PUBLICATIONS REVISION

System 80

OS/3
Dump Analysis
Programming Guide

UP-9980 Rev. 1

This Library Memo announces the release and availability of the *System 80 OS/3 Dump Analysis Programming Guide*, UP-9980 Rev. 1.

This guide is a standard library item (SLI). It is part of the standard library provided automatically with the purchase of the product.

This guide describes how to interpret Operating System/3 (OS/3) dumps to help you understand job or system conditions. It contains the following:

- Descriptions of the structure and functions of OS/3 software
- Explanations of the task control block (TCB) and the program status word (PSW)
- Descriptions of three OS/3 dumps
- Sample dump analyses

For Release 12.0, this guide adds parameters to the suboption SELECT and incorporates minor technical and editorial changes.

Destruction Notice: This revision supersedes and replaces the *Dump Analysis User Guide/Programmer Reference*, UP-9980, released on Library Memo dated February 1984. Please destroy all copies of UP-9980, all updates, and all Library Memos.

Additional copies may be ordered through your local Unisys representative.

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS
Mailing Lists MBZ, MCZ, MMZ, MDE, MGZ, and MHA	Mailing Lists MBW, MBVO, MB00, and MB01. (116 pages plus Memo)	Library Memo for UP-9980 Rev. 1 RELEASE DATE: October 1988

UNISYS

System 80
OS/3
Dump Analysis
Programming Guide

OS/3 Release 12.0

October 1988

Priced Item

Printed in U S America
UP-9980 Rev. 1

UNISYS

System 80
OS/3
Dump Analysis
Programming Guide

OS/3 Release 12.0
October 1988

Printed in U S America

Priced Item
UP–9980 Rev. 1

Correspondence regarding this publication should be forwarded, using the User Comments form at the back of this manual or remarks addressed directly to Unisys Corporation, to E/MSG Product Information, P.O. Box 500, M.S. E5-114, Blue Bell, PA 19424 U.S.A.

PAGE STATUS SUMMARY
ISSUE: UP-9980 Rev. 1

Part/Section	Page Number	Update Level
Cover		
Title Page/Disclaimer		
PSS	iii	
About This Guide	v thru ix	
Contents	xi thru xv	
1	1	
2	1 thru 16	
3	1 thru 26	
4	1 thru 53	
Appendix A	1, 2	
Appendix B	1, 2	
User Comments Form		
Back Cover		

Part/Section	Page Number	Update Level

Part/Section	Page Number	Update Level

About This Guide

Purpose

This guide discusses dump analysis primarily as it applies to user jobs.

Scope

This guide contains the following:

- Explanations of the task control block (TCB) and the program status word (PSW), two of the most important structures in dump analysis, and an overview of the structure and functions of Operating System/3 (OS/3) software

- An overview of OS/3 dumps

- User dump analysis with examples

Audience

The intended audience for this guide is the programmer, who will use dump analysis to help develop and maintain Unisys software, programs, and files.

Prerequisites

Anyone using this guide should understand the following:

- Unisys OS/3 machine language

- How to read a dump

- Job organization

How to Use This Guide

Read the entire guide to familiarize yourself with the basic concepts it presents; then use it for reference as needed.

Organization

This document contains four sections and two appendixes:

Section 1. Introduction to Dump Analysis

Defines dump analysis, and explains when you need dumps.

Section 2. OS/3 Overview

Gives a basic overview of OS/3, including the supervisor and user region, and several data structures and registers.

Section 3. Dumps and Their Formats

Describes the three types of dumps OS/3 provides: $Y$DUMP, JOBDUMP, and EOJ dump.

Section 4. Sample Dump Analyses

Presents examples of dump analyses applied to actual user programs by using the ideas presented in the previous sections.

Appendix A. Program Exceptions

Lists the interrupt codes and their causes.

Appendix B. $Y$DUMP File Allocation

Shows the number of cylinders required for $Y$DUMP file allocation, depending on your system's main storage capacity and the type of disk device you are using.

Results

After reading this document, programmers will be able to use dump analysis to correct program errors.

Related Product Information

The following Unisys documents may be useful in understanding and implementing dump analysis.

Note: *Throughout this guide, when we refer you to another manual, use the version that applies to the software level at your site.*

Processor - System 80 Programmer Reference (UP-8881)

Provides the detailed hardware-oriented information required to program the System 80 processor. Includes a brief functional description and configurations.

Supervisor Technical Overview (UP-8831)

Provides an overview of the OS/3 supervisor and its functions for OS/3 high-level language programmers and site administrators.

Assembler Programming Guide (UP-8913)

Describes, for both novice and experienced programmers, the OS/3 basic assembly language (BAL) and its use. Discusses general language concepts, assembler instructions, and programming techniques.

System Messages Reference Manual (UP-8076)

Lists the system messages and describes them. Message description composed of the remedial action or response required as applicable.

Job Control Programming Guide (UP-9986)

Provides information on the format and use of job control statements and job control procedure calls (jprocs).

Hardware and Software Programming Quick-Reference Guide (UP-8868)

Summarizes machine instructions, supervisor related information, physical input/output control system (PIOCS) information, and I/O sense data byte definitions.

System Service Programs (SSP) Operating Guide (UP-8841)

Describes the system services programs, the utility programs that support the operation and organization of the operating system.

Operations Guide (UP-8859)

Describes the hardware configuration of each System 80 model, presents procedures for initializing the system, and describes all commands and procedures used within the OS/3 environment.

Installation Guide (UP-8839)

Provides the system administrator with information and procedures necessary to install, tailor, and maintain OS/3 software in a System 80 environment.

Notation Conventions

The notation conventions used in this guide are the following:

- Capital letters, commas, equal signs, and parentheses must be coded exactly as shown. For example:

```
SET SY,LOFF

(SELECT)

FILE=$Y$DUMP
```

- Lowercase letters and words are generic terms representing information that must be supplied by the user. Such lowercase terms may contain hyphens and acronyms (for readability). For example:

  `vsn` (volume serial number)

  `decimal-area-number`

  `job/symbiont-name`

  `did` (device identification number)

- Information contained within braces represents mandatory entries of which one must be chosen, such as:

```
DISPLAY= ⎧startaddr:n                              ⎫
         ⎪X'xxxxxxxx' [startaddr-endaddr]         ⎪
         ⎪C'cccccccc'                              ⎪
         ⎪CR                                       ⎪
         ⎨IO                                       ⎬
         ⎪PR                                       ⎪
         ⎪RR                                       ⎪
         ⎪SN                                       ⎪
         ⎪SR                                       ⎪
         ⎩JOBS                                     ⎭
```

- Information contained within brackets represents optional entries that (depending upon program requirements) are included or omitted. Braces within brackets signify that one of the specified entries must be chosen if that parameter is to be included. For example:

```
[,P=did]

⎡,V= ⎧vsn     ⎫⎤
⎣    ⎩(vsn,AT)⎭⎦
```

- An optional parameter having a list of optional entries may have a default specification that is supplied by the operating system when the parameter is not specified by the user. Although you can specify the default, you do not have to. For easy reference, when a default specification occurs in the format delineation it is printed on a shaded background. For example:

```
⎡,SPL ⎧PRINTER⎫⎤
⎣     ⎩TAPE   ⎭⎦

⎡,FILE=⎧$Y$DUMP⎫⎤
⎣      ⎩lblname⎭⎦
```

  If, by parameter omission, the operating system performs some complex processing other than parameter insertion, it is explained in the parameter description.

- When a portion of a parameter is underlined, only that portion need be specified. For example:

  $\texttt{[,MLIB=}\underline{\texttt{N}}\texttt{O]}$

  can be coded as:

```
[,MLIB=N]
```

Contents

Figures

Tables

Section 1
Introduction to Dump Analysis

1.1. General

Dump analysis is one of the set of diagnostic tools available for your use in developing and maintaining Unisys Operating System/3 (OS/3) software, programs, and files. A raw dump itself is a hexadecimal image of the system main storage and registers at the time of the dump, which is output to a printer. Dump facilities in OS/3 supplement a raw dump by extracting from it information needed to interpret job or system conditions. Dump analysis is the process by which you interpret the dump and is one of your primary tools for uncovering errors and problems in your OS/3 programs.

1.2. When Dumps Are Needed

Dumps are used on a number of occasions:

- When a HALT AND PROCEED (HPR) occurs. Many HPRs listed in the *System Messages Reference Manual* (UP-8076) request that the operator perform a dump.

- When a system hang occurs. If the system remains in a run state but cannot communicate with the operator, a dump is often taken.

- When a user job terminates with an error. If certain job control language (JCL) options are specified, the system will react to the error by generating a dump and sending it to a printer.

- When some system errors will automatically generate a system dump.

While the dump is an important diagnostic tool to you, it is an equally important tool for your Unisys representative, if a system problem is serious. For less serious problems, though, you can analyze your own dumps and resolve your program problems. In fact, the emphasis in this guide is on analyzing job failures, since system failures usually require the services of a Unisys representative.

Section 2
OS/3 Overview

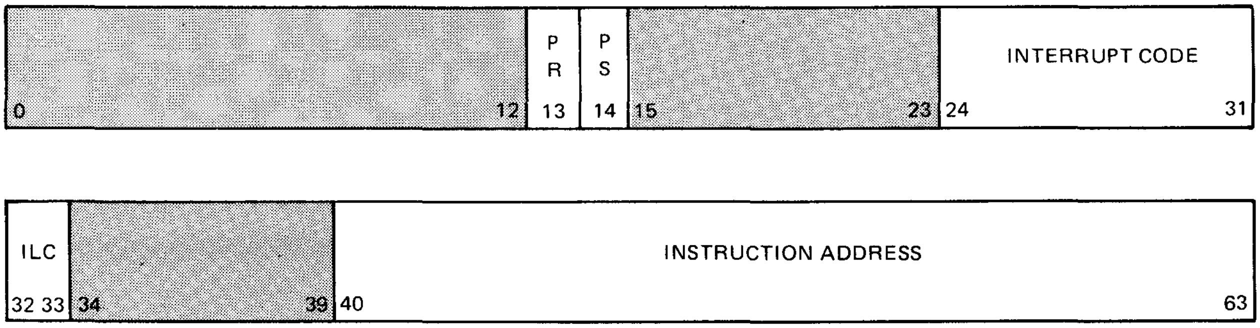
2.1. General

To do dump analysis, you should know some of the operation of OS/3 software. For OS/3 operation, main storage can be divided into two parts: the supervisor and the user region. The supervisor is the complex of routines that prepares user programs for running, serves as the primary I/O software interface, and, most importantly for dump analysis, handles unexpected or error conditions. The user region contains your program and is where most user-solvable problems occur. The two areas, supervisor and user region, occupy separate parts of main storage and share in the use of several data structures and registers, the most important of which are the program status word (PSW) and the task control block (TCB).

2.1.1. Program Status Word (PSW)

One of the most important registers in OS/3 is the current program status word. This register contains the address of the next instruction to be executed as well as other indicators and flags which, taken together, define the hardware status of the system.

The format and description of the PSW are:



NOTE:

▨ = Not used

A complete description of the PSW can be found in the appropriate processor programmer reference. For our discussion, you need know only about the unshaded fields shown in the previous illustration:

- PR (bit 13)

  Defines the general register set to be used in executing an instruction. The processor contains two sets of 16 general registers: problem registers and supervisor registers. The set the processor uses is determined by the following settings of bit 13:

PR=1	Problem registers
PR=0	Supervisor registers (not for user analysis)

- PS (bit 14)

  Specifies one of two processor modes of operation:

PS=1	Problem mode
PS=0	Supervisor mode (not for user analysis)

  In problem mode, all privileged machine instructions are invalid, and their attempted execution results in a program exception (interrupt code 02; see Table A-1). In supervisor mode, all instructions may be executed.

- Interrupt code (bits 24-31)

  Contains the interrupt code used in software analysis of interrupts. A list of the interrupt codes used in OS/3 is shown in Table A-1.

- ILC (bits 32-33)

  Contains the instruction length code, the length in half words of the instruction being executed at the time of an interrupt. For each instruction, the instruction address field is incremented by the number of bytes indicated in the ILC for that instruction.

- Instruction address (bits 40-63)

  Contains the address of the next instruction to be executed, obtained when the OS/3 hardware adds the address of the currently executing instruction to the length of that instruction.

Because the last field of the PSW, the instruction address, points to the next instruction, it can be altered by hardware branch instructions. The address can also be stored in main storage, if the current program is interrupted, and later loaded back into the PSW, permitting the program to resume without loss of control. The OS/3 software handles a number of such interrupt types, including:

- Program exception - issued by the hardware when it detects the improper use of a machine instruction or data.
- Machine check - issued by the hardware when it detects a hardware failure.
- Supervisor call - issued by user programs through use of the SVC instruction. SVC is the user's primary interface to the supervisor and may be used to call transients (2.2.1).

The OS/3 software contains error routines that handle each type of interrupt. Figure 2-1 shows how an interrupt is handled. When an interrupt occurs, the processor microcode swaps the current PSW of the interrupted program (PROG1) with a new PSW used for handling the interrupt.

The PSW for PROG1 (old PSW) is stored in a reserved slot and is used later to determine at which instruction PROG1 is to resume. The new (now current) PSW of PROG2 is loaded into the PSW register and sets the system in a new state that branches to the supervisor error routine (PROG2) that is designed to handle the interrupt.

When PROG2 routine is finished, a LOAD PSW instruction usually loads the old PSW back into the PSW register, thus returning processor control to PROG1. The hardware maintains a pair of PSWs, old and new, for each type of interrupt.

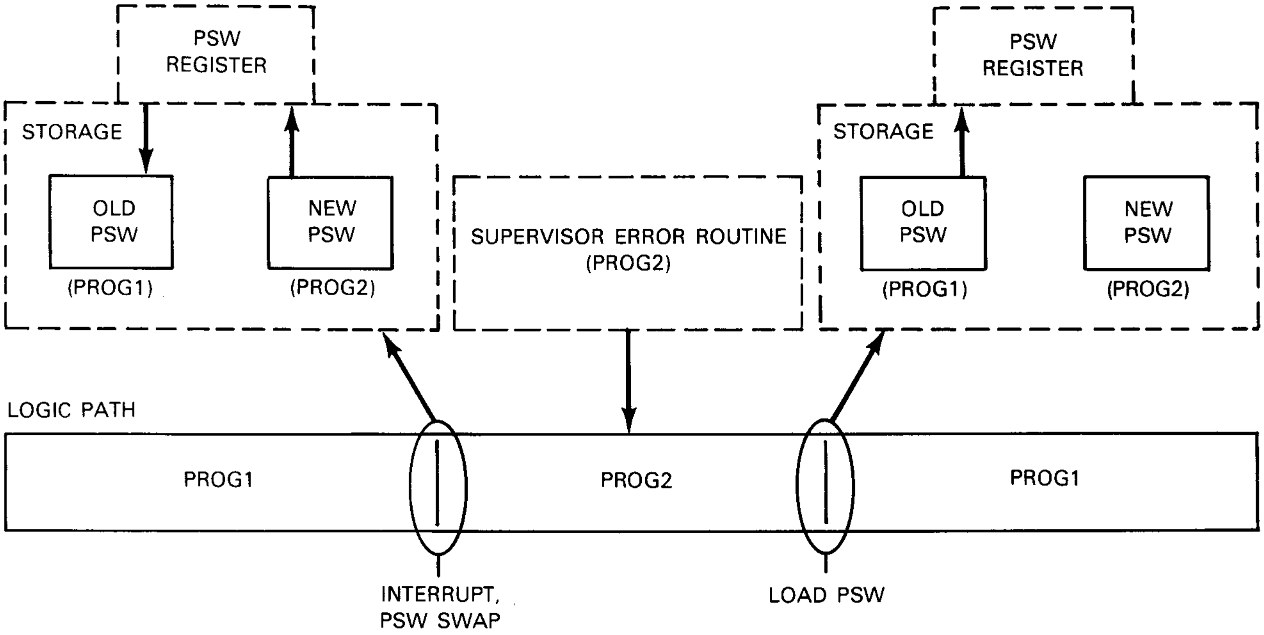


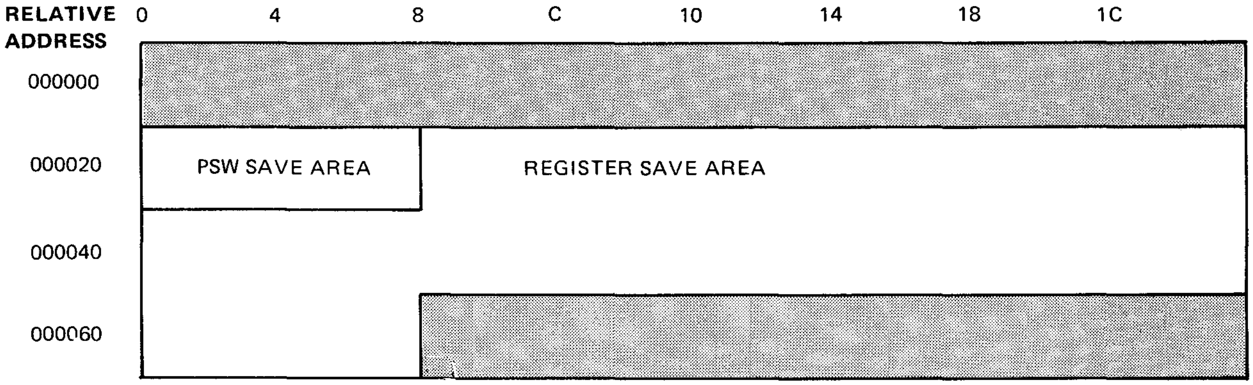
Figure 2-1. Handling an Interrupt

Because these interrupts may represent either an error condition or the operating system's response to one, their old PSWs can provide clues to the cause of an error.

2.1.2. Task Control Block (TCB)

Besides PSWs, the structure you may analyze most thoroughly in a dump is the task control block (TCB). Up to 255 TCBs may be active in OS/3 at any time, and each of these represents a task that is competing for processor time with all other tasks. A TCB contains fields that completely define its task to the rest of the system; the TCB thus serves as a software counterpart to the hardware-oriented PSW.

For dump analysis, certain fields are important, and these are represented by the unshaded portion of the following TCB format diagram:



NOTE:

▨ = Not used

- PSW Save Area (byte offset 20_{16}-27_{16})

  Two full words into which the old PSW is stored in the event of an interrupt occurring in the task. When the task resumes processor control, this data is loaded back into the current PSW.

  On abnormal termination, the PSW is also stored in the last five bytes of the preamble and the interrupt code and instruction address are displayed and written to the job log. This information is useful if a dump is not obtained.

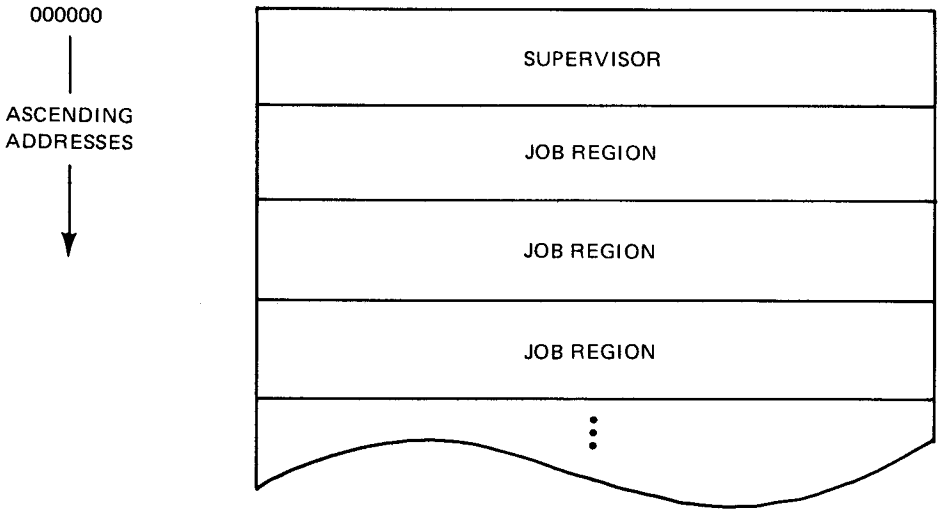
- Register Save Area (byte offset 28_{16}-67_{16})

  Sixteen full words into which the contents of the current register set are stored when an interrupt occurs in the task. When the task resumes processor control, these full words are loaded back into the proper register set.

You can find more detailed information of TCBs in the *Supervisor Technical Overview* (UP-8831).

2.2. OS/3 Structural Overview

This subsection summarizes some of the important OS/3 structures that are resident in main storage and describes how they are linked together. A knowledge of these is important to dump analysis. Broadly, main storage is allocated as follows:



The OS/3 supervisor occupies the low-order part of main storage. Job regions occupy some parts of high-order main storage, with free regions taking up the remainder. The formats of the supervisor and of the job regions are discussed in the following subsections. Samples of these are in the dumps shown in Section 3.

2.2.1. Supervisor

Three areas of the supervisor are often used in dump analysis: low-order main storage, the system information block, and the transient area.

Low-Order Main Storage and Relocation Registers

The first 4096 bytes of the main memory unit for the System 80, models 8 through 20 are reserved for special purposes by the operating system and thus cannot be used as a storage area for ordinary data and instructions. In System 80, models 3 through 6, low-order main storage occupies the lowest 256 bytes of main storage. One area of low-order main storage is of interest in dump analysis; this is shown in the following illustration:

Byte Address (Hexadecimal)	0 1 2 3	4 5 6 7	8 9 A B	C D E F
00X	▨			
01X	▨			
02X	I/O old PSW		I/O new PSW	
03X	Exigent machine check old PSW		Exigent machine check new PSW	
04X	Program check old PSW		Program check new PSW	
05X	Supervisor call old PSW		Supervisor call new PSW	
06X	External old PSW		External new PSW	
07X	Repressible machine check* old PSW		Repressible machine check* new PSW	
08X	PER old PSW		PER new PSW	
09X	Restart old PSW*		Restart new PSW*	
0AX	▨			

* Not used in Model 8.

Low-order locations 000020 through 00009F contain, for each type of interrupt supported in System 80, the new PSW that is always loaded into the current PSW register, and the old PSW that is swapped out. The PSWs for program checks, machine checks, and SVCs have already been discussed; of these, the program exception will figure the most in the dump analyses shown in Section 4.

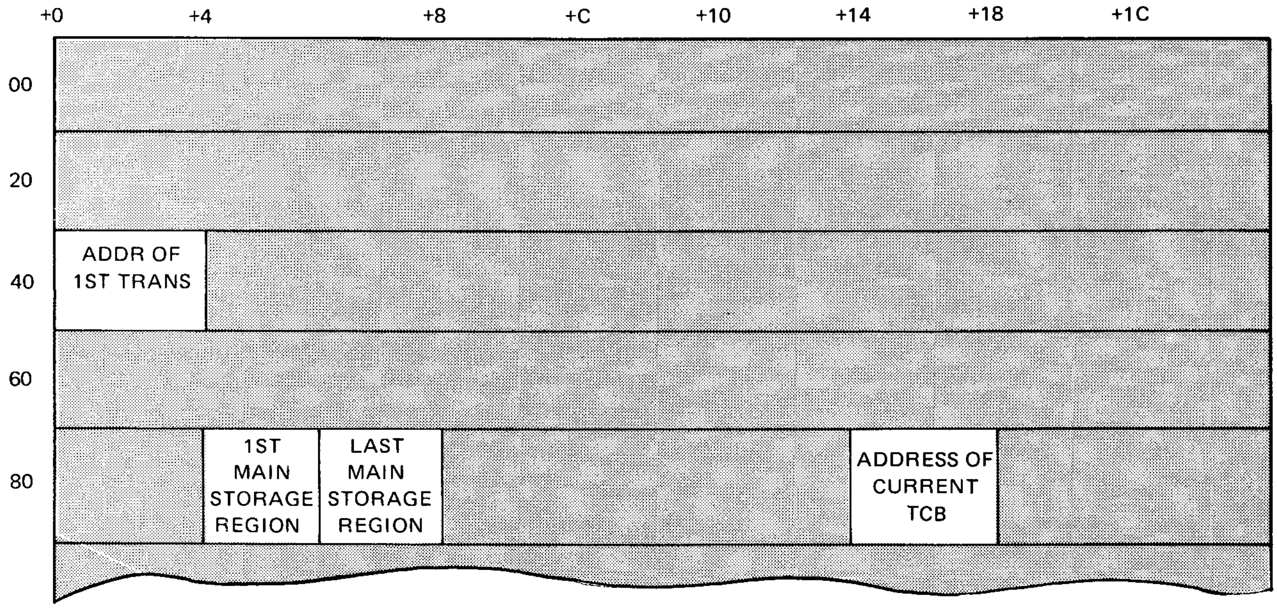
In addition to the data kept in low-order main storage, the system uses 16 relocation registers (identified as 0 through F_{16}) for models 3 through 6 and 64 relocation registers (identified as $1\text{-}40_{16}$) for models 8 through 20. In the operating system, models 8 through 20 provide up to 48 user jobs, each using one of the registers $1\text{-}30_{16}$ as its base. For models 3 through 6, the maximum number of user jobs is 14 and each job base is one of the registers $1\text{-}E_{16}$.

These registers should not be confused with the base registers used within programs, especially by BAL programmers; in fact, every effective address within a job is added to the job's relocation register to get the absolute address required, a process generally transparent to users.

All TCBs within a job specify the relocation register for that job in byte 0. While the job has processor control, its register number occupies bits 8-11 of the current PSW. Register 0 is reserved for the supervisor; the field contains a value of zero, which permits supervisor routines to address the entire address space of a system. Because it is important to distinguish between system errors and job errors in dump analysis, job regions and relocation registers are further discussed in 2.2.2.

System Information Block (SIB)

The system information block (SIB) occupies supervisor space just past low-order main storage. Fields within the SIB are often used in dump analysis and are presented in the following illustration:



NOTE: ▨ = Not used

Bytes 40 to 43 contain the full-word address of the first transient area (see 2.2.1). The two half words in bytes 84 to 87 define the user region with the keys to the first and last user blocks. The address in bytes 94 to 97 points to the current TCB, which is the last TCB given control by the system (see 2.3.1).

Transients

To handle system tasks OS/3 uses two basic types of routines: supervisor overlays and transient routines. Supervisor overlays occupy a single 1192-byte area within the supervisor; these are routines that are loaded when needed to handle urgent tasks, such as operator communications. Transient routines are loaded when needed, occupy several contiguous 1192-byte areas of supervisor, and can be called by user jobs or by other transients.

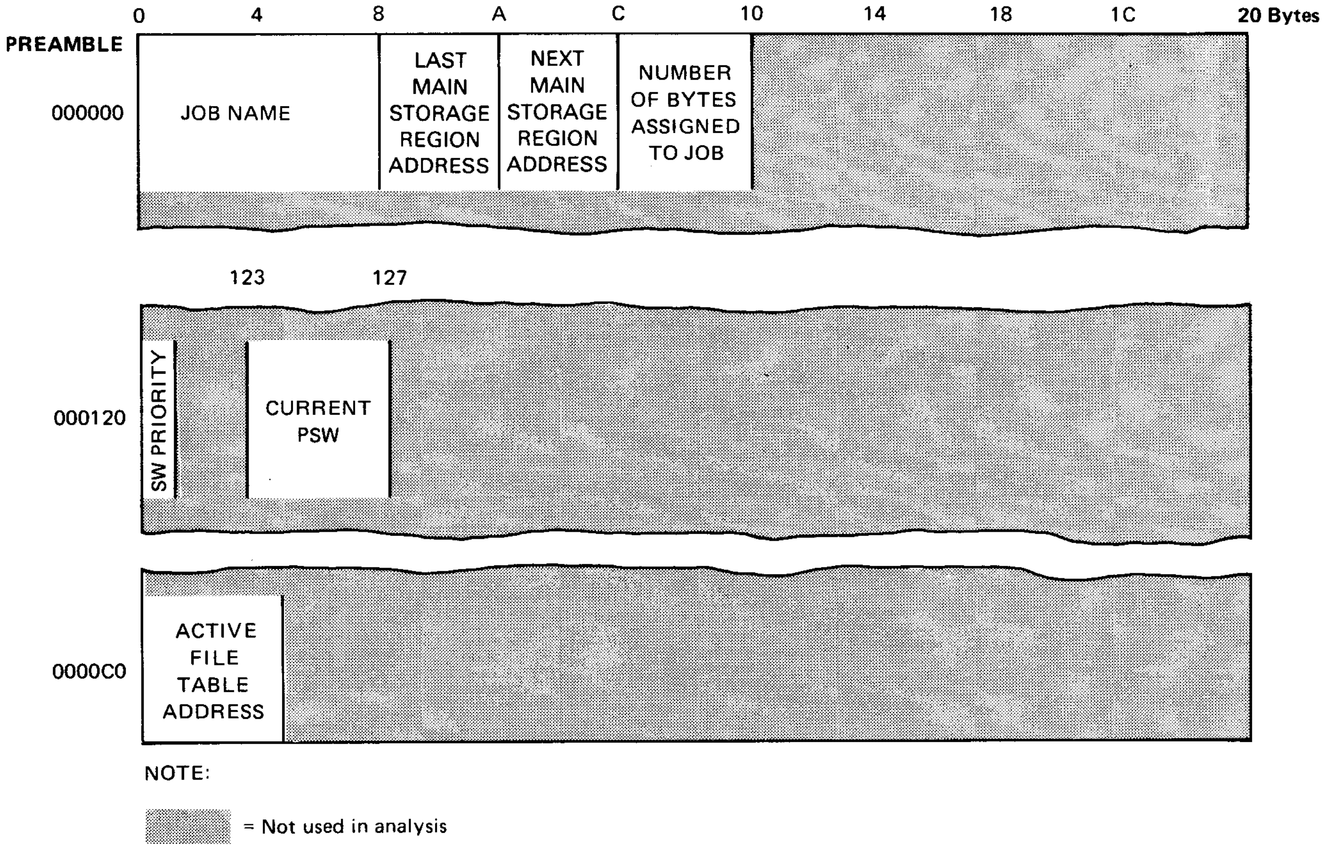
2.2.2. Job Regions

Each user job in OS/3 occupies a single job region in main storage. Within that region are all the fields unique to that job, including:

- Job preamble
- Job TCBs
- Load module

The job preamble contains data about the job that is used by the supervisor in scheduling, running, accounting, and terminating the job. The TCBs, of which at least one exists for each job, represent the individual tasks that may compete for processor time while the job is run. The load module is scheduled and loaded through job control language (JCL) or as a symbiont and may contain one or more phases.

As mentioned in 2.2.1., models 3 through 6 have 14 user slots and models 8 through 20 have 48 user slots. This means that System 80 allows you to run simultaneously up to 14 user jobs on models 3 through 6 and up to 48 user jobs on models 8 through 20. Pointers within each job preamble point to other jobs and to free regions (see 2.2.4). Each job uses one of the relocation keys discussed in 2.2.1. Five fields in the preamble are often used in dump analysis, as shown in the following illustration:


0
4
8
A
C
10
14
18
1C
20 Bytes
PREAMBLE
000000
JOB NAME
LAST MAIN STORAGE REGION ADDRESS
NEXT MAIN STORAGE REGION ADDRESS
NUMBER OF BYTES ASSIGNED TO JOB
123
127
000120
SW PRIORITY
CURRENT PSW
0000C0
ACTIVE FILE TABLE ADDRESS
NOTE:
= Not used in analysis

The eight bytes starting at location 0 contain the name of the user's job coded in EBCDIC. The two half words starting at locations 8 and A point to the user regions immediately following and preceding this region, thus defining the space addressable by this job. A user region can be a job, a symbiont, or a free region. Location C contains the number of bytes assigned to this job.

Location 120 contains the switch priority, if a switch command is entered that is to be in effect for the entire job. Otherwise, this byte will be zero.

Location 123 through 127 will contain the low-order five bytes of the current PSW (containing the interrupt code and instruction address) when a job abnormally terminates. Job control will write this information to the job log and display.

Location C0 contains the address of the active file table, which is used as an interface between the job and the files assigned to it (see 4.4.2).

2.2.3. Symbionts

Symbionts are dynamic extensions of the supervisor which occupy user main storage in much the same way as do user jobs. Each symbiont has a job-like preamble, TCBs preceding the code executed in the symbiont, and other necessary structures.

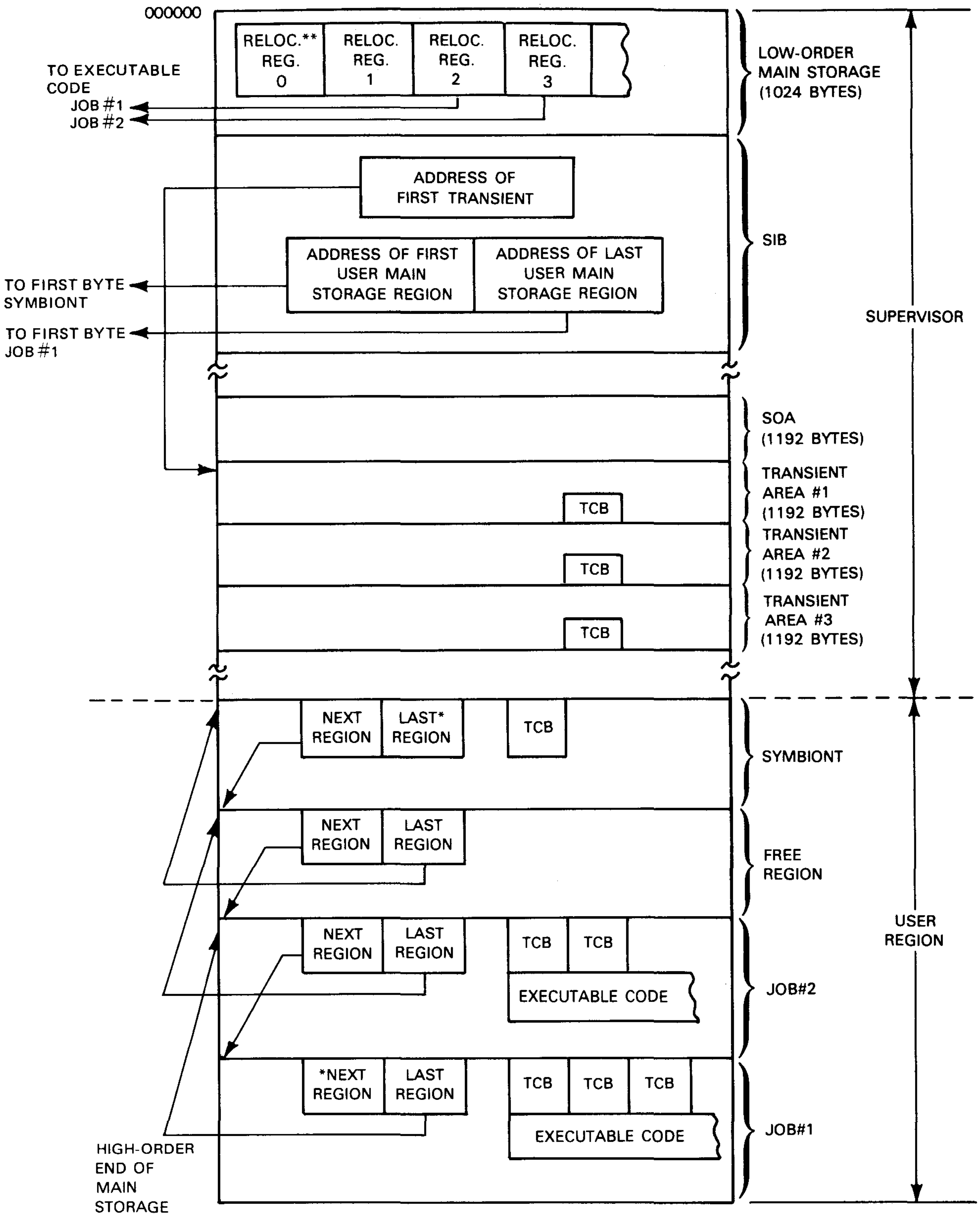
Some of the differences between symbionts and jobs lie in priority (see 2.3.1) and relocation keys: symbionts use key 0 (the supervisor) and run in supervisor mode; also, symbionts are loaded from the low-order end of user main storage upwards and are scheduled and loaded separately from job control.

2.2.4. Free Regions

As explained in 2.2, free regions are those parts of the user region not allocated to user jobs or symbionts. These regions are not simply ignored in OS/3, however: like a job or symbiont, each such region has a preamble with pointers in locations 8-B indicating the jobs/symbionts immediately preceding and following it, thus maintaining forward and backward links throughout the entire user region.

2.2.5. Structure Summary

The separate structures and pointers of an OS/3 system can be pulled together into a single diagram, shown in Figure 2-2.



*Contains 0's indicating that it does not point to another region. **Relocation registers not in main storage

Figure 2-2. Summary of OS/3 in Main Storage

In Figure 2-2, two jobs and one symbiont are running concurrently, leaving a free region between them. Note that two relocation registers in low-order main storage are each pointing to the first byte of executable code in a job region. In analyzing dumps caused by errors within a user job, these relocation values can help determine specifically what job caused the error. Note that each job can have more than one TCB, although only one TCB can be active at any time.

2.3. Functional Description

This subsection summarizes some of the major operations of OS/3 software. You can find more information in the *Supervisor Technical Overview* (UP-8831). For dump analysis purposes, this discussion focuses on OS/3 tasks and task handling.

2.3.1. TCBs and Multitasking

As described previously, OS/3 tasks are processes that compete with other tasks for processor time. Each task is defined by an associated TCB. Each job or symbiont in the system is considered a primary task to which other tasks may be attached. The supervisor overlay and transient areas are individual tasks having one TCB each.

Tasks can be active or waited. An active task is one capable of getting processor control, but a waited task is made ineligible for processor control and must wait, usually for some event to occur, until it can be made active again.

Tasks are allotted processor time by the task switcher, a supervisor routine that scans all active tasks and chooses one among them to get the processor's attention. The TCB having the highest priority gets the processor first. Figure 2-3 shows the relative priorities assigned to different classes of TCBs.

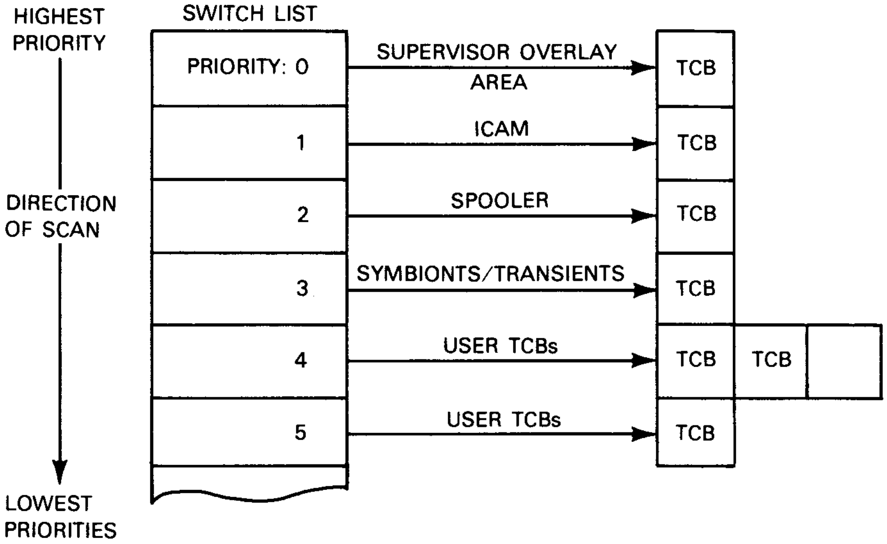


Figure 2-3. OS/3 Task Priorities

Note that the supervisor overlay TCB has the highest priority since supervisor routines must be run as soon as they are needed. As you move down the list, the tasks move from supervisor-oriented to user-oriented until you reach priority 4, the highest priority a user task can have.

Some priority levels have more than one task, although the tasks may be associated with different jobs and routines. When one task finishes or is interrupted, the switcher usually gives processor time to the next TCB at that priority if one exists and is active. Otherwise, the switcher scans lower priorities until it finds another active TCB to which to give processor control.

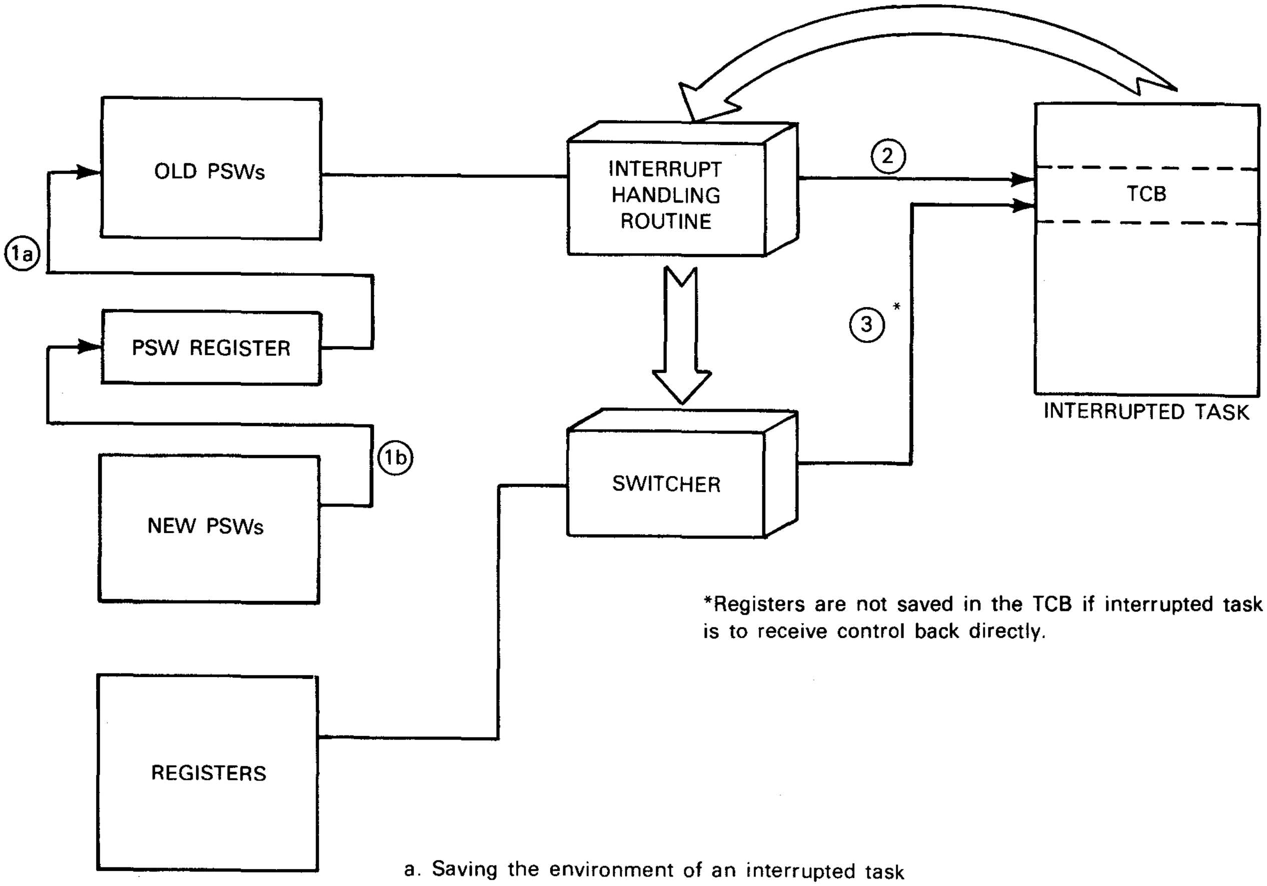
2.3.2. Task Switching Considerations

Switcher activity within a system can be determined from a dump and can thus help tell you why an error occurred within a job (which, remember, is itself a task operated on by the switcher).

Much task switching is done in response to an interrupt. When an interrupt is processed (Figure 2-4a) the current PSW is stored in the appropriate old PSW region of low-order main storage (step 1A). The corresponding new PSW is then loaded into the current PSW, thus handling processor control to an interrupt handling routine (step 1B). That routine in turn copies the old PSW into the double word PSW save area in bytes 20-27 of the interrupted task's TCB (step 2). Next, the routine in control stores the 16 registers used by the interrupted task into bytes 28-67 of the same TCB (step 3). (This step is omitted if control goes back to the same task as the one interrupted.) Finally, the interrupt handler returns processor control to the switcher.

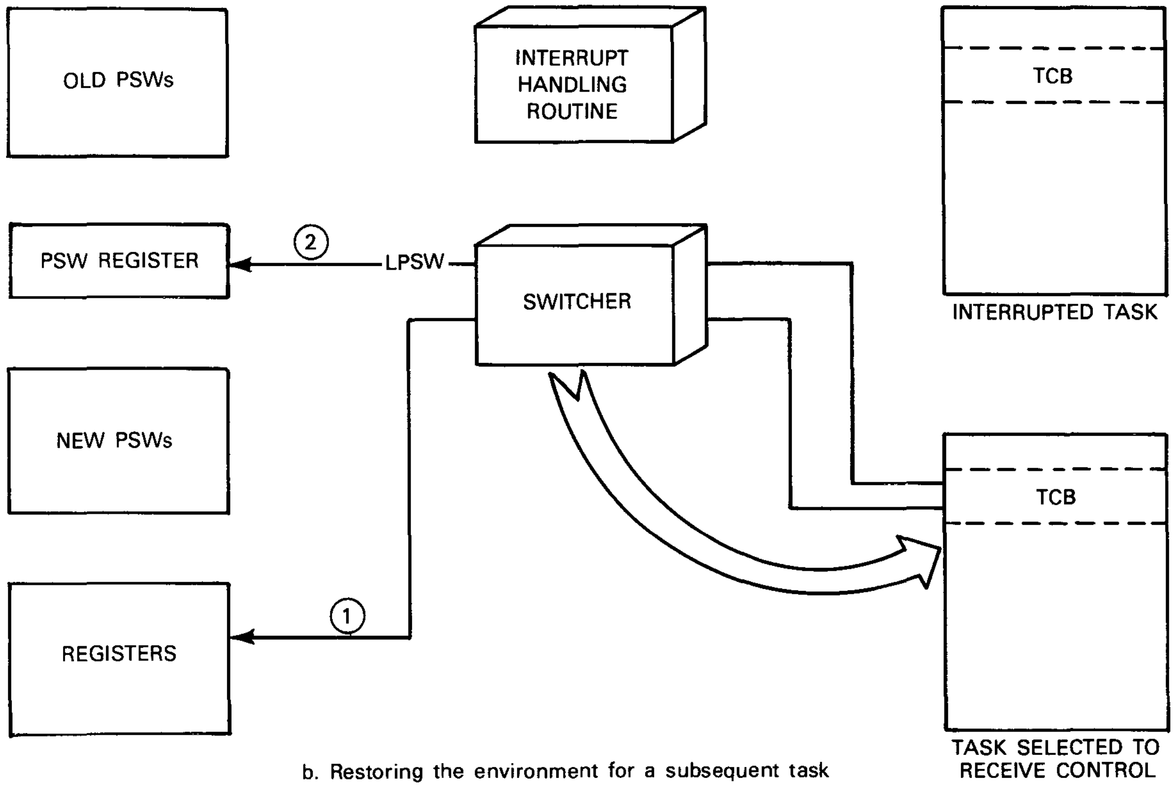
The result of these operations is that control can be easily returned to the interrupted task when the time comes. This is shown in Figure 2-4b when the switcher gives control to a task which, like the task in Figure 2-4a, has had its registers and PSW saved. The switcher simply loads the 16 registers from the register save area (step 1) and the current PSW from the PSW save area (step 2). Control then passes to the instruction pointed to by the now-current PSW and the newly activated task begins processing.

As you will see later, task switching within a job is often prompted by interrupts responding to errors: in these cases, the old PSW (saved in the TCB of the interrupted task) shows which instruction caused the error. This is one of the most important techniques of dump analysis and is used extensively in the dump analyses in Section 4.


OLD PSWs
INTERRUPT HANDLING ROUTINE
2
TCB
1a
PSW REGISTER
3
*
INTERRUPTED TASK
1b
SWITCHER
NEW PSWs
*Registers are not saved in the TCB if interrupted task is to receive control back directly.
REGISTERS

a. Saving the environment of an interrupted task

Figure 2-4. Passing Control from Task to Task (Part 1 of 2)



b. Restoring the environment for a subsequent task

Figure 2-4. Passing Control from Task to Task (Part 2 of 2)

Section 3
Dumps and Their Formats

3.1. General

OS/3 provides three types of dumps. While all are basically hexadecimal images of main storage at the time they are taken, they differ in scope and purpose as follows:

- SYSDUMP

  Dumps all or part of main storage and is run in two phases: main storage write and dump printout. In addition to the raw dump, the printout can provide a picture of your system in charts and text.

  To produce an OPTION SYSDUMP, the RUN pack must have enough contiguous free space to store a copy of the system's main storage. See Appendix B for a procedure to allocate enough file space for your system.

- JOBDUMP

  Dumps a user's job region upon abnormal termination of the job or execution of the DUMP or CANCEL macros. The raw dump is supplemented by charts and text interpreting the state of the job. JOBDUMP requires space on the first VTOC cylinder of SYSRUN to allocate files. You should eliminate unused files on SYSRUN.

- EOJ Dump

  Dumps a user's job region without translation as well as the registers and the PSW.

The three types of dumps contain some common features, particularly in the user regions. The type of dump you choose to take will depend on some of the following considerations:

- HALT AND PROCEED (HPR) errors often require a SYSDUMP so that Unisys representatives can analyze system problems.

- System loops or hangs, usually appearing as operator communications that receive no reply from the system console, require a SYSDUMP. Like HPR errors, these usually require that a Unisys representative analyze your dump for system problems.

- User job errors generally occur within the user's own job region, thus requiring nothing more than a JOBDUMP or EOJ dump. These are the dumps you are most likely to work with in analyzing user program errors.

3.2. System Dump Routine (SYSDUMP)

3.2.1. SYSDUMP Makeup

The system dump routine (SYSDUMP) is most often used to determine why your system failed. But it may be used at any other time (even during normal processing). For example, you can use it interactively to dump all or specific regions of your job and associated system activity. SYSDUMP provides you with a listing that translates the state of the operating system when the dump was taken. In the case of a system failure, the operating system is shown just as it was at the time of the failure.

SYSDUMP is made up of two parts:

- A translated dump, which translates the state of the entire operating system into charts and text

- A labeled hexadecimal/character main storage dump

SYSDUMP is part of the supervisor and is always included at system generation (SYSGEN) time. It is designed to be run in a multiprogramming environment.

To get a system dump, the following two phases must occur:

1. The supervisor must write an image of main storage to the system dump file ($Y$DUMP). This phase is called the main storage write phase.

2. The SYSDMP load module, residing in load library $Y$LOD, must be called in and run. It is SYSDMP that reads the $Y$DUMP file and prints the translated and main storage dumps. This phase is called the execution phase.

Note: *The system dump file, $Y$DUMP, does not have to be on the SYSRES disk. It can be assigned to any disk. Refer to the* Installation Guide *(UP-8839) for more information.*

Generally, the execution phase of SYSDUMP immediately follows the main storage write phase. (The only exception to this is a SYSDUMP obtained from the operator controls. For this the operator must call SYSDMP; see 3.2.2.) Because the contents of main storage at the time of the main storage write are frozen in $Y$DUMP, you need not worry about the execution phase or any other job changing the contents of the system dump before you see the listing. This also means that you may use the SYSDUMP SAVE and RESTORE options (see 3.2.4) to perform a main storage write phase at one site and an execution phase at an entirely different site.

The operating system can generate a system dump in response to action by the operator, in response to a system error, or upon encountering an error in your job.

3.2.2. Operator-Initiated SYSDUMP

You can initiate a dump from either the operator controls or the system console/console workstation. Perform Step 1A if you are using the operator controls (located on the console workstation); perform Step 1B if you are using the system console.

Note: *The handbook referenced in these procedures is the* Operations Guide *(UP-8859).*

Step 1A. Starting from the operator controls

To initiate the main storage write phase, follow this procedure:

- For models 3 through 6:

  While pressing the FUNCTION key:

  - Press the D key to place the system in debug mode.

  - Press the RESTART key.

  The dump is completed when the following appears on the screen:

  ```
  IPL
  ```

- For models 8 through 20:

  - Press the ESCAPE key on the console, then press M (for maintenance). A menu appears on the screen.

  - Select L (for system reset) in the menu and transmit. The same screen is displayed.

  - Press U (for run), then transmit. (Alternatively, you may press ESCAPE, followed by R, a system function.) When the dump is successfully written to the $Y$DMP file on disk, the screen shows 99999999 in the bottom right corner of the screen.

  - Perform an initial program load (IPL) on the system according to the directions in the *Operations Guide* (UP-8859).

When the IPL operation is finished, SYSDUMPO (the job stream that runs the SYSDMP module) will run automatically.

Note: *After Step 1A, you may get an HPR of 999F. This means that the $Y$DUMP file still contains information to be processed by SYSDMP (from a previous main storage write operation that has not yot been printed). If this happens, you can:*

- *Print it. - Ordinarily, the information in $Y$DUMP is the system dump data most helpful to the Unisys representative who maintains your system. To print it, perform another IPL on the system according to the procedure in Step 1A.*
- *Overwrite it. - If you are sure you don't need the information in $Y$DUMP, you can write over it with an image of the system's main storage at the time you perform Step 1A. To do this, press*
  - *The FUNCTION and START console keys for models 3 through 6*
  - *The letter U and transmit (or the ESCAPE key followed by the letter R) for models 8 through 20*

  and wait for memory write completion. Perform an IPL on the system according to the directions in the Operations Guide *(UP-8859). When the IPL operation is finished, the SYSDUMPO jobstream will run automatically.*

Step 1B. Starting from the system console

To initiate a dump, key in the following:

```
SYSDUMP
```

The SYSDUMP command writes main storage to the $Y$DUMP file, then enters a job in the job queue, whose only function is to print the system dump. The job is named SYSDMPxx, where xx is the unique number assigned to this job by the system. When the job is scheduled and run, the following message appears on the system console:

```
DUMP OPTION (ALL, NONE, DUMP, JOBS, EDIT, MINI, SAVE, RESTORE)
```

See Step 2 for an explanation of these parameters.

Note: *After the supervisor writes main storage to $Y$DUMP, it locks that file until job SYSDMPxx is complete. If job SYSDMPxx is removed from the system prior to displaying the DUMP OPTION message, you must unlock the $Y$DUMP file by entering the following console command:*

```
SET SY,LOFF
```

Two examples of when this command must be used are the following:

1. *If you delete job SYSDMPxx from the job queue before it is scheduled*

2. *If the run processor encounters an error while trying to put job SYSDMPxx in the job queue*

Step 2.

Enter the SYSDUMP command in one of the following three formats, depending on the output you want (unless the supervisor has already called it automatically).

Format 1

```
RV SYSDUMP
```

Produces an output identical to that produced by running JOB SYSDUMPO with the dump option ALL. (See Formats 2 and 3).

Format 2

```
RV SYSDUMPO [ ,,DO= ⎧ ALL     ⎫ [ ,V= ⎧ vsn             ⎫ ] [,P=did] ]
                    ⎪ NONE    ⎪       ⎩ (vsn[,A],[dev]) ⎭
                    ⎪ DUMP    ⎪
                    ⎪ JOBS    ⎪
                    ⎨ EDIT    ⎬
                    ⎪ MINI    ⎪
                    ⎪ SAVE    ⎪
                    ⎩ RESTORE ⎭
```

Keyword Parameter DO

```
DO= ⎧ ALL     ⎫
    ⎪ NONE    ⎪
    ⎪ DUMP    ⎪
    ⎪ JOBS    ⎪
    ⎨ EDIT    ⎬
    ⎪ MINI    ⎪
    ⎪ SAVE    ⎪
    ⎩ RESTORE ⎭
```

Specifies the dump option you want. Available options are:

ALL

The dump listing displays and translates the state of your entire system, including a header page, low-order main storage, the system information block, the physical unit block, the main storage map (giving the locations of everything in main storage), the system switch list, the job region for each job or symbiont in the system, and the free region. Essentially, the ALL option combines the DUMP and EDIT options, supplying a hexadecimal dump with an English translation.

NONE

No output listing is produced.

DUMP

Writes the entire main storage in hexadecimal format.

JOBS

Produces a hexadecimal dump and a translation for all the jobs, symbionts, and shared code in main storage at the time it was written to the $Y$DUMP file.

EDIT

Provides an English language description of the state of the system.

Note: *EDIT replaces TRANSLATED. Both are supported and they are functionally equivalent.*

MINI

Provides a printed hexadecimal dump, including a table of contents, of specific main storage regions dependent upon the HPR or system error code. This is the recommended option for dumps to be sent to Unisys for analysis.

SAVE

Saves the $Y$DUMP file to magnetic tape or diskette (see 3.2.4).

RESTORE

Restores the $Y$DUMP file from magnetic tape or diskette (see 3.2.4).

If you omit keyword parameter DO or if the supervisor has already called SYSDMP (as with the SYSDUMP console command), then the system will display the following message:

```
DUMP OPTION (ALL, NONE, DUMP, JOBS, EDIT, MINI, SAVE, RESTORE)
```

You then enter the *one* option you want. SYSDMP redisplays this message after finishing a SAVE or RESTORE function, so you can perform other SYSDMP operations within the same job.

Keyword Parameter V

```
V= { vsn               }
   { (vsn[,A],[dev])   }
```

This parameter allows you to display a dump not residing on the booted SYSRES device.

`vsn`

If specified, all data to be processed reside on the specified vsn. If not specified, the booted SYSRES is used.

`A`

Indicates that the SYSDMP load module will also be executed from the specified vsn. The vsn item must be an alternate SYSRES disk.

`dev`

Indicates that the device number of the disk pack is specified. Use this option when you have two volume serial numbers that are the same.

Keyword Parameter P

`P=did`

If you wish to assign a specific printer as the output device for SYSDUMP, enter its 3-character device address using this keyword parameter. You may wish to do this if your system supports spooling but you do not want to spool the SYSDUMP listing.

If you want to enter more specifics in the RV SYSDUMPO command, use the suboptions shown in Format 3.

Format 3

```
RV SYSDUMPO [,,DO= { ALL[(SYSTEM)]               } [,V= {vsn              }] [,P=did] [,FILE= {$Y$DUMP}]]
                   { NONE                        }      {(vsn[,A],[dev])  }                   {lblname}
                   { DUMP [(NOSHARE) ]           }
                   {      [(NOBUFFER)]           }
                   {      [(CACHE)   ]           }
                   {      [(SYSTEM)  ]           }
                   {      [(SELECT)  ]           }
                   { JOBS [(NOSHARE) ]           }
                   {      [(NOBUFFER)]           }
                   {      [(SELECT)  ]           }
                   { EDIT                        }
                   { MINI                        }
                   { SAVE                        }
                   { RESTORE                     }
            [, {FMT= {F}} ] [,MLIB=NO] [,SPL= {PRINTER}] [,FDD=NO] [,BUF= {NO }]
            [  {     {C}} ]               {TAPE   }                  {YES}
            [  {PT=NO   } ]
            [   PT=YES    ]
```

Note: *Of the Format 3 suboptions, SELECT is mutually exclusive of NOSHARE, NOBUFFER, CACHE, and SYSTEM.*

Keyword Parameter DO

Same as in Format 2, except with the following suboptions and options added.

(NOSHARE)
: Excludes shared code modules in SYSDUMPs. NOSHARE can be used with any SYSDUMP option except with SAVE, RESTORE, or SELECT.

(NOBUFFER)
: Excludes the display of dynamic buffers and the dynamic shared code.

(CACHE)
: Displays the entire CACHE main storage. If not selected, dump options JOB and DUMP will display only 6K of CACHE.

(SYSTEM)

Displays the following preselected regions of the supervisor

Low Core

System Information Block (SIB)

Switch List

Physical Unit Blocks Table (PUBS)

Shared Code Directory

Supervisor Debug Tables

Supervisor Overlay Area (SOA)/Transient Regions

System Spool Control Table

Dynamic Buffer Information Block

Resident Buffer Control Block

Shared Code Information Block

(SELECT)

Allows you to interactively select particular regions of the main storage dump. *This suboption creates an interactive environment at the operator's console.* You will then be prompted to enter requests as follows:

```
SD28    SELECT(DISPLAY,JOBNAME,MEMORY,REGION,SHARED...
SD28    ....TRANAREA,TERMINATE)
```

An abbreviated version of this message:

```
SD28    SELECT(DIS,JOB,MEM,REG,SHAR,TRAN,TERM)
```

appears after the first selection is made. Thereafter, the complete message is displayed only after an incorrect response or when the HELP prompter is invoked.

Note: *At any time during entry of SELECT suboptions, you can enter a SELECT keyword followed by a blank space, and a help prompter for that keyword will appear on screen.*

You can then enter one of the following keyword requests:

```
⎧ DISPLAY=⎧ startaddr:nn                                  ⎫ ⎫
⎪         ⎪  { X'xxxxxxxx' } [startaddr-endaddr]          ⎪ ⎪
⎪         ⎪  { C'cccccccc' }                              ⎪ ⎪
⎪         ⎪  ⎧ CR   ⎫                                     ⎪ ⎪
⎪         ⎪  ⎪ IO   ⎪                                     ⎪ ⎪
⎪         ⎨  ⎪ PR   ⎪                                     ⎬ ⎪
⎪         ⎪  ⎨ RR   ⎬                                     ⎪ ⎪
⎨         ⎪  ⎪ SN   ⎪                                     ⎪ ⎬
⎪         ⎪  ⎪ SR   ⎪                                     ⎪ ⎪
⎪         ⎩  ⎩ JOBS ⎭                                     ⎭ ⎪
⎪ JOBNAME=                                                  ⎪
⎪ MEMORY=                                                   ⎪
⎪ SHARED=                                                   ⎪
⎪ TRANAREA=                                                 ⎪
⎪ REGION=                                                   ⎪
⎩ TERMINATE=                                                ⎭
```

where:

`DISPLAY=startaddr:nn`

Displays 1 to 10 lines of main storage on the system console or 1 to 20 lines on a user workstation, from the start address forward. Enter a hexadecimal value for startaddr.

When this keyword is in effect, a blank response results in the next logical main storage addresses being displayed. The line count is the same as the previous response.

```
DISPLAY= { X'xxxxxxxx' } [(startaddr-endaddr)]
         { C'cccccccc' }
```

Displays the address of the first occurrence of the sequence specified. For the address of the next occurrence, return a blank response to the SELECT request with this keyword in effect.

where:

xxxxxxxx
Is a hexadecimal character string of up to 16 bytes.

cccccccc
Is an ASCII character string of up to 16 characters.

startaddr-endaddr

Limits the search for the hexadecimal or ASCII character string to the range specified.

```
DISPLAY=( CR   )
        | IO   |
        | PR   |
        < RR   >
        | SN   |
        | SR   |
        ( JOBS )
```

Displays the contents of specified registers.

where:

CR

Represents the control registers.

IO

Represents the I/O relocation registers (models 3 through 6 only).

PR

Represents the problem registers.

RR

Represents the relocation registers.

SN

Represents the snap registers. (Not available on models 3 through 6.)

SR

Represents the supervisor registers.

JOBS

Lists the names and addresses of all the jobs in the system.

JOBNAME=job/symbiont-name

Specifies an 8-character job/symbiont name, which produces a hexadecimal dump of that job/symbiont. If you choose the JOB option, an English language description is also provided.

If there is more than one copy of a job or symbiont, SYSDUMPO displays a list of the duplicate jobs or symbionts and their addresses. You can then choose the copy you want. For example:

```
15?SYSDMP76 SD28    SELECT (DIS,JOB,MEM,REG,SHAR,TRAN,TERM)
15 J=SL$$OWOO
16 SYSDMP76 SD40    THERE ARE 002 COPIES OF SL$$OWOO
17 SYSDMP76 SD40    ENTER NUMBER OF DESIRED ADDRESS.  (0=NONE)
18?SYSDMP76 SD40    ENTER 1=06D100  2=076800
18 1
19?SYSDMP76 SD40    SELECT (DIS,JOB,MEM,REG,SHAR,TRAN,TERM)
```

```
MEMORY= { startaddr { -endaddr     } }
        {           { ,byte-count  } }
        {           { [ -END ]     } }
        { ALL       { [ ,END ]     } }
```

where:

startaddr-endaddr

Dumps main storage from the start address to the end address specified. Note the use of the dash. Enter hexadecimal values for startaddr and endaddr.

startaddr,byte-count

Dumps the number of bytes of main storage specified, starting at the address specified. Note the use of the comma. Enter a hexadecimal value for startaddr and a decimal value for byte-count.

startaddr-END

Dumps main storage from the start address to the end of machine capacity. Enter a hexadecimal value for startaddr.

startaddr,END

Same as startaddr-END.

ALL

Dumps all the main storage.

SHARED=module-name

Specifies an 8-character shared code module name whose module is dumped.

```
TRANAREA= ⎧ALL                 ⎫
          ⎪decimal-area-number ⎪
          ⎨D'ID'               ⎬
          ⎩X'ID'               ⎭
```

where:

`ALL`
Dumps all transient areas.

`decimal-area-number`
Dumps a specified transient area.

`D'ID'`
Decimal ID of the requested transient area.

`X'ID'`
Hexadecimal ID of the requested transient area.

```
REGION
```

You may enter one of the following acronyms or words designating a particular region:

```
⎧CACHE       ⎫
⎪DBIB        ⎪
⎪DEBUG       ⎪
⎪DISABLED    ⎪
⎪FREE        ⎪
⎪ISIB        ⎪
⎪LOWCORE     ⎪
⎪PUBS        ⎪
⎨RBCB        ⎬
⎪SCD         ⎪
⎪SCIB        ⎪
⎪SIB         ⎪
⎪SOA         ⎪
⎪SUPERVISOR  ⎪
⎪SWITCHLIST  ⎪
⎪TRACE       ⎪
⎩UNIDENTIFIED⎭
```

where:

`CACHE`
Cache module and buffers

`DBIB`
Dynamic buffer information block

DEBUG
System debug tables

DISABLED
Main storage as disabled

FREE
Main storage as free

ISIB
Interactive services information block

LOWCORE
Main storage from location zero to system information block (SIB)

PUBS
Physical unit blocks table

RBCB
Resident buffer control block

SCD
Shared code directory

SCIB
Shared code information block

SIB
System information block

SOA
Supervisor overlay area

SUPERVISOR
Dumps the entire supervisor region

SWITCHLIST
Switch list

TRACE
System trace table

UNIDENTIFIED
Unlabeled region of main storage (MEMORY) in synopsis

TERMINATE

Ends the dump session. When a selection has been processed, you will always be prompted to enter another selection until you choose TERMINATE. A keyword remains in effect until you choose TERMINATE. For example, if you select REGION=SIB, you may then enter any accepted region acronym without repeating the REGION=keyword.

Keyword Parameter V

The parameters are defined in Format 2.

Keyword Parameter P

The parameters are defined in Format 2.

Keyword Parameter FILE

$$\text{FILE=}\left\{\begin{array}{l}\text{\$Y\$DUMP}\\ \text{lblname}\end{array}\right\}$$

where:

FILE=$Y$DUMP

This default parameter allows a dump image to be processed from system file $Y$DUMP.

FILE=lblname

Allows a dump image to be processed from some file other than system file $Y$DUMP.

Keyword Parameter FMT

$$\text{FMT=}\left\{\begin{array}{l}\text{F}\\ \text{C}\end{array}\right\}$$

where:

FMT=F

Provides a full System Maintenance Change (SMC) listing after the dump.

FMT=C

Provides a condensed SMC listing after the dump. (This is the default.)

Keyword Parameter PT

```
{ PT=NO  }
{ PT=YES }
```

where:

PT=NO
: System Maintenance Change (SMC) listing is not printed. If this is entered, you cannot specify any FMT.

PT=YES
: SMC listing is printed. If nothing else is specified (FMT or PT=NO), this is the default.

Keyword Parameter MLIB

MLIB=NO
: Prevents dumping of the $Y$SDF file.

Keyword Parameter SPL

```
SPL= { PRINTER }
     { TAPE    }
```

where:

SPL=PRINTER
: Sends all spooled output to the printer. (This is the default.)

SPL=TAPE
: Sends all spooled output to tape. If this is not specified, then output is printed.

Keyword Parameter FDD

FDD=NO
: Turns off printing of the FDDO diskette.

BUF=NO
: Turns off printing of the dynamic buffer summary.

BUF=YES
: Turns on printing of the dynamic buffer summary. (This is the default.)

Table 3-1 is a summary of the run statement keywords.

Table 3-1. Run Statement Keyword

Keyword Parameter	Option	Suboption	Format
DO	ALL		DO=AL
		SYSTEM	DO=A(SYS)
	NONE		DO=NO
	DUMP		DO=DU
		NOSHARE	DO=DU(NOS)
		NOBUFFER	DO=DU(NOB)
		CACHE	DO=DU(CAC)
		SYSTEM	DO=DU(SYS)
		SELECT	DO=DU(SEL)
	JOBS		DO=JO
		NOSHARE	DO=JO(NOS)
		NOBUFFER	DO=JO(NOB)
		SELECT	DO=JO(SEL)
	EDIT		DO=ED
	MINI		DO=MI
	SAVE		DO=SA
	RESTORE		DO=RE
V	vsn		V=vsn
	(vsn,A)		V=(vsn[,A],[dev])
P	did		P=did

continued

Table 3-1. Run Statement Keywords (cont.)

Keyword Parameter	Option	Suboption	Format
FILE	$Y$DUMP[1]		FILE=$Y$DUMP
	lblname		FILE=lblname
FMT	F		FMT=F
	C[1]		FMT=C
PT	NO		PT=N
	YES[1]		PT=Y
MLIB	NO		MLIB=N
SPL	PRINTER[1]		SPL=PRINTER
	TAPE		SPL=TAPE
FDD	NO		FDD=N
BUF	NO		BUF=N

[1] Default

3.2.3. Supervisor-Initiated SYSDUMP

The supervisor will automatically generate a system dump in either of two situations:

- An error occurs within a user program.
- An error occurs within the supervisor itself.

These two situations are discussed in more detail in 3.2.4.

SYSDUMP from User Program Errors

You can enable the supervisor to generate a SYSDUMP if it encounters a program error or the program executes either the DUMP or CANCEL macroinstruction. In this case, the main storage write phase and execution phases are called and executed automatically, one after the other. To accomplish this, the supervisor calls and executes the SYSDMP load module in your job region within the same step in which the program error occurred.

To enable a supervisor-initiated SYSDUMP you must:

- Include a printer device assignment set with the LFD name PRNTR
- Include //ΔOPTION SYSDUMP job control statement preceding the //ΔEXEC statement of the program for which the dump is to be taken

If you want to enable the SYSDUMP option for more than one step, use the // OPTION GSYSDUMP job control statement. Inserting it in your control stream enables the SYSDUMP option for all steps following it right up to the end of the job.

If you want to restart a job while OPTION SYSDUMP is still processing, use the // OPTION PSYSDUMP job control statement. This option immediately terminates the failing job and processes the OPTION PSYSDUMP under a cover name of JOBNAME#.

If a program run with //ΔOPTION SYSDUMP fails, the supervisor will write a main storage image to $Y$DUMP, load the SYSDMP module for the execution phase, then display the following message:

```
DUMP OPTION (ALL, NONE, DUMP, TRANSLATED, JOBS, RESTORE, SAVE)
```

You may want to tell the system operator beforehand what SYSDUMP option to enter should a system dump be generated from your program. See Step 2 in 3.2.2 for an explanation of these options.

SYSDUMP from System Errors

Some system errors (errors occurring within the supervisor) may automatically initiate a SYSDUMP. The supervisor will attempt to write a main storage image to $Y$DUMP; if it succeeds, an SE15 message will be displayed on the system console, as in the following example:

```
SE15 SYSTEM ERROR 20 IN TRANS # 33-SYSDUMP WRITTEN TO DISK
```

The system will then schedule job SYSDMPxx to print the contents of $Y$DUMP. This example of message SE15 shows that a program check (error code 20) occurred in transient number 33. The SYSDUMP WRITTEN TO DISK message indicates that job SYSDMPxx has been scheduled. When you see the DUMP OPTION message, you should reply with the DUMP, MINI, or SAVE option, and send the resulting printout to your Unisys representative for analysis.

If the $Y$DUMP file is in use or locked when a system error occurs, a SE16 message will be displayed on the console showing the error and the current contents of the program status word (PSW) and registers. In this case, no main storage write will be performed. The system will, however, continue to run.

3.2.4. SYSDUMP SAVE and RESTORE Options

The end product of a SYSDUMP is the printed listing that shows the state of the operating system and user jobs at the time the system crashed. For analysis purposes you may wish to send a SYSDUMP listing to another site. The size and bulk of a printed listing, however, may make sending it a slow and expensive process. To help you avoid this problem, Unisys provides the SAVE and RESTORE options with SYSDUMP. You may use the SAVE option to copy the $Y$DUMP file to diskettes or a magnetic tape volume. With the RESTORE option you copy the data on that diskette or tape back onto the $Y$DUMP file. This means, for example, that at one site you can copy a $Y$DUMP file to a diskette (with SAVE), send only the diskette to another site, copy the diskette data to the $Y$DUMP file there (with RESTORE), and run SYSDUMPO to get the printed listing of the dump, all at reduced cost and in less time.

You can use the SAVE and RESTORE options with single and multivolume tapes and diskettes; however, there are some requirements.

- Tape
  - Requires Consolidated Data Management (CDM)
  - Requires 9-track tape
- Diskette
  - Requires Consolidated Data Management

SAVE Option

To use the SAVE option you must first prep a diskette or magnetic tape using the canned job control stream SD$PREP. When you call SYSDUMPO and get the DUMP OPTION message, you key in SAVE. The system will reply with the following console message:

```
ENTER DUMP DEVICE INFORMATION (DISKETTE, TAPE, NONE)
```

Key in the device you will use in saving the $Y$DUMP file, DISKETTE or TAPE, or key in NONE to terminate SAVE without saving the $Y$DUMP file. If you enter DISKETTE or TAPE, the system will then ask you to mount a prepped diskette or magnetic tape on an unused drive. When you have done so, the system will copy the $Y$DUMP file. After it finishes it will display:

```
$Y$DUMP SAVED ON TO { TAPE     }
                    { DISKETTE }
```

The system will then display the DUMP OPTION message so that you may select another SYSDUMPO option or terminate SYSDUMPO altogether (with NONE). In other words, when you run SYSDUMPO with the SAVE option, you will be assigning a diskette/tape in addition to supplying a brief "problem description".

You can send tapes containing saved dumps to the Customer Support Center for analysis. Please include the following on the label of the tape:

- Whether or not the system that created the tape was generated for tape block numbering
- Streaming or non-streaming type of the tape drive
- Your company name
- Applicable reference number
  - UCF number (supplied by Unisys)
  - Authorization number (supplied by Unisys)
- Date
- Number multivolume tapes giving the number of the tapes and the total number of tapes (for example: 1 of 3, 2 of 3, 3 of 3)
- The dump name (please note on UCF form in comments area), for example, HENRC001

Note: *The SYSDUMP will print the SMC LIST and error log. Include these listings with the tape or tapes.*

RESTORE Option

You use the RESTORE option much like the SAVE option. In response to the SYSDUMPO DUMP OPTION message you key in RESTORE. The system will reply with the following console message:

```
ENTER DUMP DEVICE INFORMATION (DISKETTE, TAPE, NONE)
```

Key in the device type that contains the saved $Y$DUMP file, DISKETTE or TAPE, or key in NONE to terminate RESTORE without restoring the $Y$DUMP file. If you enter DISKETTE or TAPE, the system will then ask you to mount the magnetic tape or diskette. When you have done so, the system will copy the data to the $Y$DUMP on the SYSRES volume, destroying any data previously stored there. After it finishes, it will display:

```
$Y$DUMP RESTORED FROM {TAPE    }
                      {DISKETTE}
```

The system will then display the DUMP OPTION message so that you may select another SYSDUMPO option or terminate SYSDUMPO altogether (with NONE). You can now have the system print the SYSDUMP listing in any of its available formats.

Note: *When the SYSDUMP main storage write and execution phases take place in the same system, the main storage map will include a list of the CSECTs in main storage. If you run SYSDUMPO on a system other than the one whose main storage image is in the $Y$DUMP file, you will not get the CSECT listing. All other parts of the SYSDUMP listing, however, will be printed.*

Prepping a Diskette or Magnetic Tape for SAVE/RESTORE

Before you can use a diskette or magnetic tape with the SAVE or RESTORE options, you must prep it using the canned job control stream SD$PREP. The control stream accepts a diskette or magnetic tape straight from the factory, formats it if necessary, and allocates a single-volume file called $Y$DMP, the file which the SAVE and RESTORE options use.

Note: *If you send a dump on tape to the Customer Support Center, make sure another tape is prepped and ready for future dumps.*

To call SD$PREP, key in:

```
RV SD$PREP,,TYPE={TAPE    },NUMBER={1 },VOLUME={1 },SIDES={1}
                 {DISKETTE}        {nn}        {vv}       {2}
```

where:

```
TYPE={TAPE    }
     {DISKETTE}
```

Specifies the type of device to be prepped.

```
NUMBER={1 }
       {nn}
```

nn specifies 01 to 16 tape or diskette volumes.

```
VOLUME= { 1  }
        { vv }
```

vv specifies the first volume to be prepped.

```
SIDES= { 1 }
       { 2 }
```

Specifies a 1- or 2-sided diskette.

The system will ask you to mount the diskette or magnetic tape to be prepped. When you have done so, the system preps it.

These are a few things to remember when using SD$PREP:

1. Prepping a magnetic tape or diskette for SYSDUMPO destroys all data previously recorded on that medium.

2. A diskette or magnetic tape prepped by SD$PREP cannot be used for other files or programs. To use a magnetic tape or diskette (for other files) that has already been prepped with SD$PREP, you must prep it again using the appropriate system service program described in the *System Service Programs (SSP) Operating Guide* (UP-8841).

3. Tapes or diskettes prepped prior to Release 8 are no longer valid.

3.3. Job Dump Routine (JOBDUMP)

You use the job dump routines JOBDUMP and ABRDUMP to determine what caused your job to terminate abnormally.

3.3.1. Full Job Dump (OPTION JOBDUMP)

JOBDUMP provides a method for determining what caused the job to terminate abnormally. It prints out a listing of the state of the job region when the job crashed. JOBDUMP is made up of two parts:

- A dump that translates the state of the job region into charts and text

- A labeled hexadecimal/character main storage dump

To execute JOBDUMP, a // OPTION JOBDUMP job control statement is required and must precede the EXEC job control statement. The first statement invokes JOBDUMP if the job terminates abnormally or upon execution of a DUMP or CANCEL macroinstruction in the assembler program.

There must also be a printer device assignment present in the control stream with a LFD file name of PRNTR.

3.3.2. Abbreviated Job Dump (OPTION ABRDUMP)

The abbreviated job dump provides you with a shortened listing of the full job dump to help you determine what caused the job to terminate abnormally. When ABRDUMP is called, only the area in the vicinity of the last instruction executed, along with the address and contents of the I/O buffers associated with the OPEN DTFs, is printed.

To execute ABRDUMP, a // OPTION ABRDUMP job control statement is used in place of the // OPTION JOBDUMP job control statement. The abbreviated job dump is called similarly to the full job dump.

3.4. EOJ Dump Routine

The EOJ dump routine is called by either the DUMP macro used in place of the EOJ macro in your assembler program or by an abnormal termination of your job. In either case, the // OPTION DUMP job control statement must be present in your control stream. The EOJ is in hexadecimal format and is divided into four sections: problem program registers, job preamble, task control blocks (TCBs), and your program region. In addition, the dump also gives you the PSW at interrupt time, the error code that caused the abnormal termination, and the next TCB address.

3.5. Supervisor Trace Analysis

The supervisor trace facility is automatically included in any system dump, whether operator- or supervisor-initiated; you don't need to activate it.

The module for the supervisor trace facility is SM$TRACE.

3.5.1. How to Read a Supervisor Trace

1. Find the module heading SM$TRACE in the system dump printout. It should be located near the end of the resident supervisor portion of the dump.

2. Find the word TRAC in the interpreted (edited) section on the right side of the system dump printout.

3. Now find the equivalent of TRAC in the hexadecimal portion of the printout. This hex value marks the beginning of the trace table where the supervisor activities are logged.

   Note that the entry for each supervisor activity consists of four hexadecimal words. The supervisor trace table can contain up to 400 entries and will wrap (overwrite the beginning of the table) when the end of the table is reached, so that only the 400 most current entries are listed.

4. Find the word following the hex equivalent of TRAC. This word contains the address of the next available entry in the table (i.e., where the next entry will be logged).

5. Look for this address in the hex portion of the printout. Note the entry in *front* of this address. This entry is the last (most current) one logged; it is the logical end of the supervisor trace table.

Table 3-2 describes the supervisor area trace table entry formats. Note that word 1 for each entry appears in the interpreted portion at the right of the dump printout.

Table 3-2 also provides the interpretations of the hex values of words 2, 3, and 4 for each supervisor area. Note that all zeros indicate that there are no values (no entries to be made) for that word. In some cases, words 2 and 3 are combined to provide one value.

Table 3-2. Supervisor Area Trace Table Entry Formats

Supervisor Area	Word 1	Word 2	Word 3	Word 4
Timer	TI	00000000	00000000	Interval timer register
Machine check	MC	Machine check old program status word (PSW)		Interval timer register
Program check	PC	Program check old PSW		Interval timer register
Transient overlay	TO	Transient area address	Overlay transient id	Interval timer register
Task switch	TS	00000000	TCB address of new task	Interval timer register
SVC call	SV	SVC old PSW		Interval timer register
Transient release	TR	Transient area address	Transient id	Interval timer register
Shared code call	SC	TCB address of task using shared code	Name of shared code module being called	Interval timer register
Shared code release	SR	TCB address of task using shared code	Name of shared code module being exited	Interval timer register

Section 4
Sample Dump Analyses

4.1. General

In the preceding sections, you got a general overview of OS/3 dumps and of how OS/3 interacts with user programs. In this section, you will see examples of dump analyses applied to actual user programs by using the ideas already presented. A number of programs are shown here that have failed while being executed. Although all have been compiled or assembled correctly, errors have nonetheless remained. It is in uncovering these errors that dump analysis is a useful tool.

Three programs are presented in this section, each written in a language supported by OS/3:

- BALOBJ, a program that reads data from cards, adds the data, and outputs the result to a printer, is written in Basic Assembler Language (BAL).

- COBOBJ, a program that performs the same functions as BALOBJ, is written in COBOL.

- RPGOBJ, a program that performs the same functions as BALOBJ, is written in RPG II.

In these examples, you will be working with different types of dumps, as well as with other diagnostic tools supplied with the individual language processors. Each example takes the following form:

- An outline of the program and the circumstances under which it failed

- A list of the dumps and other materials used in the analysis

- A brief outline of the particular type of dump used or of an analysis technique

- A step-by-step dump analysis narrative

- An edited copy of the output listings used in the analysis

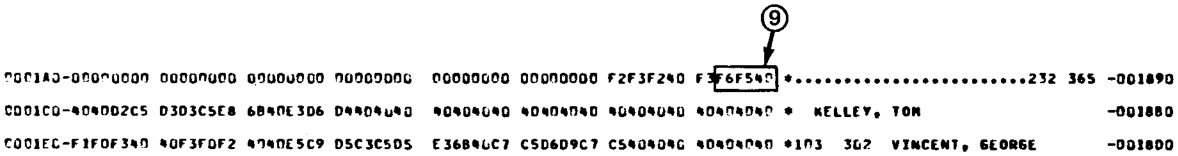
The dump analysis itself will consist of two parts; the narrative and the listings. Most important among these listings is the dump itself, but compiler/assembler listings and other materials are also included where needed.

In a typical dump analysis, you will move forward and backward within and often outside the dump. With analysis, you are trying to uncover the logical (and not necessarily the physical) sequence of events leading up to a program failure.

To help you find your way through the following analyses, they are broken down into logical steps numbered ①, ②, ③, etc. With each step, a relevant portion of the listings is reproduced within which an identically numbered pointer indicates the data used in that step. All the pointers reappear in the listings themselves, together with an index indicating where each pointer is.

As an example, we show two excerpts from a dump analysis, one taken from the narrative and the other from the output listings.

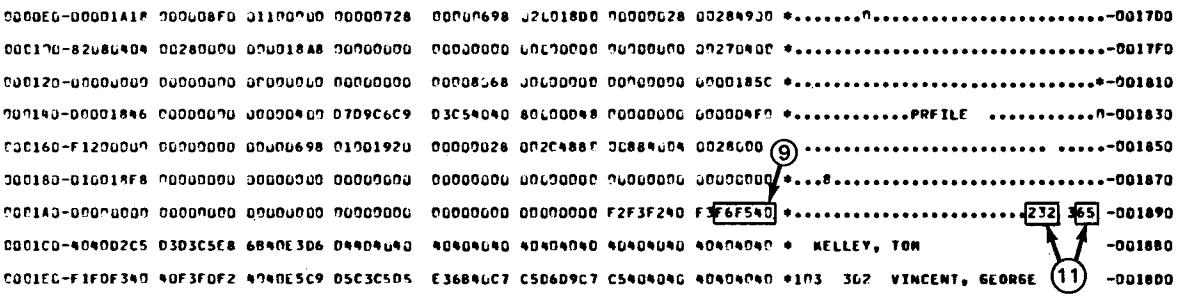
Dump Analysis Narrative



```
0001A0-00000000 00000000 00000000 00000000  00000000 00000000 F2F3F240 F3F6F540 *........................232 365 -001890
0001C0-4040D2C5 D3D3C5E8 6B40E3D6 D4404040  40404040 40404040 40404040 40404040 *  KELLEY, TOM                    -0018B0
0001E0-F1F0F340 40F3F0F2 4040E5C9 D5C3C5D5  E36B40C7 C5D6D9C7 C5404040 40404040 *103  302  VINCENT, GEORGE        -0018D0
```

At location 18AD ⑨ we see that the three bytes operated on by the PACK instruction...

Dump Analysis Output Listing



```
0000E0-00001A18 000008F0 01100700 00000728  00000698 02C018D0 00000028 00284930 *.......n........................-0017D0
000100-82080404 00280000 000018A8 00000000  00000000 00C00000 00000000 00270400 *................................-0017F0
000120-00000000 00000000 0C0000C0 00000000  00008568 00C00000 00000000 0000185C *...............................*-001810
000140-00001846 C0000070 00000400 D7D9C6C9  D3C54040 80C00048 C0000000 000004F0 *.............PRFILE  ...........n-001830
000160-F1200000 00000000 00000698 01001920  00000028 002C488C 0C884004 002800    .......................  .....-001850
000180-010018F8 00000000 00000000 00000000  00000000 00C00000 00000000 00000000 *...8............................-001870
0001A0-00000000 00000000 00000000 00000000  00000000 00000000 F2F3F240 F3F6F540 *........................232 365 -001890
0001C0-4040D2C5 D3D3C5E8 6B40E3D6 D4404040  40404040 40404040 40404040 40404040 *  KELLEY, TOM                    -0018B0
0001E0-F1F0F340 40F3F0F2 4040E5C9 D5C3C5D5  E36B40C7 C5D6D9C7 C5404040 40404040 *103  302  VINCENT, GEORGE        -0018D0
```

These analyses assume that you have a knowledge of machine language, especially main storage addressing. If you need to learn about how machine instructions in general are executed, refer to the appropriate processor programmer reference. To learn what an individual machine instruction does, refer to the *Hardware and Software Programming Quick-Reference Guide* (UP-8868).

4.2. BAL Dump Analysis

In this subsection, we will analyze an EOJ dump generated from a BAL program called BALOBJ. We will use some of the TCB fields discussed earlier to help uncover the error causing the dump.

Program BALOBJ, run as part of a job named BALJOB, reads numeric data from an input card, adds the numbers, and outputs them to a printer together with a copy of a character string punched on the input card. The program as written is assembled without any errors, but when we run it, it fails, generating an error code of 20 - a program exception. Anticipating the possibility of such an error, we have put an // OPTION DUMP card in the JCL runstream immediately preceding the EXEC instruction that loads and runs BALOBJ. The program exception causes the operating system to dump the job area containing BALOBJ; because the EOJ dump goes directly to the user's log file, there is no need to assign a printer exclusively to the dump.

4.2.1. Materials Used

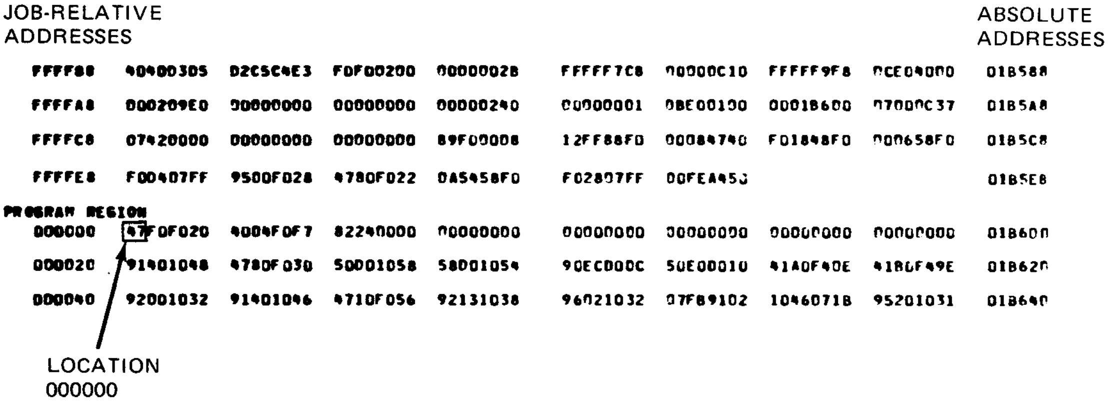
The materials we will use in the analysis are contained in 4.2.4. They include the following as shown in Figure 4-2:

- The BAL source code for BALOBJ
- The linkage editor allocation map
- The EOJ dump, edited for clarity

4.2.2. Outline of EOJ Dump

We will use the step-pointer system outlined in 4.1 to help guide our way around the output listings shown in 4.2.4. An important part of using this system is knowing how the EOJ dump is organized. You should compare the examples and chart presented here with the actual dump shown in 4.2.4.

A portion of a typical EOJ dump looks like this:



```
JOB-RELATIVE                                                                                          ABSOLUTE
ADDRESSES                                                                                             ADDRESSES
  FFFF80   40400305  D2C5C4E3  F0F00200  0000002B    FFFFF7C0  00000C10  FFFFF9F8  0CE04000   01B580
  FFFFA0   000209E0  00000000  00000000  00000240    00000001  0BE00100  00018600  07000C37   01B5A0
  FFFFC0   07420000  00000000  00000000  89F00008    12FF88F0  00084740  F01848F0  000658F0   01B5C0
  FFFFE0   F00407FF  9500F028  4780F022  0A5458F0    F02807FF  00FEA453                       01B5E0
PROGRAM REGION
  000000   47F0F020  4004F0F7  82240000  00000000    00000000  00000000  00000000  00000000   01B600
  000020   91401048  4780F030  50001058  58001054    90ECD00C  50E00010  41A0F40E  41B0F49E   01B620
  000040   92001032  91401046  4710F056  92131038    96021032  07F89102  1046071B  95201031   01B640
```

As you can see, each line of the dump contains 32 bytes of data. The location of that data is given by the two addresses flanking each line. Both addresses always refer to the hexadecimal location of the first byte in the line but differ in their addressing: the right-hand address is the address of the location relative to the entire system, while the left-hand address is relative to the program region. Since the job prologue comes before the program region, its left-hand addresses are given as negative (twos complement) values.

The characteristics of the EOJ dump are summarized in Figure 4-1.

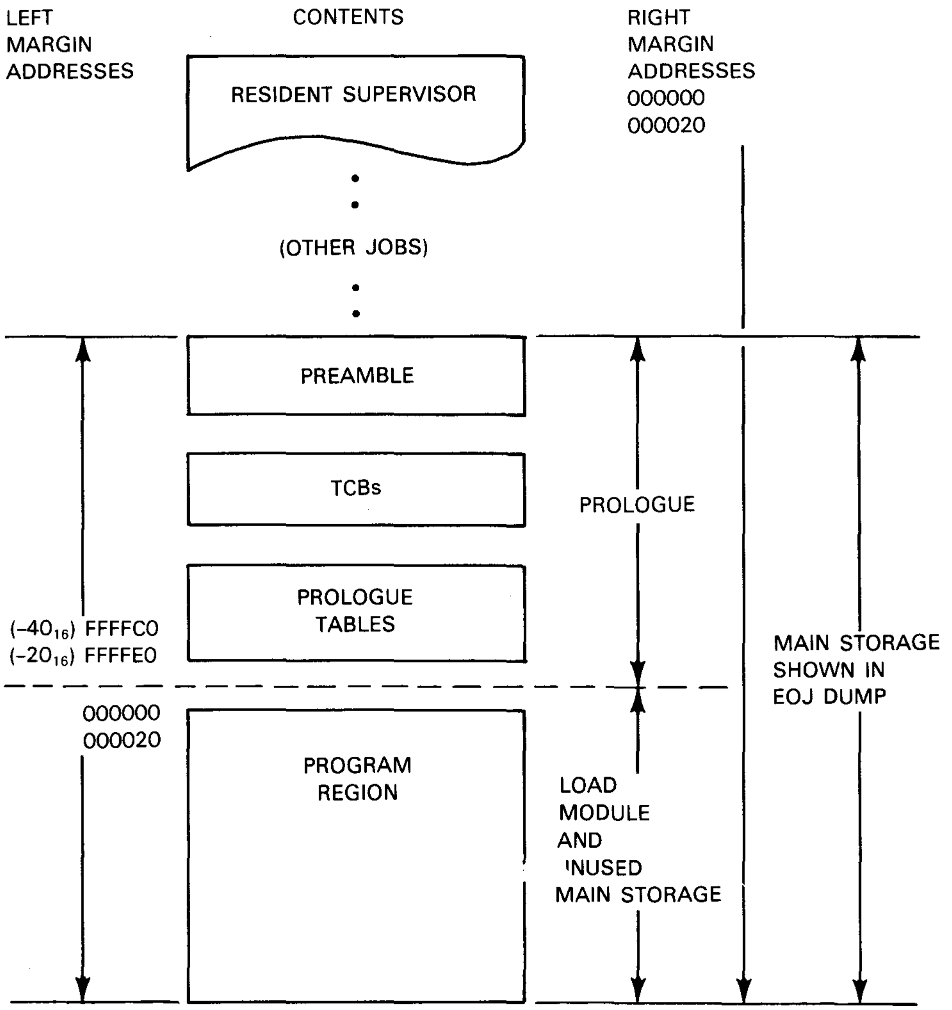


Figure 4-1. EOJ Dump Organization

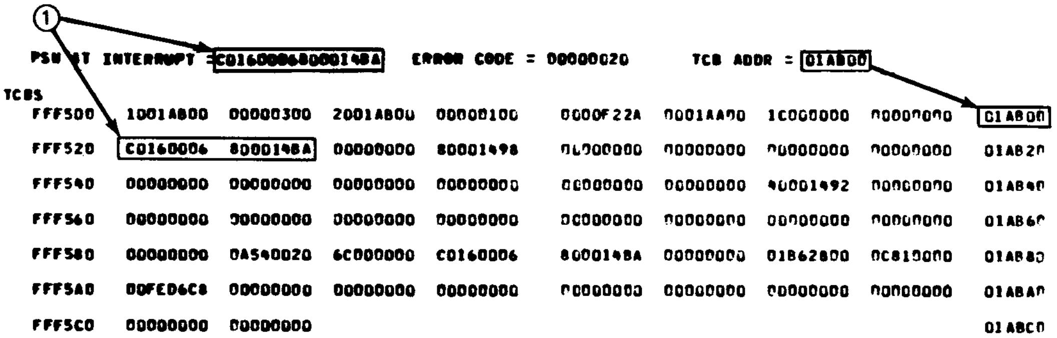
As Figure 4-1 shows, the EOJ dump presents the entire job region, broken up into a number of blocks: the preamble, each TCB, other prologue tables, and the program region. To read and analyze an EOJ dump, you will need to know about the structure of the TCB, discussed earlier in this manual. As we go through the following analysis you may want to review 2.2 for the structure of the TCB.

Not shown in Figure 4-1 are several other items that are present at the beginning of the EOJ dump:

- Job name
- System version
- PSW at the time the program failed
- Error code
- Address of the TCB that was active when the program failed
- Contents of problem registers 0 through 15 when the program failed

4.2.3. Analysis

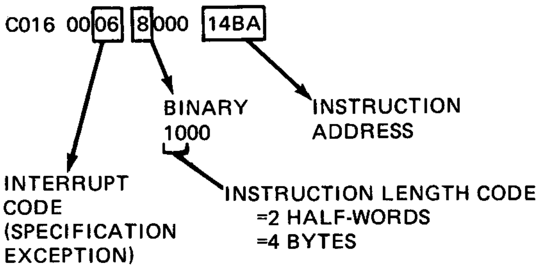
To determine what caused program BALOBJ to fail, we proceed as follows:



```
 PSW AT INTERRUPT =C0160006800014BA   ERROR CODE = 00000020      TCB ADDR = 01AB00
TCBS
  FFF500   1001AB00 00000300 2001AB00 00000100    0000F22A 0001AA00 1C000000 00000000   01AB00
  FFF520   C0160006 8000148A 00000000 80001498    00000000 00000000 00000000 00000000   01AB20
  FFF540   00000000 00000000 00000000 00000000    00000000 00000000 40001492 00000000   01AB40
  FFF560   00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   01AB60
  FFF580   00000000 0A540020 6C000000 C0160006    8000148A 00000000 01B62800 0C810000   01AB80
  FFF5A0   00FED6C8 00000000 00000000 00000000    00000000 00000000 00000000 00000000   01ABA0
  FFF5C0   00000000 00000000                                                            01ABC0
```

We begin by looking for the instruction that directly caused the failure of the program. The TCB at location $1AB00_{16}$ is the primary TCB for the job; therefore, its PSW save area (offset 20_{16}-27_{16}) ① holds the PSW at the time of the error.

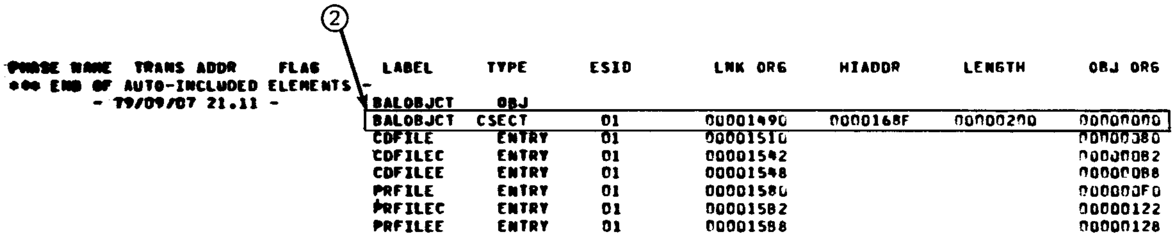
Let's look more closely at the task PSW. Using the PSW illustration in 2.1.1, we can extract the following information:



From this PSW, we can determine where the error occurred and what caused it. We find the error location by remembering that all interrupt handlers store the address of the next instruction to be processed in bits 48-63 of the PSW. The length of the instruction being processed at interrupt time is stored in bits 32-33. From this information, we can find the address of the instruction causing the interrupt:

14BA	PSW instruction address
- 4	ILC (= 2 half words)
14B6	Address of instruction causing interrupt

As to what caused the interrupt, we can determine that from bits 24-31, the interrupt code. In our case, the code is 06, which, according to Table A-1, is a specification exception involving the improper use of the instruction at 14B6.

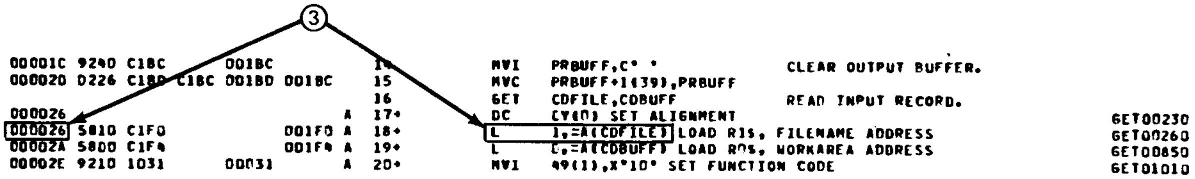


```
PHASE NAME  TRANS ADDR    FLAG    LABEL     TYPE     ESID      LNK ORG     HIADDR     LENGTH      OBJ ORG
*** END OF AUTO-INCLUDED ELEMENTS -
       - 79/09/07 21.11 -         BALOBJCT   OBJ
                                  BALOBJCT  CSECT     01      00001490    0000168F   00000200    00000000
                                  CDFILE     ENTRY    01      00001510                           00000080
                                  CDFILEC    ENTRY    01      00001542                           000000B2
                                  CDFILEE    ENTRY    01      00001548                           000000B8
                                  PRFILE     ENTRY    01      0000158C                           000000FC
                                  PRFILEC    ENTRY    01      000015B2                           00000122
                                  PRFILEE    ENTRY    01      000015B8                           00000128
```

To use the information we have so far, we will need to find the instruction that caused BALOBJ to fail. Keep in mind that a load module like the one loaded and run in this job can comprise one or more object modules like BALOBJ, bound together by the linkage editor. We can use the linkage editor allocation map to determine in what portion of the load module our failing instruction lies.

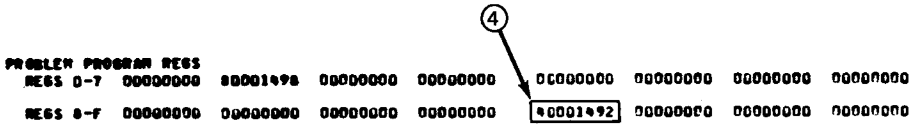
Pointer ② indicates an entry on the allocation map. Three items of the entry are of interest to us: the label BALOBJ, the LNKORG address 1490, and the HIADDR address 168F. The LNKORG and HIADDR addresses define the boundaries of the object module named BALOBJ and we see that address 14B6, indicated by ① , lies within BALOBJ. The exact location of the address can be determined as follows:

14B6	Address of failing instruction in load module
-1490	LNKORG address (first address of BALOBJ)
26	Offset of failing instruction within BALOBJ



```
00001C 9240 C1BC       001BC              14        MVI    PRBUFF,C' '              CLEAR OUTPUT BUFFER.
000020 D226 C1BD C1BC 001BD 001BC         15        MVC    PRBUFF+1(39),PRBUFF
                                          16        GET    CDFILE,CDBUFF            READ INPUT RECORD.
000026                                A   17+       DC     CY(0) SET ALIGNMENT                                      GET00230
000026 5810 C1F0                001F0 A   18+       L      1,=A(CDFILE) LOAD R1S, FILENAME ADDRESS                  GET00260
00002A 5800 C1F4                001F4 A   19+       L      0,=A(CDBUFF) LOAD R0S, WORKAREA ADDRESS                  GET00850
00002E 9210 1031       00031          A   20+       MVI    49(1),X'10' SET FUNCTION CODE                            GET01010
```

We now know that the failing instruction we seek does lie within our BAL program, rather than within one of the other object modules making up our load module. We turn to the assembler listing for program BALOBJ and look at offset 26_{16} ③, the instruction location calculated in ② . We find there that our failing instruction is one that LOADs register 1 from a storage location determined by base register 12 (C_{16}) and offset $1F0_{16}$. Looking at Table A-1 for possible causes of our error, we see that the most likely explanation is that the LOAD does not refer to a proper boundary. LOAD instructions can operate only on data residing on full-word boundaries; might the main storage location given by ③ not satisfy this requirement? To see, we go to ④ , located at the beginning of the EOJ dump.



```
PROBLEM PROGRAM REGS
  REGS 0-7  00000000  80001498  00000000  00000000   00000000  00000000  00000000  00000000

  REGS 8-F  00000000  00000000  00000000  00000000   40001492  00000000  00000000  00000000
```

To find the storage location referred to by the LOAD instruction in ③, we must first find what is in base register 12 ④. Ignoring the high-order byte, which plays no part in address formation, we see that register 12 contains the value 1492. Once again, the instruction in ③ is:

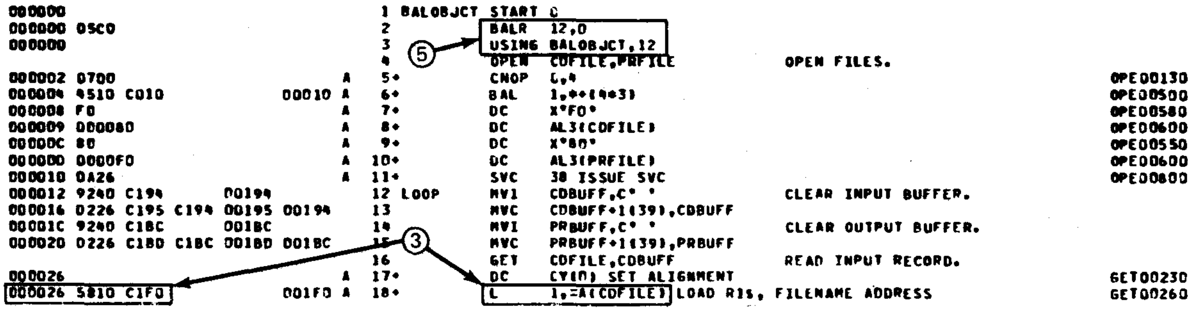
```
5810        C1F0
```

The storage location is determined as follows:

```
 1492   Register 12 contents
+ 1F0   Displacement
 1682   Referenced main storage location
```

Location 1682 does not lie on a full-word boundary; therefore, any LOAD instruction referencing it will generate a program exception like the one that has occurred in this program.

One question remains: why does the main storage location fail to lie on a full-word boundary? As ③ shows, the location is represented by an address constant (for CDFILE) and address constants are assembled on full-word boundaries. For more clues, we will have to look backwards in the program to see what events could have disrupted it.



```
000000                               1 BALOBJCT START C
000000 05C0                          2          BALR  12,0
000000                               3          USING BALOBJCT,12
                                     4          OPEN  CDFILE,PRFILE           OPEN FILES.
000002 0700                        A  5+         CNOP  C,4                                              OPE00130
000004 4510 C010             00010 A  6+         BAL   1,*+(4*3)                                        OPE00500
000008 F0                          A  7+         DC    X'F0'                                            OPE00580
000009 000080                      A  8+         DC    AL3(CDFILE)                                      OPE00600
00000C 80                          A  9+         DC    X'80'                                            OPE00550
00000D 0000F0                      A 10+         DC    AL3(PRFILE)                                      OPE00600
000010 0A26                        A 11+         SVC   38 ISSUE SVC                                     OPE00800
000012 9240 C194      00194          12 LOOP     MVI   CDBUFF,C' '            CLEAR INPUT BUFFER.
000016 0226 C195 C194 00195 00194    13          MVC   CDBUFF+1(39),CDBUFF
00001C 9240 C1BC      001BC          14          MVI   PRBUFF,C' '            CLEAR OUTPUT BUFFER.
000020 0226 C1BD C1BC 001BD 001BC    15          MVC   PRBUFF+1(39),PRBUFF
                                     16          GET   CDFILE,CDBUFF          READ INPUT RECORD.
000026                             A 17+         DC    CY(0) SET ALIGNMENT                              GET00230
000026 5810 C1F0             001F0 A 18+         L     1,=A(CDFILE) LOAD R1S, FILENAME ADDRESS          GET00260
```

Looking at ③ again, we see that the displacement $1F0_{16}$ is evenly divisible by 4. For a full-word location, the value contained in register 12 must also be evenly divisible by 4. As ④ shows us, however, the register 12 value of 1492 is not. So we look at previous instructions affecting register 12 and find at ⑤ a BALR instruction and a USING directive.

The USING directive marks the location of symbol BALOBJ as the base address for this assembly and names register 12 as the base register. Since BALOBJ occupies location 00 (1490 when linked into the load module), all references to main storage use register 12, presuming it has a value of 0. However, the previous instruction at ⑤, BALR 12,0 effectively loads a value of 2 (in fact, a load module-relative address of 1492) into register 12. Since the BALR instruction takes effect only at execution time, it introduces a 2-byte address offset that was never taken into account by the assembler. This unexpected offset causes the LOAD instruction in ③ to try to access a main storage location that is not on a full-word boundary. The result is the specification exception that caused BALOBJ to fail.

The solution to the problem is to rearrange or rewrite those introductory BAL statements in ⑤. One correct sequence is:

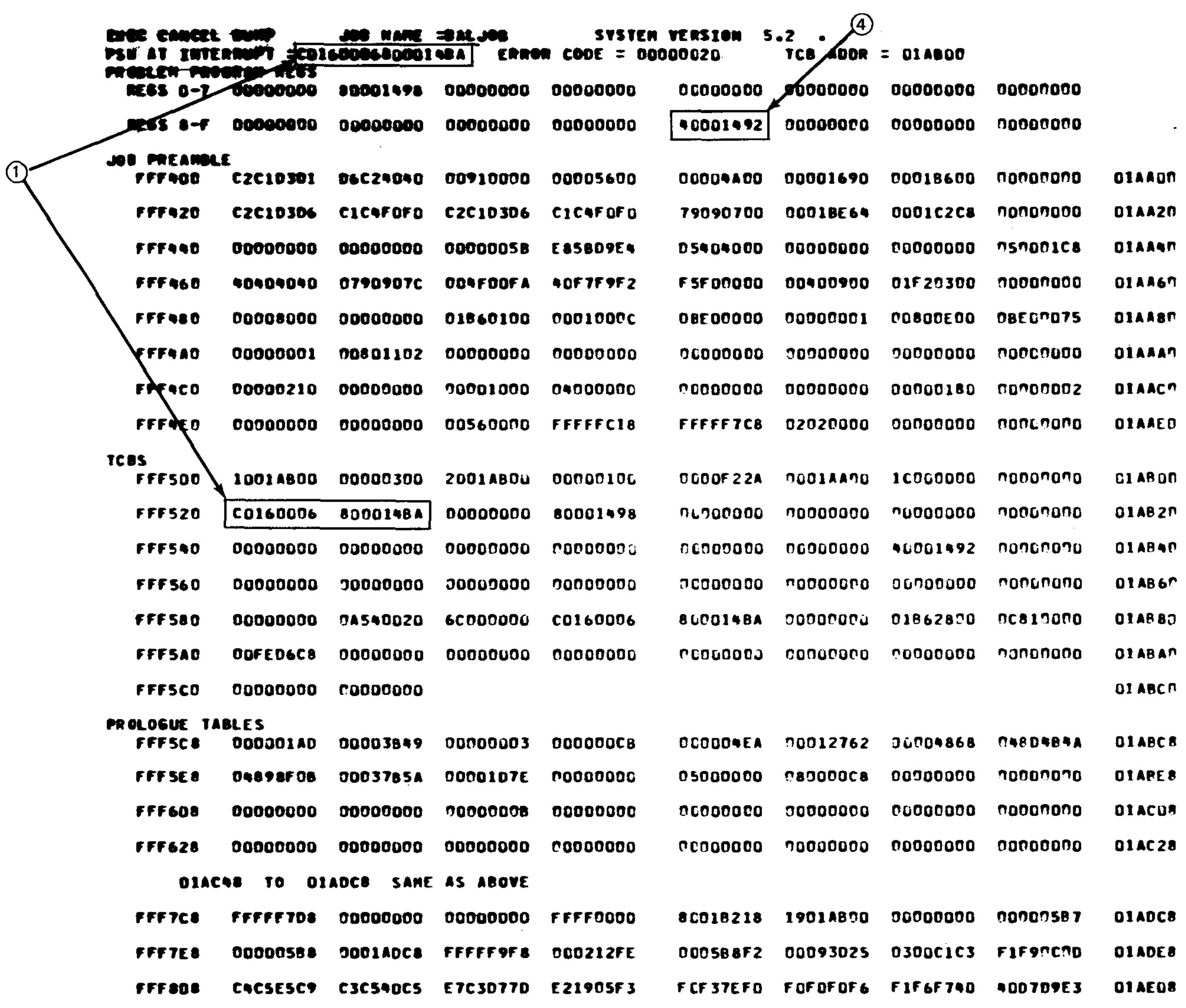
```
BALOBJ  START 0
        BALR  12,0
        USING *,12
          .
          .
          .
```

We should note that the statements at ③ and ⑤ were assembled without error. It was only after they were executed that a problem emerged. We should also note by looking at Figure 4-2 that the steps we take to analyze BALOBJ follow a logical, not a physical, sequence - step ⑤ lies above ③, for example. It is the time sequence that we are most concerned with, and the EOJ dump can be helpful to us in that respect.

4.2.4. Dump Analysis Materials

The edited printout from job BALJOB, including the assembler listing for program BALOBJ, the link edit, and the EOJ dump are contained in Figure 4-2. All the pointers referred to in 4.2.3 are shown in the list below in numerical order, each pointer referring to the part of Figure 4-2 on which it appears. Use this list to assist you in looking for these pointers.

①	part 1
②	part 7
③	part 5
④	part 1
⑤	part 5

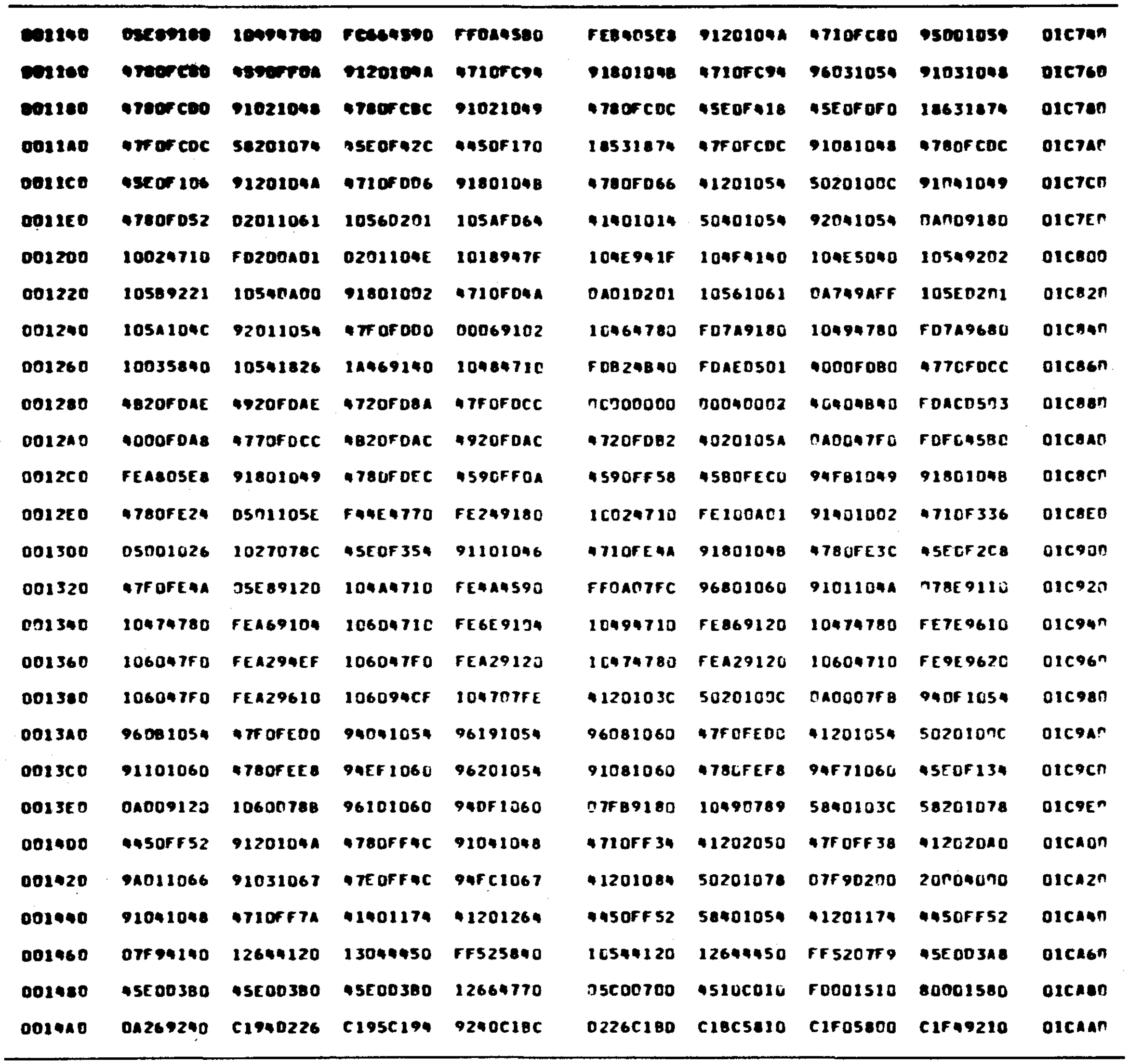


```
EXEC CANCEL DUMP        JOB NAME =BALJOB          SYSTEM VERSION  5.2  .
PSW AT INTERRUPT =C0160006800014BA   ERROR CODE = 00000020       TCB ADDR = 01AB00
PROBLEM PROGRAM REGS
  REGS 0-7  00000000  80001498  00000000  00000000   00000000  00000000  00000000  00000000

  REGS 8-F  00000000  00000000  00000000  00000000   40001492  00000000  00000000  00000000

JOB PREAMBLE
   FFF400   C2C1D3D1  D6C24040  00910000  00005600   00004A00  00001690  00018600  00000000   01AA00

   FFF420   C2C1D3D6  C1C4F0F0  C2C1D3D6  C1C4F0F0   79090700  0001BE64  0001C2C8  00000000   01AA20

   FFF440   00000000  00000000  0000005B  E85BD9E4   D5404000  00000000  00000000  050001C8   01AA40

   FFF460   40404040  0790907C  004F00FA  40F7F9F2   F5F00000  00400900  01F20300  00000000   01AA60

   FFF480   00008000  00000000  01860100  0001000C   0BE00000  00000001  00800E00  0BE00075   01AA80

   FFF4A0   00000001  00801102  00000000  00000000   00000000  00000000  00000000  00000000   01AAA0

   FFF4C0   00000210  00000000  00001000  04000000   00000000  00000000  00000180  00000002   01AAC0

   FFF4E0   00000000  00000000  00560000  FFFFFC18   FFFFF7C8  02020000  00000000  00000000   01AAE0

TCBS
   FFF500   1001AB00  00000300  2001AB00  0000010C   0000F22A  0001AA00  1C000000  00000000   01AB00

   FFF520   C0160006  800014BA  00000000  80001498   00000000  00000000  00000000  00000000   01AB20

   FFF540   00000000  00000000  00000000  00000000   00000000  00000000  40001492  00000000   01AB40

   FFF560   00000000  00000000  00000000  00000000   00000000  00000000  00000000  00000000   01AB60

   FFF580   00000000  0A540020  6C000000  C0160006   800014BA  00000000  01862800  0C810000   01AB80

   FFF5A0   00FED6C8  00000000  00000000  00000000   00000000  00000000  00000000  00000000   01ABA0

   FFF5C0   00000000  00000000                                                                01ABC0
PROLOGUE TABLES
   FFF5C8   000001AD  00003849  00000003  000000C8   000004EA  00012762  00004868  048D484A   01ABC8

   FFF5E8   04898F08  0003785A  00001D7E  00000000   05000000  080000C8  00000000  00000000   01ABE8

   FFF608   00000000  00000000  00000008  00000000   00000000  00000000  00000000  00000000   01AC08

   FFF628   00000000  00000000  00000000  00000000   00000000  00000000  00000000  00000000   01AC28

       01AC48   TO   01ADC8   SAME AS ABOVE

   FFF7C8   FFFFF7D8  00000000  00000000  FFFF0000   8C018218  1901AB00  00000000  000005B7   01ADC8

   FFF7E8   000005B8  0001ADC8  FFFFF9F8  000212FE   0005B8F2  00093025  0300C1C3  F1F9F0C0   01ADE8

   FFF808   C4C5E5C9  C3C540C5  E7C3D77D  E21905F3   FCF37EFD  F0F0F0F6  F1F6F740  400709E3   01AE08
```

Figure 4-2. BAL Dump Analysis Listing (Part 1 of 7)

```
   FFFFE8   F00407FF   9500F028   4780F022   0A5458F0     F02807FF   00FEA45J                           01B5E0
PROGRAM REGION
   000000   47F0F020   4004F0F7   82240000   00000000     00000000   00000000   00000000   00000000     01B600
   000020   91401048   4780F030   50001058   58001054     90EC000C   50E00010   41A0F40E   41B0F49E     01B620
   000040   92001032   91401046   4710F056   92131038     96021032   07FB9102   1046071B   95201031     01B640
   000060   4780F206   91801047   4780F07C   954B1031     4780F120   954C1031   4780F11C   9180104A     01B660
   000080   4710F08E   92141038   96021032   07FB958B     10314780   F0A2958C   10314780   F0A247F0     01B680
   0000A0   F0849104   104A078B   47F0F0AC   948F104A     958B1031   4770F0BC   9610104A   12004780     01B6A0
   0000C0   F0CA5000   102847F0   F0CE9640   104A9101     10494780   F0E09118   104A47B0   F0E007FB     01B6C0
   0000E0   94F3104A   91181046   4780F0EE   05EA9140     104A4710   F1005820   10285020   D01007FB     01B6E0
   000100   92BF103C   91401047   4710F110   96021047     0A009118   10460776   05EA07FB   96101048     01B700
   000120   948F1047   18229118   10464780   F13005EA     95000016   4780F146   43200016   4570F18E     01B720
   000140   0A009640   10479540   00174780   F30E9140     10474780   F15C05EA   94BF1047   918C103C     01B740
   000160   4780F168   94F3104A   954C1031   4770F174     96101048   96201047   96201048   43200017     01B760
   000180   4570F18E   0200104E   103C47F0   F3F44130     00089110   10484780   F1C01923   4740F1C0     01B780
   0001A0   4140F1E6   91071049   4780F1B8   1A239104     10494780   F1B81A23   43224000   47F0F1C8     01B7A0
   0001C0   89200003   41202001   4220103C   91101048     4780F108   9680103C   91201C48   4710F1E8     01B7C0
   0001E0   9606103C   92021043   94CF1048   07F71109     19210929   39394149   51590969   39394179     01B7E0
   000200   51590969   39391850   58401050   4690102C     91181046   4780F21A   05EA4830   104C9174     01B800
   000220   10484710   F27E9101   104B4710   F2429110     10464710   F25E1864   18694330   600147F0     01B820
   000240   F2629548   10444770   F2525830   105847F0     F2624330   10445833   000047F0   F2624330     01B840
   000260   50011B39   4780F26A   18334930   104C4700     F27E4830   104C9602   10489680   10324030     01B860
   000280   10429117   10494770   F29E9101   1043478C     F29E4300   10254203   40009A01   10429102     01B880
   0002A0   10474780   F2AE94FD   104747F0   F28A9150     10474780   F2C0948F   10470200   103C104F     01B8A0
   0002C0   91201047   4780F202   96101047   0200103C     104E9110   10464780   F2E81A59   12334780     01B8C0
   0002E0   F2E80630   4430F408   91081046   4780F316     47F0F308   4780F300   9103103C   4710F316     01B8E0
   000300   9106103C   4710F316   5800103C   02021030     10515000   10509104   10484780   F3961822     01B900
   000320   91101046   4710F332   06404320   400047F0     F3380650   43205000   4170F396   4140F38E     01B920
```

Figure 4-2. BAL Dump Analysis Listing (Part 2 of 7)

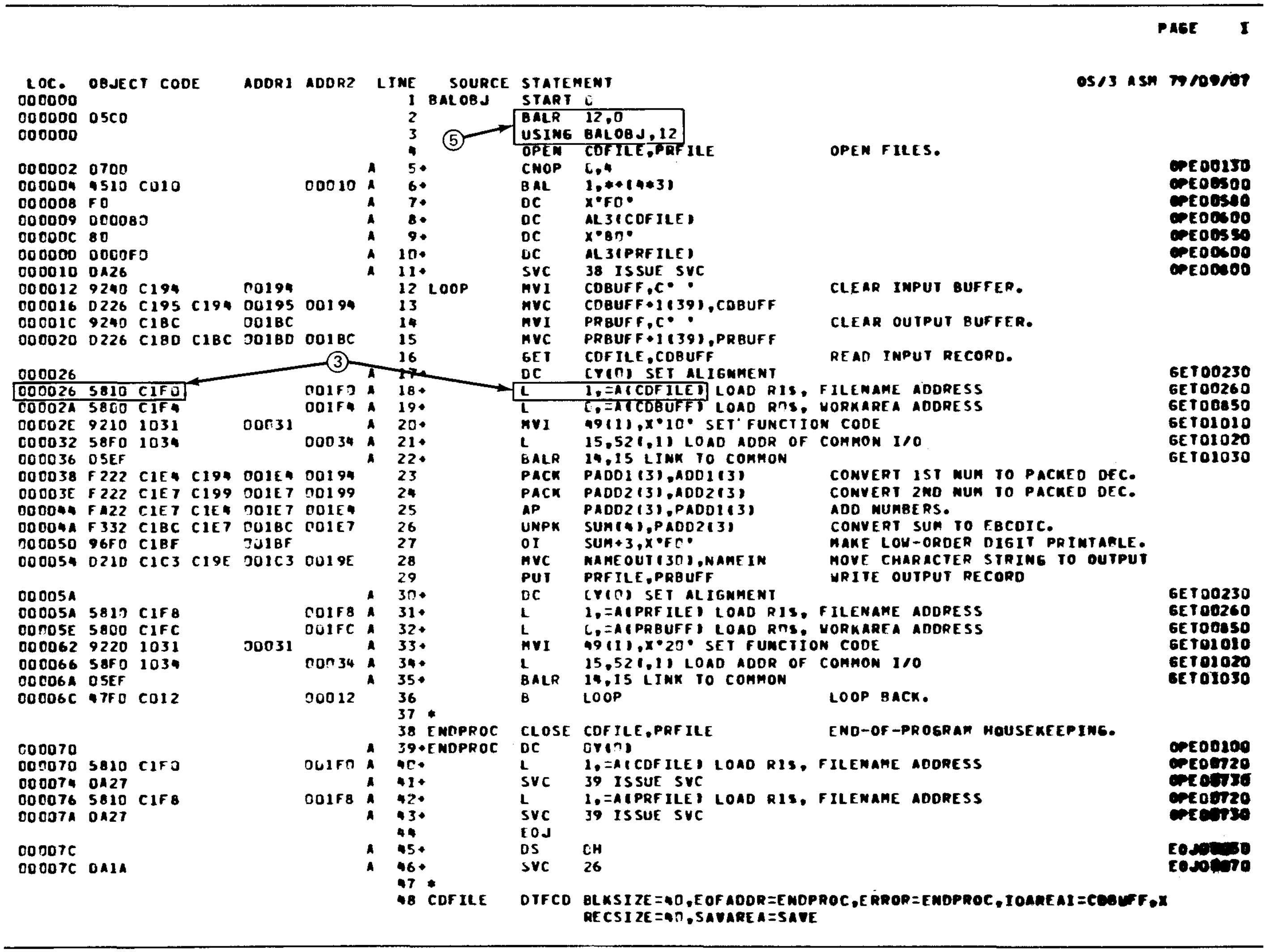


```
001140   05E89180  10494780  FC664590  FF0A4580    FE8405E8  9120104A  4710FC80  95001059   01C740
001160   4780FC80  4590FF0A  9120104A  4710FC94    9180104B  4710FC94  96031054  91031048   01C760
001180   4780FCD0  91021048  4780FCBC  91021049    4780FCDC  45E0F418  45E0F0F0  18631874   01C780
0011A0   47F0FCDC  58201074  45E0F42C  4450F170    18531874  47F0FCDC  91081048  4780FCDC   01C7A0
0011C0   45E0F106  9120104A  4710FDD6  9180104B    4780FD66  41201054  5020100C  91041049   01C7C0
0011E0   4780FD52  D2011061  1056D201  105AFD64    41401014  50401054  92041054  0A009180   01C7E0
001200   10024710  FD200A01  0201104E  1018947F    104E941F  104F4140  104E5040  10549202   01C800
001220   10589221  1054DA00  91801002  4710FD4A    0A01D201  10561061  0A749AFF  105ED201   01C820
001240   105A104C  92011054  47F0FD00  00069102    10464780  FD7A9180  10494780  FD7A9680   01C840
001260   10035840  10541826  1A469140  1048471C    FDB24B40  FDAED501  4000FDB0  4770FDCC   01C860
001280   4820FDAE  4920FDAE  4720FD8A  47F0FDCC    0C000000  00040002  40404B40  FDACD503   01C880
0012A0   4000FDA8  4770FDCC  4820FDAC  4920FDAC    4720FDB2  4020105A  0A0047F0  FDF64580   01C8A0
0012C0   FEA805E8  91801049  4780FDEC  4590FF0A    4590FF58  4580FEC0  94FB1049  9180104B   01C8C0
0012E0   4780FE24  D501105E  F44E4770  FE249180    10024710  FE100A01  91401002  4710F336   01C8E0
001300   05001026  1027078C  45E0F354  91101046    4710FE4A  9180104B  4780FE3C  45E0F2C8   01C900
001320   47F0FE4A  05E89120  104A4710  FE4A4590    FF0A07FC  96801060  9101104A  078E9110   01C920
001340   10474780  FEA69104  1060471C  FE6E9104    10494710  FE869120  10474780  FE7E9610   01C940
001360   106047F0  FEA294EF  106047F0  FEA29120    10474780  FEA29120  10604710  FE9E9620   01C960
001380   106047F0  FEA29610  106094CF  104707FE    4120103C  5020100C  0A0007FB  940F1054   01C980
0013A0   960B1054  47F0FE00  94041054  96191054    96081060  47F0FEDC  41201054  5020100C   01C9A0
0013C0   91101060  4780FEE8  94EF1060  96201054    91081060  4780FEF8  94F71060  45E0F134   01C9C0
0013E0   0A009120  1060078B  96101060  940F1060    07FB9180  10490789  5840103C  58201078   01C9E0
001400   4450FF52  9120104A  4780FF4C  91041048    4710FF34  41202050  47F0FF38  412020A0   01CA00
001420   9A011066  91031067  47E0FF4C  94FC1067    41201084  50201078  07F90200  20004000   01CA20
001440   91041048  4710FF7A  41401174  41201264    4450FF52  58401054  41201174  4450FF52   01CA40
001460   07F94140  12644120  13044450  FF525840    10544120  12644450  FF5207F9  45E003A8   01CA60
001480   45E003B0  45E003B0  45E003B0  12664770    05C00700  4510C010  F0001510  80001580   01CA80
0014A0   0A269240  C194D226  C195C194  9240C18C    D226C18D  C18C5810  C1F05800  C1F49210   01CAA0
```

Figure 4-2. BAL Dump Analysis Listing (Part 3 of 7)

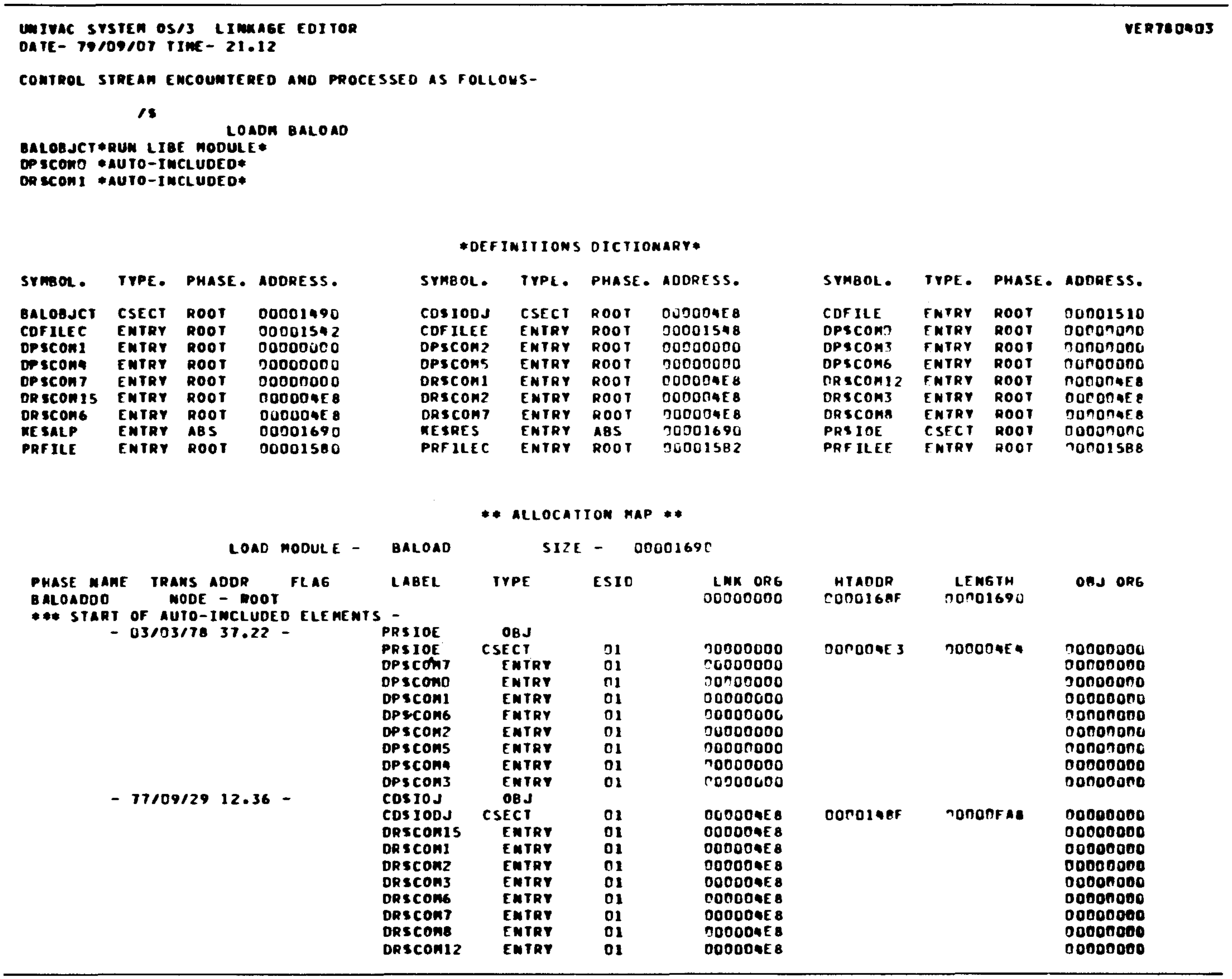
```
0014C0   103158F0  103405EF  F222C1E4  C194F222    C1E7C199  FA22C1E7  C1E4F332  C1BCC1E7    01CAC0
0014E0   96F0C1BF  D210C1C3  C19E5810  C1F85800    C1FC9220  103158F0  103405EF  47F0C012    01CAE0
001500   5810C1F0  0A275810  C1F80A27  0A1A0700    0C000000  00000000  00000000  00000000    01CB00
001520   00000000  00000000  00000000  C3C4C6C9    D3C54040  00000000  00001500  000008F0    01CB20
001540   01000000  000004E8  00001500  00001624    0C000000  00000101  82008004  00280000    01CB40
001560   00000000  00000000  00000000  00000000    00000000  00280400  0000150C  3C000001    01CB60
001580   00000000  00000000  00000000  00000000    0C000000  00000000  00000000  D7D9C6C9    01CB80
0015A0   03C54040  00000000  00000000  000004F0    FC000000  00000000  00001500  0000164C    01CBA0
0015C0   00000000  00000400  C0000004  00280009    00000000  0000150C  00000000  5830D01C    01CBC0
0015E0   1A300207  30809002  02076DEE  90020207    79DE9002  92126DF6  41106DEE  41000001    01CBE0
001600   0A251200  47406622  02014000  6DF20203    4C086DEE  9A014006  920C008E  9218008F    01CC00
001620   47F06640  02254040  40404040  40404040    40404040  40404040  40404040  40404040    01CC20
001640   40404040  40404040  40404040  40404040    40404040  40404040  40404040  40404040    01CC40
001660   40404040  40404040  40404040  40404040    40404040  404047E0  67AC5830  00FC1A3D    01CC60
001680   00001510  00001624  00001580  0000164C    478067AC  950F9001  47706686  02076792    01CC80
0016A0   900245F0  673A1200  47F06E3E  41106792    4100679A  89000008  430066CB  41110000    01CCA0
0016C0   561066C6  47F066CC  11000000  00230A5E    12004740  67864780  668695C6  67984770    01CCC0
0016E0   6E145820  81180507  67A32008  47706772    91802000  47806E4E  48002002  5A46B118    01CCE0
001700   43107171  4410678E  47806772  91012000    4710672A  48104004  89100008  5A102004    01CD00
001720   41101018  50102004  96012000  5820B118    4C203000  41303002  47F066AA  41106792    01CD20
001740   4100679A  89000008  4300675B  41110000    56106756  47F0675C  11000000  00630A43    01CD40
001760   12004740  676407FF  41000207  45F07A5E    47F06D1A  41202010  95FF2000  477066E4    01CD60
001780   41000204  47F06E26  41000205  47F07A56    91004001  00000004  050CF0F0  4390F000    01CD80
0017A0   00000000  00000016  90C2C103  D6C11B99    914C0078  47806702  94BFD078  07017A66    01CDA0
0017C0   7A6645F0  7A420501  7A667A68  470067C0    47F06808  91200078  47806804  45F07A42    01CDC0
0017E0   95089001  477067F2  95009006  4770670A    47F0685E  12114780  6C449509  9001478C    01CDE0
001800   6A6247F0  67DA45F0  7A421B22  43209001    89200002  44026816  47F06C44  47F06804    01CE00
001820   47F0688E  47F06C44  47F06804  47F06804    47F06804  47F0698C  47F06856  47F06A62    01CE20
```

Figure 4-2. BAL Dump Analysis Listing (Part 4 of 7)



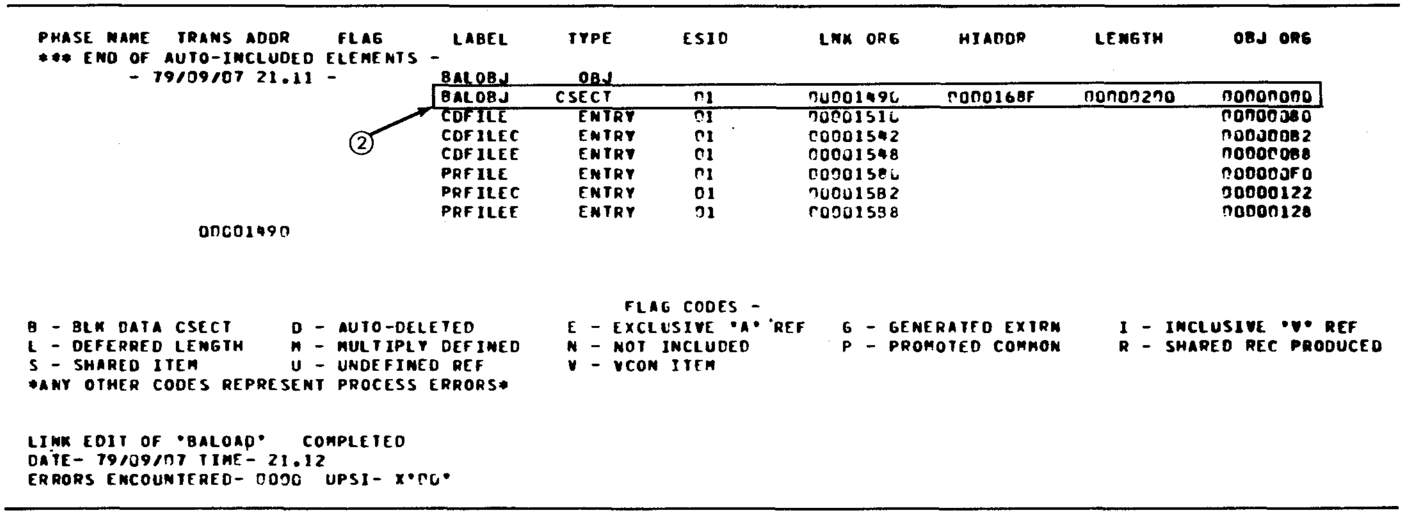
```
                                                                                                            PAGE    1

LOC.   OBJECT CODE    ADDR1 ADDR2  LINE  SOURCE STATEMENT                                            OS/3 ASM 79/09/07
000000                               1 BALOBJ  START 0
000000 05C0                          2         BALR  12,0
000000                               3         USING BALOBJ,12
                                     4         OPEN  CDFILE,PRFILE          OPEN FILES.
000002 0700                 A   5+        CNOP  0,4                                                    OPE00130
000004 4510 C010      00010 A   6+        BAL   1,*+(4*3)                                              OPE00500
000008 F0                   A   7+        DC    X'F0'                                                  OPE00540
000009 000080               A   8+        DC    AL3(CDFILE)                                            OPE00600
00000C 80                   A   9+        DC    X'80'                                                  OPE00550
00000D 0000F0               A  10+        DC    AL3(PRFILE)                                            OPE00600
000010 0A26                 A  11+        SVC   38 ISSUE SVC                                           OPE00800
000012 9240 C194     00194            12 LOOP    MVI   CDBUFF,C' '            CLEAR INPUT BUFFER.
000016 D226 C195 C194 00195 00194     13         MVC   CDBUFF+1(39),CDBUFF
00001C 9240 C1BC     001BC            14         MVI   PRBUFF,C' '            CLEAR OUTPUT BUFFER.
000020 D226 C1BD C1BC 001BD 001BC     15         MVC   PRBUFF+1(39),PRBUFF
                                     16         GET   CDFILE,CDBUFF          READ INPUT RECORD.
000026                      A  17+        DC    0Y(0) SET ALIGNMENT                                    GET00230
000026 5810 C1F0      001F0 A  18+        L     1,=A(CDFILE) LOAD R1S, FILENAME ADDRESS                GET00260
00002A 5800 C1F4      001F4 A  19+        L     0,=A(CDBUFF) LOAD R0S, WORKAREA ADDRESS                GET00850
00002E 9210 1031     00031       A  20+        MVI   49(1),X'10' SET FUNCTION CODE                          GET01010
000032 58F0 1034            00034 A  21+        L     15,52(,1) LOAD ADDR OF COMMON I/O                      GET01020
000036 05EF                 A  22+        BALR  14,15 LINK TO COMMON                                   GET01030
000038 F222 C1E4 C194 001E4 00194     23         PACK  PADD1(3),ADD1(3)         CONVERT 1ST NUM TO PACKED DEC.
00003E F222 C1E7 C199 001E7 00199     24         PACK  PADD2(3),ADD2(3)         CONVERT 2ND NUM TO PACKED DEC.
000044 FA22 C1E7 C1E4 001E7 001E4     25         AP    PADD2(3),PADD1(3)        ADD NUMBERS.
00004A F332 C1BC C1E7 001BC 001E7     26         UNPK  SUM(4),PADD2(3)          CONVERT SUM TO EBCDIC.
000050 96F0 C1BF     001BF            27         OI    SUM+3,X'F0'            MAKE LOW-ORDER DIGIT PRINTABLE.
000054 D21D C1C3 C19E 001C3 0019E     28         MVC   NAMEOUT(30),NAMEIN     MOVE CHARACTER STRING TO OUTPUT
                                     29         PUT   PRFILE,PRBUFF          WRITE OUTPUT RECORD
00005A                      A  30+        DC    0Y(0) SET ALIGNMENT                                    GET00230
00005A 5810 C1F8      001F8 A  31+        L     1,=A(PRFILE) LOAD R1S, FILENAME ADDRESS                GET00260
00005E 5800 C1FC      001FC A  32+        L     0,=A(PRBUFF) LOAD R0S, WORKAREA ADDRESS                GET00850
000062 9220 1031     00031       A  33+        MVI   49(1),X'20' SET FUNCTION CODE                          GET01010
000066 58F0 1034            00034 A  34+        L     15,52(,1) LOAD ADDR OF COMMON I/O                      GET01020
00006A 05EF                 A  35+        BALR  14,15 LINK TO COMMON                                   GET01030
00006C 47F0 C012            00012     36         B     LOOP                   LOOP BACK.
                                    37 *
                                    38 ENDPROC CLOSE CDFILE,PRFILE          END-OF-PROGRAM HOUSEKEEPING.
000070                      A  39+ENDPROC DC    0Y(0)                                                  OPE00100
000070 5810 C1F0      001F0 A  40+        L     1,=A(CDFILE) LOAD R1S, FILENAME ADDRESS                OPE00720
000074 0A27                 A  41+        SVC   39 ISSUE SVC                                           OPE00730
000076 5810 C1F8      001F8 A  42+        L     1,=A(PRFILE) LOAD R1S, FILENAME ADDRESS                OPE00720
00007A 0A27                 A  43+        SVC   39 ISSUE SVC                                           OPE00730
                                    44         EOJ
00007C                      A  45+        DS    0H                                                     EOJ00050
00007C 0A1A                 A  46+        SVC   26                                                     EOJ00070
                                    47 *
                                    48 CDFILE  DTFCD BLKSIZE=40,EOFADDR=ENDPROC,ERROR=ENDPROC,IOAREA1=CDBUFF,X
                                                     RECSIZE=40,SAVAREA=SAVE
```

Figure 4-2. BAL Dump Analysis Listing (Part 5 of 7)



```
UNIVAC SYSTEM OS/3  LINKAGE EDITOR                                                                          VER780403
DATE- 79/09/07 TIME- 21.12

CONTROL STREAM ENCOUNTERED AND PROCESSED AS FOLLOWS-

          /$
                  LOADM BALOAD
BALOBJCT*RUN LIBE MODULE*
DP$COM0 *AUTO-INCLUDED*
DR$COM1 *AUTO-INCLUDED*


                                              *DEFINITIONS DICTIONARY*

SYMBOL.   TYPE.  PHASE. ADDRESS.          SYMBOL.   TYPE.  PHASE. ADDRESS.          SYMBOL.   TYPE.  PHASE. ADDRESS.

BALOBJCT  CSECT  ROOT   00001490          CD$IODJ   CSECT  ROOT   000004E8          CDFILE    ENTRY  ROOT   00001510
CDFILEC   ENTRY  ROOT   00001542          CDFILEE   ENTRY  ROOT   00001548          DP$COM0   ENTRY  ROOT   00000000
DP$COM1   ENTRY  ROOT   00000000          DP$COM2   ENTRY  ROOT   00000000          DP$COM3   ENTRY  ROOT   00000000
DP$COM4   ENTRY  ROOT   00000000          DP$COM5   ENTRY  ROOT   00000000          DP$COM6   ENTRY  ROOT   00000000
DP$COM7   ENTRY  ROOT   00000000          DR$COM1   ENTRY  ROOT   000004E8          DR$COM12  ENTRY  ROOT   000004E8
DR$COM15  ENTRY  ROOT   000004E8          DR$COM2   ENTRY  ROOT   000004E8          DR$COM3   ENTRY  ROOT   000004E8
DR$COM6   ENTRY  ROOT   000004E8          DR$COM7   ENTRY  ROOT   000004E8          DR$COM8   ENTRY  ROOT   000004E8
KE$ALP    ENTRY  ABS    00001690          KE$RES    ENTRY  ABS    00001690          PR$IOE    CSECT  ROOT   00000000
PRFILE    ENTRY  ROOT   00001580          PRFILEC   ENTRY  ROOT   00001582          PRFILEE   ENTRY  ROOT   00001588


                                                ** ALLOCATION MAP **

                         LOAD MODULE -   BALOAD         SIZE -   00001690

 PHASE NAME  TRANS ADDR     FLAG       LABEL      TYPE       ESID         LNK ORG     HTADDR      LENGTH       OBJ ORG
 BALOAD00       NODE - ROOT                                               00000000    0000168F    00001690
 *** START OF AUTO-INCLUDED ELEMENTS -
       - 03/03/78 37.22 -             PR$IOE     OBJ
                                      PR$IOE    CSECT       01            00000000    000004E3    000004E4     00000000
                                      DP$COM7     ENTRY     01            00000000                             00000000
                                      DP$COM0     ENTRY     01            00000000                             00000000
                                      DP$COM1     ENTRY     01            00000000                             00000000
                                      DP$COM6     ENTRY     01            00000000                             00000000
                                      DP$COM2     ENTRY     01            00000000                             00000000
                                      DP$COM5     ENTRY     01            00000000                             00000000
                                      DP$COM4     ENTRY     01            00000000                             00000000
                                      DP$COM3     ENTRY     01            00000000                             00000000
       - 77/09/29 12.36 -             CD$IOJ     OBJ
                                      CD$IODJ   CSECT       01            000004E8    0000148F    00000FA8     00000000
                                      DR$COM15    ENTRY     01            000004E8                             00000000
                                      DR$COM1     ENTRY     01            000004E8                             00000000
                                      DR$COM2     ENTRY     01            000004E8                             00000000
                                      DR$COM3     ENTRY     01            000004E8                             00000000
                                      DR$COM6     ENTRY     01            000004E8                             00000000
                                      DR$COM7     ENTRY     01            000004E8                             00000000
                                      DR$COM8     ENTRY     01            000004E8                             00000000
                                      DR$COM12    ENTRY     01            000004E8                             00000000
```

Figure 4-2. BAL Dump Analysis Listing (Part 6 of 7)



```
PHASE NAME  TRANS ADDR     FLAG       LABEL      TYPE       ESID          LNK ORG     HIADDR      LENGTH       OBJ ORG
*** END OF AUTO-INCLUDED ELEMENTS -
        - 79/09/07 21.11 -            BALOBJ      OBJ
                                      BALOBJ     CSECT       01          00001490    0000168F    00000290     00000000
                                      CDFILE      ENTRY      01          00001510                             00000080
                                      CDFILEC     ENTRY      01          00001542                             000000B2
                                      CDFILEE     ENTRY      01          00001548                             000000B8
                                      PRFILE      ENTRY      01          00001580                             000000F0
                                      PRFILEC     ENTRY      01          000015B2                             00000122
                                      PRFILEE     ENTRY      01          000015B8                             00000128
              00001490


                                                         FLAG CODES -
B - BLK DATA CSECT     D - AUTO-DELETED         E - EXCLUSIVE 'A' REF    G - GENERATED EXTRN      I - INCLUSIVE 'V' REF
L - DEFERRED LENGTH    M - MULTIPLY DEFINED     N - NOT INCLUDED         P - PROMOTED COMMON      R - SHARED REC PRODUCED
S - SHARED ITEM        U - UNDEFINED REF        V - VCON ITEM
*ANY OTHER CODES REPRESENT PROCESS ERRORS*


LINK EDIT OF 'BALOAD'   COMPLETED
DATE- 79/09/07 TIME- 21.12
ERRORS ENCOUNTERED- 0000  UPSI- X'00'
```

Figure 4-2. **BAL Dump Analysis Listing** (Part 7 of 7)

4.3. COBOL Dump Analysis

In this subsection, we analyze a JOBDUMP generated from a COBOL program called COBOBJ. This program acts much like the BALOBJ program in 4.2: it reads two numbers off a punched card, adds them, and outputs the sum to the printer. In our situation, the program aborts after reading the third data card in the deck. We have inserted an // OPTION JOBDUMP card immediately preceding the EXEC instruction for COBOBJ. We have also, as required, routed the JOBDUMP output to a separate printer file with an LFD of PRNTR, thus causing the JOBDUMP to appear after the entire user log has been output.

4.3.1. Materials Used

The materials we will use in this analysis are contained in 4.3.4. They are as shown in Figure 4-4:

- The COBOL source code for COBOBJ
- The COBOL object map produced by the COBOL compiler in response to the LST=O parameter
- The linkage editor allocation map
- The edited JOBDUMP

4.3.2. Outline of JOBDUMP

As before, we will use the step-pointer method to find our way through our dump analysis. Before we begin analysis, you should look at the portion of the JOBDUMP shown in Figure 4-4. As you can see, these pages interpret the raw dump and present pertinent information in the form of charts, tables, and other narrative.

The remainder of the JOBDUMP shows the contents of the user job region in hexadecimal. Like the EOJ dump shown in 4.2, both margins contain the address of the first byte in each line. Unlike the EOJ dump, though, the JOBDUMP interprets dump data in EBCDIC in a column to the right of the data itself. Also, unlike the EOJ dump, the JOBDUMP uses different base addresses according to the following scheme, shown in Figure 4-3.

As you can see, the simple EOJ dump division of a job into its prologue and its program region is extended in the JOBDUMP. Each table in the prologue has a header and its own self-relative addresses in its left margin. Likewise, each CSECT (control section) within the program region begins with a header and has its own set of self-relative addresses. (Certain spooling and related tables in the prologue keep their negative job-relative addresses in a manner like that of the EOJ dump.)

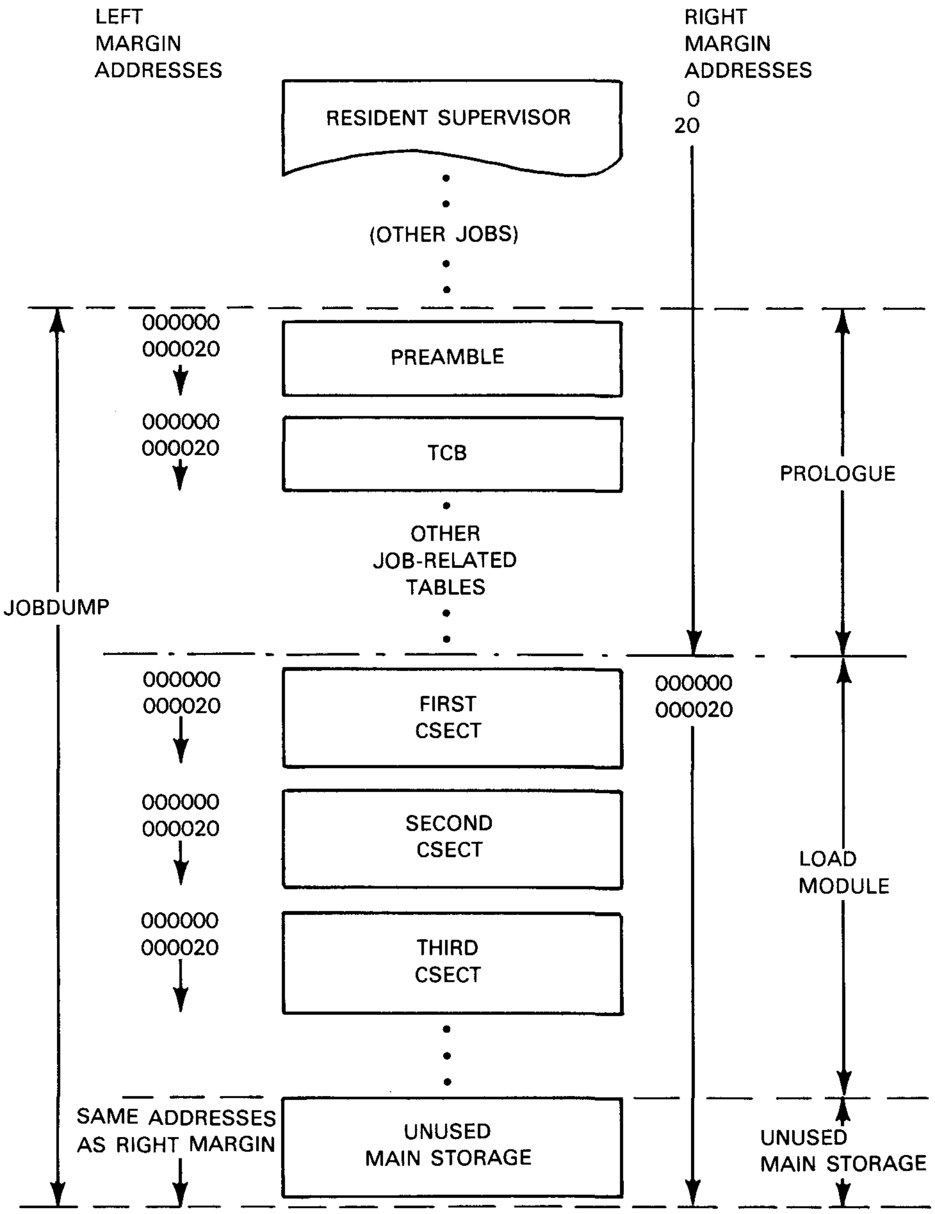


Figure 4-3. JOBDUMP Organization

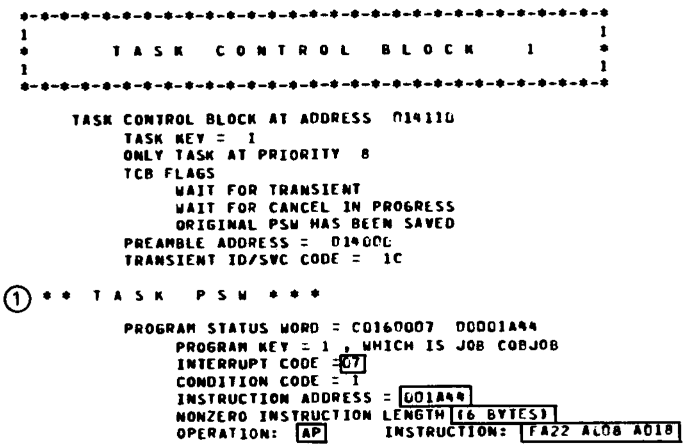
The prologue/program region division of the EOJ dump is carried over to the right margin except that the JOBDUMP prologue contains absolute system addresses while the JOBDUMP program region has self-relative addresses extending down the entire region. Unused main storage within the job region (that main storage not contained within the program region) is addressed relative to the job in both margins.

The JOBDUMP margin addresses will find much use in the upcoming analysis, so look at the JOBDUMP, Figure 4-4 (parts 6-14), to familiarize yourself with the JOBDUMP addressing scheme.

JOBDUMP requires space on the first VTOC cylinder of SYSRUN to allocate files. Users should eliminate unused files on SYSRUN.

4.3.3. Analysis

To determine what caused the JOBDUMP we proceed as follows:



```
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
I                                                       I
*       TASK  CONTROL  BLOCK    1                       *
I                                                       I
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

     TASK CONTROL BLOCK AT ADDRESS  014110
          TASK KEY =  1
          ONLY TASK AT PRIORITY  8
          TCB FLAGS
               WAIT FOR TRANSIENT
               WAIT FOR CANCEL IN PROGRESS
               ORIGINAL PSW HAS BEEN SAVED
          PREAMBLE ADDRESS =  01400C
          TRANSIENT ID/SVC CODE =  1C

① ** TASK  PSW ***

          PROGRAM STATUS WORD = C0160007  00001A44
               PROGRAM KEY = 1 , WHICH IS JOB COBJOB
               INTERRUPT CODE =07
               CONDITION CODE = 1
               INSTRUCTION ADDRESS = 001A44
               NONZERO INSTRUCTION LENGTH (6 BYTES)
               OPERATION: AP      INSTRUCTION: FA22 A008 A018
```

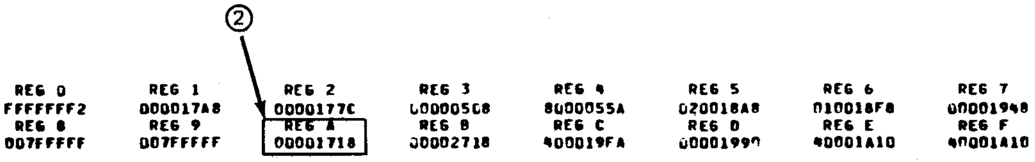
We first look at the task PSW in the JOBDUMP narrative ①. There, all information contained in the PSW is extracted for you. Using this information we see that:

- The interrupt code of 07 indicates a data exception (see Table A-1).
- The failing instruction is add decimal (AP).

- The instruction address can be found by subtracting the AP instruction length from the address given in the PSW:

1A44	PSW address
- 6	Instruction length
1A3E	Failing instruction address

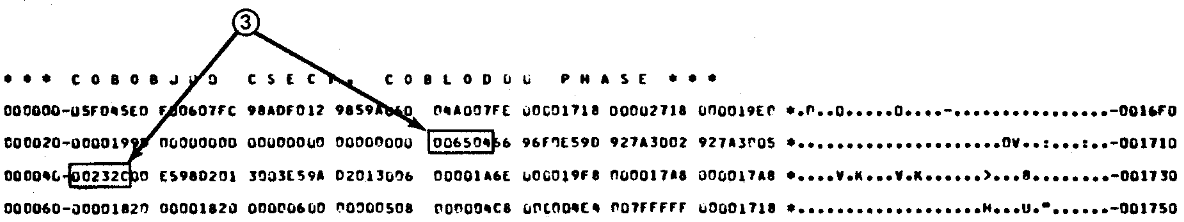
As before, all addresses given are relative to the job base address. Since data exceptions are associated with operands in main storage, we should find the operands addressed by the AP instruction.



```
 REG 0      REG 1       REG 2      REG 3      REG 4      REG 5      REG 6      REG 7
FFFFFFF2   000017A8   0000177C   000005C8   8000055A   020018A8   010018F8   00001948
 REG 8      REG 9       REG A      REG B      REG C      REG D      REG E      REG F
007FFFFF   007FFFFF   00001718   00002718   400019FA   00001990   40001A10   40001A10
```

The two operands of the AP instruction shown in ① are both of register-displacement type using register 10 (A_{16}). ② points to the contents of register 10 at the time the exception occurred. From this we can find the two operand addresses:

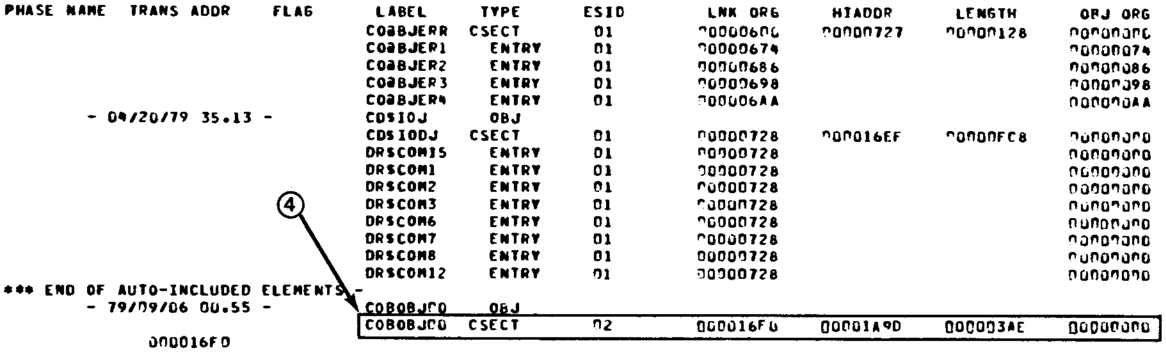
1718	Register 10 contents	1718	Register 10 contents
+ 8	Operand 1 displacement	+ 18	Operand 2 displacement
1720	Operand 1 address	1730	Operand 2 address



```
*** C O B O B J 0 0   C S E C T .   C O B L O D 0 0   P H A S E ***
000000-05F045E0 F00607FC 98ADF012 9859A060  04A007FE 00C01718 00002718 000019EC *.Ŋ..0.....0....¬................-0016F0
000020-00001990 00000000 00000000 00000000  00650456 96F0E590 927A3002 927A3C05 *......................0V..:...:..-001710
000040-00232C00 E5980201 3003E59A D2013006  00001A6E 000019F8 000017A8 000017A8 *....V.K...V.K.......>...8........-001730
000060-00001820 00001820 00000600 00000508  000004C8 000004E4 007FFFFF 00001718 *......................H...U.*......-001750
```

In ③ we see the two AP operands at the addresses calculated in ②. Note that the AP instruction operates on three bytes beginning with each operand. In decimal instructions such as AP, the low-order half byte of each operand is treated as the sign for that operand. Operand 2, at 1730, has a sign value of C, indicating a positive number. Operand 1, however, has an invalid sign value of 4. We can see now that this invalid sign caused a data exception when the AP instruction attempted to operate on it.

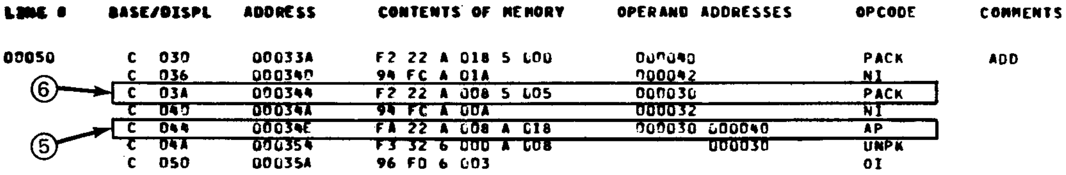
At this point, we have found the immediate cause of the error, an erroneous sign. We now attempt to find how the error occurred.



```
PHASE NAME  TRANS ADDR     FLAG      LABEL      TYPE       ESID         LNK ORG      HIADDR      LENGTH       OBJ ORG
                                     CO@BJERR  CSECT        01         00000600     00000727    00000128     00000000
                                     CO@BJER1    ENTRY      01         00000674                              00000074
                                     CO@BJER2    ENTRY      01         00000686                              00000086
                                     CO@BJER3    ENTRY      01         00000698                              00000098
                                     CO@BJER4    ENTRY      01         000006AA                              000000AA
      - 04/20/79 35.13 -             CD$IOJ      OBJ
                                     CD$IODJ   CSECT        01         00000728     000016EF    00000FC8     00000000
                                     DR$COM15    ENTRY      01         00000728                              00000000
                                     DR$COM1     ENTRY      01         00000728                              00000000
                                     DR$COM2     ENTRY      01         00000728                              00000000
                                ④    DR$COM3     ENTRY      01         00000728                              00000000
                                     DR$COM6     ENTRY      01         00000728                              00000000
                                     DR$COM7     ENTRY      01         00000728                              00000000
                                     DR$COM8     ENTRY      01         00000728                              00000000
                                     DR$COM12    ENTRY      01         00000728                              00000000
*** END OF AUTO-INCLUDED ELEMENTS -
      - 79/09/06 00.55 -             COBOBJ00    OBJ
                                     COBOBJ00  CSECT        02         000016F0     00001A9D    000003AE     00000000
           000016F0
```

Because the COBOL compiler generated all the instructions leading up to the error, we now go to the COBOL listing and linkage editor allocation map for program COBOBJ. We will have to look at the allocation map first because the COBOL object program we seek is only one among a number of object modules occupying load module COBLOD. We see from ④ that the AP instruction address of 1A3E lies between the LNKORG address of 16F0 and the HIADDR address of 1A9D, thus placing it inside the COBOBJ00 CSECT. (This CSECT is also labeled in the JOBDUMP; see ③ .) We will need to know the displacement of the AP instruction within COBOBJ00 and this can be done as follows:

1A3E	Instruction address from ①
- 16F0	LNKORG address
34E	AP instruction displacement within COBOBJ00



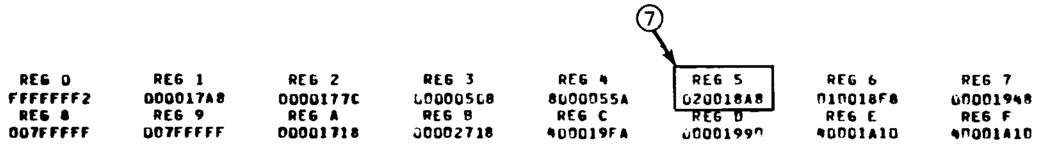
```
LINE #    BASE/DISPL    ADDRESS      CONTENTS OF MEMORY       OPERAND ADDRESSES      OPCODE     COMMENTS

00050       C  030      00033A      F2 22 A 018 5 000         000040                PACK        ADD
            C  036      000340      94 FC A 01A               000042                NI
  ⑥ ──►     C  03A      000344      F2 22 A 008 5 005         000030                PACK
            C  040      00034A      94 FC A 00A               000032                NI
  ⑤ ──►     C  044      00034E      FA 22 A 008 A 018         000030 000040         AP
            C  04A      000354      F3 32 6 000 A 008                000030         UNPK
            C  050      00035A      96 F0 6 003                                     OI
```

Looking at address 34E ⑤ on the COBOL object map, we can see the COBOL instruction that generated the AP. Under the heading OPERAND ADDRESSES are listed the addresses of the two AP operands. We can confirm this by finding their addresses relative to COBOBJ:

16F0	LNKORG address	16F0	LLNKORG address
+ 30	Operand 1 address from ⑤	+ 40	Operand 2 address from⑤
1720	Operand 1 load module address	1730	Operand 2 load module address

Compare the two addresses to the addresses calculated in ② .

Because operand 1 is the operand that contains the erroneous sign, we should look at instructions prior to the AP instruction that used operand 1 (see illustration above ⑤). Using the OPERAND ADDRESS column, we find that a PACK instruction ⑥ used operand 1 as its destination field. We next look at the source operand.

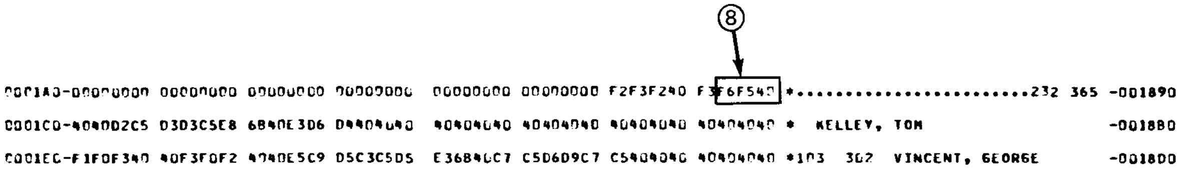


```
  REG 0      REG 1      REG 2      REG 3      REG 4      REG 5      REG 6      REG 7
FFFFFFF2   000017A8   0000177C   00000508   8000055A   020018A8   010018F8   00001948
  REG 8      REG 9      REG A      REG B      REG C      REG D      REG E      REG F
007FFFFF   007FFFFF   00001718   00002718   400019FA   00001990   40001A10   40001A10
```

The source field, like the destination field, appears in displacement/base form and uses register 5. Looking at the register contents ⑦ , we can calculate the source field address as follows:

(02)0018A8	Register 5 contents (leading byte ignored)
+ 5	Displacement
18AD	Address of source field

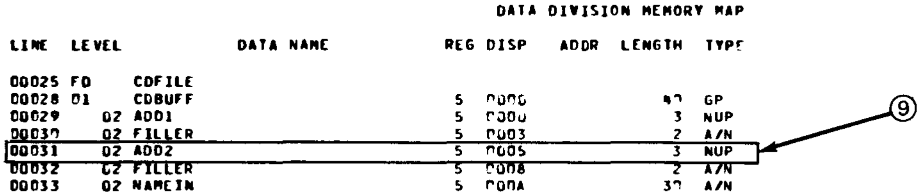
For a 4-byte address such as the one in register 5 above, only bits 8-31, the low-order three bytes, are actually used in addressing main storage.



```
0001A0-00000000 00000000 00000000 00000000  00000000 00000000 F2F3F240 F3F6F540 *........................232 365 -001890
0001C0-4040D2C5 D3D3C5E8 6B40E3D6 D4404040  40404040 40404040 40404040 40404040 *  KELLEY, TOM                -0018B0
0001E0-F1F0F340 40F3F0F2 4040E5C9 D5C3C5D5  E36B40C7 C5D6D9C7 C5404040 40404040 *103   302  VINCENT, GEORGE       -0018D0
```

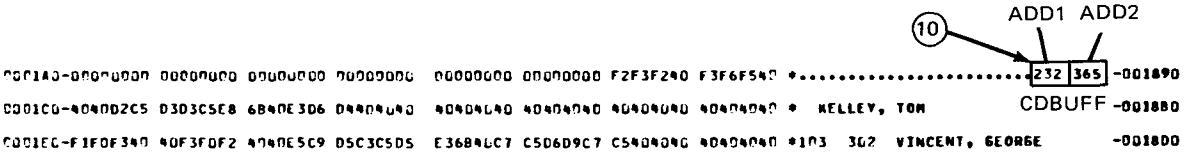
At location 18AD ⑧ , we see that the three bytes operated on by the PACK instruction of ⑥ comprise two EBCDIC digits and a blank. The PACK instruction had treated the blank (X'40') as a signed digit by inverting its half bytes (X'40' becoming X'04') thus causing later difficulties by accidentally putting an invalid hexadecimal digit (4) in the sign.

We see by the interpreted data in the right half of the dump printout that these bytes formed part of a character string, very likely part of COBOBJ's data division. To check this we look next at the data division memory map output by the COBOL compiler.



```
                                                          DATA DIVISION MEMORY MAP

LINE  LEVEL               DATA NAME               REG DISP   ADDR  LENGTH  TYPE

00025 FD      CDFILE
00028 01      CDBUFF                               5  0000             40   GP
00029      02 ADD1                                 5  0000              3   NUP
00030      02 FILLER                               5  0003              2   A/N
00031      02 ADD2                                 5  0005              3   NUP
00032      02 FILLER                               5  0008              2   A/N
00033      02 NAMEIN                               5  000A             30   A/N
```

Looking at the data division main storage map of the COBOBJ compilation listing ⑨, we see that location 18AD lies within the input buffer CDBUFF and has the data-name ADD2. (Compare this address with the source field address in the PACK instruction calculated in ⑦.) We see too that ADD2 is a 3-character unsigned numeric field that is offset within the buffer by five bytes. Let's see what ADD2 contained a the time COBOBJ failed.



```
0001A0-00000000 00000000 00000000 00000000  00000000 00000000 F2F3F240 F3F6F540 *.......................232 365 -001890
0001C0-4040D2C5 D3D3C5E8 6B40E3D6 D4404040  40404040 40404040 40404040 40404040 *  KELLEY, TOM                  -0018B0
0001E0-F1F0F340 40F3F0F2 4040E5C9 D5C3C5D5  E36B40C7 C5D6D9C7 C5404040 40404040 *103  302  VINCENT, GEORGE        -0018D0
```

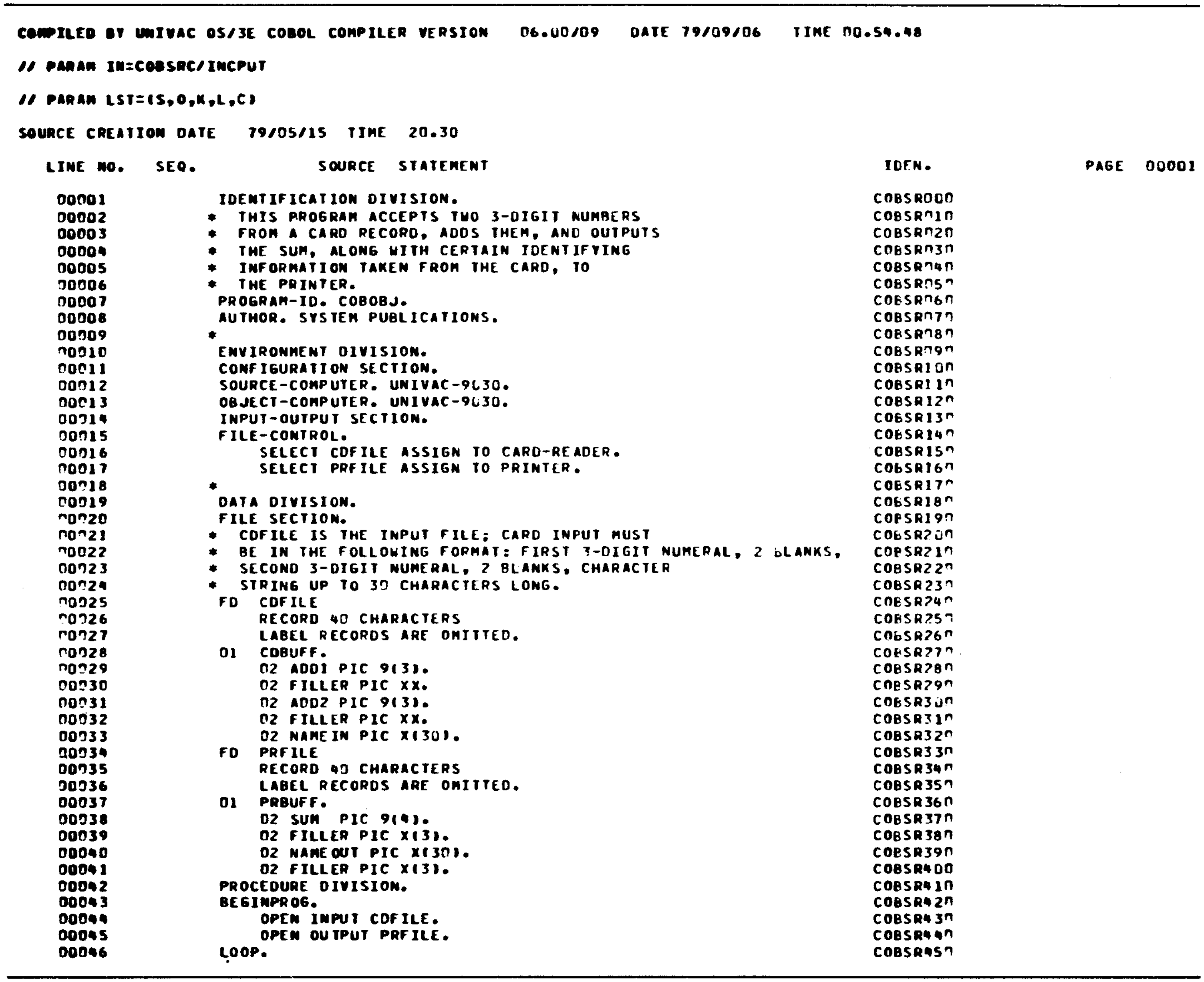
10
ADD1
ADD2
CDBUFF

The interpreted data at 18AD ⑩ shows that input numeral 365 incompletely overlaps data area ADD2, the leading digit falling outside of it altogether. By looking at the COBOBJ data division main storage map, it becomes apparent that the program exception was caused when the numeral to be input as ADD2, 365, was punched one column to the left of where it should have been. Repunching the card correctly will resolve this particular problem. In addition, you should include error handling routines in your program to prevent future recurrences.

4.3.4. Dump Analysis Materials

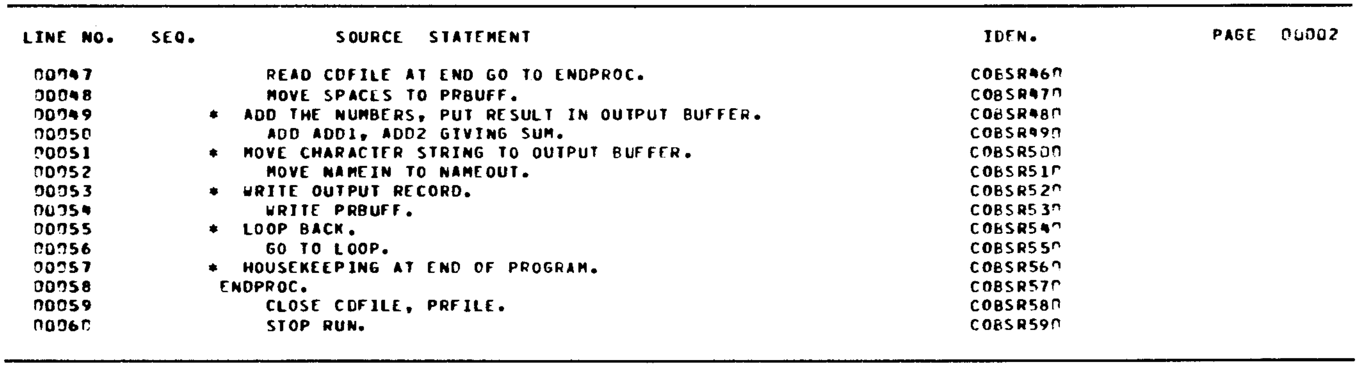
This subsection contains the edited printout from job COBADD, including the COBOL source program, compilation, and link edit for program COBOBJ. In addition, an edited JOBDUMP produced from the execution of JOBDUMP is shown here. All the pointers referred to in 4.3.3 are shown in numerical order, each pointer referring to the part of Figure 4-4 on which it appears.

①	part 7	⑥	part 3
②	part 8	⑦	part 8
③	part 12	⑧	part 13
④	part 5	⑨	part 4
⑤	part 3	⑩	part 13



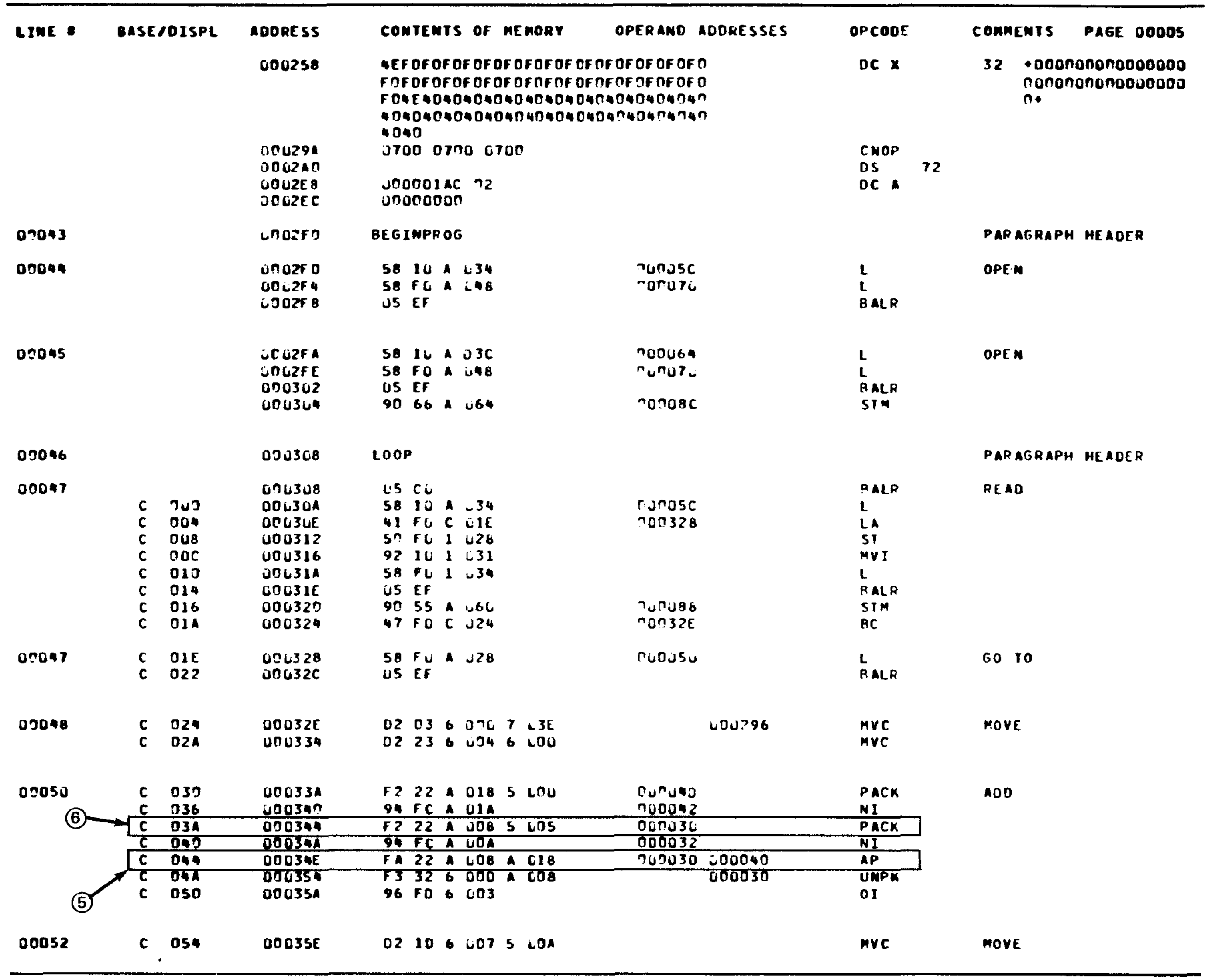
```
COMPILED BY UNIVAC OS/3E COBOL COMPILER VERSION   06.00/09   DATE 79/09/06   TIME 00.54.48

// PARAM IN=COBSRC/INCPUT

// PARAM LST=(S,O,K,L,C)

SOURCE CREATION DATE   79/05/15  TIME  20.30

   LINE NO.   SEQ.            SOURCE  STATEMENT                                        IDEN.                PAGE  00001

    00001          IDENTIFICATION DIVISION.                                            COBSR000
    00002          *  THIS PROGRAM ACCEPTS TWO 3-DIGIT NUMBERS                         COBSR010
    00003          *  FROM A CARD RECORD, ADDS THEM, AND OUTPUTS                       COBSR020
    00004          *  THE SUM, ALONG WITH CERTAIN IDENTIFYING                          COBSR030
    00005          *  INFORMATION TAKEN FROM THE CARD, TO                              COBSR040
    00006          *  THE PRINTER.                                                     COBSR050
    00007          PROGRAM-ID. COBOBJ.                                                 COBSR060
    00008          AUTHOR. SYSTEM PUBLICATIONS.                                        COBSR070
    00009          *                                                                   COBSR080
    00010          ENVIRONMENT DIVISION.                                               COBSR090
    00011          CONFIGURATION SECTION.                                              COBSR100
    00012          SOURCE-COMPUTER. UNIVAC-9030.                                       COBSR110
    00013          OBJECT-COMPUTER. UNIVAC-9030.                                       COBSR120
    00014          INPUT-OUTPUT SECTION.                                               COBSR130
    00015          FILE-CONTROL.                                                       COBSR140
    00016              SELECT CDFILE ASSIGN TO CARD-READER.                            COBSR150
    00017              SELECT PRFILE ASSIGN TO PRINTER.                                COBSR160
    00018          *                                                                   COBSR170
    00019          DATA DIVISION.                                                      COBSR180
    00020          FILE SECTION.                                                       COBSR190
    00021          *  CDFILE IS THE INPUT FILE; CARD INPUT MUST                        COBSR200
    00022          *  BE IN THE FOLLOWING FORMAT: FIRST 3-DIGIT NUMERAL, 2 BLANKS,     COBSR210
    00023          *  SECOND 3-DIGIT NUMERAL, 2 BLANKS, CHARACTER                      COBSR220
    00024          *  STRING UP TO 30 CHARACTERS LONG.                                 COBSR230
    00025          FD  CDFILE                                                          COBSR240
    00026              RECORD 40 CHARACTERS                                            COBSR250
    00027              LABEL RECORDS ARE OMITTED.                                      COBSR260
    00028          01  CDBUFF.                                                         COBSR270
    00029              02 ADD1 PIC 9(3).                                               COBSR280
    00030              02 FILLER PIC XX.                                               COBSR290
    00031              02 ADD2 PIC 9(3).                                               COBSR300
    00032              02 FILLER PIC XX.                                               COBSR310
    00033              02 NAMEIN PIC X(30).                                            COBSR320
    00034          FD  PRFILE                                                          COBSR330
    00035              RECORD 40 CHARACTERS                                            COBSR340
    00036              LABEL RECORDS ARE OMITTED.                                      COBSR350
    00037          01  PRBUFF.                                                         COBSR360
    00038              02 SUM  PIC 9(4).                                               COBSR370
    00039              02 FILLER PIC X(3).                                             COBSR380
    00040              02 NAMEOUT PIC X(30).                                           COBSR390
    00041              02 FILLER PIC X(3).                                             COBSR400
    00042          PROCEDURE DIVISION.                                                 COBSR410
    00043          BEGINPROG.                                                          COBSR420
    00044              OPEN INPUT CDFILE.                                              COBSR430
    00045              OPEN OUTPUT PRFILE.                                             COBSR440
    00046          LOOP.                                                               COBSR450
```

Figure 4-4. COBOL Dump Analysis Listings (Part 1 of 14)



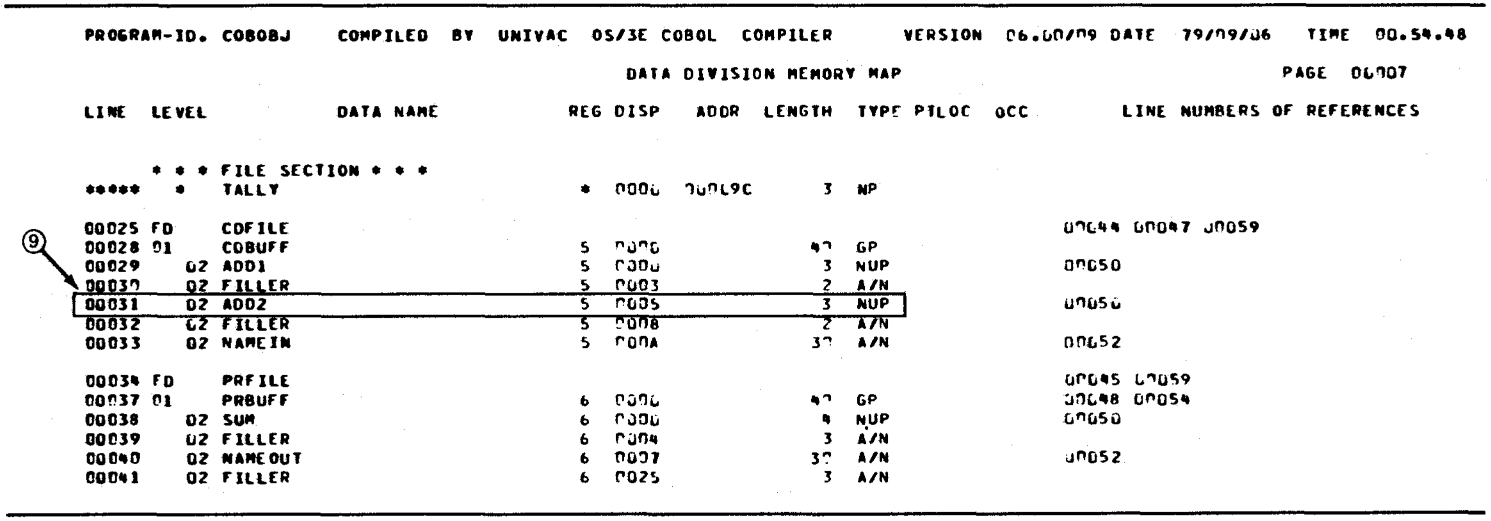
```
LINE NO.   SEQ.            SOURCE  STATEMENT                                                    IDEN.             PAGE  00002

 00047                 READ CDFILE AT END GO TO ENDPROC.                                      COBSR460
 00048                 MOVE SPACES TO PRBUFF.                                                 COBSR470
 00049             *  ADD THE NUMBERS, PUT RESULT IN OUTPUT BUFFER.                           COBSR480
 00050                 ADD ADD1, ADD2 GIVING SUM.                                             COBSR490
 00051             *  MOVE CHARACTER STRING TO OUTPUT BUFFER.                                 COBSR500
 00052                 MOVE NAMEIN TO NAMEOUT.                                                COBSR510
 00053             *  WRITE OUTPUT RECORD.                                                    COBSR520
 00054                 WRITE PRBUFF.                                                          COBSR530
 00055             *  LOOP BACK.                                                              COBSR540
 00056                 GO TO LOOP.                                                            COBSR550
 00057             *  HOUSEKEEPING AT END OF PROGRAM.                                         COBSR560
 00058             ENDPROC.                                                                   COBSR570
 00059                 CLOSE CDFILE, PRFILE.                                                  COBSR580
 00060                 STOP RUN.                                                              COBSR590
```

Figure 4-4. COBOL Dump Analysis Listings (Part 2 of 14)



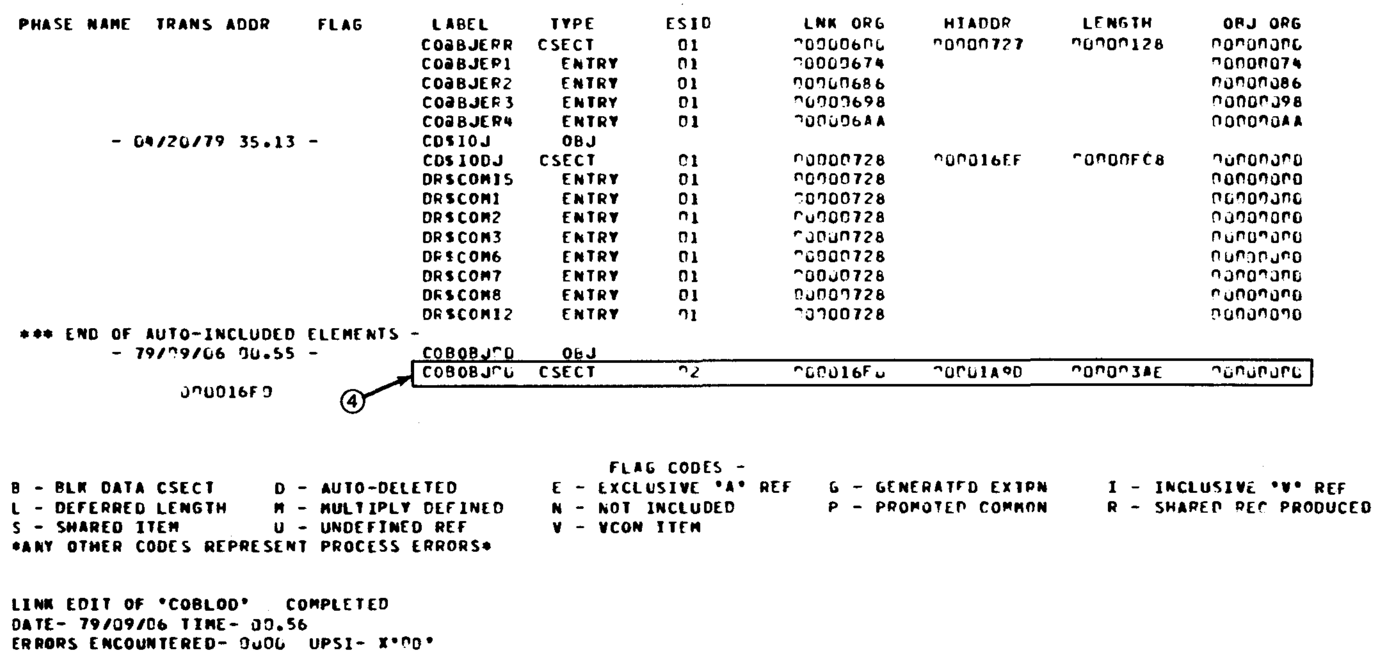
```
LINE #   BASE/DISPL   ADDRESS    CONTENTS OF MEMORY      OPERAND ADDRESSES     OPCODE     COMMENTS   PAGE 00005

                      000258     4EF0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0        DC X       32  +0000000000000000
                                 F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0                       0000000000000000
                                 F04E404040404040404040404040404040404040                     0+
                                 404040404040404040404040404040404040
                                 4040
                      00029A     0700 0700 0700                                CNOP
                      0002A0                                                   DS      72
                      0002E8     000001AC 02                                   DC A
                      0002EC     00000000

00043                 0002F0    BEGINPROG                                                 PARAGRAPH HEADER

00044                 0002F0     58 10 A 034          00005C                   L          OPEN
                      0002F4     58 F0 A 046          000070                   L
                      0002F8     05 EF                                         BALR


00045                 0002FA     58 10 A 03C          000064                   L          OPEN
                      0002FE     58 F0 A 048          000070                   L
                      000302     05 EF                                         BALR
                      000304     90 66 A 064          00008C                   STM


00046                 000308    LOOP                                                      PARAGRAPH HEADER

00047                 000308     05 C0                                         BALR       READ
         C   000      00030A     58 10 A 034          00005C                   L
         C   004      00030E     41 F0 C 01E          000328                   LA
         C   008      000312     50 F0 1 028                                   ST
         C   00C      000316     92 10 1 031                                   MVI
         C   010      00031A     58 F0 1 034                                   L
         C   014      00031E     05 EF                                         BALR
         C   016      000320     90 55 A 060          000086                   STM
         C   01A      000324     47 F0 C 024          00032E                   BC

00047    C   01E      000328     58 F0 A 028          000050                   L          GO TO
         C   022      00032C     05 EF                                         BALR


00048    C   024      00032E     D2 03 6 000 7 03E              000296         MVC        MOVE
         C   02A      000334     D2 23 6 004 6 000                             MVC


00050    C   030      00033A     F2 22 A 018 5 000    000040                   PACK       ADD
         C   036      000340     94 FC A 01A          000042                   NI
         C   03A      000344     F2 22 A 008 5 005    000030                   PACK
         C   040      00034A     94 FC A 00A          000032                   NI
         C   044      00034E     FA 22 A 008 A 018    000030 000040            AP
         C   04A      000354     F3 32 6 000 A 008              000030         UNPK
         C   050      00035A     96 F0 6 003                                   OI

00052    C   054      00035E     D2 10 6 007 5 00A                             MVC        MOVE
```

Figure 4-4. COBOL Dump Analysis Listings (Part 3 of 14)



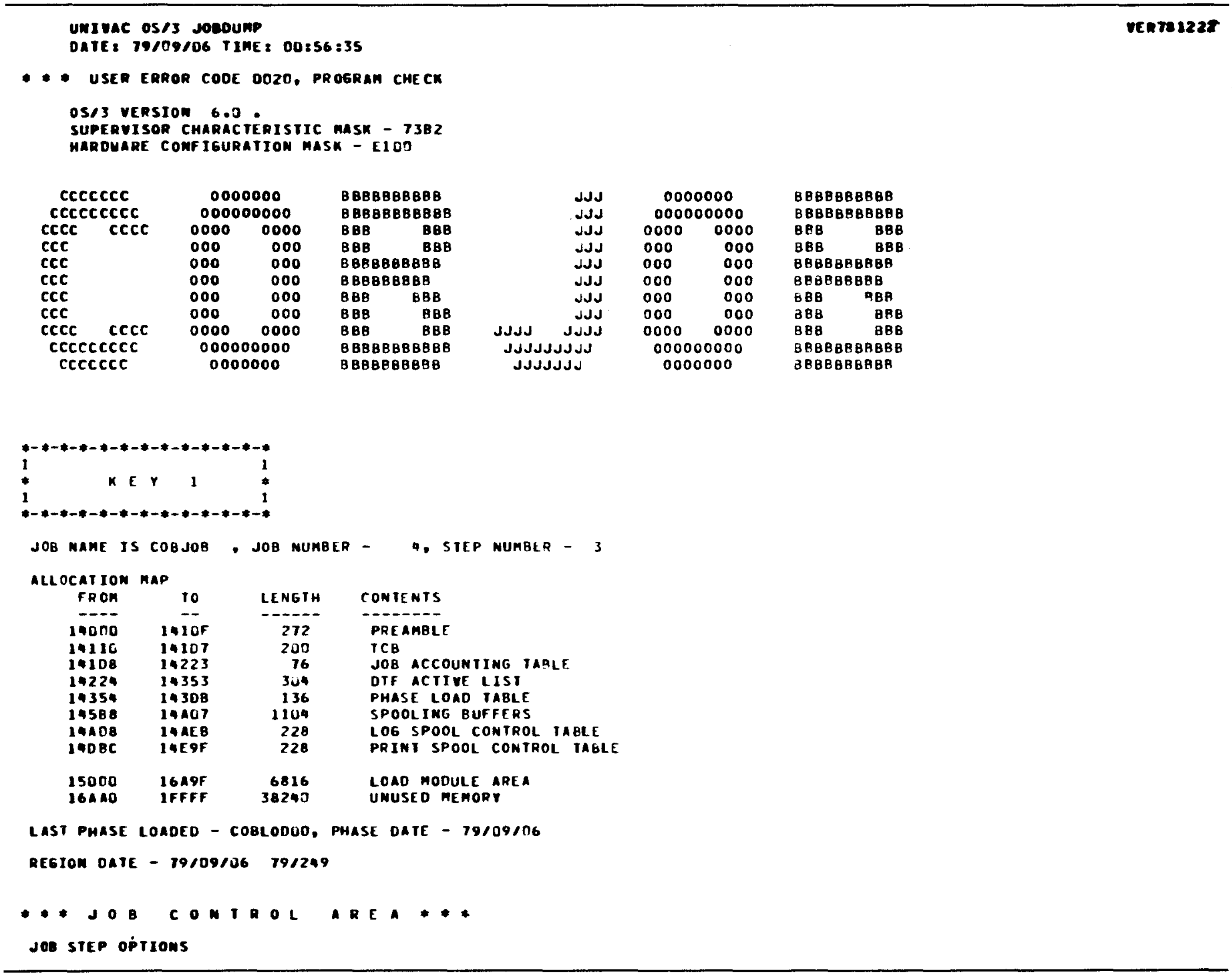
```
PROGRAM-ID. COBOBJ    COMPILED  BY  UNIVAC  OS/3E COBOL  COMPILER     VERSION  06.00/09 DATE  79/09/06   TIME  00.54.48

                                                    DATA DIVISION MEMORY MAP                                   PAGE  00007

LINE  LEVEL            DATA NAME             REG DISP    ADDR  LENGTH  TYPE PTLOC  OCC        LINE NUMBERS OF REFERENCES


      * * * FILE SECTION * * *
*****   *    TALLY                            *  0000  000C9C      3  NP

00025 FD     COFILE                                                                         00044 00047 00059
00028 01     COBUFF                           5  0000             40  GP
00029     02 ADD1                             5  0000              3  NUP                  00050
00030     02 FILLER                           5  0003              2  A/N
00031     02 ADD2                             5  0005              3  NUP                  00050
00032     02 FILLER                           5  0008              2  A/N
00033     02 NAMEIN                           5  000A             30  A/N                  00052

00034 FD     PRFILE                                                                         00045 00059
00037 01     PRBUFF                           6  0000             40  GP                   00048 00054
00038     02 SUM                              6  0000              4  NUP                  00050
00039     02 FILLER                           6  0004              3  A/N
00040     02 NAMEOUT                          6  0007             30  A/N                  00052
00041     02 FILLER                           6  0025              3  A/N
```

Figure 4-4. COBOL Dump Analysis Listings (Part 4 of 14)



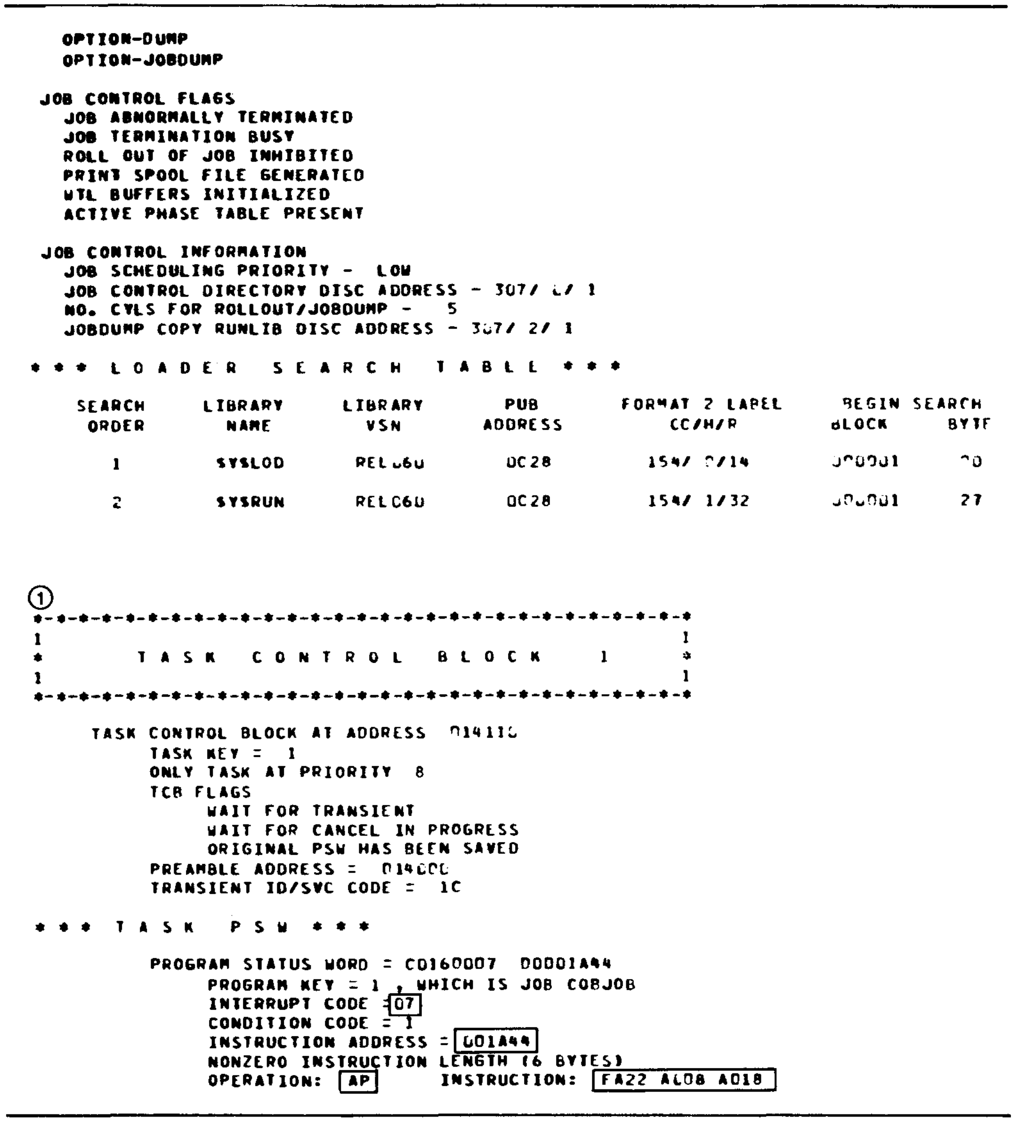
```
PHASE NAME  TRANS ADDR     FLAG     LABEL      TYPE        ESID          LNK ORG      HIADDR      LENGTH       OBJ ORG
                                    CO@BJERR  CSECT         01          00000600    00000727    00000128     00000000
                                    CO@BJER1    ENTRY       01          00000674                             00000074
                                    CO@BJER2    ENTRY       01          00000686                             00000086
                                    CO@BJER3    ENTRY       01          00000698                             00000098
                                    CO@BJER4    ENTRY       01          000006AA                             000000AA
      - 04/20/79 35.13 -            CD$IOJ      OBJ
                                    CD$IODJ   CSECT         01          00000728    000016EF    00000FC8     00000000
                                    DR$COM15    ENTRY       01          00000728                             00000000
                                    DR$COM1     ENTRY       01          00000728                             00000000
                                    DR$COM2     ENTRY       01          00000728                             00000000
                                    DR$COM3     ENTRY       01          00000728                             00000000
                                    DR$COM6     ENTRY       01          00000728                             00000000
                                    DR$COM7     ENTRY       01          00000728                             00000000
                                    DR$COM8     ENTRY       01          00000728                             00000000
                                    DR$COM12    ENTRY       01          00000728                             00000000
*** END OF AUTO-INCLUDED ELEMENTS -
      - 79/09/06 00.55 -            COBOBJ00    OBJ
                                    COBOBJ00  CSECT         02          000016F0    00001A9D    000003AE     00000000
          000016F0


                                           FLAG CODES -
B - BLK DATA CSECT     D - AUTO-DELETED       E - EXCLUSIVE 'A' REF   G - GENERATED EXTRN     I - INCLUSIVE 'V' REF
L - DEFERRED LENGTH    M - MULTIPLY DEFINED   N - NOT INCLUDED        P - PROMOTED COMMON     R - SHARED REC PRODUCED
S - SHARED ITEM        U - UNDEFINED REF      V - VCON ITEM
*ANY OTHER CODES REPRESENT PROCESS ERRORS*


LINK EDIT OF 'COBLOD'   COMPLETED
DATE- 79/09/06 TIME- 00.56
ERRORS ENCOUNTERED- 0000  UPSI- X'00'
```

Figure 4-4. COBOL Dump Analysis Listings (Part 5 of 14)



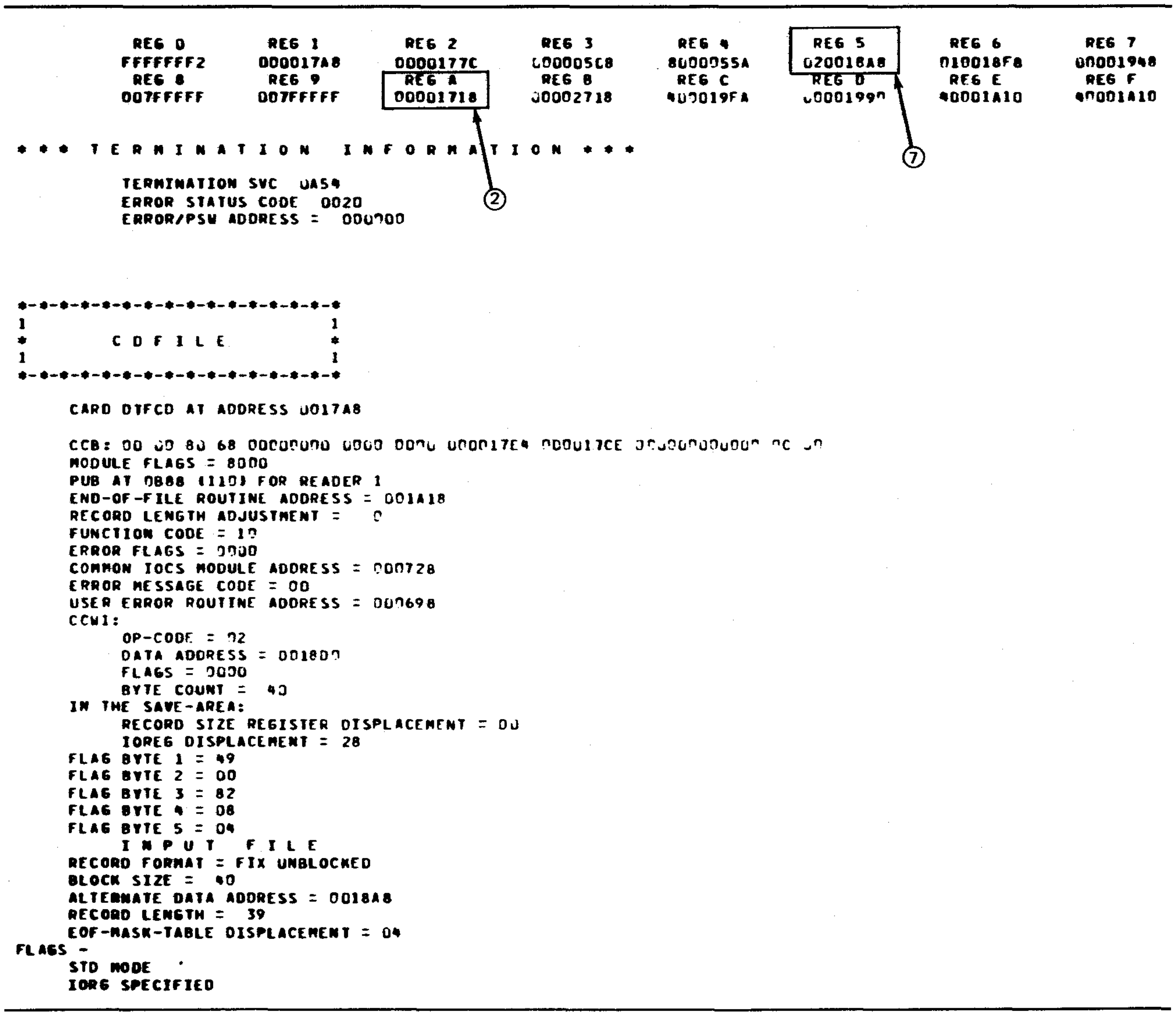
```
     UNIVAC OS/3 JOBDUMP                                                                                    VER781222
     DATE: 79/09/06 TIME: 00:56:35

* * *  USER ERROR CODE 0020, PROGRAM CHECK

     OS/3 VERSION  6.0 .
     SUPERVISOR CHARACTERISTIC MASK - 73B2
     HARDWARE CONFIGURATION MASK - E100

    CCCCCCC        0000000      BBBBBBBBBB             JJJ      0000000     BBBBBBBBBB
   CCCCCCCCC      000000000     BBBBBBBBBBB            JJJ     000000000    BBBBBBBBBBB
  CCCC   CCCC    0000   0000    BBB     BBB            JJJ    0000   0000   BBB     BBB
  CCC            000     000    BBB     BBB            JJJ    000     000   BBB     BBB
  CCC            000     000    BBBBBBBBBB             JJJ    000     000   BBBBBBBBBB
  CCC            000     000    BBBBBBBBB              JJJ    000     000   BBBBBBBBB
  CCC            000     000    BBB    BBB             JJJ    000     000   BBB    BBB
  CCC            000     000    BBB     BBB            JJJ    000     000   BBB     BBB
  CCCC   CCCC    0000   0000    BBB     BBB    JJJJ   JJJJ    0000   0000   BBB     BBB
   CCCCCCCCC      000000000     BBBBBBBBBBB     JJJJJJJJ       000000000    BBBBBBBBBBB
    CCCCCCC        0000000      BBBBBBBBBB       JJJJJJ         0000000     BBBBBBBBBB


*-*-*-*-*-*-*-*-*-*-*-*-*-*
I                         I
*       K E Y   1         *
I                         I
*-*-*-*-*-*-*-*-*-*-*-*-*-*

 JOB NAME IS COBJOB   , JOB NUMBER -     4, STEP NUMBER -  3

 ALLOCATION MAP
      FROM       TO       LENGTH    CONTENTS
      ----       --       ------    --------
     14000     1410F        272      PREAMBLE
     14110     141D7        200      TCB
     141D8     14223         76      JOB ACCOUNTING TABLE
     14224     14353        304      DTF ACTIVE LIST
     14354     143DB        136      PHASE LOAD TABLE
     145B8     14A07       1104      SPOOLING BUFFERS
     14A08     14AEB        228      LOG SPOOL CONTROL TABLE
     14DBC     14E9F        228      PRINT SPOOL CONTROL TABLE

     15000     16A9F       6816      LOAD MODULE AREA
     16AA0     1FFFF      38240      UNUSED MEMORY

 LAST PHASE LOADED - COBLOD00, PHASE DATE - 79/09/06

 REGION DATE - 79/09/06  79/249

* * *  J O B   C O N T R O L   A R E A  * * *

 JOB STEP OPTIONS
```

Figure 4-4. COBOL Dump Analysis Listings (Part 6 of 14)



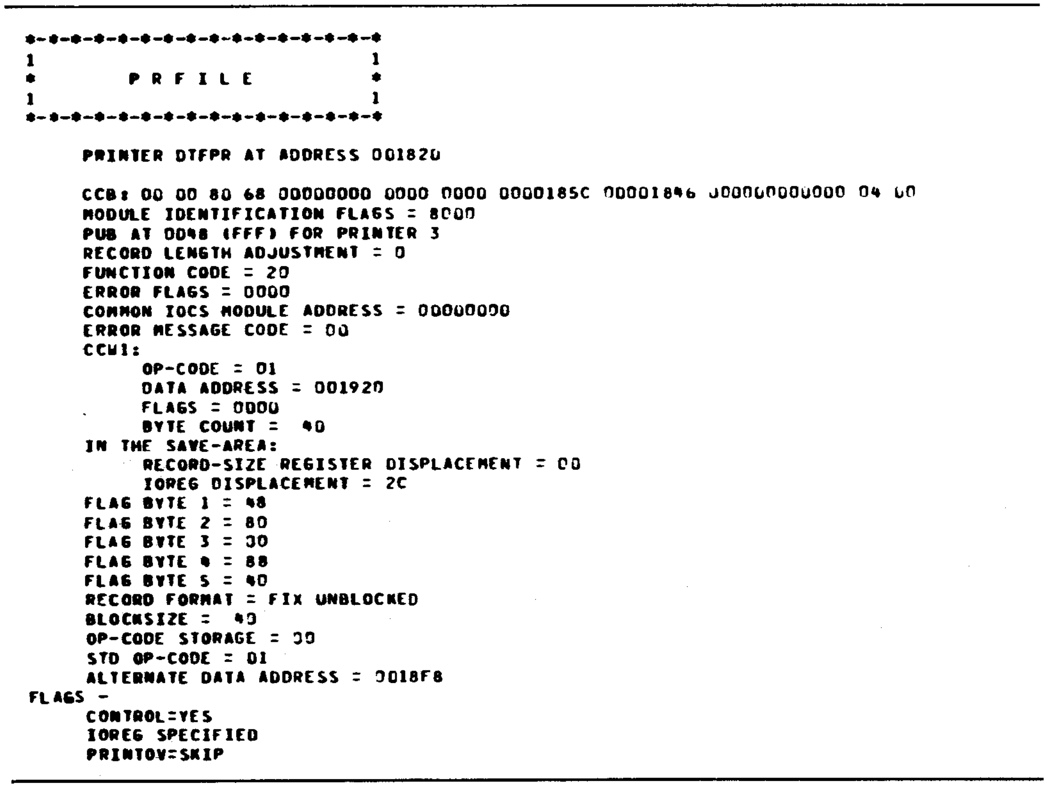
```
   OPTION-DUMP
   OPTION-JOBDUMP

 JOB CONTROL FLAGS
   JOB ABNORMALLY TERMINATED
   JOB TERMINATION BUSY
   ROLL OUT OF JOB INHIBITED
   PRINT SPOOL FILE GENERATED
   WTL BUFFERS INITIALIZED
   ACTIVE PHASE TABLE PRESENT

 JOB CONTROL INFORMATION
   JOB SCHEDULING PRIORITY -  LOW
   JOB CONTROL DIRECTORY DISC ADDRESS - 307/ 0/ 1
   NO. CYLS FOR ROLLOUT/JOBDUMP -    5
   JOBDUMP COPY RUNLIB DISC ADDRESS - 307/ 2/ 1

* * *  L O A D E R   S E A R C H   T A B L E  * * *

    SEARCH     LIBRARY     LIBRARY      PUB      FORMAT 2 LABEL    BEGIN SEARCH
    ORDER       NAME         VSN      ADDRESS        CC/H/R        BLOCK    BYTE

      1        SYSLOD      REL060       0C28       154/ 0/14       000001    00

      2        SYSRUN      REL060       0C28       154/ 1/32       000001    27


①
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
I                                                     I
*     T A S K   C O N T R O L   B L O C K    1        *
I                                                     I
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

     TASK CONTROL BLOCK AT ADDRESS  014110
          TASK KEY =  1
          ONLY TASK AT PRIORITY  8
          TCB FLAGS
               WAIT FOR TRANSIENT
               WAIT FOR CANCEL IN PROGRESS
               ORIGINAL PSW HAS BEEN SAVED
          PREAMBLE ADDRESS =  014000
          TRANSIENT ID/SVC CODE =  1C

* * *  T A S K   P S W  * * *

          PROGRAM STATUS WORD = C0160007  00001A44
               PROGRAM KEY = 1 , WHICH IS JOB COBJOB
               INTERRUPT CODE =07
               CONDITION CODE = 1
               INSTRUCTION ADDRESS = 001A44
               NONZERO INSTRUCTION LENGTH (6 BYTES)
               OPERATION: AP      INSTRUCTION: FA22 A008 A018
```

Figure 4-4. COBOL Dump Analysis Listings (Part 7 of 14)



```
          REG 0       REG 1       REG 2       REG 3       REG 4       REG 5       REG 6       REG 7
         FFFFFFF2    000017A8    0000177C    000005C8    8000055A    020018A8    010018F8    00001948
          REG 8       REG 9       REG A       REG B       REG C       REG D       REG E       REG F
         007FFFFF    007FFFFF    00001718    00002718    400019FA    00001990    40001A10    40001A10

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

         TERMINATION SVC  0A54
         ERROR STATUS CODE  0020
         ERROR/PSW ADDRESS =  000000


*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
1                               1
*        C D F I L E            *
1                               1
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

     CARD DTFCD AT ADDRESS 0017A8

     CCB: 00 00 80 68 00000000 0000 0000 000017E4 000017CE 000000000000 0C 00
     MODULE FLAGS = 8000
     PUB AT 0B88 (110) FOR READER 1
     END-OF-FILE ROUTINE ADDRESS = 001A18
     RECORD LENGTH ADJUSTMENT =    0
     FUNCTION CODE = 10
     ERROR FLAGS = 0000
     COMMON IOCS MODULE ADDRESS = 000728
     ERROR MESSAGE CODE = 00
     USER ERROR ROUTINE ADDRESS = 000698
     CCW1:
          OP-CODE = 02
          DATA ADDRESS = 001800
          FLAGS = 0000
          BYTE COUNT =  40
     IN THE SAVE-AREA:
          RECORD SIZE REGISTER DISPLACEMENT = 00
          IOREG DISPLACEMENT = 28
     FLAG BYTE 1 = 49
     FLAG BYTE 2 = 00
     FLAG BYTE 3 = 82
     FLAG BYTE 4 = 08
     FLAG BYTE 5 = 04
          I N P U T   F I L E
     RECORD FORMAT = FIX UNBLOCKED
     BLOCK SIZE =  40
     ALTERNATE DATA ADDRESS = 0018A8
     RECORD LENGTH =   39
     EOF-MASK-TABLE DISPLACEMENT = 04
FLAGS -
     STD MODE
     IORG SPECIFIED
```

Figure 4-4. COBOL Dump Analysis Listings (Part 8 of 14)

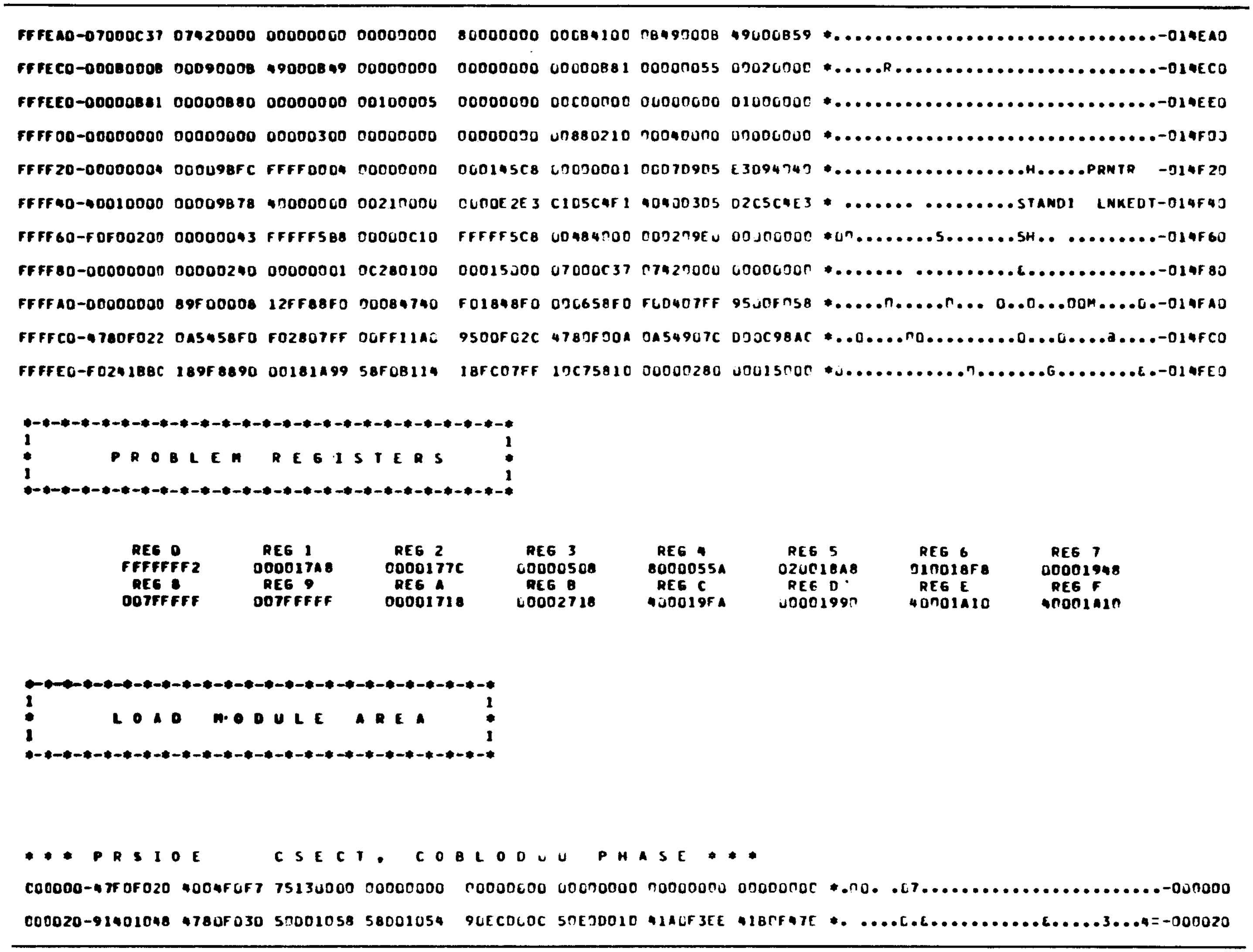


```
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
1                               1
*       P R F I L E             *
1                               1
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

     PRINTER DTFPR AT ADDRESS 001820

     CCB: 00 00 80 68 00000000 0000 0000 0000185C 00001846 000000000000 04 00
     MODULE IDENTIFICATION FLAGS = 8000
     PUB AT 0048 (FFF) FOR PRINTER 3
     RECORD LENGTH ADJUSTMENT = 0
     FUNCTION CODE = 20
     ERROR FLAGS = 0000
     COMMON IOCS MODULE ADDRESS = 00000000
     ERROR MESSAGE CODE = 00
     CCW1:
          OP-CODE = 01
          DATA ADDRESS = 001920
          FLAGS = 0000
          BYTE COUNT =  40
     IN THE SAVE-AREA:
          RECORD-SIZE REGISTER DISPLACEMENT = 00
          IOREG DISPLACEMENT = 2C
     FLAG BYTE 1 = 48
     FLAG BYTE 2 = 80
     FLAG BYTE 3 = 00
     FLAG BYTE 4 = 88
     FLAG BYTE 5 = 40
     RECORD FORMAT = FIX UNBLOCKED
     BLOCKSIZE =  40
     OP-CODE STORAGE = 00
     STD OP-CODE = 01
     ALTERNATE DATA ADDRESS = 0018F8
FLAGS -
     CONTROL=YES
     IOREG SPECIFIED
     PRINTOV=SKIP
```

Figure 4-4. COBOL Dump Analysis Listings (Part 9 of 14)

```
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
I                                           I
*        P R O L O G U E    A R E A         *
I                                           I
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*


* * *  P R E A M B L E  * * *

000000-C3D6C2D1 D6C24040 00900000 0000C000  0000B000 00C01A9E 00015000 00000354 *COBJOB  ..................&.....-014000
000020-C3D6C2D3 D6C4F0F0 C3D6C2D3 D6C4F0F0  79090600 00C00490 00000BB8 00000000 *COBLOD00COBLOD00................-014020
000040-00000000 00000000 00000000 00000000  00000000 00000000 00000000 060001D8 *...............................0-014040
000060-40404040 0790906C 304F00F9 40F7F9F2  F4F90C00 20440908 01F20305 01330102 *    ...%. .9 79249.......2......-014060
000080-00008000 00000000 01330100 0001000C  0C280000 00C00001 009A0E00 0C280027 *................................-014080
0000A0-00000001 009A2001 00000000 00000000  00000000 00C00000 00000000 00000000 *................................-0140A0
0000C0-0C090224 08014110 00001000 04000000  00000000 00C00000 00000180 00000004 *................................-0140C0
0000E0-00000000 00000090 00C00000 FFFFFA08  FFFFF588 02C20000 00000000 00000000 *...................5............-0140E0
000100-00000000 00000000 00000000 00000000                                      *................                -014100


* * *  T C B  * * *

000000-10014110 00000300 20014110 00000100  00002456 00C14000 1C000000 00000000 *...................... .........-014110
000020-C0160007 D0001A44 FFFFFFF2 000017A8  0000177C 00000508 8000055A 020018A8 *...........2.......@.......].....-014130
000040-010018F8 00001948 007FFFFF 007FFFFF  00001718 00002718 400019FA 00001990 *...8......"..."........... .......-014150
000060-40001A10 40001A10 00000000 00000000  00000000 00000000 00000000 00000000 * ... ...........................-014170
000080-00000000 0A540020 00000000 C0160007  D0001A44 00000000 01330301 00100000 *................................-014190
0000A0-00FF43F4 00000000 00000000 00000000  00000000 00000000 00000000 00000000.*...4...........................-0141B0
0000C0-00000000 00000000                                                        *........                         -014100
```

Figure 4-4. COBOL Dump Analysis Listings (Part 10 of 14)



```
FFFEA0-07000C37 07420000 00000000 00000000  80000000 000B4100 0B490008 49000B59 *..............................-014EA0
FFFEC0-00080008 00090008 49000849 00000000  00000000 00000881 00000055 00020000 *.....R........................-014EC0
FFFEE0-00000881 00000880 00000000 00100005  00000000 00C00000 00000000 01000000 *..............................-014EE0
FFFF00-00000000 00000000 00000300 00000000  00000000 00880210 00040000 00000000 *..............................-014F00
FFFF20-00000004 00009BFC FFFF0004 00000000  000145C8 00000001 0007D9D5 E3D94040 *..................H.....PRNTR  -014F20
FFFF40-40010000 00009B78 40000000 00210000  0000E2E3 C1D5C4F1 40400305 D2C5C4E3 * ....... .........STANDI  LNKEDT-014F40
FFFF60-F0F00200 00000043 FFFFF5B8 00000C10  FFFFF5C8 00484000 000209E0 00000000 *00........5.......SH.. .........-014F60
FFFF80-00000000 00000240 00000001 0C280100  00015000 07000C37 07420000 00000000 *....... ..........&.............-014F80
FFFFA0-00000000 89F00008 12FF88F0 00084740  F01848F0 000658F0 F0D407FF 9500F058 *.....0.....0... 0..0...00M....0.-014FA0
FFFFC0-4780F022 0A5458F0 F02807FF 00FF11AC  9500F02C 4780F00A 0A54907C D00C98AC *..0....00.........0...0....@....-014FC0
FFFFE0-F0241BBC 189F8890 00181A99 58F0B114  18FC07FF 10C75810 00000280 00015000 *0.............0.......G........&.-014FE0

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
I                                               I
*       P R O B L E M    R E G I S T E R S      *
I                                               I
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*


         REG 0      REG 1      REG 2      REG 3      REG 4      REG 5      REG 6      REG 7
        FFFFFFF2   000017A8   0000177C   00000508   8000055A   020018A8   010018F8   00001948
         REG 8      REG 9      REG A      REG B      REG C      REG D`     REG E      REG F
        007FFFFF   007FFFFF   00001718   00002718   400019FA   00001990   40001A10   40001A10


*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
I                                             I
*       L O A D    M O D U L E    A R E A     *
I                                             I
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*


* * *  P R S I O E       C S E C T ,   C O B L O D 0 0    P H A S E  * * *

C00000-47F0F020 4004F0F7 75130000 00000000  00000600 00000000 00000000 0000000C *.00. .07........................-000000
000020-91401048 4780F030 50001058 58001054  90ECD00C 50E0D010 41A0F3EE 41B0F47E *. ....C.&...........&.....3...4=-000020
```

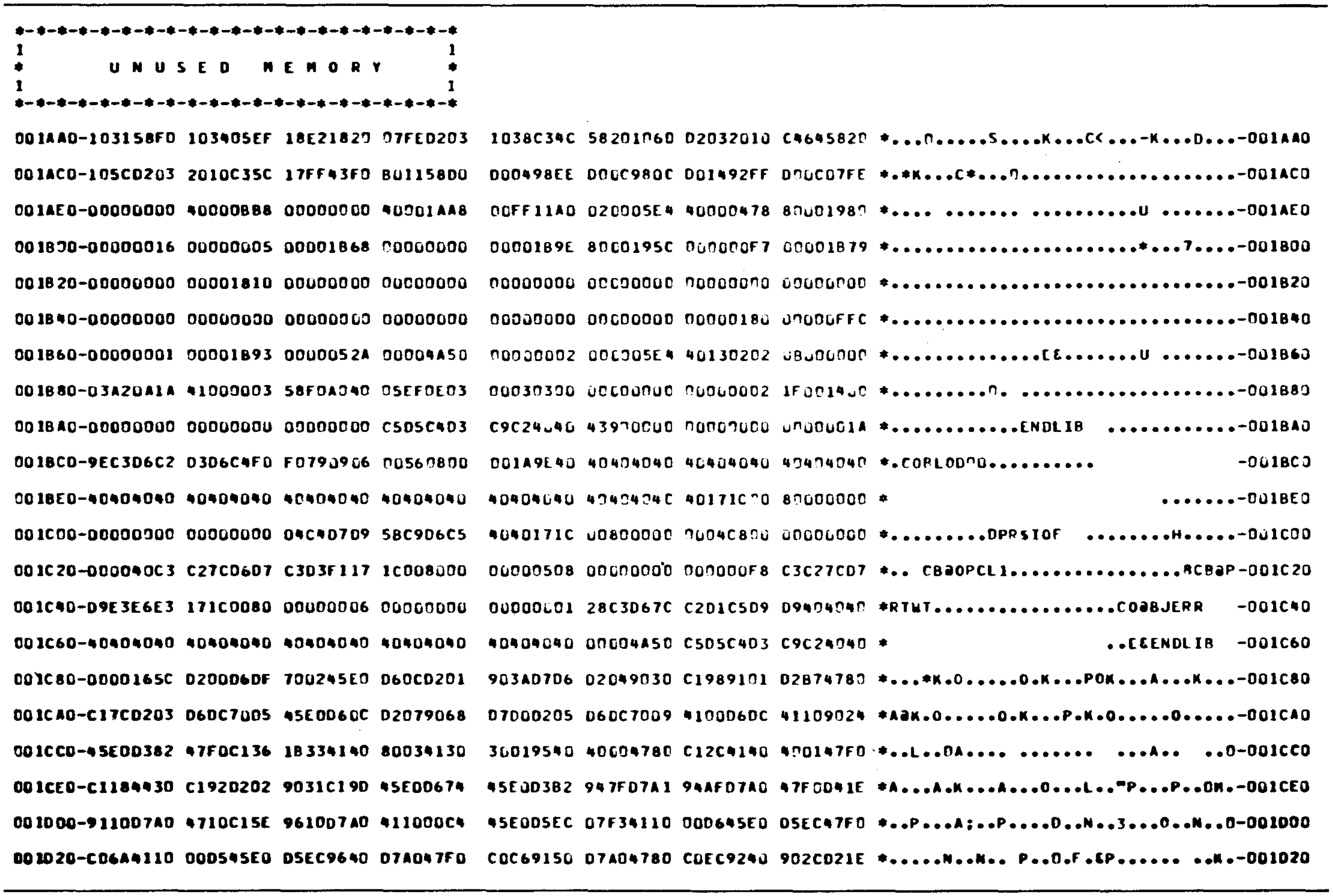
Figure 4-4. COBOL Dump Analysis Listings (Part 11 of 14)

```
000020-10560201 105AFD84 41401014 50401054  92041054 0AC09180 10024710 FD470A01 *..K..J... ..& ............... ..-001448
000040-0201104E 1018947F 104E941F 104F4140  104E5040 10549202 10589221 10540A00 *K..+..."..+... . .+L .....$.......-001468
000060-91801002 4710FD6A 0A010201 10561061  0A749AFF 105ED201 105A104C 92011054 *...........K..../.....;K..J.<....-001488
000080-47F0F0F0 00069102 10464780 FD9A9180  10494780 FD9A9680 10035840 10541826 *.∩.∩....................... ....-0014A8
0000A0-1A469140 10484710 FD024840 FDCED501  4000FD00 4770FDEC 4B20FDCE 4920FDCE *... .....K. ..N. ..............-0014C8
0000C0-4720FDAA 47F0FDEC 00000000 00040002  40404640 FDCCD503 4000FDC8 4770FDEC *.....∩..........  . ..N. ..H....-0014E8
0000E0-4820FDCC 4920FDCC 4720FDD2 4020105A  0A0047F0 FE1045BC FEC805E8 91801049 *............K ..J...∩.....H.Y....-001508
000E00-4780FE0C 4590FF2A 4590FF78 4580FEE0  94FB1049 9180104B 4780FE44 D501105E *...........................N..;-001528
000E20-F4524770 FE449180 10024710 FE300A01  91401002 4710F336 D5001026 1027078C *4................ ....3.N.......-001548
000E40-45E0F354 91101046 4710FE6A 9180104B  4780FE5C 45E0F2C8 47F0FE6A 05E89120 *..3.................*..2H.∩...Y..-001568
000E60-104A4710 FE6A4590 FF2A07FC 96801060  9101104A 078E9110 10474780 FEC69104 *.[..............-...[.........F..-001588
000E80-10604710 FE8E9104 10494710 FEA69120  10474780 FE9E9610 106047F0 FEC294EF *.-.......................-.∩.B..-0015A8
000EA0-106047F0 FEC29120 10474780 FEC29120  10604710 FEBE9620 106047F0 FEC29610 *.-.∩.B.......B...-........-.∩.B..-0015C8
000EC0-106094CF 104707FE 4120103C 5020100C  0A0007FB 94CF1054 960B1054 47F0FEF0 *.-...........&.................∩.∩-0015E8
000EE0-94041054 96191054 96081060 47F0FEF0  41201054 5020100C 91101060 4780FF08 *............-.∩.∩....&.......-....-001608
000F00-94EF1060 96201054 91081060 4780FF18  94F71060 45E0F134 0A009120 10600768 *...-.......-.....7.-..1.......-..-001628
000F20-96101060 940F1060 07FB9180 10490789  5840103C 58201078 4450FF72 9120104A *...-...-......... .......&.....[-001648
000F40-4780FF6C 91041048 4710FF54 41202050  47F0FF58 412020A0 9A011066 91031067 *...%............&.∩..............-001668
000F60-47E0FF6C 94FC1067 41201084 50201078  07F9D200 20004000 91041048 4710FF9A *...%.........&.....9K... .........-001688
000F80-41401174 41201264 4450FF72 58401054  41201174 4450FF72 07F94140 12644120 *. ........&... ........&...9. ....-0016A8
000FA0-13044450 FF725840 41606004 47F0F5F6  FF7207F9 007892F5 74A34110 749E41F0 *...&... .--..∩56...9...5.......∩-0016C8
000FC0-E0004190 E00047F0                                                        *.......∩                          -0016E8

                 3

*** COBOBJ00 CSECT, COBLOD00 PHASE ***
000000-05F045E0 F00607FC 98ADF012 9859A060  04A007FE 00001718 00002718 000019E0 *.∩..0.....0....-.................-0016F0
000020-00001990 00000000 00000000 00000000  00650466 96F0E590 927A3002 927A3005 *.....................0V..:...:..-001710
000040-00232C00 E5980201 3003E59A D2013006  00001A6E 000019F8 000017A8 000017A8 *....V.K...V.K......>...8.........-001730
000060-00001820 00001820 00000600 00000508  000004C8 000004E4 007FFFFF 00001718 *..................H...U.".......-001750
```

Figure 4-4. COBOL Dump Analysis Listings (Part 12 of 14)

```
000080-0000178C 00001990 020018A8 010018F8  00001948 007FFFFF 007FFFFF 00000F00 *...............8.....".."......-001770
0000A0-00000000 F004F004 4770EA32 41000206  45F074D2 00C00000 00008068 00000900 *....0.0..........0.K............-001790
0000C0-00000000 000017E4 000017CE 00000000  00000C00 C3C4C6C9 D3C54040 80000B88 *........U............CDFILE  ....-001780
0000E0-00001A18 000008F0 01100000 00000728  00000698 02C018D0 00000028 00284930 *.......0.......................-0017D0
000100-82080404 00280000 000018A8 00000000  00000000 00C00000 00000000 00270400 *...............................-0017F0
000120-00000000 00000000 00000000 00000000  00008368 00C00000 00000000 0000185C *..............................*-001810
000140-00001846 00000000 00000400 D7D9C6C9  D3C54040 80C00048 00000000 000004F0 *.............PRFILE  ...........0-001830
000160-F1200000 00000000 00000698 01001920  00000028 002C488C 3C88* 04 00280001 *1......................  .....-001850
                                                                  (8)
000180-010018F8 00000000 00000000 00000000  00000000 00C00000 00000000 00000000 *...8....................ADD1  ADD2 001870
0001A0-00000000 00000000 00000000 00000000  00000000 00000000 F2F3F240 F3F6F540 *.........................232 365 -001890
0001C0-4040D2C5 D3D3C5E8 6B40E3D6 D4404040  40404040 40404040 40404040 40404040 *  KELLEY, TOM  (10)   CDBUFF  -001880
0001E0-F1F0F340 40F3F0F2 4040E5C9 D5C3C5D5  E36B40C7 C5D6D9C7 C5404040 40404040 *103  302  VINCENT, GEORGE       -0018D0
000200   TO   00021F     SAME AS LAST WORD    0018F0  TO  0019JF
000220*40404040 40404040 40404040 40404040  F0F4F8F5 404040C3 C8C1D7D4 C1D56B40 *                  0485  CHAPMAN, -001910
000240-C7C5D6D9 C7C54040 40404040 40404040  40404040 40404040 4EF0F0F0 F0F0F0F0 *GFORGE                   +0000000-001930
000260-F0F0F0F0 F0F0F0F0 F0F0F0F0 F0F0F0F0  F0F0F0F0 F0F0F0F0 F04E4040 40404040 *0000000000000000000000000+       -001950
000280-40404040 40404040 40404040 40404040  40404040 40404040 40300700 07000700 *                          .......-001970
0002A0-103841E0 C25E50E0 10385820 40001A10  40001A10 FFFFFFF2 000017A8 0000177C *....B;&.....  ...  ......2.......@-001990
0002C0-00000508 8000055A 020018A8 010018F8  00001948 007FFFFF 007FFFFF 00001718 *........J........8....."..."......-001980
0002E0-00002718 400019FA 0000189C 00000000  5810A034 58F0A048 05EF5810 A03C58F0 *....  ................0.........0-001900
000300-A04805EF 9066A064 05C05810 A03441F0  C01E50F0 10289210 103158F0 103405EF *.................0..&0.......0....-0019F0
000320-9055A060 47F0C024 58F0A028 05EFD203  6000703E D2236004 6000F222 A0185000 *...-.0...0....K.-...K.-.-.2...&.-001A10
000340-94FCA01A F222A008 500594FC A00AFA22  A008A018 F3326000 A00896F0 6003D210 *....2...&...........3.-....0-.K.-001A30
000360-6007600A 41000001 5810A03C 4120A064  58F0A044 05EF4000 58F0A02C 05EF05C0 *-.&...............0....  ..0......-001A50
000380-5810A034 58F0A04C 05EF5810 A03C58F0  A04C05EF 5810C01C 47F0C020 5CC1D3D3 *.....0.<.......0.<.......0..*ALL-001A70
0003A0-0A27BA1A 41000003 58F0A040 05EF9220                                      *.........0.  ....                -001A90
```

Figure 4-4. COBOL Dump Analysis Listings (Part 13 of 14)



```
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
I                                       I
*       U N U S E D   M E M O R Y       *
I                                       I
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

001AA0-103158FD 103405EF 18E21820 07FED203  1038C34C 58201060 D2032010 C4645820 *...0.....S....K...C<...-K...D...-001AA0
001AC0-105CD203 2010C35C 17FF43F0 B01158D0  000498EE D00C980C D01492FF D00CD7FE *.*K...C*...0....................-001AC0
001AE0-00000000 40000B88 00000000 40001AA8  00FF11A0 020005E4 40000478 80001980 *.... ...... ..........U .......-001AE0
001B00-00000016 00000005 00001868 00000000  0000189E 80C0195C 000000F7 00001B79 *.....................*...7....-001B00
001B20-00000000 00001810 00000000 00000000  00000000 00C00000 00000000 00000000 *...............................-001B20
001B40-00000000 00000000 00000000 00000000  00000000 00000000 00000180 00000FFC *...............................-001B40
001B60-00000001 00001B93 0000052A 00004A50  00000002 00C005E4 40130202 0B000000 *..............C&........U .......-001B60
001B80-03A20A1A 41000003 58F0A040 05EF0E03  00030300 00C00000 00000002 1F001400 *.........0. ......................-001B80
001BA0-00000000 00000000 00000000 C5D5C4D3  C9C24040 43900000 00000000 0000001A *............ENDLIB  ............-001BA0
001BC0-9EC3D6C2 D3D6C4F0 F0790906 00560800  001A9E40 40404040 40404040 40404040 *.CORLOD00..........             -001BC0
001BE0-40404040 40404040 40404040 40404040  40404040 4040404C 40171C00 80000000 *                        .......-001BE0
001C00-00000000 00000000 04C40709 5BC906C5  4040171C 00800000 0004C800 00000000 *.........DPRSIOF  ........H.....-001C00
001C20-000040C3 C27CD6D7 C3D3F117 1C008000  00000508 00000000 000000F8 C3C27CD7 *.. CB@OPCL1................ACB@P-001C20
001C40-D9E3E6E3 171C0080 00000006 00000000  00000C01 28C3D67C C201C5D9 D9404040 *RTWT.................CO@BJERR   -001C40
001C60-40404040 40404040 40404040 40404040  40404040 00004A50 C5D5C4D3 C9C24040 *                      ..C&ENDLIB  -001C60
001C80-0000165C D200D6DF 700245E0 D60CD201  903AD7D6 02049030 C1989101 D2B74780 *...*K.O.....O.K...POK...A...K...-001C80
001CA0-C17CD203 D6DC7005 45E0D60C D2079068  07000205 D60C7009 4100D6DC 41109024 *A@K.O.....O.K...P.K.O.....O.....-001CA0
001CC0-45E0D382 47F0C136 1B334140 80034130  30019540 40004780 C12C4140 40014 7F0 *..L..OA.... ......  ...A..  ..O-001CC0
001CE0-C1184430 C192D202 9031C19D 45E00674  45E0D382 947FD7A1 94AFD7A0 47F0D41E *A...A.K...A...O...L.."P...P..OM.-001CE0
001D00-9110D7A0 4710C15E 9610D7A0 411000C4  45E0D5EC 07F34110 000645E0 05EC47F0 *..P...A;..P....D..N..3...O..N..O-001D00
001D20-C06A4110 000545E0 D5EC9640 D7A047F0  C0C69150 07A04780 C0EC9240 902CD21E *.....N..N.. P..O.F.&P...... ..K.-001D20
```

Figure 4-4. COBOL Dump Analysis Listings (Part 14 of 14)

4.4. RPG II Dump Analysis

In this subsection, we analyze a JOBDUMP taken from an RPG II program named RPGOBJ. This program reads two numbers from a card like the programs analyzed in 4.2 and 4.3. Unlike them, it divides one number by the other and prints the results along with a character string taken directly from the input card.

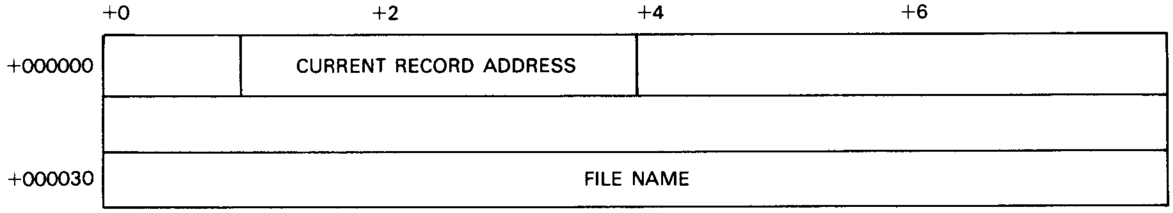
4.4.1. Materials Used

The materials used in this analysis are contained in 4.4.5. They are as follows:

- Job log for job RPGCHG (the job which ran RPGOBJ)
- RPG II compilation for RPGOBJ
- Edited JOBDUMP

4.4.2. Program-File Interface - the Input/Output Request Block

In this analysis, we will look at a software structure used by RPG II as an interface between an RPG II program and a file. That interface is called the input/output request block (IORB). The IORB is 56 bytes long and contains information that is useful in debugging an RPG II program. For our present purposes the format of the IORB is as follows:



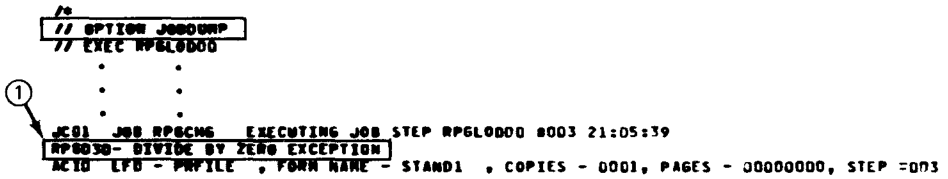
An IORB exists within your RPG II program for each file used by the program. More information on the IORB can be found in the *System Messages Reference Manual* (UP-8076).

4.4.3. Program Check Island Code

Another OS/3 facility that we will discuss in this analysis is the program check island code. This code is a software feature that permits users to handle program exceptions with their own routines rather than simply allowing their programs to fail. Several language processors, RPG II included, automatically include island code in load modules, mainly to print error messages and perform other recovery functions when a program fails. As we will see later, the program check island code is designed to save the hardware environment of a failing program in such a way as to let us see what caused the failure. More information on island codes can be found in the *Supervisor Technical Overview* (UP-8831).

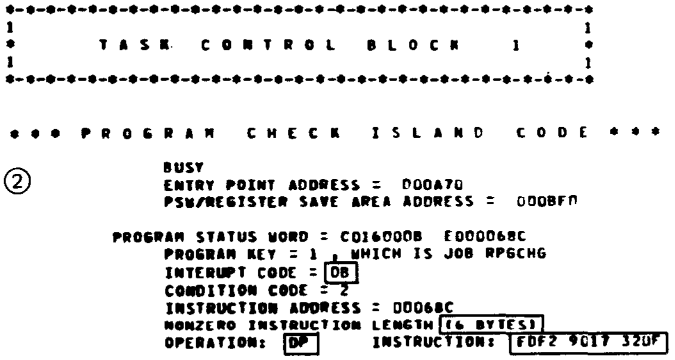
4.4.4. Analysis

To determine what caused RPGOBJ to fail, we proceed as follows:



```
 /*
 // OPTION JOBDUMP
 // EXEC RPGLODOO
        .     .
①      .     .
        .     .
 JC01  JOB RPGCHG   EXECUTING JOB STEP RPGLODOO #003 21:05:39
 RPG030- DIVIDE BY ZERO EXCEPTION
 ACIO  LFD - PRFILE  , FORM NAME - STAND1  , COPIES - 0001, PAGES - 00000000, STEP =003
```

Here we have loaded and executed our load module RPGLOD00, containing RPGOBJ. The program, however, fails when the RPG030 message ① is printed in the job log. By including an // OPTION JOBDUMP statement in the JCL runstream immediately before the // EXEC RPGLOD00 statement, we can cause OS/3 to generate a JOBDUMP. In doing so we look at the JOBDUMP narrative.



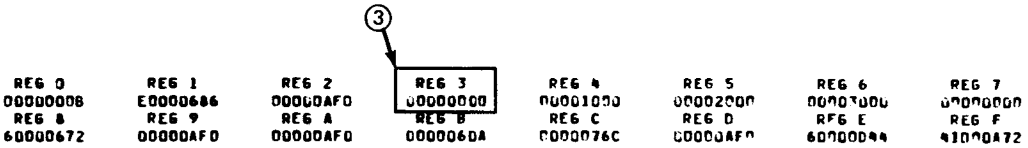
```
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
I                                                   I
*      T A S K   C O N T R O L   B L O C K    1     *
I                                                   I
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

* * *  P R O G R A M   C H E C K   I S L A N D   C O D E  * * *

                  BUSY
 ②                ENTRY POINT ADDRESS =  000A70
                  PSW/REGISTER SAVE AREA ADDRESS =  000BF0

             PROGRAM STATUS WORD = C016000B  E000068C
                  PROGRAM KEY = 1 , WHICH IS JOB RPGCHG
                  INTERUPT CODE = 0B
                  CONDITION CODE = 2
                  INSTRUCTION ADDRESS = 00068C
                  NONZERO INSTRUCTION LENGTH (6 BYTES)
                  OPERATION: DP       INSTRUCTION: FDF2 9017 320F
```

At ② we see that a program exception occurred, which activated our program check island code. The major purpose of the island code is to generate the RPG030 message indicated by ① . But the island code also saves the PSW under which the error occurred. We can thus use this PSW to determine that:

- The failing instruction was a divide decimal (DP)

- The interrupt code of 0B indicates an oversize quotient, most commonly generated when dividing by zero

Since operand 2 of the DP instruction represents the denominator, we can confirm that it contains zero, first going to ③.

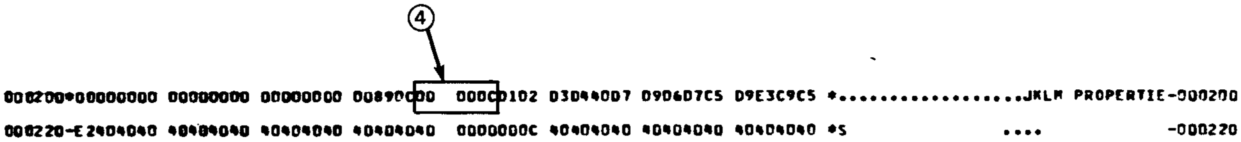


```
 REG 0      REG 1      REG 2      REG 3      REG 4      REG 5      REG 6      REG 7
00000008   E0000686   00000AF0   00000000   00001000   00002000   00003000   00000000
 REG 8      REG 9      REG A      REG B      REG C      REG D      REG E      REG F
60000672   00000AF0   00000AF0   000006DA   C000076C   00000AF0   60000D44   41000A72
```

Operand 2 of the DP instruction is in base/displacement form using register 3 ③. From register 3, the effective address of operand 2 can then be determined:

0000	Register 3 contents
+ 20F	Displacement
020F	Operand 2 address

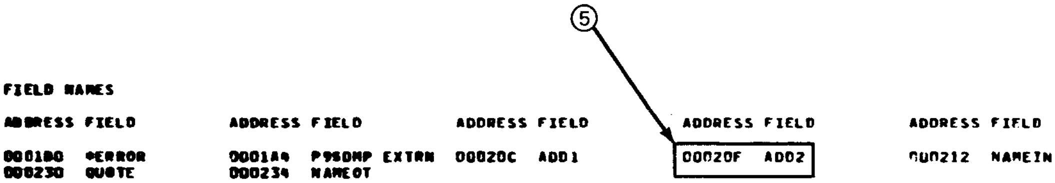
Having found its address we now see what operand 2 contains.



```
000200*00000000 00000000 00000000 00890C00  000C0102 03044007 09060 7C5 D9E3C9C5 *..................JKLM PROPERTIE-000200
000220-E2404040 40404040 40404040 40404040  0000000C 40404040 40404040 40404040 *S               ....              -000220
```

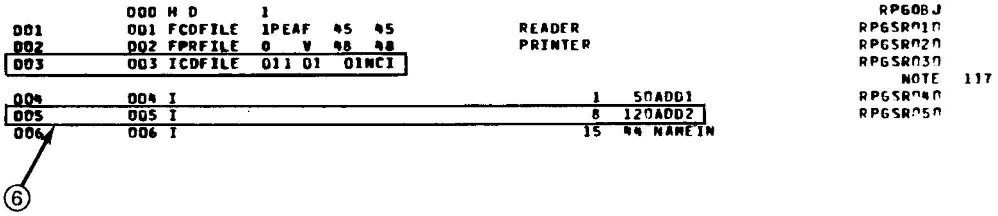
In the JOBDUMP, ④ points to address 20F. The DP instruction specifies that operand 2 is three bytes long, so we highlight addresses 20F-211 to show that operand 2 does indeed contain a packed decimal value of zero.

The next question is how the zero value got into operand 2. To help answer that question, we turn to the RPG II compilation listing that generated RPGOBJ.



```
FIELD NAMES

ADDRESS FIELD          ADDRESS FIELD          ADDRESS FIELD          ADDRESS FIELD          ADDRESS FIELD

000180  *ERROR         0001A4  P9SDMP EXTRN  00020C  ADD1          00020F  ADD2           000212  NAMEIN
000230  QUOTE          000234  NAMEOT
```

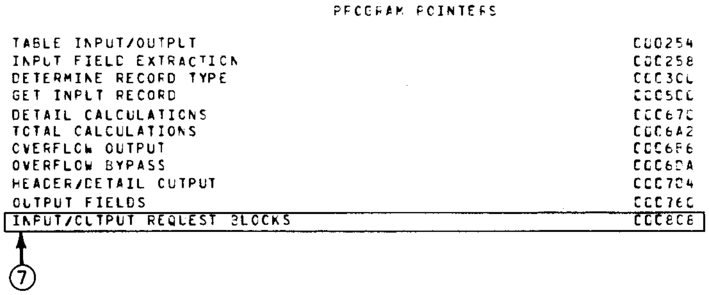
The field name list in our RPG II listing shows that our operand 2 address of 20F corresponds to a field within our program named ADD2 ⑤. We turn to the RPG II source listing itself to see what ADD2 is.



```
                 000 H D         1                                                           RP60BJ
001              001 FCDFILE   1PEAF   45   45            READER                          RPGSR010
002              002 FPRFILE   0    V   48   48            PRINTER                         RPGSR020
003              003 ICDFILE   011 01   01NCI                                              RPGSR030
                                                                                               NOTE   117
004              004 I                                                     1    50ADD1     RPGSR040
005              005 I                                                     8   120ADD2     RPGSR050
006              006 I                                                    15    44 NAMEIN
```

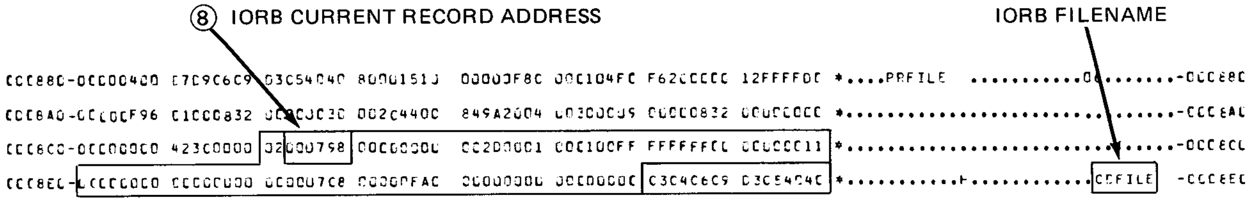
Line 005 of the source program ⑥ shows that ADD2 is a 5-byte decimal number contained in bytes 8-12 of the input record for card file CDFILE (defined in line 003). Knowing this, we can then look for the input card that presented a zero value in the ADD2 field, thus causing the program to fail.

To pick out the error card we first find the data read into the job from that card. The RPG II compiler listing and the *System Messages Reference Manual* (UP-8076) can both help us here. Our intention is to find the IORB associated with file CDFILE. To do so we first look at the RPG II compiler listing under the heading "PROGRAM POINTERS":



```
                                 PFCGFAM FCINTEFS

TABLE INPUT/OUTPLT                                                  CU0254
INPLT FIELC EXTRACTICN                                              COC258
DETERMINE RECOFD TYPE                                               CCC3CL
GET INPLT RECORD                                                    CCC5CC
DETAIL CALCULATICNS                                                 CCC67C
TCTAL CALCULATIONS                                                  CCC6A2
CVERFLCW OUTPUT                                                     CCC6F6
OVERFLOW BYPASS                                                     CCC6DA
HEACER/CETAIL CUTPUT                                                CCC7C4
OLTPUT FIELDS                                                       CCC76C
INPUT/CLTPUT REQLEST BLOCKS                                         CCC8C8
```

As we can see the input/output request blocks ⑦ are located starting at address 8C8. We go next to that address and follow the IORB format shown in 4.4.2. RPG II always arranges IORBs in the same sequence as the file description statements for their respective files. Because file CDFILE is the first file defined in the program, its IORB occupies the first 56-byte IORB slot, at address 8C8:

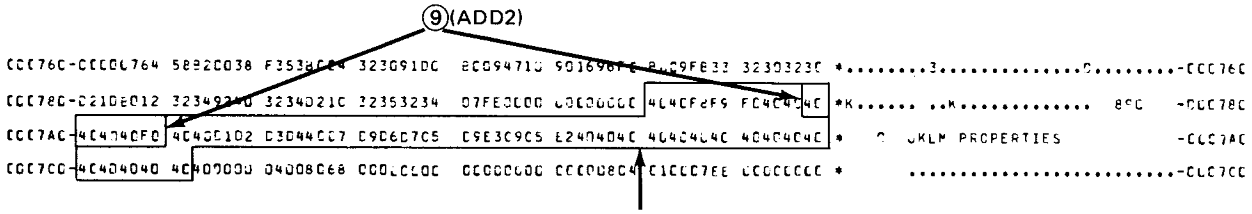


8 IORB CURRENT RECORD ADDRESS
IORB FILENAME

```
CCC880-0CC00400 C7C9C6C9 D3C54040 80001510  00003F8C 00C104FC F62CCCCC 12FFFFD0 *....PRFILE  ............0.........-0CC880
CCC8A0-CCC00F96 C1CCC832 0C0C0C3C 002C440C  849A2004 C03C0C09 C0CCC832 C0CCCCCC *..................................-CCC8A0
CCC8C0-0CC000C0 423C0000 32000798 00C0000C  CC2D0001 C0C10CFF FFFFFFCC CCCCCC11 *..................................-0CC8C0
CCC8E0-CCCCC0C0 CCCCCU00 0C00U7C8 0000PFAC  00000000 00C0000C C3C4C6C9 D3C54040 *............F.............CDFILE   -CCC8E0
```

Bytes 1-3 ⑧ of the CDFILE IORB hold address 798, the current record address.

At address 798 we find:



```
000760-00000764 58920038 F3538024 32309100  80094710 9016906E 8009FE33 32303230 *........3...............0.........-000760
000780-0210E012 32349240 32340210 32353234  07FE0000 00000000 4040F2F9 F0404040 *K...... ..K..............  8°0   -000780
0007A0-404040F0 4040D1D2 D3D44007 D9D6D7C5  D9E3C9C5 E2404040 40404040 40404040 *   0  JKLM PROPERTIES            -0007A0
0007C0-40404040 40400000 04008068 00000000  00000000 00000804 C10007EE 00000000 *       ........................-0007C0
```

⑨(ADD2)

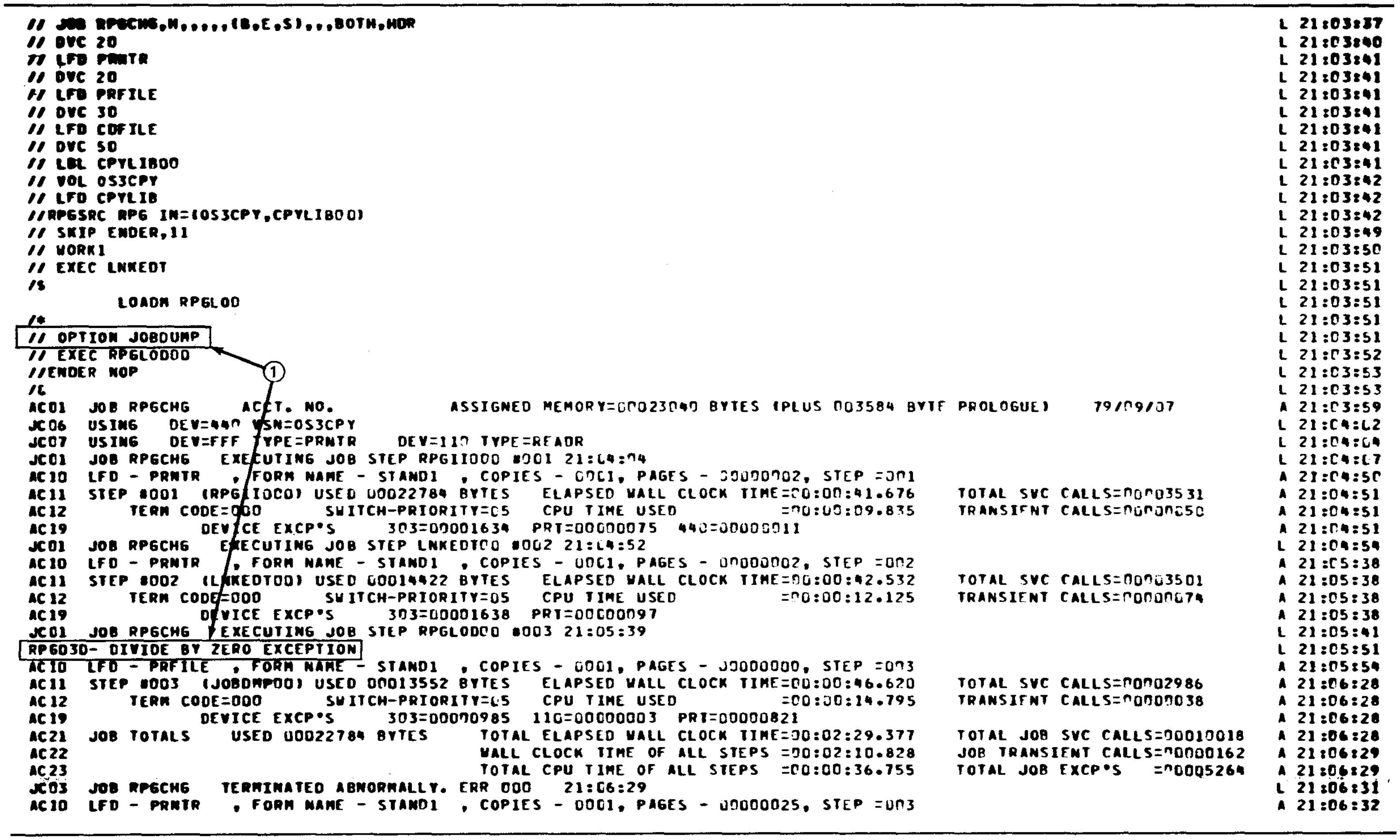
CDFILE CURRENT RECORD

The large block indicates the CDFILE record most recently released to the program, and ⑨ points to bytes 8-12 within the record, the field defined by RPG II field AD2 (see ⑥). We see that ADD2 does contain a value of zero, perhaps a mispunch. We can also tell, from the interpreted dump in the right column, just which input card (JKLM PROPERTIES) contained the faulty data.

4.4.5. Dump Analysis Materials

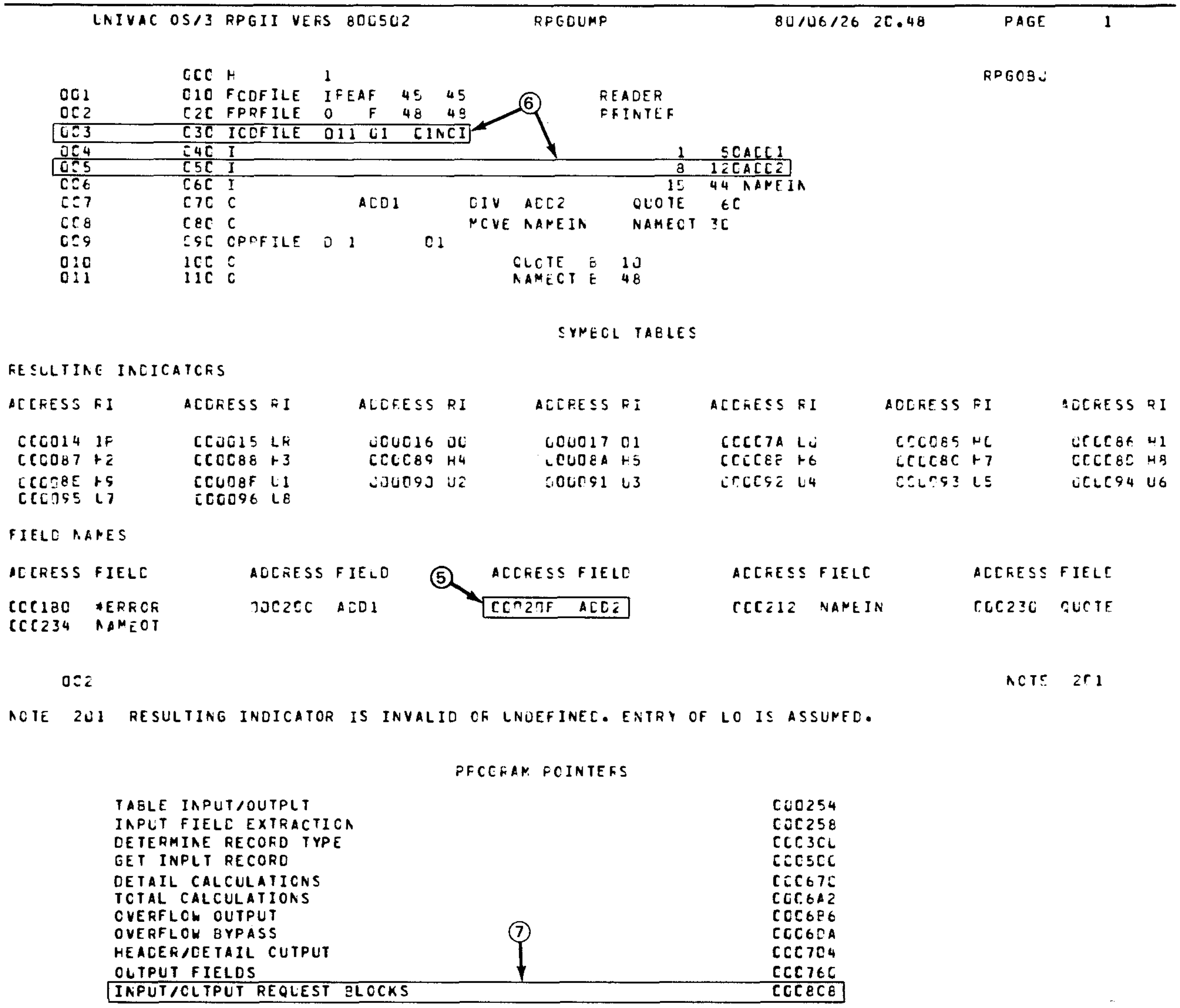
On the following pages is Figure 4-5, the edited printout from which the dump analysis of 4.4.4 is taken. The following list indicates where in Figure 4-5 each of the pointers of 4.4.4 can be found.

①	part 1
②	part 5
③	part 6
④	part 7
⑤	part 2
⑥	part 2
⑦	part 2
⑧	part 9
⑨	part 8



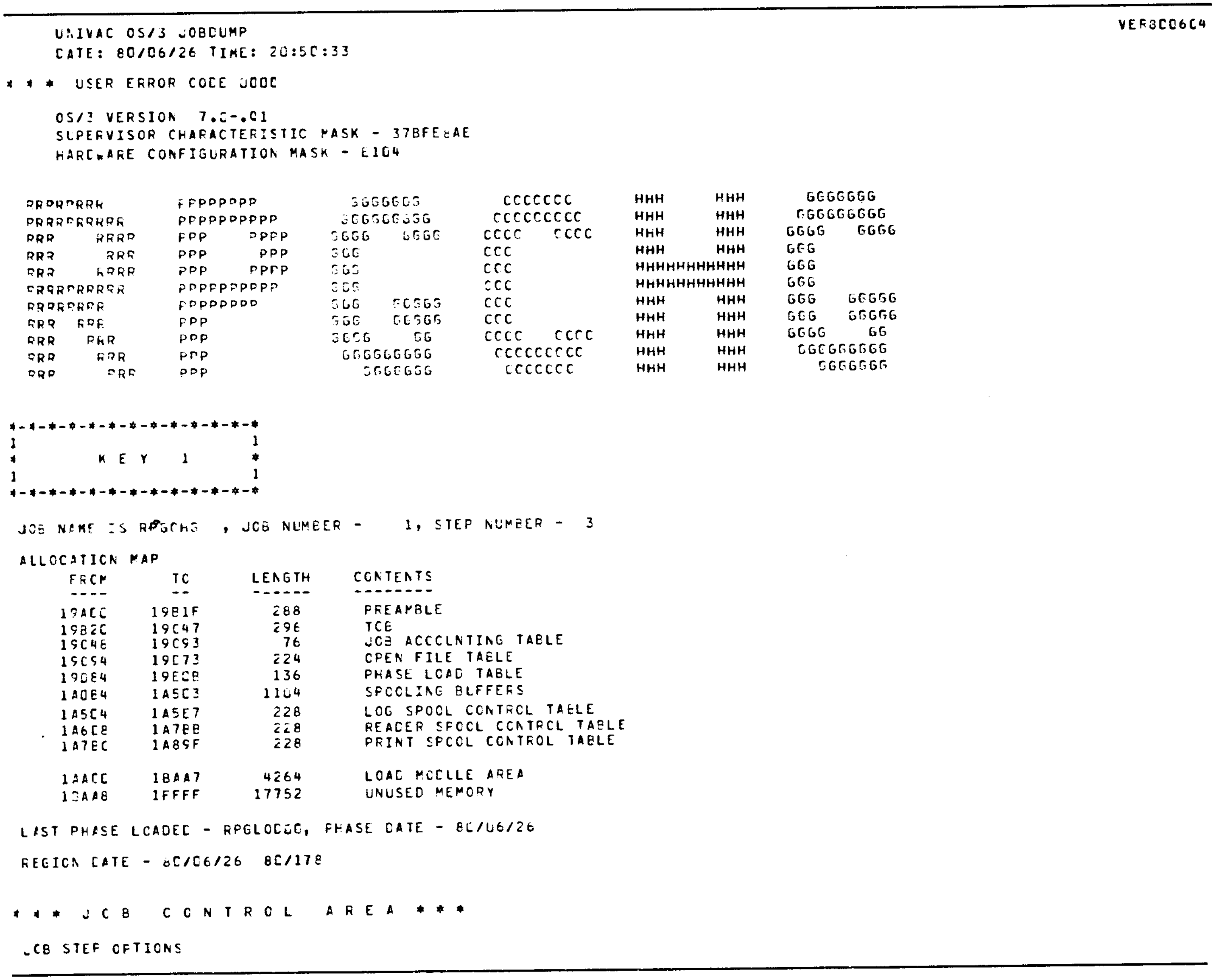
```
// JOB RPGCHG,M,,,,,(B,E,S),,,BOTH,HDR                                                                                           L 21:03:37
// DVC 20                                                                                                                        L 21:03:40
// LFD PRNTR                                                                                                                     L 21:03:41
// DVC 20                                                                                                                        L 21:03:41
// LFD PRFILE                                                                                                                    L 21:03:41
// DVC 30                                                                                                                        L 21:03:41
// LFD CDFILE                                                                                                                    L 21:03:41
// DVC 50                                                                                                                        L 21:03:41
// LBL CPYLIB00                                                                                                                  L 21:03:41
// VOL OS3CPY                                                                                                                    L 21:03:42
// LFD CPYLIB                                                                                                                    L 21:03:42
//RPGSRC RPG IN=(OS3CPY,CPYLIB00)                                                                                                L 21:03:42
// SKIP ENDER,11                                                                                                                 L 21:03:49
// WORK1                                                                                                                         L 21:03:50
// EXEC LNKEDT                                                                                                                   L 21:03:51
/$                                                                                                                               L 21:03:51
       LOADM RPGLOD                                                                                                              L 21:03:51
/*                                                                                                                               L 21:03:51
// OPTION JOBDUMP                                                                                                                L 21:03:51
// EXEC RPGLOD00                                                                                                                 L 21:03:52
//ENDER NOP                                                                                                                      L 21:03:53
/&                                                                                                                               L 21:03:53
AC01  JOB RPGCHG      ACCT. NO.          ASSIGNED MEMORY=00023040 BYTES (PLUS 003584 BYTE PROLOGUE)    79/09/07              A 21:03:59
JC06  USING    DEV=440 VSN=OS3CPY                                                                                            L 21:04:02
JC07  USING    DEV=FFF TYPE=PRNTR    DEV=110 TYPE=READR                                                                      L 21:04:04
JC01  JOB RPGCHG   EXECUTING JOB STEP RPGII000 #001 21:04:04                                                                 L 21:04:07
AC10  LFD - PRNTR    , FORM NAME - STAND1  , COPIES - 0001, PAGES - 00000002, STEP =001                                      A 21:04:50
AC11  STEP #001  (RPGII000) USED 00022784 BYTES   ELAPSED WALL CLOCK TIME=00:00:41.676    TOTAL SVC CALLS=00003531           A 21:04:51
AC12      TERM CODE=000       SWITCH-PRIORITY=05   CPU TIME USED          =00:00:09.835    TRANSIENT CALLS=00000050           A 21:04:51
AC19              DEVICE EXCP'S      303=00001634  PRT=00000075  440=00000011                                                A 21:04:51
JC01  JOB RPGCHG   EXECUTING JOB STEP LNKEDT00 #002 21:04:52                                                                 L 21:04:54
AC10  LFD - PRNTR    , FORM NAME - STAND1  , COPIES - 0001, PAGES - 00000002, STEP =002                                      A 21:05:38
AC11  STEP #002  (LNKEDT00) USED 00014422 BYTES   ELAPSED WALL CLOCK TIME=00:00:42.532    TOTAL SVC CALLS=00003501           A 21:05:38
AC12      TERM CODE=000       SWITCH-PRIORITY=05   CPU TIME USED          =00:00:12.125    TRANSIENT CALLS=00000074           A 21:05:38
AC19              DEVICE EXCP'S      303=00001638  PRT=00000097                                                              A 21:05:38
JC01  JOB RPGCHG   EXECUTING JOB STEP RPGLOD00 #003 21:05:39                                                                 L 21:05:41
RPG030- DIVIDE BY ZERO EXCEPTION                                                                                             L 21:05:51
AC10  LFD - PRFILE   , FORM NAME - STAND1  , COPIES - 0001, PAGES - 00000000, STEP =003                                      A 21:05:54
AC11  STEP #003  (JOBDMP00) USED 00013552 BYTES   ELAPSED WALL CLOCK TIME=00:00:46.620    TOTAL SVC CALLS=00002986           A 21:06:28
AC12      TERM CODE=000       SWITCH-PRIORITY=05   CPU TIME USED          =00:00:14.795    TRANSIENT CALLS=00000038           A 21:06:28
AC19              DEVICE EXCP'S      303=00000985  110=00000003  PRT=00000821                                                A 21:06:28
AC21  JOB TOTALS    USED 00022784 BYTES     TOTAL ELAPSED WALL CLOCK TIME=00:02:29.377    TOTAL JOB SVC CALLS=00010018       A 21:06:28
AC22                                        WALL CLOCK TIME OF ALL STEPS =00:02:10.828    JOB TRANSIENT CALLS=00000162       A 21:06:29
AC23                                        TOTAL CPU TIME OF ALL STEPS  =00:00:36.755    TOTAL JOB EXCP'S   =00005264       A 21:06:29
JC03  JOB RPGCHG   TERMINATED ABNORMALLY. ERR 000    21:06:29                                                                L 21:06:31
AC10  LFD - PRNTR    , FORM NAME - STAND1  , COPIES - 0001, PAGES - 00000025, STEP =003                                      A 21:06:32
```

Figure 4-5. RPG Dump Analysis Listing (Part 1 of 9)



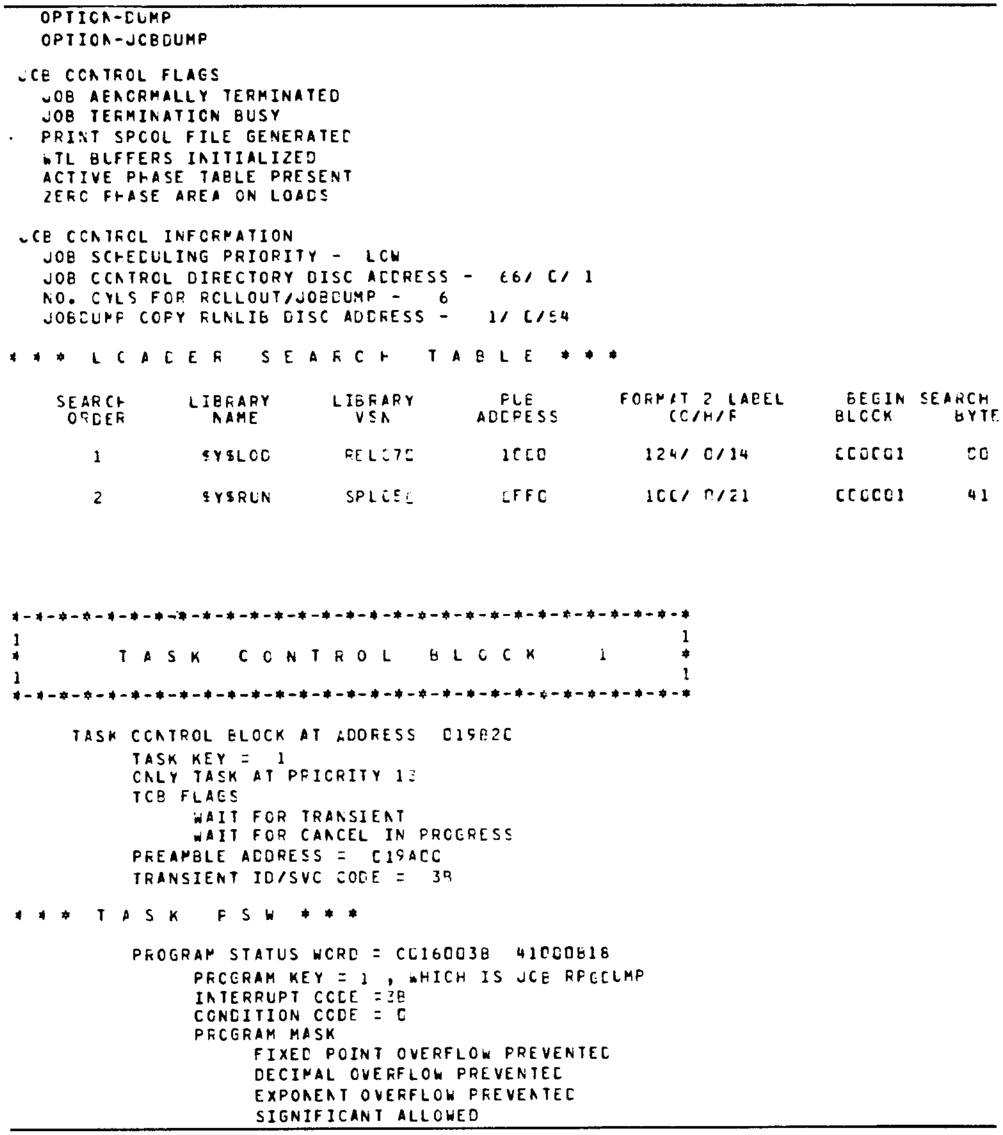
```
        LNIVAC OS/3 RPGII VERS 80C502              RPGDUMP                  8U/06/26 2C.48          PAGE       1

                CCC H           1                                                                RPGOBJ
OC1             C1C FCDFILE  IFEAF  45  45                 READER
OC2             C2C FPRFILE  O    F  48  48                PRINTEF
OC3             C3C ICDFILE  O11 C1   CINCI
OC4             C4C I                                                      1   5CACC1
OC5             C5C I                                                      8  12CACC2
CC6             C6C I                                                     15  44 NAMEIN
CC7             C7C C              ACD1        DIV  ACC2        QUOTE   6C
CC8             C8C C                          MCVE NAMEIN      NAMECT 3C
CC9             C9C CPPFILE  D 1        C1
O1C             1CC C                               CLCTE  B  1D
O11             11C C                               NAMECT E  48

                                                       SYMBCL TABLES

RESLLTING INDICATORS

ACCRESS RI        ACCRESS RI        ALCFESS RI        ACCRESS RI        ACCRESS RI        ACCRESS RI        ACCRESS RI

 CCC014 1P         CCC015 LR         CCC016 CC         CCC017 01         CCCC7A LC         CCC085 HC         CCCC86 H1
 CCC087 F2         CCC088 F3         CCCC89 H4         CCC08A H5         CCCC8F F6         CCCC8C F7         CCCC8C H8
 CCC08E F9         CCC08F L1         CCC09C U2         CCC091 L3         CCCC92 L4         CCCC93 L5         CCCC94 U6
 CCC095 L7         CCC096 L8

FIELC NAMES

ACCRESS FIELC            ACCRESS FIELC            ACCRESS FIELC            ACCRESS FIELC            ACCRESS FIELC

CCC180  *ERROR           CCC2CC  ACD1             CC020F  ACC2             CCC212  NAMEIN           CCC230  QUCTE
CCC234  NAMEOT

    OC2                                                                                        NCTE  2C1

NCTE  201  RESULTING INDICATOR IS INVALID CF LNDEFINEC. ENTRY OF LO IS ASSUMED.

                                                 PFCGRAM POINTEFS

         TABLE INPUT/OUTPLT                                                                 CC0254
         INPUT FIELC EXTRACTICN                                                             CCC258
         DETERMINE RECORD TYPE                                                              CCC3CL
         GET INPLT RECORD                                                                   CCC5CC
         DETAIL CALCULATICNS                                                                CCC67C
         TCTAL CALCULATIONS                                                                 CCC6A2
         CVERFLOW OUTPUT                                                                    CCC6B6
         OVERFLOW BYPASS                                                                    CCC6DA
         HEACER/CETAIL CUTPUT                                                               CCC7C4
         OLTPUT FIELDS                                                                      CCC76C
         INPUT/CLTPUT REQLEST BLOCKS                                                        CCC8C8
```

Figure 4-5. RPG Dump Analysis Listing (Part 2 of 9)



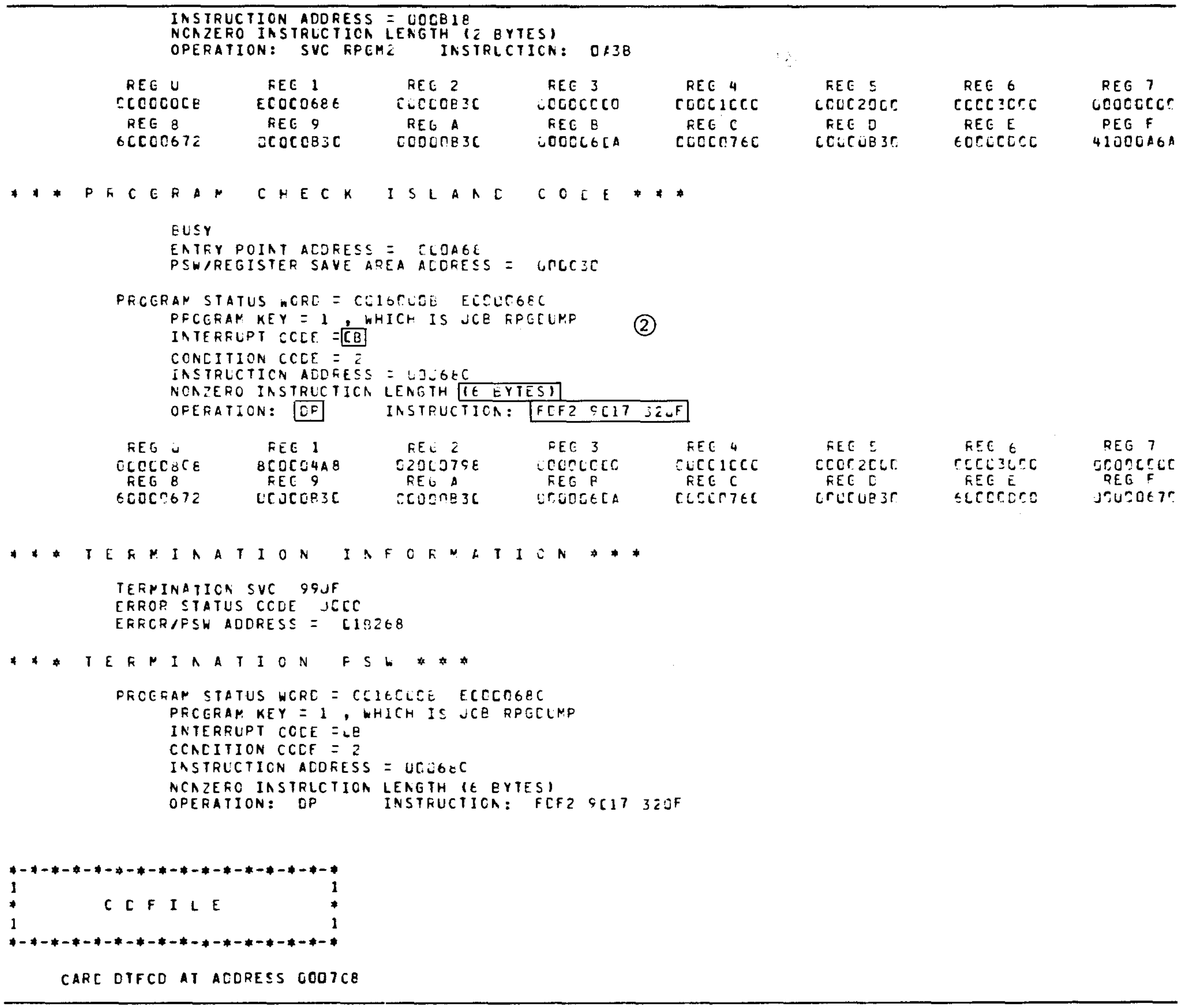
```
                                                                                                    VER8C06C4
   UNIVAC OS/3 JOBDUMP
   DATE: 80/06/26 TIME: 20:50:33

* * *  USER ERROR CODE 000C

   OS/3 VERSION  7.0-.01
   SUPERVISOR CHARACTERISTIC MASK - 37BFE8AE
   HARDWARE CONFIGURATION MASK - E104

 RRRRRRRR         PPPPPPPP          GGGGGGG         CCCCCCC        HHH       HHH       GGGGGGG
 RRRRRRRRRRR      PPPPPPPPPP       GGGGGGGGG       CCCCCCCCC       HHH       HHH      GGGGGGGGG
 RRR     RRRR     PPP     PPPP    GGGG    GGGG    CCCC    CCCC     HHH       HHH     GGGG    GGGG
 RRR      RRR     PPP      PPP    GGG             CCC              HHH       HHH     GGG
 RRR     RRRR     PPP     PPPP    GGG             CCC              HHHHHHHHHHHHH     GGG
 RRRRRRRRRRR      PPPPPPPPPP      GGG             CCC              HHHHHHHHHHHHH     GGG
 RRRRRRRRR        PPPPPPPP        GGG    GGGGG    CCC              HHH       HHH     GGG    GGGGG
 RRR  RRR         PPP             GGG    GGGGG    CCC              HHH       HHH     GGG    GGGGG
 RRR   RRR        PPP             GGGG      GG    CCCC    CCCC     HHH       HHH     GGGG      GG
 RRR    RRR       PPP              GGGGGGGGG       CCCCCCCCC       HHH       HHH      GGGGGGGGG
 RRR     RRR      PPP                GGGGGGG         CCCCCCC       HHH       HHH        GGGGGGG


*-*-*-*-*-*-*-*-*-*-*-*-*-*
1                         1
*        K E Y   1        *
1                         1
*-*-*-*-*-*-*-*-*-*-*-*-*-*

 JOB NAME IS RPGCHG   , JOB NUMBER -     1, STEP NUMBER -  3

 ALLOCATION MAP
      FROM       TO       LENGTH     CONTENTS
      ----       --       ------     --------

     19AC0     19B1F       288        PREAMBLE
     19B20     19C47       296        TCB
     19C48     19C93        76        JOB ACCOUNTING TABLE
     19C94     19D73       224        OPEN FILE TABLE
     19D84     19EC8       136        PHASE LOAD TABLE
     1A0E4     1A5C3      1104        SPOOLING BUFFERS
     1A5C4     1A5E7       228        LOG SPOOL CONTROL TABLE
     1A6D8     1A7BB       228        READER SPOOL CONTROL TABLE
     1A7EC     1A89F       228        PRINT SPOOL CONTROL TABLE

     1AAC0     1BAA7      4264        LOAD MODULE AREA
     1BAA8     1FFFF     17752        UNUSED MEMORY

 LAST PHASE LOADED - RPGLOD00, PHASE DATE - 80/06/26

 REGION DATE - 80/06/26  80/178

* * *  J O B   C O N T R O L   A R E A  * * *

 JOB STEP OPTIONS
```

Figure 4-5. RPG Dump Analysis Listing (Part 3 of 9)



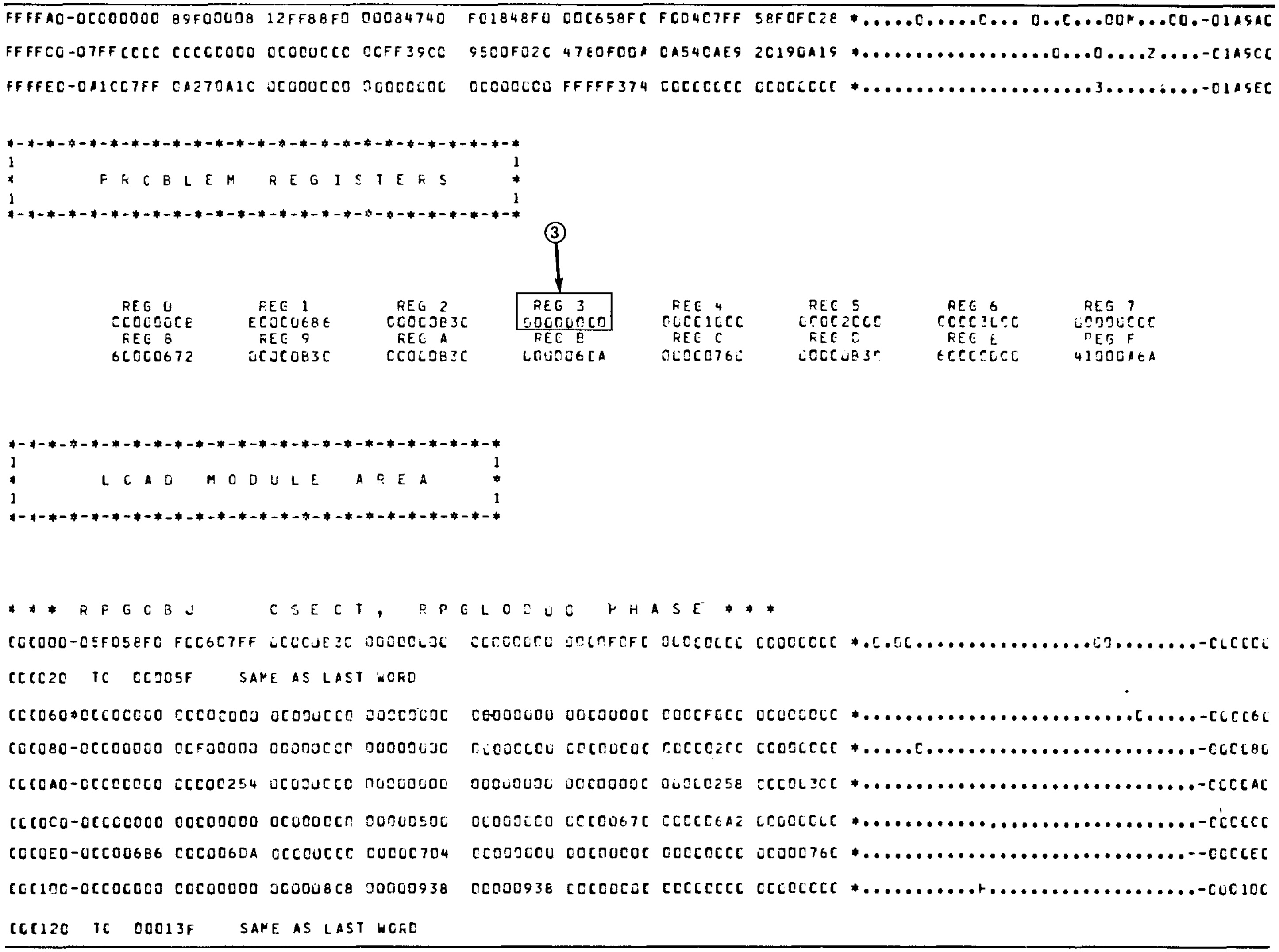
```
   OPTICN-CUMP
   OPTION-JCBDUMP

 JCB CCNTROL FLAGS
   JOB AENCRMALLY TERMINATED
   JOB TERMINATICN BUSY
.  PRINT SPCOL FILE GENERATEC
   WTL BLFFERS INITIALIZED
   ACTIVE PHASE TABLE PRESENT
   ZERC FHASE AREA ON LOADS

 JCB CCNTRCL INFORMATION
   JOB SCHEDULING PRIORITY -  LCW
   JOB CCNTROL DIRECTORY DISC ACDRESS -  66/ C/ 1
   NO. CYLS FOR RCLLOUT/JOBCUMP -   6
   JOBCUMP COPY RLNLIB DISC ADDRESS -   1/ C/54

* * *  L C A C E R   S E A R C H   T A B L E  * * *

    SEARCH      LIBRARY      LIBRARY       PLE        FORMAT 2 LABEL     BEGIN SEARCH
    ORDER        NAME          VSN        ADCPESS         CC/H/F         BLOCK     BYTE

      1         $YSLOD       RELC7C        1CCO         124/ 0/14        CCOCO1     CO

      2         $YSRUN       SPLC5C        CFFC         1CC/ 0/21        CCCCO1     41



*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
1                                                       1
*        T A S K   C O N T R O L   B L O C K    1       *
1                                                       1
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

     TASK CCNTROL BLOCK AT ADDRESS  C19B2C
          TASK KEY =  1
          CNLY TASK AT PRICRITY 13
          TCB FLAGS
               WAIT FOR TRANSIENT
               WAIT FOR CANCEL IN PROGRESS
          PREAMBLE ADDRESS =  C19ACC
          TRANSIENT ID/SVC CODE =  3B

* * *  T A S K   P S W  * * *

          PROGRAM STATUS WCRD = CC16003B  41000B18
               PRCGRAM KEY = 1 , WHICH IS JCB RPGCUMP
               INTERRUPT CCDE =3B
               CONDITION CCDE = C
               PRCGRAM MASK
                    FIXED POINT OVERFLOW PREVENTEC
                    DECIMAL OVERFLOW PREVENTEC
                    EXPONENT OVERFLOW PREVENTEC
                    SIGNIFICANT ALLOWED
```

Figure 4-5. RPG Dump Analysis Listing (Part 4 of 9)



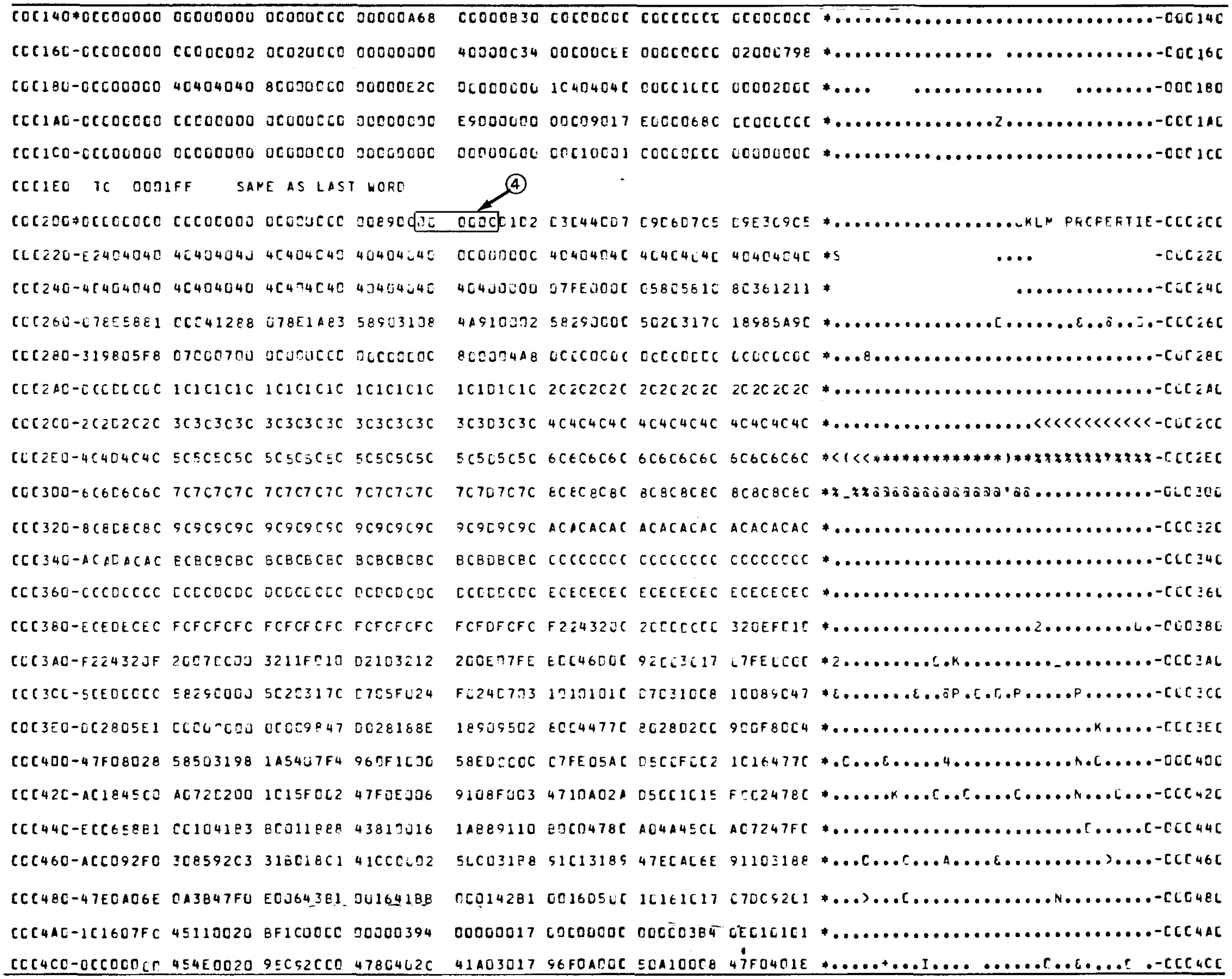
```
              INSTRUCTION ADDRESS = 000B18
              NONZERO INSTRUCTION LENGTH (2 BYTES)
              OPERATION:  SVC RPGM2    INSTRUCTION:  0A3B

          REG 0       REG 1       REG 2       REG 3       REG 4       REG 5       REG 6       REG 7
        C00000CB    E0C00686    C0C00B3C    00000000    C000100C    C000200C    C000300C    0000000C
          REG 8       REG 9       REG A       REG B       REG C       REG D       REG E       REG F
        60C00672    3C0C0B3C    C0000B3C    60000C6EA   C000076C    C000B3C    60C0C0CC    41000A6A

*** P R O G R A M   C H E C K   I S L A N D   C O D E ***

              BUSY
              ENTRY POINT ADDRESS =  C00A6E
              PSW/REGISTER SAVE AREA ADDRESS =  000C3C

          PROGRAM STATUS WORD = CC16C0CB  EC0C068C
              PROGRAM KEY = 1 , WHICH IS JOB RPGDUMP          ②
              INTERRUPT CODE =CB
              CONDITION CODE = 2
              INSTRUCTION ADDRESS = 00068C
              NONZERO INSTRUCTION LENGTH (6 BYTES)
              OPERATION:  DP       INSTRUCTION:  FDF2 9C17 320F

          REG 0       REG 1       REG 2       REG 3       REG 4       REG 5       REG 6       REG 7
        0C0C08C8    8C0C04A8    020C0798    C000C0C0    C0CC1CCC    C00C2C0C    C0CC30CC    50009C0C
          REG 8       REG 9       REG A       REG B       REG C       REG D       REG E       REG F
        60C09672    0C0C0B3C    CC000B3C    0C000C6EA   C0C0076C    C0C0B30    6CC0C0C0    30C0067C

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

          TERMINATION SVC  990F
          ERROR STATUS CODE  0C0C
          ERROR/PSW ADDRESS =  C18268

*** T E R M I N A T I O N   P S W ***

          PROGRAM STATUS WORD = CC16C0CB  EC0C068C
              PROGRAM KEY = 1 , WHICH IS JOB RPGDUMP
              INTERRUPT CODE =CB
              CONDITION CODE = 2
              INSTRUCTION ADDRESS = 00068C
              NONZERO INSTRUCTION LENGTH (6 BYTES)
              OPERATION:  DP       INSTRUCTION:  FDF2 9C17 320F


*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
1                             1
*        C D F I L E          *
1                             1
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

    CARD DTFCD AT ADDRESS 0007C8
```

Figure 4-5. RPG Dump Analysis Listing (Part 5 of 9)



```
FFFFA0-0CC00000 89F00008 12FF88F0 00084740  FC1848F0 00C658FC FC04C7FF 58F0FC28 *.....0.....C... 0..C...00M...C0.-01A9AC
FFFFC0-07FFCCCC CCCCC000 0C000CCC 0CFF39CC  9500F02C 4780F00A 0A540AE9 2C190A19 *.................0...0....Z....-C1A9CC
FFFFEC-0A1C07FF CA270A1C 0C000CC0 30000000C  0C0000C00 FFFFF374 CCCCCCCC CC00CCCC *....................3.........-01A9EC

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
1                                               1
*       P R C B L E M   R E G I S T E R S       *
1                                               1
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

                                                 ③
      REG 0       REG 1       REG 2       REG 3       REG 4       REG 5       REG 6       REG 7
    CC0000CE    EC0C0686    C00C0B3C    00000000C0    0CCC1CCC    CC0C2CCC    C0CC3CCC    C000CCCC
      REG 8       REG 9       REG A       REG B       REG C       REG D       REG E       REG F
    6C000672    0C0C0B3C    CC0C0B3C    C00006CA    0C0C076C    C0CC0B3C    6CCCCCCC    41000A6A

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
1                                             1
*       L C A D   M O D U L E   A R E A       *
1                                             1
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

* * *  R P G C B J      C S E C T ,   R P G L O D U C   P H A S E  * * *

CCC000-05F058F0 FCC6C7FF CCCC0B3C 00000C3C  CCCCCCC0 00C0F0FC 0CCCCCCC C000CCCC *.C.5C..................C0.........-CCCCCC

CCCC20  TC  CC005F     SAME AS LAST WORD

CCC060*0CC0CCC0 CCC0C000 0C00CCC0 00C0C0C  C0000000 00C0000C C00CFCCC 0C0CC0CC *............................C......-CCCC6C

CCC080-0CC00000 0CF00000 000000CCP 0000003C  CC00CCC0 CCCC0C0C CCCCC2CC CC0CCCCC *.....C...........................-CCCC80

CCC0A0-0CCCCCC0 CCC00254 0C000CC0 00000000  00000000 00C0000C 00CC0258 CCC0C3CC *................................-CCCCAC

CCC0C0-0CCC0000 00C00000 0C0000C0 0000050C  0C000CC0 CCC0067C CCCCC6A2 CC00CCCC *................................-CCCCCC

C0C0E0-0CC006B6 CCC006DA 0CCCCCCC C000C704  CC000C00 00C0000C C00C0CCC 0C00076C *...............................--CCCCEC

CCC100-0CC0C000 C0C00000 0C0008C8 00000938  00000938 CCC00C0C CCCCCCCC CCC0CCCC *............+....................-C0C10C

CCC120  TC  00013F     SAME AS LAST WORD
```

Figure 4-5. RPG Dump Analysis Listing (Part 6 of 9)



```
COC140*0CC00000 0C000000 0C000CCC 00000A68  CC000B30 COCCOCOC COCCCCCC CCCOCOCC *................................-00C14C
CCC16C-CCCOC000 CCOOC002 0C020CC0 00000000  40000C34 00C00CEE 000CCCCC 0200C798 *................ ...............-COC16C
CCC180-0CC00000 40404040 8C000CC0 00000E2C  0C00000C 1C40404C 00CC1CCC 0000200C *....     ..............    .........-00C180
CCC1A0-CCCOCCCC CCC00000 0C000CCC 0C0000C0  E9000000 00C09017 E0CC068C CCOCCCCC *................Z...............-CCC1AC
CCC1C0-CCC00000 0C000000 0C000CC0 00C0000C  00000C0C 0CC1000I COCCCCCC C000000C *................................-CCC1CC
CCC1E0   TC   0001FF     SAME AS LAST WORD
CCC200*0CCCCCC0 CCC0C000 0C00UCCC 00890C0C  000C01D2 C3C44CD7 C9C6D7C5 C9E3C9C5 *...................KLM PRCPERTIE-CCC2CC
CCC220-E2404040 4C40404J 4C404C40 40404C40  0C00000C 4040404C 404C4C4C 4040404C *S                  ....         -CCC22C
CCC240-4C404040 4C404040 4C404C40 40404C40  4040C000 07FE000C C580561C 8C361211 *                  ..............-CCC24C
CCC260-C78E5881 CCC41288 078E1A83 58903108  4A910002 5829000C 502C317C 18985A9C *.................C.......&..@..:.-CCC26C
CCC280-319805F8 07C00700 0C0C0CCC 0CCC0C0C  8C0004A8 0CCC0C0C 0CCC0CCC CCCCCCOC *...8............................-CCC28C
CCC2A0-CCCCDCCC 1C1C1C1C 1C1C1C1C 1C1C1C1C  1C1D1C1C 2C2C2C2C 2C2C2C2C 2C2C2C2C *................................-CCC2AC
CCC2C0-2C2D2C2C 3C3C3C3C 3C3C3C3C 3C3C3C3C  3C3D3C3C 4C4C4C4C 4C4C4C4C 4C4C4C4C *....................<<<<<<<<<<<<-CCC2CC
CCC2E0-4C4D4C4C 5C5C5C5C 5C5C5C5C 5C5C5C5C  5C5D5C5C 6C6C6C6C 6C6C6C6C 6C6C6C6C *<(<<*************)**%%%%%%?%%%%-CCC2EC
CCC300-6C6D6C6C 7C7C7C7C 7C7C7C7C 7C7C7C7C  7C7D7C7C 8C8C8C8C 8C8C8C8C 8C8C8C8C *%_%%@@@@@@@@@@@@'@@............-CCC30C
CCC320-8C8D8C8C 9C9C9C9C 9C9C9C9C 9C9C9C9C  9C9D9C9C ACACACAC ACACACAC ACACACAC *................................-CCC32C
CCC340-ACADACAC BCBCBCBC BCBCBCBC BCBCBCBC  BCBDBCBC CCCCCCCC CCCCCCCC CCCCCCCC *................................-CCC34C
CCC360-CCCDCCCC CCCCDCDC DCDCDCCC DCDCDCDC  DCDCDCDC ECECECEC ECECECEC ECECECEC *................................-CCC36C
CCC380-ECEDECEC FCFCFCFC FCFCFCFC FCFCFCFC  FCFDFCFC F224320C 2CCCCCCC 320EFC1C *.....................2.........L.-CC038C
CCC3A0-F22432JF 2007CC00 3211F010 D2103212  200E07FE ECC46D0C 92CC3C17 C7FECCCC *2..........C.K.........._.........-CCC3AC
CCC3C0-5CEDCCCC 5829C00J 5C2C317C C7C5F024  FC24C703 1010101C C7C310C8 10089C47 *&.........&..@P.C.C.P.....P.......-CCC3CC
COC3E0-0C2805E1 CCC0C000 0C0C9847 D028168E  18909502 8CC4477C 8C28D2CC 9C0F8DC4 *.........................K.....-CCC3EC
CCC400-47F08028 58503198 1A5407F4 960F1C00  58EDCC0C C7FE05AC D5CCFCC2 1C16477C *.C...&.....4...............N.C.....-CCC40C
CCC42C-AC1845C0 A072C200 1C15F0C2 47F0E006  9108F003 4710A02A D5CC1C15 FCC2478C *......K...C..C....C.....N...C...-CCC42C
CCC44C-ECC65881 CC1041B3 BC011888 43810016  1A889110 80C0478C A04A45CC AC7247FC *.........................C.....C-CCC44C
CCC460-ACC092F0 308592C3 31B018C1 41CCC002  5CC031B8 91C13189 47ECAC6E 91103188 *...C...C...A....&..........>....-CCC46C
CCC48C-47E0A06E 0A3847F0 E0064381 00164188  0C014281 0016050C 1C161C17 C7DC92C1 *...>...C.................N.........-CCC48C
CCC4AC-1C1607FC 45110020 BF1C00CC 00000394  00000017 C0C0000C 00CC03B4 CEC1C1C1 *................................-CCC4AC
CCC4C0-0CC000CC 454E0020 95C92CC0 478C4C2C  41A03017 96F0AC0C 50A100C8 47F0401E *.....+...I.... ......C..&....C .-CCC4CC
```

Figure 4-5. RPG Dump Analysis Listing (Part 7 of 9)

```
COC4E0-BCC6C000 C394D203 1C044C1A 45FE0032  B004016C 41A30065 5DA100C8 47F0E006 *......K... ........_....&....C..-CCC4EC
CCC500-058047F0 80148010 00U00C00 000003B8  CC0000C0 GC2E181C 58DC315C 5C1DCCC8 *...C.........................&&...-000500
CCC52C-5CED0004 C7C8307B 3C7847F0 80BA58CC  31084AC1 00C2582C CCCC5C2C 317C589C *&...P..#.#.C......CA......&..@..-CCC52C
CCC540-8C861A93 58A10008 96FCA0C0 41F03085  19AF4770 806E50CC 31B892C1 31B091C1 *.........C...C.......>&....A....-CCC54C
CCC56C-318947E0 8C6E911U 318847EC 806E0A3B  58AC30B8 4AA08012 5C1ACCCC 58F1CCCC *.....>.......>......C...&....1..-CCC56C
CCC58C-12FF478C 8C921AF3 58C030C0 58AC0184  1AA305EF 58EDJC04 58CCCCC8 C7FEBC1E *........3........................-COC58C
CUC5A0-0CC004A4 CC000000 UC0000C0 FUFUFUFU  FUFUFUF0 FCFCFCFC 0UCCCCCC 58108C9E *............CCCCuCUCU0GG........-CCC5AC
CUC5CC-1A1358C0 31C84A01 JC261890 58FU8U0A  1AF305EF 92C0900E 95AA1C1C 477060EA *......C......0...3..............-CCC5CC
COC5E0-58F080CE 1AF305EF 47F061C2 58FC311C  C5EF911G 9C1647EC 81C258FC 9CU412FF *.C...3...C...C.............C....-CCC5EC
CCC60C-47708106 95F03085 58A03088 4770812C  58A030B8 4AA08012 16EE5CEA UCUC47FC *.....C..............C.....&....C-0CC60C
CCC62C-8C929522 9C134770 81564111 0U0492FU  100158FU 80A212FF 478C8146 5CF08C9E-*..............C...C........&C..-CCC62C
CUC640-D7C360A2 8CA247FU 8C3A96FC 30150208  3C788CAA 47F08092 5C8CCC14 58F030BC *P......C...C..K..A...0..&....C..-CCC64C
CCC660-41FFCC00 C5EF588D JC1447F0 8U2C0L0C  C580589U 315090DE 9164C732 9CCC9CCC *...........C.........&....P.....-CCC66C
CCC680-F8F29017 320CFDF2 9C1732UF F83C3230  9017940F 3230021C 32343212 98CE9164 *82.....3....8.........K.........-0CC68C
CCC6AC-C7FECS8C 58903150 9CCE9164 98CE9164  41900UFF C7FE58AC 315C9CCE ACUC582C *.......C..................C......-CCC6AC
CCC6CC-31C858D0 310C58C0 3CFC588C 30E4D607  30C3000C U7E8968C 318E947F 318B058C *.............L0............."....-CCC6CC
CCC6EC-95F02043 4770801A 92272047 92022046  41020036 C5ED920C 2C4892CC 2C4C98DF *.C..........................<..-0CC6EC
CCC70C-ACC0C7FE 58AC3150 9CCEACC0 58203108  580031CC 58C030FC 058C95FC 3C17477C *.......C.....................C....-0CC70C
CCC720-8C2E9251 2C489210 2C4A5892 C038924C  9000C22E 90C1900C 589C0000 05E992C2 *.........C..... ..K..........Z..-CCC72C
CCC740-2C4641C2 0C3805ED J58047F0 8CCE96FC  8CC198DE ACC007FE 98CEACCC C7FECCCC *...........C..C.................-C0074C
CCC760-CCCCC764 5882C038 F353BCC4 3230910C  8C094710 901696FC 8UC9FE33 3230323C *.........3.................0.........-CCC76C
                                          ⑨ (ADD2)
CCC78C-C210E012 3234924U 3234021C 32353234  07FE0000 U0C0000C 4C4CF8F9 FC4C4C4C *K...... ..K............  8°C    -0CC78C
CCC7AC-4C4040F0 4C40C1D2 D3D44CC7 C9D6D7C5  C9E3C9C5 E240404C 4C4C4C4C 4040404C *   ?  UKLM PROPERTIES             -CCC7AC
CCC7CC-4C404040 4C400000 04008068 000CCC0C  0C0000C0 CCC008C4 C1CCC7EE CCOCCCCC *       ...........................-CCC7CC
CCC7EC-0CCC0400 C3C4C6C9 D3C54040 80001650  0000105A U0C008FC C11CCLCC 19FFFFDC *....CDFILE   ...&...I...C.........-0CC7EC
CCC800-0CCC1062 C2C0C798 CC000U2E 0U2C410C  82080204 002C0C0C 0CCCC798 C0U0CCCC *...............................-CCC80C
COC820-0CC00000 CCCC0C00 0C000CC0 002C0430  CC104U40 4040F0FC F0FCF2F6 4040404C *..................    000C26    -CCC82C
CCC840-4C404040 C1C2C340 E3D9E4C3 D2C9D5C7  4CC3D64B 4C4C4C4C 4C4C4C4C 4C4C4C4C *    ABC TRUCKING CC.             -0CC84C
CCC860-4C40C0C0 CCC00000 0C008C68 00000C0C  0G000000 00C008A4 C2CC088E CC00C0CC *  ...............................-CCC86C
```

Figure 4-5. RPG Dump Analysis Listing (Part 8 of 9)

```
COC880-0CC00400 C7C9C6C9 03C54040 8000151J  00000F8C 00C104FC F62CCCCC 12FFFFDC *....PRFILE  ............06......-00088C
                                      (8) IORB CURRENT RECORD ADDRESS
COC8A0-CCCOOCF96 C1000832 0C0CJC30 002C440C  849A2004 C030OCU9 0000C832 0000C0CC *................................-CCC8AC
CCC8C0-0CC00000 423C0000 02000798 00C0000C  0C200001 00C100FF FFFFFFCC 0CU0CC11 *................................-00C8CC
                                                                                              IORB FILENAME
CCC8EC-CCCCGOC0 CCCCC000 0C00U7C8 C000CFAC  0C000000 J0C0000C C3C4C6C9 D3C5404C *..............F..............CCFILE  -CCC8EC
CCC900-0CC0C832 CC000000 0C30UCC1 J00202FF  FFFFFF00 J0C00011 1C2CCCCC CCUCCCCC *................................-000900
CCC920-CCCJ0868 CC0CCE48 CCCCJCCC 30CCJCJC  C7C9C6C9 C3C5404C C5FC9C42 FCEA187F *................PPFILE  .C..C.."-0CC92C
CCC940-1E409101 40194710 7C8E9180 401647EJ  70269182 4CCE476C 7C2E58FC 3CB4C5EF *. .. ....... ....... ......C....-CCC940
CCC96C-95204018 477C706A 95CC4CCE 477C7C6A  58904000 95C04C0E 477C7C5C 1B55435C *.. ....... ....... ... ....*...&-0CC96C
CCC98C-4CC90650 42507053 92409CCC 02C09U01  900047F0 706A1814 58FC3114 05EF057C * ..&.&... ..K.......C.....C......-CCC98C
CGC9A0-58707060 58C03150 41DDJ11C 41CC7UEA  5CC0UC04 58104C2C 587C4C24 C5E7C57C *........&.........&...... ... ..X..-0UC9AC
CCC9C0-58707090 47F07U9J 1BFF12FF 47807JCA  92FJ3085 91104016 471C70B2 5CF07CE6 *.....C............C.... .....&C.W-CUC9CC
CCC9E0-C2C131B1 7CE6C2JJ 31B37CE9 98427JEA  91013189 47EJF0CC CA3B58EC 318CC7FE *K....WK....Z...........0.........-CCC9EC
COCA00-47FC7CEC 918C4016 47EC7CEC 95CC4C0E  47707CEC 58F030B4 C5EF9842 7CEAC7FE *.0.... ....... ......0...........-CCCACC
CCCA2C-CCC0C9CC CCCG100J JC0C2CCC COCC3COC  CC00JUCC 5CC0U502 CCCCC8C8 CCCCC83C *.......................&.......F....-CCCA2C
CCCA40-0CCC0832 CCC0C760 CCJCJE3C 5UCC05F2  5CC0093A CCC0UEC8 CCCCC4A4 CCCCC8C8 *........-....&..2&.......H........F-CCCA4C
CCCA60-CCCCCCC1 CCC00000 U5FC5820 315092F0  30855610 21C4501C 31884B1C FCBE9505 *...........C....5.5........&.....C...-CCCA6C
CCCA80-21C34780 FC32C201 31B61CC2 95072103  478CFU52 95CB2133 478CF088 411021CC *....C.K..............C.........C......-CCCA8C
CCCAA0-5C103188 92F0318U 91033189 477GFCAC  91C8318A 4710FC9C 47FCFCAC 91CC1CCC *&....C..........C........0..CC......-CCCAAC
CCCACC-47E0F032 92E731BJ 91033189 477UFU6E  9108318A 4710FC9C 5U1C21C4 98CF21C8 *..C..Y.........C>.......C.&.........-CCCACC
CCCAEC-0A3BC5EC 5810E098 47F0ECCC 8C000000  0A0F92E9 31E39103 3189477C FCAC91C8 *...........C...........Z.........C...-CCCAEC
COCB00-318A47E0 FCAC581U 31A45C10 21C49604  318A47FU FCAEJA3E 561CFLB6 47FCF0BA *....C......&..........CG......C..CG.-CCCBCC
CCCB2C-8CCCG0C0 CACF4CB8 0C06JCCC 0000JC0C  CC0U0CCU CCCC0CUC CCCCCCCC CCUCCCCC *......<.........................-CUCB2C
CCCB40-CCCCCCC0 CCCCC000 0CU0UCCC C0000000  0C0J0000 00E90C0C CCCCCCCC CCCCCCCC *................................-CCCB4C
CCCB6C-CCCCCCC0 CCCCC000 0A11581C 3150410C  0C01412U 00C841EC 308E41F0 00F042CC *................&...............C.C..-CCCB6C
CCCB80-CC539180 10004780 0C5E42F2 E0C0890C  CC014620 C04E47FC C07C4CC6 C6D94CC1 *..........;.2..........+.0.ā FCR A-000B8C
CCCBA0-D3C340C6 E4E3C7E4 E340D3C9 18300730  41101000 5610DC8C 47FCDC9C 4CCCCCCC *LL OUTPLT LI..............C.. ...-CCCBAC
CCCBC0-0A11120C 4780C09A 0A1CD5C4 10CCD19A  4770D0AE 96C13189 C2C1D1EC C2B8D5C5 *...........N...J.............K.J.K.N.-CCCBCC
CCCBE0-1CC0D19F 477CC0BC 94FE3189 D5C71CCC  C1A5477U C0C09603 3189C2C1 C1BCD2B8 *..J..........N...J...........K.J.K.-CCCBEC
CCCC00-47F0C172 5080318C 41B0D1BE 50803174  1B0J58F0 3CA405EF 91C831EC 4710D2FC *.CJ.&.......J.&.......C......F....K.-CCCCOC
```

Figure 4-5. RPG Dump Analysis Listing (Part 9 of 9)

Appendix A
Program Exceptions

Table A-1. Program Exceptions

Interrupt Code	**Interrupt Cause**
01	Operation exception: An illegal operation has been attempted or an operation using a noninstalled processor feature has been attempted.
02	Privileged operation exception: A privileged operation has been attempted by a program operating in the problem mode (PS, bit 14 of current PSW, set to 1).
03	Execution exception: The subject instruction of an execute instruction is an execute instruction.
04	Protection exception: A storage protection violation occurs on a program-generated address when the storage protect feature is installed.
05	Address exception: A main storage location outside the range of the installed main storage is referenced by a program-specified address. For the load-control-storage (LCS) instruction only, the referenced control storage location is nonexistent.
06	Specification exception: • The unit of information referenced is not on an appropriate boundary. • An invalid modifier field is specified in the service timer register (STR) instruction. • The r_1 field of an instruction that uses an even/odd pair of registers (64-bit operand) does not specify an even register. • A floating-point register other than 0, 2, 4, or 6 is specified. • A multiplier or divisor in decimal arithmetic exceeds 15 digits and sign. • The first operand field is shorter than, or equal in length to, the second operand in decimal, multiply, and divide instructions.

continued

Table A-1. Program Exceptions (cont.)

Interrupt Code	Interrupt Cause
06 (cont.)	• The four low-order address bits specified by the contents of r_2 comprise a set storage key (SSK) or insert storage key (ISK) instruction and are not equal to 0. • The function specified by the I_2 field of a diagnose instruction was not loaded in the transient area of control storage. • A SOFTSCOPE instruction (SSFS or SSRS) was issued without the supporting microcode loaded in the control storage transient area.
07	Data exception: • An invalid sign or digit code is detected in decimal operands. • Fields in decimal arithmetic overlap incorrectly. • The first operand of the multiply decimal instruction does not have sufficient number of high-order 0 digits.
08	Fixed-point overflow exception: A fixed-point add or subtract operation exceeds the capacity of the first operand field. This interrupt is masked by b, bit 36 of the current PSW.
09	Fixed-point divide exception: The quotient of a fixed-point divide operation exceeds the capacity of the first operand (including division by 0), or the result of a convert-to-binary instruction exceeds 31 bits.
0A	Decimal overflow exception: The result of an add decimal, subtract decimal, or zero-and-add instruction exceeds the capacity of the first operand location. This interrupt is masked by d, bit 37 of the current PSW.
0B	Decimal divide exception: The quotient of a divide decimal (DP) instruction exceeds the capacity of the quotient part of the first operand field.
0C	Exponent overflow exception: The final characteristic resulting from a floating-point arithmetic operation exceeds 127.
0D	Exponent underflow exception: The final characteristic resulting from a floating-point arithmetic operation is less than 0. This interrupt is masked by e, bit 38 of the current PSW.
0E	Significance exception: The final fraction resulting from a floating-point addition or subtraction is equal to 0. This interrupt is masked by s, bit 39 of the current PSW.
0F	Floating-point divide exception: The divisor fraction in a floating-point divide operation is equal to 0.

Appendix B
SYSDUMP File Allocation

This appendix contains information about $Y$DUMP file allocation.

Table B-1 shows the number of cylinders required, depending on your system's main storage capacity and the type of disk device you are using.

For example, a system with 8MB main storage requires 53 cylinders for the SYSDUMP file on an 8433 disk drive.

Table B-1. $Y$DUMP File Size in Cylinders

Storage Capacity (MB)	Disk Type							
	8416	8417	8418	8419	8430	8433	8470	8494
	Cylinders Required for Use							
1	15	5	15	12	8	7	2	3
1.5	22	8	22	18	11	10	2	4
2	30	10	30	24	15	14	3	5
2.5	37	13	37	30	18	17	4	6
3	44	15	44	36	22	20	4	7
3.5	52	18	52	41	26	23	5	8
4	59	20	59	47	29	27	6	9
5	74	25	74	59	36	33	7	11
6	88	30	88	71	44	40	8	13
7	103	35	103	82	51	46	10	15
8	118	40	118	94	58	53	11	17

Use the following procedure to expand the size of the DUMP file after an increase in storage:

1. Use the following JCL to scratch the $Y$DUMP file on SYSRES:

```
// JOB SCRDUMP
// DVC 20 // LFD PRNTR
// DVC RES // LBL $Y$DUMP // LFD SYSDUMP
// SCR SYSDUMP
/&
// FIN
```

2. Use Table B-1 to determine the number of cylinders required for your system.

3. Use the following JCL to allocate a MIRAM file (labeled $Y$DUMP) and to execute the SG$OPN load module:

```
// JOB ALLOCATE
// DVC 20 // LFD PRNTR
// DVC RES
// EXT MI,C,1,CYL,xx            (where xx is the number of cylinders)
// LBL $Y$DUMP // LFD SYSDUMP
// EXEC SG$OPN
// PARAM SYSDUMP,100
/&
// FIN
```

You now have sufficient contiguous free space to store a dump of your system's main storage.

UNISYS

USER COMMENTS

We will use your comments to improve subsequent editions.

NOTE: Please do not use this form as an order blank.

____

(Document Title)

________________ ________________ ________________

(Document No.) *(Revision No.)* *(Update Level)*

Comments:

From:

____

(Name of User)

____

(Business Address)

Fold on dotted lines, and mail. (No postage is necessary if mailed in the U.S.A.)
Thank you for your cooperation

CUT

FOLD

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

Unisys Corporation
E/MSG Product Information Development
PO Box 500 — E5-114
Blue Bell, PA 19422-9990

FOLD