.

START hhhhhhhh
The address of the beginning of the direct access device extent. The first four characters represent the cylinder address and the last four characters represent the track address.

END hhhhhhhh
The address of the end of the data extent. Cylinder and track references are formatted as in the extent beginning address, described above.

TRKS hhhh
The number of direct access tracks bounded by the starting and ending addresses shown in the previous two columns.

<u>MVT TASK INPUT/OUTPUT TABLE (TIOT) FORMATTING</u>: A task input output table (TIOT) was constructed for each task in the dumped system by MVT job management routines. Residing in the system queue area, this table contained primary pointers to control blocks used by I/O support routines. As the functions of several TIOT fields were dependent upon the state of associated external storage devices, multiple definitions may apply. The TIOT that was constructed in an MVT system is formatted as shown.

TIOT hhhhhh
The starting address of the task input/output table being displayed.

JOB cccccccc
The eight-character name of the job for which this TIOT was constructed.

STEP cccccccc
The eight-character name specified in the label field of the EXEC JCL statement associated with this job step.

PROC cccccccc
If the job step for which this TIOT was constructed was invoked from a cataloged procedure, the procedure name, as contained in the EXEC JCL statement, is displayed in this field.

Each data set associated with the indicated task is represented by a separate DD entry that is included in the TIOT. Each TIOT entry is displayed on a separate line in columnar format. The use and meaning of each column is given below:

OFFSET hhhh

The offset of this DD entry from the beginning of the TIOT in hexadecimal.

LN-STA hhhhhhhh

Four bytes of length and status information, described below:

byte 0

The total length (including all device entries) in bytes of the DD entry being displayed on this line.

byte 1

Status byte A, one of three status bytes in a TIOT entry. The meanings of the status byte settings are:

x... .x.. Bits 0 and 5 indicate the tape label processing that was to have been performed. The meanings of the settings are:

0... .0.. Nonlabeled tape or an indication to bypass label processing.

0... .1.. Standard labels and standard user labels.

1... .0.. Nonstandard labels.

.1.. .... The setting of status bit 1 has two meanings, depending upon the processing phase that had been reached at the time the system was dumped. During allocation processing, the setting of this bit indicates that this entry represents a split cylinder primary space allocation DD. If the dump was taken during step termination processing, the setting of this bit indicated that no unallocation of space was necessary.

..1. .... The setting of status bit 2 works under the same philosophy as status bit 1. During allocation processing, the setting of this bit indicates that this entry represents a split cylinder secondary space allocation DD. If the dump was taken during step termination processing, the indication was one of rewinding with no unload.

...1 .... Bit 3 set indicates that this DD entry represents a JOBLIB.

.... 1... Bit 4 set indicates that direct access device space management was deemed necessary.

.... ..1. The setting of bit 6 specifies that the tape volume was to have been rewound and unloaded.

....1 The setting of bit 7 specifies that the tape volume was to have been rewound.

byte 2

The third byte of this column has meaning only during the allocation phase. This displays the number of devices that were requested by the data set represented by the TIOT entry displayed on this line.

byte 3

The last byte of the LN-STA field displays a TIOT field that had meaning at two points during the processing of this task. During the allocation process, this field contained a link to the appropriate prime split, unit affinity, volume affinity or suballocate TIOT entry. After CLOSE processing, this byte was used thusly:

1... .... The setting of bit 0 indicates that the data set represented by this DD entry was a SYSOUT data set that contained data.

.xxx xxxx Bits 1 through 7 are reserved for future use.

DDNAME cccccccc

The eight-character DD name associated with the TIOT entry being displayed.

TTR-STC hhhhhhhh

The first three bytes of this column display the relative track address (TTR) of the job file control block (JFCB) associated with this entry.

STB-UCB hhhhhhhh

The last column in a TIOT display contains information taken from the one-word device entries that are appended to each TIOT entry. One TIOT device entry exists for each allocated device. This display field shows this information in the following format:

byte 0

Status byte B. The status bits have the following meanings:

1... .... Bit 0 set indicates that the data set associated with this line of the TIOT display was present on the device represented by this TIOT device entry.

.1.. .... Bit 1 set indicates that the data set associated with this line of the TIOT display would have used the device represented by this TIOT device entry.

..1. .... Bit 2 set indicates that the device represented by this device entry violated separation.

...1 .... Bit 3 set indicates that a volume serial number was present.

.... 1... Bit 4 set indicates that a setup message was required.

.... .x.. Bit 5 indicates the device disposition that would have taken place had the dumped system been allowed to continue processing this task. The settings for this bit are:

.... .0.. Indicates that if the volume was required to be unloaded, the volume was to have been deleted.

.... .1.. Indicates that if the volume was requires to be unloaded, the unloaded volume was to have been retained.

.... ..1. Bit 6 indicates that an unload requirement had been made.

....1 Bit 7 set indicates that a load or label verification requirement had been made.

bytes 1-3

The address of the UCB that was used in all cases except when the device was a 2321 data cell drive. For a 2321, this address is that of the description in the UCB of the cell in the bin.

MFT Control Block Formatting

The formats described below are repeated for each requested task that is printed. A sample of the major system control blocks, as formatted from an MFT dump, is shown in Figure 23.

MFT TASK CONTROL BLOCK (TCB) FORMATTING: The task control block (TCB) contains information pertaining to the specific task identified in the heading lines at the top of the dump listing page. It is formatted as follows:

TCB hhhhhh

The address of the task control block being displayed is given in this first display field.

RBP hhhhhhhh

The starting address of the request block (RB) that was currently associated with the task represented by this TCB.

PIE hhhhhhhh

The address of the first program interrupt element (PIE) enqueued by this TCB.

DEB hhhhhhhh

The address of the beginning of the data extent block (DEB) queue that was associated with this task. Information on the contents of each DEB in the queue is given in a separate portion of this MFT task's dump listing.

TIO hhhhhhhh

The starting address of the task input/output table (TIOT) that was constructed during device allocation for the task represented by this TCB. The contents of this table are displayed in a later portion of this task's display.

CMP hhhhhhhh

This word contains ABEND indicators and user and system completion codes as follows:

byte 0

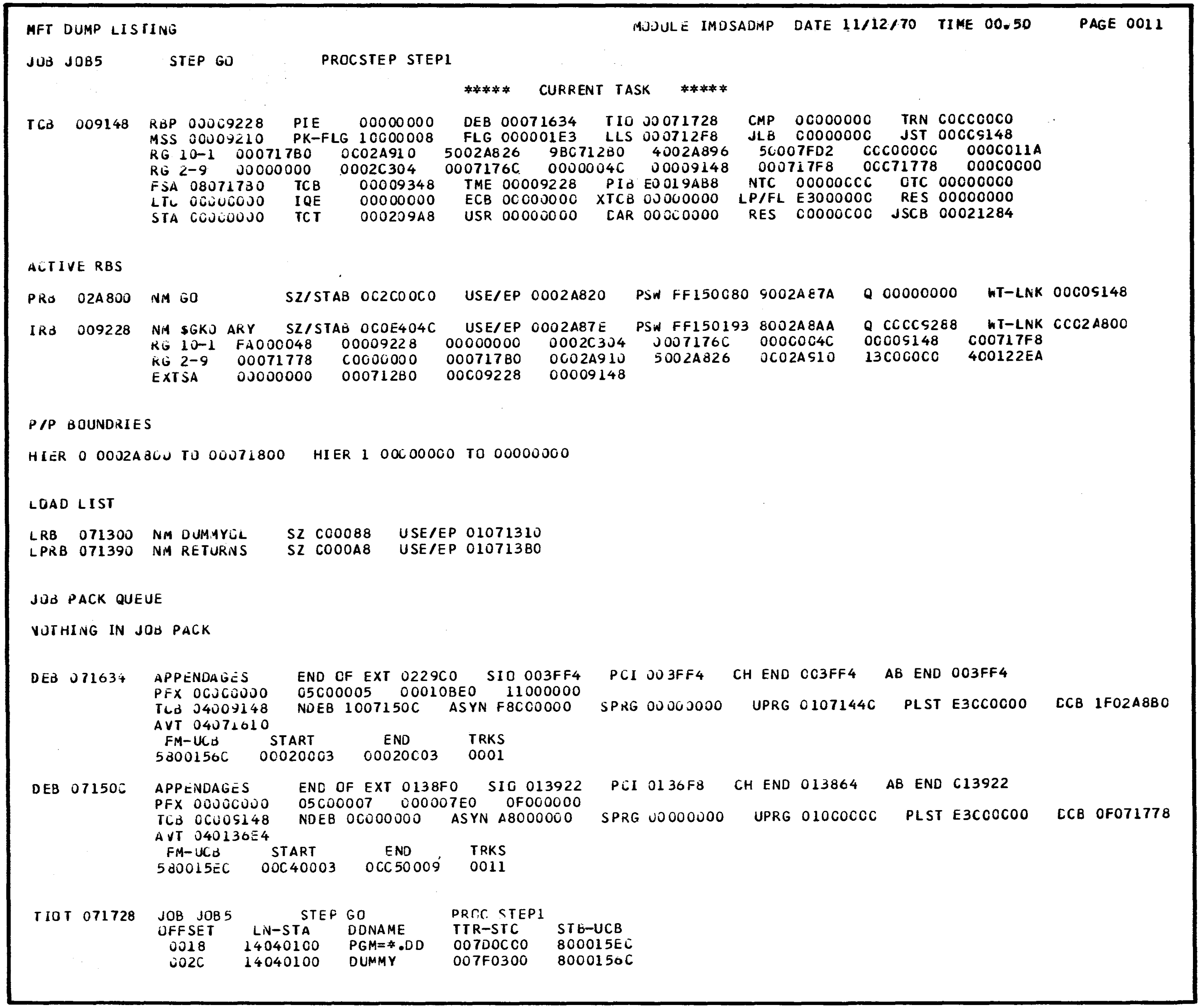
1... ... Bit 0 set indicates that a dump had been requested.

.1.. .... Bit 1 is reserved for future use but is set for MVT compatibility.

..1. .... Bit 2 set indicates that a portion of the problem program's main storage area was overlaid by a second load of ABEND. A first load overlay is indicated by the setting of bit 14 of the PK-FLG field.

...x .... Bit 3 is reserved for future use.

.... 1... Bit 4 set indicates that a double ABEND occurred in the dumped task.



```
MFT DUMP LISTING                                                        MODULE IMDSADMP  DATE 11/12/70  TIME 00.50     PAGE 0011

JOB JOB5        STEP GO          PROCSTEP STEP1

                                                *****   CURRENT TASK   *****

TCB   009148  RBP 00009228   PIE      00000000   DEB 00071634   TIO 00071728   CMP  00000000   TRN C0CC00C0
              MSS 00009210   PK-FLG 10000008     FLG 000001E3   LLS 000712F8   JLB  C000000C   JST 000C9148
              RG 10-1   000717B0    0C02A910    5002A826    9BC712B0    4002A896    50007FD2    CCC0000G    000C011A
              RG 2-9    00000000    0002C304    0007176C    0000004C    00009148    000717F8    0CC71778    000C0000
              FSA 08071730   TCB      00009348   TME 00009228   PIB E0019AB8   NTC  00000CCC   OTC 00000000
              LTC 00000000   IQE      00000000   ECB 0C000000   XTCB 00000000  LP/FL E300000C  RES 00000000
              STA C0000000   TCT      000209A8   USR 00000000   CAR 000C0000   RES  C0000C0C   JSCB 00021284


ACTIVE RBS

PRB   02A800  NM GO          SZ/STAB 0C2C00C0    USE/EP 0002A820   PSW FF150C80 9002A87A    Q 00000000    WT-LNK 000C9148

IRB   009228  NM $GKO ARY    SZ/STAB 0C0E404C    USE/EP 0002A87E   PSW FF150193 8002A8AA    Q CCCC9288    WT-LNK CCC2A800
              RG 10-1   FA000048    00009228    00000000    0002C304    0007176C    000C004C    00009148    C00717F8
              RG 2-9    00071778    C0000000    000717B0    0C02A910    5002A826    0C02A910    13C000C0    400122EA
              EXTSA     00000000    000712B0    00C09228    00009148


P/P BOUNDRIES

HIER 0 0002A800 TO 00071800   HIER 1 00C00000 TO 00000000


LOAD LIST

LRB   071300  NM DUMMYGL     SZ C00088    USE/EP 01071310
LPRB  071390  NM RETURNS     SZ C000A8    USE/EP 010713B0


JOB PACK QUEUE

NOTHING IN JOB PACK


DEB 071634    APPENDAGES      END OF EXT 0229C0   SIO 003FF4   PCI 003FF4   CH END 003FF4   AB END 003FF4
              PFX 00000000    05C00005    00010BE0    11000000
              TCB 04009148    NDEB 1007150C    ASYN F8C00000    SPRG 00000000    UPRG 0107144C    PLST E3CC0000    DCB 1F02A8B0
              AVT 04071610
               FM-UCB      START       END       TRKS
              5800156C    00020003    00020C03    0001

DEB 07150C    APPENDAGES      END OF EXT 0138F0   SIO 013922   PCI 0136F8   CH END 013864   AB END C13922
              PFX 00000000    05C00007    000007E0    0F000000
              TCB 0C009148    NDEB 0C000000    ASYN A8000000    SPRG 00000000    UPRG 010C0C0C    PLST E3C00C00    DCB 0F071778
              AVT 040136E4
               FM-UCB      START       END       TRKS
              580015EC    00C40003    0CC50009    0011


TIOT 071728   JOB JOB5        STEP GO           PROC STEP1
              OFFSET     LN-STA      DDNAME       TTR-STC     STB-UCB
               0018     14040100     PGM=*.DD     007D0CC0    800015EC
               002C     14040100     DUMMY        007F0300    8000156C
```

Figure 23. Sample of MFT Control Block Format

CMP hhhhhhhh -- byte 0 -- (continued)

.... .1.. Bit 5 set indicates that a dump message (WTO) was to have been issued.

.... ..1. Bit 6 set indicates that the dumped system's scheduler was to have printed an indicative dump.

....1 Bit 7 set indicates that an ABEND message, to be printed by the ABDUMP routine, was provided.

bytes 1-3

The first 12 bits contain a system completion code. These codes and their meanings are explained in the publication <u>IBM System/360 Operating System: Messages and Codes</u>, GC28-6631 under the heading "System Completion Messages." A user completion code is contained in the last 12 bits.

TRN hhhhhhhh

Contains flags as follows:

byte 0

1... .... Bit 0 set indicates that decimal simulator programs were being used on a System/360 model 91 machine.

.1.. .... Bit 1 set indicates that checkpoints were not taken for this step.

..1. .... Bit 2 set indicates that the TCB being displayed was associated with either a graphics foreground job or the graphic job processor.

...1 .... Bit 3 set indicates that the TCB being displayed was associated with a 7094 emulator task that was being run on a System/360 model 85 machine.

.... xxxx Bits 4 through 7 are reserved for future use.

bytes 1-3

Reserved.

MSS hhhhhhhh

Main storage supervision as follows:

byte 0

This byte is reserved for future use.

bytes 1-3

This subfield displays one of two addresses. If the TCB being displayed represents a job step, this subfield contains the address of the boundary box. If this TCB represents a subtask, this field displays the address of the gotten queue element (GQE). GQEs are preset only if the dumped system issued a GETMAIN macro instruction for the space.

PK-FLG hhhhhhhh

The storage protection key and a series of flags associated with the task being displayed. This field is divided into several subfields. These are:

byte 0

xxxx .... The storage protection key associated with the task represented by this TCB.

.... 0000 Always contain zeros.

byte 1

1... .... Bit 0 set indicates that an abnormal termination was in progress at the time the dump was taken.

.1.. .... Bit 1 set indicates that a normal termination was in progress at the time the dump was taken.

..1. .... Bit 2 set indicates that ABEND was initiated by the resident abnormal termination routine.

...1 .... Bit 3 set indicates that recursion through ABEND was permitted.

.... 1... Bit 4 set indicates that the graphics abnormal termination routine had been entered for the task represented by the TCB being displayed.

.... .1.. Bit 5 set indicates that the CLOSE routine was initiated by ABEND.

.... ..1. Bit 6 set indicates that a portion of the problem program's main storage area was overlaid in order to process ABEND routines. (See also bit 2 of the CMP display field.)

....1 Bit 7 set indicates that the queueing of asynchronous exits for the task represented by the TCB being displayed, was prohibited.

byte 2

1... .... Bit 0 set indicates that ABEND was prohibited for this task. The setting of this bit has meaning only if the TCB being displayed represents a system task.

.xx. ..x. Bits 1, 2 and 6 are reserved for future use.

...1 .... Bit 3 set indicates that the task represented by the TCB being displayed had a forced completion imposed upon it. Other tasks in the dumped system could not have been performed until this task had been completed.

.... 1... Bit 4 set indicates that the job step had a forced completion imposed upon it. Other tasks in the dumped system could not have been performed until this job step had been completed.

.... .1.. Bit 5 indicates that dump processing had been initiated in ABEND.

....1 Bit 7 set indicates that the task represented by the TCB being displayed was a member of a time sliced group.

byte 3

xx.x ...x Bits 0, 1, 3 and 7 are reserved for future use.

..1. .... Bit 2 is an exit effector indicator. The setting of this bit indicates that at the time the dump was taken, system error routines were operating on this task.

.... 1... Bit 4 set indicates that floating point registers existed in the dumped system.

.... .1.. Bit 5 set indicates that at the time the dump was taken, job scheduler routines were processing.

.... ..1. Bit 6 set indicates that at the time the dump was taken, an XCTL routine was changing the storage protection key in the PSW from zero to the one used by the problem program.

FLG hhhhhhhh

This field displays a further series of flags and certain priority indicators. This word is formatted as follows:

byte 0

Reserved for future use.

byte 1

xxxx xxx. Bits 0 through 6 are reserved for future use.

....1 Bit 7 is the primary non-dispatchability indicator. Setting of this bit indicates that one or more of the secondary non-dispatchability bits (bytes 1-3 of the DAR field) was set at the time the dump was taken. If this bit is set, the task represented by this TCB was considered to be non-dispatchable.

byte 2

This byte contains the number of resources for which the task represented by this TCB was enqueued.

byte 3

This byte displays the dispatching priority of the task represented by this TCB.

LLS hhhhhhhh

The address of the last request block (RB) that was created by the loading of a module that used the LOAD macro instruction.

JLB hhhhhhhh

The address of the data control block (DCB) representing the JOBLIB associated with this task.

JST hhhhhhhh

Job step information. The contents of this field have meaning only when the dumped MFT system was operating with the subtasking option. If this was the case, this field shows the address of the first TCB for a job step.

RG 0-7 and RG 8-15

The register save area of the TCB being displayed. This pointer is useful in locating the entry points of first routines and in tracing the save area chains.

FSA hhhhhhhh

This field displays two data elements and is formatted as follows:

byte 0

The TCB identification code.

byte 1-3

The address of the first problem program save area.

TCB hhhhhhhh
The address of the TCB that had the next lowest priority on the ready queue at the time the dump was taken.

TME hhhhhhhh
The address of the timer element.

PIB hhhhhhhh
The PIB field displays two items of information in the following format:

byte 0
This byte contains flags that identify the partition attributes. These flags are:

xx.. .... Bits 0 and 1 indicate the function of the partition. The possible functions are given below:

00.. .... System task partition.

01.. .... Reader partition.

10.. .... Writer partition.

11.. .... Processing program partition.

..x. .... Bit 2 gives the partition size. The meanings of the possible settings are:

..0. .... Small partition.

..1. .... Large partition.

...1 .... Bit 3 set indicates that CPU timing was stopped by FINCH until a transient routine was loaded.

.... xx.. Bits 4 and 5 are reserved for future use.

.... ..1. Bit 6 set indicates that the partition associated with this task was a writer partition. This bit is used by ABEND, transient writers and resident writers.

.... ..1. Bit 7 set indicates that at the time the system was dumped, the scheduler was in control. Had this task's TIOT been written to SYS1.SYSJOBQE, this bit would not be set.

bytes 1-3
The last portion of the PIB field shows the address of the partition information block (PIB) that was associated with this task's partition.

NTC hhhhhhhh
The address of the previous TCB that existed on the originating task's queue of subtask TCBs (sister). If the TCB was the first on the queue, this field contains zeros. The contents of the NTC field have meaning only if the dumped system was operating with the MFT subtasking option.

OTC hhhhhhhh
The OTC field is applicable only when the dumped system was operating under MFT subtasking option. If this was the case, this field displays the address of the TCB representing the originating task (mother).

LTC hhhhhhhh
The address of the last TCB that existed on the originating task's queue of subtask TCBs (daughter) at the time the dump was taken. If this TCB was the last on the queue, this field contains zeros. This field is applicable only if the dumped system was operating under the MFT subtasking option.

IQE hhhhhhhh
The address of the interruption queue element (IQE) that was used in scheduling the ETXR routine on the originating task. The contents of this field have no meaning unless the dumped system was operating under the MFT subtasking option.

ECB hhhhhhhh
If the dumped system was operating under the MFT subtasking option, this field displays the address of the event control block (ECB) that would have been posted by the supervisor's task termination routines had either normal or abnormal task termination been allowed to occur.

XTCB hhhhhhhh
The XTCB field in this TCB display is reserved for future use.

LP/FL hh hhhhhh
Priority and dump information on tasks that were operating under the subtasking option of MFT. The LP/FL field displays its data as follows:

byte 0
The limit priority of the task represented by the TCB being displayed.

byte 1
Dump information flags.

xxxx x... Bits 0 through 4 are reserved for future use.

.... .1.. Bit 5 set indicates that the task represented by the TCB being displayed was the top task in the tree of abnormally terminating tasks.

.... ..1.	Bit 6 set indicates that an abnormal termination dump had been completed.
....1	Bit 7 set indicates that the task represented by this TCB was enqueued on a dump data set.

byte 2
This byte contains more dump information flag bits. The meanings of these bits are:

1...	Bit 0 set indicates that at the time the system was dumped, an OPEN was in process for the dump data set.
.xxx x..x	Bits 1 through 4 and bit 7 are reserved for future use.
.... .1..	Bit 5 set indicates that the dump data set was open for the job step.
.... ..x.	Bit 6 indicates the type of dump data set. The possible setting are:
.... ..0.	SYSUDUMP data set.
.... ..1.	SYSABEND data set.

byte 3
This last byte of the LP/FL field shows abnormal termination flags as follows:

xxx. x.xx	Bits 0, 1, 2, 4, 6 and 7 are reserved for future use.
...1	Bit 3 set indicates that a valid message recursion occurred in ABEND.
.... .1..	Bit 5 set indicates that no abnormal termination dumps could be provided within the job step associated with the TCB being displayed.

RES hhhhhhhh
This field is reserved for future use.

STA hhhhhhhh
Internal STAE routine flags and the address of the STAE control block that was in effect at the time the dump was taken.

TCT hhhhhhhh
Information pertaining to the dumped system's timing control table (TCT). The TCT field is divided into the following two subfields:

byte 0
This byte is reserved for future use.

byte 1-3
If the system management facilities option was presented in the dumped system, these bytes contain the address of the dumped system's timing control table (TCT).

USR hhhhhhhh
This word is available to the user of the dumped system. It contains any information placed in it by the user.

DAR hhhhhhhh
The contents of this field were used by the damage assessment routine (DAR). Certain subfields displayed in this word were also used to control the dispatchability of the dumped task. The DAR field is divided into the following subfields.

byte 0
The first byte of the DAR field contains DAR flags. The flags are as follows:

1...	Bit 0 set indicates that primary DAR recursion occurred in the dumped system. The damage assessment routine failed while writing a main storage image dump.
.1..	Bit 1 set indicates that secondary DAR recursion occurred in the dumped system. The damage assessment routine failed while attempting to reinstate a failing partition.
..1.	Bit 2 set indicates that only the dump capability of the damage assessment routine was requested.
...x	Bit 3 is reserved for future use.
.... 1...	Bit 4 set indicates that the system error task is failing. The DAR dump should not request any error recovery procedure (ERP) processing.
.... .xx.	Bits 5 and 6 are reserved for future use.
....1	Bit 7 set indicates that an SVC dump is executing for this task.

byte 1
Bytes 1 through 3 of the DAR display field are used to show the settings of secondary non-dispatchability flags bits. If any of the flags in this subfield were set, the primary non-dispatchability flag (the last bit in the FLG field) will also have been set and the task represented by this

TCB will have been non-dispatchable. The bit settings that may appear in byte 1 and their meanings are:

xx.. .... Bits 0 and 1 were set by the damage assessment routines. Their meanings are:

1... .... Bit 0 set indicates that the task represented by the TCB being displayed was flagged temporarily non-dispatchable.

.1.. .... Bit 1 set indicates that the task represented by this TCB was deemed permanently non-dispatchable.

..xx .... Bits 2 and 3 are recovery management support and system error recovery flags. Their meanings are:

..1. .... Bit 2 set indicates that the task represented by this TCB was flagged temporarily non-dispatchable.

...1 .... Bit 3 set indicates that the task represented by the TCB being displayed was deemed permanently non-dispatchable.

.... x... Bit 4 is reserved for future use.

.... .1.. Bit 5 set indicates that this task is temporarily non-dispatchable. Time services have been requested and the time-of-day clock is still inoperative.

.... ..xx Bits 6 and 7 are reserved for future use.

byte 2

1... .... Bit 0 indicates that at the time the dumped system was active, ABDUMP was processing. The setting of this flag bit has meaning only if the dumped system was operating with the subtasking option of MFT.

.x.. .... Bit 1 is reserved for future use.

..1. .... Bit 2 set indicates that this task is non-dispatchable. An SVC dump is executing for another task.

..x xxx. Bits 3 through 6 are reserved for future use.

....1 Bit 7 set indicates that at the time the system was dumped, the dump data set was in the process of being opened.

byte 3

1... .... The setting of this first bit has meaning only if the dumped system was operating with the MFT subtasking option. If this bit is set, the indication is that the task represented by the TCB being displayed was terminated.

.1.. .... Bit 1 set indicates that had the dumped MFT system, operating with the subtasking option, been allowed to continue processing without intervention by the dump program, the task represented by this TCB would have been terminated by ABEND.

..xx xxxx Bits 2 through 7 are reserved for future use.

RES hhhhhhhh
Reserved for future use.

JSCB hhhhhhhh
Contains the address of the job step control block.

MFT ACTIVE REQUEST BLOCK (RB) FORMATTING: Request blocks (RBs) were used by the dumped system's supervisor to maintain information concerning a task. RBs associated with the task identified in the heading lines at the top of the dump page and in the preceding TCB display, are listed in the portion of the dump listing labeled "ACTIVE RBS". Information on each RB associated with the task is formatted as shown below:

PRB
LPRB
SVRB hhhhhh
SIRB
IRB

Each RB display is preceded by a field that indicates the type and starting address of the RB being displayed. The five types of RBs that may be displayed under an MFT task are:

PRB
program request block

LPRB
loaded program request block

SVRB

supervisor request block (SVRBs may be divided into two categories; type 2 for resident routines and type 3 or 4 for transient routines).

SIRB

system interrupt request block

IRB

interruption request block

The type acronym for each RB is displayed in the first portion of the field. The starting address of the indicated request block appears in the last portion of the field. The contents of certain fields in the body of the formatted RB display are dependent upon the type of RB being displayed. Variations in display field usage are noted in the descriptions of the fields in which they occur.

NM cccccccc

The variations associated with the usage of this field are:

- PRBs and LPRBs use this field to display the name of the program they represented.

- SVRBs display the SVRB type in this field.

- SIRBs use this field to present the eight-character name of the error routine that was occupying the supervisor transient area at the time the dump was taken.

- IRBs display meaningful information in this field only if the timer was being used. If this was the case, the first character in this field represents the setting of the timer flags. The remainder of the NM field is meaningless.

SZ/STAB hhhhhhhh

This field displays two data elements; RB size information and STAB flag bit settings. This field is subfielded as follows:

bytes 0-1

The number of contiguous doublewords that were occupied by the request block, the associated program (if applicable), and associated supervisor work areas. If a program extent list was present, the program size is not included in this figure.

byte 2

STAB flag bit settings. The meaning of these flags are depends upon the type of request block being displayed. These flags are presented, by RB type, below:

PRB

The following bit settings are applicable to program request block displays:

0000 .... indicates that the program represented by this PRB was not loaded by a LOAD macro instruction; nor did it have minor entries identified by an IDENTIFY macro instruction.

0001 .... indicate that the program represented by this PRB was not loaded by a LOAD macro instruction but did have minor entries identified by an IDENTIFY macro instruction.

.... xx.. Bit 4 and 5 have no meaning in PRB displays.

.... ..1. indicates that the program represented by this PRB was hierarchy block loaded and that a program extent list existed.

....1 indicates that the program module represented by this PRB was refreshable.

LPRB

Loaded program request blocks being displayed may have the following bit settings in this byte:

0010 .... indicates that the program represented by this LPRB was not loaded by a LOAD macro instruction; nor did it have minor entries identified by an IDENTIFY macro instruction.

0011 .... indicates that the program represented by this LPRB was not loaded by a LOAD macro instruction but did have minor entries identified by an IDENTIFY macro instruction.

1110 .... indicates that this LPRB describes a minor entry identified by an IDENTIFY macro instruction.

.... xx.. Bits 4 and 5 have no meaning in LPRB displays.

.... ..1. indicates that the program represented by this LPRB was hierarchy block loaded and that a program extent list existed.

....1 indicates that the program module represented by this LPRB was refreshable.

SVRB

Supervisor request blocks display the following bit settings in this subfield:

1100 .... indicates that the program represented by this SVRB is a type 2 SVC routine that had not been loaded at the time the dump was taken.

1101 .... indicates that the program represented by this SVRB is a type 3 or SVC routine that had been loaded.

.... 1... indicates that the type 3 or 4 SVC routine was resident.

.... .1.. indicates that while the dumped system was active, a checkpoint could have been taken in a user exit from the SVC routine represented by this SVRB.

.... ..xx bits 6 and 7 have no meaning in SVRB displays.

SIRB

The flag bit setting applicable to supervisor interrupt request block displays is as follows:

1000 .... indicates that the RB being displayed is a supervisor interrupt request block (SIRB).

.... xxxx bits 4 through 7 have no meaning in SIRB displays.

IRB

Interrupt request block displays use these flag bits in the following manner.

0100 .... indicates that the RB being displayed is an interrupt request block (IRB).

.... xxxx bits 4 through 7 have no meaning in IRB displays.

byte 3

The last byte of the SZ/STAB field displays more status and attribute flags. The possible settings for this subfield and their meanings are:

1... .... Bit 0 set indicates that the WT-LNK field in this RB display contains, in its last three bytes, the address of the TCB to which this request block is linked.

.1.. .... Bit 1 set indicates that at the time the dumped system was active, the program associated with the RB being displayed was active.

..1. .... Bit 2 set indicates that had the dumped system been allowed to continue processing without intervention by the dump program, general registers 2 through 14 would have been restored from this RB's general register save area, displayed on the following two lines. The setting of this bit is valid only for IRB, SIRB and SVRB displays.

...1 .... Bit 3 set indicates that the program module represented by this request block was reenterable or reusable.

.... xx.. Bits 4 and 5 are used only in IRB or LPRB displays. The settings of these bits and their meanings are:

.... 00.. This setting indicates that the IRB being displayed had no interrupt queue elements (IQEs) associated with it.

.... 01.. This setting indicates that the IRB being displayed had associated with it interrupt queue elements that were request queue elements (RQEs).

.... 10.. This setting indicates that the request block being displayed is a dummy LPRB, in a partition that represents a program in the reenterable load module area. The LPRB for the program is in the reenterable load module area.

.... 11.. This setting indicates that the IRB being displayed had interrupt queue elements associated with it that were not request queue elements (RQEs).

.... ..1. Bit 6 set indicates that when the dumped system was active, request block storage was to have been freed when the program returned.

....x Bit 3 indicates wait request conditions. The meanings of the two possible settings for this bit are:

....0 Bit 7 not set indicates that the request had to wait for a single event or for all of a number of events.

....1 Bit 7 set indicates that the request had to wait for a number of events. This number of events was less than the total number of events that were waiting.

USE/EP hhhhhhhh

The USE/EP field, as indicated by the field identifier, displays two data elements. These are shown in the following format:

byte 0

The first byte of this field contains the use count that was applied to the program module represented by the request block being displayed. This use count was calculated by subtracting the number of invocations of the DELETE macro instruction from the number of times the LOAD macro instruction was used.

byte 1-3

The second portion of the USE/EP field displays the address of the entry point of the module represented by this request block.

PSW hhhhhhhh hhhhhhhh

The two words of the PSW field display to the user the dumped system's old program status word. If the dumped system had been allowed to continue processing without interruption by the dump program, operation would have resumed on this PSW.

Q hhhhhhhh

The information displayed in this field depends upon type of RB being displayed. The contents of this display field are described below, by RB type:

- PRBs and LPRBs use this field to display the address of an LPRB describing an entry that was identified via the IDENTIFY macro instruction.

- SVRBs representing type 3 or 4 SVCs use this field to indicate the size of the program they represent in bytes.

- SIRBs and IRBs display in this field the address of a 12- or 16-byte request element.

WT-LNK hhhhhhhh

This field displays information pertaining to wait counts and request block linkages. The field is divided into the following two subfields:

byte 0

The number of requests that were pending at the time the dump was taken (wait count).

byte 1-3

The address of the next request block on the RB queue. If the RB being displayed was the last request block on the queue, this field shows the address of the task control block (TCB) that enqueued this RB.

RG 0-7 and RG 8-15

The sixteen-word register save area appears only after IRB, SIRB or SVRB displays. These two lines display the contents of general registers 0 through 15 as they were stored in the request block.

<u>MFT PROBLEM PROGRAM BOUNDARIES INFORMATION</u>: Each task operating under the MFT option of the operating system was assigned a main storage partition in which to operate. If

the system configuration included 2361 Large Core Storage, partitions may have included area from both hierarchy 0 (main storage) and hierarchy 1 (low speed main storage). If 2361 Large Core Storage was not available or was not used, hierarchy 1 pointers were set to zero. Each MFT task displays in its dump listing the limits of the partition in which it operated. This display is presented under the heading "P/P BOUNDARIES" (problem program boundaries) in the following format:

HIER 0 hhhhhhhh
: The starting address of the problem program's hierarchy 0 partition.

TO hhhhhhhh
: The ending address of the problem program's hierarchy 0 main storage partition.

HIER 1 hhhhhhhh
: The starting address of the problem program's hierarchy 1 partition. If this field contains zeros, the indication is that 2361 Large Core Storage was either not available or not utilized by this task.

TO hhhhhhhh
: This last field indicates the high limit of the problem program's hierarchy 1 partition if one was used. If this field contains zeros, either 2361 Large Core Storage was not available or it was not used by this task.

<u>MFT LOAD LIST FORMATTING</u>: A load list was maintained by the dumped system's supervisor in order to keep track of the load modules that were in main storage and the area of main storage each occupied. A load list created by an MFT supervisor is composed of loaded request blocks (LRBs) and loaded program request blocks (LPRBs). A formatted listing of the dumped MFT system's load list appears as follows:

LRB
LPRB hhhhhhhh
: The type of request block being displayed and its starting address.

NM cccccccc
: The eight-character name of the program module represented by the request block being displayed.

SZ hhhhhh
: The number of contiguous double words that were occupied by the request block, the associated program (if applicable) and associated supervisor work areas. If a program extent list was present, the program size is not included.

USE/EP hhhhhhhh
: Use count and entry point address as follows:

byte 0
: The use count that was applied to the program module represented by the request block being displayed. This use count was calculated by subtracting the number of times the DELETE macro instruction was issued from the number of times the LOAD macro instruction was used.

byte 1-3
: The address of the entry point of the program module named in the NM field of this RB display line.

<u>MFT JOB PACK QUEUE FORMATTING</u>: A job pack area queue was maintained by the dumped system's supervisor for each job step that used a program not in the resident reenterable load module area. A job pack queue created by an MFT supervisor consists of loaded request blocks (LRBs), loaded program request blocks (LPRBs) and FINCH request blocks (FRBs). A formatted job pack area queue display appears as follows:

LRB
LPRB hhhhhh
FRB
: The type of request block being displayed and its starting address.

NM cccccccc
: The eight-character name of the module represented by the request block being displayed.

SZ hhhhhh
: The number of contiguous doublewords that were occupied by the request block, the associated program (if applicable) and associated supervisor work areas. If a program extent list was present, the program size is not included.

USE/EP hhhhhhhh
XRWTL
: The usage of this display field is dependent upon the type of request block being displayed:

USE/EP
: is used for LRBs and LPRBs and displays the use count and entry point address as follows:

byte 0
: The use count that was applied to the program module represented by the request block being displayed. This use count was calculated by subtracting the number of times the DELETE macro

instruction was issued from the number of times the LOAD macro instruction was used.

bytes 1-3
The address of the entry point of the program module named in the NM field of this display line.

XRWTL
is used for FRBs and shows the starting address of the wait list element.

XRREQ hhhhhhhh
This field appears only in FRB displays, and shows the address of the TCB representing the task on whose behalf this FRB was constructed.

XRTLPRB hhhhhhhh
This field appears only in FRB displays and shows the starting address of the area of main storage that was acquired by the FETCH routine for the module identified by the NM field of this line.

MFT DATA EXTENT BLOCK (DEB) FORMATTING: Data extent blocks (DEBs), describing a data set's external storage requirements, were queued to those task control blocks (TCBs) that represented tasks requiring auxiliary storage input/output processing. External storage information, taken from each DEB, is formatted as shown below:

DEB hhhhhh
The starting address of the basic section of the DEB being displayed.

APPENDAGES
The word "appendages" informs the user that the five named fields on this line contain information taken from the appendage vector table preceding the DEB being displayed. The named fields appearing on the rest of this line are:

END OF EXT hhhhhh
The entry point of the end-of-extent appendage routine.

SIO hhhhhh
The entry point of the start I/O appendage routine.

PCI hhhhhh
The entry point of the program-controlled-interruption appendage routine.

CH END hhhhhh
The entry point of the channel-end appendage routine.

AB END hhhhhh
The entry point of the abnormal-end appendage routine.

PFX hhhhhhhh hhhhhhhh hhhhhhhh
The second line of a DEB display contains information taken from the prefix section of the DEB being displayed. The area is subdivided as follows:

byte 0
The first byte of the prefix area contains the contents of the I/O support work area. This area is used only by DEBs dealing with direct access storage devices.

bytes 1-7
The next seven bytes of the DEB prefix section are used by DEBs associated with direct access storage device functions. This subfield displays the data set control block's (DSCB) address used by I/O support. The address is expressed in the following format:

bytes 1 and 2	the bin (cell) number.
bytes 3 and 4	the cylinder address.
bytes 5 and 6	the track address.
byte 7	the record number.

bytes 8-11
The third word of the PFX field contains the data control block (DCB) modification mask that was used by I/O support.

byte 12
The length of the DEB in double words.

bytes 13-15
The remainder of the DEB prefix section is reserved for future use.

TCB hhhhhhhh
This field marks the beginning of the basic section of the data extent block. The TCB field is divided into two subfields as follows:

byte 0
The number of subroutines for which a LOAD macro instruction was issued during the execution of the OPEN executor routines.

bytes 1-3
The starting address of the task control block to which this DEB was enqueued.

NDEB hhhhhhhh

byte 0
The overall length of a data extent block includes the length of a

variable length access method dependent section. The first byte of the NDEB field expresses the length of the access method dependent section in bytes. If the access method was BDAM, this indicator is expressed as a number of full words.

bytes 1-3
The last portion of the NDEB field displays the starting address of the basic section of the next DEB on the task's queue. If this DEB was the last on the queue, the content of this field is the starting address of the TCB that enqueued this DEB.

ASYN hhhhhhhh
This field contains data set status flags and the address of the associated IRB:

byte 0
The first byte of the ASYN field contains data set status flags. These flags have the following meanings:

xx.. .... Bits 0 and 1 indicate the data set's disposition. The possible settings are:

01.. .... This setting indicates that the disposition was OLD.

10.. .... This setting indicates that the disposition of the data set was MOD (modify).

11.. .... This setting indicates that the disposition was NEW.

..1. .... Bit 2 set indicates that an end-of-volume (EOV) or end-of-file (EOF) condition had been encountered.

...1 .... The setting of bit 3 has one of two meanings depending upon the external storage medium. For disk, this indicator reflects a release of unused external storage. For tape, this indicator means that an emulator tape with second generation format was being used.

.... 1... Bit 4 set is a data control block (DCB) modification indicator.

.... .1.. Bit 5 set has two meanings, depending upon the auxiliary storage recording medium. For disk, the setting of bit 5 indicates that a split cylinder was encountered. For tape, this flag indicates that an emulator tape with possible mixed parity records was used.

.... ..1. Bit 6 set indicates the use of nonstandard labels.

....1 Bit 7 set indicates that reduced error recovery procedures were used on magnetic tapes containing the data set represented by this DEB.

bytes 1-3
The last portion of the ASYN field shows the starting address of the IRB that was associated with asynchronous appendage exit scheduling.

SPRG hhhhhhhh
This field contains information on I/O processing methods and the system PURGE routine.

byte 0
The first byte of this field contains flags that indicate the method of input/output processing and the disposition of the data set that was to have been performed when an end-of-volume condition occurred. These flag settings are:

1... .... Bit 0 was set by ABEND. The setting of this bit indicates that the data set associated with this DEB was a SYSABEND or SYSUDUMP data set.

.0.. .... Bit 1 is always zero.

..xx .... Bit 2 and 3 show the end-of-volume disposition procedure. The values for this flag are:

..01 .... REREAD

..11 .... LEAVE

.... xxxx The last half of this byte contains flags that indicate the type of input/output processing that was performed on the data set represented by this DEB. The values for this flag are:

.... 0000 INPUT

.... 1111 OUTPUT

.... 0011 INOUT

.... 0111 OUTIN

.... 0001 RDBACK

.... 0100 UPDAT

byte 1
The quiesce count. The byte is associated with the system PURGE routines (SVC 16), and indicates the

number of auxiliary storage devices that were executing the user's channel programs.

bytes 2-3
Reserved for future use.

UPRG hhhhhhhh
The UPRG field contains extent information and data used by the user's purge routines. This field is divided into the following two subfields:

byte 0
The number of extents that were specified in the DSCBs associated with this DEB.

bytes 1-3
The address of the first input/output block (IOB) in the user's purge chain.

PLST hhhhhhhh
Task priority and supervisor purge information are contained in this field. This field is formatted as follows:

byte 0
The priority of the task under which this DEB was enqueued.

bytes 1-3
The starting address of a parameter list that was used to locate the purge event control block (ECB) for a supervisor purge request.

DCB hhhhhhhh
The DCB field contains three data elements. These are displayed in the format given below:

byte 0

xxxx	The storage protection key that was associated with the task under which this DEB was enqueued.
.... 1111	A hexadecimal "F" in bits 4 through 7 of this field identify this control block as a data extent block (DEB).

bytes 1-3
The starting address of the data control block (DCB) that was associated with this DEB.

AVT hhhhhhhh
The AVT field displays two DEB data elements and is subfielded as follows:

byte 0
The DEB extent scale that is used to determine the size of the device dependent section of this DEB. For direct access devices, a 4 is displayed in this subfield. For a nondirect access device or a communication device, a 2 is displayed.

bytes 1-3
In most cases, the last portion of the AVT field shows the starting address of the appendage vector table preceding this DEB. This table of appendage routine addresses appears on the first line of this DEB's display.

OP-UCB hhhhhhhh
The contents of this field have meaning only when the DEB being displayed describes a data set that was assigned to a unit record or magnetic tape device. This information is formatted from the device dependent section of the DEB. The OP-UCB field is subfielded as follows:

byte 0
This first subfield is applicable only to data sets assigned to magnetic tape devices, and shows the SET MODE operation code. For a data set that was assigned to a unit record device, this subfield is reserved.

bytes 1-3
The starting address of the unit control block (UCB) associated with the data set described by the DEB being displayed.

The following four fields are present only for data sets assigned to the IBM 3525 Card Punch for multi-function. The information is formatted as shown below:

UCB hhhhhhhh
byte 0
The device modifier field (not used for the 3525).
bytes 1-3
The starting address of the unit control block (UCB) associated with the data set described by the DEB being displayed.

RDRDCB hhhhhhhh
The starting address of the data control block (DCB) for the read associated data set.

PCHDCB hhhhhhhh
The starting address of the data control block (DCB) for the punch associated data set.

WTRDCB hhhhhhhh
The starting address of the data control block (DCB) for the print associated data set.

The final portion of a DEB display shows information pertaining to a data set that was assigned to a direct access device. This information, taken from the DEB's device dependent section, is arranged in columnar format with a line for each extent. The information is formatted as shown below:

FM-UCB hhhhhhhh
The first column displays two data elements and is formatted as follows:

byte 0
The device modifier showing the file mask.

bytes 1-3
The starting address of the unit control block (UCB) that was associated with the data extent.

START hhhhhhhh
The address of the beginning of the direct access device extent. The first four characters represent the cylinder address and the last four characters represent the track address.

END hhhhhhhh
The address of the end of the data extent. Cylinder and track references are formatted as in the extent beginning address, described above.

TRKS hhhh
The number of direct access tracks bounded by the starting and ending addresses shown in the previous two columns.

MFT TASK INPUT/OUTPUT TABLE (TIOT) FORMATTING: A task input/output table (TIOT) was constructed for each task in the dumped system by MFT job management routines. This table contained primary pointers to control blocks used by I/O support routines. As the functions of several TIOT fields were dependent upon the state of associated external storage devices, multiple definitions may apply. The TIOT that was constructed in the dumped MFT system is formatted as shown.

TIOT hhhhhh
The starting address of the task input/output table being displayed.

JOB cccccccc
The eight-character name of the job for which this TIOT was constructed.

STEP cccccccc
The eight-character name specified in the label field of the EXEC JCL statement associated with this job step.

PROC cccccccc
If the job step for which this TIOT was constructed was invoked from a cataloged procedure, the procedure name, as contained in the EXEC JCL statement, is displayed in this field.

Each data set associated with the indicated task is represented by a separate DD entry that is included in the TIOT. Each TIOT entry is displayed on a separate line in columnar format. The use and meaning of each column is given below:

OFFSET hhhh
The offset of this DD entry from the beginning of the TIOT in hexadecimal.

LN-STA hhhhhhhh

byte 0
The total length (including all device entries) in bytes of the DD entry being displayed on this line.

byte 1
Status byte A, one of three status bytes in a TIOT entry. The meanings of the status byte settings are:

x... .x..	Bits 0 and 5 indicate the tape label processing that was to have been performed. The meanings of the settings are:
0... .0..	Nonlabeled tape or an indication to bypass label processing.
0... .1..	Standard labels or standard user labels.
1... .0..	Nonstandard labels.
.1..	The setting of status bit 1 has two meanings, depending upon the processing phase that had been reached at the time the system was dumped. During allocation processing, the setting of this bit indicates that this entry represents a split cylinder primary space allocation DD. If the dump was taken during step termination processing, the setting of this bit indicates that no unallocation of space was necessary.

..1. .... The setting of status bit 2 works under the same philosophy as status bit 1. During allocation processing, the setting of this bit indicates that this entry represents a split cylinder secondary space allocation DD. If the dump was taken during step termination processing, the indication was one of rewinding with no unload.

...1 .... Bit 3 set indicates that this DD entry represents a JOBLIB.

.... 1... Bit 4 set indicates that direct access device space management was deemed necessary.

.... ..1. The setting of bit 6 specifies that the tape volume was to have been rewound and unloaded.

....1 The setting of bit 7 specifies that the tape volume was to have been rewound.

byte 2

The third byte of this column has meaning only during the allocation phase. This displays the number of devices that were requested by the data set represented by the TIOT entry displayed on this line.

byte 3

The last byte of the LN-STA field displays a TIOT field that had meaning at two points during the processing of this task. During the allocation process, this field contained a link to the appropriate prime split, unit affinity, volume affinity or suballocate TIOT entry. After CLOSE processing, this byte was used as follows:

1... .... The setting of bit 0 indicates that the data set represented by this DD entry was a SYSOUT data set that contained data.

.xxx xxxx Bits 1 through 7 are reserved for future use.

DDNAME cccccccc

The eight character DD name associated with the TIOT entry being displayed.

TTR-STC hhhhhhhh

The first three bytes of this column display the relative track address (TTR) of the job file control block (JFCB) associated with this entry.

STB-UCB hhhhhhhh

The last column in a TIOT display contains information taken from the one-word device entries that are appended to each TIOT entry. One TIOT device entry exists for each allocated device. This display field shows this information in the following format:

byte 0

Status byte B. The status bits have the following meanings:

1... .... Bit 0 set indicates that the data set associated with this line of the TIOT display was present on the device represented by this TIOT device entry.

.1.. .... Bit 1 set indicates that the data set associated with this line of the TIOT display would have used the device represented by this TIOT device entry.

..1. .... Bit 2 set indicates that the device represented by this device entry violated separation.

...1 .... Bit 3 set indicates that a volume serial number was present.

.... 1... Bit 4 set indicates that a setup message was required.

.... .x.. Bit 5 indicates the device disposition that would have taken place had the dumped system been allowed to continue processing this task. The settings for this bit are:

.... .0.. Indicates that if the volume was required to be unloaded, the volume was to have been deleted.

.... .1.. Indicates that if the volume was required to be unloaded, the unloaded volume was to have been retained.

.... ..1. Bit 6 indicates that an unload requirement had been made.

....1 Bit 7 set indicates that a load or label verification requirement had been made.

bytes 1-3

The address of the UCB that was used in all cases except when the device was a 2321 data cell drive. For a 2321, this address is that of the description in the UCB of the cell in the bin.

TSO System Block Formatting

The TSO control blocks are divided into two groups: system and user. The control blocks are discussed in the order in which they appear when both groups are requested. Some control blocks are formatted and printed when either group is requested.

An example of a TSO system and user dump listing is shown in Figure 24.

TIME SHARING COMMUNICATIONS VECTOR TABLE (TSCVT) FORMATTING: The time sharing communications vector table is a secondary CVT to meet the time sharing requirements. 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 is zero.

TSCVT hhhhhh
The address of this time sharing communications vector table.

TJB hhhhhhhh
The address of the time-sharing job block (TJB) table. This table contains all of the TJBs allowed TSO users. The first TJB is for the terminal job identification (TJID) equal to zero.

RCB hhhhhhhh
The address of the region control block (RCB) table. It is an indexed table containing one RCB 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.

RPT hhhhhhhh
The address of the reference point table (RPT). It is used by the terminal input output coordinator (TIOC).

FLG hhhh
These flags indicate functions requested from the time sharing control task (TSC).

byte 0

```
1... .... TSCSWPND:  Bit 0 set
          indicates that a swap has
          ended.

.1.. .... TSCSWPBG:  Bit 1 set
          indicates that a swap should
          be started.

..1. .... TSCLOGON:  Bit 2 set
          indicates that a logon is
          required.

...1 .... TSCDISC:  Bit 3 set
          indicates that a disconnect
          is required.

.... xxxx Bits 4 through 7 are
          reserved for future use.
```

byte 1
Reserved for future use.

FL1 hhhh
These flags indicate atypical functions required by the time sharing control task (TSC).

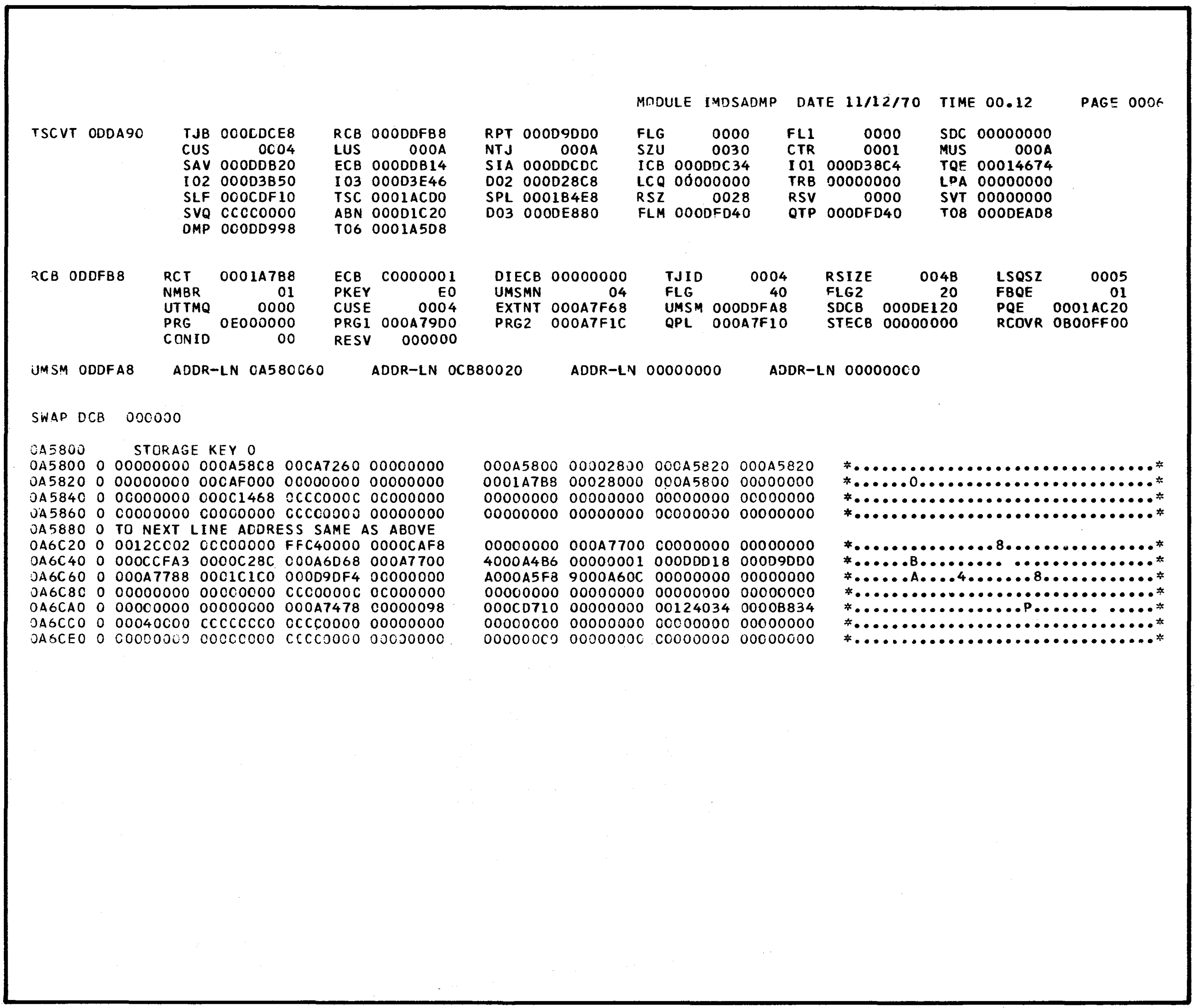
byte 0

```
1... .... TSCSSTOP:  Bit 0 indicates
          that a system stop has been
          requested and the time
          sharing system is in the
          process of stopping.

.1.. .... TSCRSTOP:  Bit 1 indicates
          that a region stop has been
          requested.

..1. .... TSCASTOP:  Bit 2 is the
          ABEND-STOP flag.  When set,
          it indicates to the time
          sharing control task (TSC)
          that time sharing should be
          stopped.  This flag is set
          by (1) the TSO/RMS interface
          return when a machine check
          occurs in TCAM or (2) the
          TCAM STAE exit when TCAM
          abnormally terminates.

...x xxxx Bits 3 through 7 are
          reserved for future use.
```

byte 1
Reserved for future use.

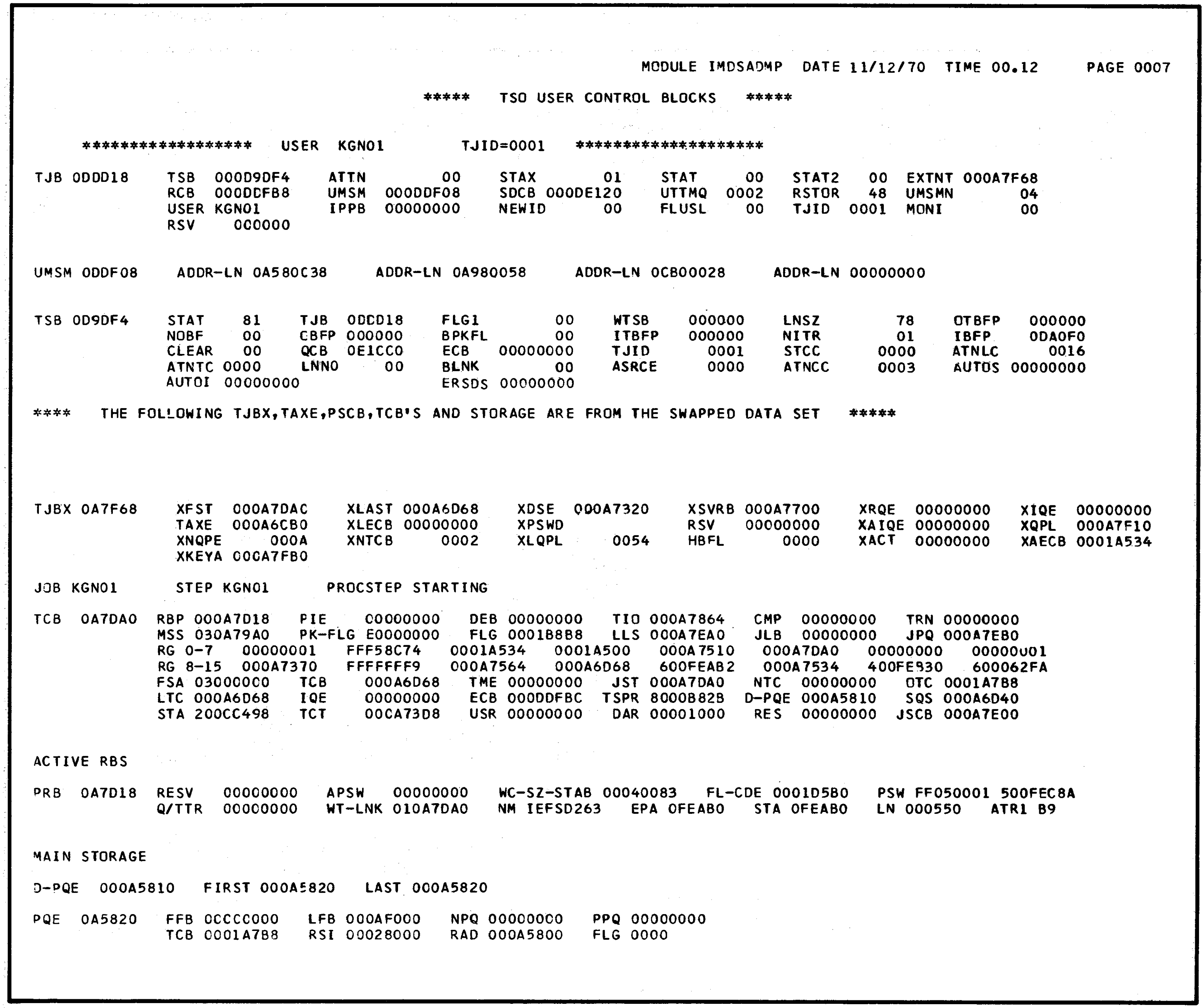
SDC hhhhhhhh
The address of the first data control block (DCB) for swap data sets.

CUS hhhh
A count of the current TSO users logged onto the system. For additional users to be logged onto the system, this number must be less than the value in LUS.



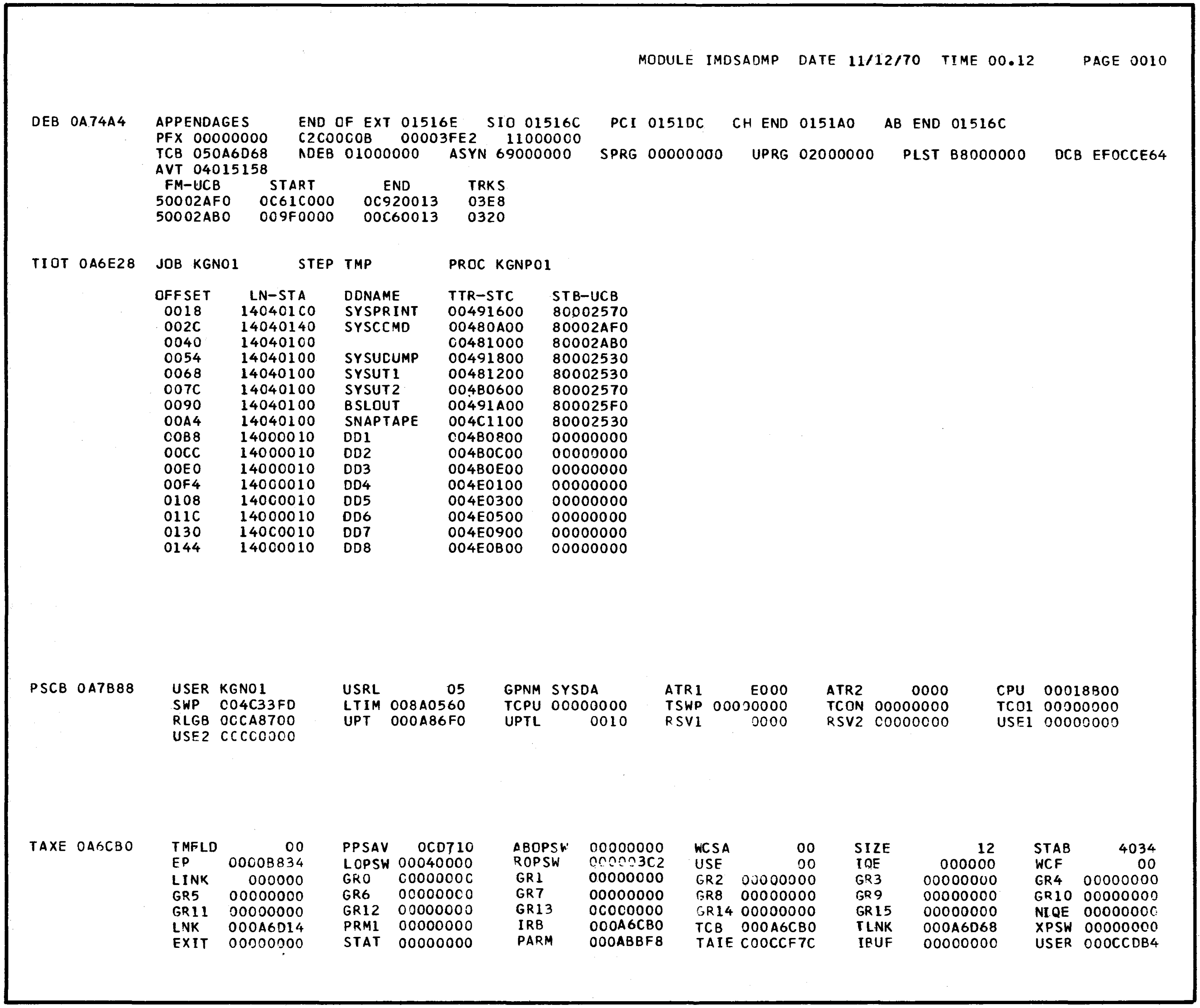
```
                                                                        MODULE IMDSADMP  DATE 11/12/70  TIME 00.12     PAGE 0006

TSCVT 0DDA90       TJB 000CDCE8     RCB 000DDFB8     RPT 000D9DD0     FLG      0000    FL1      0000    SDC 00000000
                   CUS     0C04     LUS     000A     NTJ     000A     SZU      0030    CTR      0001    MUS     000A
                   SAV 000DDB20     ECB 000DDB14     SIA 000DDCDC     ICB 000DDC34    I01 000D38C4    TQE 00014674
                   I02 000D3B50     I03 000D3E46     D02 000D28C8     LCQ 00000000    TRB 00000000    LPA 00000000
                   SLF 000CDF10     TSC 0001ACD0     SPL 0001B4E8     RSZ      0028    RSV      0000    SVT 00000000
                   SVQ CCCC0000     ABN 000D1C20     D03 000DE880     FLM 000DFD40    QTP 000DFD40    T08 000DEAD8
                   DMP 000DD998     T06 0001A5D8


RCB 0DDFB8       RCT    0001A7B8     ECB   C0000001     DIECB 00000000     TJID       0004     RSIZE       004B     LSQSZ        0005
                 NMBR         01     PKEY        E0     UMSMN       04     FLG          40     FLG2          20     FBQE           01
                 UTTMQ      0000     CUSE      0004     EXTNT 000A7F68     UMSM 000DDFA8     SDCB   000DE120     PQE    0001AC20
                 PRG    0E000000     PRG1 000A79D0     PRG2  000A7F1C     QPL  000A7F10     STECB 00000000     RCOVR 0B00FF00
                 CONID        00     RESV    000000

UMSM 0DDFA8     ADDR-LN 0A580C60     ADDR-LN 0CB80020     ADDR-LN 00000000     ADDR-LN 000000C0


SWAP DCB  000000

0A5800      STORAGE KEY 0
0A5800 0 00000000 000A58C8 00CA7260 00000000     000A5800 00002800 000A5820 000A5820    *................................*
0A5820 0 00000000 00CAF000 00000000 00000000     0001A7B8 00028000 000A5800 00000000    *......O.........................*
0A5840 0 00000000 000C1468 0CCC000C 0C000000     00000000 00000000 00000000 00000000    *................................*
0A5860 0 C0000000 C0000000 CCC00000 00000000     00000000 00000000 00000000 00000000    *................................*
0A5880 0 TO NEXT LINE ADDRESS SAME AS ABOVE
0A6C20 0 0012CC02 0CC00000 FFC40000 0000CAF8     00000000 000A7700 C0000000 00000000    *...............8................*
0A6C40 0 000CCFA3 0000C28C C00A6D68 000A7700     4000A4B6 00000001 000DDD18 000D9DD0    *......B.........  ..............*
0A6C60 0 000A7788 0001C1C0 000D9DF4 0C000000     A000A5F8 9000A60C 00000000 00000000    *......A....4.......8............*
0A6C80 0 00000000 00000000 CC00000C 0C000000     00000000 00000000 00000000 00000000    *................................*
0A6CA0 0 00000000 00000000 000A7478 C0000098     000CD710 00000000 00124034 0000B834    *..................P.......  ....*
0A6CC0 0 00040000 CCCCCCC0 0CCC0000 00000000     00000000 00000000 00000000 00000000    *................................*
0A6CE0 0 C0000000 000C0000 CCCC0000 00000000     000000C0 0000000C C0000000 00000000    *................................*
```

Figure 24. Sample of TSO Control Block Format (Part 1 of 3)



```
                                                                 MODULE IMDSADMP  DATE 11/12/70  TIME 00.12     PAGE 0007
                                          *****   TSO USER CONTROL BLOCKS   *****

     ******************   USER  KGN01        TJID=0001   ********************

TJB 0DDD18    TSB  000D9DF4    ATTN        00    STAX       01    STAT     00    STAT2   00  EXTNT 000A7F68
              RCB  000DDFB8    UMSM  000DDF08    SDCB 000DE120    UTTMQ  0002    RSTOR   48  UMSMN       04
              USER KGN01       IPPB  00000000    NEWID      00    FLUSL    00    TJID  0001  MONI        00
              RSV    000000

UMSM 0DDF08    ADDR-LN 0A580C38     ADDR-LN 0A980058     ADDR-LN 0CB00028     ADDR-LN 00000000

TSB 0D9DF4    STAT     81    TJB  0DCD18    FLG1          00    WTSB      000000    LNSZ         78    OTBFP    000000
              NOBF     00    CBFP 000000    BPKFL         00    ITBFP     000000    NITR         01    IBFP     0DA0F0
              CLEAR    00    QCB  0E1CC0    ECB     00000000    TJID        0001    STCC       0000    ATNLC      0016
              ATNTC 0000     LNNO     00    BLNK          00    ASRCE       0000    ATNCC      0003    AUTOS  00000000
              AUTOI 00000000                ERSDS 00000000
****   THE FOLLOWING TJBX,TAXE,PSCB,TCB'S AND STORAGE ARE FROM THE SWAPPED DATA SET   *****



TJBX 0A7F68    XFST  000A7DAC    XLAST 000A6D68    XDSE  000A7320    XSVRB 000A7700    XRQE  00000000    XIQE  00000000
               TAXE  000A6CB0    XLECB 00000000    XPSWD             RSV   00000000    XAIQE 00000000    XQPL  000A7F10
               XNQPE     000A    XNTCB     0002    XLQPL      0054   HBFL      0000    XACT  00000000    XAECB 0001A534
               XKEYA 000A7FB0

JOB KGN01      STEP KGN01      PROCSTEP STARTING

TCB  0A7DA0  RBP 000A7D18   PIE     00000000   DEB 00000000   TIO 000A7864   CMP   00000000   TRN 00000000
             MSS 030A79A0   PK-FLG E0000000    FLG 0001B8B8   LLS 000A7EA0   JLB   00000000   JPQ 000A7EB0
             RG 0-7    00000001   FFF58C74   0001A534   0001A500   000A7510   000A7DA0   00000000   00000001
             RG 8-15   000A7370   FFFFFFF9   000A7564   000A6D68   600FEAB2   000A7534   400FE930   600062FA
             FSA 03000000   TCB     000A6D68   TME 00000000   JST 000A7DA0   NTC   00000000   OTC 0001A7B8
             LTC 000A6D68   IQE     00000000   ECB 000DDFBC  TSPR 8000B82B  D-PQE 000A5810   SQS 000A6D40
             STA 200CC498   TCT     00CA73D8   USR 00000000   DAR 00001000   RES   00000000  JSCB 000A7E00


ACTIVE RBS

PRB  0A7D18  RESV    00000000   APSW    00000000   WC-SZ-STAB 00040083   FL-CDE 0001D5B0   PSW FF050001 500FEC8A
             Q/TTR   00000000   WT-LNK 010A7DA0    NM IEFSD263   EPA 0FEAB0   STA 0FEAB0   LN 000550   ATR1 B9

MAIN STORAGE

D-PQE  000A5810   FIRST 000A5820   LAST 000A5820

PQE  0A5820   FFB 0CCCC000   LFB 000AF000   NPQ 00000000   PPQ 00000000
              TCB 0001A7B8   RSI 00028000   RAD 000A5800   FLG 0000
```

Figure 24. Sample of TSO Control Block Format (Part 2 of 3)



```
                                                                      MODULE IMDSADMP  DATE 11/12/70  TIME 00.12     PAGE 0010


DEB 0A74A4    APPENDAGES      END OF EXT 01516E   SIO 01516C    PCI 0151DC    CH END 0151A0    AB END 01516C
              PFX 00000000    C2C0000B    00003FE2    11000000
              TCB 050A6D68    NDEB 01000000    ASYN 69000000    SPRG 00000000    UPRG 02000000    PLST B8000000    DCB EF0CCE64
              AVT 04015158
               FM-UCB      START       END        TRKS
              50002AF0    0C61C000    0C920013    03E8
              50002AB0    009F0000    00C60013    0320


TIOT 0A6E28   JOB KGN01       STEP TMP        PROC KGNP01

              OFFSET    LN-STA     DDNAME     TTR-STC     STB-UCB
               0018    140401C0    SYSPRINT   00491600    80002570
               002C    14040140    SYSCCMD    00480A00    80002AF0
               0040    14040100               00481000    80002AB0
               0054    14040100    SYSUDUMP   00491800    80002530
               0068    14040100    SYSUT1     00481200    80002530
               007C    14040100    SYSUT2     004B0600    80002570
               0090    14040100    BSLOUT     00491A00    800025F0
               00A4    14040100    SNAPTAPE   004C1100    80002530
               00B8    14000010    DD1        004B0800    00000000
               00CC    14000010    DD2        004B0C00    00000000
               00E0    14000010    DD3        004B0E00    00000000
               00F4    14000010    DD4        004E0100    00000000
               0108    14000010    DD5        004E0300    00000000
               011C    14000010    DD6        004E0500    00000000
               0130    14000010    DD7        004E0900    00000000
               0144    14000010    DD8        004E0B00    00000000




PSCB 0A7B88    USER KGN01        USRL        05    GPNM SYSDA       ATR1      E000    ATR2      0000    CPU  00018B00
               SWP  C04C33FD     LTIM 008A0560    TCPU 00000000    TSWP 00000000    TCON 00000000    TC01 00000000
               RLGB 0CCA8700     UPT  000A86F0    UPTL      0010    RSV1      0000    RSV2 C0000000    USE1 00000000
               USE2 CCCC0000




TAXE 0A6CB0    TMFLD        00   PPSAV   0CD710    ABOPSW  00000000   WCSA        00   SIZE          12   STAB      4034
               EP     0000B834   LOPSW 00040000    ROPSW   000003C2   USE         00   IQE       000000   WCF         00
               LINK     000000   GR0   C000000C    GR1     00000000   GR2   00000000   GR3     00000000   GR4   00000000
               GR5    00000000   GR6   000000C0    GR7     00000000   GR8   00000000   GR9     00000000   GR10  00000000
               GR11   00000000   GR12  00000000    GR13    0C000000   GR14  00000000   GR15    00000000   NIQE  00000000
               LNK    000A6D14   PRM1  00000000    IRB     000A6CB0   TCB   000A6CB0   TLNK    000A6D68   XPSW  00000000
               EXIT   00000000   STAT  00000000    PARM    000ABBF8   TAIE  C00CCF7C   IBUF    00000000   USER  000CCDB4
```

Figure 24. Sample of TSO Control Block Format (Part 3 of 3)

LUS hhhh
The maximum number of TSO users that may be logged onto the system. For additional users to be logged onto the system, the value of LUS must be greater than the value in CUS. LUS cannot exceed the value in NTJ. LUS is set by the time sharing control task (TSC). This field is initially set to the same value as MUS; however, if TSO encounters I/O errors while swapping users in and out, the time sharing control task reduces this value to limit the number of TSO users.

NTJ hhhh
The number of time-sharing job blocks (TJBs) and terminal status blocks (TSBs) allocated when TSO was started. The dummy TJB for the terminal job identification (TJID) equal zero is not included. The value of LUS cannot exceed this number.

SZU hhhh
The number of bytes in the time sharing job block (TJB).

CTR hhhh
Contains the number of region control blocks (RCBs) allocated when TSO was started. This number cannot be increased after the TSO system is started.

MUS hhhh
The maximum number of users that may be logged onto a TSO system. This field is set by the START and MODIFY commands issued by the operator.

SAV hhhhhhhh
The beginning address of three 18-word save areas used by the time sharing control task (TSC), the time sharing interface program (TSIP), and the time sharing dispatcher.

ECB hhhhhhhh
The address of the table control block (TSECBTAB) which contains the event control blocks (ECBs) used to post the time sharing control task (TSC), the region control tasks (RCTs), and the terminal input output coordinator (TIOC).

SIA hhhhhhhh
The address of the time sharing interface area (TSIA).

ICB hhhhhhhh
The address of the time sharing interface control block (TSICB).

I01 hhhhhhhh
The address of the branch entry point IKJEAI01 in the time sharing interface program (TSIP).

TQE hhhhhhhh
The address of the timer queue element (TQE) used by TSO for time slicing.

I02 hhhhhhhh
The address of the entry point IKJEAI02 in the time sharing dispatcher.

I03 hhhhhhhh
The address of the entry point IKJEAI03 in the time sharing dispatcher.

D02 hhhhhhhh
The address of the entry point to the TSO driver routine (IKJEAD02), or the equivalent entry in a user written routine.

LCQ hhhhhhhh
The address of the first element in the logon communications queue.

TRB hhhhhhhh
The first address in the trace control block chain. This address is established and used by the statistics collection routine. It is set to zero by the time sharing control task (TSC).

LPA hhhhhhhh
The address of the first contents directory entry (CDE) in the time sharing link pack area.

SLF hhhhhhhh
The address of the system-initiated logoff routine.

TSC hhhhhhhh
The address of the task control block (TCB) for the time sharing control task (TSC).

SPL hhhhhhhh
The address of the start parameter list.

RSZ hhhh
The minimum number of 2K blocks for a region during logon.

RSV hhhh
Reserved for future use.

SVT hhhhhhhh
The contents of the SVC table entry used by the time sharing interface program (TSIP).

SVQ hhhhhhhh
The contents of the SVC table entry used by the TCAM/TIOC interface program.

ABN hhhhhhhh
The address of the out-of-main storage abnormal termination routine (IKJEAT07). The routine is resident in main storage.

D03 hhhhhhhh
The entry point address to the TSO driver MODIFY routine (IKJEAD03), or the equivalent entry point address in a user written routine.

FLM hhhhhhhh
The entry point address IKJEFLM for the system initiated logoff routine.

QTP hhhhhhhh
The entry point address IKJGGQT1 for the branch entry to the TCAM interface program (QTIP).

T08 hhhhhhhh
The entry point address to the TSO command routine (IJEAT08) for TSO dumps taken by the time sharing control task (TSC) TSO dumps.

DMP hhhhhhhh
The address of the TSO dump control block.

T06 hhhhhhhh
The TCB address of the TSO dump routine (IKJEAT06) for the time sharing control task (TSC) modify routine.

<u>TIME SHARING REGION CONTROL BLOCK (RCB) FORMATTING</u>: 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 control tasks region, they are contiguous, and they are created during initialization of the time sharing controller.

RCB hhhhhh
The address of the RCB.

RCT hhhhhhhh
The address of the task control block (TCB) for this region control task (RCT). The TCB contains the address of the partition queue element (PQE) that defines the region.

ECB hhhhhhhh
The event control block (ECB) on which this region control block (RCB) waits. This ECB must be posted before this region control task (RCT) can perform one of its functions.

DIECB hhhhhhhh
The event control block (ECB) that is posted upon completion of this region control task (RCT). The time sharing control task (TSC) waits for this ECB to be posted.

TJID hhhh
The terminal job identification (TJID) for the time sharing job currently executing in this region.

RSIZE hhhh
The number of 2K blocks in this region. It is set by the time sharing control task (TSC) when the time sharing system is started.

LSQSZ hhhh
The number of 2K blocks in the local system queue space (LSQS) for this region. It is set by the time sharing control task (TSC) when the time sharing system is started.

NMBR hh
The identification number assigned to this region.

PKEY hh
The protect key (PKEY) for the time sharing job currently executing in this region.

UMSMN hh
The number of entries in the main storage map which describes the main storage image that was initialized during logon.

FLG hh
This field contains the first byte of the region control block (RCB) flags. The flags indicate various functions to be performed by the region control task (RCT) and time sharing control task (TSC). These flags are set by the time sharing interface program (TSIP), the time sharing control task (TSC), and the terminal input/output coordinator (TIOC). These flags are tested and reset by the region control task (RCT) and the time sharing control task (TSC).

1... .... RCBFQO: Bit 0 is the quiesce flag. When set, this flag indicates that the current user of this region should be quiesced.

.1.. .... RCBFSO: Bit 1 is the swap out flag. When set, this flag indicates that the current user should be swapped out.

..1. .... RCBFSI: Bit 2 is the swap in flag. When set, this flag indicates that the current user of this region should be swapped in. The user's terminal job identification (TJID) is in the region control block (RCB).

...1 .... RCBFRS: Bit 3 is the restore flag. When set, this flag indicates that the user, whose terminal job identification (TJID) is in the region control block (RCB), should be restored by the region control task (RCT).

.... 1... RCBOCAB: Bit 4 set indicates that the out-of-main storage abnormal termination routine was invoked.

.... .x.. Bit 5 is reserved for future use.

.... ..1. RCBFAT: Bit 6 is the attention exit flag. When set, this flag indicates that an attention exit has been requested for one or more users.

....1 RCBFND: Bit 7 is the END region control task (RCT) flag. When set, this flag indicates that the region control task (RCT) should terminate normally and return control to the time sharing control task (TSC).

FLG2 hh
This field contains the second byte of the region control block (RCB) flags. See FLG.

1... .... RCBFSE: Bit 0 is the swap end flag. When set, this flag indicates that the swap-in operation for the current user of this region is complete.

.1.. .... RCBSTOP: Bit 1 is the region stop flag. When set, this flag indicates that a request has been made to stop the region. Every user of this region will be logged off.

..1. .... RCBACTV: Bit 2 indicates the active status of the region control task (RCT). The flag is set to one when the region control task is initialized; it is set to zero when the region control task is terminated.

...1 .... RCBSTR1: Bit 3 indicates that a region start has been requested, and the region control task should be attached.

.... 1... RCBSTR2: Bit 4 indicates that a region start has been requested, and a swap logon image should be created.

.... .xxx Bits 5 through 7 are reserved for future use.

FBQE hh
The number of free block queue elements (FBQEs) for this region.

UTTMQ hhhh
The relative track address (TT) of the map queue pointer. The map queue pointer describes the location of the region's initialized logon image on the swap data set.

CUSE hhhh
The number of users logged on to use this region. The time sharing control task (TSC) increments the count before disconnect (DISC) and decrements the count during logon.

EXTNT hhhhhhhh
The address of the initialized time sharing job block extension (TJBX). The TJBX is created during the logon initialization for this region.

UMSM hhhhhhhh
The address of the user main storage map. This map describes the initialized logon main storage image for this region.

SDCB hhhhhhhh
The address of the swap data set control block (SDCB). This block points to the location of the initialized logon image on the swap data set for this region.

PQE hhhhhhhh
The address of the partition queue element (PQE) pointer in the system queue space (SQS). The PQE describes the main storage space assigned to this region. The PQE pointer is used to manipulate main storage when (1) this region control task's (RCT's) region is obtained during start time sharing initialization and (2) this region control task's (RCT's) region

is freed during region control task termination.

PRG hhhhhhhh
PRG1 hhhhhhhh
PRG2 hhhhhhhh
These three words constitute the SVC I/O purge parameter list. For further information, see the "Purge Macro Instruction" in the publication IBM System/360: System Programmer's Guide, GC28-6550.

QPL hhhhhhhh
The address of the quiesce I/O parameter list.

STECB hhhhhhhh
An event control block (ECB). During a subsystem recovery, the time sharing control task (TSC) waits for this ECB to be posted by the region control task (RCT). The posting is done during end processing.

RCOVR hhhhhhhh
These bits indicate the current recovery status of the region control task (RCT) in the event of a subsystem failure.

byte 0

1... .... RCBRCOVR: Bit 0 set indicates that the status bits in the following 3 bytes are valid.

.xxx xx.. Reserved for future use.

.... ..x. RCBWTOR: WTOR restore processing complete.

....x RCBTACMP: Transient area restore processing complete.

byte 1 -- RCBRSFLG

1... .... RCBRSTRT: Bit 0 set indicates a restore.

.1.. .... RCBTCBDN: Bit 1 set indicates that the task control blocks (TCBs) have been requeued.

..1. .... RCBQELCM: Bit 2 set indicates that the queue element (QEL) restore processing is complete.

...1 .... RCBTQECM: Bit 3 set indicates that the timer queue element (TQE) restore processing is complete.

.... 1... RCBRQIQC: Bit 4 set indicates that both the request queue element (RQE) and the interrupt queue element (IQE) restore processing is complete.

.... .1.. RCBIORSC: Bit 5 set indicates that the I/O restore processing is complete.

.... ..xx Bits 6 and 7 are reserved for future use.

byte 2 -- RCBQUFLG

1... .... RCBQUSTR: Bit 0 set indicates that quiesce has started.

.1.. .... RCBIOSTR: Bit 1 set indicates that the first entry into the I/O purge routine is complete.

..1. .... RCBTADON: Bit 2 set indicates that the transient area quiesce is complete.

...1 .... RCBWTORD: Bit 3 set indicates that the write to operator with reply (WTOR) quiesce is complete.

.... 1... RCBQELDN: Bit 4 set indicates that the queue element (QEL) quiesce is complete.

.... .1.. RCBIODON: Bit 5 set indicates that the second entry I/O purge is complete.

.... ..1. RCBTQEDN: Bit 6 set indicates that the timer queue element (TQE) quiesce is complete.

....1 RCBRQIQD: Bit 7 set indicates that both the request queue element (RQE) and the interrupt queue element (IQE) are complete.

byte 3

1... .... RCBSWTCH: Bit 0 indicates the method of search used by various subroutines in IKJEAT07. When equal to zero, all system users are purged according to the terminal job identification (TJID). When equal to one, all users in this region are purged as indicated by the region control block addresses.

.1.. .... RCBSWTON: When bit 1 is set along with bit 0 being set, all system users are purged. A search is made according to the terminal job identification (TJID) and the request control block (RCB).

..xx xxxx Bits 2 through 7 are reserved for future use.

CONID hh
The routing code of the console that issued the last START, MODIFY, or STOP command.

RESV hhhhhh
Reserved for future use.

USER MAIN STORAGE MAP (UMSM) FORMATTING: The UMSM is used in the swap operation. One user main storage map 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 describe the main storage space allocated to the time sharing user. The number of UMSM extents has established defaults that can be modified by the operator when he starts the time sharing system. The number of extent entries is stored in the time sharing job block (TJB) at TJBUMSMN. Unused extent fields contain zeros.

UMSM hhhhhh
The address of the user main storage map.

ADDR-LN hhhhhhhh

bytes 0 and 1
Begin Address: This field contains the two high order bytes of the beginning address of the main storage segment allocated to the time sharing user. Since main storage is allocated in 2K blocks, the low order byte is always zero and, therefore, need not be kept in a control block.

bytes 2 and 3
This field contains the two high-order bytes designating the length of the main storage space allocated to the time sharing user. Since main storage is allocated in 2K blocks, the low-order byte is always zero and, therefore, need not be kept in a control block.

SWAP DATA CONTROL BLOCK (SWAP DCB) FORMATTING: The swap data control block (SWAP DCB) is used whenever a time sharing user's region is swapped into or out of main storage. Each region control task (RCT) has one swap data control block. Following the address of the swap data control block is the contents of the main storage data that was written on the swap data set.

SWAP DCB hhhhhhhh
The address of the swap data control block.

TIME SHARING JOB BLOCK (TJB) FORMATTING: The time sharing job block (TJB) contains status information about the 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 terminals 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.

TJB hhhhhh
The address of this TJB.

TSB hhhhhhhh
The address of the terminal status block (TSB) that owns this terminal job. If zero, this job was started by an operator command.

ATTN hh
A count of the unprocessed attention interrupts for this job.

STAX hh
The number of scheduled specify terminal attention exits (STAXs).

STAT hh
This field contains flags that indicate the status of the time sharing job.

1... .... TJBNJB: Bit 0 set indicates that this TJB is currently unused.

.1.. .... TJBINCOR: Bit 1 set indicates that this user is currently in main storage.

..1. .... TJBLOGON: Bit 2 set indicates that the logon start has been set by the terminal input output coordinator (TIOC) during a dialup to request a logon. This bit is reset by the time sharing control task (TSC).

...1 .... TJBIWAIT: Bit 3 set indicates that the terminal job is in an input wait state.

.... 1... TJBOWAIT: Bit 4 set indicates that the terminal job is in an output wait state.

.... .1.. TJBSILF: Bit 5 set indicates that the user is to be logged off the system. This bit is set by the IKJSILF subroutine and tested by the region control task (RCT) restore routine that posts the logon ECB. This bit is tested and reset by the logon/logoff routine.

.... ..1. TJBDISC: Bit 6 set indicates that a request has been made to the terminal input output coordinator (TIOC) to disconnect the line.

....x Bit 7 is reserved for future use.

STAT2 hh

These flags indicate the status of the time sharing job.

1... .... TJBHUNG: Bit 0 set indicates that the user's communication line disconnected.

.1.. .... TJBHOLD: Bit 1 set indicates that an output wait (OWAIT) exists because of a hold option.

..1. .... TJBOCAB: Bit 2 set indicates an out-of-main storage abnormal termination has occurred for this user.

...1 .... TJBRNAV: Bit 3 set indicates that the user cannot be logged onto the time sharing system because (1) a machine check occurred in the user's region or (2) the region is too small for the user.

.... 1... TJBSURSV: Bit 4 set indicates that on the next swap in the swap unit is not marked as available for the user.

.... .xxx Bits 5 through 7 are reserved for future use.

EXTNT hhhhhhhh

The address of the terminal job block extension (TJBX) when it is in main storage.

RCB hhhhhhhh

The address of the region control block (RCB) for this job.

UMSM hhhhhhhh

The address of the user main storage map (UMSM) for this job.

SDCB hhhhhhhh

The address of the swap data control block (DCB) for this job.

UTTMQ hhhh

1... .... TJBUTTMP: Bit 0 of byte 0 set indicates a parallel swap.

.111 1111 Bits 1 through 7 of byte 0 along with byte 1 contain the offset into the map queue. The map queue contains a chain of allocation units for this user on the swap data set. The address of the queue is in the UTTMQ field of the TSO region control block (RCB).

RSTOR hh

This field contains the status flags used by the region control task (RCT) restore operation.

1... .... TJBOWP: Bit 0 set indicates to the terminal input output coordinator (TIOC) to end the output wait (OWAIT) condition.

.1.. .... TJBIWP: Bit 1 set indicates to the terminal input output coordinator (TIOC) to end the input wait (IWAIT) condition.

..x. .... Bit 2 is reserved for future use.

...1 .... TJBLOGP: Bit 3 set indicates that the event control block (ECB) waited for by the logon image should be posted. This flag is set by the time sharing control task (TSC) logon routine and by the IKJSILF subroutine.

.... 1... TJBLWAIT: Bit 4 set indicates that if the user is not made ready by restore processing, he should be swapped out again.

.... .x.. Bit 5 is reserved for future use.

.... ..1. TJBFAT: Bit 6 set indicates that an attention exit is requested for this user's job.

....x Bit 7 is reserved for future use.

UMSMN hh
The number of entries in the user main storage map (UMSM).

USER cccccccc
The userid of the user who owns this job. This field may have trailing blanks when the user identification contains less than eight characters.

IPPB hhhhhhhh
An address pointer to the beginning of a chain of inter-partition post blocks that indicate the event control blocks (ECBs) to be posted by the restore operation.

NEWID hh
Identifies the region where the user should be logged on. When this field is zero, the TSO driver should select the region. When this field is set by the end-of-routine for logon/logoff, it identifies the new region to which the user will be shifted.

FLUSL hh
Reserved for future use.

TJID hhhh
This field contains the terminal job identification (TJID) for this time sharing job.

MONI hh
These flags indicate various processing functions that cause operator messages to be sent to this terminal. The flags are set and reset when the terminal user issues the MONITOR subcommand of the OPERATOR command.

1... .... TJBMDSN: Bit 0 set indicates that the first non-temporary data set allocated to a new volume should be displayed as part of the mount and keep messages.

.1.. .... TJBMJBN: Bit 1 set indicates that the name of each job is to be displayed on the console when each job is initiated and terminated, and that the unit record allocations are to be displayed when a job step is initiated.

..1. .... TJBMSES: Bit 2 set indicates that when a terminal session is initiated or terminated a message is displayed on the operator console.

...1 .... TJBMSPA: Bit 3 set indicates that the available space on a direct access device is to be displayed on the operator console as part of the demount message.

.... 1... TJBMSTA: Bit 4 set indicates that, at the end of a job or job step, certain data set disposition information should be printed with the demount messages. These dispositions are: KEEP, CATLG, or UNCATLG.

.... .xxx Bits 5 through 7 are reserved for future use.

RSV hhhhhh
Reserved for future use.

<u>TERMINAL STATUS BLOCK (TSB) FORMATTING</u>: Each terminal status block (TSB) contains status information about one terminal user. 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 in the TIOCRPT.

TSB hhhhhh
The address of this terminal status block (TSB).

STAT hh
This field contains the terminal status indicator flags.

1... .... TSBINUSE: Bit 0 set indicates that this TSB is being used.

.1.. .... TSBLWAIT: Bit 1 set indicates that the terminal keyboard is locked due to a lack of input buffer space.

..1. .... TSBDSPLY: Bit 2 set indicates that this TSB represents a terminal which is a graphic device.

...1 .... TSBNOBUF: Bit 3 set indicates that TPUT found no time sharing buffers.

.... 1... TSBITOFF: Bit 4 set indicates that this user wishes to prevent inter-terminal communications.

.... .1.. TSBDISC: Bit 5 set indicates that this TSB has been processed by logoff.

.... ..x. Bit 6 is reserved for future use.

....1 TSBATNLD: Bit 7 set indicates an attention for an input line deletion.

TJB hhhhhh
The address of the time sharing job block (TJB) currently used by this terminal. This field contains zeros when this terminal is not associated with a time sharing job block.

FLG1 hh
This field contains terminal status flags.

1... .... TSBANSR: Bit 0 set indicates that an attention simulation is requested.

.1.. .... TSBOFLSH: Bit 1 set indicates that the output trailer queue is to be flushed. This bit is set by TCLEARQ.

..1. .... TSBOWIP: Bit 2 set indicates that a TPUT operation is in progress.

...1 .... TSBWOWIP: Bit 3 set indicates that a task is waiting for another task to complete a TPUT operation.

.... 1... TSBIFLSH: Bit 4 set indicates that an input queue flush is in progress.

.... .1.. TSBTJOW: Bit 5 set indicates that this user is already using the maximum number of output buffers that can be allocated. This TSB waits on event control block (ECB) for this TCB. This bit is set by a TPUT macro instruction with a terminal job identification (TJID).

.... ..x. Bit 6 is reserved for future use.

....1 TSBTJBF: Bit 7 set indicates that no time sharing buffers were available when the SVC for TPUT with the terminal job identification (TJID) was issued. The system waits for the TJB event control block (ECB) to be posted.

WTSB hhhhhh
Reserved for future use.

LNSZ hh
The number of characters that can be printed on one line for this terminal. This field is set by either logon or STSIZE.

OTBFP hhhhhh
The address of the trailer buffer if the heading buffer for a message has been removed from the message queue. This field is reset to zeros when the message has been completely moved to the TCAM buffers.

NOBF hh
The number of buffers on the output queue.

OBFP hhhhhh
The address of the first buffer on the output buffer queue.

BRKFL hh
These flags indicate the status of the communication line.

1... .... TSBBIPI: Bit 0 set indicates to the TSINPUT that a partial line exists for prompting. Set by TSOUTPUT.

.1.. .... TSBAUTON: Bit 1 set indicates that automatic input line numbering is requested.

..1. .... TSBBRKIN: Bit 2 set indicates that TPUT is using the breakin option and a partial line was assigned to this function. This bit is set by TSINPUT. TSINPUT is a TCAM subtask.

...1 .... TSBAULST: Bit 3 set indicates that automatic line numbering has started.

.... 1... TSBAUTOC: Bit 4 set indicates that automatic character prompting is used.

..... .1.. TBSTAUT: Bit 5 set indicates that the user is being prompted with the next line number.

....11 TSBSATN1: Bits 6 and 7 contain a count of the number of characters used to simulate attention.

ITBFP hhhhhh
The address of the first buffer in the trailer input buffer chain.

NIBF
The number of buffers on the input queue.

IBFP hhhhhh
The address of the first buffer in the input buffer queue chain.

CLEAR hh
This field contains terminal status flags.

1... .... TSBATTN: Bit 0 set indicates that an attention from this terminal has been ignored.

.1.. .... TSBTJMSG: Bit 1 set indicates that TSOUTPUT is processing a terminal job identification (TJID) message.

..1. .... TSBSPIT: Bit 2 set indicates that breakin prompt and automatic prompt are suppressed.

...1 .... TSBNBKSP: Bit 3 set indicates that the next character in the user's buffer is a backspace character.

.... xxxx Bits 4 through 7 are reserved for future use.

QCB hhhhhh
The address of the queue control block (QCB) that contains the destination for the message being sent.

ECB hhhhhhhh
The event control block (ECB) at which the inter-terminal communication (TPUT with TJID) waits (1) when there are no time sharing buffers, (2) when the TSBOWIP bit is set, or (3) when the TSBOQHLD bit is set.

TJID hhhh
The terminal job identification (TJID) of the task waiting on this TCB's event control block (ECB).

STCC hhhh
These two bytes define special purpose characters that may be redefined by the terminal user.

byte 0
TSBLNDCC: This byte contains the line delete character.

byte 1
TSBKSPCC: This byte contains the character delete character.

ATNLC hhhh
The number of successive lines of printed output between attention simulation reads.

ATNTC hhhh
The number of seconds between attention simulation reads.

LNNO hh
When a graphic terminal device is used, this is the number of line that can be displayed.

BLNK hh
Reserved for future use.

ASRCE hhhh
This field contains the same information as the PRFSRCE field in the TCAM buffer prefix.

ATNCC cccc
This field contains from one to four characters that are used to simulate attention. Some of the character positions may contain blanks.

AUTOS hhhhhhhh
This field initially contains the starting line number for the first input line. While the line of input information is being received from the terminal user, this field is updated to contain the value of the current line number.

AUTOI hhhhhhhh
This field contains the value that is used to automatically increment the value of the input line numbers. This field can be modified by the terminal user.

ERSDS cccc
When a graphic terminal device is used, this word contains the characters used to erase the display screen.

TIME SHARING JOB BLOCK EXTENSION (TJBX) FORMATTING: 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.

TJBX hhhhhh
The address of the TJBX.

XFST hhhhhhhh
The address of the logon TCB. The logon TCB is the first TCB on the user's ready queue.

XLAST hhhhhhhh
The address of the last TCB on the user's ready queue.

XDSE hhhhhhhh
The address of the data set extension (DSE) used by TSO dynamic allocation.

XSVRB hhhhhhhh
The address of the first supervisor request block (SVRB) purged from the transient area queue.

XRQE hhhhhhhh
The address of the first request queue element (RQE) purged from the asynchronous exit queue.

XIQE hhhhhhhh
The address of the first interrupt queue element (IQE) purged from the asynchronous exit queue.

TAXE hhhhhhhh
The address of the queue of terminal attention exit elements (TAXEs) used to schedule the attention exits.

XLECB hhhhhhhh
The logon event control block (ECB) that was posted by the region control task (RCT) to activate logon/logoff.

XPSWD cccccccc
The password entered by the terminal user during logon. If the password contains less than eight characters, the field is padded to the right with blanks. The entire field contains blanks when the user is not required to enter a password.

RSV hhhhhhhh
Reserved for future use.

XAIQE hhhhhhhh
The address of the attention interrupt queue element (IQE) currently being processed by the attention prologue.

XQPL hhhhhhhh
The address of the quiesce parameter list (QPL).

XNQPE hhhh
The number of entries in the quiesce parameter list (QPL).

XNTCB hhhh
The number of task control blocks (TCBs) active in the user's job step. When the value in XNTCB exceeds XNQPE, the quiesce parameter list is enlarged.

XLQPL hhhh
The number of bytes in the quiesce parameter list.

HBFL hhhh
Reserved for future use.

XACT hhhhhhhh
The relative track and record address (TTR) for the account control table (ACT) on SYSJOBQE.

XAECB hhhhhhhh
This field contains either: (1) The address of the logon/logoff event control block (ECB) when logon processing begins. (2) The address of the command scheduling block (CSCB's) cancel event control block (ECB) after the CSCB is created.

XKEYA hhhhhhhh
The address of the storage key save area.

PROTECTED STEP CONTROL BLOCK (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 microseconds.

PSCB hhhhhh
The address of this PSCB.

USER ccccccc
These seven bytes contain the userid entered by the terminal user during logon. If necessary, it is padded to the right with blanks. This field uniquely identifies each terminal user in the time sharing system.

USRL hh
The number of nonblank characters in the userid.

GPNM cccccccc
An eight-byte group name initialized by logon from the user attribute data set (UADS). When a name is not available from UADS, the unit name used by the dynamic allocation

interface routine (DAIR) is used, if a name is required.

ATR1 hhhh
Sixteen bits used to define terminal user attributes.

byte 0

```
1... .... PSCBCTRL:  Bit 0 set
          indicates that the user may
          use the OPERATOR command.

.1.. .... PSCBACCT:  Bit 1 set
          indicates that the user may
          use the ACCOUNT command.

..1. .... PSCBJCL:  Bit 2 set
          indicates that the user may
          use the SUBMIT, STATUS,
          CANCEL, and OUTPUT commands.

...x xxxx Bits 3 through 7 are
          reserved for future use.
```

byte 1
Reserved for future use.

ATR2 hhhh

bytes 0 and 1
Reserved for use by IBM customers.

CPU hhhhhhhh
The cumulative CPU time used by this terminal user during this session. The CPU field is set to zero during logon.

SWP hhhhhhhh
The cumulative time that this terminal user has been resident in the region. The SWP field is set to zero during logon.

LTIM hhhhhhhh
The actual time of day that this user logged on to the time sharing system for this session.

TCPU hhhhhhhh
The total CPU time used by this terminal user, excluding the current session.

TSWP hhhhhhhh
The total time that the terminal user has been resident in the region during this accounting period, excluding the current session.

TCON hhhhhhhh
TC01 hhhhhhhh
TCON and TC01 are a single eight byte field. This field contains the total connect time for this terminal user during this accounting period, excluding the current session.

RLGB hhhhhhhh
The address of the re-logon buffer block used by logon as a pointer to the re-logon command buffer.

UPT hhhhhhhh
The address of the user profile table (UPT).

UPTL hhhh
The number of bytes in the user profile table.

RSV1 hhhh
RSV2 hhhhhhhh
RSV1 and RSV2 are a single six byte field that is reserved for future use.

USE1 hhhhhhhh
USE2 hhhhhhhh
USE1 and USE2 are a single eight byte field reserved for use by IBM customers.

<u>TERMINAL ATTENTION EXIT ELEMENT (TAXE) FORMATTING</u>: The TSO terminal attention exit element (TAXE) consists of a regular 24 word interrupt request block (IRB) plus a TSO addendum. 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.

TAXE hhhhhh
The address of this TAXE when it is in main storage.

TMFLD hh
This field contains indicators for the time routines.

```
1... .... Bit 0 set indicates that the
          timer element was not
          queued.

.1.. .... Bit 1 set indicates that the
          local time-of-day option is
          used.

..00 .... Bits 2 and 3 set to
          zero-zero indicate that the
          time interval was requested
          in timer units (26.04166
          microseconds).

..01 .... Bits 2 and 3 set to zero-one
          indicate that the time
          interval was requested in
          binary units.

..10 .... Reserved for future use.
```

..11 .... Bit 2 and 3 set to one-one indicate that the time interval was requested in decimal digits.

.... 1... Bit 4 set indicates that the time interval has expired.

.... .000 Bits 5 through 7 set to zero-zero-zero indicate an STIMER task time request.

.... .001 Bits 5 through 7 set to zero-zero-one indicate an STIMER wait request.

.... .011 Bits 5 through 7 set to zero-one-one indicate an STIMER REAL time request.

.... .100 Bits 5 through 7 set to one-zero-zero indicate an STIMER task time request with a specified exit.

.... .111 Bits 5 through 7 set to one-one-one indicate an STIMER REAL time request with a specified exit.

Other combinations of bits 5 through 7 are reserved for future use.

PPSAV hhhhhh
The starting address of the register save area for the problem program.

ABOPSW hhhhhhhh
This field displays the right half (bytes 4 through 7) of the program status word (PSW) that was active in the dump system during the execution of an ABEND or ABTERM routine. If these routines have not been invoked, then this field contains zeros.

WCSA hh
The number of requests waiting when termination occurred.

SIZE hh
The number of doublewords in this request block.

STAB hhhh
This field contains two bytes of status and attribute information.

byte 0
The TAXE is a type of interrupt request block (IRB). Byte zero identifies the type of request block; however, for the TAXE, only the IRB identification is used.

01.. .... Bits 0 and 1 set to zero-one indicate that this is an interrupt request block (IRB).

byte 1
This byte contains various request block indicators.

1... .... Bit 0 set indicates that the RBLINK field points to the task control block (TCB).

.1.. .... Bit 1 set indicates that the program related to the interrupt request block (IRB) is active.

..1. .... Bit 2 set indicates that this interrupt request block (IRB) is for an exit routine (ETXR).

...x .... Bit 3 is reserved for future use.

.... 00.. Bits 4 and 5 set to zero-zero indicate that the request queue element (IQE) is not to be returned.

.... 01.. Bits 4 and 5 set to zero-one indicate that the interrupt request block (IRB) has queue elements for asynchronously executed routines that are request queue elements (RQEs).

.... 10.. Bits 4 and 5 set to one-zero indicate that an interrupt queue element (IQE) is not to be returned at EXIT.

.... 11.. Bits 4 and 5 set to one-one indicate that the interrupt request block (IRB) has queue elements for asynchronously executed routines that are interrupt queue elements (IQEs).

.... ..1. Bit 6 set indicates that the request block storage can be freed at exit.

....0 Bit 7 set to zero indicates a wait for a single event or all of a number of events.

....1 Bit 7 set to one indicates a wait for a number of events that is less than the total number of events that are waiting.

EP hhhhhhhh
The address of the routine that was asynchronously executed.

LOPSW hhhhhhhh (Left half of PSW)
ROPSW hhhhhhhh (Right half of PSW)
This program status word (PSW) contains the status of the program represented by the request block being displayed when a new request block was created. Had the dumped system been allowed to continue processing normally, the operation would have been resumed with this PSW.

USE hh
This field contains the use count as used by ATTACH.

IQE hhhhhh
The address of the list origin for the interrupt queue element (IQE).

WCF hh
The number of requests that were pending when this dump was taken.

LINK hhhhhh
The address of the next request block (RB) on this RB queue. If this is the last request block on the queue, then this field contains the address of the task control block (TCB).

GR0 hhhhhhhh
.
.
.
GR15 hhhhhhhh
The general register save area used by the supervisor.

NIQE hhhhhhhh
The address of the next available interrupt queue element (IQE).

LNK hhhhhhhh
The address of the next interrupt queue element (IQE).

PRM1 hhhhhhhh
The address of the parameter list for the asynchronous exit routine.

IRB hhhhhhhh
The address of the interrupt request block (IRB) to be scheduled next.

TCB hhhhhhhh
The address of the task control block (TCB) for this TAXE.

TLNK hhhhhhhh
The address of the next TAXE on this queue.

XPSW hhhhhhhh
The left half (bytes 0 through 3) of the program status word (PSW) for the user attention exit routine.

EXIT hhhhhhhh
The address of the user attention exit routine.

STAT hhhhhhhh
This field contains status flags for this TAXE.

byte 0

1... .... TAXEFKEY: Bit 0 set indicates that the task issuing the specify terminal attention exit (STAX) macro instruction is a problem program.

.1.. .... TAXEMOD: Bit 1 set indicates that the task issuing the specify terminal attention exit (STAX) macro instruction is in problem program mode.

..1. .... TAXEFFREQ: Bit 2 set indicates that the requested TAXE is not available for scheduling.

...x xxxx Bits 3 through 7 are reserved for future use.

bytes 1-3
Reserved for future use.

PARM hhhhhhhh
The address of the parameter list for the specify terminal attention exit (STAX) macro instruction.

TAIE hhhhhhhh
The address of the terminal attention interrupt element.

IBUF hhhhhhhh
The address of the user input buffer.

USER hhhhhhhh
The address of the user parameter list from the specify terminal attention exit (STAX) macro instruction.

Task Control Block Summaries

If, during the course of program execution, the IMDPRDMP program formatted the major system control blocks of more than one MVT or MFT task, a summary of each displayed task's TCB is presented at the end of the control block portion of the dump listing. Depending upon the operating system option under which the dumped task was operating, either the MVT/MFT-with-subtasking TCB summary format (Figure 25), or the abridged MFT-without-subtasking TCB summary format (Figure 26) is presented.

Both summary formats are identified by two lines of heading information. The first heading line displays the optional dump listing title, the name of the module that invoked the dump, and the date and time that the information was captured from the dumped system. The second line of heading displays the identifying phrase "**** TCB SUMMARY ****."

The individual TCB summaries contain the following information:

MVT or MFT with Subtasking TCBs: Are summarized in the two-line array illustrated in Figure PROUT-9 and described below:

JOB cccccccc
The JOB field in the first line of each task control block array displays to the user the eight-character name of the job associated with the TCB.

STEP cccccccc
The STEP field shows the eight-character name of the job step as it appeared on the label field of the EXEC JCL statement associated with the step.

TCB hhhhhh
The starting address of the task control block.

CMP hhhhhhhh
This field shows the ABEND indicators and user and system completion codes associated with this TCB. (See the relevant TCB discussion for the contents of this field.)

NTC hhhhhhhh
This word contains the address of the TCB that occurred previous to this one on the originating task's subtask queue. If the TCB being summarized was the first on the queue, this field displays zeros.

OTC hhhhhhhh
The OTC field displays the address of the TCB representing the originating task.

LTC hhhhhhhh
This field contains the address of the TCB that occurred last on the originating task's subtask queue at the time the dump was taken. If the TCB being summarized was the last on the subtask queue, this field contains zeros.

PAGE ddd
The page of the dump listing on which the formatted control blocks associated with this TCB, may be found.

MFT Without Subtasking TCBs: Are summarized in the two line abridged array illustrated in Figure PROUT-10 and described below:

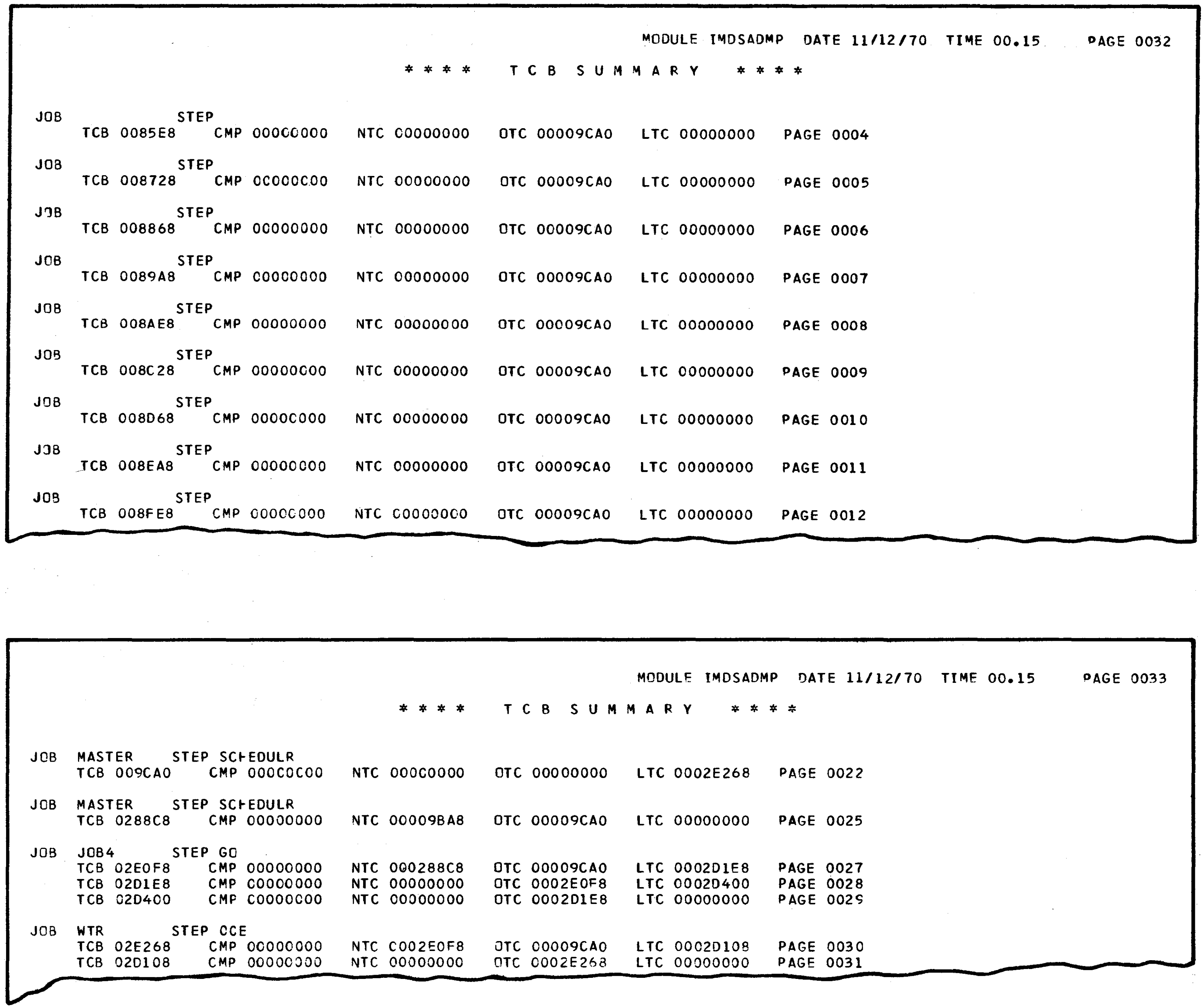
JOB cccccccc
The JOB field in the first line of each task control block array, displays to the user the eight-character name of the job associated with the TCB being summarized.

STEP cccccccc
The STEP field shows the eight-character name of the job step as it appeared in the label field of the EXEC JCL statement associated with the step.

TCB hhhhhh
The starting address of the task control block being summarized is given in the first field of this second line.

CMP hhhhhhhh
This field shows the ABEND indicators and user and system completion codes associated with the TCB. (See the MFT TCB discussion for a description of the contents of this field.)

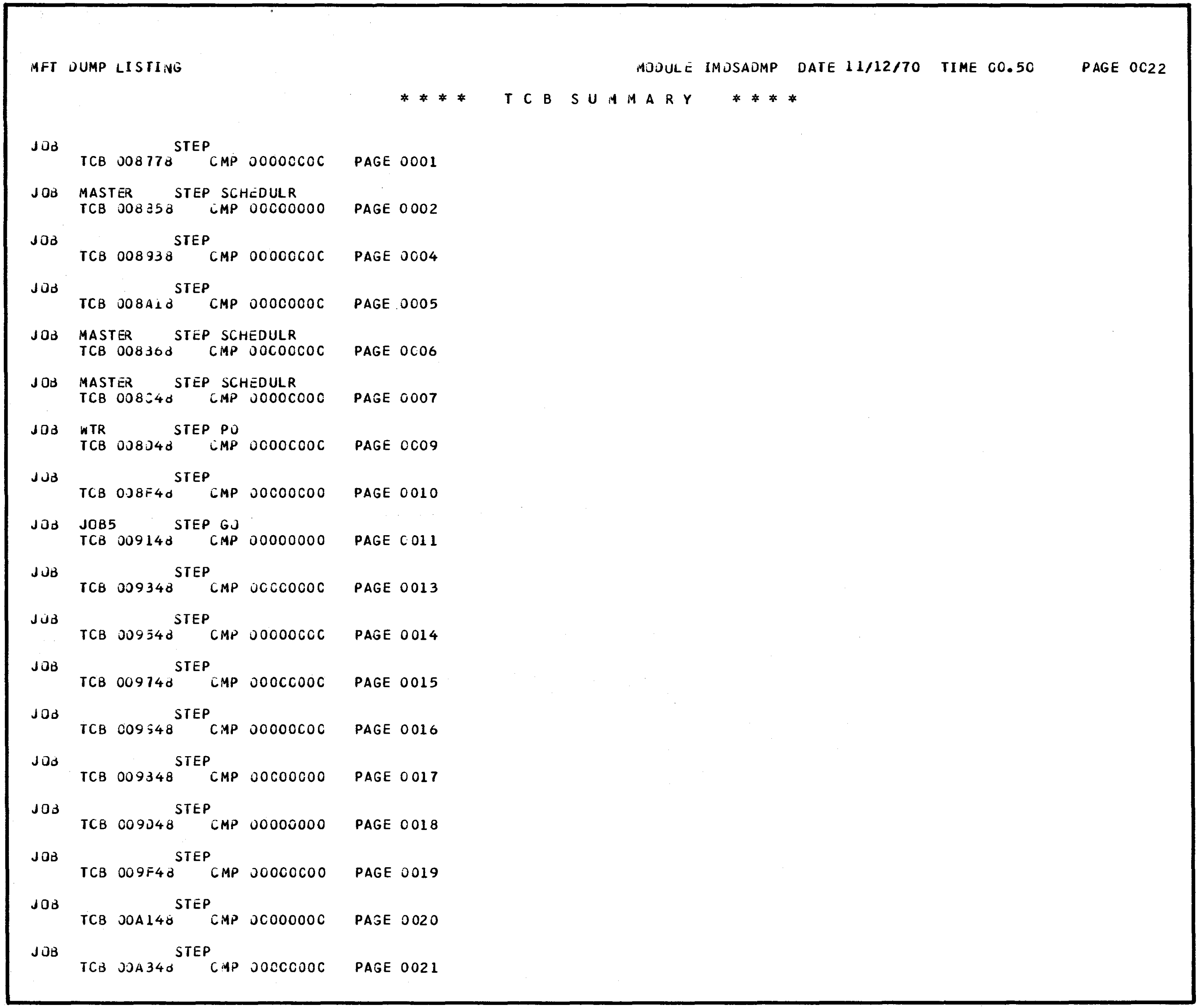
PAGE ddd
The page of the dump listing on which the formatted control blocks associated with this TCB, are found.



```
                                                      MODULE IMDSADMP  DATE 11/12/70  TIME 00.15     PAGE 0032
                                     * * * *    T C B   S U M M A R Y    * * * *

JOB           STEP
    TCB 0085E8     CMP 00000000    NTC 00000000    OTC 00009CA0    LTC 00000000    PAGE 0004

JOB           STEP
    TCB 008728     CMP 00000000    NTC 00000000    OTC 00009CA0    LTC 00000000    PAGE 0005

JOB           STEP
    TCB 008868     CMP 00000000    NTC 00000000    OTC 00009CA0    LTC 00000000    PAGE 0006

JOB           STEP
    TCB 0089A8     CMP 00000000    NTC 00000000    OTC 00009CA0    LTC 00000000    PAGE 0007

JOB           STEP
    TCB 008AE8     CMP 00000000    NTC 00000000    OTC 00009CA0    LTC 00000000    PAGE 0008

JOB           STEP
    TCB 008C28     CMP 00000000    NTC 00000000    OTC 00009CA0    LTC 00000000    PAGE 0009

JOB           STEP
    TCB 008D68     CMP 00000000    NTC 00000000    OTC 00009CA0    LTC 00000000    PAGE 0010

JOB           STEP
    TCB 008EA8     CMP 00000000    NTC 00000000    OTC 00009CA0    LTC 00000000    PAGE 0011

JOB           STEP
    TCB 008FE8     CMP 00000000    NTC 00000000    OTC 00009CA0    LTC 00000000    PAGE 0012
```

```
                                                      MODULE IMDSADMP  DATE 11/12/70  TIME 00.15     PAGE 0033
                                     * * * *    T C B   S U M M A R Y    * * * *

JOB MASTER    STEP SCHEDULR
    TCB 009CA0     CMP 00000000    NTC 00000000    OTC 00000000    LTC 0002E268    PAGE 0022

JOB MASTER    STEP SCHEDULR
    TCB 0288C8     CMP 00000000    NTC 00009BA8    OTC 00009CA0    LTC 00000000    PAGE 0025

JOB JOB4      STEP GO
    TCB 02E0F8     CMP 00000000    NTC 000288C8    OTC 00009CA0    LTC 0002D1E8    PAGE 0027
    TCB 02D1E8     CMP 00000000    NTC 00000000    OTC 0002E0F8    LTC 0002D400    PAGE 0028
    TCB 02D400     CMP 00000000    NTC 00000000    OTC 0002D1E8    LTC 00000000    PAGE 0029

JOB WTR       STEP OCE
    TCB 02E268     CMP 00000000    NTC 0002E0F8    OTC 00009CA0    LTC 0002D108    PAGE 0030
    TCB 02D108     CMP 00000000    NTC 00000000    OTC 0002E268    LTC 00000000    PAGE 0031
```

Figure 25. TSB Summary Sample for System That Operated Under MVT or MFT With Subtasking



```
MFT DUMP LISTING                                                    MODULE IMDSADMP  DATE 11/12/70  TIME C0.5C     PAGE 0C22
                                          * * * *   T C B  S U M M A R Y   * * * *

JOB            STEP
    TCB 008778    CMP 00000C0C   PAGE 0001

JOB  MASTER    STEP SCHEDULR
    TCB 008358    CMP 00C00000   PAGE 0002

JOB            STEP
    TCB 008938    CMP 00000C0C   PAGE 0004

JOB            STEP
    TCB 008A18    CMP 0000000C   PAGE 0005

JOB  MASTER    STEP SCHEDULR
    TCB 008368    CMP 00C000C0C  PAGE 0C06

JOB  MASTER    STEP SCHEDULR
    TCB 008C48    CMP 0000C000   PAGE 0007

JOB  WTR       STEP P0
    TCB 008D48    CMP 0000C00C   PAGE 0C09

JOB            STEP
    TCB 008F48    CMP 00000C00   PAGE 0010

JOB  JOB5      STEP GO
    TCB 009148    CMP 00000000   PAGE C011

JOB            STEP
    TCB 009348    CMP 0CCC000C   PAGE 0013

JOB            STEP
    TCB 009548    CMP 00000CCC   PAGE 0014

JOB            STEP
    TCB 009748    CMP 000CC00C   PAGE 0015

JOB            STEP
    TCB 009948    CMP 00000C0C   PAGE 0016

JOB            STEP
    TCB 009B48    CMP 00C00000   PAGE 0017

JOB            STEP
    TCB 009D48    CMP 00000000   PAGE 0018

JOB            STEP
    TCB 009F48    CMP 00000C00   PAGE 0019

JOB            STEP
    TCB 00A148    CMP 0C00000C   PAGE 0020

JOB            STEP
    TCB 00A348    CMP 000C000C   PAGE 0021
```

Figure 26. TCB Summary Sample for Systems that Operated Under MFT Without Subtasking

THE GENERAL FORMAT

The IMDPRDMP program uses a general format to display the hexadecimal contents of main storage. The particular areas of main storage displayed are determined by the parameters entered after the PRINT user control verb.

To identify various dump printouts, IMDPRDMP prints specific headings on each dump, such as ALLOCATED STORAGE, PRINT STORAGE, and NUCLEUS and SQA PRINT. A sample of a general format dump is shown in Figure 27.

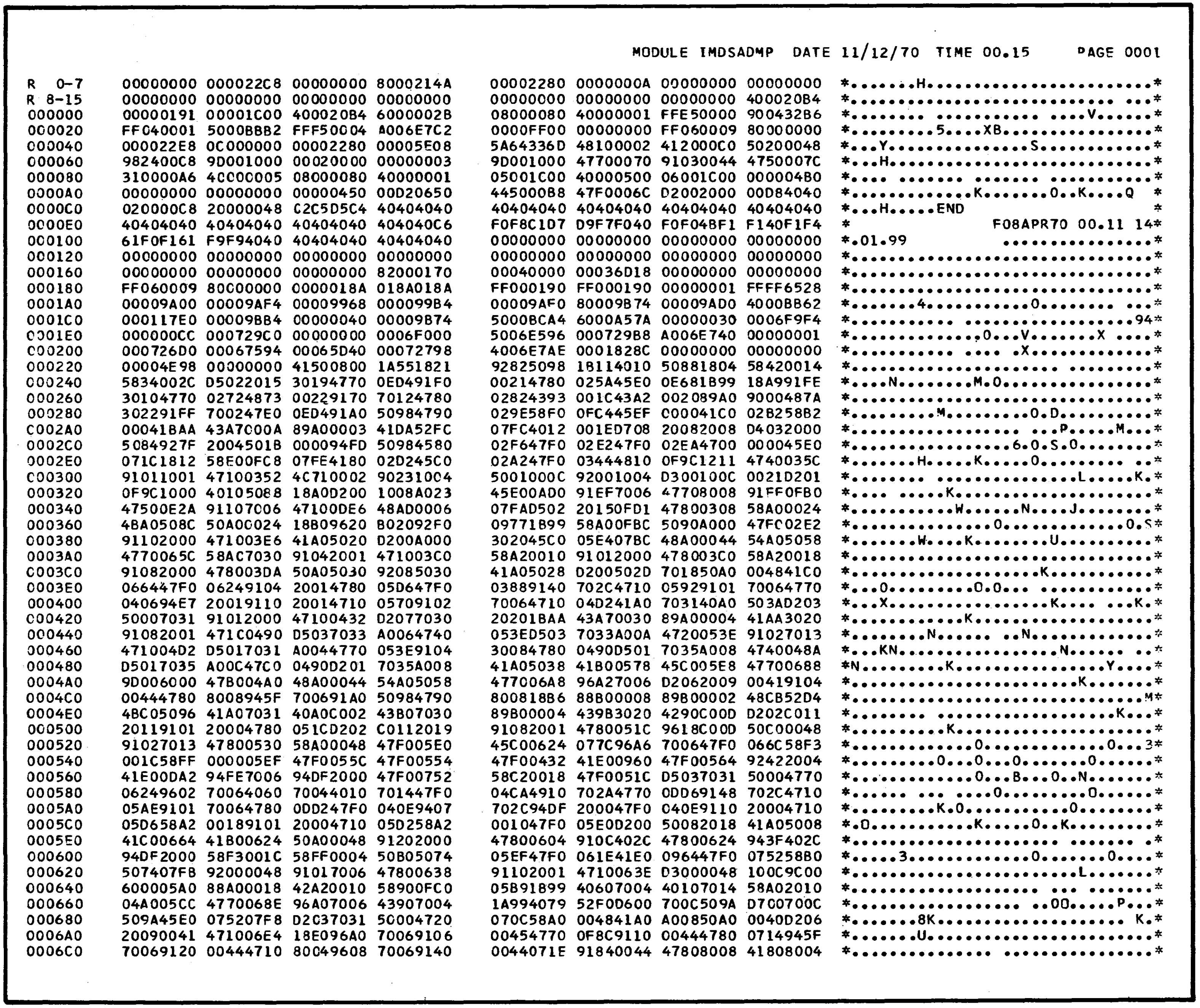
The IMDPRDMP program also reverts to the general format if it is unable to format control block information because it encountered either a control block error or one of several user control statement format errors.

Each page of an IMDPRDMP program dump listing containing information displayed in the general format is identified by a heading line. This heading line shows the optional title supplied by the user followed by the date and time that the information was taken from the dumped system. A sequential page number also appears in each heading line.

Listings being produced under control of the PRINT ALL, PRINT CURRENT, or PRINT STORAGE (no operands) format control statement display the contents of the sixteen general purpose registers. If the dump was obtained from a multiprocessing system and both sets of registers were obtained, then the contents of both sets of registers are displayed. Where applicable, the beginning of each main storage region is noted by a line that gives the job, step and procedure step name of the owning task, followed by the status of the region (BORROWED, ROLL-OUT, OWNED).

Then, starting at an address requested by the user, as specified in a PRINT user control statement, (or location zero if no address was specified) the contents of main storage are displayed. Each line of the general format displays eight words of main storage. Preceding each line of information is the address of the first byte displayed followed by a one-character storage protection key indicator representing the key associated with the area of main storage being displayed on this line. Following each line of information, a 32-character translation field is printed. This field gives the EBCDIC translation of the translatable characters in the eight hexadecimal words. Untranslatable bytes are represented by positional periods.

Printing of any line that duplicates the contents of the line printed previously, is suppressed. Duplicate lines are indicated by the phrase "TO NEXT LINE ADDRESS SAME AS ABOVE" following the line duplicated.



```
                                                                   MODULE IMDSADMP  DATE 11/12/70  TIME 00.15     PAGE 0001

R  0-7     00000000 000022C8 00000000 8000214A    00002280 0000000A 00000000 00000000   *.......H........................*
R 8-15     00000000 00000000 00000000 00000000    00000000 00000000 00000000 400020B4   *............................ ...*
000000     00000191 00001C00 400020B4 6000002B    08000080 40000001 FFE50000 900432B6   *........ ........... ....V......*
000020     FF040001 5000BBB2 FFF50004 A006E7C2    0000FF00 00000000 FF060009 80000000   *.........5....XB................*
000040     000022E8 0C000000 00002280 00005E08    5A64336D 48100002 412000C0 50200048   *...Y...............S............*
000060     982400C8 9D001000 00020000 00000003    9D001000 47700070 91030044 4750007C   *...H............................*
000080     310000A6 40C00005 08000080 40000001    05001C00 40000500 06001C00 000004B0   *.... ....... ....... ...........*
0000A0     00000000 00000000 00000450 00D20650    445000B8 47F0006C D2002000 00D84040   *.............K.......0..K....Q  *
0000C0     020000C8 20000048 C2C5D5C4 40404040    40404040 40404040 40404040 40404040   *...H.....END                    *
0000E0     40404040 40404040 40404040 404040C6    F0F8C1D7 D9F7F040 F0F04BF1 F140F1F4   *               F08APR70 00.11 14*
000100     61F0F161 F9F94040 40404040 40404040    00000000 00000000 00000000 00000000   *.01.99          ................*
000120     00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *................................*
000160     00000000 00000000 00000000 82000170    00040000 00036D18 00000000 00000000   *................................*
000180     FF060009 80C00000 0000018A 018A018A    FF000190 FF000190 00000001 FFFF6528   *................................*
0001A0     00009A00 00009AF4 00009968 000099B4    00009AF0 80009B74 00009AD0 4000BB62   *.......4...........0........ ...*
0001C0     000117E0 00009BB4 00000040 00009B74    5000BCA4 6000A57A 00000030 0006F9F4   *........... ..................94*
C001E0     000000CC 000729C0 00000000 0006F000    5006E596 000729B8 A006E740 00000001   *..............0...V.......X ....*
C00200     000726D0 00067594 00065D40 00072798    4006E7AE 0001828C 00000000 00000000   *........... .... .X.............*
000220     00004E98 00000000 41500800 1A551821    92825098 18114010 50881804 58420014   *...................... .........*
000240     5834002C D5022015 30194770 0ED491F0    00214780 025A45E0 0E681B99 18A991FE   *....N........M.0................*
000260     30104770 02724873 00229170 70124780    02824393 001C43A2 002089A0 9000487A   *................................*
000280     302291FF 700247E0 0ED491A0 50984790    029E58F0 0FC445EF 000041C0 02B258B2   *.........M.........0.D..........*
C002A0     00041BAA 43A7C00A 89A00003 41DA52FC    07FC4012 001ED708 20082008 D4032000   *.................. ...P.....M...*
0002C0     5084927F 2004501B 000094FD 50984580    02F647F0 02E247F0 02EA4700 000045E0   *.................6.0.S.0........*
0002E0     071C1812 58E00FC8 07FE4180 02D245C0    02A247F0 03444810 0F9C1211 4740035C   *.......H.....K.....0......... ..*
C00300     91011001 47100352 4C710002 90231004    5001000C 92001004 D300100C 0021D201   *........ ...............L.....K.*
000320     0F9C1000 40105088 18A0D200 1008A023    45E00AD0 91EF7006 47708008 91FF0FB0   *.... .....K.....................*
000340     47500E2A 91107006 47100DE6 48AD0006    07FAD502 20150FD1 47800308 58A00024   *...........W......N....J........*
000360     4BA0508C 50A00024 18B09620 B02092F0    09771B99 58A00FBC 5090A000 47FC02E2   *...............0.............0.S*
000380     91102000 471003E6 41A05020 D200A000    302045C0 05E407BC 48A00044 54A05058   *.......W....K........U..........*
0003A0     4770065C 58AC7030 91042001 471003C0    58A20010 91012000 478003C0 58A20018   *................................*
C003C0     91082000 478003DA 50A05030 92085030    41A05028 D200502D 701850A0 004841C0   *....................K...........*
0003E0     066447F0 06249104 20014780 05D647F0    03889140 702C4710 05929101 70064770   *...0.........O.0... ............*
000400     040694E7 20019110 20014710 05709102    70064710 04D241A0 703140A0 503AD203   *...X.................K.... ...K.*
C00420     50007031 91012000 47100432 D2077030    20201BAA 43A70030 89A00004 41AA3020   *............K...................*
000440     91082001 471C0490 D5037033 A0064740    053ED503 7033A00A 4720053E 91027013   *........N...... ..N.............*
000460     471004D2 D5017031 A0044770 053E9104    30084780 0490D501 7035A008 4740048A   *...KN.................N...... ..*
000480     D5017035 A00C47C0 0490D201 7035A008    41A05038 41B00578 45C005E8 47700688   *N.........K................Y....*
0004A0     9D006000 47B004A0 48A00044 54A05058    477006A8 96A27006 D2062009 00419104   *........................K.......*
0004C0     00444780 8008945F 700691A0 50984790    800818B6 88B00008 89B00002 48CB52D4   *...............................M*
0004E0     4BC05096 41A07031 40A0C002 43B07030    89B00004 439B3020 4290C00D D202C011   *........ ...................K...*
000500     20119101 20004780 051CD202 C0112019    91082001 4780051C 9618C00D 50C00048   *..........K.....................*
000520     91027013 47800530 58A00048 47F005E0    45C00624 077C96A6 700647F0 066C58F3   *.............0.............0...3*
000540     001C58FF 000005EF 47F0055C 47F00554    47F00432 41E00960 47F00564 92422004   *.........0...0...0.......0......*
000560     41E00DA2 94FE7006 94DF2000 47F00752    58C20018 47F0051C D5037031 50004770   *.............0...B...0..N.......*
000580     06249602 70064060 70044010 701447F0    04CA4910 702A4770 0DD69148 702C4710   *...... ... ....0.........O......*
0005A0     05AE9101 70064780 0DD247F0 040E9407    702C94DF 200047F0 040E9110 20004710   *.........K.0...........0........*
0005C0     05D658A2 00189101 20004710 05D258A2    001047F0 05E0D200 50082018 41A05008   *.O...........K.....0..K.........*
0005E0     41C00664 41B00624 50A00048 91202000    47800604 910C402C 47800624 943F402C   *...................... ....... .*
000600     94DF2000 58F3001C 58FF0004 50B05074    05EF47F0 061E41E0 096447F0 075258B0   *.....3.............0.......0....*
000620     507407FB 92000048 91017006 47800638    91102001 4710063E D3000048 100C9C00   *........................L.......*
000640     600005A0 88A00018 42A20010 58900FC0    05B91899 40607004 40107014 58A02010   *.................... ... .......*
000660     04A005CC 4770068E 96A07006 43907004    1A994079 52F0D600 700C509A D7C0700C   *.................. ..0O.....P...*
000680     509A45E0 075207F8 D2C37031 50004720    070C58A0 004841A0 A00850A0 0040D206   *.......8K.................... K.*
0006A0     20090041 471006E4 18E096A0 70069106    00454770 0F8C9110 00444780 0714945F   *.......U........................*
0006C0     70069120 00444710 80049608 70069140    0044071E 91840044 47808008 41808004   *............... ................*
```

Figure 27. Sample of General Format Dump

OUTPUT COMMENTS

The following output comments are printed within the body of a formatted dump whenever IMDPRDMP is unable to locate, format and print a control block. These comments explain why the referenced control block is not printed within the dump listing, these output comments are separated from the main storage information by a blank line both before and after each output comment. Note: Output Comments produced when IMDPRDMP is processing GTF output are shown in Section 3 of this publication under the heading 'IMDPRDMP Output Comments - GTF Processing'.

DUPLICATE PREFIX FOLLOWS - ID 'A'

Explanation: While processing a dump from a Model 65 multiprocessing system, IMDPRDMP has determined that the CPU prefixes (CPUIDs) are the same. If the task that performs the dump is initiated on one CPU, interrupted, dispatched to the second CPU, and completed the rest of the processing on the second CPU, then the prefix shown in this output comment is that of the first CPU to which the task was assigned. Processing continues.

END OF FILE ON DUMP TAPE

Explanation: While trying to locate a block of main storage on the dump tape, IMDPRDMP reached the end of the tape. This message is printed only if IMDPRDMP is either trying to extract the CVT pointer or trying to extract an area of storage for printing.

Processing terminates. If IMDPRDMP did no formatting and the tape does not contain a low-speed dump produced by IMDSADMP, the job may be rerun using the CVT control statement to direct IMDPRDMP to the CVT in this dump. Low-speed dumps produced by IMDSADMP can not be formatted by IMDPRDMP.

ERROR FINDING REGION BOUNDARIES FOR TCB aaaaaa.

Explanation: While attempting to determine the region boundaries for the family of TCBs attached to the job step TCB at address aaaaaa, one of the following conditions occurred:

- IMDPRDMP found a chain with more than fifty partition queue elements (PQEs); or,
- IMDPRDMP found (1) a TCB family chain pointer, (2) a partition queue element (PQE) pointer (TCB + X'98'), or (3) a pointer within a PQE that:
  1. Addressed an area that was not on a word boundary; or,
  2. Addressed an area that was higher than the highest address in the dump; or,
  3. Could not be extracted from the dump decause either an I/O error was encountered while attempting to read the block containing the pointer or the block containing the pointer was missing from the dump; a possible cause for a missing block is that the routine that produced the dump encountered an I/O error while attempting to write the block.

Processing continues.

ERROR FORMATTING TCB

Explanation: One of the fields in the TCB required for formatting could not be extracted from the dump because:

- IMDPRDMP encountered an I/O error while attempting to read the block that contains the required data; or,
- The block containing the required data was missing from the dump; a possible cause is that the routine that produced the dump encountered an I/O error while attemtping to write the block.

Processing continues.

ERROR IN DEB CHAIN

Explanation: The routine that formats the data extent block (DEB) found one of the following errors:

- A DEB chain pointer:
  1. Was not on a word boundary; or,
  2. Addressed an area of main storage higher than the highest address in the dump; or,

- The address of the DEB was invalid causing the address of the DEB prefix (DEB - 16) to be zero or negative; or,

- A DEB chain pointer or one of the fields necessary to format the DEB could not be extracted from the dump because:
  1. IMDPRDMP encountered an I/O error attempting to read the block that contained the data; or,
  2. The block containing the data was missing from the dump; a possible cause is that the routine that produced the dump encountered an I/O error while attempting to write the block.

Processing continues.

ERROR IN EXTENT LIST

Explanation: While formatting the load list or job pack area of an MVT dump, IMDPRDMP encountered a contents directory entry (CDE) that had a block extent list with a relocation factor (extent list + 4) of zero or greater than twenty-five. A relocation factor of zero is an error; however, a value greater than twenty-five can be valid. The value of twenty-five was established by IMDPRDMP as a reasonable limit; it is improbable that a normal task would have a program that has more than twenty-five CSECTs causing it to get an extent list with a relocation factor greater than twenty-five. Processing continues with the next CDE.

ERROR IN JOB PACK QUEUE

Explanation: The routine that formats the job pack area encountered one of the following errors:

- A job pack queue chain pointer addressed an area that:
  1. Was not on a word boundary, or,
  2. Was greater than the highest address in the dump.

- A job pack queue chain pointer or one of the fields in a job pack area control block could not be extracted from the dump because:
  1. IMDPRDMP encountered an I/O error attempting to read the block containing the needed data, or,
  2. The block containing the needed data was missing from the dump; a possible cause is that the routine that produced the dump encountered an I/O error while attempting to write the block.

Processing continues.

ERROR IN LOAD LIST

Explanation: The load list print routine encountered one of the following errors:

- A pointer in the load list control block chain referenced an area of main storage that:
  1. Was not on a word boundary, or
  2. Was greater than the highest address in the dump.

- A field in a load list queue control block could not be extracted from the dump because:
  1. IMDPRDMP encountered an I/O error attempting to read the block that contained the data needed to format the load list; or,
  2. The block containing the data was missing from the dump; a possible cause is that the routine that produced the dump encountered an I/O error while attempting to write the block.

Processing continues.

ERROR IN TCB CHAIN TCB aaaaaa

Explanation: The routine that formats the TCBs encountered one of the errors given below; the address of the TCB associated with the error replaces the aaaaaa field of the output comment.

- A TCB pointer for one of the TCBs in the TCB family chain addressed an area not on a word boundary; or,

- A TCB pointer or the TIOT pointer in the TCB at location aaaaaa points to an area that could not be extracted from the dump because:
  1. IMDPRDMP encountered an I/O error while attempting to read the block that contains the pointer; or,
  2. The routine that produced the dump encountered an I/O error while writing the block that contains the pointer; therefore, the block is missing from the dump.

ERROR IN TIOT

Explanation: The format routine found one of the following errors:

- The task input output table (TIOT) pointer (TCB + X'C') was not on a word boundary; or,
- One of the fields required to format the TIOT could not be extracted from the dump because:
  1. IMDPRDMP encountered an I/O error while attempting to read the block that contains the required data, or,
  2. The block containing the required data was missing from the dump; a possible cause is that the routine that produced the dump encountered an I/O error while attempting to write the block.

ERROR WHILE FORMATTING CONTROL BLOCKS .... CONTINUING

Explanation: While building a list of job step TCB's for all partition regions in the dump data set, PRDMP encountered one of the following conditions:

1. One of the TCB chain pointers was greater than the highest address in the dump.
2. One of the TCB chain pointers addressed an area that was missing from the dump data set.

PRDMP will attempt to use the partial list and continue with its formatting.

ERROR WHILE FORMATTING PSCB

Explanation: One of the following errors occurred while IMDPRDMP was formatting the protected step control block (PSCB):

- The address of the PSCB in the time sharing job block extension (TJBX) was greater than the highest main storage address in dump; or,
- An I/O error occurred while reading the block of dump information that contained the needed data; or,
- A block of dump information containing part of the PSCB was not found on either the dump or swap data sets.

Processing continues. IMDPRDMP attempts to format the control blocks for the next TSO user.

ERROR WHILE FORMATTING RCB

Explanation: One of the following errors occurred while IMDPRDMP was formatting the time sharing region control blocks (RCBs):

- The address of the RCB table in the time sharing communication vector table (TSCVT) was greater than the highest main storage address in the dump; or,
- An I/O error occurred while reading the block of dump information that contained the needed data; or,
- A block of dump information containing part of an RCB was not found on the dump data set. This happens when an I/O error occurred while the dump routine was writing the data onto the dump data set.

Processing continues. IMDPRDMP attempts to format the next entry in the RCB table.

ERROR WHILE FORMATTING SWAP CONTROL BLOCK

Explanation: One of the following errors occurred while IMDPRDMP was formatting the swap control block (SWAP DCB):

- The address of the SWAP DCB in the time sharing communication vector table (TSCVT) was greater than the highest main storage address in the dump; or,

- An I/O error occurred while reading the block of dump information that contained the data; or,

- A block of dump information containing part of the SWAP DCB was not found on the dump data set. This happens when an I/O error occurred while the dump routine was writing the data onto the dump data set.

Processing continues. IMDPRDMP attempts to continue formatting the time sharing job block (TJB).

ERROR WHILE FORMATTING TAXE

Explanation: One of the following errors occurred while IMDPRDMP was formatting the terminal attention exit element (TAXE):

- The address of the TAXE in the time sharing job block extension (TJBX) was not aligned in a fullword boundary; or,

- The address of the TAXE in the TJBX was greater than the highest main storage address in the dump; or,

- An I/O error occurred while reading the block of dump information that contained the needed data; or,

- A block of dump information containing part of the TJBX was not found on the dump or swap data sets.

Processing continues. IMDPRDMP attempts to format the control blocks for the next TSO user.

ERROR WHILE FORMATTING TJB

Explanation: One of the following errors occurred while IMDPRDMP was formatting the time sharing job block (TJB):

- The address of the TJB table in the time sharing communication vector table (TSCVT) was greater than the highest main storage address in the dump; or,

- An I/O error occurred while reading the block of dump information that contained the needed data; or,

- A block of dump information containing part of the TJB was not found on the dump data set. This happens when an I/O error occurred while the dump routine was writing the data onto the dump data set.

Processing continues. IMDPRDMP attempts to format the next active TJB.

ERROR WHILE FORMATTING TJBX

Explanation: One of the following errors occurred while formatting the time sharing job block extension (TJBX):

- The terminal job block (TJB), that contained the address of the TJB was not aligned on a fullword boundary; or,

- The address of the TJBX in the TJB was greater than the highest address in the system dump; or,

- An I/O error occurred while reading the block of dump information that contained the needed data; or,

- A block of dump information containing part of the TJBX was not found on either the dump or swap data sets.

Processing continues. IMDPRDMP attempts to format the control blocks associated with the next TSO user.

ERROR WHILE FORMATTING TSB

Explanation: One of the following errors occurred while IMDPRDMP was formatting the terminal status block (TSB):

- The address of the TSB table in the time sharing communication vector table (TSCVT) was greater than the highest main storage address in the dump; or,

- An I/O error occurred while reading the block of dump information that contained the needed data; or,

- A block of dump information containing part of the TSB was not found on the dump data set. This happens when a I/O error occurred while the dump routine was writing the data onto the dump data set.

Processing continues. IMDPRDMP attempts to format the associated time sharing job block extension (TJBX).

ERROR WHILE FORMATTING TSCVT

<u>Explanation</u>: One of the following errors occurred while IMDPRDMP was formatting the time sharing communication vector table (TSCVT):

- The address of the TSCVT in the communication vector table (CVT) was greater than the highest main storage address in the dump; or,
- An I/O error occurred while reading the block of dump information that contained the needed data; or,
- A block of the dump information containing part of the TSCVT was not found on the dump data set. This happens when a I/O error occurred while the dump routine was writing the data onto the dump data set.

Processing continues. IMDPRDMP attempts to format the time sharing region control blocks (RCBs).

ERROR WHILE FORMATTING USER MAIN STORAGE MAP

<u>Explanation</u>: One of the following errors occurred while IMDPRDMP was formatting the user main storage map (UMSM):

- The address of the UMSM associated with the region control task (RCT) or time sharing job block (TJB) was greater than the highest main storage address in the dump; or,
- An I/O error occurred while reading the block of dump information that contained needed data; or,
- A block of dump information containing part of the UMSM was not found in the dump data set. This happens when an I/O error occurred while the dump routine was writing the data onto the dump data set.

Processing continues. IMDPRDMP attempts to continue formatting with the terminal status block (TSB).

FORMAT ERROR DURING TCB SUMMARY

<u>Explanation</u>: The routine that prints the TCB summary must extract a TCB completion code (TCB + X'16') or a TCB family chain pointer from the dump. In this case, IMDPRDMP was unable to do so because:

- IMDPRDMP encountered an I/O error while attempting to read the block containing the completion code or pointer; or,
- The block containing the completion code or pointer was missing from the dump; a possible cause is that the routine that produced the dump encountered an I/O error while attempting to write the block.

Processing for the current control statement is terminated.

FORMAT ERROR IN MAIN STORAGE BLOCKS

<u>Explanation</u>: While formatting main storage control blocks, IMDPRDMP encountered one of the following errors:

- A pointer in a main storage control block addressed an area that:
  1. Was not on a word boundary; or,
  2. Was higher than the maximum address in the dump; or,
- One of the fields in a main storage control block could not be extracted from the dump because:
  1. IMDPRDMP encountered an I/O error while attempting to read the block that contains the required field; or,
  2. The block containing the required field is missing from the dump; a possible cause is that the routine that produced the dump encountered an I/O error while attempting to write the block.

Processing continues.

INFINITE LOOP IN DEB CHAIN

Explanation: While formatting the data extent blocks (DEBs), IMDPRDMP found more than 200 DEBs chained to the TCB. The limit of 200 DEBs prevents IMDPRDMP from looping. When the limit is exceeded, a loop is assumed which causes this comment to be printed. Processing continues after the first 200 DEBs are printed.

INFINITE LOOP IN PQES

Explanation: The main storage print routine found more than 50 partition queue elements (PQEs) chained to the TCB. A limit of 50 PQEs has been established by IMDPRDMP to prevent a possible looping condition. When the limit is exceeded, a loop is assumed and this comment is issued. The first 50 PQEs are printed and then processing continues.

INFINITE LOOP IN RB CHAIN

Explanation: The RB print routine found more than 50 request blocks (RBs) on the RB chain. A limit of 50 RBs has been established within IMDPRDMP to prevent a possible looping condition. When the limit is exceeded, a loop is assumed and this comment is issued. The first 50 RBs are printed and then processing continues.

INVALID TIOT

Explanation: While formatting the task input output table (TIOT), the FORMAT routine found an invalid job name in the TIOT. To be valid, the first character of the job name must be A through Z, or $, #, @ or a blank (X'40'). Processing continues.

NO ELEMENTS ON LOAD LIST

Explanation: The load list pointer in the TCB (displacement X'24') is zero. The zero pointer indicates that (1) no programs were loaded by the LOAD macro instruction or (2) the load list pointer was overlaid with zero. Processing continues.

NO EXTENT LIST

Explanation: While formatting the load list and job pack queue for an MVT dump, IMDPRDMP encountered zeros in the extent list pointer (CDE + '20') in a major contents directory entry (CDE). This zero pointer usually indicates an error condition in which the extent list pointer was overlaid with zeros. Processing continues with the next CDE.

NO LINK PACK AREA QUEUE

Explanation: In MFT, an LPAMAP was requested but the link pack area queue pointer (CVT + X'BC') was zero. Processing continues.

NO MAJOR QCBS

Explanation: The QCB TRACE routine found zeros as the pointer to the first major queue control block (QCB). This indicates that no resources have been enqueued at the time of the dump or that the pointer to the QCB queue has been overlaid with zeros. Processing continues.

NOTHING IN JOB PACK

Explanation: In MVT, the job pack queue field of the TCB (TCB + X'2C') is zero. In MFT, the partition information block (PIB) field (TCB + X'7C') or the job pack queue pointer (PIB + X'24') is zero. PCP does not have a job pack pointer; therefore, this comment does not appear in a PCP dump. A zero job pack queue pointer is usually a normal condition, especially for a system task. Processing continues.

RB FORMAT ERROR

Explanation: While formatting a request block (RB), the RB print routine found that the request block (RB) chain pointer addressed an area of main storage that:

- Was not on a word boundary; or,
- Was higher than the highest address in main storage; or,
- Could not be extracted from the dump because:

1. IMDPRDMP encountered a I/O error while attempting to read the block that contained the pointer; or,
2. The block that contained the pointer was missing from the dump. One possible cause for this is that the program that produced the dump may have encountered an I/O error while writing the block.

- A field in the RB, or a contents directory entry (CDE) associated with the RB, necessary to formatting the RB could not be extracted from the dump. Either IMDPRDMP encountered an I/O error while trying to read the block, or the block that contained the pointer is missing from the dump.

REGISTERS FROM OTHER CPU ARE INVALID-NOT FORMATTED

<u>Explanation</u>: Multiprocessing systems only. Only the registers for the CPU in which the dump program was executed will be displayed on the dump listing. This can occur when the dump is taken on a multiprocessing system either when the NOMP option of IMDSADMP is used or when the direct control feature is not operational.

TASK HAS NO OPEN DATA SETS

<u>Explanation</u>: IMDPRDMP found the data extent block (DEB) pointer in the TCB (TCB + X'8') to be zero. This situation indicates that there were actually no open data sets or the DEB pointer in the TCB was overlaid with zeros. Processing continues.

TASK HAS NO TIOT

<u>Explanation</u>: While attempting to format the task input output table (TIOT), IMDPRDMP found that the TIOT pointer (TCB + T'C') was either zero or larger than the highest address in the dump. The zero TIOT pointer could be a normal condition for a system communication task, but for a problem program task this is an error condition. Processing continutes.

TASK HAS TERMINATED

<u>Explanation</u>: After formatting a TCB, this comment is printed below the TCB if the first bit (the terminated bit) of the flag byte at X'21' of the TCB is set. Processing continues with the next TCB.

TCB CHAIN ERROR IN F03 PRINT ROUTINE
TCB aaaaaa...CONTINUING WITH NEXT TCB

<u>Explanation</u>: The Print F03 routine encountered a TCB chain pointer that:

- Was not on a word boundary; or,

- Addressed an area that could not be extracted from the dump because:
  1. The pointer addressed an area higher than the maximum address in the dump; or,
  2. IMDPRDMP encountered an I/O error trying to read the record containing the area addressed by the pointer; or,
  3. The block containing the addressed area was missing from the dump, probably because the routine that produced the dump encountered an I/O error while attempting to write the block.

The address of the TCB associated with the error replaces the aaaaaa field in the message. Processing continues with the next TCB.

GUIDE TO STORAGE DUMPS

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

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

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 dumped 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 invoke by system routines and these routines can be identified by module names appearing in the most recent request block (RB) for the failing task. 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.

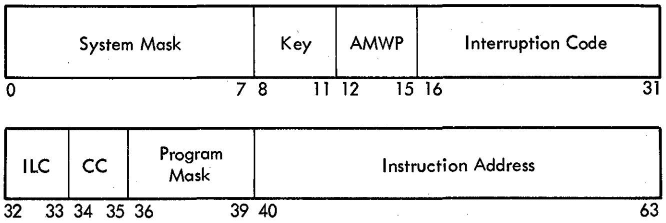
Main storage locations from zero to 128 (hexadecimal 80) are permanently assigned and contain hardware control words. Figure 28 shows these fields, their location, their length, and their purpose.

Address		Length	
Dec	Hex	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

Figure 28. Permanently Assigned Hardware Control Words

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. 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

MFT System (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. Figure 28 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 System (With Subtasking)</u>

For MFT with 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 Figure 32.

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.

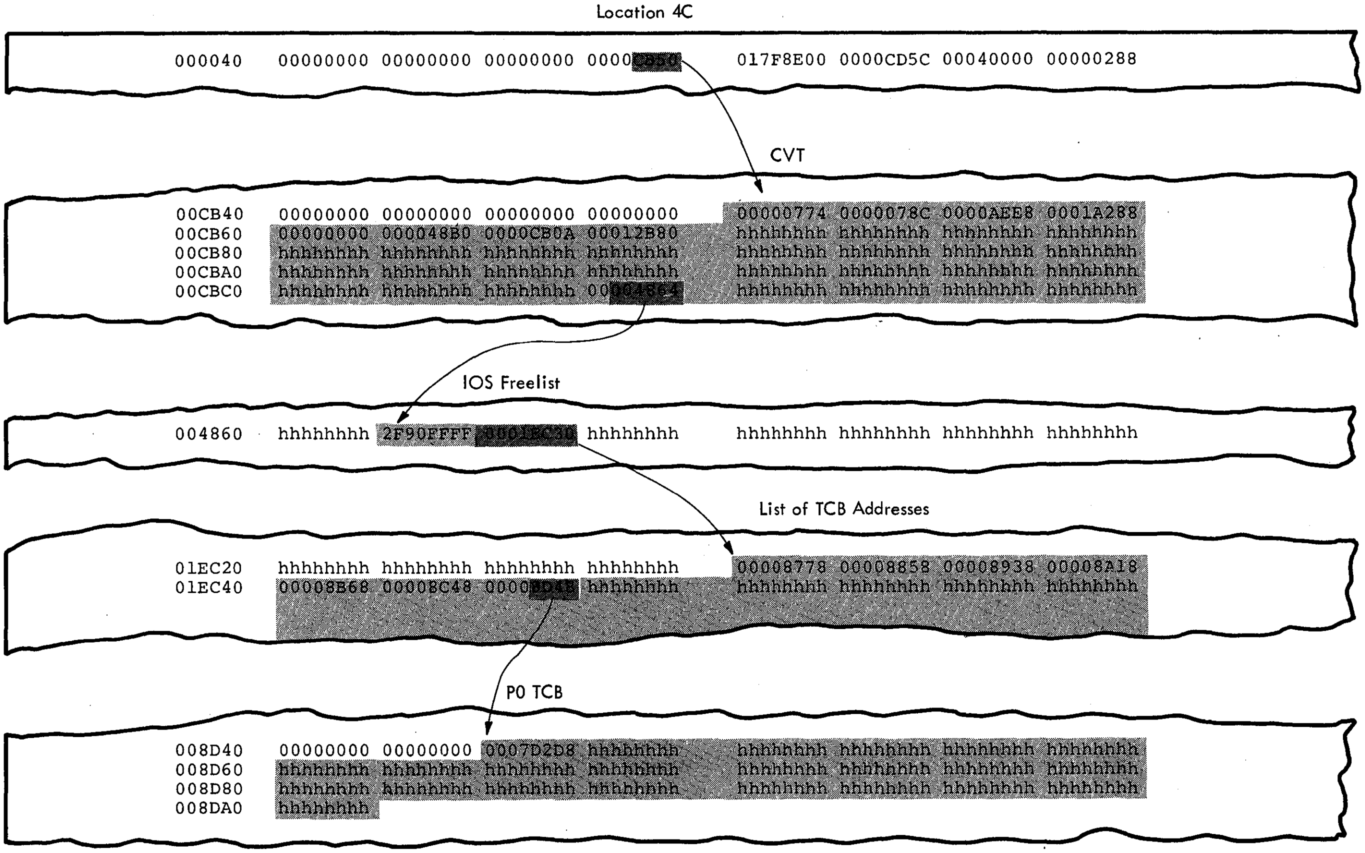
To recreate the task structure within any partition, simply locate the job step TCB, and follow the TCB pointers - as explained in the previous section.

<u>MVT System</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 Figure 32.

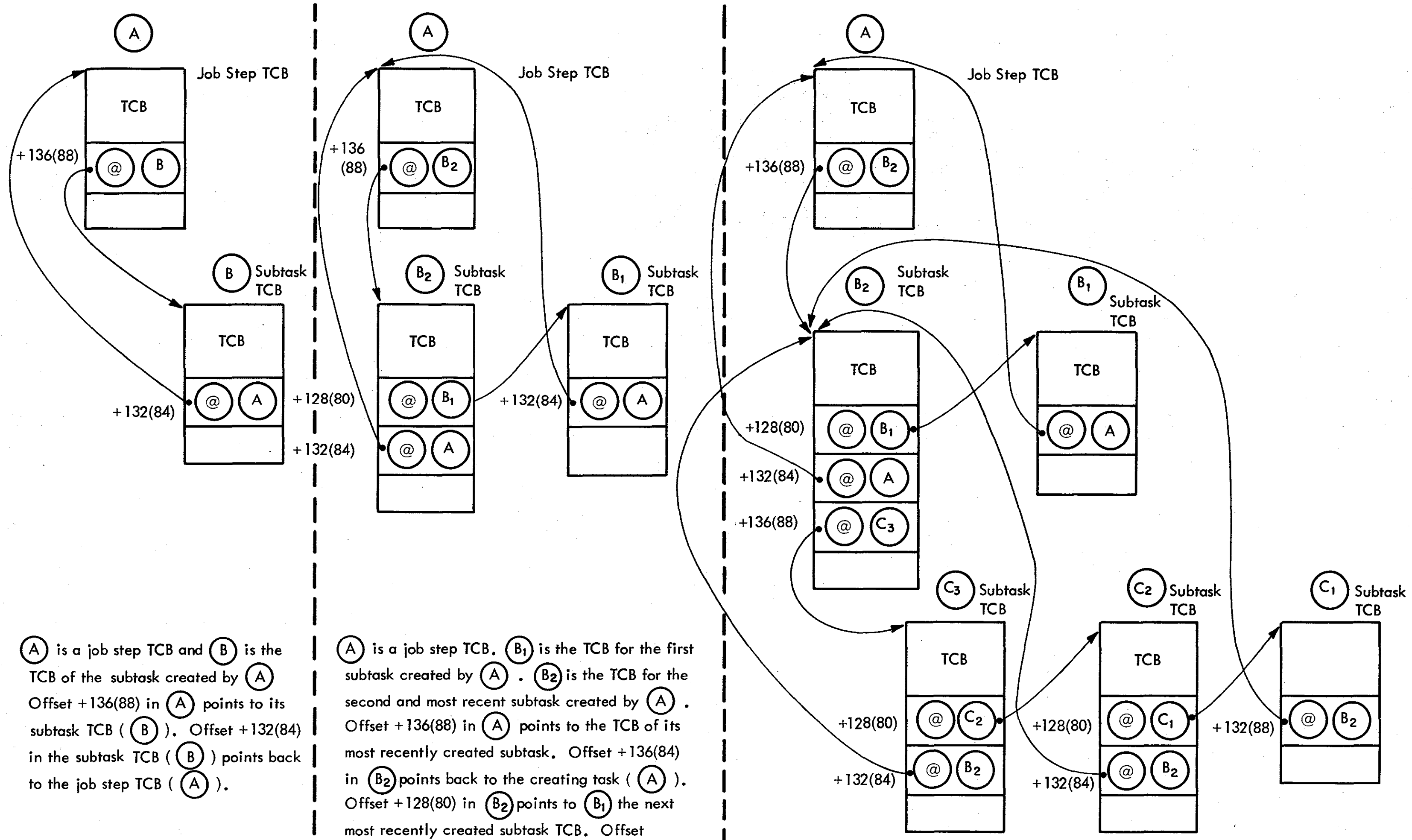
If the first word of the current TCB points to itself, there are no ready tasks to be dispatched, and the system has been placed in an enabled wait state. This TCB, now in control, is called the <u>system wait TCB</u>.

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. Figure 30 shows how to locate all of the TCBs in the system.



```
000040   00000000 00000000 00000000 0000CB50   017F8E00 0000CD5C 00040000 00000288

00CB40   00000000 00000000 00000000 00000000   00000774 0000078C 0000AEE8 0001A288
00CB60   00000000 000048B0 0000CB0A 00012B80   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
00CB80   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
00CBA0   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
00CBC0   hhhhhhhh hhhhhhhh hhhhhhhh 00004864   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh

004860   hhhhhhhh 2F90FFFF 0001EC30 hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh

01EC20   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   00008778 00008858 00008938 00008A18
01EC40   00008B68 00008C48 00008D48 hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh

008D40   00000000 00000000 0007D2D8 hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
008D60   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
008D80   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
008DA0   hhhhhhhh
```

Figure 29. Finding the Partition TCBs in MFT



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).

Figure 30. Finding the TCB

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 TCBs 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 31 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.

TASK STATUS - ACTIVE RB QUEUE

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 back to the TCB.

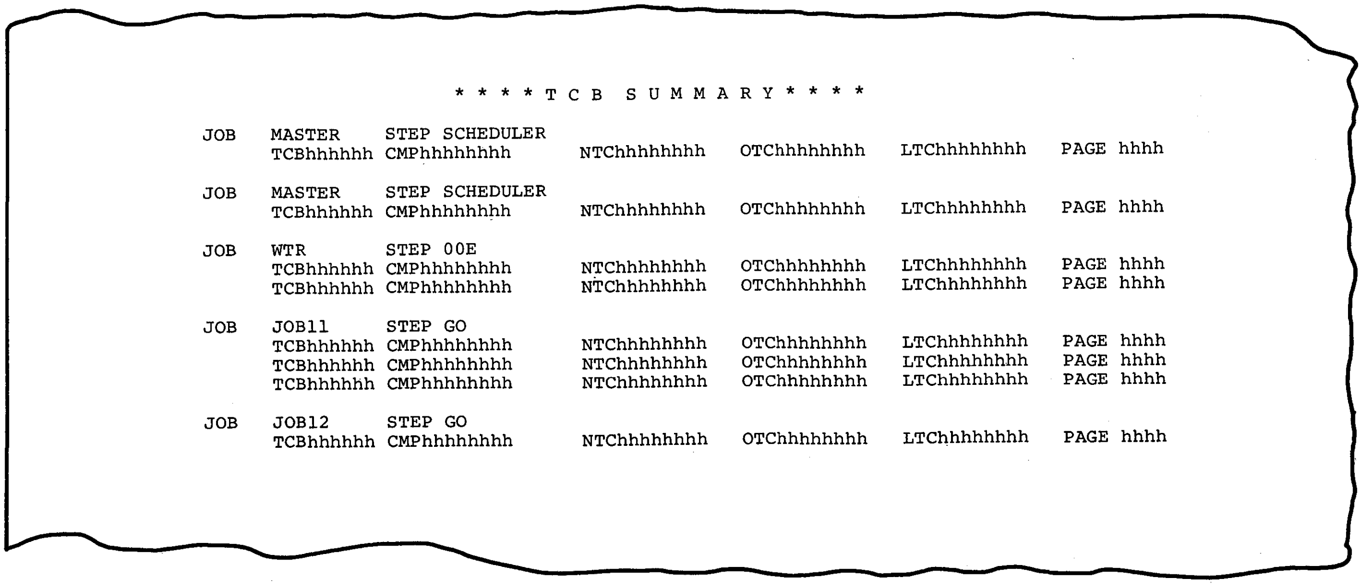
You can determine the idenity of the load module by looking either in the first and/or second words of the RB for its EBCDIC name or in the last 3 digits of the resume PSW in the previous RB for its SVC number. The entry point to the module is in the last 3 bytes of the fourth word in the RB, RB-13(D).

In an 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 (CDE).

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 32 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 (unless marked non-dispatchable). Such a task will be dispatched or become the current task when all TCBs of higher priority are waiting for



```
                         * * * * 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 31. IMDPRDMP TCB Summary

the completion of an event. To determine the events for which a task is waiting, use the 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 (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)

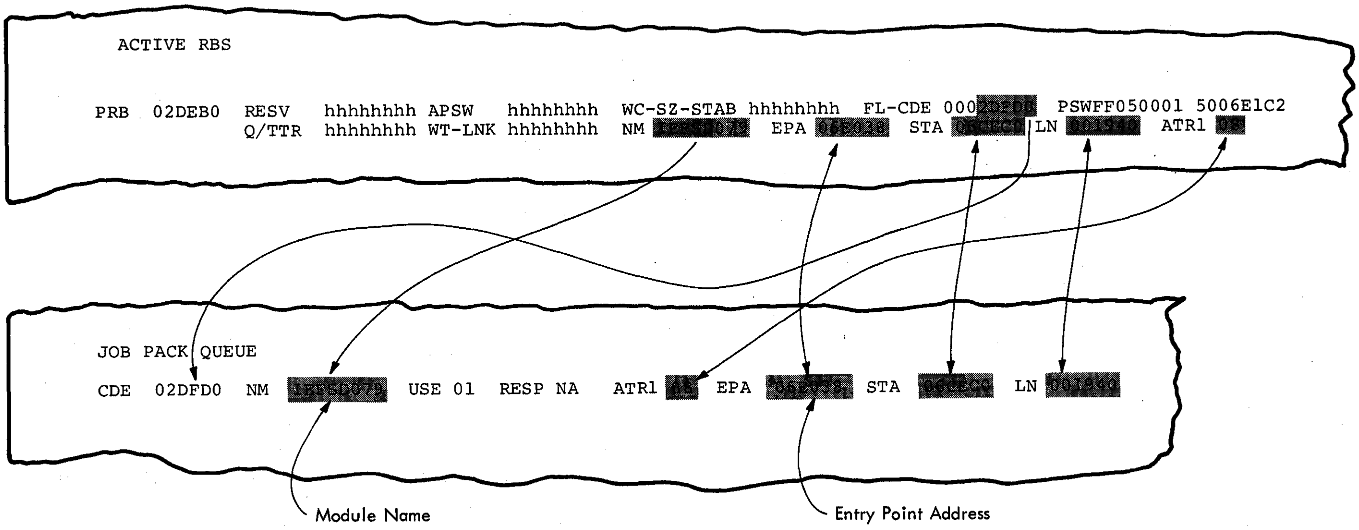
To construct the load list associated with the task in control, look at the tenth word in the TCB, TCB+36(24), for a pointer to the most recent load list entry (LLE). Each LLE contains the address of the next most recent entry (bytes 0-3), the count (byte 4), and the address of the CDE for the associated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

Job Pack Area Queue (MFT With Subtasking, MVT)

In systems with MFT with subtasking or MVT control programs, 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 Status-Active RB Queue".

MFT System: 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



```
ACTIVE RBS

PRB  02DEB0  RESV   hhhhhhhh APSW   hhhhhhhh  WC-SZ-STAB hhhhhhhh  FL-CDE 0002DFD0  PSWFF050001 5006E1C2
             Q/TTR  hhhhhhhh WT-LNK hhhhhhhh  NM IEFSD079  EPA 06E038  STA 06CEC0 LN 001940  ATR1 08
```

```
JOB PACK QUEUE

CDE  02DFD0  NM IEFSD079  USE 01  RESP NA  ATR1 08  EPA 06E038  STA 06CEC0  LN 001940
```

Figure 32. Determining Module From CDE in MVT

contains zeros if the WLE is the last one. The second word points 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 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.

MVT System: 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 MFT 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 (MFT)

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) (MVT): 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 33 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

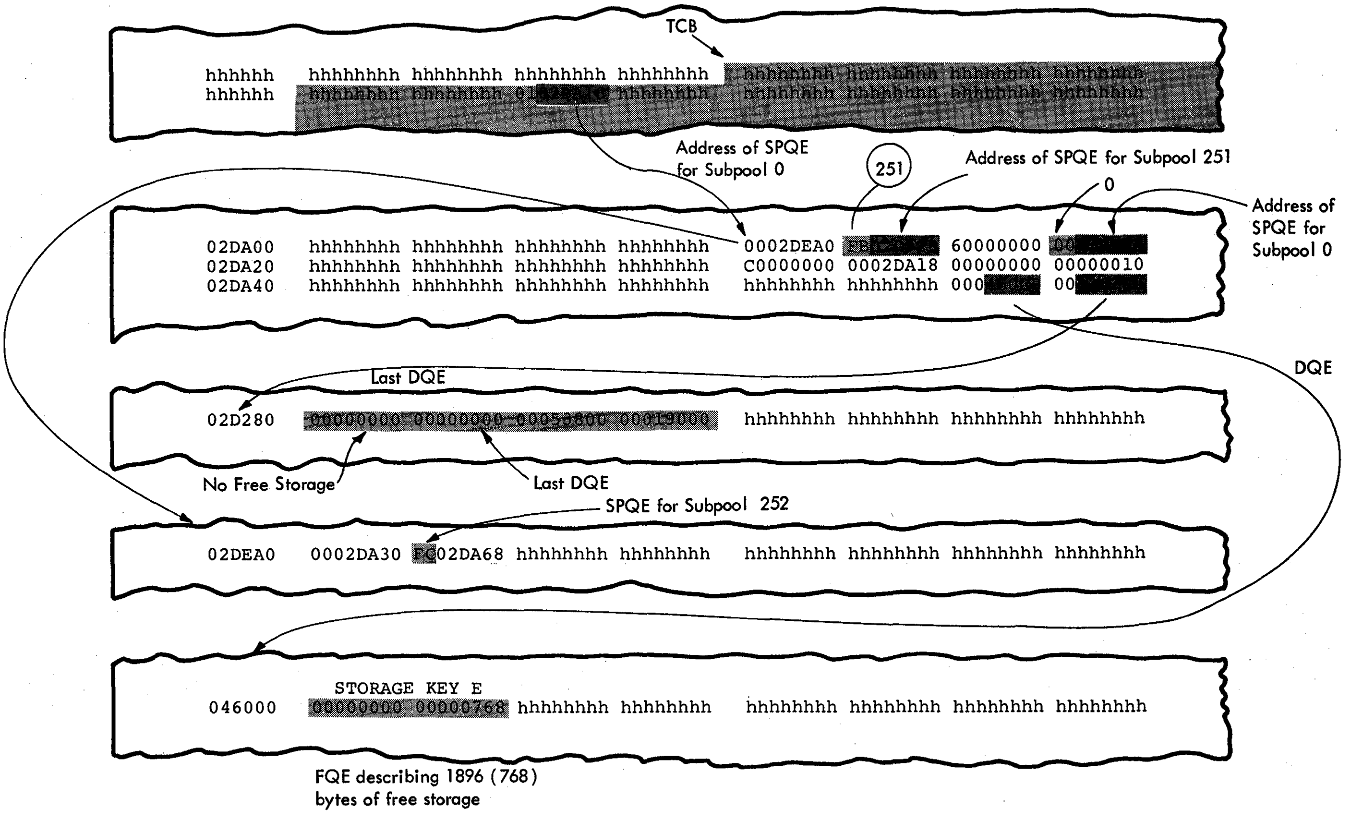
You can find unit information for each device in your system in the unit control block (UCB) for that device. The address of the UCB is contained in the last 3 bytes of the ninth word of the DEB, DEB+33(21). If the DEB queue is empty, scan the dump around location 4096(1000) for words whose fifth and 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 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, with the exception of those for devices with extended sense, begin in the last two bytes of the sixth UCB word, UCB+22(16), and continue from 1 to 6 bytes depending on the device type. For the extended sense devices, UCB+22 and UCB+23 are ignored. UCB+24(18) in this case contains the number of bytes of sense information to be found starting at the address specified in UCB+25(19). Sense bytes are given in Appendix G of this publication.

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



```
hhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
hhhhhh   hhhhhhhh hhhhhhhh 01       hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh

02DA00   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   0002DEA0 FB       60000000 00
02DA20   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   C0000000 0002DA18 00000000 00000010
02DA40   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh 000      00

02D280   00000000 00000000 00053800 00019000   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh

02DEA0   0002DA30 FC02DA68 hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh

           STORAGE KEY E
046000   00000000 00000768 hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
```

TCB
Address of SPQE for Subpool 0
251
Address of SPQE for Subpool 251
0
Address of SPQE for Subpool 0
DQE
Last DQE
No Free Storage
Last DQE
SPQE for Subpool 252
FQE describing 1896 (768) bytes of free storage

Figure 33. Subpool Descriptions in MVT - IMDPRDMP Storage Print

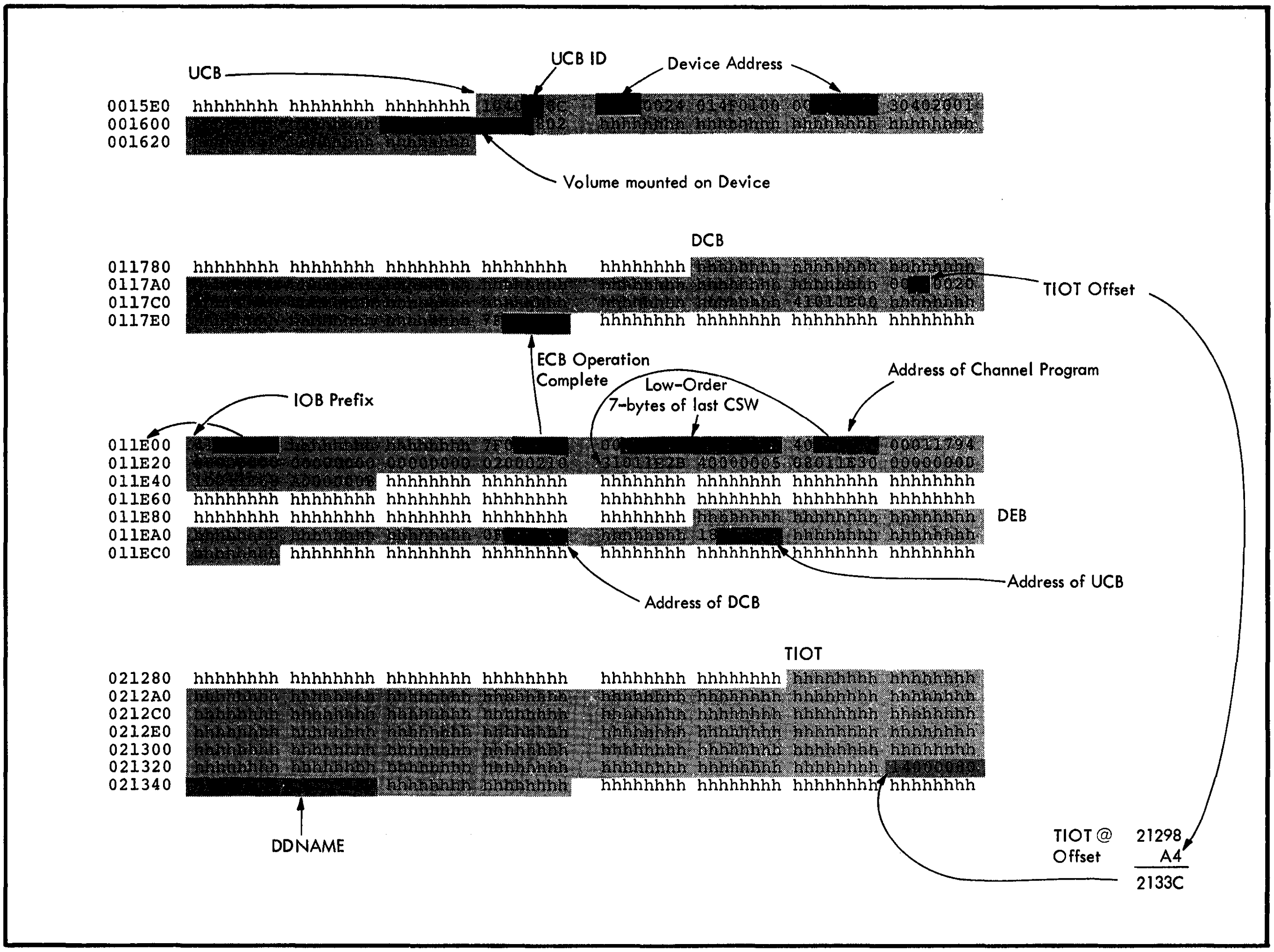


Figure 34. I/O Control Blocks

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 F. 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 34 shows the DEB, UCB, DCB, and IOB for a BSAM data set.

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.

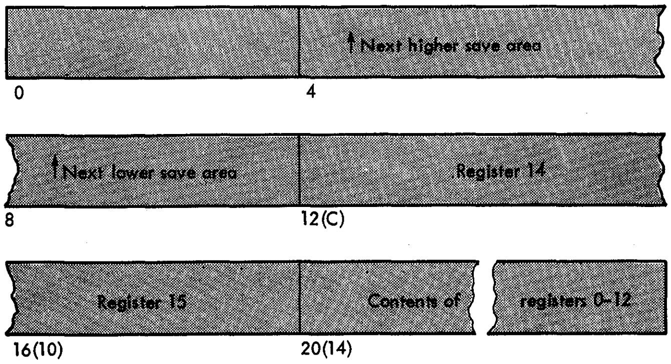
For a more detailed description of the TSO control blocks formatted by the IMDPRDMP program, see the Control Block and/or TSO Control Program PLM publications.

Tracing aids available are the save area chain, trace option, and Generalized Trace Facility (GTF). This section provides a description of each tracing aid, and, for GTF, describes its output after processing by the IMDPRDMP service aid.

Save Area Chain

The save area chain is edited and clearly identified in ABEND/SNAP dumps, and can be located easily in storage dumps produced by system dump facilities or the IMDSADMP service aid.

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 35.

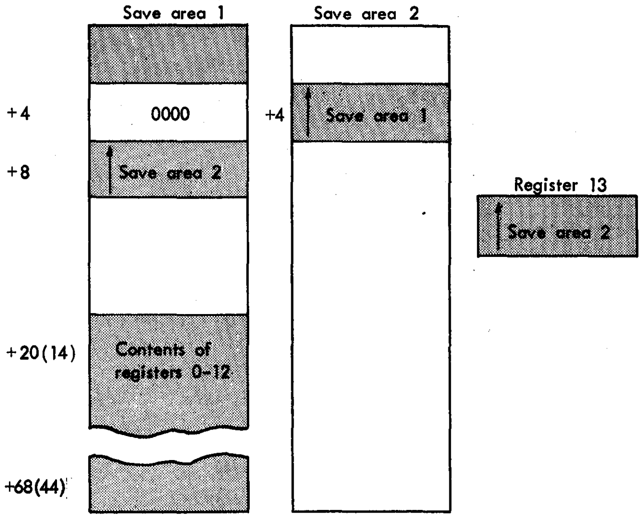


Figure 35. Save Area Trace

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 OPTION

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. 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 MFT: Trace table entries for systems with MFT are 4 words long and represent SIO, I/O, SVC and dispatcher task-switching interruptions. Figure 36 shows the word contents of each type of entry.

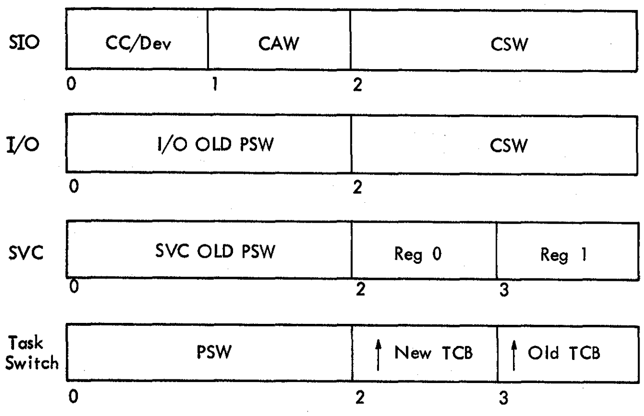


Figure 36. Trace Table Entries (MFT)

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 37 shows the word contents of trace table entries for SNAP dumps and stand-alone dumps. Figure 38 shows the contents of trace table entries as filled by MVT with Model 65 multiprocessing. (SSM -- set system mask -- entries are optional.)

INTERPRETING TRACE TABLE ENTRIES

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 MFT: Trace table entries for systems with MFT are 4 words long and represent occurrences of SIO, I/O, SVC, and task-switching interruptions. Figure 39 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).

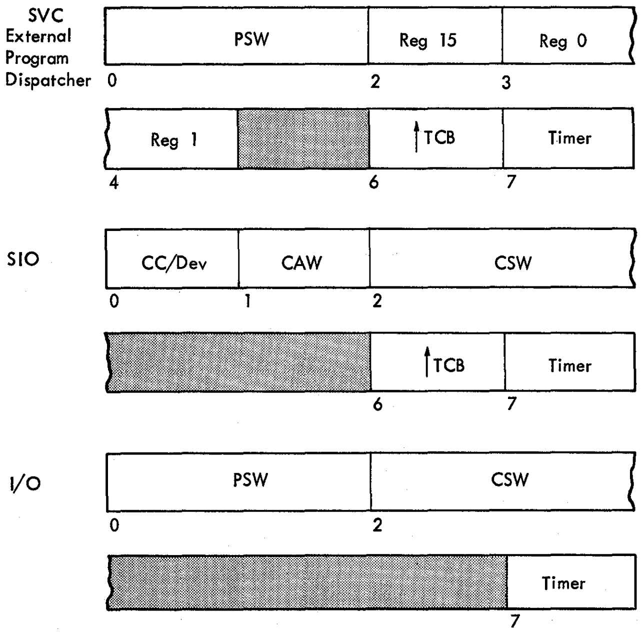


Figure 37. Trace Table Entries (MVT)

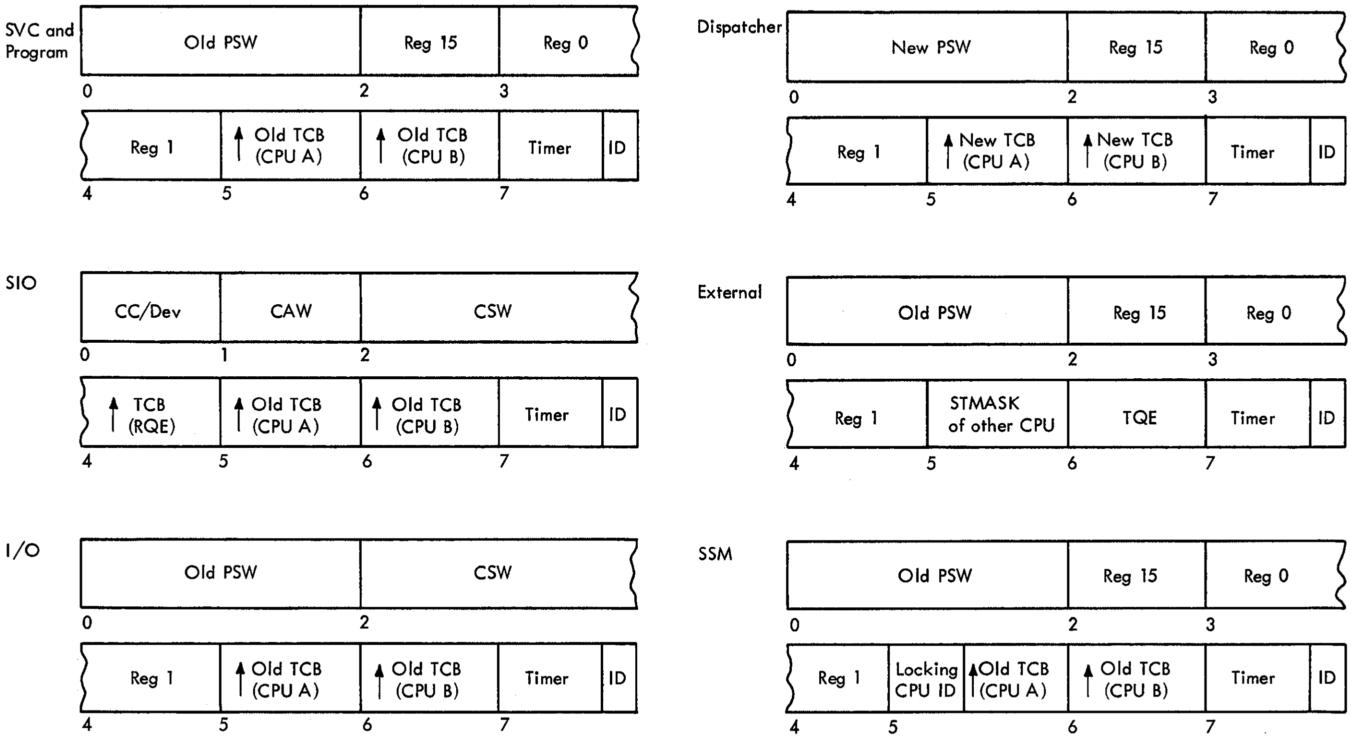


Figure 38. Trace Table Entries (MVT with Model 65 multiprocessing)

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 B.

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 40 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.

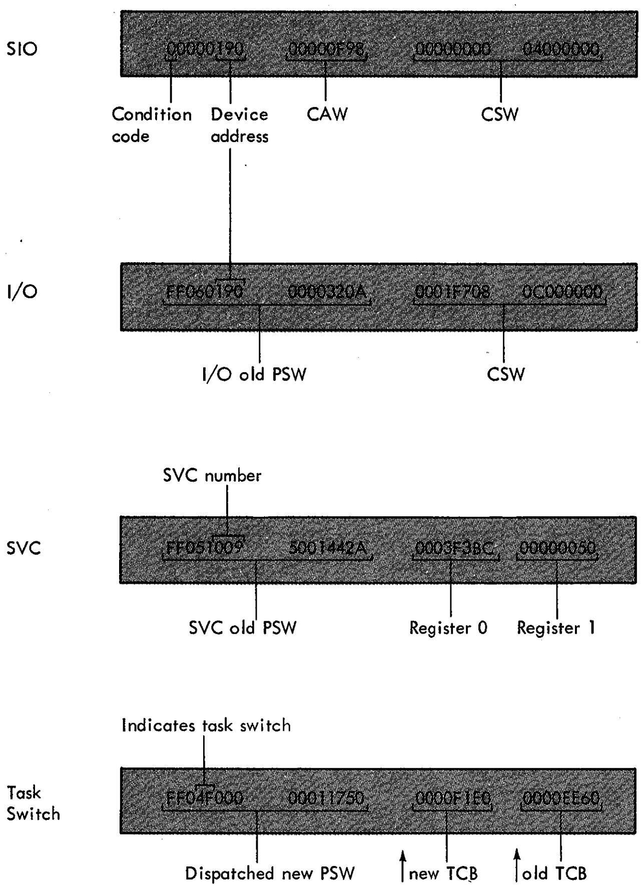


Figure 39. Sample Trace Table Entries (MFT)

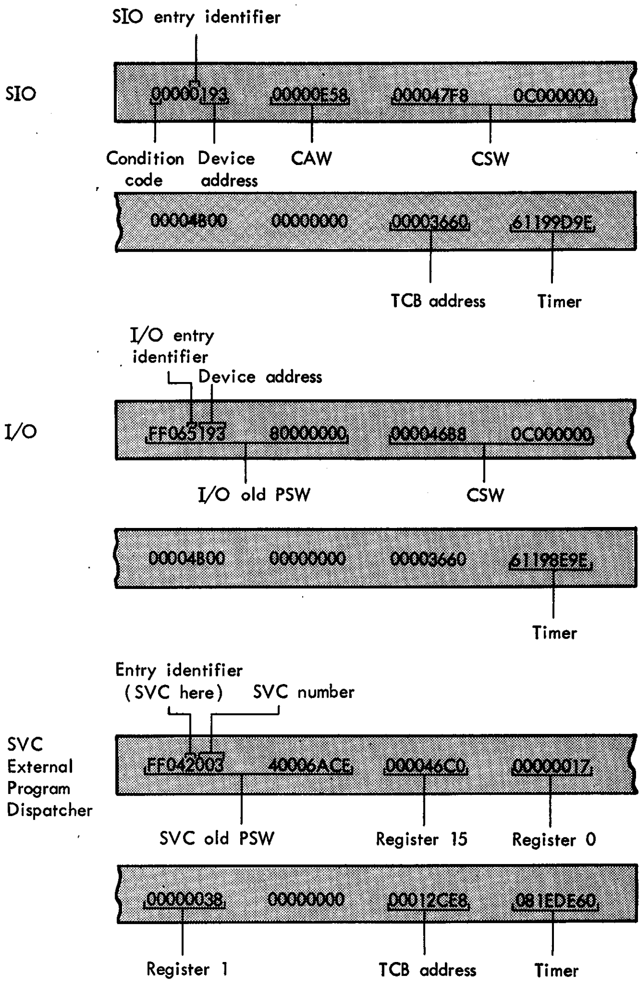


Figure 40. Sample Trace Table Entries (MVT)

Generalized Trace Facility

The Generalized Trace Facility (GTF) traces system and application program events and records information about these events. Trace records can be stored internally -- in a table similiar to the trace table of the Trace Option -- or they can be recorded externally in a data set that becomes input to the IMDPRDMP service aid program. (When stored internally the trace table is formatted in ABEND/SNAP dumps.) The IMDPRDMP service aid edits and formats the GTF external trace records as specified in an EDIT control statement.

This section describes the output of GTF; it does not tell how to use GTF. For a description of the functions performed by GTF and IMDPRDMP refer to the Service Aids publication.

System events traced by GTF in MFT, MVT, and MVT-M65MP systems are:

IO interrupts
SIO operations
SVC interrupts
Program interrupts
External interrupts
Task Switches by the system dispatcher
SSM interrupts in multi-processing systems

GTF MINIMAL TRACE RECORDS

The following material describes the records produced under the minimal trace option (SYSM) of GTF. The formats described appear in both ABEND/SNAP dumps (under the heading GTF TRACE TABLE) and in IMDPRDMP output. Minimal trace records are produced for IO and PCI/IO, SIO, SVC, PGM, EXT, DSP, and SSM events.

IO and PCI/IO Minimal Trace Record

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

Figure 41. IO and PCI/IO Minimal Trace Record

{A}
{B}
    appears in MVT-M65MP system records; identifies the CPU associated with the event.

IO }
PCI}
    identifies the type of trace record.

OLD PSW hhhhhhhh hhhhhhhh
    the program status word that was current at the time the IO or PCI/IO interrupt occurred.

CSW hhhhhhhh hhhhhhhh
    the channel status word associated with the IO or PCI/IO interrupt being traced.

```
       {********}
RQE TCB{hhhhhhhh}
       {N/A     }
```

****
    indicates that an error occurred while gathering the information.

hhhhhhhh
    is the address of the TCB of the task for which this I/O operation is being performed.

N/A
    indicates the interrupt was unsolicited: either the I/O supervisor did not issue an SIO instruction to the device; or there is no valid UCB for the device.

OLD TCB hhhhhhhh
    in MFT and MVT system trace records, the address of the TCB for the task that was in control when the interrupt occurred.

    in MVT-M65MP systems the OLA and OLB fields replace the OLD TCB field and contain the address of the TCB for the task in control of CPU A and CPU B respectively, at the time the interrupt occurred.

SIO Minimal Trace Record

```
 (A)                                                                  (********)(OLA hhhhhhhh  OLB hhhhhhhh)
{   } SIO  CC/DEV/CAW hhhhhhhh hhhhhhhh  CSW hhhhhhhhhhhhhhhh  RQE TCB{hhhhhhhh}{OLD TCB hhhhhhhh          }
 (B)                                                                  (N/A     )(O                          )
```

Figure 42. SIO Minimal Trace Record

{A}
{B}

appears in MVT-M65MP system records; identifies the CPU associated with the event.

SIO

identifies the type of trace record.

CC/DEV/CAW hhhhhhhh hhhhhhhh

displays the SIO condition code, the device address, and the CAW (channel address word) for the I/O operation just initiated.

The first four digits represent the condition code returned from the SIO operation; the next four digits represent the device address; and the last eight digits represent the CAW.

CSW hhhhhhhh hhhhhhhh

the channel status word associated with this event.

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

****

indicates that an error occurred while gathering the information.

hhhhhhhh

is the address of the TCB of the task for which this I/O operation is being performed.

N/A

indicates the interrupt was unsolicited, i.e., the I/O supervisor did not issue an SIO instruction to the device; or, there is no valid UCB for the device.

OLD TCB hhhhhhhh

in MFT and MVT system trace records, the address of the TCB for the task that was in control when the interrupt occurred.

In MVT-M65MP systems the OLA and OLB fields replace the OLD TCB field and contain the address of the TCB for the task in control of CPU A and CPU B respectively, at the time the interrupt occurred.

DSP Minimal Trace Record

```
                                                                          ⎧NUA hhhhhhhh  NUB hhhhhhhh⎫
⎧A⎫     ⎧RES PSW⎫
⎩B⎭ DSP ⎩NEW PSW⎭ hhhhhhhh hhhhhhhh  R15/R0 hhhhhhhh hhhhhhhh  R1 hhhhhhhh ⎩NEW TCB hhhhhhhh           ⎭
```

Figure 43. DSP Minimal Trace Record

```
⎧A⎫
⎩B⎭
```

appears in MVT-M65MP records; identifies the CPU associated with the event.

DSP

identifies the type of record.

NEW PSW hhhhhhhh hhhhhhhh

the PSW for the task about to be dispatched.

In a record obtained from a MVT-M65MP system this field will be labeled RES PSW.

R15/R0 hhhhhhhh hhhhhhhh

the contents of general purpose registers 15 and 0 as they will be when the task being dispatched is given control.

R1 hhhhhhhh

the contents of general purpose register 1 as it will be when the task being dispatched is given control.

NEW TCB hhhhhhhh

the address of the TCB for the task about to be dispatched.

In a record obtained from a MVT-M65MP system this field is replaced by the NUA and NUB fields containing the addresses of the tasks to be dispatched on CPU A and CPU B when processing resumes.

EXT Minimal Trace Record

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

Figure 44. EXT Minimal Trace Record

```
{A}
{B}
```

appears in MVT-M65MP records; identifies the CPU associated with the event.

EXT

identifies the type of trace record.

OLD PSW hhhhhhhh hhhhhhhh

the program status word that was current at the time the external interrupt occurred.

R15/R0 hhhhhhhh hhhhhhhh

the contents of general purpose registers 15 and 0 at the time the interrupt occurred.

R1 hhhhhhhh

the contents of general purpose register 1 at the time the interrupt occurred.

STMSK hhhhhhhh

appears in MVT-M65MP records only; displays the SHOULDER TAP MASK at the time the interrupt occurred.

```
        ********
TQE TCB{hhhhhhhh}
          N/A
```

****

indicates that an error occurred while gathering the information.

hhhhhhhh

is the address of the TCB of the task that requested this timer interrupt.

N/A

indicates the interrupt was other than a timer interrupt.

PGM Minimal Trace Record

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

Figure 45. PGM Minimal Trace Record

$\left\{\begin{matrix}A\\B\end{matrix}\right\}$

appears in MVT-M65MP system records; identifies the CPU associated with the event.

PGM

identifies the type of trace record.

OLD PSW hhhhhhhh hhhhhhhh

the program status word that was current at the time the program interrupt occurred.

R15/R0 hhhhhhhh hhhhhhhh

the contents of general purpose registers 15 and 0 at the time the interrupt occurred.

R1 hhhhhhhh

the contents of general purpose register 1 at the time the interrupt occurred.

OLD TCB hhhhhhhh

the address of the TCB for the task that was in control when the interrupt occurred.

In MVT-M65MP trace records this field is replaced by the OLA and OLB fields that contain, respectively, the address of the TCB for the tasks in control of CPU A and CPU B at the time the interrupt occurred.

SVC Minimal Trace Record

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

Figure 46. SVC Minimal Trace Record

{A}
{B}

appears in MVT-M65MP system records; identifies the CPU associated with the event.

SVC

identifies the type of trace record.

OLD PSW hhhhhhhh hhhhhhhh

the program status word that was current at the time the interrupt occurred. The SVC number, e.g., SVC 51, is represented by the last two hexadecimal digits in the first word.

R15/R0 hhhhhhhh hhhhhhhh

the contents of general purpose registers 15 and 0 at the time the interrupt occurred.

R1 hhhhhhhh

the contents of general purpose register 1 at the time the interrupt occurred.

OLD TCB hhhhhhhh

the address of the TCB for the task that issued the SVC.

In MVT-M65MP systems the OLA and OLB fields replace the OLD TCB field and contain the address of the TCB for the task in control of CPU A and CPU B respectively, at the time the interrupt occurred.

SSM Minimal Trace Record

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

Figure 47. SSM Minimal Trace Record

{A}
{B}
    indicates the CPU associated with the event.

SSM
    identifies the type of trace record.

IK c

    CPU affinity byte:
        A indicates CPU A executing disabled.
        B indicates CPU B executing disabled.
        0 Neither CPU executing disabled.

OPSW hhhhhhhh hhhhhhhh
    the program status word that was current at the time the interrupt occurred. Obtained from the CPU on which the interrupt occurred.

R15/R0 hhhhhhhh hhhhhhhh R1 hhhhhhhh
    The contents of general purpose registers 15, 0, and 1 from the CPU on which the interrupt occurred, at the time the interrupt occurred.

OLA hhhhhhhh OLB hhhhhhhh
    the addresses of the TCBs of the tasks in control in CPU A and CPU B respectively at the time the interrupt occurred.

GTF COMPREHENSIVE TRACE RECORDS

The following material describes the records produced when comprehensive tracing is specified at the invoking of GTF (MODE=EXT). The formats described appear in the output from IMDPRDMP service aid processing of the data recorded by GTF.

Comprehensive trace records are produced for IO, PCI/IO, SIO, DSP, EXT, PGM, SSM, and SVC events.

IO and PCI/IO Comprehensive Trace Record

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

                                (******** ******** ********)       (********)
 CSW hhhhhhhh hhhhhhhh     RQE{hhhhhhhh hhhhhhhh hhhhhhhh} RQE TCB{hhhhhhhh} SENS{hhhhhhhh}
                                (N/A                       )       (N/A     )     (N/A     )
```

Figure 48. IO and PCI/IO Comprehensive Trace Record

```
{A}
{B}
```

This field appears only in MVT-M65MP system I/O or PCI trace records and identifies the computer associated with the event.

```
{IO }
{PCI}
```

This field identifies the type of trace record -- input/output (IO) or program controlled interrupt (PCI).

cuu

This field displays the device address for the device associated with the interrupt in channel/unit form.

OLD PSW hhhhhhhh hhhhhhhh

This field displays the program status word that was current at the time the IO or PCI interrupt being traced, occurred.

```
    (cccccccc)
JOBN{********}
    (  N/A   )
```

This field has three possible entries, as follows:

cccccccc

is the one to eight character name of the job associated with the interrupt being traced.

****

asterisks indicate that a bad control block chain prevented the jobname from being obtained.

N/A

in PCI trace records N/A indicates that the interrupt was issued by the system and there is no associated jobname; in IO interrupt trace records N/A indicates either a system issued interrupt as for PCI or an interrupt issued without a valid UCB for the device issuing the interrupt.

```
    (cccccccc)
DDNM{********}
    (  N/A   )
```

This field has three possible entries, as follows:

cccccccc

is the name of the DD statement associated with the interrupt being traced.

****

asterisk indicate that a bad control block chain prevented the data definition name from being obtained.

N/A

N/A appears in the DDNM field for one of the following reasons:

- An interrupt was issued without a valid UCB for the device issuing the interrupt.
- The post bit in the UCB is 'off.'
- The data event block (DEB) pointer to the TCB is set to 0.
- The DCB is not opened.
- The DCB TIOT offset is outside the valid range.
- The TCB TIOT pointer is set to 0.
- The DDNAME in the TIOT is not recorded in EBCDIC characters.

OLTCB hhhhhhhh

In MFT and MVT system trace records this field displays the address of the TCB that was current at the time the IO or PCI interrupt being traced, occurred.

In MVT-M65MP system IO and PCI trace records the following fields replace the OLTCB field:

OLA hhhhhhhh
This field displays the address of the A computer TCB that was current when the IO or PCI interrupt occurred.

OLB hhhhhhhh
This field displays the address of the B computer TCB that was current when the IO or PCI interrupt occurred.

CSW hhhhhhhh hhhhhhhh
This field displays the channel status word from permanent storage location 64.

$$\text{RQE}\left\{\begin{array}{c}\text{hhhhhhhh hhhhhhhh hhhhhhhh}\\ \text{******** ******** ********}\\ \text{N/A}\end{array}\right\}$$

This field has three possible entries as follows:

hhhhhhhh hhhhhhhh hhhhhhhh
is the content of the first three words of the Request Queue Element associated with the IO or PCI interrupt.

******** ******** ********
asterisks indicate that a bad control block chain prevented the RQE information from being obtained.

N/A
indicates that the interrupt was issued without a valid UCB for the device issuing the interrupt.

$$\text{RQE TCB}\left\{\begin{array}{c}\text{hhhhhhhh}\\ \text{********}\\ \text{N/A}\end{array}\right\}$$

This field has three possible entries as follows:

hhhhhhhh
is the address of the TCB associated with the Request Queue Element

****
asterisks indicate that a bad control block chain prevented the TCB address from being obtained.

N/A
indicates that the interrupt was issued without a valid UCB for the device issuing the interrupt.

$$\text{SENS}\left\{\begin{array}{c}\text{hhhhhhhh}\\ \text{N/A}\end{array}\right\}$$

This field has two possible entries as follows:

hhhhhhhh
is the content of the four sense bytes in the UCB beginning at UCB + 22 which describe the IO or PCI interrupt being traced.

N/A
indicates that the interrupt was issued without a valid UCB for the device issuing the interrupt.

SIO Comprehensive Trace Record

```
                                                   (OLA hhhhhhhh  OLB hhhhhhhh)
(A) SIO  cuu  CC hh  CAW hhhhhhhh  JOBN(cccccccc) (OLTCB hhhhhhhh             )
(B)                                    (N/A     )

 CSW hhhhhhhh hhhhhhhh  RQE hhhhhhhh hhhhhhhh hhhhhhhh  RQE TCB hhhhhhhh
```

Figure 49. SIO Comprehensive Trace Record

{A
B}

appears in MVT-M65MP system trace records; identifies the computer associated with the event.

SIO

the type of trace record.

cuu

the device address in channel/unit form for the device associated with the record.

CC hh

hh - is the condition code set by the SIO event.

CAW hhhhhhhh

the channel address word associated with this event -- taken from permanent storage location 72.

JOBN {cccccccc
N/A}

cccccccc

is the one to eight character jobname of the job associated with this event.

N/A

indicates the SIO was issued by the system and there is no associated jobname.

OLTCB hhhhhhhh

in MFT/MVT systems the address of the TCB that was current when the SIO was issued.

in MVT-M65MP systems the OLA and OLB fields replace the OLTCB field.

OLA hhhhhhhh

is the A computer address of the TCB that was current when the SIO was issued.

OLB hhhhhhhh

is the B computer address of the TCB that was current when the SIO was issued.

CSW hhhhhhhh hhhhhhhh

the channel status word associated with this event -- taken from permanent storage location 64.

RQE hhhhhhhh hhhhhhhh hhhhhhhh

the first three words of the Request Queue Element associated with the SIO operation.

RQE TCB hhhhhhhh

the address of the TCB associated with the request queue element.

DSP Comprehensive Trace Record

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

Figure 50. DSP Comprehensive Trace Record

{A}
{B}

MVT-M65MP systems only. Identifies the computer associated with the event.

DSP

the type of trace record.

RES PSW hhhhhhhh hhhhhhhh

the PSW for the task about to be dispatched. If this task was interrupted at some previous point in time, then this was the current PSW at the interrupt.

JOBN{cccccccc}
    {N/A}

cccccccc

is the eight character name of the job associated with the task being dispatched.

N/A

indicates the task switch is for a system task; no jobname is available.

```
     (WAITTCB )
     (SVC-cccc)
MODN (SVC-RES )
     (**IRB***)
     (cccccccc)
     (Icccccc )
```

WAITTCB

the WAIT task is about to be given control.

SVC-cccc

indicates a type 3 or 4 SVC is about to get control; cccc is the last four characters in the module name.

SVC-RES

indicates a resident type SVC routine is about to be given control.

IRB*

an asynchronous routine is about to be dispatched and the module name is not available.

cccccccc

an asynchronous routine is about to be dispatched and the module name is not available.

cccccccc

in MVT systems the eight character module name from the CDE associated with the task to be dispatched; or, the name of an error exit routine from the SIRB associated with the task.

in MFT systems the eight character name from the LRB, LPRB, PRB or FRB associated with the task being dispatched; or an error exit routine name from the SIRB associated with the task.

Icccccc

indicates that error fetch is in the process of loading an error recovery module. The last seven characters of the module name are shown.

NUTCB hhhhhhhh

the address of the new TCB -- the TCB of the next-to-be-dispatched task.

in MVT-M65MP systems the following fields replace the NUTCB field:

NUA hhhhhhhh

the address of the TCB of the next-to-be-dispatched task in the A computer.

NUB hhhhhhhh

the address of the TCB of the next-to-be-dispatched task in the B computer.

PRTY hh

hh

the dispatching priority of the next-to-be-dispatched task.

EXT Comprehensive Trace Record

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

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

Figure 51. EXT Comprehensive Trace Record

{A
B}

This field appears only in MVT-M65MP system EXT trace records and identifies the computer associated with the event.

EXT

This field identifies the trace record as an EXT trace record.

OLD PSW hhhhhhhh hhhhhhhh

This field displays the program status word that was current at the time the external interrupt occurred.

JOBN {cccccccc
N/A}

This field has two possible entries as follows:

cccccccc

is the one to eight character name of the job associated with the event.

N/A

indicates that the interrupt was issued by the system and there is not associated job name.

MODN {WAITTCB
SVC-cccc
SVC-RES
IRB*
cccccccc
Icccccccc}

WAITTCB

The WAIT task was interrupted.

SVC-cccc

A type 3 or 4 SVC routine was interrupted; cccc is the last four characters of the routine name.

SVC-RES

a resident SVC routine was interrupted.

IRB*

the EXT interrupt occurred during execution of an asynchronous routine with an associated IRB.

cccccccc

in MVT systems the eight character name of the module that was interrupted - taken from the CDE associated with the task; or the name of an error routine - taken from the SIRB associated with the task.

in MFT systems the eight character name of the module that was interrupted - taken from either the LRB, LPRB, PRB, or FRB; or the name of an error routine - taken from the SIRB associated with the task.

Icccccccc

indicates that error fetch was in the process of loading an error recovery routine when the interrupt occurred. The last seven characters of the module name are shown.

OLTCB hhhhhhhh

In MFT/MVT systems the address of the TCB that was current when the interrupt occurred.

In MVT-M65MP systems the OLA and OLB fields replace the OLTCB field.

OLA hhhhhhhh

is the address of the TCB in the A computer that was current when the interrupt occurred.

OLB hhhhhhhh

is the address of the TCB in the B computer that was current when the interrupt occurred.

STMSK hhhhhhhh

In MVT-M65MP systems only - the 'shoulder tap' mask from location X'2BC' in the other computers prefix.

$$\text{TQEFLG/TCB}\left\{\begin{array}{l}\text{N/A}\\ \text{********}\\ \text{hhhhhhhh}\end{array}\right\}$$

hhhhhhhh

is the first word of the timer queue element (TQE). The first byte of the word is the TQEFLGS and the remaining three bytes the TQETCB, which is the address of the TCB for the task in which this timer element is being used.

****

asterisks indicate that a bad control block chain prevented the information from being obtained.

N/A

indicates that this EXT interrupt was not caused by the timer.

$$\text{EXIT}\left\{\begin{array}{l}\text{hhhhhhhh}\\ \text{N/A}\\ \text{********}\end{array}\right\}$$

hhhhhhhh

is the address of the exit routine - taken from the eighth word of the TQE.

N/A

indicates that this EXT interrupt was not caused by the timer.

****

asterisks indicate that a bad control block chain prevented the information from being obtained.

PGM Comprehensive Trace Records

```
                                                           (WAITTCB  )
                                                           |SVC-cccc | (OLA hhhhhhhh   OLB hhhhhhhh)
(A) PGM   ccc OLD PSW hhhhhhhh hhhhhhhh   JOBN(cccccccc) MODN|SVC-RES  | (OLTCB hhhhhhhh            )
(B)                                           (N/A     )     |**IRB*** |
                                                           |cccccccc |
                                                           (Icccccc  )

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

Figure 52. PGM Comprehensive Trace Record

{A}
{B}
MVT-M65MP systems only; identifies the computer associated with the interrupt.

PGM
the type of trace record.

ccc
the completion code for the program interrupt.

OLD PSW hhhhhhhh hhhhhhhh
the program status word that was current at the time the program interrupt occurred.

JOBN {cccccccc / N/A}

cccccccc
is the one to eight character jobname of the job associated with this event.

N/A
indicates a system task program checked and no jobname is available.

MODN {WAITTCB / SVC-cccc / SVC-RES / **IRB*** / cccccccc / Icccccc}

SVC-ccc
A type 3 or 4 SVC routine was interrupted; cccc is the last four characters of the routine name.

SVC-RES
a resident SVC routine was interrupted.

IRB*
the program check interrupt occurred in an asynchronous routine with an associated IRB.

cccccccc
in MVT systems the eight character name of the module that was interrupted - taken from the CDE associated with the task; or, the name of an error routine - taken from the SIRB associated with the task.

Iccccccc
indicates that error fetch was in the process of loading an error recovery routine when the interrupt occurred. The last seven characters of the module name are shown.

OLTCB hhhhhhhh
in MFT/MVT systems the address of the TCB that was current when the interrupt occurred.

In MVT-M65MP systems the OLA and OLB fields replace the OLTCB field.

OLA hhhhhhhh
is the A computer address of the TCB that was current when the interrupt occurred.

OLB hhhhhhhh
is the B computer address of the TCB that was current when the interrupt occurred.

R0 hhhhhhhh
to
R15 hhhhhhhh
the content of general purpose registers zero through fifteen at the time of the interrupt.

SSM Comprehensive Trace Record

```
                                                     ⎧WAITTCB ⎫
                                                     ⎪SVC-cccc⎪
⎧A⎫ SSM OLD PSW hhhhhhhh  JOBN⎧cccccccc⎫ MODN⎨SVC-RES ⎬ OLA hhhhhhhh  OLB hhhhhhhh  LKID C
⎩B⎭                           ⎩N/A     ⎭     ⎪**IRB***⎪
                                             ⎪cccccccc⎪
                                             ⎩Iccccccc⎭
```

Figure 53. SSM Comprehensive Trace Record

{A}
{B}
identifies the computer associated with the SSM interrupt.

SSM
identifies this trace record as an SSM trace record.

OLD PSW hhhhhhhh hhhhhhhh
the program status word that was current at the time the set system mask instruction was issued.

JOBN {cccccccc / N/A}

cccccccc
is the one to eight character name of the job associated with SSM interrupt.

N/A
indicates that the system originated the interrupt and there is no associated jobname.

MODN {WAITTCB / SVC-cccc / SVC-RES / **IRB*** / Iccccccc}

WAITTCB
the WAIT task was interrupted.

SVC-cccc
a type 3 or 4 SVC routine was interrupted; cccc is the last four characters of the routine name.

SVC-RES
a resident SVC routine was interrupted.

IRB*
the SSM interrupt occurred during execution of an asynchronous routine with an associated IRB.

cccccccc
the eight character name of the module that was interrupted - taken from the content directory element (CDE) for the task; or the name of an error routine - taken from the SIRB associated with the task.

Iccccccc
indicates that error fetch was in the process of loading an error recovery routine when the interrupt occurred. The last seven characters of the module name are shown.

OLA hhhhhhhh
is the A computer address of the TCB that was current when the interrupt occurred.

OLB hhhhhhhh
is the B computer address of the TCB that was current when the interrupt occurred.

LKID c
CPU affinity byte:

A indicates CPU A executing disabled.
B indicates CPU B executing disabled.
0 Neither CPU executing disabled.

TIME AND LOST EVENT RECORDS

GTF produces two types of time records and a lost event record as follows:

TIME ddddd.dddddd

appears on the last line of every event record if TIME=YES was specified in the GTF start command, and designates in decimal the number of seconds and microseconds since the last midnight.

***DATE: DAY ddd YEAR dddd TIME dd.dd.dd

This timestamp record appears at the beginning of the printout of each buffer filled by GTF and represents the time the first record was placed in the buffer.

DAY ddd
is the Julian date.

YEAR dddd
is the year.

TIME dd.dd.dd
is the time since midnight in a twenty-four hour format (hours.minutes.seconds).

*** LOST EVENTS: NUM dddddddddd TIME dd.dd.dd [GTF DISABLED]

The lost event record appears whenever GTF loses records, whether it is because the GTF buffers overflowed or because GTF was temporarily disabled by ABEND. The record is not produced if GTF terminates when the buffers are full.

NUM dddddddddd
is the number of records that were lost; one to ten decimal digits.

TIME dd.dd.dd
is the time GTF resumed recording; 24-hour format starting at midnight.

GTF DISABLED
appears only if the events were lost because GTF was temporarily disabled, e.g., ABEND temporarily disables GTF in order to format GTF output for an ABEND dump.

HEXADECIMAL FORMAT RECORD

```
(HEXFORMAT)
(USER     )
(SYSTEM   )  AID hh  FID hh  EID hh    hhhhhhhh hhhhhhhh hhhhhhhh ∫ hhhhhhhh
(SUBSYS   )
```

Figure 54. Hexadecimal Format Record

Under some circumstances IMDPRDMP formats and prints GTF records in hexadecimal notation. The conditions under which GTF records are formatted and printed in hexadecimal format by IMDPRDMP are presented in the discussion of the hexformat record that follows:

HEXFORMAT
This label identifies a record dumped in hex format at the request of the user on a GTRACE macro. This request was made by not specifying a format appendage, that is FID=00.

USR
This label identifies this record as dumped in hexformat because the user requested a format appendage on the GTRACE macro that could not be found. This format appendage was identified by FID=hh, and therefore its name is IMDUSR hh.

SYSTEM
This label identifies a record that was dumped in hex format because either it is a GTF error record or the format appendage for it has been scratched by the user. If relative bytes 0, 1 or 8, 9 contain X'EEEE', then this is an error record produced by GTF. This error record was produced as a result of an unrecoverable error in a GTF data gathering routine. When the error was encountered message IHL118I was written on the master system console indentifying the error and the action taken. This message is not issued if the error occurred while building a comprehensive SVC trace record. Except for comprehensive SVC records, this was the last record of its type produced during the run of GTF that produced it. If the X'EEEE' were not in the record, then it was dumped in hexformat because the IMDPRDMP format appendage that formats this type of record was not found by IMDPRDMP.

SUBSYS
This label identifies this record as dumped in hexformat because the subsystem format appendage requested by the subsystem on a GTRACE was not found by IMDPRDMP. The request was made via FID=hh, and therefore, it's name is IMDUSRhh.

AID hh
This field contains the AID of this record, and should always be X'FF'. The AID is the application identifier, and GTF's is always X'FF'.

FID hh
This field contains the FID, or format identifier. It is appended to 'IMDSYS' or 'IMDUSR' to obtain the name of the format appendage that was to have formatted this record.

EID hhhh
This field contains the EID, or event identifier, for this record. The EID uniquely identifies the event that produced this record.

hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
up to 64 words (256 bytes) of record in the GTF internal format. The internal format of GTF records is available in the Service Aids PLM.

GTF SVC COMPREHENSIVE TRACE RECORDS

There are four groups of GTF SVC Comprehensive Trace records.

Group 1 -- Those with Basic Fields
Group 2 -- Those with Basic Fields plus a DDNAME Field
Group 3 -- Those with Basic Fields plus a Parameter List Field
Group 4 -- Those with Basic Fields plus Variable Field(s)

The following sub-index lists the SVCs in sequence, identifies the group to which they belong, and gives the page where register contents and other variable fields are noted.

SVC #	Group	Page
0	4	190
1	3	182
2	1	170
3	1	171
4	3	182
5	3	183
6	4	190
7	4	190
8	4	190
9	4	191
10	3	171
11	1	171
12	1	171
13	4	191
14	4	191
15	4	191
16	4	191
17	4	192
18	3	183
19	3	183
20	3	183
21	4	192
22	3	183
23	3	183
24	2	181
25	4	192
26	4	193
27	4	194
28	4	195
29	4	195
30	4	195
31	2	181
32	4	195
33	4	195
34	1	171
35	3	183
36	1	171
37	3	184
38	1	171
39	3	184
40	3	184
41	4	196
42	4	196
43	1	171
44	4	197
45	3	185
46	1	172
47	3	185
48	3	185
49	1	172
50	null	
51	4	197
52	1	172
53	2	181
54	4	198
55	2	181
56	3	186
57	2	181
58	2	181
59	1	172
60	3	186
61	1	172
62	4	198
63	3	187
64	3	187
65	4	198
66	4	199
67	4	199
68	1	172
69	2	181
70	3	187
71	4	199
72	1	173
73	3	187
74	3	187
75	4	199
76	1	173
77	3	187
78	4	200
79	1	173
80	3	188
81	4	200
82	4	200
83	1	173
84	1	173
85	1	173
86	4	201
87	3	188
88	4	202
89	4	202
90	3	188
91	1	173
92	1	173
93	1	174
94	1	175
95	1	177
96	3	188
97	1	178
98	4	202
99	3	189
100	1	178
101	1	179
102	3	189
103	1	179
104	1	179
105	1	180
109	1	180
116	1	180
117	1	180

SVC Comprehensive Trace Records Group 1 -- Basic Fields

```
{A}                                                                {OLA hhhhhhhh  OLB hhhhhhhh}
{B} SVC ddd  OLD PSW hhhhhhhh hhhhhhhh  JOBN cccccccc  MODN cccccccc {OLTCB hhhhhhhh           }

 R15/R0 hhhhhhhh hhhhhhhh  R1 hhhhhhhh
```

Figure 55. Basic SVC Comprehensive Trace Record

```
{A}
{B}
```

this field appears only in MVT-M65MP records and identifies the CPU associated with the event.

SVC nnn
the decimal number of the SVC

OLD PSW hhhhhhhh hhhhhhhh
the program status word that was current at the time the SVC interrupt occurred. When SVC processing is completed, operation is resumed under control of this PSW.

```
     {********}
JOBN {cccccccc}
     {  N/A   }
```

asterisks indicate an error occurred while attempting to retrieve the jobname, e.g., an incorrect TIOT address in the TCB could result in asterisks being placed in this field.

cccccccc is the eight character jobname of the job issuing the SVC.

N/A indicates that the SVC was issued by the system and there is no associated jobname.

```
     {**IRB***}
     {SVC-RES }
MODN {SVC-nnnn}
     {*ccccccc}
     {cccccccc}
     {  N/A   }
     {********}
```

IRB* indicates the SVC was issued by an asychronously executed routine with an associated IRB.

SVC-RES indicates the SVC was issued by a resident SVC with an associated SVRB.

SVC-nnnn indicates the SVC was issued by a transient SVC module with an associated SVRB. nnnn denotes the last four characters of the module name.

*ccccccc indicates that error fetch is in the process of loading an error recovery module. ccccccc is the last seven characters of the module name.

cccccccc is, in MVT systems, the eight character name of the module issuing the SVC -- taken from the CDE associated with the task; or the name of an error routine -- taken from the SIRB associated with the task.

In MFT systems the module name is taken from the LRB, LPRB, PRB, or FRB and the error routine name is taken from the SIRB associated with the task.

N/A indicates the RB CDE pointer was zero.

******** indicates that an error occurred while attempting to retrieve the module name.

OLTCB hhhhhhhh
the address of the TCB that was current when the SVC was issued.

In MVT-M65MP systems the OLA and OLB fields replace the OLTCB field and indicate the addresses of the TCBs that were current in CPU A and CPU B when the SVC was issued.

R15/R0 hhhhhhhh hhhhhhhh R1 hhhhhhhh
the contents of registers 15, 0, and 1 when the SVC was issued.

SVC Comprehensive Trace Records Group 1 -- Basic Fields

SVC 2 (POST)

R15 contains no applicable information.
R0 contains the completion code to be placed in the ECB.
R1 contains the address of the ECB to be posted.

SVC 3 (EXIT)
registers contain no applicable information.

SVC 10 (REGMAIN)
R15 contains no applicable information.
R0 contains the number of the subpool requested in the high order byte, and the number of bytes requested in the low order three bytes.
R1 contains any negative value if the request is for a GETMAIN; contains the address of the storage to be freed if the request is for a FREEMAIN; contains zero value if the request is for a FREEMAIN for an entire subpool.

SVC 11 (TIME)
R15 contains no applicable information.
R0 contains no applicable information.
R1 contains flag bits in the low order byte that designate how the time is to be returned in Register 0.

If the low order byte is:

x'00'
register 0 is to contain a 32 bit unsigned binary number representing the number of timer units that have elapsed. (A timer unit is 26.04 micro-seconds).

x'01'
register 0 is to contain elapsed time in hundredths of a second.

x'02'
register 0 is to contain packed decimal digits representing elapsed time in hours, minutes, seconds, tenths of a second, and hundredths of a second (HHMMSSth).

SVC 12 (SYNCH)
R15 contains the address of the entry point for the processing program that is to be given control.
R0 contains no applicable information.
R1 contains no applicable information.

SVC 34 (MGCR)

R15 contains no applicable information.

R0 and R1 contents are as follows:
R1, if positive, contains a pointer to the command buffer of the command to be processed. R0 is not used in this case.

If R1 is negative and R0 is zero, then R1 contains a pointer to the CSCB that is to be either added to the chain or deleted from the chain.

If R1 is negative and R0 is positive, then R1 contains a pointer to the CIB that is to be added to or deleted from the chain. R0 contains a pointer to the beginning of the chain.

If R1 is negative and R0 is negative, then R0 contains a pointer to the CIB in which the CIB count is to be set and R1 contains the value to which the CIB count is to be set.

SVC 36 (WTL)
R15 contains no applicable information.
R0 contains no applicable information.
R1 if positive, contains a pointer to the user record that is to be written to the system log dataset.

If negative, contains a pointer to the LCA indicating either initialization, (both data sets have to be opened), or data set switching is required.

SVC 38 (TTROUTER)
Registers 15, 0, and 1 do not contain any applicable information.

SVC 43 (CIRB)
R15 contains no applicable information.
R0 contains the entry point address of the user's asynchronous exit routine.
R1 contains option bit flags in the high order halfword and the size of the work area requested (in double words) in the low order halfword.

Flag settings are:

flag byte 1	
1.......	DIRB
0.......	CIRB
.1000...	bits 1-4 always set as shown
.....1..	problem program key
.....0..	supervisor key
......1.	problem program state
......0.	supervisor state
.......1	save area for registers requested
.......0	no save area requested

flag byte 2	
xxxx..xx	reserved
....1...	do not return IQEs at exit
.....1..	return IQEs at exit

SVC 46 (TTIMER)

R15 contains no applicable information.
R0 contains no applicable information.
R1 the low order three bytes carry code determining how TTIMER should work, as follows:

x'00'
the time remaining in the current tasks time interval is to be returned in register 0; the interval timer is not to be canceled.

x'01'
the current task's time interval is to be canceled.

x'02'
the time interval of a related task is to be canceled.

SVC 49 (TTOPEN)

Registers 15, 0, and 1 do not contain any applicable information.

SVC 52 (Restart/SMB Reader)

Registers 15 and 0 have no applicable information.

R1 contents are as follows:

If SVC 52 is issued by the Initiator for the purpose of reading SMBs (containing JCL) for an automatic step or checkpoint restart, register 1 points to a job queue DCB, SMB buffer, and general work space.

If SVC 52 is issued from module IEFRSTRT to initiate a check point restart, register 1 contains a pointer to a parameter list.

SVC 59 (OLTEP)

R15 contains no applicable information.
R0 contains a pointer to a three word parameter list, which, in turn contains pointers as follows:

Word 1 -- pointer to UCB
Word 2 -- pointer to DEB
Word 3 -- pointer to IECIOLTS (I/O interrupt handler).

R1 contains a call code used to locate the particular OLTEP function requested. The value will be greater than x'00' and equal or less than x'94'.

SVC 61 (TSAV)

Registers 15 and 0 have no applicable information.

R1 contains zeroes if the routine is being entered from the Overlay Supervisor.

R1 contains the address of the DCB used to fetch the module (set to a negative value) if the routine is being entered from the Contents Supervisor.

SVC 68 (SYNADAF/SYNADRLS)

Entry from SYNADAF:

R15 contains a flag byte in the high-order position and three bytes of user data or an address of an entry point to the SYNAD routine.

The flag byte contains codes as follows:

00	EXCP request
01	BPAM request
02	BSAM request
03	QSAM request
04	BDAM request
05	BISAM request
06	QISAM request
07	BTAM request
08	QTAM request
09	GAM request

R0 contains, in the three low order bytes, the address of the DECB (BSAM, BPAM, BDAM) or the address of the IOB (BISAM, QISAM, QSAM).

Additionally, when a QSAM request is made, the high-order byte contains the offset of the first CCW in the IOB.

R1 contains a flag byte and the address of the DCB in the high-order byte and the three low-order bytes respectively.

The flag byte bit settings are:

00000000	BISAM and QISAM
1.......	error caused by input
.1......	error caused by output
..1.....	error caused by BSP, CNTRL, or POINT
...1....	record had been successfully read
....1...	INVALID request
.....1..	PT conversion - invalid character
......1.	BDAM only - hardware error
.......1	BDAM only - no space for record

Entry from SYNADRLS:

Registers 0 and 1 have no applicable information.

R15 contains x'FF' in the high-order byte, indicating the SVC routine is being entered from the SYNADRLS macro instruction and three bytes of user data.

SVC 72 (CHATR)

Registers 15 and 0 have no applicable information.

R1 contains the address of a parameter list with the following structure:

Offset	
0	address of parameter list+8
4	address of DCB
8	module name for XCTL
16	code for OPEN/CLOSE (1 byte); address of UCM entry (3 bytes)
20	address of UCM
24	address of return

SVC 76 (IFBSTAT)

R15 contains no applicable information.

The content and applicability of Registers 0 and 1 vary with the presence or absence of RDE (Reliability Data Extractor) routines in the control program.

If RDE is present:
R0 contains a positive 0 or 8.
R1 has no applicable information.

A positive 0 in R0 indicates that EOD recording is requested; a positive 8 indicates that IPL recording is requested.

If RDE is not present:
R0 contains a negative number representing the length in bytes of a record to be placed in the SYS1.LOGREC data set.
R1 contains the address of the record to be written.

SVC 79 (STATUS)

R15 has no applicable information.

R0 contains the START/STOP code; 07 for START, 06 for STOP

R1 contains, in its three low order bytes, the address of the subtask TCB which is to have its START/STOP count adjusted.

SVC 83 (SMFWTM)

Registers 15 and 0 contain no applicable information.

R1 contains a pointer as follows:
If positive a pointer to the record that is to be written to the SMF data set.

if negative a pointer to the SMCA indicating either initialization or processing for a SWITCH command to switch SMF data sets.

SVC 84 (Restart Address Routine)

SVC 84 is issued by the GPS Graphic I/O Control Routine to have the buffer restart address stored in the UCB associated with the display unit for which the routine builds a channel program.

R15 contains no applicable information.

R0 contains the buffer restart address to be stored in the UCB in the high order two bytes. The low order two bytes point to the UCB.

R1 contains a zero

SVC 85 (SWAP)

Registers 15, 0, and 1 do not contain any applicable information.

SVC 91 (VOLSTAT)

R15 contains no applicable information.
R0 when negative, contains the address of the UCB. Note: If device type is disk go to SVC 91 load 2.
R0 when positive, contains the address of the DCB.
R1 contents are as follows:
if zero, the SVC was issued by CLOSE
if X'32', the SVC was issued by DDR
if X'33' the SVC was issued by EOD
if X'63', the SVC was issued by EOV
if any other than the above, the SVC was issued by UNALLOCATION

SVC 92 (TCBEXCP)

R15 contains no applicable information

R0 contains the address of the TCB for the issuers task.

R1 contains the address of the IOB.

SVC 93 (TGET/TPUT)

Entry from TGET

R15 contains no applicable information

R0 the two high-order bytes are reserved. The two low-order bytes contain the buffer size in bytes.

R1 contains a flag byte and an address as follows:
the high order byte is a flag byte with these bit settings.

1.......	Denotes "TGET" specified
0.......	Denotes "TPUT" specified
.1......	Reserved.
..1.....	Reserved for TPUT
...1....	Denotes "NOWAIT" specified means that control should be returned to the program that issued the TGET whether or not an input line is available from the terminal if no input line is obtained, a return code of 4 will be found in register 15.
...0....	Denotes "WAIT" specified means that control will not be returned to the program that issued the TGET until an input line has been put into the program's buffer if an input line is not available from the terminal, the issuing program is put into a wait state until a line does become available and is placed in the program's buffer
....1...	Reserved for TPUT
.....1..	Reserved for TPUT
......10	Reserved for TPUT
......01	Denotes "ASIS" specified means that normal or minimal editing will be performed.
......00	Denotes "EDIT" specified means that in addition to the normal ("ASIS") editing, further editing will be performed.

the low-order three bytes contain the address of the buffer that is to receive the input line.

Entry from TPUT

R15 contains no applicable information.

R0 the two high-order bytes contain the Terminal Job Identifier number; the two low-order bytes contain the size of the input buffer in bytes.

R1 contains a flag byte and an address as follows:
the high-order byte is a flag byte with these bit settings:

1.......	Denotes "TGET" specified
0.......	Denotes "TPUT" specified
.1......	Reserved
..1.....	Denotes "LOWP" specified means that the terminal will not receive any inter-terminal messages if TSBITOFF is on even if a key-zero task is sending the messages may only be specified on a TPUT with TJID.
..0.....	Denotes "HIGHP" specified means that the terminal will receive inter-terminal messages even if TSBITOFF is on if a key-zero task is sending the messages may only be specified on a TPUT with TJID.
...1....	Denotes "NOWAIT" specified means that control should be returned to the program that issued the TPUT whether or not system output buffers are available for the output line if no buffers are available, a return code of 4 will be found in register 15.
...0....	Denotes "WAIT" specified means that control will not be returned to the program that issued the TPUT until the output line has been placed in a system output buffer if no buffers are

available, the issuing program will be put into a wait state until buffers do become available and the output line is placed in them.

....1... Denotes "HOLD" specified means that the program that issued the TPUT cannot continue its processing until this output line has been either written to the terminal or deleted.

....0... Denotes "NOHOLD" specified means that control should be returned to the program that issued the TPUT as soon as the output line has been placed on the output queue.

.....1.. Denotes "BREAKIN" specified means that output has precedence over input; that is, if the user at the terminal is transmitting, he is interrupted, and this output line is sent any data that was received before the interruption is kept and displayed at the terminal following this output line.

.....0.. Denotes "NOBREAK" specified means that input has precedence over output; that is, the output message will be placed on the output queue to be printed at some future time when the terminal user is not entering a line.

......10 Denotes "CONTROL" specified means that this line is composed of terminal control characters and will not print or move the carriage on the terminal.

......01 Denotes "ASIS" specified; means that normal or minimal editing will be performed.

......00 Denotes "EDIT" specified; means that in addition to the normal ("ASIS") editing, further editing will be performed.

the low-order three bytes contain the address of the buffer that is to hold the line of output.

<u>SVC 94 (TERMCTL)</u>

<u>Entry from TCLEARQ</u>:

R15 contains no applicable information.

R0 Contents:

<u>Bytes</u>	
0	01 -- Entry code
1-3	0 -- Reserved

R1 Contents:

<u>Bytes</u>	
0	80 -- "INPUT" specified
	00 -- "OUTPUT" specified
1-3	0 -- Reserved

<u>Entry from STBREAK</u>:

R15 contains no applicable information.

R0 Contents:

<u>Bytes</u>	
0	04 -- Entry code
1-3	0 -- Reserved

R1 Contents:

<u>Bytes</u>	
0	80 -- "YES" specified
	00 -- "NO" specified
1-3	0 -- Reserved

<u>Entry from STCOM</u>:

R15 contains no applicable information.

R0 Contents:

<u>Bytes</u>	
0	05 -- Entry code
1-3	0 -- Reserved

R1 Contents:

<u>Bytes</u>	
0	80 -- YES specified
	00 -- NO specified
1-3	0 -- Reserved

<u>Entry from STTIMEOU</u>:

R15 contains no applicable information.

R0 Contents:

Bytes	
0	06 -- Entry code
1-3	0 -- Reserved

R1 Contents:

Bytes	
0	80 -- "YES" specified 00 -- "NO" specified
1-3	0 -- Reserved

Entry from STCC:

R15 contains no applicable information.

R0 Contents:

Bytes	
0	07 -- Entry code
1-3	0 -- Reserved

R1 Contents:

Bytes	
0	Flag byte as follows: 1....... first operand specified .1...... ATTN specified ..1..... LD specified ...1.... CD specified 00000000 no operands specified, retain previously-used characters.
1	0 -- Reserved
2	hh -- line delete control character. The hexadecimal representation of any EBCDIC character on the terminal keyboard except the new line (NL) and carriage return (CR) control characters. c -- the character representation of any EBCDIC character on the terminal keyboard.
3	hh -- character delete control character. The hexadecimal representation of any EBCDIC character on the terminal keyboard except the new line (NL) and carriage return (CR) characters. c -- the character representation of any EBCDIC character on the terminal keyboard.

Entry from STATTN:

R15 contains no applicable information.

R0 Contents:

Bytes	
0	08 -- Entry code
1	00 -- Reserved
2	hh -- Lines byte. The number of consecutive lines of output that can be directed to the terminal before the keyboard will unlock. 00 -- Output line counting is not used.
3	hh -- Tens byte. The tens of seconds that can elapse before the keyboard will unlock. 00 -- Locked keyboard timing is not used.

R1 Contents:

Bytes	
0	Flag byte as follows: 1....... LINES specified .1...... TENS specified ..1..... input address specified 00000000 no operands specified, results in a NOP instruction.
1-3	hhhhhh -- Character string address. 000000 -- no character string was specified.

Entry from STAUTOLN:

R15 contains no applicable informtion.

R0 Contents

Bytes	
0	09 -- Entry code
1-3	hhhhhh -- the address of a fullword containing the number to be assigned to the first line of terminal input.

R1 Contents:

Bytes	
0	00 -- Reserved
1-3	hhhhhh -- the address of a fullword containing the increment value used in assigning line numbers.

Entry from STSIZE:

R15 contains no applicable information.

R0 Contents:

Bytes	
0	0A -- Entry code
1,2	0000 -- Reserved.
3	hh -- lines byte. The number of lines (depth) that can appear on the screen.

R1 Contents:

Bytes	
0-2	000000 -- Reserved
3	hh -- size byte. The logical line size (width) in characters of the terminal.

Entry from GTSIZE, STAUTOCP, SPAUTOPT, RTAUTOPT

R15 contains no applicable information.

R0 Contents:

Bytes	
0	Entry codes as follows: 0B -- GTSIZE 0C -- STAUTOCP 0D -- SPAUTOPT 0E -- RTAUTOPT
1-3	000000 -- Reserved

R1 Contents:
No applicable information, will be zeroed.

Entry from STCLEAR:

R15 contains no applicable information.

R0 Contents:

Bytes	
0	10 -- Entry code
1-3	000000 -- Reserved

R1 Contents:

Bytes	
0	00 -- Reserved.
1-3	hhhhhh -- erasure character string address.

Entry from TCABEND

R15 contains no applicable information.

R0 Contents:

Bytes	
0	00 -- Entry code
1-3	0 -- Reserved

R1 Contents:
No applicable information will be zeroed.

Entry from TSABEND

R15 contains no applicable information.

R0 Contents:

Bytes	
0	0F -- Entry code
1-3	0 -- Reserved

R1 Contents:
No applicable information will be zeroed.

SVC 95 (TSIP)

R15 contains no applicable information.

R0 Contents:

Bytes	
0,1	zero or Terminal Job Identifier (TJID) or not applicable.
2	00 -- Reserved
3	Entry code as follows:

Entry Code	Calling Routine
00	Problem Program (TMP)
01	Timer Second - Level Interruption Handler
02	TGET/TPUT
03	Region Control Task
04	Dequeue, TIOC (Attention, TSINPUT, TSOUTPUT), Timer SLIH, WTOR
05	Region Control Task
06	Enqueue
07	Dequeue
08	TSO Dispatcher
09	TSO Dispatcher
0A	TSO Dispatcher
0B	TSO Dispatcher
0C	Region Control Task (Quiesce)

0D	Region Control Task (Quiesce)
0E	Time Sharing Control Task (Swap)
0F	Time Sharing Control Task (Swap)
10	Time Sharing Control Task (Swap)
11	Time Sharing Control Task (Swap)
12	Region Control Task (Restore)
13	Region Control Task (Restore)
14-18	Reserved

R1 Contents:

Bytes
0,1,2,3 variable as follows:

Entry Code	Content
00	Address of 8-character command name sign-bit: 0-ended 1-beginning
01	not applicable
02	Sign-bit: 0-Input 1-Output Bytes 3&4: Number of free buffers
03-05	not applicable
06	Estimated must complete time
07-0C	not applicable
0D	Number of FBQEs
0E	Byte 0: Swap Units Byte 1: Swap device code (0,4,8,c) Bytes 2&3: Swap size in 2K blocks
0F-13	not applicable

SVC 97 (TEST(TSO))

Entered from:

Any module of the tested program, when used as a breakpoint handler. If used as a breakpoint handler the TCBTCP bit is '1' in the current TCB and registers 15, 0, and 1 contain no applicable information.

Any module of the TSO Test Command Processor when used as a subroutine of TSO TEST. In this case the current TCBTCP bit is '0' and registers are as follows:

R15 contains no applicable information.

R0 Contents:

Bytes	
0	Entry code as follows: 40 -- Set TCBTCP bit to '1' 20 -- Set TCBTCP bit to '0' 10 -- Alter TCBTRN field 08 -- Alter second word of RBOPSW field 04 -- Alter specific register in SVC 97's SVRB register save area 04 -- Alter all registers in SVC 97's SVRB register save area 02 -- Alter floating-point register in TCB save area 01 -- Set RB wait count to 0 (zero).
1-3	Address of target TCB, PRB, or IRB

R1 Contents:

Register 1 contents are variable as follows:

	Bytes	
Entry code 40	0123	not applicable
entry code 20	0123	not applicable
entry code	0	not applicable
	1,2,3	TCBTRN value
entry code 08	0	instruction length, completion code program mask
08	1,2,3	address of value for second word of RBOPSW field.
entry code	0	register number
07	1,2,3	address of new value
entry code	0	x'FF'
04	1,2,3	address of 64-byte value
entry code	0	floating-point register number
02	1,2,3	address of new value for register
entry code 01	0,1, 2,3	not applicable

SVC 100

SVC 100 is used by the SUBMIT, OUTPUT, OPERATOR, and CANCEL/STATUS processors.

Contact your FE programming representative for information concerning the content of General Purpose Registers 15, 0, and 1 upon entry to SVC 100.

SVC 101 (QTIP)

SVC 101 is used only by the TSO sub-system and the MCP and provides an interface between them for inter-region communication and data movement.

R15 Contents:

Bytes	
0	0 -- zeroed. by entry code in R0
1-3	hhhhhh -- variable by entry code in R0 as follows: 00 -- not applicable 03 -- entry address of QTIP0030 within IEDAYAA 04-0D -- not applicable 0E -- (with savearea address in R1) not applicable. (Without savearea address in R1) entry address of QTIP0140 within IEDAYOO 0F-11 -- not applicable 12-16 -- entry address of IKJGGQT1, branch entry to QTIP SVC 17 -- address of TSB being logged off 18 -- (same as 12-16) 19-1A -- not applicable 1C -- entry address of QTIP0280 within IEDAYII 1D -- not applicable.

R0 Content:

Bytes	
0	0 -- zeroed.
1-3	hh -- entry codes as follows 00 -- invokes IEDAYAA 03 -- invokes IEDAYAA 04 -- invokes IEDAYHH 05-09 -- invokes IEDAYII; 0A -- invokes IEDAYLL; 0B-11 -- invokes IEDAYOO 12-14 -- invokes IEDAYGP 15-16 -- invokes IEDAYAA; 17 -- invokes IKJGG088 18 -- invokes IEDAYOO; 19-1A -- IEDAYZZ invoked 1C -- invokes IEDAYII 1D -- IEDAYGP invoked;

R1 Content:

Bytes	
0	0 -- zeroed.
1-3	hhhhhh -- variable by entry code in R0 as follows: 00 -- address of savearea within AVT 03 -- not applicable 04-0D -- address of savearea within AVT 0E -- (without entry address in R15; address of savearea in AVT) (with entry address in R15; not applicable) 0F-11 -- address of savearea within AVT 12-16 -- not applicable 17 -- zeroed; indicates no savearea is being passed 18 -- not applicable 19-1A -- address of savearea within AVT 1C -- not applicable 1D -- address of savearea within TIOCRPT

SVC 103 (XLATE)

R15 contains no applicable information.

R0 contains the length of the field to be translated.

R1 Contents:

Bytes		
0	hh	action byte as follows: 80-translate from EBCIDIC to ASCII 00-translate from ASCII to EBCDIC
1-3	hhhhhh	address of field to be translated

SVC 104 (TCAM)

R15 contains no applicable information

R0 indicates the subroutine to be executed as follows:

Bytes		
0-3	00000001	IGC0010D entry point routine
	00000002	GTFIELDA decode routine
	00000003	STTNME operator command addressing routine
	00000004	IEDQCA02 scan routine

R1 contains the address of the operator control work area

SVC 105 (IMGLIB)

R15 contains no applicable information

R0 contains no applicable information

R1 indicates actions to be taken as follows:

Bytes		
0-3	00000000	construct a DCB and DEB for SYS1.IMGLIB
	hhhhhhhh	delete DCB at this address and also the DEB pointed to by the DCB.

SVC 109

Type 3 and type 4 SVC routing routine.

R15 contains an index value, converted to 3 digit EBCDIC number and appended to name IGC00. This routine is then called.

R0/R1 contain no applicable information for SVC 109, contents are to be used by called routine IGX00.

SVC 116

Type 1 SVC routing routine.

R15 contains an index value, used in binary form to index into a table to call other SVC routines.

R0/R1 contain no applicable information for SVC 116, contents are to be used by called routines.

SVC 117

Type 2 SVC routing routine.

R15 contains an index value, used in binary form to index into a table to call other SVC routines.

R0/R1 contain no applicable information for SVC 117, contents are to be used by called routines.

SVC Comprehensive Trace Records Group 2 - Basic Fields Plus DDNAME Field

Group 2 SVC comprehensive trace records add a DDNAME field to the fields composing the basic record. The format is:

$$\text{DDNAME}\left\{\begin{matrix}\text{********}\\ \text{cccccccc}\\ \text{N/A}\end{matrix}\right\}$$

****
: asterisks indicate an error occurred while gathering the information.

cccccccc
: the name of the associated DD statement.

N/A
: indicates that the DD name could not be obtained for the following reasons:

The DCB was not opened
The DCB TIOT offset was outside the valid range
The DEB TCB pointer was set to 0
The TCB TIOT pointer was set to 0
The DD name in the TIOT was not in EBCDIC notation

Following are descriptions of register 15, 0, and 1 content for the Group 2 SVCs.

SVC 24 (DEVTYPE)

R15 contains no applicable information.

R0 contains the address of the output area or the two's compliment of the output area address.

R1 contains the address of the DD name, or the two's compliment of the DD name address.

When control returns from the DEVTYPE SVC routine, the output area will contain 8, 20, or 24 bytes of device data, depending on the value (+ or -) of R0 and R1, and the device type associated with the DDNAME as follows.

	Output Area Size (Bytes)		
	RPS-DA	DA	Non-DA
R0 and R1 positive	20	20	8
R0 negative and R1 positive	20	20	8
R0 and R1 negative	24	20	8

SVC 31 (FEOV)

R15 and R0 contain no applicable information

R1 contains the address of the DCB

SVC 53 (RELEX)

R15 contains no applicable information

R0 contains the address of a parameter list which contains either:

hhhhhhhh	relative block or TTR
MBBCCHHR	actual address

R1 contains the address of the DCB

SVC 55 (EOV)

R15 contains no applicable information

R0 contains the IOB address if the following are true:

DCBOFLAGS = ...1....
DCBMACRF = 0.......
and R0 is not equal to x'00001000'

R1 contains the DCB address

SVC 57 (FREEDBUF)

R15 contains no applicable information

R0 contains the address of the DECB

R1 contains the address of the DCB

SVC 58 (REQBUF/RELBUF)

R15 contains no applicable information

R0 contains the request count or release address

R1 contains the DCB address

SVC 69 (BSP)

R15 and R0 contain no applicable information

R1 contains the address of the DCB

SVC Comprehensive Trace Records; Group 3 - Basic Fields Plus Parameter List Field

Group 3 SVC comprehensive trace records add a parameter list field to the fields composing the basic record. The parameter list field displays all or a portion of the parameter list being passed to the SVC routine by the caller. The format is:

```
     (N/A                                  )
PLIST<hhhhhhhh hhhhhhhh hhhhhhhh ...       >
     (******** ******** ********           )
```

N/A
indicates that there is no applicable information

hhhhhhhh hhhhhhhh ...
parameter list display. Content and amount varies with the SVC being traced.

******** ********
indicates that an error occurred while gathering the information.

Following are descriptions of register 15, 0, and 1 content, and PLIST content for the Group 3 SVCs.

SVC 1 (WAIT)

R15 contains no applicable information

R0 contains the count of the events being waited on. If zero the wait is treated as a NOP.

R1 if positive, contains the address of the ECB being waited on. If negative, contains the address of a list of ECBs, in two's complement form.

PLIST may contain up to 40 bytes of information. It consists of a list of ECB addresses up to a maximum of 10.

SVC 4 (GETMAIN)

R15 and R0 contain no applicable information.

R1 contains the address of the parameter list passed when the SVC was called. (If R1 is zero there is no parameter list and the PLIST field will not be present.)

PLIST is ten bytes in length and breaks down as follows:

Bytes

0-3 hhhhhhhh

a. For a single area request - the length requested.

b. For a variable request - the address of a doubleword containing the minimum and maximum length requested as shown below:

Bytes	
0	zero
1,2,3	minimum length
4	zero
5,6,7	maximum length

c. For a list request - the address of a list of GETMAIN length requests (1 word per request) the last word containing x'80' in byte 0.

4 hh

Hierarchy identifier (optional)

5-7 hhhhhhhh

a. For a single area request - the address of a word GETMAIN will initialize as the beginning allocated core area.

b. For a variable area request - the address of a doubleword which GETMAIN will initialize with the address of the GETMAINed area and the actual length allocated.

c. For a list area request - the address of a list of words which GETMAIN will initialize with the address of allocated areas.

8 hh Flag byte as follows:

00	unconditional single area request
20	conditional single area request
C0	unconditional variable request
E0	conditional variable request

80 unconditional list request
A0 conditional list request

9 hh Subpool identification

SVC 5 (FREEMAIN)

R15 and R0 contains no applicable information.

R1 contains the address of the parameter list passed when the SVC was called. (If R1 is zero, no list passed, and PLIST will not appear.)

PLIST is 10 bytes in length and breaks down as follows:

Bytes

0-3 a. For a single area request the length to be freed.

b. For a list area request -- the address of a list of FREEMAIN length requests (1 word per request), the list word containing x'80' in byte 0.

4-7 a. For a single area request -- the address of an area to be freed.

b. For a list area request -- the address of a list of addresses of the areas to be freed.

8 hh Flag byte as follows:

00 unconditional single area request
20 conditional single area request
80 unconditional list area request
A0 conditional list area request

9 hh Subpool identification.

SVC 18 (BLDL/FIND - Type D)

R15 contains no applicable information.

R0 contains the address of the parameter list.

R1 contains the address of the DCB and indicates the macro instruction that issued the SVC call; if R1 is positive -- BLDL; if R1 is negative -- FIND.

PLIST
The BLDL parameter list is 12 bytes in length:

Bytes

0,1 the numbering entries
2,3 entry length
4-11 the hexadecimal representation of the member name for which the BLDL was issued.

The FIND parameter list is 8 bytes in length:

Bytes

0-7 the hexadecimal representation of the member name for which the FIND was issued.

SVC 19,20,22,23 (OPEN,CLOSE,OPENJ,TCLOSE)

R15 and R0 contain no applicable data.

R1 contains the address of the parameter list.

PLIST is up to 40 bytes in length and consists of a series of 4-byte entries (up to 10). Each entry breaks down as follows:

Bytes

0 hh Option byte as shown below:

Bits

1...	....	Last Entry indicator
.011	....	LEAVE
.001	....	REREAD
.100	....	REWIND
.010	....	IDLE
.000	....	DISP
....	0000	INPUT
....	1111	OUTPUT
....	0011	INOUT
....	0111	OUTIN
....	0100	UPDAT
....	0001	RDBACK

1-3 hhhhhh DCB address

SVC 35 (WTO/WTOR)

R15 contains no applicable information.

R0 contains console source ID.

R1 contains the address of the parameter list being passed to the SVC.

PLIST is 12 bytes in length for WTO and 20 bytes in length for WTOR.

The PLIST field for WTO breaks down as follows:

Bytes

0 00-- indicates WTO parameter list.

1	hh--	message length plus four.
2,3	hhhh--	MCS flag bytes; bit settings as follows:

Byte 2

1.......	Invalid entry
.1......	Message is to be queued to the console whose source ID is passed in Register 0.
..1.....	the WTO is an immediate command response.
....1...	the WTO macro instruction is a reply to a WTOR macro instruction.
.....1.	Message should be broadcast to all active consoles.
......1	Message queued for hard copy only.
.......1	Message queued unconditionally to the console whose source ID is passed in register 0.

Byte 3

1.......	time is not appended to the message.
.1111...	Invalid entry
.....1..	message is not queued for hard copy
......11	invalid entry

4-11	First eight bytes of message

The PLIST field for WTOR breaks down as follows:

Bytes	
0	hh--length of reply
1-3	hhhhhh--address of reply buffer
4-7	hhhhhhhh--address of reply ECB
8	00--zeroed
9	hh--message length plus four
10,11	hhhh--MCS flag bytes, see WTO PLIST
12-19	first eight bytes of message.

SVC 37 (SEGLD/SEGWT)

R15 contains no applicable information.

R0 if zero, entry was from SEGLD; non-zero indicates entry from SEGWT.

R1 contains the address of the parameter list.

PLIST is 12 bytes in length and breaks down as follows:

Bytes		
0-3	hhhhhhhh	branch instruction (to SVC 45)
4-7	hhhhhhhh	address of Referred-to Symbol
8	hh	"To" segment number
9-11	hhhhhh	Previous caller or 0

SVC 39 (LABEL)

R15 and R0 contain no applicable information.

R1 contains the address of the parameter list.

PLIST is 20 bytes in length and breaks down as follows:

Bytes	
0-2	c00004 -- REWIND option c00006 -- UNLOAD option
3	hh relative UCB in TIOT to use for mounting purposes.
4-7	hhhhhhhh address of 8 byte DDNAME for DD card that allocates devices for mounting tapes.
8-11	hhhhhhhh--address of volume label set.
12,13	hhhh-- length of one volume label.
14	hh-- number of labels in volume label set
15	hh-- command byte of control CCW
16-19	hhhhhhhh-- address of the first 10 bytes of volume header label.

SVC 40 (Extract)

R15 and R0 contain no applicable information.

R1 contains the address of the parameter list.

PLIST is 12 bytes in length and breaks down as follows:

Bytes		
0	--	Reserved
1-3	hhhhhh	address of list area in which the extracted information will be stored.
4	00	Reserved

5-7	0^0000	EXTRACT will obtain information from the current TCB and/or its related control blocks.
	hhhhhh	address of TCB from which EXTRACT is to get requested information.
8	hh	flags byte; indicates the fields to be extracted as follows:

Bits	
1.......	address of the general register save area
.1......	address of floating point register save area
..1.....	reserved
...1....	address of end-of-task exit routine
....1...	limit priority & dispatching priority
.....1..	task completion code
......1.	address of TIOT
.......1	address of the command scheduler communication list in the CSCB

9	hh	TSO only flags byte; indicates the TSO fields to be extracted as follows:

Bits	
1.......	address of time-sharing flags in TCB
.1......	address of protected storage control block
..1.....	terminal job identifier for task
...xxxxx	reserved

10,11	0000	reserved

SVC 45 (OVLYBRCH)

R15 contains the address of the Entry Table entry which caused the SVC to be issued.

R0 and R1 contain no applicable information.

PLIST is 12 bytes in length and breaks down as follows:

0-3	hhhhhhhh	Branch (inst. to SVC 45)
4-7	hhhhhhhh	address of Referred-to-Symbol
8	hh	"To" segment number
9-11	hhhhhh	Previous caller or 0

SVC 47 (STIMER)

R15 contains no applicable information

R0 contents:

Bytes		
0	hh	STIMER option byte as follows: x'40' TOD option x'30' DINTVL option x'10' BINTVL option x'00' TUINTVL option
1-3	hhhhhh	exit address

R15 contains the address of the time value

PLIST is four or eight bytes in length depending on the option in force:

a. For the DINTVL and TOD options PLIST is eight bytes in length and represents the time value.

b. For the BINTVL and TUNINTVL options PLIST is 4 bytes in length and represents the time value.

SVC 48 (DEQ)

R15 and R0 contain no applicable information.

R1 contains the address of the parameter list.

PLIST is 16 bytes in length and breaks down as follows:

Bytes		
0	hh	if set to x'FF' indicates the last element in the parameter list. Otherwise no meaning.
1	hh	the length of the minor name whose address is in bytes 8, 9, 10 and 11 of this element.
	00	the length of the minor name is in the first byte of the minor name field whose address is in bytes 8, 9, 10, and 11 of this element (does not include length byte itself).
2	hh	DEQ parameters byte as follows:

Bit Settings	
0.......	Exclusive request
1.......	Shared request
.0......	MINOR name is known only to job step

	.1......	the scope of minor name is SYSTEM
	..1.....	Set must complete equal to SYSTEM
	...1....	Set must complete equal to STEP
	000	RET=NONE
	001	RET=HAVE
	010	RET=CHNGE
	011	RET=USE
	111	RET=TEST
	....1...	RELEASE
3	hh	return code field for codes returned to the issuer by DEQ
4-7	hhhhhhhh	address of major resource name (QNAME)
8-11	hhhhhhhh	address of minor resource name (RNAME)
12-15	hhhhhhhh	if the DEQ parameters byte bit 4 (RELEASE) is set on this word contains the UCB address; otherwise the content of this word is unpredictable.

SVC 56 (ENQ)

R15 and R0 contain no applicable information

R1 contains the address of the parameter list

PLIST is 16 bytes in length and breaks down as follows:

Bytes		
0	hh	if set to x'FF' indicate the last element in the parameter list. Otherwise no meaning.
1	hh	the length of the minor name whose address is in bytes 8, 9, 10, and 11 of this element.
	00	the length of the minor name is in the first byte of the minor name field whose address is in bytes 8, 9, 10, and 11 of this element (does not include length byte itself).
2	hh	ENQ parameters byte as follows:

Bit Settings

0.......	Exclusive request
1.......	shared request
.0......	MINOR name is known only to job step
.1......	the scope of minor name is SYSTEM
..1.....	Set must complete equal to SYSTEM
...1....	Set must complete equal to STEP
.....000	RET=NONE
.....001	RET=HAVE
.....010	RET=CHNGE
.....011	RET=USE
.....111	RET=TEST
....1...	RESERVE

3	hh	return code field for codes returned to the issuer by ENQ
4-7	hhhhhhhh	address of major resource name (QNAME)
8-11	hhhhhhhh	address of minor resource name (RNAME)
12-15	hhhhhhhh	if the ENQ parameters byte bit 4 (RESERVE) is set on, this word contains the UCB address; otherwise the content of this word is unpredictable.

SVC 60 (STAE/STAI)

R15 contains no applicable information
R0 contents:
00 -- Create
04 -- Cancel
08 -- Overlay

R1 contains the address of the parameter list. The high-order bit is set to one if the XCTL=YES parameter was coded.

PLIST is eight bytes in length and breaks down as follows:

Bytes		
0		flag byte as follows: x '80' for STAI processing x '20' for STAE processing
1-3	hhhhhh	If zero, the 'CAMCE:' operand is in effect; otherwise this is the address of the STAE/STAI exit routine.
4-7	hhhhhhhh	address of the exit routine parameter list; if zero no exit routine parameter list exists.

SVC 63 (CHKPT)

R15 and R0 contain no applicable info.

R1 contents:
a. the address of the parameter list
b. Zero if a CANCEL request

PLIST is eight bytes in length and breaks down as follows:

Bytes		
0	00	check ID address provided via the second parameter of CHKPT macro instruction
	80	No check ID address provided
1-3	hhhhhh	address of checkpoint DCB
4	00	check ID address not provided
	01 to 10	check ID length provided via third parameter of the CHKPT macro instruction
	FF	"S" specified as third parameter of CHKPT macro instruction; the system generated check ID is to be placed at the address specified in bytes 5-7
5-7	hhhhhh	address for storing system generated check ID or address of user provided check ID

SVC 64 (RDJFCB)

R15 and R0 contain no applicable information

R1 contains the address of the parameter list

PLIST is up to forty bytes in length and consists of a series of 4-byte entries containing the DCB address. The high-order byte has bit 0 set to one to indicate the last entry.

SVC 70 (GSERV)

R15 and R0 contain no applicable information.

R1 contents:

Bytes		
0	hh	is a mask indicating which bits in the Graphic Control Byte (GCB) should be reset.
1-3	hhhhhh	the address of a fullword field that identifies the DCB related to the GCB in which bits are to be reset.

PLIST is four bytes in length and displays the fullword pointed to by R1. Byte 0 is a unit index factor used to locate the UCB address in the DEB associated with the DCB. (The GCB to be reset is in the UCB).

SVC 73 (SPAR)

R15 and R0 contain no applicable information

R1 contains the address of the parameter list

PLIST is up to 40 bytes in length and consists of a series of 4-byte entries. The first entry breaks down as follows:

Bytes		
0	hh	the priority specified for the attention routine by the SPAR macro instruction.
1	hh	Reserved
2,3	hhhh	the number of words in the parameter list.

Each additional entry contains a GACB address as specified by the SPAR macro.

SVC 74 (DAR)

R15 and R0 contain no applicable information.

R1 contains the address of the parameter list.

PLIST is up to forty bytes in length, consisting of 4-byte entries. The first entry breaks down as follows:

Bytes		
0,1	hh	Reserved
2,3	hh	the number of words in the parameter list.

Each additional entry contains a GACB address specified by the DAR macro.

SVC 77 (ONLT)

R15 contains the address of the UCB of the line for the terminal being tested.

R0 contains the address of the first of five '9's in the test request buffer for ONLT (five '9's' indicate a request for an online test).

R1 contains the address of the parameter list.

PLIST is 14 bytes in length and breaks down as follows:

Bytes		
0-3	hhhhhhhh	address of the ECB and the prefix of the request buffer.
4-7	hhhhhhhh	address of the GETMAIN parameters and terminal test pattern table.
8-11	hhhhhhhh	address of special line control characters
12	hh	00 means test is valid 01 means test is invalid and not set up
13	hh	00 means no answer on dial line 01 means answer on dial line

SVC 80 (GJP/GFX)

(The SVC 80 Processing Routine serves as a communication link between GJP routines and the GFX Task, and between the GFX task and ABEND Hook routine.)

R15 contains no applicable information.

R0 contains the address of the parameter list.

R1 contains the address of the console control table.

PLIST is eight bytes in length and breaks down as follows:

Bytes		
0-3	cccc	indicates which routine passed to SVC 80 as follows: PLOG -- Log Off PBEG -- Begin Job Processor ABDH -- Abend Hook Routine IERR -- Internal Error Routine NPRO -- Initial Processor
4-7	hhhhhhhh	the 2250 unit address that indicates which graphic job processor is using the SVC 80 routine.

SVC 87 (DOM)

R15 contains no applicable information.

R0 the value (positive or negative) of R0 determines the content of R1.

R1 If R0 is not negative, R1 contains a message ID word (which is also displayed in the PLIST field).

If R0 is negative, R1 contains the address of a list of message ID words.

PLIST is up to 40 bytes in length, consisting of 4-byte entries. Each entry is a message ID word. The last entry is identified by the 0 bit in the high-order byte being set to 1.

SVC 90 (XQMNGR)

R15 and R0 contain no applicable information.

R1 contains the address of the QMPA.

PLIST is 36 bytes in length and contains the QMPA fields. The QMPA and its associated control blocks are described in the MVT Job Management PLM, Order No. GY28-6660.

SVC 96 (STAX)

R15 and R0 contain no applicable information.

R1 contains the address of the parameter list.

PLIST is 20 bytes in length and breaks down as follows:

Bytes		
0-3	hhhhhhhh	address of user program to get control at attention interrupt.
4,5	hhhh	size of input buffer (max 4095)
6,7	hhhh	size of output buffer (max 4095)
8-11	hhhhhhhh	address of output buffer
12-15	hhhhhhhh	address of input buffer
16	hh	STAX option flag byte as follows: Bits 1.......Reserved .0......replace=YES .1......replace=NO ..1.....defer=YES ...1....defer=NO 1111Reserved
17-19	hhhhhh	address of user parameters for user program.

SVC 99 (TSO Dynamic Allocation)

R15 and R0 contain no applicable information.

R1 contains the address of the parameter list.

PLIST is up to 40 bytes in length. Consult your FE programming representative for information concerning the data displayed in this field.

SVC 102 (TCAM)

R15 and R0 contain no applicable information.

R1 contains the address of the parameter list.

PLIST is up to 12 bytes in length depending on the function and breaks down as follows:

Bytes		
0	hh	Action code byte for SVC 102 as follows: 1... .... Flag issuing task not eligible for rollout .1.. .... Post rollout/ rollin ECB complete ..1. .... Post standard or TSO ECB complete ...1 .... Flag issuing task not eligible for swap .... 1... Move data across partition boundary .... .1.. Enqueue element on disabled ready queue and post MCP ECB complete .... ..1. Flag issuing task eligible for swap ....1 Flag issuing task eligible for rollout
1-3	hhhhhh	varies by action code as follows: Action Code 80,40,01 ECB address 20,02,10 08,04 Data Address
4	hh	varies by action code as follows: 20 x'80', last four bytes 80,40,01, 08,04,02, 10 x'00' reserved
5-7	hhhhhh	varies by action code as follows: Action Code 20,02,10 TSO job ID address 80,40,01, TCB address 08,04 Target address (for enqueuing an element the target address is the address of the disabled ready queue in the TCAM AVT).
8	hh	varies with action code as follows: Action Code 80,40,20,10,08, 04,02,01 x'80', last four bytes
9-11	hhhhhh	varies with action code as follows: Action Code 80,40,01 DEB address 08,04 Length address 10,20,02 TCB address

SVC Comprehensive Trace Records; Group 4 - Basic Fields Plus Variable Fields

GTF Group 4 SVC comprehensive trace records have a variety of fields -- differing from SVC to SVC -- added to the fields composing the basic SVC record (Group 1). Format and content of the additional fields for each SVC are discussed in the following material.

SVC 0 (EXCP)

Additional fields -- DDNAME, DCB, DEB.

Register 15, 0, and 1 content, and DDNAME DCB, and DEB format and content follow:

R15 and R0 contain no applicable information.

R1 contains the address of the IOB associated with this request.

DDNAME cccccccc
N/A

See explanation of DDNAME field under Group 2.

DCB hhhhhhhh

address of the DCB associated with this I/O request.

DEB hhhhhhhh

address of the DEB associated with this I/O request.

SVC 6 (LINK)

Additional fields -- PLIST, NAME

Register 15, 0, and 1 content, and PLIST and NAME format and content follow:

R15 contains the address of the parameter list.

R0 and R1 contain no applicable information.

PLIST hhhhhhhh hhhhhhhh
is eight bytes in length and breaks down as follows:

Bytes		
0	hh	flag byte as follows: 80 DE form of macro instruction 00 EP and EPLOC form of macro instruction
1-3	hhhhhh	If byte 0 is 80; the address of the directory entry list. If byte 0 is 00; the address of the entry point name.
4	hh	hierarchy ID as follows: 00 -- no hierarchy 01 -- hierarchy 0 02 -- hierarchy 1
5	hhhhhh	address of DCB or zero.

NAME cccccccc
is the entry point/directory entry (EP/DE) name of the module to be linked to or control transferred to.

SVC 7 (XCTL)

(Same as SVC 6)

SVC 8 (LOAD)

Additional field -- NAME

R15 contains no applicable information.

R0 Content:
If byte 0 contains x'00', bytes 1, 2, and 3 contain the address of the entry point name.

If byte 0 contains x'80', bytes 1, 2, and 3 contain the address of the directory entry list.

R1 Content:
In LCS systems, byte 0 contains the hierarchy ID as follows:

00 -- no hierarchy
01 -- hierarchy 0
02 -- hierarchy 1

In systems without LCS byte 0 contains no significant information.

Bytes 1, 2, and 3 contain the DCB address or zero if the default for DCB was specified.

NAME cccccccc
is the entry point/directory entry name of the module to be loaded.

SVC 9 (Delete)

Additional field -- NAME

R15 and R1 contain no applicable information.

R0 contains the address of the entry point name.

NAME cccccccc
is the entry point name of the module to be deleted.

SVC 13 (ABEND)

Additional field -- CMP CODE

R15 and R0 contain no applicable information.

R1 contains significant information only if SVC 13 was not called by the ABTERM routines. In this case R1 contains the following:

Bytes		
0	hh	Flag byte as follows: Bits 1... .... DUMP option .1.. .... STEP option ..xx xxxx reserved
1-3	hhhhhh	ABEND completion code

CMP CODE hhhhhhhh
is the ABEND completion code if SVC 13 was called by the ABTERM routines. It is the content of the TCBCMP field of the current TCB at the time the SVC interrupt occurred. If ABEND recursion has occurred this field will contain the recursive completion code.

SVC 14 (SPIE)

Additional field -- PICA

R15 and R0 contain no applicable information.

R1 contains the address of the program interrupt control area (PICA).

PICA hhhhhhhh hhhh
displays the program interrupt control area from the associated SPIE macro instruction.

SVC 15 (ERREXCP)

Additional fields -- DDNAME, RQE, RQE TCB, CUU hhhh

R15 and R0 contain no applicable information.

R1 contains the address of the Request Queue Element (RQE) which was assigned to this I/O request by IOS.

DDNAME cccccccc
is the name of the DD statement associated with this I/O request.

RQE hhhhhhhh hhhhhhhh hhhhhhhh
is the first 12 bytes of the RQE assigned to this request by IOS. The breakdown is:

Bytes		
0,1	hhhh	not applicable
2,3	hhhh	address of the UCB
4	hh	TCB ID for MFT
5,6,7	hhhhhh	address of IOB
8	hh	priority byte
9	hhhhhh	address of DEB

RQE TCB hhhhhhhh
is the address of the TCB associated with the I/O request.

CUU hhhh
device address in channel-unit form of the device associated with this I/O request.

SVC 16 (PURGE)

Additional fields -- DDNAME, DCB, PLIST

R15 and R0 contain no applicable information.

R1 address of the purge parameter list.

$$\text{DDNAME}\left\{\begin{array}{l}\text{N/A}\\ \text{cccccccc}\\ \text{********}\end{array}\right\}$$

cccccccc
is the name of the DD statement associated with the requests being purged.

DCB hhhhhhhh
is the address of the DCB associated with the purge request.

PLIST hhhhhhhh hhhhhhhh hhhhhhhh
displays the PURGE parameter list which breaks down as follows:

Bytes		
0	hh	option byte as follows:
	0...	Purge request elements in complete DEB chain starting with DEB specified in address field.

```
          1... .... Purge the requests
                    associated with the
                    DEB specified in
                    address field.
          .1.. .... Post the purge
                    requests with x'48'.
          ..0. .... Allow the active
                    request to quiesce.
          ..1. .... Halt the I/O
                    operations.
          ...0 .... Purge all requests.
          ...1 .... Purge only related
                    requests.
          .... .0.. Purge AEQ, RB and IOS
                    logical channel
                    queue.
          .... .1.. Purge AEQ and IOS
                    logical channel
                    queue.
          .... ..0. Purge by DEB
          .... ..1. Purge by TCB

    1-3   hhhhhh    address of DEB.
    4     hh        completion code
    5-7   hhhhhh    address of TCB
    8     hh        quiesce indicator:
                    01 if one or more
                    requests are
                    quiescing.
    9-13  hhhhhh    address of IOB.
```

SVC 17 (RESTORE)

Additional fields -- DDNAME, DCB, DEB

R15 and R0 contain no applicable information.

R1 contains the address of a pointer to the chain of IOBs to be restarted.

$$\text{DDNAME}\begin{Bmatrix}\text{N/A}\\ \text{cccccccc}\\ \text{********}\end{Bmatrix}$$

cccccccc
is the name of the DD statement associated with this IOB.

DCB hhhhhhhh
is the address of the DCB associated with the IOB.

DEB hhhhhhhh
is the address of the DEB associated with the IOB.

SVC 21 (STOW)

Additional fields -- DDNAME, PLIST

R15 contains no applicable information.

R0 contains the address of the parameter list.

R1 contains the address of the associated DCB.

The values, positive or negative, of R0 and R1, indicate the directory action STOW is to take as follows:

R0	R1	Action
+	+	ADD
+	-	REPLACE
-	+	DELETE
-	-	CHANGE

```
       N/A
DDNAME cccccccc
       ********
```

cccccccc is the name of the associated DD statement.

PLIST
hhhhhhhh ...(2 or 4 words)

is eight or 16 bytes in length, depending on the directory action being performed:

For ADD, REPLACE, or DELETE actions the PLIST field is eight bytes long and contains, the member name or alias of the PDS directory entry being acted upon.

For CHANGE the PLIST field is 16 bytes long, the first eight bytes containing the old member name or alias, and the second eight bytes contain the new member name or alias.

SVC 25 (TRKBAL)

Additional fields -- DDNAME, DCBFDAD, DCBTRBAL

R15 and R0 contain no applicable information.

R1 contains the address of the associated DCB. Note: If R1 is negative, the address is in complement form and the DCBFDAD and DCBTRBAL fields are meaningless.

$$\text{DDNAME}\begin{Bmatrix}\text{N/A}\\ \text{cccccccc}\\ \text{********}\end{Bmatrix}$$

is the name of the associated DD statement.

DCBFDAD hhhhhhhh hhhhhhhh
is the full direct access address (MBBCCHHR) from the DCB pointed to by R1.

DCBTRBAL hhhh
is the track balance -- the number of bytes remaining on the current track after a write. The field is negative if no bytes remain.

SVC 26 (CATALOG/INDEX/LOCATE)

Additional fields -- PLIST, DSN

R15 and R0 contain no applicable information.

R1 contains the address of the parameter list when CATALOG or INDEX issue the SVC call.

R1 contains the address of CAMLST as generated by the CAMLST macro instruction when LOCATE issues the SVC call.

DSN cccccccc...
is the data set name.

PLIST hhhhhhhh ... (4 words)

is the parameter list passed to the SVC routine by the calling macro instruction. Its content varies, depending on the macro instruction issuing the call.

Entry from CATALOG:

Bytes

0 hh option byte as follows:

Bits	
1...	Search is to start on specified CVOL
0...	Search is to start on SYS.RES
..1.	Catalog a data set
...1	Recatalog a data set
.... 1...	Uncatalog a data set

1 hh option bytes as follows:

Bits	
.1..	Build all missing index levels
.... 1...	Delete all unneeded index levels except the high level
.... ..1.	Indicate presence of DSCB TIR

2 00

3 00

4 00 Reserved

5-7 hhhhhh address of the area that contains the data set name

8 00 Reserved
9-11 hhhhhh the address of the CVOL ID, or zeroed.

12 00

13-15 hhhhhh address of the volume list

Entry from INDEX:

Bytes

0 hh option byte as follows:

Bits	
1...	Search is to start on specified CVOL
0...	Search is to start on SYS.RES

1 hh option byte as follows:

Bits	
.1..	Build an index
..1.	Build a generation index
...1	Build an alias
.... 1...	Connect CVOLs
.... .1..	Delete an index
....1	Delete an alias

2 hh option byte as follows:

Bits	
1...	Disconnect CVOLs
.1..	Indicate DELETE option
.... 1...	Indicate EMPTY option

3 hh size of generation data group

4 00

5-7 hhhhhh a. address of the index name.
b. address of an eight byte area that contains a high-level index name.

		c. address of an area that contains an alias to be deleted.
8	00	
9-11	hhhhhh	the address of the area that contains the CVOL ID, or zeroed.
12	00	
13-15	hhhhh	a. address of an eight-byte area that contains an alias for a high-level index. b. address of a ten-byte area that contains the 4-byte device code of the CVOL to be connected followed by its 6-byte volume serial number.

<u>Entry from LOCATE</u>:

<u>Bytes</u>		
0	hh	option byte as follows: <u>Bits</u> 1... .... Search is to start on specified CVOL 0... .... Search is to start on SYS.RES .... ..1. Read a block by TTR. ..00 0.0. LOCATE a name
1	hh	option byte as follows: <u>Bits</u> .000 0000 LOCATE a name
2	hh	option byte as follows: <u>Bits</u> 0... .... LOCATE a name
3	00	
4	00	
5-7	hhhhhh	address of the data set name or the relative track address (TTR) of the desired block in the catalog.
8	00	
9-11	hhhhhh	address of the CVOL ID or zeroes.
12	00	
13-15,	hhhhhh	address of a 265 byte workarea which must be on a doubleword boundary. If the issuer of LOCATE has a non-zero protect key, then the workarea must have a matching storage protect key.

<u>SVC 27 (OBTAIN)</u>

Additional fields -- PLIST, VOLSER, DSN/CCHHR

R15 and R0 contain no applicable information.

R1 contains the address of the parameter list.

PLIST hhhhhhhh ... (4 words) displays the OBTAIN parameter list which breaks down as follows:

<u>Bytes</u>		
0-3	hhhhhhhh	operation code as follows: C1000000 SEARCH for DSNAME C1800000 SEEK for track address
4-7	hhhhhhhh	address of data set name or address of track address of DSCB, CCHHR depending on operation code.
8-11	hhhhhhhh	address of the volume serial number
12-15	hhhhhhhh	address of 14-byte workarea.

$\text{VOLSER}\begin{Bmatrix}\text{N/A}\\ \text{cccccc}\end{Bmatrix}$

cccccc is the volume serial number of the associated volume.

N/A indicates that the volser pointer in the parameter list was zero.

$\text{DSN/CCHHR}\begin{Bmatrix}\text{nnnnn}\\ \text{cccccccccc ...}\end{Bmatrix}$

nnnnn is the track address in EBCDIC notation and is displayed when the operation code in Word 1 of the parameter list indicates SEEK.

cccccc ... is the data set name and is displayed when the operation code in word 1 of the parameter list indicates SEARCH. N/A if the name is unavailable.

SVC 28 (OPENEXT)

Additional fields -- content of R13

R15 contains no applicable information.

R0 contains zeroes, or the DCB address of the SYSCTLG to be processed.

R1 contains the UCB address of the volume whose SYSTCLG is to be opened, if R0 contains zeroes.

SVC 29 (SCRATCH)

Additional fields -- PLIST, DSN

R15 contains no applicable information.

R0 contains zeroes; or, the address of a UCB or a SUBUCB (for a 2321 device) for the device upon which unmounted volumes may be mounted.

PLIST hhhhhhhh ... (4 words)
displays the SCRATCH parameter list which breaks-down as follows:

Bytes		
0-3	hhhhhhhh	operation code as follows: 41004000 -- check purge date 41005000 -- override purge date
4-7	hhhhhhhh	address of data set name
8-11		not used
12-15		address of the volume list

DSN ccccccccc ...
is the data set name. N/A if the name is unavailable.

SVC 30 (RENAME)

Additional fields -- PLIST, OLD DSN, NEW DSN

R15 contains no applicable information.

R0 contains the address of the UCB for the device on which unmounted volumes should be mounted, or zero.

R1 contains the address of the parameter list.

PLIST hhhhhhhh ... (4 words)

displays the RENAME parameter list which breaks-down as follows:

Bytes		
0-3	x'41002000'	
5-7	hhhhhhhh	address of old data set name
8-11	hhhhhhhh	address of new
12-15	hhhhhhhh	address of the volume list

OLD DSN ccccc ...
is the fully qualified name of the data set to be renamed. N/A if the name is unavailable.

NEW DSN ccccc ...
is the new name for the data set being renamed. N/A if the name is unavailable.

SVC 32 (ALLOCATE)

Additional fields -- CUU, DSN

R15 contains no applicable information

R0 when positive, contains the address of the associated job file control block; when negative (not complemented--high-order bit is set on), contains the address of the associated model DSCB.

R1 contains the address of the UCB list.

CUU ccc
is the unit address from the UCB pointed to by R1.

DSN cccccccc ...
is the data set name from the DSN field of the JFCB or DSCB pointed to by R0. N/A if the DSN field was blank.

SVC 33 (IOHALT)

Additional fields -- CUU

R15 and R0 contain no applicable information

R1 contains the address of the UCB associated with the request to be halted.

CUU hhhh
is the device address associated with the device being halted.

SVC 41 (IDENTIFY)

Additional fields -- EPNAME

R15 contains no applicable information.

R0 contains the entry point name address

R1 contains the main storage address for the entry point name being added.

EPNAME cccccccc
is the entry point name being added.

SVC 42 (ATTACH)

Additional fields -- SUPRVLIST, PPLIST

R15 contains the address of the parameter list being passed to the SVC routine.

R0 contains no significant information.

R1 contains the address of the parameter list being passed to the called program, or zero (no parameter list being passed).

SUPRVLIST hhhhhhhh ... (36 bytes)
is the parameter list being passed to the SVC routine and breaksdown as follows:

Bytes		
0	hh	EP/DE flag byte: 00 -- EP or EPLOC specified 80 -- DE specified
1-3	hhhhhh	address of the EP name or directory entry (determined by byte 0).
4	hh	hierarchy flag (used if option chosen): 00 -- no hierarchy specified 01 -- hierarchy 0 02 -- hierarchy 1
5-7	hhhhhh	address of the DCB; or zero.
8	hh	Reserved.
9-11	hhhhhh	address of the ECB
12	hh	GSP flag byte: 00 -- bytes 13-15 contain subpool number 01 -- bytes 13-15 contain the address of a listing of subpool numbers.
13-15	hhhhhh	a subpool number or address of subpool list (determined by byte 12)
16	hh	SHSP flag byte: 00 -- bytes 17-19 contain a subpool number 01 -- bytes 17-19 contain the address of a list of subpool numbers.
17-19	hhhhhh	a subpool number or address of a subpool list (determined by byte 16)
20	hh	Roll-In/Roll-Out flag: 00 -- new task may not be rolled-out and cannot invoke roll-out. 01 -- new task may not be rolled-out but can invoke roll-out 02 -- new task may be rolled-out but cannot invoke roll-out 03 -- new task may be rolled-out and can invoke roll-out
21-23	hhhhhh	address of the end-of-task exit routine
24,25	hhhh	dispatching priority number
26	hh	limit priority number
27	hh	Key Flags byte as follows:

Bits	
x...	Reserved
.0..	Propagate the JSCB field from the originating task
.1..	If the originating task has a protect key of 0, move the specified JSCB address into the

attached TCB; otherwise, propagate the originating task's TCBJSCB field

..0. .... Subpools 251 and 252 and the job pack queue pointer of the originating task are not given to the attached task.

..1. .... Subpools 251 and 252 of the job pack queue pointer are given to the attached task.

...1 .... the attached task is to have a protect key of 0.

.... 0... Subpool zero is to be shared with other tasks.

.... 1... Subpool zero is not to be shared

.... .0.. A save area of 72 bytes is to be obtained for the task.

.... .1.. No save area is to be obtained.

.... ..0. Propagate the TCBJSTCB field from the originating task.

.... ..1. The TCBJSTCB of the new task is to point to the new task.

....0 The new task is to operate in problem program mode.

....1 The new task is to operate in supervisor mode.

28-35 hhhh the entry point name for EP; or blank for EPLOC or DE specification.

PPLIST hhhhhhhh hhhhhhhh hhhhhhhh ... (up to 40 bytes)
is the parameter list being passed to the called program and consists of a series of four-byte entries, each entry having it's high-order byte reserved, and an address in the low-order three bytes.

SVC 44 (CHAP)

Additional fields -- CHAP TCB

R15 contains no applicable information.

R0 contains a signed value to be added to the dispatching priority of the specified task. A negative value will be in two's-complement form.

R1 contains the address of an area containing the address of the TCB whose priority is to be changed; or zero. If zero, it indicates that the active task's priority is to be changed.

CHAP TCB hhhhhhhh
is the address of the TCB of the active task at the time the SVC interrupt occurred.

SVC 51 (SNAP)

Additional fields -- PLIST, MODN

R15 and R0 contain no applicable information.

R1 contains the address of the parameter list.

PLIST:
The PLIST field when SVC 51 is called by the SNAP macro instruction is 12 bytes in length and breaksdown as follows:

PLIST hhhhhhhh hhhhhhhh hhhhhhhh
displays three words of the parameter list passed to SVC 51 by SNAP.

Bytes		
0	hh	ID number to be printed in the identification heading of the dump.
1	00	
2	hh	option flag bytes as follows:

Bits	
0...	ABEND request
1...	SNAP request
.1..	TCB address given
..1.	Display all supervisor data
...1	Display trace table
.... 1...	Display nucleus
.... .1..	Snapshot list is given

.... ..1.	ID given
....1	Display QCBs

3	hh	option flag byte as follows:

Bits	
1...	Save area (see next flags)
.0..	Display entire save area
.1..	Display heading only
..1.	Display registers on entry to ABEND or SNAP
...1	Display link pack area
.... 1...	Display job pack area
.... .1..	Display PSW on entry to ABEND or SNAP
.... ..1.	Display all subpools less than subpool 128
....x	Reserved

4	00	
5-7	hhhhhh	address of DCB
8	00	
9-11	hhhhhh	address of the TCB specified in the SNAP macro instruction; or zero. If zero, the dump is for the current task.

Certain calls for SVC 51 may result in a 16 byte PLIST field being recorded. If there is a problem in this area please contact your FE programming representative for programming support.

$$\text{MODN}\begin{Bmatrix}\text{N/A}\\ \text{cccccccc}\end{Bmatrix}$$

cccccccc is the name of the module calling SVC 51.

N/A appears if no module name is available.

SVC 54 (DISABLE)

Additional fields -- DDNAME, DCB, DEB

R15 and R0 contain no applicable information.

R1 contains the address of the associated DCB

$$\text{DDNAME}\begin{Bmatrix}\text{N/A}\\ \text{cccccccc}\\ \text{********}\end{Bmatrix}$$

is the name of the DD statement associated with this request.

DCB hhhhhhhh
is the address of the associated DCB.

DEB hhhhhhhh
is the address of the associated DEB.

SVC 62 (DETACH)

Additional fields -- DETACH TCB

R15 and R0 contain no applicable information.

R1 contains the address of an area containing the address of the TCB to be detached.

Note: If R1 contains zero the DETACH TCB field is meaningless.

DETACH TCB hhhhhhhh
is the address of the TCB to be detached.

SVC 65 (QWAIT)

Additional fields -- R2, QCB

R15, R0 and R1 contain no applicable information.

R2 contains the address of the QCB for the element being waited on.

QCB hhhhhhhh hhhhhhhh hhhhhhhh
is the queue control block pointed to by R2, and breakdown as follows:

Bytes		
0	hh	queue status: 01 -- not on ready queue 02 -- not waiting 03 -- waiting
1-3	hhhhhh	address of first element on the queue.
4	.hh	priority of the queue when linked onto the ready queue.
5-7	hhhhhh	address of the next item on the ready queue.
8	hh	reserved.
9-11	hhhhhh	address of the STCB for the subtask to be activated.

SVC 66 (BTAM TEST)

Additional fields -- IOBERINF

R15 and R0 contain no applicable information.

R1 contains the address of the IOB pointed to when the SVC was issued.

IOBERINF hhhhhhhh ... (4 words)
is the error information field used by BTAM error recovery routines.

SVC 67 (QPOST)

Additional fields -- R2, QCB

R15 and R0 contain no applicable information.

R1 contains the address of the element being posted.

R2 contains the address of the QCB to which the element is being posted.

QCB hhhhhhhh hhhhhhhh hhhhhhhh
is the queue control block pointed to by R2 and breaksdown as follows:

Bytes		
0	hh	queue status: 01 -- not on Ready queue 02 -- not waiting 03 -- waiting
1-3	hhhhhh	address of first element on the queue.
4	hh	priority of the queue when linked onto the ready queue.
5-7 7	hhhhhh	address of the next item on the ready queue.
8	hh	reserved.
9-11	hhhhhh	address of the STCB for the subtask to be activated.

SVC 71 (ASGNBFR/RLSEBFR/BUFINQ)

Additional fields -- DDNAME, PLIST

R15 and R0 contain no applicable information

R1 contains the address of the parameter list.

DDNAME cccccccc
is the name of the DD statement associated with the DCB specified by the macro instruction.

PLIST hhhhhhhh hhhh ... (up to 12 bytes)
displays the parameter list pointed to by R1. The content varies according to the macro instruction calling the SVC.

Entry from ASGNBFR:

Bytes		
0	04	request byte, 04 indicates ASGNBFR
1-3	hhhhhh	the DCB address
4-7	hhhhhhhh	the address of a half-word field containing the number of bytes of buffer to be assigned.

Entry from RLSEBFR:

Bytes		
0	hh	request byte: 08 indicates RLSEBFR 0C indicates RLSEBFR ALL
1-3	hhhhhh	the DCB address
4-7	hhhhhhhh	the address of a half-word field containing the number of bytes of buffer to be released.

Entry from BUFINQ:

Bytes		
0	10	request byte, 10 indicates BUFINQ
1-3	hhhhhh	the DCB address
4-7	hhhhhh	address of the table of buffer addresses (must be on a fullword boundary)
8-11	hhhhhhhh	the number of bytes specified to be available for the table of buffer addresses

SVC 75 (Dequeue Routine)

Additional fields -- IQE

R15 contains no applicable information

R0 contains the address of the next IQE on the IRB active list for the attention routine when ATTNINQ has specified clear mode; otherwise, contains zero.

R1 Content:

Bytes		
0	hh	is a unit index to identify a particular 2260 display station; or 00 for a 2250 station.
1-3	hhhhhh	the GACB address

N/A
IQE hhhhhhhh hhhhhhhh hhhhhhhh

when ATTNINQ specifies clear mode this field displays the first 3 words of the IQE pointed to by R0:

Bytes		
0-3	hhhhhhhh	the address of the next IQE in the chain, or zero
4-7	hhhhhhhh	not meaningful
8-11	hhhhhhhh hhhhhhhh	the address of the IRB associated with the IQE.

N/A
will appear in this field whenever the ATTNINQ macro instruction did not specify clear mode.

SVC 78 (LSPACE)

Additional fields -- CUU

R15 and R1 contain no applicable information

R0 contains the address of the associated UCB

CUU hhhh is the unit address

SVC 81 (SETPRT)

Additional fields -- DDNAME, PLIST

R15 and R0 contain no applicable information

R1 contains the address of the parameter list.

DDNAME cccccccc
is the name of the DD statement associated with the data set being printed.

PLIST hhhhhhhh .. (four words)
is four words of the parameter list being passed to SVC 81 and breaks down as follows:

Bytes		
0-3	hhhhhhhh	address of the DCB
4-7	hhhhhhhh	EBCDIC character set image ID
8	hh	LOAD MODE indicator:
	.0..	no fold
	.1..	fold
	x.xx xxxx	reserved
9	hh	verification indicator:
	...1	verify
	...0	don't verify
	xxx. xxxx	reserved
10	hh	data check indicator:
	1...	block
	.1..	unblock
	00..	as DCB specifies
	.... 1...	unfold UCS 3211
	.... .1..	fold UCS 3211
	..xx ..xx	reserved
11-14	hhhhhhhh	EBCDIC FCB image ID
15	hh	FCB parameter options:
	1...	verify FCB
	1	align
	.xxx xxx.	reserved

SVC 82 (DISKANAL)

Entered from modules: IEHDANAL, IEHOGETA, IEHDCELL, IEHDLABL, IEHDREST, IEHDDUMP

Additional fields -- VOLSER, DA-ADDR, PLIST

R15 and R0 contain no applicable information

R1 contains the address of the parameter list.

VOLSER cccccc
is the volume serial number

DA-ADDR N/A
hhhhhhhh hhhhhhhh

displays a six or eight byte track address or N/A, dependent on the options in effect for the SVC routine. The breakdown is:

Option	DA-ADDR Content
analyze or format	six-byte track address
post UCB	eight-byte trac address

address of alternate track CCHH	N/A
unlabeled volume	eight-byte track address
new volume	N/A

PLIST hhhhhhhh ... (16 bytes maximum)
is either 8, 12, or 16 bytes of the parameter list pointed to by R1. The first four bytes always consist of a flag byte, defining the function to be performed, and a 3-byte UCB address. The fifth, ninth, and thirteenth bytes, when present, will contain a flag indicating the last element (4-bytes) in the list. The breakdown is as follows:

Bytes		
0	hh	function byte as follows: 8F -- new volume 1F -- address of alternate track CCHH 00 -- ANALYZE or FORMAT 08 -- POST UCB 88 -- unlabeled volume
1-3	hhhhhh	address of UCB
(function 8F)		
4	80	flag byte -- last element
5-7	hhhhhh	address of DCB
(function 1F)		
4	80	flag byte -- last element
5-7	hhhhhh	address of alternate track CCHH
(function 00)		
4-70	hhhhhhhh	address of alternate track CCHH
8	80	flag byte -- last element
9-11	hhhhhh	address of alternate track information
(function 08)		
4-7	hhhhhhhh	address of serial number
8	80	flag byte -- last element
9-11	hhhhhh	address of VTOC address of VTOC
(function 88)		
4-7	hhhhhhhh	address of serial number
8-11	hhhhhhhh	address of VTOC
12	80	flag byte -- last element
13-15	hhhhhh	address of DEB

SVC 86 (ATLAS)

Additional fields -- PLIST, CCHHR

R15 and R0 contain no applicable information

R1 contains the address of the parameter list

PLIST hhhhhhhh hhhhhhhh
is the parameter list passed to SVC 86 and breaks down as follows:

Bytes		
0	hh	flag byte as follows:
	1...	User's channel program can not be re-executed.
	.xxx xxxx	reserved
1-3	hhhhhh	address of IOB
4	hh	flag byte as follows:
	1...	IEHATLAS is the calling program
	.1..	a partial count (CCHH only) has been passed by the calling program
	..1.	a write special CCW is required for a track overflow record
	...1	a write special CCW is not required
	 xxxx	reserved
5-7	hhhhhh	address of count (CCHHR) or partial count (CCHH) field

CCHHR hhhhhhhhhh
is the five-byte track address of the complete (CCHHR) or partial count (CCHH) field passed by the calling program.
Note: If entry to SVC 86 is from the IEHATLAS program (byte 4, bit 0 in parameter list) this address points to the CCHH part of the count field.

SVC 88 (MOD 88)

Additional fields -- DEB, DSSTAT FLGS, DEVMOD

R15 and R0 contain no applicable information.

R1 contains the address of the DCB associated with the current task at the time the SVC was issued.

DEB hhhhhhhh
is the address of the data extent block (taken from DCB pointed to by R1)

DSSTAT hh
the data set status flags field (taken from the DEB)

DEVMOD hh
the device modifier field (taken from the DEB)

SVC 89 (EMSERV)

Additional fields -- PLIST, RESMCW

R15 and R0 contain no applicable information

R1 contains the address of the parameter list

PLIST hhhhhhhh
displays four bytes from the parameter list being passed to the SVC routine. The breakdown is:

Bytes		
0	hh	flag byte: C0 -- enter emulator mode A0 -- leave emulator mode
1-3	hhhhhh	address of control storage lead name

RESMCW hhhhhhhh hhhhhhhh
displays dight bytes of the RESMCW field from the RMS common area.

SVC 98 (TSO PROTECT)

Additional fields -- PLIST, DSN

R15 and R0 contain no applicable information

R1 contains the address of the parameter list

PLIST hhhhhhhh
displays the first four bytes of the parameter list as follows:

Byte		
0	01	entry code for the add function
	02	entry code for the replace function
	03	entry code for the delete function
	04	entry code for the list function
1-3	hhhhhh	varies by function as follows: 000000 -- add function 000000 -- replace function 000000 -- delete function hhhhhh -- 80 byte buffer address

DSN cccccccc ...
is the data set name

IMDPRDMP OUTPUT COMMENTS - GTF PROCESSING

The following comments may appear in the listing of GTF trace records.

I/O ERROR ON ddname - CONTINUE

Explanation: The EDIT function of IMDPRDMP is being used to process a GTF external trace data set. An I/O error was encountered while attempting to read the trace data set identified by ddname. Fewer than three consecutive I/O errors have occurred for this data set, so EDIT continues processing, ignoring the current block that caused the I/O error.

I/O ERROR ON ddname - EDIT PROCESSING TERMINATED

Explanation: The EDIT function of IMDPRDMP is being used to process a GTF external trace data set. Three consecutive I/O errors have been encountered while attempting to read the trace data set identified by ddname. EDIT processing terminates.

ERROR IN GTF BUFFER CHAIN

Explanation: The EDIT function of IMDPRDMP is being used to process an internal (dump) trace data set. While attempting to locate the GTF trace buffers, IMDPRDMP encountered one of the following errors:

- A buffer pointer was not on a word boundary.

- A buffer pointer addressed an area of main storage that could not be extracted from the dump for one of the following reasons:
  1. The pointer addressed an area higher than the highest address in the dump.
  2. IMDPRDMP encountered an I/O error while attempting to read the record containing the area addressed by the pointer.
  3. The block containing the addressed area was missing from the dump, perhaps because the program that produced the dump encountered an I/O error while attempting to write the block. EDIT processing is terminated.

ERROR IN GTF BUFFER - CONTINUING WITH NEXT BUFFER

Explanation: The EDIT function of IMDPRDMP is being used to process an internal (dump) trace data set. EDIT has encountered a GTF trace record with a length that does not lie within the acceptable range of 4 to 272 bytes. EDIT continues processing with the next GTF buffer.

GTF NOT ACTIVE AT TIME OF DUMP

Explanation: The Edit function of IMDPRDMP is being used to process an internal (dump) trace data set. EDIT has determined that GTF was not active at the time that the dump was taken. EDIT processing is terminated.

TRACE RECORD LL INVALID, DD ddname BLOCK NO xxxyyy - EDIT PROCESSING TERMINATED

Explanation: The EDIT function of IDMPRDMP is being used to process a GTF external trace data set. EDIT has encountered a GTF trace record with a length that does not lie within the acceptable range of 4 to 272 bytes. Ddname identifies the GTF external data set being processed; xxxyyy identifies the number of the block containing the faulty record. EDIT processing is terminated.

EDIT TERMINATED UPON USER'S REQUEST

Explanation: A user exit has requested EDIT termination by returning to EDIT with a return code of 24.

EXIT DELETED UPON USER'S REQUEST

Explanation: A user exit has requested that it no longer be invoked during the current EDIT execution. This is the result of a user exit routine return code of 16 or 20.

GTF OPTIONS IN EFFECT - option

Explanation: The input trace data set was created by GTF with trace options in effect as indicated by 'option'. The Service Aids publication describes the options available.

Appendix A: Debugging With an Operating System Dump

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 problém 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 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 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.

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.

Specialized Program Checks

In addition to the error program checks (1-15), other system events cause program checks which are normally transparent to the user. They could, however, if seen in a dump (except ABEND dumps where they do not appear, result in some confusion. One such event is the monitor call interrupt. On 360 CPU's, the monitor call appears as a 01 (operation) interrupt code in the program old PSW. To verify that a simulated monitor call occurred, check the address in the program old PSW. A monitor call occurred if:

1. The address (-4) points to an execution instruction ('44');
2. The execute is operating on an x'AF00' in low core;
3. A NOP (x'470') follows the execute.

370 CPU's support the real monitor call interrupt. The code in the program old PSW is a x'40', and the PSW address (-4) points directly at an x'AF' instruction.

On 360 CPU's, the x'AF' opcode is simulated as follows:

1. The first time an x'AF' instruction is encountered, an execute instruction is substituted for the x'AF'.

2. The execute is of an instruction in a low-core table (Class Mask Table).

3. If the monitor call should occur, the instruction in the Class Mask Table is an x'AF00'; if it should not occur, the instruction is a x'0700' (NOP).

4. Required class and ID information for the monitor call are contained in the x'470' NOP following the execute.

On 370 CPU's, the monitor call occurs under control of a mask in Control Register 8.

The Generalized Trace Facility (GTF) is a user of the monitor call interrupt. For more detailed information, refer to the Service Aids Logic PLM, GY28-6721.

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 C.

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 D.

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.

Appendix B: 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.

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	EXIT		
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	ERREXCP		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	TRKBAL		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	CVOL		
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	ALLOC		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	TTROUTER		
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	OVLYBRCH		
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				
51	33	IV	SNAP		Parameter list address
52	34	IV	RESTART		DCB address
53	35	III	RELEX	Key address	DCB address
54	36	II	DISABLE		
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	OLTEP		
60	3C	III	STAE	0 Create SCB 4 Cancel SCB 8 0	Parameter list address
61	3D	III	TTSAV		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	QWAIT		Parameter list address
66	42	IV	BTAMTEST		

(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	CHATR		Parameter list address
73	49	III	SPAR		Parameter list address
74	4A	III	DAR		Parameter list address
75	4B	III	DQUEUE		Parameter list address
76	4C	IV	IFBSTAT		
77	4D	IV	QTAMTEST		
78	4E	III	WSCAN		
79	4F	I	STATUS		
80	50	III	IKASVC		
81	51	IV	SETPRT		
82	52	IV	DASDR		
83	53	III	SMFWTM		Message address
84	54	I	GRAPHICS	UCB address and buffer restart address	
85	55	IV	DDRSWAP		
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	TCBEXCP		
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	TEST-TSO		
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
103	67		XLATE	Field length	Action byte and field address
104	68	IV	TOPCTL	Subroutine indicator	Address of operator control word area
105	69	III	IMAGLIB		Action indication
109	6D	IV		-- contents used by called routine --	
116	74	I	AT	-- contents used by called routines --	
117	75	II		-- contents used by called routines --	

(Part 5 of 5)

Appendix C: 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 9 on an ABEND/SNAP dump, or 10 through 1 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.

200 The error was detected when an I/O operation was requested and the storage protection keys of the IOB, ECB, and DCB were not the same as the key in the DEB. (checked for MVT only)

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 BLDL control program routine 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. In systems with MFT this address plus 4 is the location of the 8-byte name of the requested program that could not be loaded. In systems with MVT, registers 2 and 3 contain the name of the requested module.

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-form 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 FREEMAIN macro instructions.

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 D: 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
IEZ	System Interfaces
IFB	Environment recording routines
IFC	Environment recording and print routines
IFD	Online test executive program

Prefix	Component
IFF	Graphic programming support
IFG	Close, open, and related routines
IGC	Transient SVC routines
IGE	I/O error routines
IGF	Machine check handler program
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
IHL	Generalized Trace Facility
IHK	Remote job entry
IIN	7094 emulator program for the Model 85
IIO	7074 emulator program on the Models 155 and 165
IIP	7080 emulator program on the Model 165
IIQ	1401/1440/1460 emulator program on Models 135, 145, and 155
IIR	1440/7010 emulator program on Models 145 and 155
IIT	709/7090/7094/7094 II emulator program on the Model 165

Appendix E: 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
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 F: 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 G: UCB Sense Bytes

	BYTE 0								BYTE 1							
BIT DEVICE	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
2400	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	OVERRUN	WORD CNT ZERO	DATA CNVTT CHK	NOISE	00-NON-XST TU 01-NOT READY 10-RDY & NO RWD 11-RDY & RWDNG		7 TRK	AT LOAD POINT	WRT STATUS	FILE PROTECT	NOT CAPABLE
2311, 2841	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	OVERRUN	TRK CCND CHK	SEEK CHK	DATA CHK FLD.	TRK OVERRUN	END OF CYL	INVALID SEQ	NO REC FOUND	FILE PROT	MISSING ADR MRKR	OVERFLOW INL
2301,2302 2303,2314 2319,2820	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	OVERRUN		INVAL ADDR	DATA CHK IN COUNT	TRK OVERRUN	END OF CYL	INVAL SEQ	NO REC FOUND	FILE PROT	SERVICE OVERRUN	OVERFLOW INL
2250	CMD REJ	SHOULD NOT OCCUR	BUS OUT	SHOULD NOT OCCUR	DATA CHK	SHOULD NOT OCCUR	BUFFER RUNNING	SHOULD NOT OCCUR	LIGHT PEN DETECT	END ORDER SEQ	CHAR MODE					
2280	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	SHOULD NOT OCCUR	SHOULD NOT OCCUR	ILLGL SEG	READ COUNT CHK	FILM LOW	RECRDR FORCED GAP	SHOULD NOT OCCUR	SHOULD NOT OCCUR	2840 OUTPUT CHK	2840 INPUT CHK	GRAPHIC CHK
2282	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	SHOULD NOT OCCUR	SHOULD NOT OCCUR	ILLGL SEGN	READ COUNT CHK	FILM LOW	RECRDR FORCED GAP	FILM MOTION LIMIT	SHOULD NOT OCCUR	2840 OUTPUT CHK	2840 INPUT CHK	GRAPHIC CHK
1052, 2150	CMD REJ	INT REQ	BUS OUT	EQ CHK												
1285	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	OVERRUN	NON RCVY	KYBD CORR								
1287	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	OVERRUN	NON RCVY	KYBD CORR	TAPE MODE	LATE STKR SELECT	NO DOC FOUND	SHOULD NOT OCCUR	INVAL OP	SHOULD NOT OCCUR	SHOULD NOT OCCUR	SHOULD NOT OCCUR
1288	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	OVERRUN	NON RCVY	SHOULD NOT OCCUR	SHOULD NOT OCCUR	END OF PAGE	NO DOC FOUND	SHOULD NOT OCCUR	INVAL OP	SHOULD NOT OCCUR	SHOULD NOT OCCUR	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		UNUSUAL CMD									
3505, 3525	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	NOT USED	ABNORMAL FORMAT RESET	PERMANENT ERROR (BYPASS KEY	PERMANENT ERROR	AUTOMATIC RETRY	MOTION MALFUNCTION	RETRY AFTER INT REQ COMPLETE				
3211	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	BUFFER PARITY CHK	LOAD CHK	CH 9	CMD RETRY	PRINT CHK	PRINT QUALITY	LINE POS	FORMS CHK	CMD SUP	MECHANICAL MOTION	
1403, 1443	CMD REJ	INT REQ	BUS OUT	EQ CHK	TYPE BAR	TYPE BAR		CH 9								
1442,2596 2501,2520	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	OVERRUN ·										
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 REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	OVERRUN	LOST DATA	TIME OUT								
1419/1275 PCU	CMD REJ	INT REQ	BUS OUT	NOT USED	DATA CHK	OVERRUN	AUTO SELECT	NOT USED	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	CMD REJ	INT REQ	BUS OUT CHK	NOT USED	NOT USED	LATE STKR SELECT	AUTO SELECT	OP ATT								
3330	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	OVERRUN			PERM ERR	INVLD TRK FORMAT	END OF CYL	STATE VAR PRES	NO REC FOUND	FILE PROT	WRITE INHIBIT	OPERATION INL
3410/3411	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	OVERRUN	WORD CNT ZERO	DATA CNVTT CHK	NOISE	TU STATUS A	TU STATUS B		AT LOAD POINT	WRT STATUS	FILE PROTECT	NOT CAPABLE
2305	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	OVERRUN			PERM ERR	INVLD TRK FORMAT	END OF CYL		NO REC FOUND	FILE PROT		OPERATION INL
3420/3803	CMD REJ	INT REQ	BUS OUT	EQ CHK	DATA CHK	OVERRUN	WORD CNT ZERO	DATA CNVTT CHK	NOISE	TU STATUS A	TU STATUS B	7 TRK TU	LOAD POINT	WRT STATUS	FILE PROTECT	NOT CAPABLE

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	UNSAFE	▨	SERIALIZER CHK	TAG LINE CHK	ALU CHK	UNSEL STATUS	▨	▨
2301,2302 2303,2314 2319,2820	UNSAFE	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
3211	CARR FAILED TO MOVE	CARR SEQ CHK	CARR STOP CHK	PLATEN FAILED TO ADV	PLATEN FAILED TO RETRACT	FORMS JAM	RIBBON MOTION	TRAIN OVERLOAD
3330	▨	CORRECTABLE	▨	ENV DATA PRESENT	▨	▨	▨	▨
2305	BUF LOG FULL	CORRECTABLE	▨	▨	▨	▨	▨	▨
3505,3525	USED FOR DIAGNOSTIC PURPOSES ONLY							
3410/3411	TRACK IN ERROR							
3420/3803	TRACK IN ERROR							

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	COMPARE
2311,2841	READY	ON LINE	READ SAFETY	WRITE SAFETY	▨	END OF CYL	▨	SEEK INCMPL
2301,2302 2303,2314 2319,2820	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
3211	UCSB PARITY	PLB PARITY	FCB PARITY	COIL PROT CHK	HAMMER FIRE	FIELD ENG	USCAR SYNC CHK	SEP SYNC CHK
3330	RESTART COMMAND							
2305	RESTART COMMAND							
3505,3525	USED FOR DIAGNOSTIC PURPOSES ONLY							
3410/3411	VRC	MTE/ LRCR	SKEW	END DATA CHK/ CRCR	ENV CHK	1600 BPI SET IN TU	BKWD	▨
3420/3803	R/W VRC	MTE/ LRCR	SKEW	END DATA CHK/ CRC	VRC ENV CHK	1600 BPI SET IN TU	BKWD	C COMPARE

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 2319,2820	SEQ IND 0	SEQ IND 1	SEQ IND 2	SEQ IND 3	SEQ IND 4	SEQ IND 5	SEQ IND 6	SEQ IND 7
3330	PHYSICAL DRIVE IDENTIFICATION							
2305	▨	▨	▨	▨	▨	▨	▨	▨
3410/3411	TAPE UNIT POS CHK	TAPE UNIT REJ	EOT	▨	▨	DIAG TRK CHK	TAPE UNIT CHK	SPARE
3420/3803	ALU HARDWARE ERROR	REJ TAPE UNIT	TAPE INDICATE	WRITE TGR VRC	MICROPRGM DETECT ERROR	LWR ERROR	TAPE UNIT CHK	▨

BYTE 5

DEVICE								
2400	COMMAND IN PROGRESS WHEN OVERFLOW INCOMPLETE OCCURS OR ZERO							
2311,2841	COMMAND IN PROGRESS WHEN OVERFLOW INCOMPLETE OCCURS WRITE = X'05' OR READ = X'06' ZERO							
2301,2302 2303,2314 2319,2820	▨	▨	▨	▨	▨	▨	▨	▨
3330	CYLINDER ADDRESS							
2305	CYLINDER ADDRESS							
3410/3411	NEW SUBSYSTEM	NEW SUBSYSTEM	WRT TAPE MARK CHK	PE ID BURST	PARITY COMPARE	TACHOMETER CHK	FALSE END MARK	RPQ
3420/3803	NEW SUBSYSTEM	NEW SUBSYSTEM	WRT TAPE MARK CHK	PE ID BURST	START READ CHK	PARTIAL RECORD	EXCESSIVE POSTAMPLE OR TM	RPQ

BYTE 6

DEVICE								
3330*	REVERSE	CYL HIGH	DIFFER HIGH	HEAD ADDR				
2305*	CURRENT HEAD ADDR							
3410/3411	▨	SHORT GAP MODE	DUAL DENSITY	ALTERNATE DENSITY	TU MODEL			
3420/3803	7 TRK	WRT CURRENT FAILURE	DUAL DENSITY	NRZI DENSITY	TAPE UNIT MODEL DEFINED			

BYTE 7

DEVICE								
3330*	FORMAT OF REMAINING SENSE BYTES (8-23)				ENCODED ERROR MESSAGE			
2305*	ENCODED ERROR MESSAGE							
3410/3411	LAMP FAILURE CHK	TAPE BOTTOM LEFT COLUMN CHK	TAPE BOTTOM RIGHT COLUMN CHK	RESET KEY	DATA SECURITY ERASE CHK	▨	▨	▨
3420/3803	LAMP FAILURE	TAPE BOTTOM LEFT	TAPE BOTTOM RIGHT	RESET KEY	DATA SECURITY ERASE	ERASE HEAD	AIR BEARING PRESSURE	LOAD FAILURE

BYTE 8

BIT / DEVICE	0	1	2	3	4	5	6	7
3410/3411	▨	WRT FEED-THROUGH CHK	▨	END VELOCITY CHK	NO READ-BACK DATA	START VELOCITY CHK	▨	MARGINAL VELOCITY
3420/3803	IBG DROP WHILE WRT	FEED THRU	▨	EARLY BEGIN RDBK CHK	EARLY END RDBK CHK	SLOW BEGIN RDBK CHK	SLOW END RDBK CHK	VELOCITY RETRY

BYTE 9

0	1	2	3	4	5	6	7
▨	▨	▨	▨	▨	▨	▨	▨
SDR CNT	VELOCITY CHGE WHILE WRTNG	SDR CNTRS		▨			CU

BYTE 10

3420/3803	CMD STATUS REJ	▨	CNTRL UNIT REJ	NO BLK ON RECORD RD BK CHK	WTM NOT DETECT	TACHOMETER START FAIL	▨	VELOCITY

BYTE 11

B1 BUS PAR/LSR ADDR ERR	ROSI PAR ERR	XFR/LOW IC 1 ERR	INSTRUCTION DECODE	ALU 1 MICRO-PRGM DETECT HRDWR	D BUS PAR ALU 1	▨	BOC ALU 2

BYTE 12

3420/3803	B2 BUS PARITY LSR ADDR ERR	ROS 2 PARITY ERR	XFR/LOW IC 2 ERR	INSTR DECODE 4	ALU 2 MICRO-PRGM DETECT HRDWR	D BUS PARITY ALU 2	▨	BOC ALU 2

BYTE 13

CONTROL UNIT DENSITY	CONTROL UNIT UNIQUE ID HIGH

BYTE 14

3420/3803	CONTROL UNIT UNIQUE ID LOW

BYTE 15

TAPE UNIT UNIQUE ID

BYTE 16

3420/3803	TAPE UNIT UNIQUE ID

BYTE 17

TWO CHANNEL SW (MIS)	CONTROL UNIT DEVICE SWITCH FEATURES	EC LEVEL OF CONTROL UNIT

BYTE 18

3420/3803	POWER CHK/OVER-TEMPERATURE	▨	EC LEVEL OF TAPE UNIT

BYTE 19

PRIMED FOR DEVICE END							
TU 7	TU 6	TU 5	TU 4	TU 3	TU 2	TU 1	TU 0

BYTE 20

3420/3803	PRIMED FOR DEVICE END							
	TU F	TU E	TU D	TU C	TU B	TU A	TU 9	TU 8

BYTE 21

LOAD BUTTON DEPRESS	LEFT REEL TURNING	RIGHT REEL TURNING	TAPE PRESENT	REELS LOADED	LOAD REWIND	LOAD COMPLETE	LOAD CHECK

BYTE 22

3420/3803	FRU IDENTIFIERS FOR CONTROL UNIT

BYTE 23

FRU IDENTIFIERS FOR CONTROL UNIT

Appendix H: 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 allows the user to format and print IMSADMP output data sets, the SYS1.DUMP data set, the TSO dump data set and its associated swap data sets, and Generalized Trace Facility output data sets. IMDPRDMP can also be used to transfer a system dump from a SYS1.DUMP data set on a direct access device to another data set for later formatting and printing.

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.

IMCOSJQD — A problem program that reads, formats, and prints the contents of the system job queue data set (SYS1.SYSJOBQE). Either the entire job queue or information related to a specific job may be printed.

Because the program can be run under OS, it is not necessary to re-IPL the operating system as with IMCJQDMP.

IMBLIST — A problem program that produces formatted listings of object modules, load modules, module cross references, CSECT identification records (IDRs), and PTFs.

IMBMDMAP — A problem program 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.

IMAPTFLE A problem program that generates job control language (JCL) statements necessary to add a PTF to the Operating System in a later step, or applies PTFs to the Operating System by dynamically invoking the linkage editor.

IFCDIP00 A problem program that initializes the SYS1.LOGREC data set.

IFCEREP0 A problem program that edits, writes, and accumulates environment records on the SYS1.LOGREC data set.

Appendix J: 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 listings 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.

Appendix K: 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 56 illustrates control block relationships in the System/360 Operating System. Figure 57 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 and TSO)

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

IOB - Input/Output Block

-7	Address of next IOB (BSAM, QSAM, and BPAM)
+2	Sense bytes
+5	ECB address
+9	CSW
+17(11)	CCW list address
+21(15)	DCB address

RB - Request Block (PCP and MFT)

-8	Address of previous RB on load list
-4	Address of next RB on load list
+0	Module name
+13(D)	Entry point address
+16(10)	Resume PSW
+29(1D)	Address of previous RB

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
+16(10)	Completion code
+25(19)	MSS boundary box address
+37(25)	Address of most recent RB on load list
+113(71)	Address of first save area
+161(A1)	Address of STAE control block
+181(B5)	Address of the job step control block

TCB - Task Control Block (MFT) with Subtasking

+45(2D)	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
+181(B5)	Address of the job step control block

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 (except devices with extended sense)
+24(18)	Number of sense bytes (devices with extended sense)
+25(19)	Address of sense bytes (devices with extended sense)
+40(28)	Number of outstanding RESERVE requests (shared DASD only)

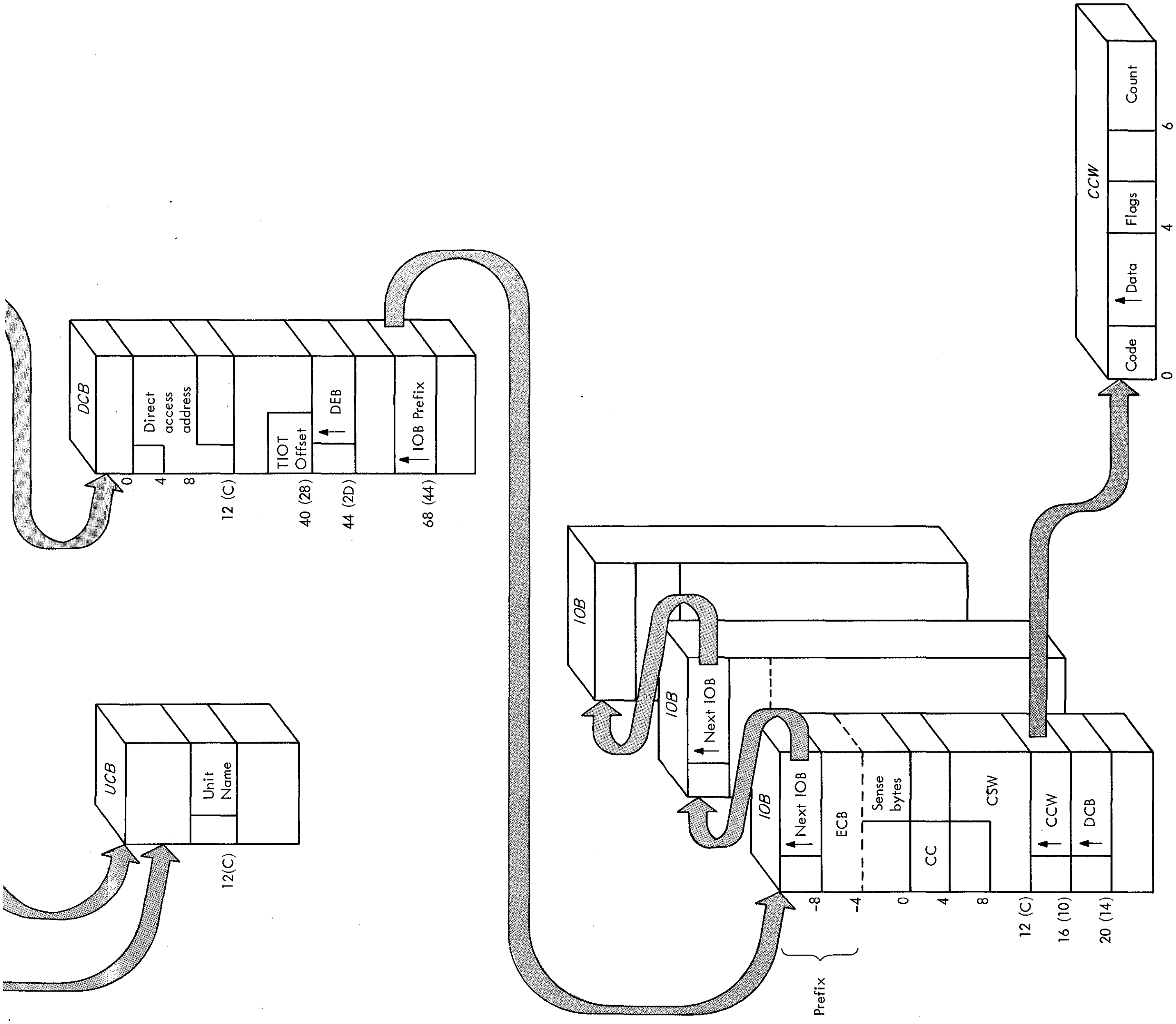


Figure 56. Control Block Flow

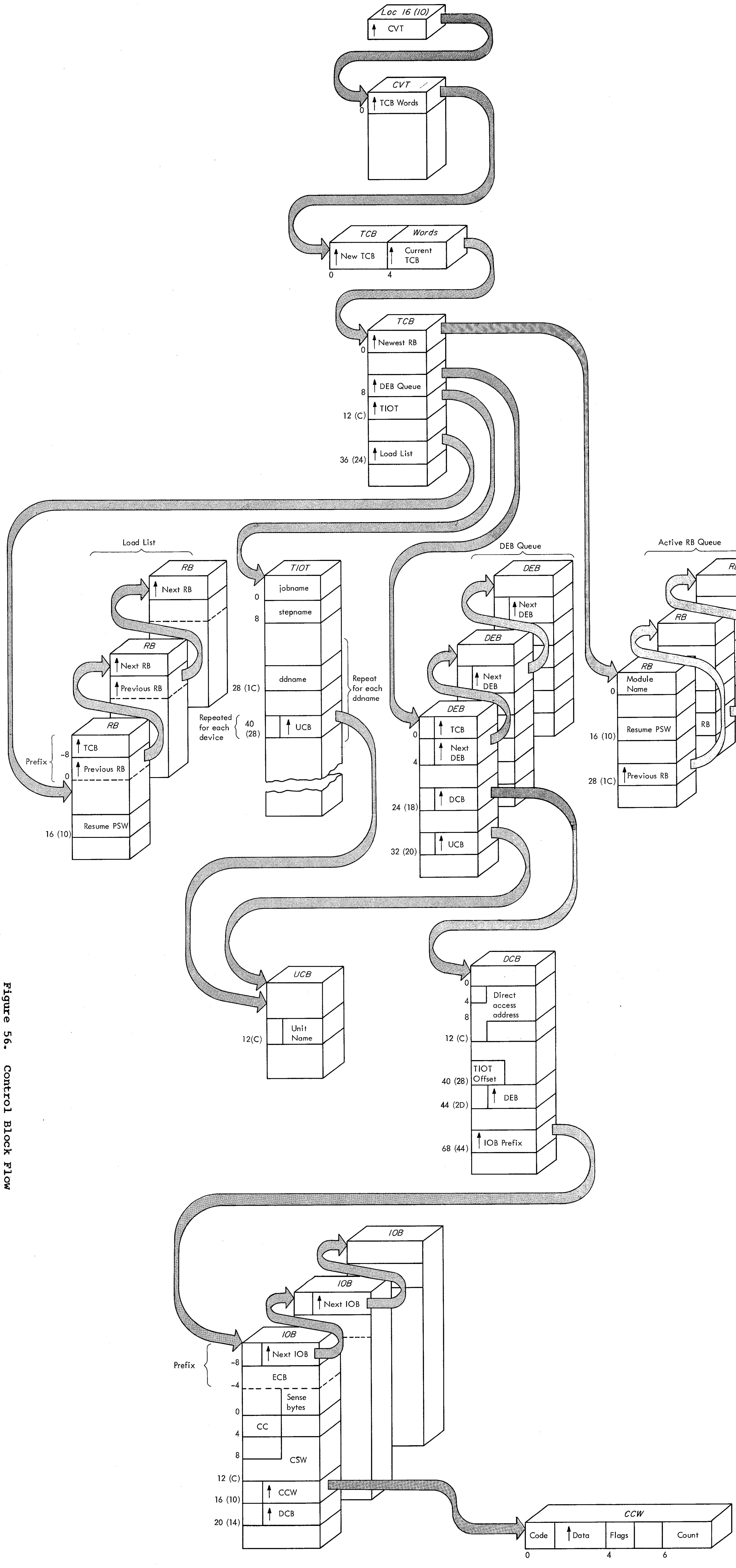


Figure 56. Control Block Flow

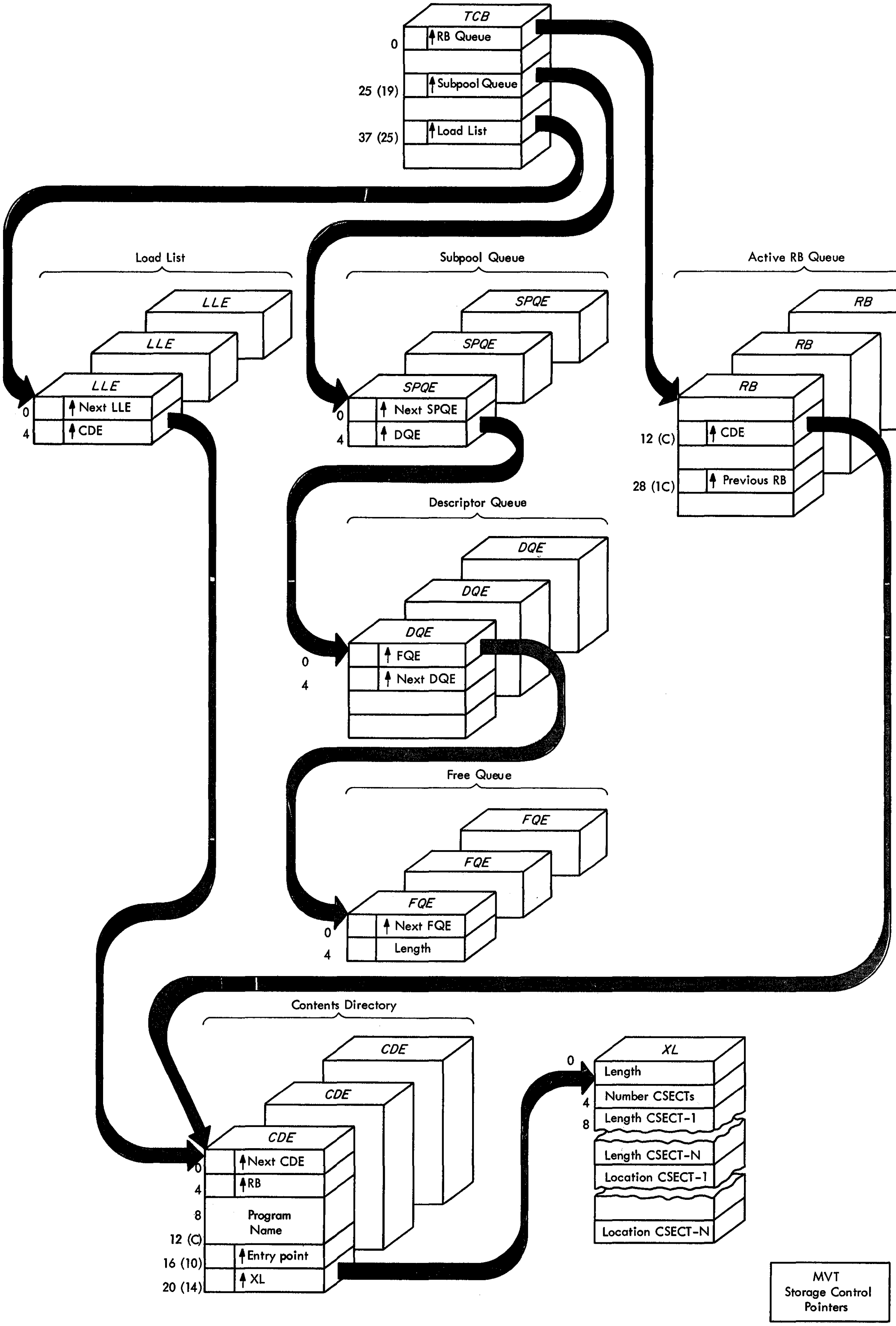


Figure 57. MVT Storage Control Flow

Appendix L: OPEN/CLOSE/EOV Debugging Aids

There are two types of traces that may be performed during OPEN/CLOSE/EOV processing, provided that GTF is active.

- ABEND trace - A trace performed before an OPEN/CLOSE/EOV problem determination module calls an ABEND routine.
- Optional work area trace - A trace performed when an OPEN/CLOSE/EOV module has finished execution. This trace is made only if DCB=DIAGNS=TRACE is specified in the DD statement of the data set for which the trace is desired.

Further information on requesting these traces is contained in <u>IBM System/360 Operating System: Data Management Services</u>, GC26-3746.

The format of both types of OPEN/CLOSE/EOV trace output is as follows:

```
USRFF  FFF  ccc    control block fields
```

USRFF

is the name (excluding the IMD prefix) of the IMDPRDMP appendage which formats the control block and work area information collected by OPEN/CLOSE/EOV and included in the GTF output data set. FF is the format ID for OPEN/CLOSE/EOV.

FFF

is the event ID which defines the event which caused the trace entry. Everything traced by OPEN/CLOSE/EOV has an event ID of FF.

ccc

is the control block that was traced to provide the problem program with OPEN/CLOSE/EOV data for debugging purposes.

When the OPEN/CLOSE/EOV ABEND trace occurs, only those control blocks meaningful to an ABEND condition will be traced. The selection of these control blocks is described in <u>IBM System/360 Input/Output Support (OPEN/CLOSE/EOV) PLM</u>, GY28-6609.

If the optional work area trace has been requested, the OPEN/CLOSE/EOV work area and the user's DCB will be traced after the execution of each OPEN/CLOSE/EOV module.

control block fields

are the contents of fields in control block ccc. For descriptions of the fields shown, refer to <u>IBM System/360 Operating System: System Control Blocks</u>, GC28-6628 or <u>IBM System/360 Operating System: Input/Output Support (OPEN/CLOSE/EOV) PLM</u>, GY28-6609.

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.

GC28-6670-5

System/360 OS Debugging Guide (S360-20) Printed in U.S.A. GC28-6670-5

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. GC28-6670-5

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.

GC28-6670-5

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. GC28-6670-5

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]