PERKIN-ELMER

PERKIN-ELMER SERIES 3200 WRITABLE CONTROL STORE (WCS) SUPPORT PROGRAMS

Reference Manual

48-096 F00 R00

The information in this document is subject to change without notice and should not be construed as a commitment by The Perkin-Elmer Corporation. The Perkin-Elmer Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license, and it can be used or copied only in a manner permitted by that license. Any copy of the described software must include the Perkin-Elmer copyright notice. Title to and ownership of the described software and any copies thereof shall remain in The Perkin-Elmer Corporation.

The Perkin-Elmer Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by Perkin-Elmer.

The Perkin-Elmer Corporation, Data Systems Group, 2 Crescent Place, Oceanport, New Jersey 07757

Printed in the United States of America

TABLE OF CONTENTS

CHAPTERS (Continued)

CHAPTERS (Continued)

APPENDIXES

FIGURES

TABLES

PREFACE

This manual is a guide for the development and operation of writable control store (WCS) microcode routines using any of the available Perkin-Elmer WCS support program products. These products include the newer WCS support product consisting of the three separate programs, WCSLINK, WCSAIDS, and the loader and power fail monitor (MPSLPFM), and the earlier WCS support product consisting of a full support program (WCSUPP) and a loader and power fail monitor (WCSLPFM). In describing each of these products, the manual takes into account the several versions of WCSAIDS and WCSUPP applicable to different Perkin-Elmer Series 3200 processors. The WCS user should be familiar with the operation of the applicable processor hardware to understand the microprogramming concepts involved with the use of WCS. The user should also be familiar with the OS/32 operating system.

Chapter 1 contains an introduction to and a functional description of the WCS. Chapter 2 describes the procedures for creating a microcode program and the use of assembly level WCS instructions and their effects on the operating system. Chapter 3 describes each of the WCS support programs and outlines the manner in which they are linked, loaded, and started. Chapter 4 presents the WCSLINK commands. Chapter 5 presents the WCSAIDS commands. MPSLPFM commands are presented in Chapter 6. Chapter 7 describes the commands available to WCSUPP and WCSLPFM. Chapter 8 provides brief descriptions of two typical WCS applications. The appendixes contain command summaries, error messages, and program examples.

This manual is intended for use with the OS/32 R06.2 software release or higher. Additional material specifically related to the Model 3200MPS System has also been included. These Model 3200MPS System features are supported by the OS/32 R07.1 software release and higher. Throughout the text these features are identified as applicable only to the Model 3200MPS System.

For information on the contents of all Perkin-Elmer 32-bit manuals, see the 32-Bit Systems User Documentation Summary.

CHAPTER 1
PERKIN-ELMER SERIES 3200 WRITABLE CONTROL STORE (WCS)

1.1 INTRODUCTION

The Perkin-Elmer Series 3200 Writable Control Store (WCS) is a hardware option for the Models 3220, 3230, 3240 and 3250 microprogrammable processors. These models are uniprocessor systems consisting of a single central processing unit (CPU) using a single copy of the operating system. It is also part of the standard configuration of the Model 3200MPS System. The Model 3200MPS System is a tightly coupled multiprocessing system consisting of a CPU and one or more auxiliary processing units (APUs) executing a single copy of the operating system. Each processing unit is capable of executing instructions simultaneously with the others, and of addressing a common pool of memory. Tasks may pass from one processing unit to another. For general block diagrams of these processors, see Figures 1-1 through 1-5. The WCS extends the flexibility of the user level processor to that of the microprocessor. The WCS provides:

- the ability to use the instructions available for writing to WCS, reading from WCS, and executing user created microcode routines located in WCS,
- 2,048 words of high speed control store memory for the Models 3220, 3240, and 3250 processors,
- 2,048 words of high speed control store memory for the CPU of the Model 3200MPS System, and
- 4,096 words of high speed control store memory for the Model 3230 processor and for the APU(s) of the Model 3200MPS System.

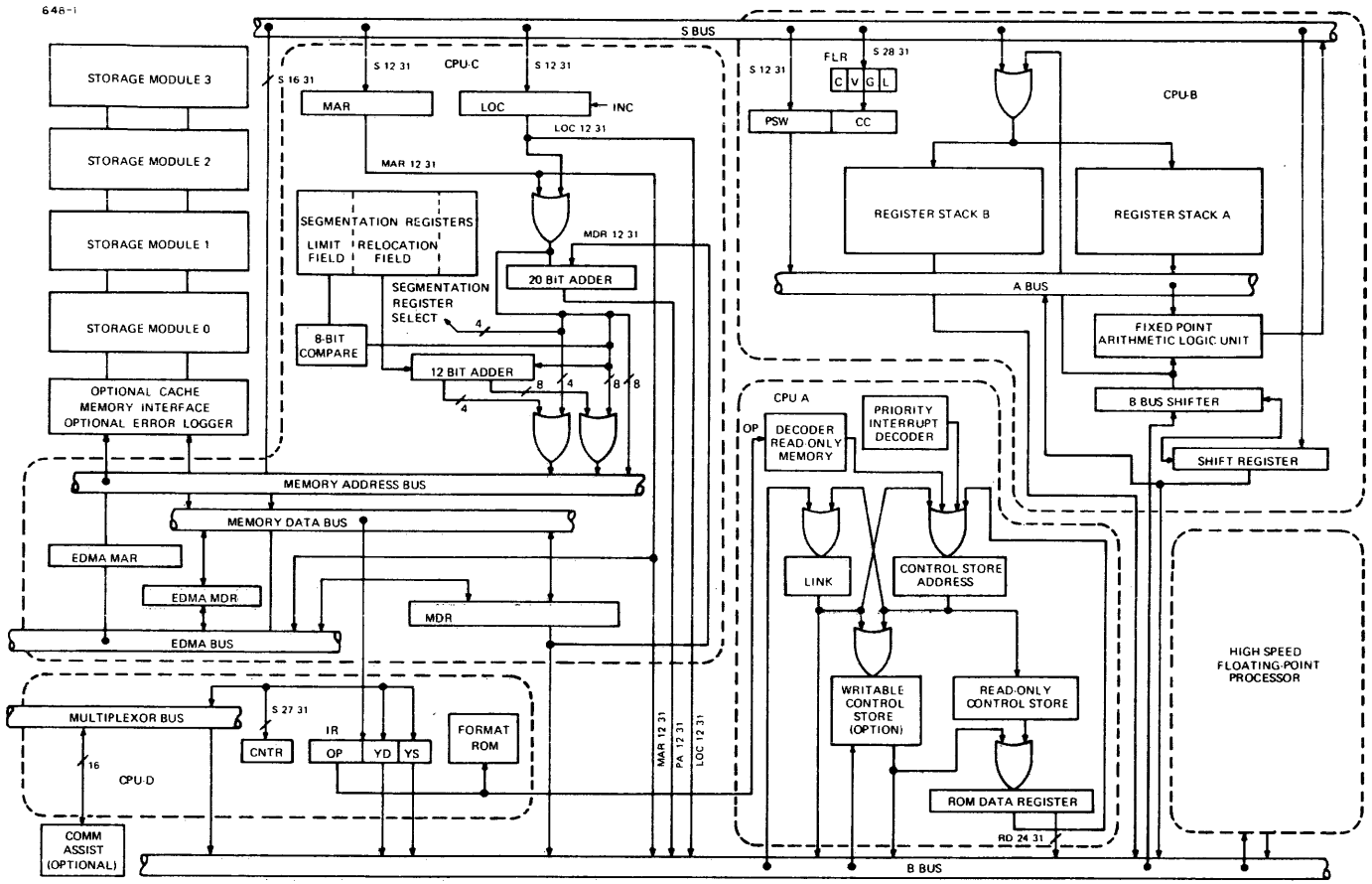


Figure 1-1 Model 3220 Processor Block Diagram

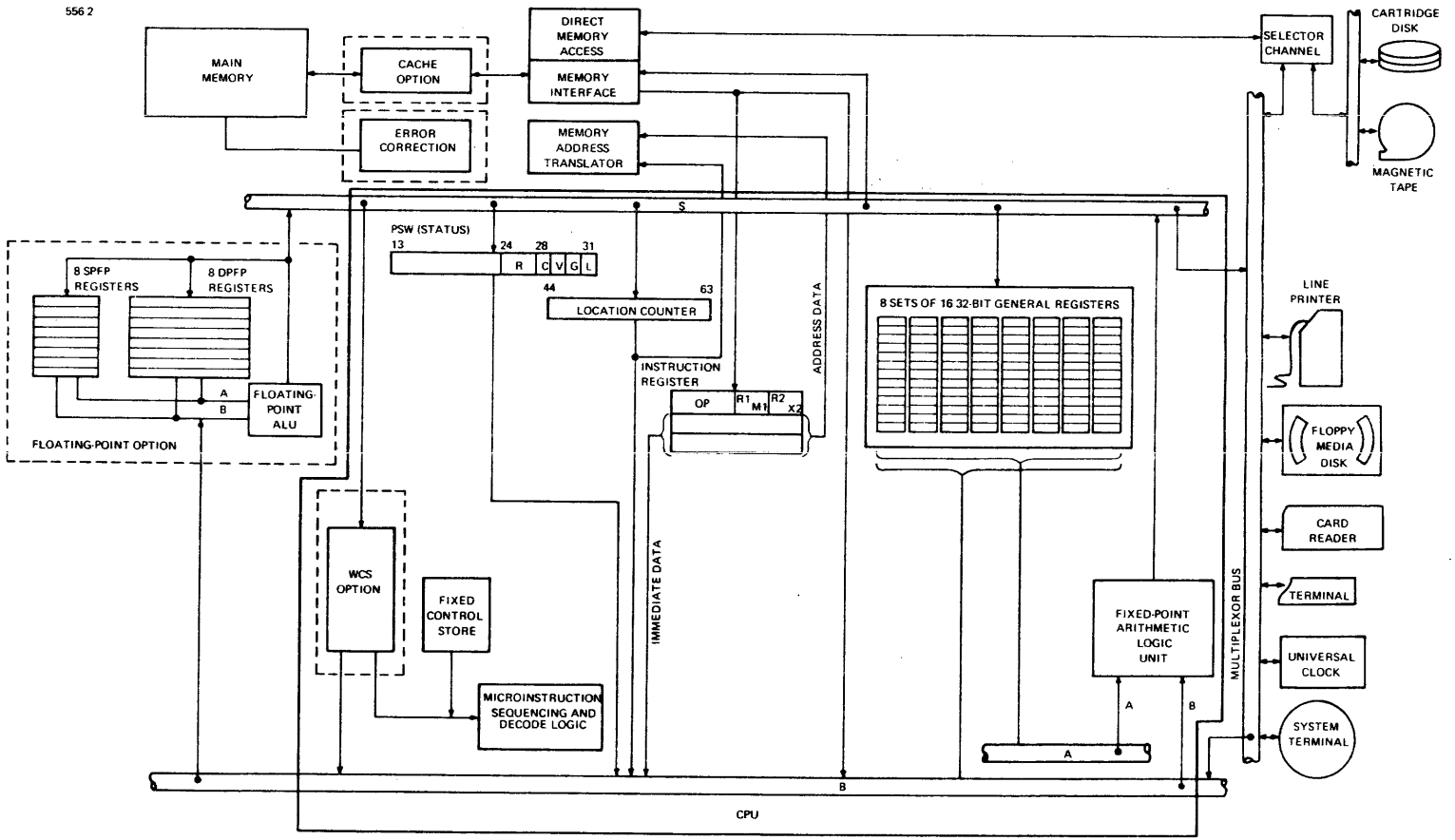


Figure 1-2 Model 3230 Processor Block Diagram

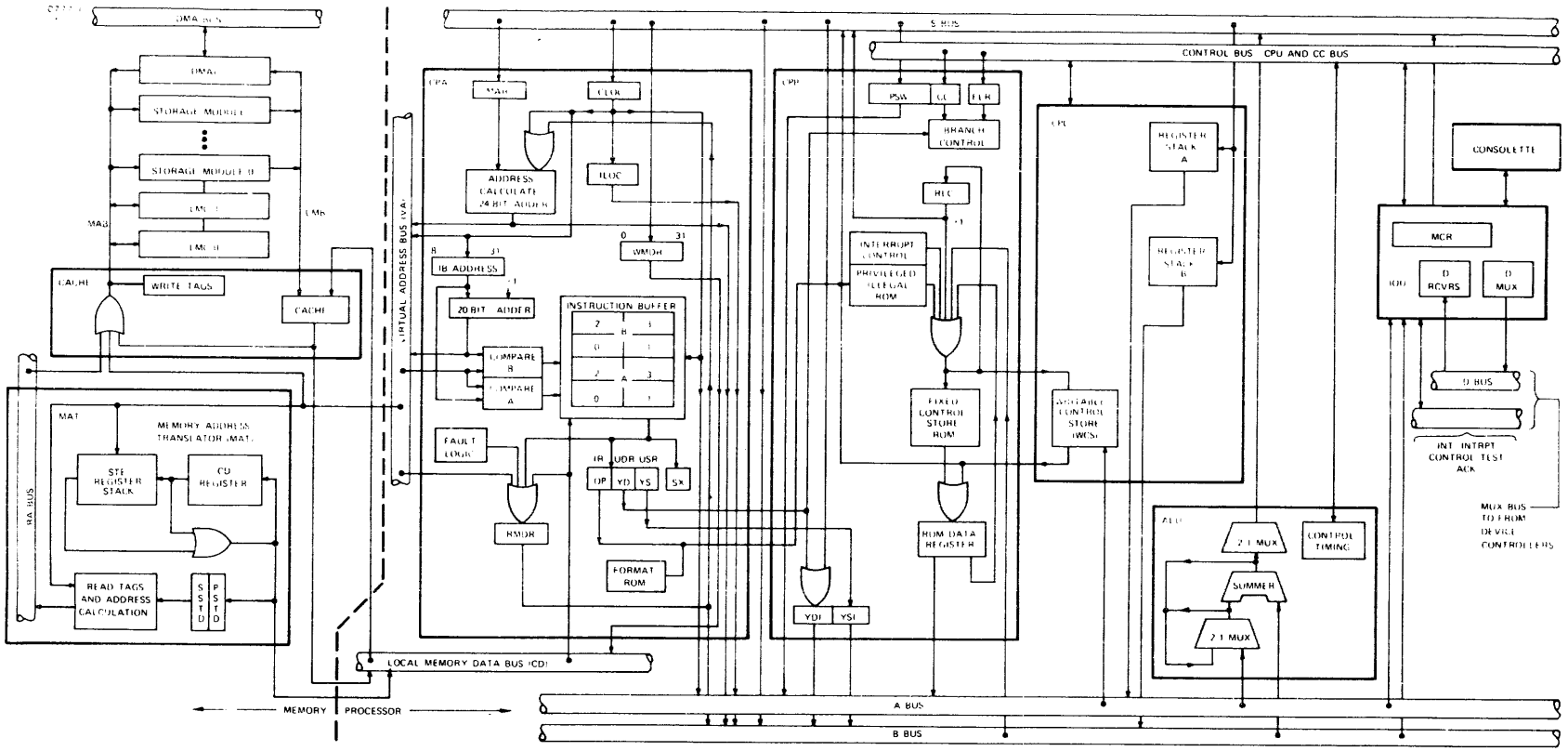


Figure 1-3 Model 3240 Processor Block Diagram

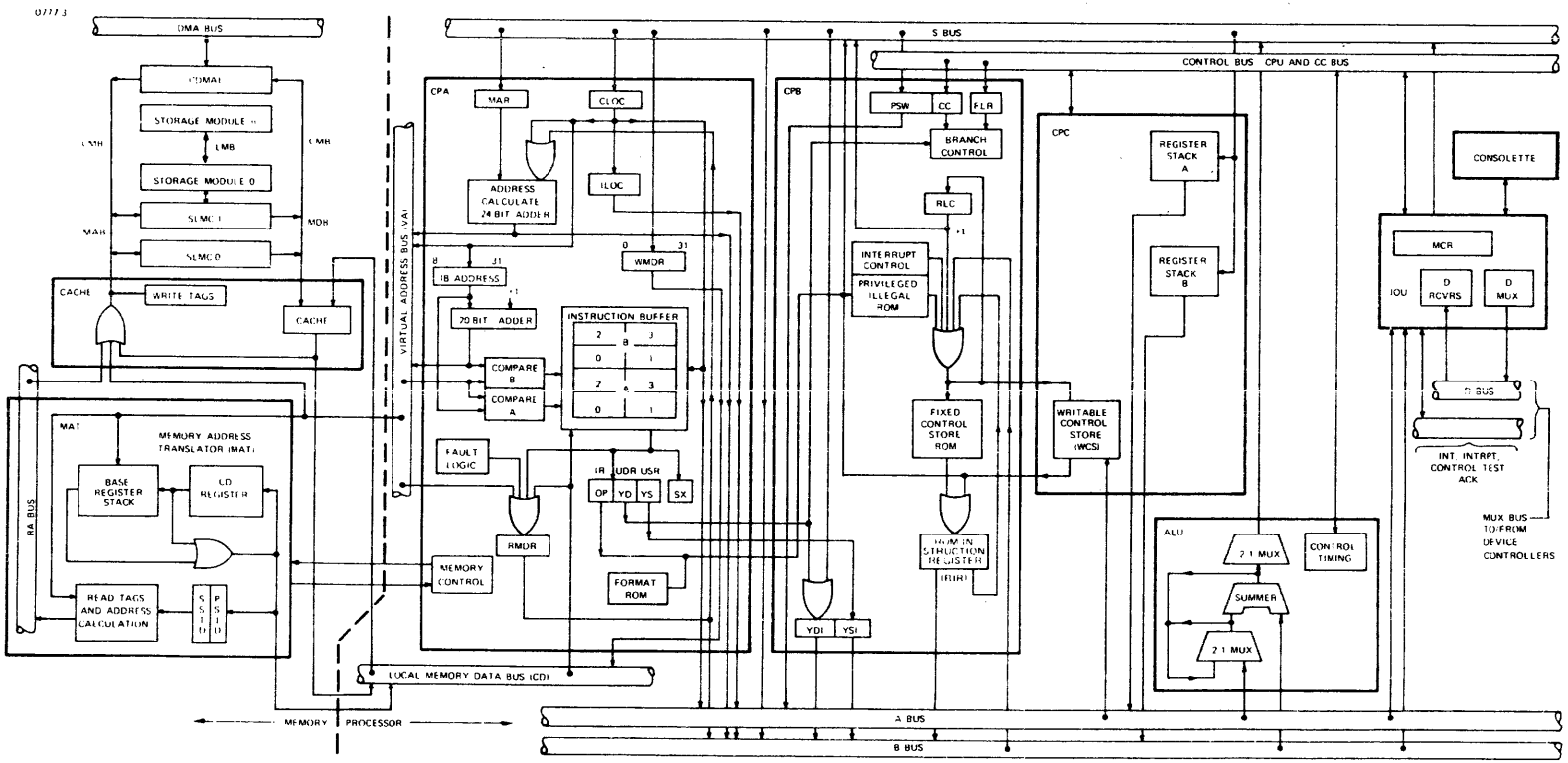
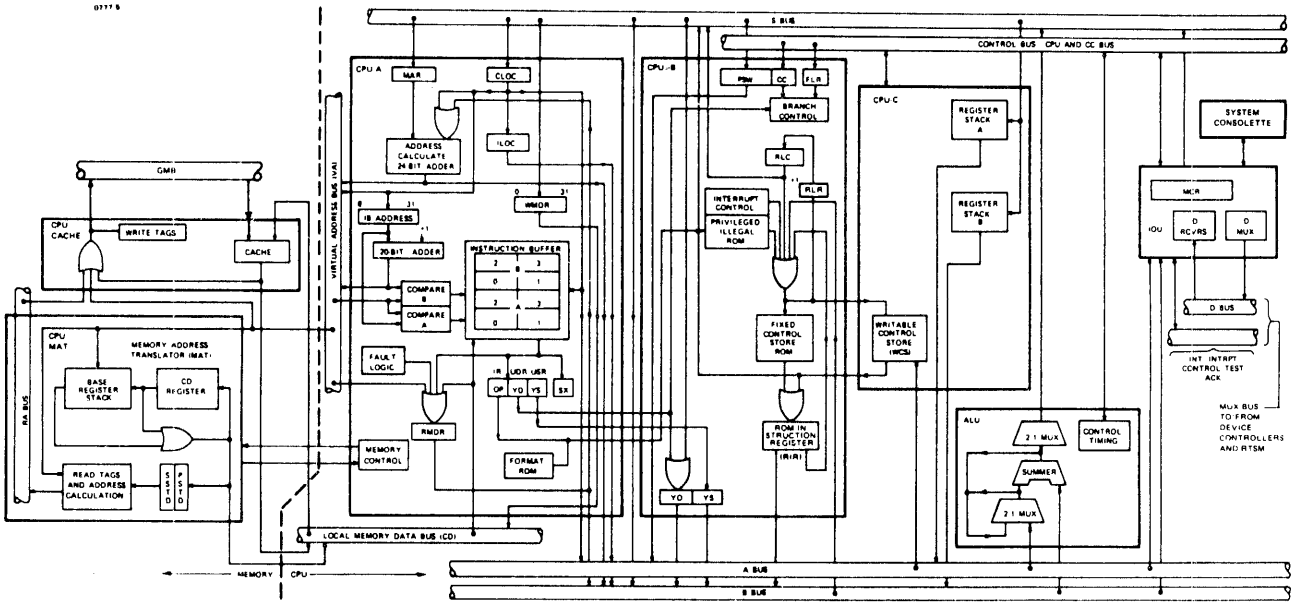
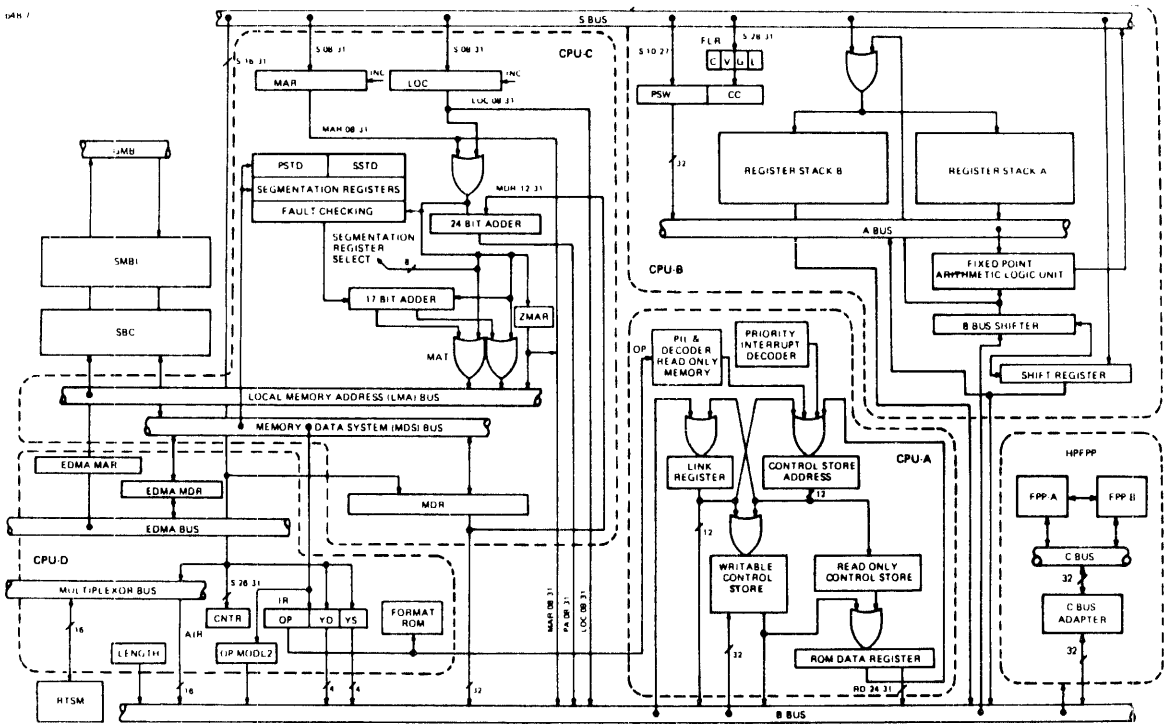


Figure 1-4 Model 3250 Processor Block Diagram



a. CPU Block Diagram



b. APU Block Diagram

Figure 1-5 Model 3200MPS System Block Diagrams

Through WCS the user can extend the machine architecture without hardware modification. With WCS instruction set enhancements, high speed algorithms, and specialized application functions can be added easily. WCS is also used to store the microcode routines supplied with the FORTRAN enhancement package (FEP). FEP provides frequently used mathematical routines to speed the execution of FORTRAN VII programs. These routines are called by the FORTRAN VII WCS run-time library (RTL) at execution time.

Microcode source programs for WCS must be written using the microinstruction set listed in the microprogramming reference manual applicable to the processor to be used. The microcode source must then be assembled by MICROCAL. The resulting microcode object program can then be transferred to WCS and debugged though the use of the appropriate WCS support program.

These support programs are:

- WCSUPP

  This is one of two WCS support programs initially made available for the earlier models of Perkin-Elmer Series 3200 processors. The WCSUPP aids in transferring microcode from main memory to WCS, and in debugging microcode. It also allows examination and modification of cells, insertion of breakpoints in WCS microcode routines, and restoration of the contents of WCS after a power fail/restore sequence. There are three versions of the WCSUPP: one for the Model 3220 processor, another for the Model 3230 processor, and a third for the Models 3240 and 3250 processors.

- WCSLPFM

  This is the second of the two WCS support programs initially made available for earlier Perkin-Elmer Series 3200 processors. WCSLPFM establishes debugged microcode stored as an image file on a direct access device. WCSLPFM cannot be used to debug microcode. There is a single version of WCSLPFM for all Perkin-Elmer Series 3200 uniprocessors.

- WCSAIDS

  This is one of three more recent and flexible WCS support programs available. The debugging facility previously available with the WCSUPP program is now easier to use with the WCSAIDS program. There are three versions of WCSAIDS: one for the Model 3220 processor, another for Model 3230 processor and the Model 3200MPS System APU(s), and a third for the Models 3240 and 3250 processors and the Model 3200MPS System CPU.

- WCSLINK

  This is the second of the more recent WCS support programs. The facility to build a WCS image file, from the microcode object file created by MICROCAL, is provided by WCSLINK. WCSLINK cannot be used to debug microcode. It can be used to build the WCS image file for any Perkin-Elmer Series 3200 processor.

- MPSLPFM

  This is the third of the more recent WCS support programs. MPSLPFM may be used to load the WCS of both the Model 3200MPS System CPU and APU(s) with debugged microcode from WCS image files stored on a direct access device. MPSLPFM does all that WCSLPFM did and has the added advantages of message responses. MPSLPFM may be used to load the WCS of any Perkin-Elmer Series 3200 processor; however, the larger memory requirements of MPSLPFM must be kept in mind if it is used in a uniprocessor configuration. MPSLPFM cannot be used to debug microcode.

In succeeding chapters, this manual will provide the information needed to use these WCS support programs.

1.2 WRITABLE CONTROL STORE (WCS) FUNCTIONAL DESCRIPTION

WCS is a writable extension of fixed read-only memory (ROM) and is addressable through the ROM location counter (RLC). For the Model 3230 processor (equipped with 4K words of WCS) and the Model 3200MPS System APU(s), the ROM is located between 000_{16} and FFF_{16}. For all other Perkin-Elmer Series 3200 processors, the ROM is located between 000_{16} and $7FF_{16}$. The WCS for the Model 3230 (equipped with 4K words of WCS) and the Model 3200MPS System APUs is located between 000_{16} and FFF_{16}. (This address space differs from the ROM address space. For more information see the appropriate processor manuals.) The WCS for all other Perkin-Elmer Series 3200 processors is located between 800_{16} and FFF_{16}.

WCS microcode is volatile; if there is a power failure, the data stored in WCS is erased. When this happens, it must be reloaded. Reloading WCS is accomplished through WCSLINK or WCSUPP.

Four assembly level instructions are provided in the processor instruction set to reference and manipulate WCS:

1. Enter control store (ECS)
2. Branch to control store (BDCS)
3. Read control store (RDCS)
4. Write control store (WDCS)

By using these instructions, the user can write into WCS, read from WCS, and transfer control to the WCS resident microcode. Once control has been transferred to the microcode routine in WCS, any microinstruction can be executed. There are microinstructions provided that can disable the memory address translator (MAT), modify the contents of all general and floating point registers, control and initialize the program status word (PSW), and disable or enable interrupts during execution of the microcode routine. These microinstructions, and the precautions for using them, are described in the appropriate microprogramming reference manual.

Although WCS can extend the flexibility of the machine, certain limitations and precautions exist. WCS memory is only a supplement to the fixed control store (FCS); therefore, the user cannot delete or modify user level instructions or machine features located in the ROM control store. Also, a new emulator cannot be created in WCS; the user can only add to the existing one.

1.3 STATEMENT SYNTAX CONVENTIONS

Throughout this manual, these statement syntax conventions are used to represent instruction formats:

CONVENTION	USE
Capital letters, parentheses, and punctuation marks	must be entered exactly as shown.
Lowercase letters	represent parameters or information provided by the user.

```
ESTABLISH progname,fd
```

Underlining	indicates only the underlined portion of the entry is required.

```
PAUSE
```

Braces	represent required parameters from which one must be chosen.

```
TRANSFER {IMAGE} staddr,endaddr
         {WCS  }
```

Brackets — represent an optional parameter that can be chosen.

DUMP IMAGE [LOADER] staddr,endaddr

Braces inside brackets — represent optional parameters from which one can be chosen.

Lettering with shading — represents a default option.

ZAP [{wcsaddr / all}]

Ellipsis — represents an indefinite number of parameters or a range of parameters.

MODIFY x_1 [x_2...x_{16}]

1.3.1 File Descriptors (fds)

fds are entered in the following format.

Format:

[{voln: / dev:}] [filename [.[ext]]] [/S]

Parameters:

voln: — is a 1- to 4-character alphanumeric string specifying the name of a volume. The first character must be alphabetic and the remaining, alphanumeric. If the volume name is omitted, the default is the system volume.

dev:	is a 1- to 4- character alphanumeric string specifying a device name. The first character must be alphabetic and the remaining, alphanumeric.
filename	is a 1- to 8-character alphanumeric string specifying the name of a file. The first character must be alphabetic and the remaining, alphanumeric. If a filename is specified when a device name is specified, the filename is ignored.
.ext	is a 1- to 3-character alphanumeric string specifying the name of the extension to a filename.
/S	indicates a system file.

The file class always defaults to a system file when using the WCS support program. Account numbers may not be specified. See the OS/32 Application Level Programmer Reference Manual for more information on fds.

Example:

M300:METER.VAN	M300: is the volume name, METER is the filename, and .VAN is the extension.

CHAPTER 2
CREATING A MICROPROGRAM

2.1 INTRODUCTION

Creating a microcode program using MICROCAL is similar to creating a user level program using common assembly language (CAL). Figures 2-1 and 2-2 show the logical flow for creating a microcode program, from analyzing the problem to debugging microcode in writable control store (WCS). When coding the microcode program, see the appropriate processor microprogramming reference manual for a complete list of microinstructions.

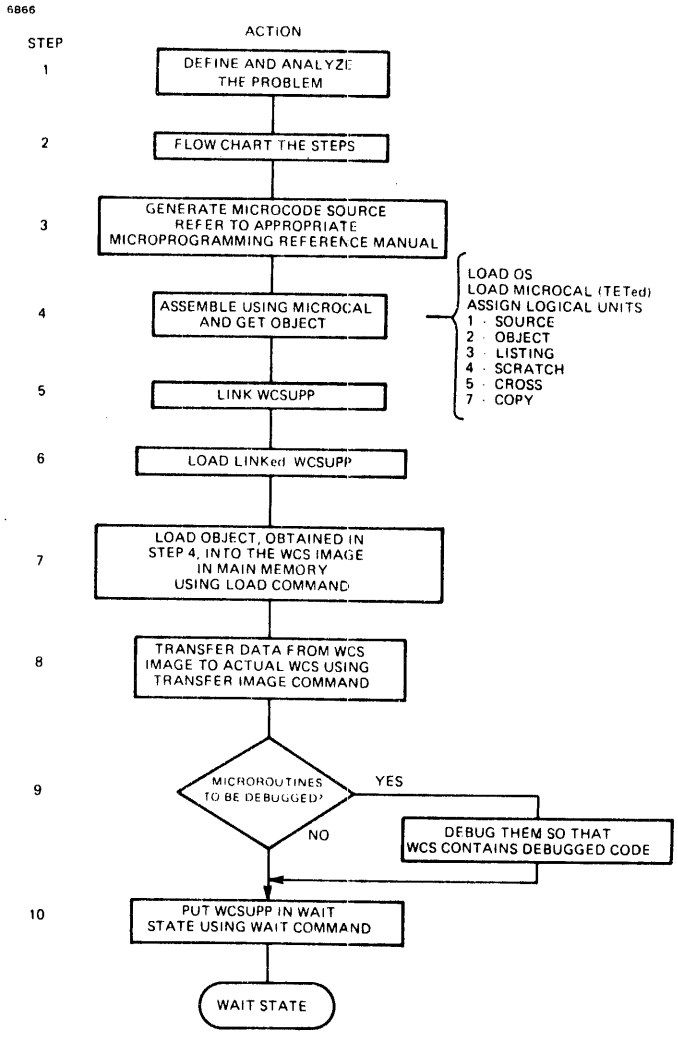


Figure 2-1 Creating Debugged Microcode Routines Using WCSUPP

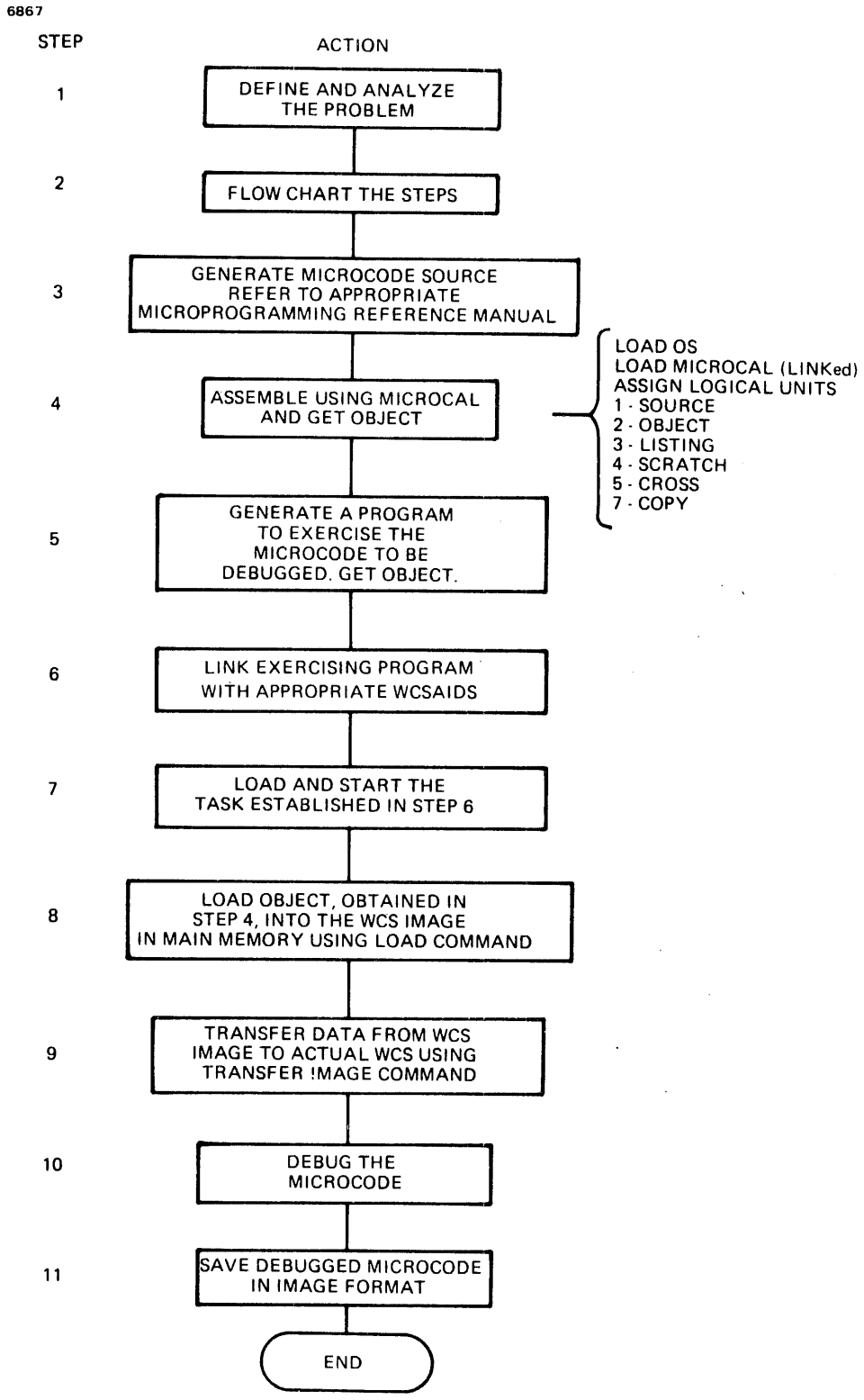


Figure 2-2 Creating Debugged Microcode Routines Using WCSAIDS

2.2 USER LEVEL WRITABLE CONTROL STORE (WCS) INSTRUCTIONS

WCS user level instructions can be used in an executive task (e-task) or a user task (u-task). Before an enter control store (ECS) instruction is executed from a task, the WCS of a processor must contain proper microcode. The first 16 words of WCS must contain the address of the user microcode routine or the illegal instruction handler. This is discussed in more detail in subsequent chapters.

CAUTION

IF AN ECS IS ATTEMPTED AND THE WCS OF A PROCESSOR IS NOT INITIALIZED PROPERLY, A SYSTEM FAILURE CAN OCCUR.

Table 2-1 presents the four WCS instructions and the task types in which each instruction can be used.

TABLE 2-1 WCS INSTRUCTIONS AND ASSOCIATED TASKS

WCS INSTRUCTION	TASK TYPE USED IN
Enter control store (ECS)	u-task e-task d-task
Branch to control store (BDCS)	e-task d-task
Read control store (RDCS)	e-task d-task
Write control store (WDCS)	e-task d-task

The BDCS, RDCS, and WDCS instructions are privileged instructions used only by diagnostic tasks (d-tasks) and e-tasks. Improper use of these three instructions can also cause system failure since e-tasks have no operating system protection. Note that caution should be used when working with e-task and stand alone programs that use these instructions.

The four WCS user level instructions provide the user with four functions: the ability to read from, write to, branch to, and enter WCS memory. See the appropriate processor user's manual for more information on using these instructions.

2.2.1 Write Control Store (WDCS)

The WDCS instruction transfers a buffer containing data located in main memory to an area in WCS. The addresses of both the buffer and the area in WCS memory are specified in the operands of this instruction.

2.2.2 Read Control Store (RDCS)

The RDCS instruction transfers an area in WCS memory to a buffer located in main memory. The addresses of the area in WCS and the buffer in memory are specified in the operands of this instruction.

2.2.3 Branch to Control Store (BDCS)

The BDCS instruction unconditionally branches to a WCS resident microcode routine. The address of the microcode routine is specified in the second operand of this instruction. This instruction can occur anywhere within a user level routine.

During execution of the BDCS instruction, the address of the next sequential instruction is placed in the current location counter (LOC in the Models 3220 and 3230 processors, and in the Model 3200MPS System auxiliary processing unit(s) (APUs); and CLOC in the Models 3240 and 3250 processors and in the Model 3200MPS System central processing unit (CPU)). If the microcode routine entered executes an instruction read (IR) microinstruction without changing the current location counter, the next user level instruction executed will be the one following the BDCS instruction. The current location counter must be modified by the microcode routine prior to an IR if the next user level instruction to be executed is not the next sequential instruction.

The BDCS instruction has unlimited addressing and can branch to any location in ROM control store or WCS. It can also branch to a nonexistent address that causes the system to enter an undefined state. If the system status can change as a result of the execution of a microcode routine, use the BDCS instruction. By using this instruction, the operating system saves the system status before executing the next instruction.

2.2.4 Enter Control Store (ECS)

The ECS instruction unconditionally branches to one of the first 16 fullword locations in WCS memory. The particular location is specified by the value of the R1 field of this instruction. Each of the first 16 fullwords that are reserved should contain a branch microinstruction that points to one of 16 separate microcode routines. If all 16 fullwords are not used, a branch to the address of the illegal instruction interrupt microcode routine should be loaded into the unused fullwords.

When the ECS instruction is read, the current location counter is incremented by four, since the ECS instruction is normally assembled in RI1 format. Figure 2-3 shows the execution sequence of a normal ECS instruction. After the ECS is executed, control is returned to the next sequential instruction. If the user causes the ECS instruction to be assembled in a different format, the microcode routine entered is responsible for correcting the value in the current location counter prior to an IR.

WARNING

IMPROPER USE OF ANY WCS INSTRUCTION CAN CAUSE A SYSTEM FAILURE.

2.3 MICROPROGRAMMING NOTES FOR THE MODEL 3230 PROCESSOR AND THE MODEL 3200MPS SYSTEM AUXILIARY PROCESSING UNIT (APU)

The following restrictions should be observed when writing microprograms for the Model 3230 processor or the Model 3200MPS System APU(s):

1. Scratchpad registers 8 through 15 should not be used or modified by any microprogram in WCS. They are reserved for use by the microprogram in fixed control store (FCS).

2. Scratchpad registers 8 through 15 may be examined through the use of the WCSAIDS EXAMINE command, but they must not be modified via the MODIFY command.

If these restrictions are not observed, a system failure may occur.

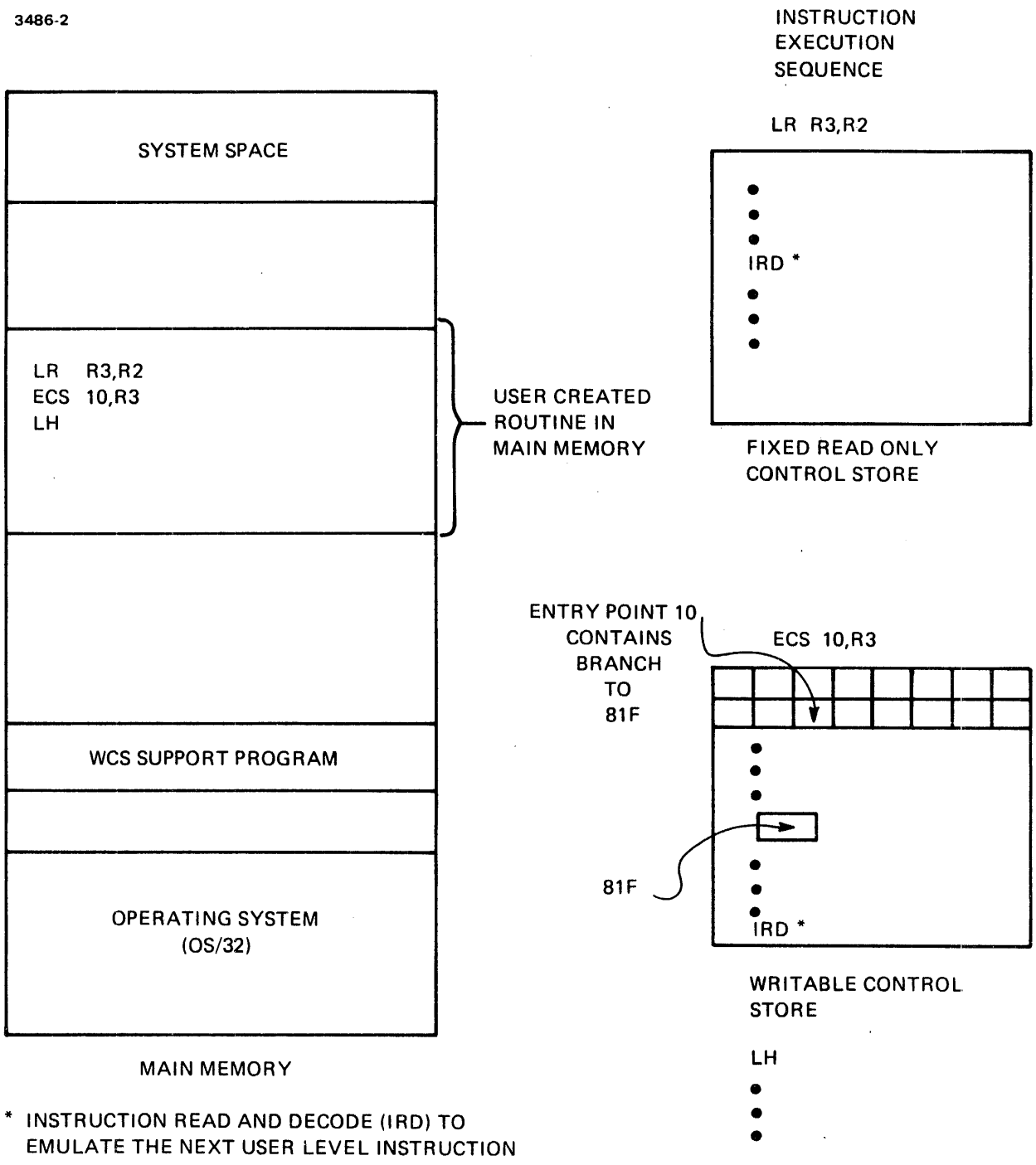


Figure 2-3 Entering Writable Control Store

CHAPTER 3
LINKING, LOADING, AND STARTING THE WRITABLE CONTROL STORE (WCS) SUPPORT PROGRAMS

3.1 INTRODUCTION

The available WCS support programs are: WCSLINK, WCSAIDS, MPSLPFM, WCSUPP, and WCSLPFM.

WCSLINK may be used to build microcode image files for any Perkin-Elmer Series 3200 processor. WCSAIDS has three processor dependent versions. It enables the user to write microcode into WCS via the WCS image buffer, examine memory locations and registers, and to debug microprograms. WCSAIDS is linked with a user program that exercises the microcode to be debugged. MPSLPFM is the loader and power fail monitor for the Model 3200MPS System. MPSLPFM provides a means of loading and initializing WCS after a power failure. Although MPSLPFM may be used on a uniprocessor system, its larger memory requirements should be considered.

WCSUPP is different for each processor. Its function is the same as that of WCSAIDS, however, WCSUPP is built alone as a task rather than linked with a user program. WCSLPFM provides a means of loading and initializing WCS after a power failure for any uniprocessor system. It does not provide any debugging facility.

This chapter describes how to link, load, and start each of the WCS support programs.

3.2 ENVIRONMENT

The three-part WCS support programs have distinct memory requirements:

- WCSLINK: 23kb
- WCSAIDS: 30kb
- WCSLPFM: 8kb
- WCSUPP: 21kb (Models 3220 and 3230 processors)
  19kb (Models 3240 and 3250 processors)
- WCSLPFM: 1kb

For the programs to operate accurately, the system must contain these features:

- Perkin-Elmer Series 3200 processor
- Required amount of main memory
- WCS hardware
- Command input device
- List device
- Load modules of WCSLINK, MPSLPFM, WCSUPP, and WCSLPFM (as required) stored on disk or magnetic tape
- Object module of WCSAIDS (if used)
- Microcode routine in object or image format
- Power fail/auto restart option

The WCS support programs also use some of the operating system resources such as:

- Supervisor calls (SVCs)

SVC 1		Input/output (I/O) operations
SVC 2	code 1	Pause
	code 5	Fetch pointer
	code 6	Unpack binary data
	code 7	Log message
	code 8	Interrogate clock
	code 9	Fetch date
	code 15	Pack numeric data
	code 16	Pack file descriptor
	code 17	Scan mnemonic table
	code 18	Move ASCII characters
SVC 3		End of task
SVC 6		Intertask communications (Model 3200MPS System only)
SVC 7		File handling services
SVC 9		Load task status word (TSW)
SVC 13		Auxiliary processing unit (APU) control (Model 3200MPS System only)

- Traps
- Trap wait feature

3.3 LINKING, LOADING, AND STARTING

The WCS support programs are shipped to the user in common assembly language (CAL) object format. Before loading the support program in the OS/32 environment, each must be built as described below.

The WCS support programs use specific logical units for input and output. Internal processing uses logical unit 1 (lul). The list output device for all error messages, response messages, and warning messages should be assigned to lu3. The command input device for all commands that load, start, and execute microcode routines for WCS should be assigned to lu5. Use of these logical units is outlined in Table 3-1.

TABLE 3-1 LU ASSIGNMENTS FOR WCS SUPPORT PROGRAMS

LU	PURPOSE	PREASSIGNED
1	Internal processing	No
3	List device	Yes
5	Command input device	Yes

The following sections deal separately with WCSLINK, WCSAIDS, and MPSLPFM, and show how to build each program as a task, and then load and start it.

3.3.1 WCSLINK

The WCSLINK program enables the user to build WCS image files from the microcode object file(s) created by MICROCAL. WCSLINK may be built as a user task (u-task).

WCSLINK may be used to build a WCS image file for any Perkin-Elmer Series 3200 processor.

The following sequence illustrates the procedures for building WCSLINK as a task using the OS/32 Link program:

```
*SET LOG PR:
*XDE WCSLINK.TSK
*LOAD,LINK,40              *Load Link.
*START                     *Start Link.
PERKIN-ELMER OS/32 LINKAGE EDITOR Rnn-nn
>MAP PR:,ADDRESS,XREF,     *Obtain maps
ALPHABETIC                 *on a line printer.
>INCLUDE WCSLINK.OBJ       *Use the WCSLINK program.
>BUILD WCSLINK.TSK
>END                       *End the Link task.
```

After WCSLINK is built using Link, the following sequence of commands is used to load and start it.

```
*LO .BG,WCSLINK          Load WCSLINK, built using Link, from
                         a file called WCSLINK.TSK.
*TA .BG                  Make it a background task .
*AS 3,CON:               Assign lu3 to a list device (CON:).
*AS 5,CON:               Assign lu5 to a command input device
                         (CON:).
*START                   Start task.
```

When WCSLINK is started, this message is written to the list device:

```
WCSLINK 03-xxx Fmm Rnn
```

The user can then enter any valid WCSLINK commands from lu5. See Chapter 4.

3.3.2 WCSAIDS

The WCSAIDS can function as a debugging aid. WCSAIDS should be linked with a user program object module(s) which exercises the microprogram to be debugged. Those user programs should be linked as diagnostic tasks (d-tasks).

The following sequence illustrates the procedures for linking WCSAIDS with user program object module(s) and building a task used to debug the microprogram(s):

```
*LO LINK
*START

 >ES TA
 OPTION WO=X100,DFL,FL,DTASK
 >INCLUDE USER.OBJ
 >INCLUDE WCSAIDS.OBJ
 >MAP PR:,ADDRESS
 >BUILD USER.TSK
 >END
```

After WCSAIDS is linked to a user program and USER.TSK is built using Link, the following sequence of commands is used to load and start it:

*LO .BG,USER.TSK	Load USER.TSK, built using Link, from a file called USER.TSK.
*TA .BG	Make it a background task
*AS 3,CON:	Assign lu3 to a list device (CON:).
*AS 5,CON:	Assign lu5 to command input device (CON:).
*START	Start task.
or	or
*START,SINGLE	Start task. Inform task that the processor has single precision floating point support (must be built with option FLOAT).
or	or
*START,DOUBLE	Start task. Inform task that the processor has single and double precision floating point support (must be built with options FLOAT and DFLOAT).

When this task is started, the following message is written to the list device:

```
WCSAIDS 03-xxx Fmm Rnn
```

When the user program (linked with WCSAIDS) is started, WCSAIDS gets control. The user may now issue the commands of WCSAIDS and start the execution of his program by issuing the WCSAIDS START command. When a breakpoint in the microcode is encountered, WCSAIDS gets control, at which time the user may issue the WCSAIDS commands again.

There are three versions of the WCSAIDS: one for the Model 3220 processor, one for the Model 3230 processor and the Model 3200MPS System APU(s), and one for the Models 3240 and 3250 processors and the Model 3200MPS System CPU. The user must run the appropriate version on the specified processor.

The commands accepted by WCSAIDS are described in detail in Chapter 5.

3.3.3 Loader and Power Fail Monitor (MPSLPFM)

The MPSLPFM program may be used to load the WCS of the processor with debugged microcode that has been SAVEd on a direct access device (by WCSLINK, WCSAIDS or WCSUPP).

The following sequence illustrates the procedures for building MPSLPFM as a task using the OS/32 Link program:

```
*LO LINK,LINK
*TA LINK
*START
 >ESTABLISH  TA
 >OPTION  ET,RES,FL,DFL,PRl=(11,11),COM,APM,APC,CON
 >INCLUDE MPSLPFM.OBJ
 >MAP PR:
 >BUILD MPSLPFM.TSK
 >END
```

After MPSLPFM is built using Link, the following sequence of commands is used to load and start it:

```
*LO MPSLPFM,MPSLPFM.TSK
*TA MPSLPFM
*START (options)          (for details on the use of the START
                          command with MPSLPFM, see Chapter
                          6).
```

3.3.4 WCSUPP

The WCSUPP program serves as a debugging aid. It differs from WCSAIDS in that it is built alone as a task rather than linked with user programs. WCS for the Model 3230 processor has been expanded to 4K words and users of this processor should be aware that WCSUPP has not been upgraded to support this expanded WCS.

The following command sequence illustrates how a WCSUPP task for a Model 3220 processor may be built using Link. The same procedure may be adapted to build a WCSUPP task for other Perkin-Elmer Series 3200 processors by properly selecting the object files.

Example:

DSC1: is a disk pack.

OS is a disk volume on the disk device (DSC1:) that contains these files:

LINK.TSK	Link task
TEMP	Temporary file
WCS3220.OBJ	WCS (CAL) object file

```
OS32 xx-yy                    Depress PROTECT on disk drive.
*MA DSC1:,ON                  Mark disk pack on (DSC1:).
 DSC1: OS                     DSC1 has a volume (called OS) of
                              files.
*V OS                         Make OS the current volume.
*SET LOG PR:
*XDE WCS3220.TSK
*LOAD LINK,LINK,40            *Load Link.
*TASK LINK                    *Make it the current task.
*START                        *Start Link.
PERKIN-ELMER OS/32 LINKAGE EDITOR Rnn-nn
>OPTION PRIORITY =
 (11,11),                     *Give the task initial and maximum
LINK>FLOAT,                   *priorities of 11, single and double
LINK>DFLOAT,                  *precision floating point registers,
LINK>RESIDENT,ETASK,          *resident and executive status,
LINK>ABSOLUTE = 0             *and no additional UDL.
***WARNING: ABSOLUTE SPACE LESS THAN 100***
>MAP PR:,ADDRESS,XREF,        *Obtain four load maps
LINK>ALPHABETIC               *on a line printer.
>INCLUDE WCS3220.OBJ          *Use either the full support program
>BUILD WCS3220.TSK            *or load and power fail monitor.
>END                          *End the Link task.
```

The WCS support program load module can now be loaded into main memory from disk file OS:WCS3220.TSK.

After the WCS full support program is built, the following sequence of commands can be used to load and start it.

*LOAD WCS,WCS3220	Load WCS full support program load module from a file called OS:WCS3220.TSK.
*TASK WCS	Make it the current task.
*ASSIGN 3,CON:	Assign list lu to console.
*ASSIGN 5,CON:	Assign command input lu to console.
*START	Start task. Inform task that the processor has no floating point support.
or	or
*START,SINGLE	Start task. Inform task that the processor has single precision floating point support (must be built with option FLOAT).
or	or
*START,DOUBLE	Start task. Inform task that the processor has single and double precision floating point support (must be built with options FLOAT and DFLOAT).

When the full support program is started, this message is written to lu3:

```
WCS SUPPORT PROGRAM 03-xxx Fmm Rnn
```

3.3.5 WCSLPFM

The following sequence of Link commands demonstrates building a WCSLPFM task:

```
*LOAD LINK
*TASK LINK
*START
-PERKIN-ELMER OS/32 LINKAGE EDITOR xx-xxx Rnn-nn
>ESTAB TASK
>OPTION ET,RES,ABS=0,PRI=(11,11)
-WARNING: ABSOLUTE SPACE LESS THAN 100
>INCLUDE WCSLPFM.OBJ
>MAP PR:
>BUILD WCSLPFM.TSK
>END
```

The next sample program sequence of commands illustrates how to start the WCSLPFM version of the support program:

*LOAD WCS,WCS2	Load WCS support program from file OSMT:WCS2.TSK.
*TASK WCS	Make it the current task.
*START ,IMAGE=fd	Start task. Inform task that the WCS image exists on the specified file. If no extension is given, .WCS is assumed.

NOTE

The WCSLPFM is not restartable; it must be reloaded before it is restarted.

When the task starts, this message is displayed on the system console:

```
WCS SUPPORT PROGRAM-LOADER/POWER FAIL MONITOR 03-xxx Fmm Rnn
```

The task performs the following sequence of operations:

1. The specified file is assigned to lul.
2. The image is transferred to WCS memory.
3. The file is rewound and the task enters a power fail/restore trap wait.

The following message is then displayed on the system console:

```
WCS LOADED
```

3.4 WRITABLE CONTROL STORE (WCS) IMAGE BUFFER

WCS support programs maintain a buffer in main memory. For WCSLINK and WCSAIDS, the length of this buffer is 4,096 words (16,384 bytes). For WCSUPP, this buffer is 2,048 words (8,192 bytes).

The WCS image is read into this buffer from object or image files. It can then be examined and/or modified through the various commands available to these programs. The contents of WCS and the WCS image buffer may also be transferred from one to the other.

CHAPTER 4
WCSLINK COMMANDS

4.1 INTRODUCTION

The WCSLINK program provides a facility for the user to build writable control store (WCS) image files from the microcode object file(s) created by MICROCAL. WCSLINK may be built as a user task (u-task) as described in Chapter 3. The commands accepted by WCSLINK are:

- TARGET
- CLEAR
- LOAD
- EXAMINE IMAGE
- MODIFY
- SAVE
- GET
- END

4.2 CHOOSING A PROCESSOR TO BE LOADED

When used to build a WCS image file from microcode object files, WCSLINK should be informed of the model number of the processor for which the image file is to be built.

The TARGET command is used to pass this information to WCSLINK, which places the information in the WCS image file for future use by the other WCS support programs.

Format:

TARGET = nnnnxxx

Parameter:

nnnnxxx	is the model number of the processor for which the microcode image file is to be built, i.e., 3220, 3230, 3240, 3250, 3200CPU (central processing unit of the Model 3200MPS System), and 3200APU (auxiliary processing unit of the Model 3200MPS System). xxx applies only to the Model 3200MPS System CPU and APU.

If the TARGET command is not used, or TARGET = 0 is specified, the microcode is assumed to be intended for the processor on which WCSLINK is running. This model number is encoded in the loader information block (LIB) of the WCS image file by the SAVE command. When MPSLPFM is used to load a WCS from a WCS image file, the model number in the LIB is checked against the model number of the processor containing the WCS to be loaded.

For the TARGET command to be effective on a WCS image file, it must be specified before the CLEAR command. As a response to the CLEAR command, WCSLINK will display the model number of the TARGET processor.

WCSLINK is not capable of doing relocation. It only merges the microcode object files generated by MICROCAL. It is the user's responsibility to see that the object contains only absolute address references.

4.3 MANAGING THE CONTENTS OF THE WRITABLE CONTROL STORE (WCS) IMAGE BUFFER

The contents of the WCS image buffer may be cleared, loaded, examined, or modified through the various WCSLINK commands.

4.3.1 Clearing the Writable Control Store (WCS) Image Buffer

Before a new microcode object routine is loaded into the WCS image buffer, clear the buffer with the CLEAR command.

Format:

CLEAR

CLEAR fills the WCS image buffer with a microcode branch to the address of the illegal instruction interrupt handler. This ensures that any errors in the execution of ECS and BDCS instructions cause illegal instruction interrupts.

For the Model 3220 processor, the image buffer (2,048 words) is filled with branch to location 007_{16} in fixed control store (FCS).

For Models 3240 and 3250 processors and the Model 3200MPS System CPU, the image buffer (2,048 words) is filled with branch to location 208_{16} in FCS.

For the Model 3230 processor and the Model 3200MPS System APU(s), the image buffer is 4,096 words in size. All words except $7FE_{16}$ and $7FF_{16}$ contain branch to $7FE_{16}$. Words $7FE_{16}$ and $7FF_{16}$ contain the instructions LI YDI,8 and B 7, YDFF, respectively. Together, these instructions cause a branch to the illegal instruction handler in FCS.

In response to the CLEAR command, WCSLINK displays the model number of the processor for which the image buffer is being cleared.

```
WCS IMAGE CLEARED FOR nnnnxxx
```

where nnnnxxx is the model number. xxx applies only to the CPU and the APU(s) of the Model 3200MPS System.

In order to properly initialize the image buffer for the appropriate processor, a TARGET command should be issued prior to the CLEAR command.

4.3.2 Loading the Writable Control Store (WCS) Image Buffer

After the buffer is cleared, the microcode routine can be loaded by using the LOAD command.

Format:

```
LOAD progname [,fd]
```

Parameters:

progname	is the program name in the label field of the PROG statement in the microcode source routine. If the DUMP IMAGE LOADER command was used to build the file specified by fd, DUMP must be specified as the progname. Discussion of the DUMP IMAGE command is deferred until Chapter 5.
fd	is the file descriptor of the device or file that the routine is stored on. If voln: is omitted, the current user volume is the default. If filename is omitted, progname is the default.

Example:

```
LO COS,DSC1:WCS.TST
```

4.3.3 Examination and Modification of the Writable Control Store (WCS) Image Buffer

The EXAMINE IMAGE and MODIFY commands for WCSLINK allow the user to examine and modify cells in the WCS image buffer. A cell, in the context of the WCS image buffer, is a fullword. The last cell examined is called the current open cell and is ready for modification. To examine the WCS image buffer, use the EXAMINE IMAGE command.

Format:

EXAMINE IMAGE nnn $\left[,\left\{\begin{matrix} m \\ 1 \end{matrix}\right\}\right]$

Parameters:

IMAGE	specifies that the WCS image buffer in main memory is to be displayed.
nnn	is the starting address of the image buffer area to be displayed. For Models 3220, 3240, and 3250 processors, and the Model 3200MPS System CPU this may be any value between 800_{16} and FFF_{16} . For the Model 3230 processor, and the Model 3200MPS System APUs, the valid range for nnn is 000_{16} to FFF_{16} .
m	is the number of cells to be examined. The maximum number for m is 16. If m is omitted, one cell is examined.

After a cell is examined, the contents of the cell are open for modification. When more than one cell is examined (by specifying a value greater than 1 for m), the current open cell is the last cell examined (nnn + m - 1). Only one cell can be modified at any one time. To modify the current open cell, use the MODIFY command.

Format:

```
MODIFY  nnnnnnnn
```

Parameter:

nnnnnnnn	is the hexadecimal number specified to replace the contents of the current open cell.

The modified cell is then displayed. The address of the current open cell is incremented to the next sequential cell. The next sequential cell then becomes the current open cell; its contents are available for modification. The user can modify cells successively without opening each consecutive cell.

4.4 SAVING THE CONTENTS OF THE WRITABLE CONTROL STORE (WCS) IMAGE BUFFER

After the WCS image buffer has been loaded with microcode object, a WCS image file may be built on a direct access device. To build the WCS image file, use the SAVE command.

Format:

SAVE fd

Parameter:

fd	is the file descriptor of the device or file that the microcode image routine is to be stored on. If .ext is omitted, .WCS is the default.

When the SAVE commad is executed, a loader information block (LIB) is built as the first record of the image file. The LIB contains the segment type (7), the size of the WCS image, the model number of the processor for which the image is built, and the date and time the file is created. After it is built, the LIB is transferred, followed by the contents of the WCS image buffer, to the filename specified. The specified device is checked for write and binary attributes. If the specified file is:

- contiguous, it must contain at least 33 records for Models 3220, 3240, and 3250 processors, and the Model 3200MPS System CPU, or 65 records for the Model 3230 processor, and the Model 3200MPS System APU(s),
- indexed, it must have a logical record length of 256 bytes, or
- nonexistent, it is allocated as a new contiguous file on the specified volume, or the system default volume if no volume name is present.

A WCS image file, built and saved in this manner, may be loaded into the WCS of a processor by using WCSLPFM or MPSLPFM. An image file may also be used by WCSUPP or WCSAIDS.

4.5 RETRIEVING A WRITABLE CONTROL STORE (WCS) IMAGE FILE

A WCS image file can be loaded into the WCS image buffer by using the GET command.

Format:

GET fd

Parameter:

fd	is the file descriptor of the device or file that the microcode image is stored on. If .ext is omitted, .WCS is the default.

If target processor information is coded in the LIB of the WCS image file, WCSLINK responds to the GET command with the following message:

THIS IMAGE FILE WAS BUILT FOR nnnnxxx

where nnnnxxx indicates the model number of the processor for which the image file was built.

4.6 TERMINATION OF WCSLINK

Execution of WCSLINK can be terminated by using the END command.

Format:

END

CHAPTER 5
WCSAIDS COMMANDS

5.1 INTRODUCTION

WCSAIDS can function as a debugging aid. WCSAIDS differs from the earlier WCSUPP program in the following respects:

- WCSAIDS does not support the feature of entering the power fail trap wait. The WAIT command is not recognized and the ESTABLISH command will only perform the CLEAR, LOAD, and TRANSFER functions.
- A new TARGET command has been added.
- WCSAIDS should be linked with a user program object module(s) that exercises the microprogram to be debugged. Those user programs should be linked as diagnostic tasks (d-tasks).

When the user program (linked with WCSAIDS) is started, WCSAIDS gets control. The user can then issue the debugging commands of WCSAIDS and start the execution of his program by issuing the START command. When a breakpoint in the microcode is encountered, control is returned to WCSAIDS, at which time the user may issue the debugging commands again.

There are three versions of WCSAIDS: one for the Model 3220 processor, another for the Model 3230 processor and the Model 3200MPS System auxiliary processing units (APUs), and a third for Models 3240, and 3250 processors and the Model 3200MPS System central processing unit (CPU).

The appropriate WCSAIDS version must be run on the appropriate processor. If an attempt is made to run a task built with a version of WCSAIDS inappropriate for the processor being used, WCSAIDS pauses after displaying the following message:

WARNING: THIS UTILITY SHOULD NOT BE RUN ON THIS PROCESSOR!

The commands accepted by WCSAIDS are:

- TARGET
- CLEAR
- LOAD
- EXAMINE
- MODIFY
- SAVE
- GET
- TRANSFER
- ESTABLISH
- INSERT
- ZAP
- START
- GO
- DUMP IMAGE
- PAUSE
- END

5.2 WCSAIDS TARGET COMMAND

If a WCS image file is to be built from microcode object files through the use of WCSAIDS, the user may specify the model number of the processor for which this image is to be built. The model number of this image file should be compatible with the model number of the processor for which that particular WCSAIDS was built.

Format:

```
TARGET = nnnnxxx
```

Parameter:

nnnnxxx	is the model number of the processor on which the microcode is to be loaded (i.e., 3220, 3230, 3240, 3250, 3200CPU, 3200APU). This model number must be compatible with the model number of the processor for which the WCSAIDS was built.

If the TARGET command is not used, or TARGET = 0 is specified, the microcode is assumed to be for the processor on which WCSAIDS is running. This model number is encoded in the loader information block (LIB) of the WCS image file created by the SAVE command. The MPSLPFM will cross-check this number (if it is non-zero) against the model number of the processor containing the WCS to be loaded.

For the TARGET command to be effective on a WCS image file, it must be specified before the CLEAR command. As a response to the CLEAR command, WCSAIDS will display the number of the TARGET processor.

5.3 CLEARING THE WRITABLE CONTROL STORE (WCS) IMAGE BUFFER

Before a new microcode object routine is loaded into the WCS image buffer, clear the buffer with the CLEAR command.

Format:

CLEAR

CLEAR fills the WCS image buffer with a microcode branch to the address of the illegal instruction interrupt handler. The image buffer is initialized as previously described in Section 4.3.1. This ensures that any errors in the execution of ECS and BDCS instructions cause illegal instruction interrupts.

5.4 LOADING THE WRITABLE CONTROL STORE (WCS) IMAGE BUFFER

After the buffer is cleared, the microcode object routine can be loaded by using the LOAD command.

LOAD progname,fd

Parameters:

progname	is the program name in the label field of the PROG statement in the microcode source routine. If the DUMP IMAGE LOADER command was used to build the file specified by fd, DUMP must be specified as the progname.
fd	is the name of the device or file the microcode object is stored on. If voln is omitted, the current user volume is the default. If filename is omitted, progname is the default value.

Example:

```
LO COS,DSC1:WCS.TST
```

5.5 CELL EXAMINATION

The EXAMINE command allows the user to examine memory cells.

A cell can be any of the following locations in memory:

- a fullword in WCS memory, or
- a fullword in the WCS image buffer, or
- a register image, or
- a halfword in main memory.

The last cell examined is called the current open cell and is ready for modification.

Throughout the remainder of this manual, several of these memory cells are referred to frequently.

The following cells pertain to all Perkin-Elmer Series 3200 processors:

- the program status word (PSW),
- the first register field of the instruction register (YDI), and
- the index register field of the instruction register (YSI).

The following cells pertain to the Models 3220 and 3230 processors, and the Model 3200MPS System APU(s):

- the memory data register (MDR),
- the location counter (LOC), and
- the shift register (SR).

The following cells pertain to the Models 3240 and 3250 processors, and the Model 3200MPS System CPU:

- the current location counter (CLOC),
- the interrupt location counter (ILOC),
- the read memory data register (RMDR), and
- the write memory data register (WMDR).

5.5.1 Models 3220 and 3230 Processors, and the Model 3200MPS System Auxiliary Processing Unit (APU) EXAMINE Command

The EXAMINE command for the Models 3220 and 3230 processors, and the Model 3200MPS System APU version of WCSAIDS has the following options:

Format:

```
                  { { IMAGE } hexaddr [ , { x } ] }
                  { { WCS   }           [ , { 1 } ] }
                  {                                 }
                  {       hexaddr [ , { y } ]       }
                  {               [ , { 1 } ]       }
                  {                                 }
                  { { ALTERNATE }                   }
                  { { DOUBLE    }                   }
EXAMINE           { { FLOATING  } reg [ , { z } ]   }
                  { { MICRO     }     [ , { 1 } ]   }
                  { { REG       }                   }
                  {                                 }
                  {       LOC                       }
                  {       MDR                       }
                  {       PSW                       }
                  {       SR                        }
                  {       YDI                       }
                  {       YSI                       }
```

Parameters:

IMAGE	specifies that the WCS image buffer in main memory is to be examined. The contents of the image buffer are displayed in disassembled microcode format.
WCS	specifies that WCS memory is to be examined. The contents of WCS memory are displayed in disassembled microcode format.
hexaddr	is the 1- to 6-character hexadecimal address of the first cell to be examined.
x	is a decimal number from 1 to 16 specifying the number of fullwords to be examined.
y	is a decimal number from 1 to 16 specifying the number of halfwords to be examined.
ALTERNATE	specifies that the contents of the alternate register images be examined.
DOUBLE	specifies that the contents of the double precision floating point register images be examined.
FLOATING	specifies that the contents of the single precision floating point register images be examined.
MICRO	specifies that the contents of the microregister images be examined.
REG	specifies that the contents of the general register images be examined. The register set is determined by the program status word (PSW).
reg	is a decimal number from 0 to 15 specifying the first register to be examined.
z	is a decimal number from 1 to 16 specifying the number of registers to be examined.
LOC	specifies that the contents of the LOC register image be examined.
MDR	specifies that the contents of the MDR register image be examined.
PSW	specifies that the contents of the PSW register image be examined.

SR	specifies that the contents of the SR register image be examined.
YDI	specifies that the contents of the YDI register image be examined.
YSI	specifies that the contents of the YSI register image be examined.

Examples:

Display the contents of two consecutive halfwords, starting at location 120_{16} .

```
*EXA 120,2
    120: 4300
    122: 90A2
```

Display the contents of the MDR register image.

```
*EXA MDR
   MDR : 0A1276D2
```

Display in disassembled format the contents of the fullword located at 860_{16} in WCS memory.

```
*EXA W 860
 860 : 006E00F0          L          SR,LENGTH
```

Display the contents of the double precision floating point register 2.

```
*EXA D 2
   DR2 : 00000084C214979A
```

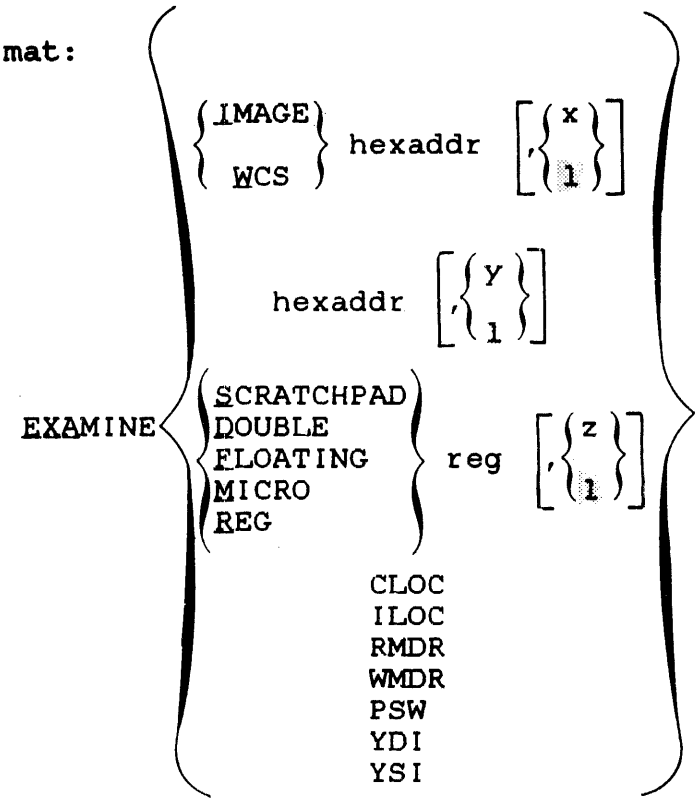
If a single or double precision floating point register is examined and that floating point option was not specified in the START command, one of these messages is displayed:

```
NO FLOATING POINT SUPPORT
NO DOUBLE PRECISION SUPPORT
```

5.5.2 Models 3240 and 3250 Processors, and the Model 3200MPS System Central Processing Unit (CPU) EXAMINE Command

The EXAMINE command for the Models 3240 and 3250 processors, and the Model 3200MPS System CPU version of WCSAIDS, shown below, differs slightly from that used with the other Perkin-Elmer Series 3200 processors.

Format:



Parameters:

IMAGE	specifies that the WCS image buffer in main memory is to be examined. The contents of the image buffer are displayed in disassembled microcode format.
WCS	specifies that WCS memory is to be examined. The contents of WCS memory are displayed in disassembled microcode format.
hexaddr	is the 1- to 6-character hexadecimal address of the first cell to be examined.
x	is a decimal number from 1 to 16 specifying the number of fullwords to be examined.
y	is a decimal number from 1 to 16 specifying the number of halfwords to be examined.

ALTERNATE	specifies that the contents of the alternate register images be examined.
DOUBLE	specifies that the contents of the double precision floating point register images be examined.
FLOATING	specifies that the contents of the single precision floating point register images be examined.
MICRO	specifies that the contents of the microregister images be examined.
REG	specifies that the contents of the general register images be examined. The register set is determined by the program status word (PSW).
reg	is a decimal number from 0 to 15 specifying the first register to be examined.
z	is a decimal number from 1 to 16 specifying the number of registers to be examined.
CLOC	specifies that the contents of the CLOC register image be examined.
ILOC	specifies that the contents of the ILOC register image be examined.
PSW	specifies that the contents of the PSW register image be examined.
RMDR	specifies that the contents of the RMDR register image be examined.
WMDR	specifies that the contents of the WMDR register image be examined.
YDI	specifies that the contents of the YDI register image be examined.
YSI	specifies that the contents of the YSI register image be examined.

5.6 CELL MODIFICATION

After a cell is examined, the contents of the cell are open for modification. Only one cell can be modified at any one time. To modify the current open cell, use the MODIFY command.

Format:

MODIFY $x_1 \left[x_2 \ldots x_{16}\right]$

Parameter:

x	is a hexadecimal character.

The characters specified replace the contents of the location last examined. The maximum number of characters that can be specified is the number displayed on one line by the EXAMINE command. The modified cell containing the new contents is then displayed. The address of the current open cell is incremented to the next sequential cell. The next sequential cell then becomes the current open cell; its contents are available for modification. The user can modify cells successively without opening each consecutive cell. Modifying a location in WCS memory does not modify the corresponding location in the WCS image buffer. The reverse is also true.

Example:

```
*EXA 120,2
    120: 4300
    122: 801E
*MO 90A2
    122: 90A2
*EXA 120,2
    120: 4300
    122: 90A2
```

5.7 SAVING THE CONTENTS OF THE WRITABLE CONTROL STORE (WCS) IMAGE BUFFER

When the WCS image buffer contains a fully debugged microcode routine, it can be saved on any auxiliary storage media. To save the contents of the WCS image buffer, use the SAVE command.

Format:

SAVE fd

Parameter:

fd	is the file descriptor of the device or file that the microcode image routine is to be stored on. If .ext is omitted, .WCS is the default.

When the SAVE command is executed, an LIB is built and transferred, followed by the contents of the WCS image buffer, to the device or the file specified. The specified device is checked for write and binary attributes. If the specified file is:

- contiguous, it must contain at least 33 records for Models 3220, 3240 and 3250 processors, and the Model 3200MPS System CPU, and 65 records for the Model 3230 processor, and the Model 3200MPS System APU(s),
- indexed, it must have a logical record length of 256 bytes, or
- nonexistent, it is allocated as a new contiguous file on the specified volume, or the system default volume if no volume name is present.

5.8 RETRIEVING A WRITABLE CONTROL STORE (WCS) IMAGE FILE

A WCS image file can be loaded into the WCS image buffer by using the GET command.

Format:

GET fd

Parameter:

fd	is the file descriptor of the device or file that the microcode image is stored on. If .ext is omitted, .WCS is the default.

5.9 TRANSFERRING MICROCODE TO AND FROM WRITABLE CONTROL STORE (WCS)

Once the microcode object is loaded in the WCS image buffer in main storage, it can be transferred to WCS memory by the TRANSFER command. Conversely, the TRANSFER command can be used to load the WCS image buffer from WCS memory.

Format:

TRANSFER { IMAGE / WCS } staddr,endaddr

Parameters:

IMAGE	specifies that data should be transferred from the WCS image buffer in main memory to the WCS memory.
WCS	specifies that data should be transferred from the WCS memory to the WCS image buffer in main memory.
staddr	is the starting address, in the source memory, of the data to be transferred.
endaddr	is the ending address, in the source memory, of the data to be transferred.

Starting and ending addresses range from 800_{16} to FFF_{16} for Models 3220, 3240, and 3250 processors, and the Model 3200MPS System CPU, and from 000_{16} to FFF_{16} for the Model 3230 processor and the Model 3200MPS System APU(s).

The data defined by the starting and ending addresses in the source memory is transferred to the same location in the destination memory.

Example:

```
T I 850,94F
```

The microcode in the WCS image buffer that occupies locations 850_{16} through $94F_{16}$ is transferred to the same locations in WCS memory.

Example:

```
T W 800,AFB
```

The microcode in WCS memory that occupies locations 800_{16} through AFB_{16} is transferred to the same locations in the WCS image buffer.

The memory, either IMAGE or WCS, specified in the TRANSFER command is the memory sending the data (source memory); the memory not specified in the TRANSFER command is the memory receiving the data (destination memory).

5.10 ESTABLISHING WRITABLE CONTROL STORE (WCS) MICROCODE ROUTINES

WCS microcode routines, when debugged and saved on a specified device and file, can be established as system WCS microcode. To establish microcode object routines as system WCS microcode, use the ESTABLISH command.

Format:

```
ESTABLISH progname,fd
```

Parameters:

progname	is the name of the microcode object routine that is the label in the PROG statement.
fd	is the file descriptor of the device or file that the microcode object routine is currently stored on.

When the ESTABLISH command is executed, this sequence of operations occurs:

1. The WCS image buffer is cleared.
2. The microcode object routine stored on the specified device and file is loaded into the WCS image buffer.
3. The contents of the entire image buffer are transferred to WCS memory.

NOTE

The ESTABLISH command of WCSAIDS, unlike that of WCSUPP, does not place the task in a power fail trap wait.

The ESTABLISH command can be used only with microcode object routines that are assembled using MICROCAL, or dumped using the DUMP command of WCSAIDS. Since the contents of the entire image buffer are loaded into WCS memory as a result of the ESTABLISH command, the first 16 fullword locations of the image buffer should contain the appropriate addresses of all the microcode routines that it contains. If an error occurs during a load operation, an error message is displayed and the next user command is requested.

After the microcode routine is established as system microcode, this message is displayed on the system console:

MICROCODE TRANSFERRED TO WCS

5.11 INSERTING AND REMOVING BREAKPOINTS IN WRITABLE CONTROL STORE (WCS) MEMORY

To aid in debugging the routine, breakpoints stop the execution of a microcode routine located in WCS memory and transfer control back to WCSAIDS. A maximum of eight breakpoints can be inserted at any one time. When a breakpoint takes place, execution of the routine stops and control is transferred to WCSAIDS. To insert a breakpoint, WCSAIDS saves the fullword microinstruction at the specified location, inserts a link microinstruction, and branches to the register save microcode routine located in high memory of WCS. (This routine is loaded in high memory of WCS by WCSAIDS whenever a breakpoint is inserted or a GO WCS command is used.) In addition to saving the double precision, single precision, general, and scratchpad (alternate) registers, the register save microcode routine saves or transfers the following registers in their corresponding areas in the image buffer and returns control to WCSAIDS.

Models 3220 and 3230 processors, and the Model 3200MPS System APU(s)

- PSW
- LOC
- MDR
- YDI
- YSI
- SR
- Microregisters

Models 3240 and 3250 processors, and the Model 3200MPS System CPU

- PSW
- ILOC
- CLOC
- RMDR
- WMDR
- YDI
- YSI
- Microregisters

See the EXAMINE command for the definitions of these keywords.

When a breakpoint is encountered, the execution of the microprogram is stopped and control is passed to WCSAIDS. The user may then issue any of the commands available to WCSAIDS. Execution of the microprogram may be continued by issuing the GO command.

NOTES

1. Use caution when setting a breakpoint at a register-to-register transfer instruction or a branch and link microinstruction. When the GO command is used to continue execution following a breakpoint, the breakpoint instruction is executed at the top of WCS.

2. The amount of high memory reserved by WCSAIDS for the register save microcode routine is processor dependent. For the Model 3220 processor, the last 64 fullwords are reserved; for the Model 3230 processor and the Model 3200MPS System APU(s), the last 66 fullwords; for Models 3240 and 3250 processors, the last 46 fullwords. Do not place microcode in the WCS memory reserved for WCSAIDS when using breakpoints.

3. A TRANSFER IMAGE command must have been issued before breakpoints can be inserted.

4. For Models 3220 and 3230 processors, and the Model 3200MPS System APU(s), the contents of the link register are destroyed when a breakpoint is encountered.

5.11.1 Inserting Breakpoints

To insert a breakpoint in a microcode routine located in WCS, use the INSERT command. Inserting a breakpoint does not change the contents of the WCS image buffer.

Format:

INSERT wcsaddr [,{n / 1}]

Parameters:

wcsaddr	is the address of a microcode instruction in a routine located in WCS memory where the breakpoint is to be inserted.
n	is a decimal number specifying the number of times the instruction is encountered before the breakpoint actually has any effect. If n is not specified, 1 is the default value.

Example:

```
I 92C,13
```

The microinstruction located at $92C_{16}$ is encountered 12 times in normal sequence. A breakpoint is taken on the thirteenth encounter.

When a breakpoint is encountered, the following message is displayed on lu3:

```
BREAKPOINT HIT AT wcsaddr
```

wcsaddr	is the address of the instruction where the breakpoint was encountered.

5.11.2 Removing Breakpoints

To remove a breakpoint previously inserted in a microcode routine located in WCS memory, use the ZAP command. Removing a breakpoint does not change the contents of the WCS image buffer.

Format:

$$\underline{Z}AP \left\{ \begin{matrix} wcsaddr \\ all \end{matrix} \right\}$$

Parameter:

wcsaddr — is the address of the breakpoint to be removed.

The breakpoint located at the specified WCS address is removed, and the microcode instruction at the specified WCS address is restored. If no WCS address is specified, all existing breakpoints are removed.

Example:

```
Z 92A
```

The breakpoint at location $92A_{16}$ is removed, and the microcode instruction located at $92A_{16}$ is restored.

5.12 STARTING THE USER PROGRAM

When the user program, developed to exercise the microprogram (and linked with WCSAIDS), is started, WCSAIDS gets control. The user can now issue the debugging commands of WCSAIDS and start the execution of the program by issuing the START command.

Format:

```
START nnnn
```

Parameter:

nnnn	is a hexadecimal address where the execution of the user program is to begin. If nnnn is omitted, execution of the user's program is started at location 100_{16} .

When a breakpoint in the microcode is encountered, WCSAIDS gets control, at which time the user may issue the debugging commands again.

5.13 MICROPROGRAM EXECUTION

A microcode routine loaded in WCS memory can be executed by issuing the GO WCS command. To start a microcode routine, use the GO WCS command.

Format:

GO WCS wstaddr

Parameter:

wstaddr	is the address in WCS where execution begins.

The WCS starting address is checked to see if it is within the range of the WCS address limits. If the address is valid, the registers are restored, and control is transferred to the WCS starting address specified in the GO command.

Example:

Start the microprogram at WCS location 920_{16} :

```
G W 920
```

A TRANSFER IMAGE command must have been issued prior to using the GO command. If it was not, this error message is displayed:

```
MICROCODE NOT YET TRANSFERRED
```

5.14 RESUMING EXECUTION AFTER A BREAKPOINT

If a microcode routine is interrupted by a breakpoint, the microcode execution may be resumed with the microcode instruction located at the address specified by the breakpoint. To continue a microcode routine after a breakpoint is encountered, use the GO command.

Format:

```
GO
```

When the GO command is executed after a breakpoint takes place, the registers are restored, and the microcode instruction located at the address specified by the breakpoint is executed in high memory of WCS. If the microcode instruction at the breakpoint is not a branch instruction, a branch is then taken to the microcode instruction following the instruction located at the breakpoint, and execution continues.

When WCSAIDS is being executed on behalf of a task, no other task should be allowed to execute any of the microcode instructions via ECS or BDCS instructions. If this restriction is not observed, results are unpredictable and a system crash may occur.

5.15 USING THE DUMP IMAGE COMMAND

Use the DUMP IMAGE command to copy portions of the WCS image buffer to an output device in disassembled microcode format or common microassembler object format.

Format:

```
DUMP IMAGE [LOADER] staddr,endaddr,fd
```

Parameters:

LOADER	specifies that the data should be dumped in common microassembler object format.
staddr	is the starting address of the area in the WCS image buffer to be dumped.
endaddr	is the ending address of the area in the WCS image buffer to be dumped.
fd	is the file descriptor of the device or file that the dump is copied to or stored on.

The DUMP IMAGE command dumps, to the specified file or device, the area in the WCS image buffer specified by the starting and ending addresses. This dump is output in disassembled microcode format (see Appendix G) unless LOADER is specified. If LOADER is specified, the fd must be capable of a binary write. This LOADER dump is output in common microassembler object format with the progname DUMP at the beginning of the output file for identification.

After the DUMP IMAGE command is executed, the following message is printed or displayed on the specified device:

```
I DUMP FROM staddr TO endaddr
```

5.16 PAUSING THE WCSAIDS TASK

The WCSAIDS task can be paused during execution to allow the user to do some intermediate processing. To pause the task, use the PAUSE command.

Format:

```
PAUSE
```

To continue execution of a task after the PAUSE command, the operator should use the operating system command CONTINUE.

5.17 TERMINATING WCSAIDS

Execution of WCSAIDS can be terminated by using the END command.

Format:

```
END
```

CHAPTER 6
LOADER AND POWER FAIL MONITOR (MPSLPFM) START OPTIONS AND MESSAGES

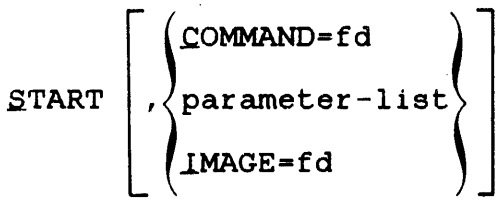
6.1 INTRODUCTION

The linking, loading, and starting of the loader and power fail monitor (MPSLPFM) program was illustrated in Section 3.3.3. MPSLPFM may be used to load the writable control store (WCS) of uniprocessors with previously debugged microcode that has been saved on a direct access device through the use of WCSLINK or WCSAIDS. With the Perkin-Elmer Model 3200MPS System, the MPSLPFM may be used to load the WCS of up to ten processors: one central processing unit (CPU) and up to nine auxiliary processing units (APUs) with debugged microcode that has been SAVEd on a direct access device by WCSLINK or WCSAIDS. The manner in which a CPU or an APU may be associated with a microcode image file is described in the following sections.

6.2 STARTING THE LOADER AND POWER FAIL MONITOR (MPSLPFM)

The format of the START command of MPSLPFM allows the specification of individual microcode images to be loaded into the WCS of the CPU and the WCS of each APU.

Format:



Parameters:

COMMAND=fd	directs the MPSLPFM to read parameters from the command file specified by fd.
parameter-list	is a list of parameters.
IMAGE=fd	specifies the image file for the CPU of the Model 3200MPS System.

The COMMAND=fd is used to direct the MPSLPFM to read a command file containing the desired parameters. The command file must be an ASCII index file. Each record contains a parameter list. An end of file or a /* in the first two columns of the record marks the end of association parameters in the file. In order to be consistent with other Perkin-Elmer products, MPSLPFM assigns logical unit 5 (lu5) to the command file for reading the parameters. When the COMMAND parameter is specified in the START command, there should not be any other START parameter.

A parameter-list is a list of parameters. Each parameter may be an association-parameter, ON or OFF. Consecutive parameters are separated by blanks or commas. An association-parameter associates one or more processors with a microcode image file. The format of the association parameter is:

```
proc-designator=fd
```

where the file descriptor (fd) is the file containing the microcode image, and proc-designator= means zero or more occurrences of proc-designator=. If the extension is not specified on the fd, a default extension of .WCS is assumed. A proc-designator is one of the following:

CPU	designates the CPU.
apu-no	is an integer decimal number designating an APU (0 also means the CPU).
APU	is all APUs which are not otherwise explicitly mentioned in the START command or the command file.

All the processors designated by the proc-designator= list are associated with the image file fd. If the extension is not specified on the fd, a default extension of .WCS is assumed.

For compatibility with the earlier version of the loader and power fail monitor (WCSLPFM), the START option IMAGE=fd has been included. The effect of this START option is identical to that of CPU=fd. Again, if the extension is not specified on the fd, a default extension of .WCS is assumed.

The ON (OFF) option directs the MPSLPFM to leave all the APUs marked ON (OFF) after loading the WCS of the respective APUs. If neither ON nor OFF is specified, the MPSLPFM assumes the default of OFF. Marking the APU ON after the completion of loading makes it possible to schedule a task to be run on an APU as soon as its WCS is defined without the need for operator intervention.

The ON (OFF) option may only be specified with association parameters. If the option is specified more than once in the START option list, or in the command file, the latest option specified becomes effective. The option is applied to all the APUs. If the START command is issued without parameters, the WCS of each processor is initialized with branches to illegal instruction traps.

Examples:

Specify a command file containing parameter lists:

```
START ,COMMAND=PARMFLE.WCS/S
```

Specify association parameters for the CPU and all APUs configured in the system:

```
START ,CPU=CPUMIC.WCS/S,APU=APUMIC.WCS/S
```

Specify association parameters for the CPU and 3 APUs; mark all APUs ON after loading:

```
START ,CPU=CPUMIC.WCS/S,1=2=3=APUMIC.WCS,ON
```

Specify an image file to be loaded to the CPU:

```
START ,IMAGE=CPUMIC.WCS/S
```

NOTES

1. An APU NUMBER greater than the maximum number of APUs cannot be specified.
2. If COMMAND= is specified in the START command, no other option may be specified.
3. If IMAGE= is specified in the START command, no other option may be specified.
4. If the extension is not specified on the image file, an extension of .WCS is assumed.

5. The command file must not contain another COMMAND= option.

6. It is the responsibility of the programmer to see that the appropriate microcode is loaded into the WCS of a processor. If the image file was built using WCSLINK or WCSAIDS, the loader information block (LIB) of the image file contains the model number of the processor for which the image file was built. MPSLPFM checks this information against the model number of the processor for which WCS is to be loaded. If the two do not match, an information message is displayed. Nevertheless, the WCS of the specified processor is loaded with microcode from the image file regardless of suitability.

If a START parameter error is detected, the MPSLPFM processes as many options as possible before terminating with an end of task code of 2.

6.3 LOADING MICROCODE

When all the START parameters are processed, the MPSLPFM starts loading the WCS of the processor(s). MPSLPFM first establishes the WCS of the CPU, followed by the WCS of the individual APUs.

6.3.1 Loading the Writable Control Store (WCS) of the Central Processing Unit (CPU)

If an image file has been specified for the CPU, the MPSLPFM assigns logical unit 1 (lul) to that file, checks the validity of the image file (for valid LIB) and loads the WCS of the CPU with the microcode image contained in that file. Validation of the LIB includes comparing the processor model number with the target information encoded in the LIB. If the CPU is not associated with an image file, the WCS of the CPU is cleared (filled with illegal instruction traps).

6.3.2 Loading the Writable Control Store (WCS) of an Auxiliary Processing Unit (APU)

An APU should be marked OFF for the MPSLPFM to initiate the process of loading its WCS. If the APU is not in the OFF state, the MPSLPFM will display an information message and continue with the next APU. If the APU is marked OFF, the MPSLPFM gets the mapping and control rights to the APU and marks that APU exclusive ON for itself. If an image file has been specified for the APU, the MPSLPFM assigns lul to that file, and loads the WCS of that APU with the microcode contained in the image file after validating the LIB. If an image file has not been specified, the WCS of the APU is loaded with the illegal instruction traps. If the APU is not equipped with WCS, the loading process is omitted for that APU.

When the loading is complete, the APU is marked ON if the option ON was specified in the START option; otherwise it is marked OFF. Finally, the MPSLPFM will release the mapping and control rights to that APU.

If the WCS of an APU is loaded with microcode from an image file, the WCS loaded flag for that APU is set; otherwise it is reset. If an APU was not in the OFF state, or if there were any errors in the process of loading the WCS of an APU, the WCS loaded and WCS initialized flags are reset; otherwise the WCS initialized flag is set. If any error is encountered during the loading process, MPSLPFM will display an error message, resetting the WCS initialized and WCS loaded flags for that processor.

When the WCS image of all the processors has been transferred to the corresponding WCS, the MPSLPFM enters the power fail/restore trap wait while enabling the task message entry.

6.3.3 Selective Loading of the Writable Control Store (WCS) of a Processor

All the processors need not be associated with a microcode image file when the MPSLPFM is initially brought up. In fact, none of the processors need be associated with an image file. An association may be established between a processor and an image file merely by sending an ESTABLISH message to the MPSLPFM. An existing association may also be changed by sending this message. Further, if for some reason, (e.g., an APU was not in the OFF state) the WCS of an APU was not loaded, it is possible to send a RESTORE message to the MPSLPFM to initiate the load without having to reestablish the association. It is also possible to invalidate the WCS of a processor by sending a CLEAR message to the MPSLPFM. It should be noted that the MPSLPFM is able to receive messages only when it is in the power fail/restore trap wait state.

If the association of an APU with its microcode image file has not yet been established, or if the association is to be altered, the ESTABLISH message may be sent to the MPSLPFM.

Format:

```
SEND ESTABLISH [ { COMMAND=fd      } ]
               [ { parameter-list  } ]
               [ { IMAGE=fd        } ]
```

The ESTABLISH message causes the MPSLPFM to associate the image file specified in the parameter with the specified processors. After completion of processing the message, the WCS of only those processors specified in the message are loaded as described in Sections 6.3.1 and 6.3.2. The APU for which an association with a microcode image file is to be established or altered should be in the OFF state in order to successfully load its WCS.

Example:

Establish associations for all APUs and mark them ON:

```
SEND ESTABLISH APU=APUMIC.WCS/S,ON
```

If the association of an APU with its microcode image file has been established earlier, the RESTORE message may be sent to the MPSLPFM to load the WCS of that APU.

Format:

```
SEND RESTORE  apu-no1 [,apu-no2 ,...] [,{ON }]
                                       [{OFF}]
```

Where:

apu-no — is an integer number designating the APU containing the WCS to be reloaded. The process of loading the WCS is described in Sections 6.3.1 and 6.3.2. The APU containing the WCS to be reloaded should be in the OFF state.

Example:

Reestablish the associations of APU numbers 3 and 5, and mark them ON:

```
SEND RESTORE 3, 5, ON
```

The WCS of a processor may be reinitialized with illegal instruction traps by clearing its contents. This may be requested by sending a CLEAR message to MPSLPFM.

Format:

SEND CLEAR apu-no$_1$ [,apu-no$_2$,...] [,{ON / OFF}]

Where:

apu-no — is an integer number designating the APU whose WCS is to be cleared.

Example:

Reinitialize APU numbers 1, 4, and 9:

```
SEND CLEAR 1, 4, 9, OFF
```

The CLEAR message causes the MPSLPFM to load the WCS of the requested APUs with illegal instruction traps. The APUs to be cleared should be in the marked OFF state for the MPSLPFM to load the WCS with the illegal instruction traps.

If the option ON is specified in the above messages, the APUs for which this option is specified is marked ON after its WCS has been loaded. Otherwise, the APU is left marked OFF. The option ON (OFF) specified in the message applies to all the APUs appearing in the message.

6.3.4 Listing of Association Parameters

A list of microcode image files associated with each of the processors can be obtained by sending the following message to the MPSLPFM:

```
SEND STATUS
```

This list will also indicate which processors have WCS loaded or cleared, or had an error in loading the WCS.

6.4 RESTORING THE WRITABLE CONTROL STORE (WCS) OF THE CENTRAL PROCESSING UNIT (CPU) AND AUXILIARY PROCESSING UNIT (APU) UPON POWER FAIL

After a power fail interrupt, the MPSLPFM first reloads the WCS of the CPU with the microcode from the image file specified for it. The process of loading the WCS is described in Section 6.3.1.

When the MPSLPFM is finished with the CPU, it loads the WCS of each APU, one after another, as described in Section 6.3.2. The APUs, however, remain in the state they were in prior to the power fail interrupt. To initiate transfer of microcode image to the WCS of an APU, it must be in the OFF state. The operating system leaves the APUs in the OFF state before invoking MPSLPFM.

It is possible to power down only a set of APUs. When the APUs are brought up again, a message can be sent to the MPSLPFM for reloading the WCS of those APUs. However, the APUs should be marked OFF before sending the message to the MPSLPFM.

The format of the message to be sent to the MPSLPFM (ESTABLISH, CLEAR, or RESTORE) is described in Section 6.3.3.

If a power fail interrupt occurs while processing the previous power fail interrupt, the MPSLPFM will pause. If this happens, MPSLPFM should be cancelled and restarted.

6.5 VERIFICATION

MPSLPFM transfers microcode to the WCS of a processor in parts. After the microcode is transferred to the WCS, the MPSLPFM reads that microcode back from the WCS and compares it with the original code. If there is any mismatch, the MPSLPFM displays an information message, resets the WCS initialized and WCS loaded flags, and proceeds with processing of subsequent APUs. The same action is taken for all kinds of errors; e.g., errors in trying to assign the image file to lu1, errors in reading the image file, improper image file, etc.

6.6 FLAGS

The WCS supported flag, located in the system pointer table (SPT), is set if the WCS of the CPU is loaded with microcode from an image file; otherwise, this flag is reset. This bit may be examined by a user task (u-task) through SVC 2 code 19, option X'01'.

The WCS initialized flag, located in the auxiliary processing block (APB), of an APU is set if the WCS of that APU is initialized either with the microcode from a specified file or with illegal instruction traps; otherwise, this flag is reset.

The WCS loaded flag, also located in the APB, of an APU is set if the WCS of that APU is loaded with microcode from a specified image file; otherwise, it is reset.

If any error is encountered during the transfer of microcode, both the WCS loaded and the WCS initialized flags are reset. These flags may be examined by a u-task through SVC 13 function code X'01' fetch APU status.

6.7 ERROR HANDLING

An illegal START option, or any error in the START option causes the MPSLPFM to terminate with a end of task code of 2. If an error is encountered in an option in the ESTABLISH/RESTORE/CLEAR message, the offending option is ignored; the load is not performed for the APUs specified in the message.

CHAPTER 7
WCSUPP/WCSLPFM PROGRAMS

7.1 INTRODUCTION

The full support program (WCSUPP) and the loader and power fail monitor (WCSLPFM) are the earlier versions of writable control store (WCS) support programs available for the Perkin-Elmer Series 3200 processors.

WCSUPP is available in three versions; one for the Model 3220 processor, another for Model 3230 processors, and a third for the Models 3240 and 3250 processors. The WCS full support program enables the user to load a microcode object routine into the WCS image buffer located in main memory, and then transfer the contents of the image buffer into WCS memory. This program also allows the user to examine and modify data located in main memory, in register images, in the WCS image buffer, or in WCS memory. It provides the debugging capability to insert and remove breakpoints, and the commands to execute WCS resident routines. This support program is also used to initialize WCS each time power is restored.

WCSLPFM does not provide debugging capabilities. It establishes previously debugged microcode from a WCS image file stored on a direct access device.

7.2 USING WCSUPP

The following sections describe the commands available to the user of the full support program.

CAUTION

A WCSUPP TASK SHOULD ONLY BE EXECUTED ON THE PROCESSOR FOR WHICH IT WAS BUILT. IF THIS RESTRICTION IS NOT OBSERVED, UNPREDICTABLE RESULTS OR A SYSTEM CRASH MAY OCCUR.

7.2.1 Managing the Writable Control Store (WCS) Image Buffer

The WCS support program maintains a buffer that contains 2,048 fullwords (8,192 bytes) located in main memory. The support program performs relative and absolute addressing for the buffer. This section describes how to clear, load, and transfer data to and from the WCS image buffer.

7.2.1.1 Clearing and Loading the Writable Control Store (WCS) Image Buffer

Before a new microcode object routine is loaded into the WCS image buffer, clear the buffer with this command:

CLEAR

CLEAR fills the WCS image buffer with a microcode branch to the address of the illegal instruction interrupt handler. This ensures that any errors in the execution of ECS and BDCS instructions cause illegal instruction interrupts.

When the buffer is cleared, the microcode object routine can be loaded by using the following command.

Format:

LOAD progname,fd

Parameters:

progname	is the program name in the label field of the PROG statement in the microcode source routine. If the DUMP IMAGE LOADER command was used to store the routine on a binary or direct access device, DUMP must be specified as the progname.
fd	is the file descriptor of the device or file that the routine is stored on. If voln: is omitted, the current user volume is the default. If filename is omitted, progname is the default.

Example:

```
LO COS,DSC1:WCS.TST
```

7.2.1.2 Transferring Microcode Routines to Writable Control Store (WCS) Memory

Once the microcode object routine is loaded in the WCS image buffer in main storage, it can be transferred to WCS memory by the TRANSFER command.

Format:

TRANSFER {IMAGE / WCS} staddr,endaddr

Parameters:

IMAGE	specifies that data should be transferred from the WCS image buffer in main memory to the WCS memory.
WCS	specifies that the data should be transferred from the WCS memory to the WCS image buffer in main memory.
staddr	is the starting address, in the source memory, of the data to be transferred.
endaddr	is the ending address, in the source memory, of the data to be transferred.

The data defined by the starting and ending addresses in the source memory is transferred to the same location in the destination memory.

Example:

```
T I 850,94F
```

The microcode routine in the WCS image buffer that occupies locations 850_{16} through $94F_{16}$ is transferred to the same locations in WCS memory.

Example:

```
T W 800,AFB
```

The microcode routine in WCS memory that occupies locations 800_{16} through AFB_{16} is transferred to the same locations in the WCS image buffer.

The memory, either image or WCS, specified in the TRANSFER command is the memory from which the data is transferred (source memory); the memory not specified in the TRANSFER command is the memory to which the data is transferred (destination memory).

7.2.2 Cell Examination

The EXAMINE command allows the user to examine cells.

A cell can be any of the following locations in memory:

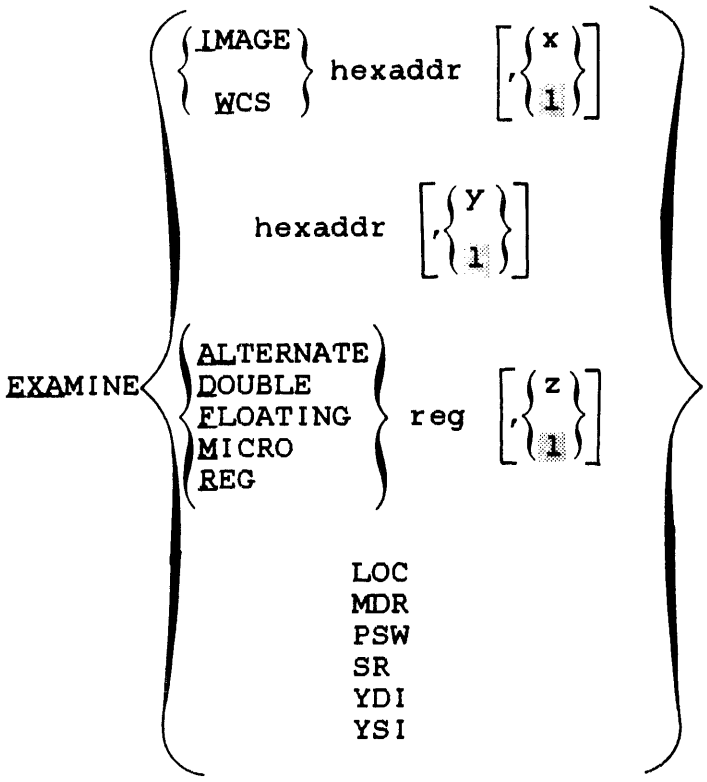
- Fullword in WCS memory
- Fullword in the WCS image buffer
- Register image
- Halfword in main memory

The last cell displayed is called the current open cell and is ready for modification. The cells that can be examined depend upon the version of WCSUPP being used.

7.2.2.1 Models 3220 and 3230 Processors EXAMINE Command

The EXAMINE command for the Models 3220 and 3230 processors version of the full support program has the following options:

Format:



Parameters:

IMAGE	specifies that the WCS image buffer in main memory is to be examined. The contents of the image buffer are displayed in disassembled microcode format.
WCS	specifies that WCS memory is to be examined. The contents of WCS memory are displayed in disassembled microcode format.
hexaddr	is the 1- to 6-character hexadecimal address of the first cell to be examined.
x	is a decimal number from 1 to 16 specifying the number of fullwords to be examined.
y	is a decimal number from 1 to 16 specifying the number of halfwords to be examined.
ALTERNATE	specifies that the contents of the alternate register images be examined.
DOUBLE	specifies that the contents of the double precision floating point register images be examined.
FLOATING	specifies that the contents of the single precision floating point register images be examined.
MICRO	specifies that the contents of the microregister images be examined.
REG	specifies that the contents of the general register images be examined. The register set is determined by the program status word (PSW).
reg	is a decimal number from 0 to 15 specifying the first register to be examined.
z	is a decimal number from 1 to 16 specifying the number of registers to be examined.
LOC	specifies that the contents of the location counter register image be examined.
MDR	specifies that the contents of the MDR register image be examined.
PSW	specifies that the contents of the PSW register image be examined.

SR	specifies that the contents of the SR register image be examined.
YDI	specifies that the contents of the YDI register image be examined.
YSI	specifies that the contents of the YSI register image be examined.

Examples:

Display the contents of two consecutive halfwords, starting at location 120_{16} :

```
*EXA 120,2
    120: 4300
    122: 90A2
```

Display the contents of the memory data register (MDR) register image:

```
*EXA MDR
   MDR : 0A1276D2
```

Display in disassembled format the contents of the fullword located at 860_{16} in WCS memory:

```
*EXA W 860
 860 : 006E00F0          L          SR,LENGTH
```

Display the contents of the double precision floating point register 2:

```
*EXA D 2
   DR2 : 00000084C214979A
```

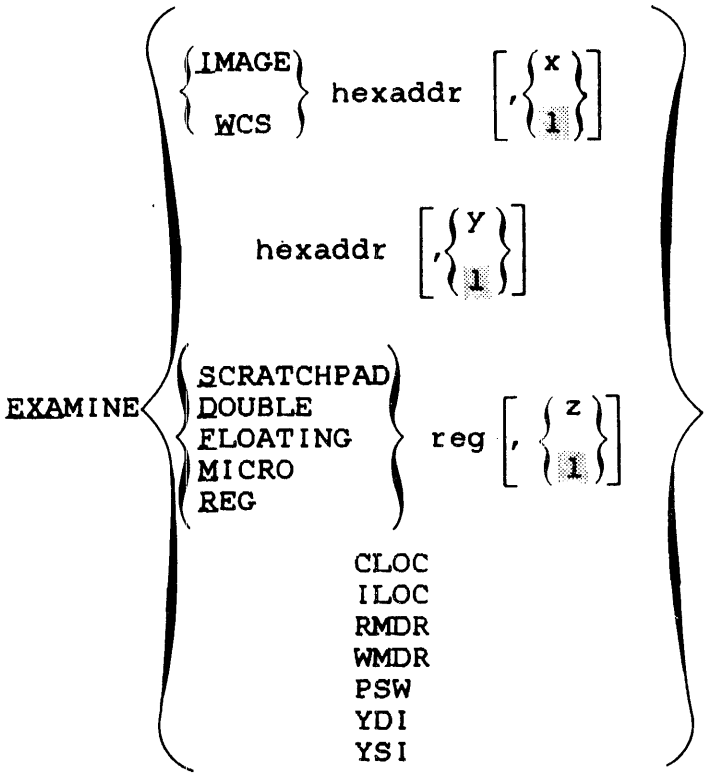
If a single or double precision floating point register is examined and that floating point option was not specified in the START command, one of these messages is displayed:

```
NO FLOATING POINT SUPPORT
NO DOUBLE PRECISION SUPPORT
```

7.2.2.2 Models 3240 and 3250 Processors EXAMINE Command

The EXAMINE command for the Models 3240 and 3250 processors version of the full support program differs slightly from that shown for the other version:

Format:



Parameters:

IMAGE	specifies that the WCS image buffer in main memory is to be examined. The contents of the image buffer are displayed in disassembled microcode format.
WCS	specifies that WCS memory is to be examined. The contents of WCS memory are displayed in disassembled microcode format.
hexaddr	is the 1- to 6-character hexadecimal address of the first cell to be examined.
x	is a decimal number from 1 to 16 specifying the number of fullwords to be examined.

y	is a decimal number from 1 to 16 specifying the number of halfwords to be examined.
ALTERNATE	specifies that the contents of the alternate register images be examined.
DOUBLE	specifies that the contents of the double precision floating point register images be examined.
FLOATING	specifies that the contents of the single precision floating point register images be examined.
MICRO	specifies that the contents of the microregister images be examined.
REG	specifies that the contents of the general register images be examined. The register set is determined by the PSW.
reg	is a decimal number from 0 to 15 specifying the first register to be examined.
z	is a decimal number from 1 to 16 specifying the number of registers to be examined.
CLOC	specifies that the contents of the CLOC register image are to be examined.
ILOC	specifies that the contents of the ILOC register image are to be examined.
PSW	specifies that the contents of the PSW register image are to be examined.
RMDR	specifies that the contents of the RMDR register image are to be examined.
WMDR	specifies that the contents of the WMDR register image are to be examined.
YDI	specifies that the contents of the YDI register image are to be examined.
YSI	specifies that the contents of the YSI register image are to be examined.

7.2.3 Cell Modification

After a cell is examined, the contents of the cell are open for modification. Only one cell can be modified at any one time. To modify the current open cell, use the MODIFY command.

Format:

MODIFY $x_1 [x_2 \ldots x_{16}]$

Parameter:

x	is a hexadecimal number.

The numbers specified replace the contents of the location last examined. The modified cell containing the new contents is then displayed. The address of the current open cell is incremented to the next sequential cell. The next sequential cell then becomes the current open cell; its contents are available for modification. The user can modify cells successively without opening each consecutive cell. Modifying a location in WCS memory does not modify the corresponding location in the WCS image buffer. The reverse is also true.

Example:

```
*EXA 120,2
    120: 4300
    122: 801E
*MO 90A2
    122: 90A2
*EXA 120,2
    120: 4300
    122: 90A2
```

7.2.4 Inserting and Removing Breakpoints in Writable Control Store (WCS) Memory

To aid in debugging the microcode routine, breakpoints stop the execution of a microcode routine located in WCS memory and transfer control back to the WCS support program. A maximum of eight breakpoints can be inserted at any one time.

When a breakpoint takes place, execution of the routine stops and control is transferred to the support program. To insert a breakpoint, the support program saves the fullword microinstruction at the specified location, inserts a link microinstruction, and branches to the register save microcode routine located in high memory of WCS. (This routine is loaded in high memory of WCS by the support program whenever a breakpoint is inserted or a GO WCS command is used.) In addition to saving the double precision, single precision, general, and scratchpad (alternate) registers, the register save microcode routine saves or transfers the following registers to their corresponding areas in the image buffer and returns control to the support program:

Models 3220 and 3230 processors

- PSW
- LOC
- MDR
- YDI
- YSI
- SR
- Microregisters

Models 3240 and 3250 processors

- PSW
- ILOC
- CLOC
- RMDR
- WMDR
- YDI
- YSI
- Microregisters

See the EXAMINE command for the definitions of these keywords.

When a breakpoint is encountered, execution of the microprogram is stopped and control is passed to WCSUPP. The user may then issue any of the commands available to WCSUPP. Execution of the microprogram may then be continued via the GO command.

NOTES

1. Use caution when setting a breakpoint at a register-to-register transfer instruction or a branch and link microinstruction. When the GO command is used to continue execution following a breakpoint, the instruction is executed from its location in high memory of the WCS.

2. The amount of high memory reserved by WCSUPP for the register save microcode routine is processor dependent. For Models 3220 and 3230 processors, the last 64 fullwords are reserved; for Models 3240 and 3250 processors, the last 46 fullwords are reserved. Do not place microcode in the WCS memory reserved by WCSUPP when using breakpoints.

3. A TRANSFER IMAGE command must be issued before breakpoints can be inserted.

4. For Models 3220 and 3230 processors, the contents of the link register (LR) are destroyed when a breakpoint is encountered.

7.2.4.1 Inserting Breakpoints

To insert a breakpoint in a microcode routine located in WCS, use the INSERT command. Inserting a breakpoint does not change the contents of the WCS image buffer.

Format:

INSERT wcsaddr $\left[,\left\{\begin{matrix} n \\ 1 \end{matrix}\right\}\right]$

Parameters:

wcsaddr	is the address of a microcode instruction in a routine located in WCS memory where the breakpoint is to be inserted.
n	is a decimal number specifying the number of times the instruction is encountered before the breakpoint actually has any effect. If n is not specified, 1 is the default value.

Example:

```
I 92C,13
```

The microinstruction located at $92C_{16}$ is encountered 12 times in normal sequence. A breakpoint is taken on the thirteenth encounter.

When a breakpoint is encountered, the following message is displayed on logical unit 3 (lu3):

```
BREAKPOINT HIT AT wcsaddr
```

where wcsaddr is the address of the instruction where the breakpoint was inserted.

7.2.4.2 Removing Breakpoints

To remove a breakpoint previously inserted in a microcode routine located in WCS memory, use the ZAP command. The ZAP command does not change the contents of the image buffer.

Format:

$$\text{ZAP} \left\{ \begin{matrix} \text{wcsaddr} \\ \textbf{all} \end{matrix} \right\}$$

Parameter:

wcsaddr	is the address of the breakpoint to be removed.

The breakpoint located at the specified WCS address is removed, and the microcode instruction at the specified WCS address is restored. If no WCS address is specified, all existing breakpoints are removed.

Example:

```
Z 92A
```

The breakpoint at location $92A_{16}$ is removed, and the microcode instruction located at $92A_{16}$ is restored.

7.2.5 Microprogram Execution

A microcode routine loaded in WCS memory may be executed by issuing the GO WCS command. To start a microcode routine, use the GO WCS command.

Format:

```
GO WCS wstaddr
```

Parameter:

wstaddr	is the address in WCS where execution begins.

The WCS starting address is checked to see if it is within the range of the WCS address limits. If the address is valid, the registers are restored, and control is transferred to the WCS starting address specified in the GO command. The current values of all registers (from the register image buffers) are also transferred to the corresponding registers.

Example:

```
G W 920
```

A TRANSFER IMAGE command must have been issued prior to using the GO command. If it was not, the following error message is displayed:

```
MICROCODE NOT YET TRANSFERRED
```

7.2.6 Resuming Execution After a Breakpoint

If a microcode routine is interrupted by a breakpoint, the microcode instruction located at the address specified by the breakpoint can be executed and microcode execution continued. To continue a microcode routine after a breakpoint is encountered, use the GO command:

Format:

```
GO
```

When the GO command is executed after a breakpoint takes place, the registers are restored, and the microcode instruction located at the address specified by the breakpoint is executed in high memory of WCS. If the microinstruction at the breakpoint was not a branch instruction, a branch is then taken to the microcode instruction following the instruction located at the breakpoint, and execution continues.

The breakpoint feature of the WCS support program can be used to debug another task that uses microcode routines; however, extreme caution should be used. When the task to be debugged enters a microcode routine and a breakpoint is taken, control is transferred to the support program, but the operating system is unaware of the change. Therefore, the support program uses the logical units that are assigned to the other task. In addition, the other task must have the memory address translator (MAT) disabled before the breakpoint is taken, or a memory fault will occur. It is recommended that the microcode routines be debugged separately before being used by other tasks.

7.2.7 Using the DUMP IMAGE Command

Use the DUMP IMAGE command to copy portions of the WCS image buffer to an output device in disassembled microcode format or common microassembler object format.

Format:

```
DUMP IMAGE [LOADER] staddr,endaddr,fd
```

Parameters:

LOADER	specifies that the data should be dumped in common microassembler object format.

staddr — is the starting address of the area in the WCS image buffer to be dumped.

endaddr — is the ending address of the area in the WCS image buffer to be dumped.

fd — is the file descriptor of the device or file that the dump is copied to or stored on.

The DUMP IMAGE command dumps to the specified file or device the area in the WCS image buffer specified by the starting and ending addresses. This dump is output in disassembled microcode format unless LOADER is specified. If LOADER is specified, the device must be a binary output device. This LOADER dump is output in common microassembler object format with the progname DUMP at the beginning of the output file for identification.

After the DUMP IMAGE command is executed, this message is printed or displayed on the specified device:

```
I DUMP FROM staddr TO endaddr
```

7.2.8 Establishing Writable Control Store (WCS) Microcode Routines

WCS microcode routines, when debugged and saved on a specified device and file, can be established as system default WCS. To establish debugged microcode object routines as system default WCS, use the ESTABLISH command.

Format:

```
ESTABLISH progname,fd
```

Parameters:

progname — is the name of the microcode object routine that is the label in the PROG statement.

fd — is the file descriptor of the device or file that the microcode object routine is currently stored on.

When the ESTABLISH command is executed, this sequence of operations occurs:

1. The WCS image buffer is cleared.

2. The microcode object routine stored on a specified device and file is loaded into the WCS image buffer.

3. The contents of the entire image buffer (2,048 fullwords) is transferred to WCS memory.

4. The WCS support program is placed in a trap wait state.

NOTE

WCSUPP differs from WCSAIDS in that the ESTABLISH command for WCSAIDS does not place the support program in a trap wait state.

The ESTABLISH command can be used only with microcode object routines that are assembled using MICROCAL, or dumped using the WCS support program. Since the contents of the entire image buffer are loaded into WCS memory as a result of the ESTABLISH command, the first 16 fullword locations of the image buffer should contain the appropriate addresses of all the microcode routines that it contains. If an error occurs during a load operation, an error message is displayed and the next user command is requested.

After the microcode routine is established as system microcode, this message is displayed on the system console:

```
SYSTEM DEFAULT WCS ESTABLISHED
```

The WCS support program is placed in a trap wait state until a power restoration trap occurs following a power fail/restore sequence.

When microcode is established as WCS system microcode with the ESTABLISH command, the support program will no longer accept command input.

7.2.9 Pausing the WCSUPP Task

The WCS support program task can be paused during execution to allow the user to do some intermediate processing. To pause the task, use the PAUSE command.

Format:

PAUSE

To continue execution of a task after the PAUSE command, the operator should use the operating system command CONTINUE.

7.2.10 Terminating a Task

Execution of the WCS support program can be terminated by using the END command.

Format:

END

7.2.11 Saving the Contents of the Writable Control Store (WCS) Image Buffer

When the WCS image buffer contains a fully debugged microcode routine, it can be saved on any auxiliary storage media. To save the contents of the WCS image buffer, use the SAVE command.

Format:

SAVE fd

Parameter:

fd	is the file descriptor of the device or file that the microcode image routine is to be stored on. If .ext is omitted, .WCS is the default.

When the SAVE command is executed, a loader information block (LIB) is built and transferred, followed by the contents of the WCS image buffer, to the device or the file specified. The specified device is checked for write and binary attributes. If the specified file is:

- contiguous, it must contain at least 33 records,
- indexed, it must contain a logical record length of 256 bytes, or

- nonexistent, it is allocated as a new contiguous file on the specified volume, or the system default volume if no volume name is present.

7.2.12 Retrieving a Writable Control Store (WCS) Image File

A WCS image file can be loaded into the WCS image buffer by using the GET command.

Format:

GET fd

Parameter:

fd	is the file descriptor of the device or file that the saved microcode image is stored on. If .ext is omitted, .WCS is the default.

7.2.13 Writable Control Store (WCS) Wait State

When the WCS support program is loaded and started, it is in the active state. When a WAIT or ESTABLISH command is executed, the support program enters the wait state, where no user interaction or program processing occurs. The support program will restore the data contained in the WCS image buffer into the WCS memory in the event of a power failure. Before going into the wait state, the WCS memory must contain a fully debugged microcode routine. To enter the wait state, use the WAIT command.

Format:

WAIT

When the WAIT command is executed, the following sequence of operations occurs:

1. The contents of the WCS memory is transferred into the WCS image buffer in main memory.
2. The WCS support program is placed into trap wait.

NOTE

WCSAIDS does not recognize the WAIT command.

After the contents of WCS memory are transferred to the WCS image buffer, the contents becomes the system default WCS image microcode, and the following message is displayed on the system console:

SYSTEM DEFAULT WCS ESTABLISHED

The microcode transferred to the buffer becomes established system microcode because it is eventually restored into WCS memory. The user can now alter the code in image buffer only through another executive task (e-task).

7.2.14 Restoring Writable Control Store (WCS) After a Power Fail

After a power failure and when the power is restored, the operating system restores the system status and displays the following message on the system console:

POWER RESTORE RESET PERIPHERALS AND ENTER GO

The user should reset all peripherals and issue a GO command. The power restoration trap is taken by the WCS support program since it is the highest priority task using WCS in the system. The contents of the WCS image buffer is then transferred back into WCS memory and this message is displayed:

WCS RESTORED

See Figure 7-1 for the states and logical flow of WCS initialization.

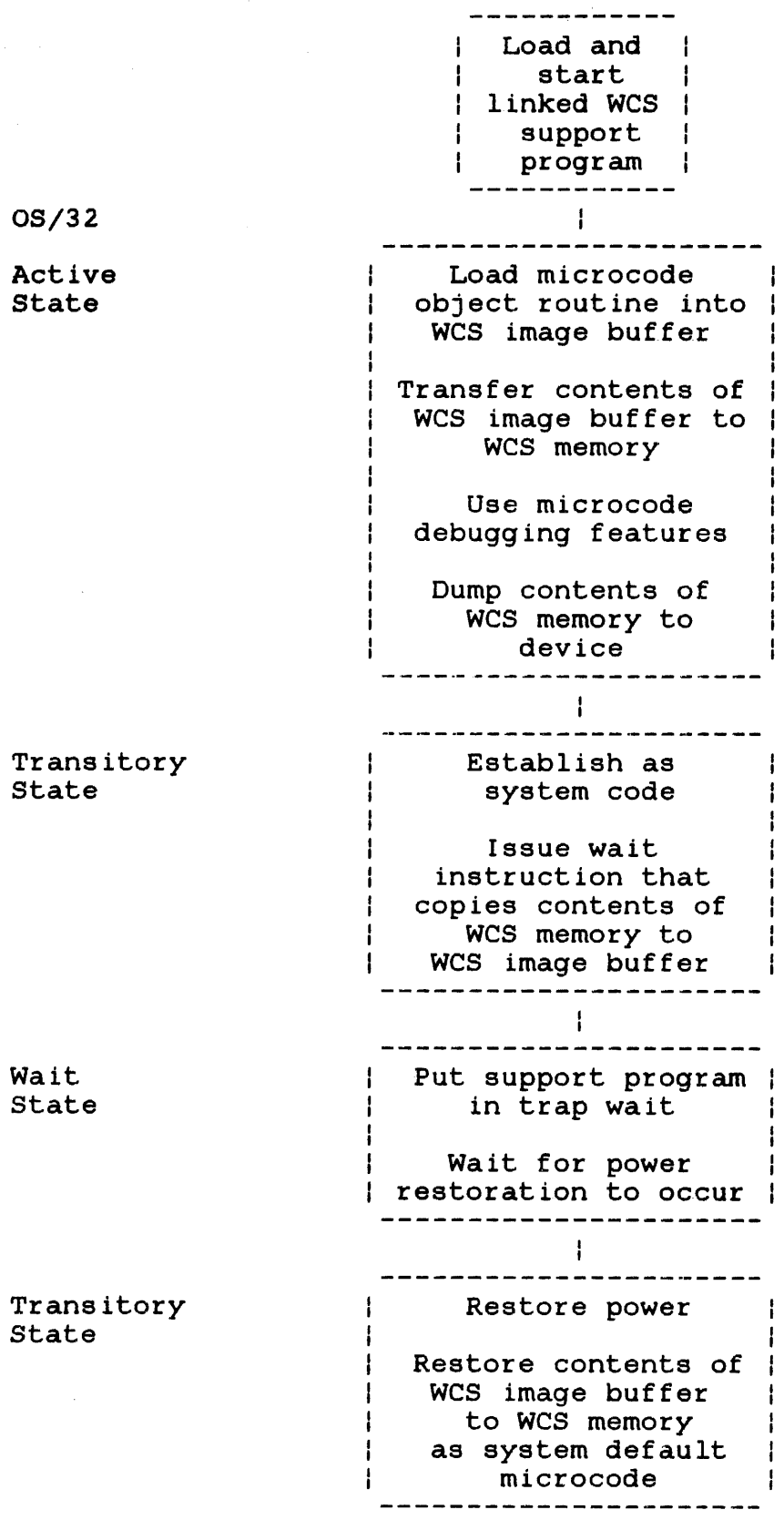


Figure 7-1 WCS Initialization

7.3 WCSLPFM

The WCSLPFM also provides the user with the ability to establish system default WCS. No debugging facility is provided by the WCSLPFM. Its main advantage lies in its much smaller memory requirements. WCSLPFM requires only 1kb of main memory for its operation.

The microcode to be established must be fully debugged. Instructions to build, load, and start the WCSLPFM were discussed in Section 3.3.5.

CHAPTER 8
TYPICAL APPLICATIONS

8.1 INTRODUCTION

This section contains sample programs that illustrate how special WCS functions are added to the system. All routines in this section use the enter control store (ECS) instruction rather than the privileged branch to control store (BDCS) instruction. The ECS instruction is assumed by the hardware to be an RI1 format instruction. When the user level instruction is read, two consecutive halfwords are read. The second halfword is placed in the MDR and can be used as a 16-bit immediate constant or memory address. Without hardware modification, the ECS instruction cannot assume the RR, RX2, RX3, or RI2 formats. Through the user defined mnemonic mechanism in CAL, however, the user can cause an ECS instruction to assemble as an RR or RI1 format instruction. See the Common Assembly Language/32 (CAL/32) Manual.

If the user chooses the RR format for the ECS instruction, the microcode routine entered must effectively decrement the location counter by 2 because the hardware will have assumed the RI1 format and will have incremented the location counter by 4.

Appendix H shows an example writable control store (WCS) microprogram for a Model 3250 processor.

8.2 FIND A IN B

This instruction set extension compares a string called A, containing a variable number of bytes, against a string called B, containing a greater number of bytes than A. When string A is found in string B, the starting address of the matching string in B is returned. The routine is presented with the start and end addresses of A and the start and end addresses of B. Since the routine may be lengthy, it is interruptible and restartable from its previous location in byte string B where the interrupt occurred. This is accomplished by modifying the given start address of B as the routine proceeds. Upon termination, the condition code field of the PSW indicates whether A was found in B. See Figure 8-1.

To interface this routine to the ECS mechanism, the symbolic register names used are broken up into microregisters and user level general registers. The nature of the ECS instruction that causes this routine to be entered is discussed.

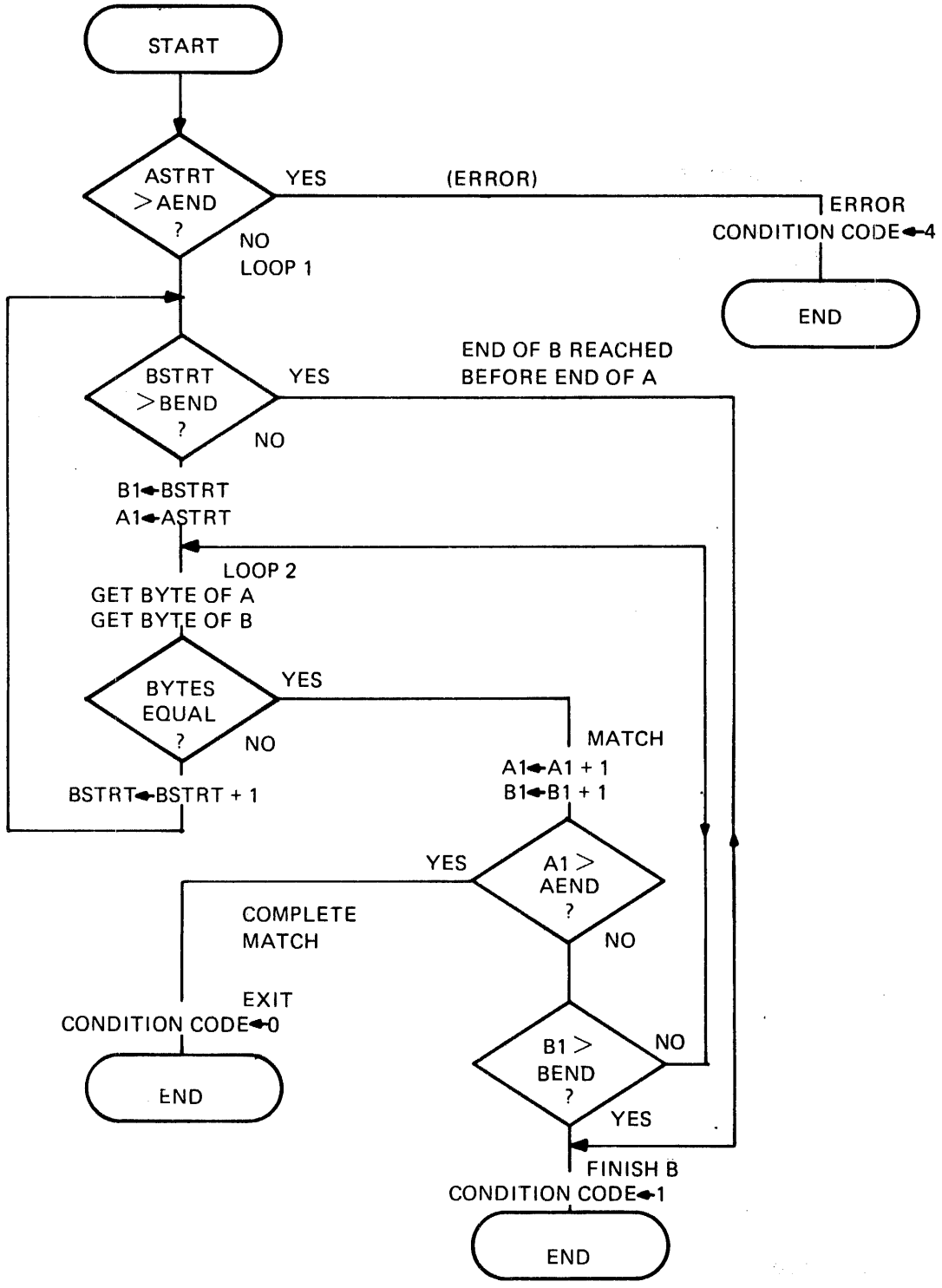


Figure 8-1 Flowchart for Finding String A in String B

Through the EQU operation in the CAL assembler, the following assignment can be made:

```
FIND  EQU  X'E92C'
```

The symbol FIND, if encountered as an operation code mnemonic, is assembled as an extended RR type instruction. See the Common Assembly Language/32 (CAL/32) Programming Reference Manual. The opcode is E9, an ECS instruction, and the R1 field is set to 2. At the assembly language level, to execute the Find A in B microroutine, write:

```
NAME  FIND  R2
```

The general register specified by R2 is the first four sequential general registers whose functions are:

R2	Start address of A
R2+1	End address of A
R2+2	Start address of B
R2+3	End address of B

Additional overhead is added to this routine to collect the four parameters. The user register containing the start address of B is modified by the microcode routine to make the routine restartable in case of an interrupt. When an interrupt occurs, LOC is pointing to the FIND ECS instruction. After the interrupt is handled, the FIND instruction is reexecuted. Because the start address of B was modified, execution resumes where it was interrupted in the routine.

An example of a microroutine, developed for a Model 3250 processor, is shown in Appendix H. The source program is assembled using MICROCAL. The program is given the name EXAMPLE by placing that symbol in the label field of the PROG statement. Although the PROG statement is not listed, the label of the PROG statement is shown in the message line on the first page.

```
EXAMPLE MICROCODE ROUTINE FOR APPENDIX H
PROG = EXAMPLE  ASSEMBLED BY MICROCAL II  (32-BIT)
```

The operand of the PROG statement, EXAMPLE MICROCODE ROUTINE FOR APPENDIX H, becomes the main header line on each page of the program listing.

The object of the EXAMPLE program is loaded into the WCS using the following WCS support program command sequence:

COMMAND	ACTION
hh:mm:ss>CLEAR	Clear WCS image
hh:mm:ss>LOAD EXAMPLE,PRTP:	Load object from a device (PTRP:)
hh:mm:ss>TRANSFER IMAGE 800,FFF	Transfer WCS image to WCS

The 16 ECS entry points, the first 16 words of the WCS, are set up in the example. ECS entry point 2 has a vector pointing to the FIND routine. The remaining entry points have vectors pointing to fixed control store (FCS) location 208_{16}, which is the start of the illegal instruction handler in the microcode. This arrangement results in:

OCCURRENCE OF ECS IN USER LEVEL CODE	ACTION BY WCS CODE
ECS 0,A(X2)	Initiate illegal instruction interrupt
ECS 1,A(X2)	Initiate illegal instruction interrupt
ECS 2,A(X2)	Evoke FIND microroutine
ECS 3,A(X2) . . . ECS F,A(X2)	Initiate illegal instruction interrupt

When the WCS is set up to support the FIND instruction, the user level program can be loaded and executed. Every occurrence of the FIND instruction in the user level program causes the FIND microprogram sequence in WCS to be evoked. The function is performed by the microcode, and control returns to the user level instruction immediately following the FIND.

Example:

```
FIND EQU   X'E92C'
     .
     .
     .
     LA    R7,SOURCE          BYTE STRING A IS AN
     LA    R8,SOURCE+7        EIGHT CHARACTER MNEMONIC
     LA    R9,TABLESRT        START ADDRESS OF TABLE
     LA    R10,TABLEND        END ADDRESS OF TABLE
     FIND  R7                 GO  FOR A MATCH
     BNE   NOMATCH            BRANCH IF NO FIND; REGISTER 9
     *                        CONTAINS ADDRESS OF MATCHING
     *                        ENTRY
     SI    R9,TABLE           SUBTRACT OUT START ADDRESS
     SRLS  R9,1               DIVIDE BY 2 FOR WORD INDEX
     L     R6,VECTOR(R9)      COLLECT SUBROUTINE ADDRESS
     BR    R6                 GO TO SUBROUTINE
```

8.3 FLOATING POINT SQUARE ROOT

Appendix H also shows a microcode routine, again for a Model 3250 processor, that calculates the square root of the floating point quantity contained in a single precision floating point register. The result replaces the original argument. The ECS entry point for square root has been filled in at control store location X'803'. At the user level source program that uses square root, the statement:

```
SQRT     EQU   X'E93C'
```

causes every occurrence of the mnemonic SQRT in the operation field of an instruction statement to assemble as an ECS instruction with an R1 field of three.

Example:

```
CODE    NAME      OPERATION     OPERAND       COMMENT

        SQRT        EQU         X'E93C'     ASSIGN MNEMONIC
          *                                 FOR SQUARE ROOT
                      .
                      .
                      .
2846                LER         4,6         COPY FLOATING POINT
          *                                 REGISTER 6 TO
          *                                 FLOATING POINT
          *                                 REGISTER 4.
E934                SQRT        4           FIND THE SQUARE ROOT
          *                                 OF THE CONTENTS OF
          *                                 FLOATING POINT
          *                                 REGISTER 4.
```

APPENDIX A
WCSLINK COMMAND SUMMARY

CLEAR

Clear writable control store (WCS) image buffer by initializing all fullwords with a branch to the illegal instruction interrupt handler.

END

End the task.

EXAMINE IMAGE nnn [,{m / i}]

Display the specified area of the WCS image buffer in main memory in dissassembled microcode format.

GET fd

Load the WCS image buffer with the contents of the specified file.

LOAD progname [,fd]

Load a microcode routine named progname from the specified file or device.

MODIFY nnnnnnnn

Replace the currently open cell (i.e., last displayed cell) with the data specified. The next logical cell becomes the currently open cell.

SAVE fd

Save the contents of the WCS image buffer preceded by the loader information block (LIB) on the specified file.

TARGET = nnnnxxx

Specify the model number of the processor on which the microcode is to be loaded. xxx applies only to the central processing unit (CPU) and auxiliary processing units (APUs) of the Model 3200MPS System.

APPENDIX B
WCSAIDS COMMAND SUMMARIES FOR SPECIFIC PROCESSORS

B.1 MODELS 3220 AND 3230 PROCESSORS, AND THE MODEL 3200MPS SYSTEM AUXILIARY PROCESSING UNIT (APU) VERSIONS

CLEAR

Clear writable control store (WCS) image buffer by initializing all fullwords with a branch to the illegal instruction interrupt handler.

DUMP IMAGE [LOADER] staddr,endaddr,fd

Copy or dump the area specified in the WCS image buffer to the specified file or device in disassembled format or, if LOADER is specified, in MICROCAL object format.

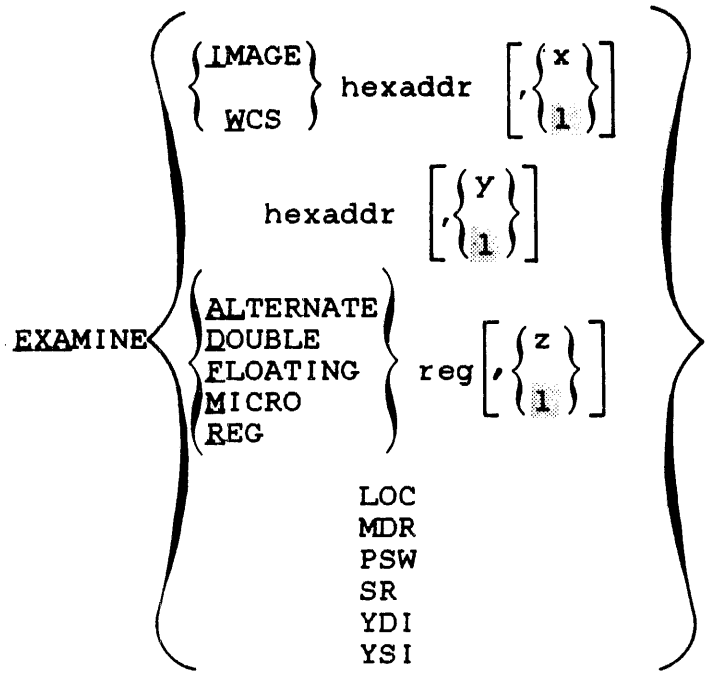
END

End the task.

ESTABLISH progname,fd

Perform the following functions in order.

1. Clear WCS image buffer.
2. Load the microcode object routine from the named file or device into the WCS image buffer.
3. Transfer the entire WCS image buffer to the WCS memory.
4. Place the WCS support program into trap wait state.



Display the contents of the specified memory location or the register image.

GET fd

Load the WCS image buffer with the contents of the specified file.

GO

Start the microcode execution at the last breakpoint encountered. The microcode instruction at the address specified by the breakpoint is executed.

GO WCS wstaddr

Start microcode execution at the specified WCS address.

INSERT wcsaddr [,n] [,1]

Insert a breakpoint at the specified address in the WCS resident microcode. The breakpoint is not taken until it has been encountered n number of times.

LOAD progname [,fd]

Load a microcode routine named progname from the specified file or device.

MODIFY $x_1 [x_2 \ldots x_{16}]$

Replace the currently open cell (i.e., last cell displayed) with the data specified. The next logical cell becomes the currently open cell.

PAUSE

Pause the task.

SAVE fd

Save the contents of the WCS image buffer, preceded by the loader information block (LIB), on the specified file.

START nnnn

Start the user microprogram that has been linked with WCSAIDS.

TARGET=nnnnxxx

Specify the model number of the processor on which the microcode is to be loaded. xxx applies only to the central processing unit (CPU) and APUs of the Model 3200MPS System.

TRANSFER { IMAGE / WCS } staddr,endaddr

Transfer the specified area in the specified memory to the corresponding locations in the unspecified memory.

ZAP { wcsaddr / all }

Remove a breakpoint at the specified WCS address in the WCS resident microcode.

B.2 MODELS 3240 AND 3250 PROCESSORS, AND THE MODEL 3200MPS SYSTEM CENTRAL PROCESSING UNIT (CPU) VERSIONS

CLEAR

Clear WCS image buffer by initializing all fullwords with a branch to the illegal instruction interrupt handler.

DUMP IMAGE [LOADER] staddr,endaddr,fd

Copy or dump the area specified in the WCS image buffer to the specified file or device in disassembled format or, if LOADER is specified, in MICROCAL object format.

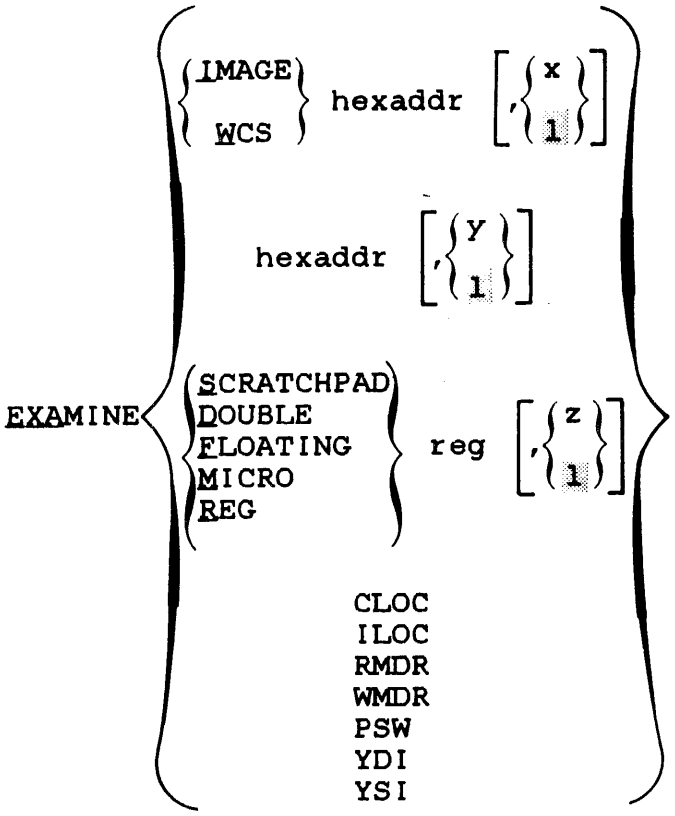
END

End the task.

ESTABLISH progname,fd

Perform the following functions in order.

1. Clear WCS image buffer.
2. Load the microcode object routine from the named file or device into the WCS image buffer.
3. Transfer the entire WCS image buffer to the WCS memory.
4. Place the WCS support program into trap wait state.



Display the contents of the specified memory location or register image.

GET fd

Load the WCS image buffer with the contents of the specified file.

GO

Start the microcode execution at the last breakpoint encountered. The microcode instruction at the address specified by the breakpoint is executed.

GO WCS wstaddr

Start microcode execution at the specified WCS address.

INSERT wcsaddr $\left[,\left\{\begin{matrix} n \\ 1 \end{matrix}\right\}\right]$

Insert a breakpoint at the specified address in the WCS resident microcode. The breakpoint is not taken until it has been encountered n number of times.

LOAD progname [,fd]

Load a microcode routine named progname from the specified file or device.

MODIFY $x_1\left[x_2 \ldots x_{16}\right]$

Replace the currently open cell (i.e., last displayed cell) with the data specified. The next logical cell becomes the currently open cell.

PAUSE

Pause the task.

SAVE fd

Save the contents of the WCS image buffer preceded by the LIB on the specifed file.

START nnnn

Start the user microprogram linked to WCSAIDS.

TARGET=nnnnxxx

Specify the model number of the processor on which the microcode is to be loaded. xxx applies only to the CPU and APUs of the Model 3200MPS System.

TRANSFER { IMAGE / WCS } staddr,endaddr

Transfer the specifed area in the specified memory to the corresponding locations in the unspecified memory.

ZAP { wcsaddr / all }

Remove a breakpoint at the specified WCS address in the WCS resident microcode.

APPENDIX C
LOADER AND POWER FAIL MONITOR (MPSLPFM) OPTION AND MESSAGE SUMMARY

SEND CLEAR apu-no$_1$ [,apu-no$_2$,...] [,{ON / OFF}]

Invalidates the writable control store (WCS) of a processor.

SEND ESTABLISH [{COMMAND=fd / parameter-list / IMAGE=fd}]

Establishes an association between a processor and an image file, or changes an existing association.

SEND RESTORE apu-no$_1$ [,apu-no$_2$,...] [,{ON / OFF}]

Initiates a load of a microcode image file into the WCS of the specified processors.

SEND STATUS

Obtains a list of microcode image files associated with each processor.

START [{COMMAND=fd / ,parameter-list / IMAGE=fd}]

Starts the MPSLPFM with the specification of individual microcode images for loading into the WCS of individual processors.

APPENDIX D
WCSUPP/WCSLPFM COMMAND SUMMARIES

D.1 MODELS 3220 AND 3230 PROCESSORS

CLEAR

Clear writable control store (WCS) image buffer by initializing all fullwords with a branch to the illegal instruction interrupt handler.

DUMP IMAGE [LOADER] staddr,endaddr,fd

Copy or dump the area specified in the WCS image buffer to the specified file or device in disassembled format or, if LOADER is specified, in MICROCAL object format.

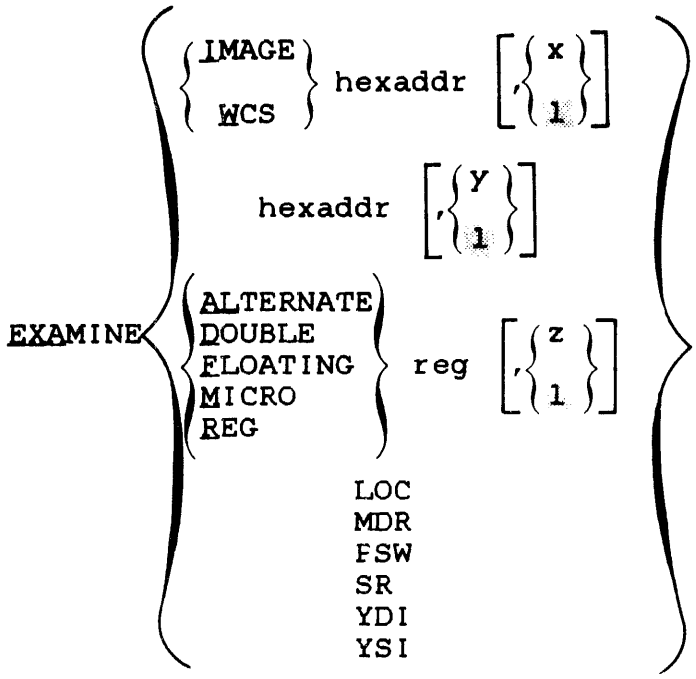
END

End the task.

ESTABLISH progname,fd

Perform the following functions in order.

1. Clear WCS image buffer.
2. Load the microcode object routine from the named file or device into the WCS image buffer.
3. Transfer the entire WCS image buffer to the WCS memory.
4. Place the WCS support program into trap wait state.



Display the contents of the specified memory location or register image.

GET fd

Load the WCS image buffer with the contents of the specified file.

GO

Start the microcode execution at the last breakpoint encountered. The microcode instruction at the address specified by the breakpoint is executed.

GO WCS wstaddr

Start microcode execution at the specified WCS address.

INSERT wcsaddr [,n] [,1]

Insert a breakpoint at the specified address in the WCS resident microcode. The breakpoint is not taken until it has been encountered n number of times.

LOAD progname[,fd]

Load a microcode routine named progname from the specified file or device.

MODIFY $x_1 [x_2 \ldots x_{16}]$

Replace the currently open cell (i.e., last cell displayed) with the data specified. The next logical cell becomes the currently open cell.

PAUSE

Pause the task.

SAVE fd

Save the contents of the WCS image buffer preceded by the loader information block (LIB) on the specified file.

TRANSFER { IMAGE / WCS } staddr,endaddr

Transfer the specified area in the specified memory to the corresponding locations in the unspecified memory.

WAIT

Establish system default WCS. Log SYSTEM DEFAULT WCS ESTABLISHED message to system console and put task in trap wait. Wait for the power restoration trap to occur. When it occurs, restore WCS and log WCS ESTABLISHED message to system console. Then place task in trap wait state.

ZAP { wcsaddr / all }

Remove a breakpoint at the specified WCS address in the WCS resident microcode.

D.2 MODELS 3240 AND 3250 PROCESSORS

CLEAR

Clear WCS image buffer by initializing all fullwords with a branch to the illegal instruction interrupt handler.

DUMP IMAGE st addr,end addr,fd

Copy or dump the area specified in the WCS image buffer to a device in disassembled format.

DUMP IMAGE[LOADER]st addr,end addr,fd

Dump the area specified in the WCS image buffer in common microassembler (MICROCAL) object format to the specified file or device.

END

End the task.

ESTABLISH progname,fd

Perform the following functions in order:

1. Clear WCS image buffer.
2. Load the microcode object routine from the named file or device into the WCS image buffer.
3. Transfer the entire WCS image buffer to the WCS memory.
4. Place the WCS support program into trap wait state.

EXAMINE hexaddr $\left[,\left\{\begin{matrix}Y\\1\end{matrix}\right\}\right]$

Display the contents of the specified number of main memory halfwords (default=1) starting from the specified hexadecimal address.

EXAMINE CLOC

Display the contents of the CLOC register image.

EXAMINE DOUBLE reg $\left[,\left\{\begin{matrix}Z\\1\end{matrix}\right\}\right]$

Display the contents of the specified number of double precision floating point register images (default=1) starting from the register number specified.

EXAMINE FLOATING reg [,{z / 1}]

Display the contents of the specified number of single precision floating point register images (default=1) starting from the register number specified.

EXAMINE ILOC

Display the contents of the ILOC register image.

EXAMINE IMAGE hexaddr [,{x / 1}]

Display the contents of the specified number of fullwords in the WCS image buffer in disassembled microcode format starting from the specified hexadecimal address.

EXAMINE MICRO reg [,{z / 1}]

Display the contents of the specified number of microregister images (default=1) starting from the register number specified.

EXAMINE PSW

Display the contents of the PSW register image.

EXAMINE REG reg [,{z / 1}]

Display the contents of the specified number of general register images (default=1) starting from the register number specified. The register set is determined by the PSW at the time of breakpoint in the microcode.

EXAMINE RMDR

Display the contents of the RMDR register image.

EXAMINE SCRATCHPAD reg [,{z / 1}]

Display the contents of the specified number of scratchpad register images (default=1) starting from the register number specified.

EXAMINE WCS hexaddr [,{x / 1}]

Display the contents of the specified number of fullwords in WCS memory in disassembled microcode format starting from the specified hexadecimal address.

EXAMINE WMDR

Display the contents of the WMDR register image.

EXAMINE YDI

Display the contents of YDI register image.

EXAMINE YSI

Display the contents of YSI register image.

GET fd

Load the WCS image buffer with the contents of the specified file.

GO

Start the microcode execution at the last breakpoint encountered. The microcode instruction at the address specified by the breakpoint is executed.

GO WCS wstaddr

Start microcode execution at the specified WCS address.

INSERT wcsaddr [,{n / 1}]

Insert a breakpoint at the specified address in the WCS resident microcode. The breakpoint is not taken until it has been encountered n numbers of times.

LOAD progname,fd

Load a microcode routine named progname from the specified file or device.

MODIFY hex data

Replace the currently open cell (i.e., last cell displayed) with the data specified. The next logical cell becomes the currently open cell.

PAUSE

Pause the task.

SAVE fd

Save the contents of the WCS image buffer, preceded by the LIB, on the specified file.

TRANSFER IMAGE staddr,endaddr

Transfer the specifed area in WCS memory to the corresponding locations in the WCS image buffer.

WAIT

Establish system default WCS. Log SYSTEM DEFAULT WCS ESTABLISHED message to system console and put task in trap wait. Wait for the power restoration trap to occur. When it occurs, restore WCS and log WCS ESTABLISHED message to system console. Then place task in trap wait state.

ZAP {wcsaddr / all}

Remove a breakpoint at the specified WCS address in the WCS resident microcode.

APPENDIX E
ERROR AND RESPONSE MESSAGES FROM WCSLINK, WCSAID, AND WCSUPP

ADDRESS ERROR

Microroutine being loaded is positioned at an address outside the WCS limits, or the writable control store (WCS) address specified in the command is outside the limits.

ALLOCATE ERROR

Error detected while allocating or checking attributes or file descriptor (fd) in SAVE command.

ALREADY A BREAKPOINT

The specified address is already a breakpoint.

ASSIGN ERROR

Assign error in supervisor call 7 (SVC 7). See the OS/32 Supervisor Call (SVC) Reference Manual.

BREAKPOINT HIT AT xxx

A breakpoint in microcode was encountered at xxx.

CHECKSUM ERROR, REQUEST TERMINATED

Checksum error detected in the last object record.

DEVICE/FILE HAS INCORRECT ATTRIBUTES

Device or file does not have binary attributes.

I DUMP FROM xxx TO xxx:

Header for WCS image words dump in disassembled format.

ILLEGAL COMMAND

Illegal or out of sequence command entered. See the individual command description.

ILLEGAL ENTRY INTO BREAKPOINT HANDLER

The register save microcode routine was entered, but not from a breakpoint.

ILLEGAL SEGMENT TYPE IN LIB

The segment type encoded in the loader information block (LIB) is not compatible with that required for the WCS image file.

INTERNAL BREAKPOINT ERROR

Data in the breakpoint directory is corrupt. The support program should be reloaded.

I/O ERROR xxxx

Input/output (I/O) error in SVC 1. See the OS/32 Programmer Reference Manual.

LOAD ERROR

Improper MICROCAL object format, or error found in object. Load correct object.

LU ERROR xx

Wrong device or fd, or I/O error from specified device while assigning a logical unit (lu) using SVC 7.

MICROCODE NOT YET TRANSFERRED

An attempt was made either to insert a breakpoint or to execute a GO command, but no TRANSFER IMAGE command was issued.

MICROCODE TRANSFERRED TO WCS

Indicates the successful transfer of microcode to WCS.

MULTIPLE DEFINED

Labels encountered while editing are defined more than once.

NO BREAKPOINTS

ZAP command used, but no breakpoints existed.

NO BREAKPOINTS HIT YET

A GO command without operands was issued, but no breakpoint was encountered, or the last breakpoint hit was removed.

NO DOUBLE PRECISION SUPPORT

An attempt was made to examine a double precision register, but START option DOUBLE was not specified.

NO FLOATING POINT SUPPORT

An attempt was made to examine a floating point register, but START option SINGLE was not specified.

NOT A BREAKPOINT

Breakpoint address specified in ZAP command is not a breakpoint.

SEQUENCE ERROR

Object record out of sequence. Reload object from the beginning.

SYNTAX ERROR

Illegal command syntax. See Appendix A and enter correct command.

SYSTEM DEFAULT WCS ESTABLISHED

During execution of the WAIT or ESTABLISH command, the program copied the WCS into the WCS image.

TARGET ERROR

The model number specified in the TARGET command is not acceptable.

THIS IMAGE WAS BUILT FOR nnnnxxx

Response from WCSLINK and WCSAIDS indicating the model number of the processor for which the image file is TARGETed.

THIS IMAGE IS NOT ACCEPTABLE ON THIS PROCESSOR

The target information encoded in the LIB of the image file indicates that the image file was built for the WCS of another processor model.

THIS TARGET IS NOT ACCEPTABLE ON THIS PROCESSOR

The model number specified in the TARGET command is incompatible with the model number for which the WCSAIDS is intended.

WARNING: THIS UTILITY SHOULD NOT BE RUN ON THIS PROCESSOR!

The WCSAIDS version is inappropriate for the processor on which it has been loaded.

TOO MANY BREAKPOINTS

A maximum of eight breakpoints can be inserted at any one time.

WCS IMAGE CLEARED FOR nnnnxxx

Response from WCSLINK and WCSAIDS to the CLEAR and ESTABLISH commands.

WCS RESTORED

After power up, the program restored the system default WCS to WCS memory.

WCS SUPPORT PROGRAM 03-xxx Fmm Rnn

Program name; nn is the revision level.

WCS SUPPORT PROGRAM-LOADER/POWER FAIL MONITOR 03-xxx Fmm Rnn

Program name; nn is the revision level.

APPENDIX F
ERROR AND RESPONSE MESSAGES FOR LOADER AND POWER FAIL MONITOR (MPSLPFM)

ILLEGAL OPTION <option>

An unrecognizable option has been encountered in the START command or in the option specified in the ESTABLISH, CLEAR, RESTORE, or STATUS message.

<association parameter>APU # TOO LARGE

The auxiliary processing unit (APU) number specified in the option is greater than the maximum number of APUs in the system.

OLD IMAGE FILE = fd NEW IMAGE FILE = fd FOR APU number

The image file is different from the already specified one for that APU. The new association is established.

IMPROPER ATTRIBUTES FOR THE FILE fd FOR APU number

The device on which the image file resides does not permit binary read, or is not a direct access device, or the record length of the file is not 256.

I/O ERROR ON COMMAND FILE - LU 5

An input/output (I/O) error has been encountered when reading an option from the command file. The LPFM would treat this error as an indication of end of parameter-list.

SVC7 ERROR - CANNOT ASIGN FILE fd

A supervisor call 7 (SVC 7) function (fetch attributes, assign, or close) fails with an error for the file 'fd'. The MPSLPFM continues execution to validate the rest of the parameter list.

NESTING OF "COMMAND=" OPTION IS NOT ALLOWED

A "COMMAND=" option is encountered in the command file; nesting of command files is not allowed.

"COMMAND=" or "IMAGE=" OPTION CANNOT BE MIXED WITH OTHER OPTIONS

When a COMMAND= or IMAGE= option is specified either in the START command or in the ESTABLISH message, any other option cannot be specified with it.

NO IMAGE FILES SPECIFIED FOR ANY PROCESSOR

A SEND STATUS is received by the MPSLPFM, but none of the APUs or the central processing unit (CPU) is associated with any image file.

MESSAGE MUST BE "ESTABLISH", "CLEAR", "RESTORE", OR "STATUS"

A message other than ESTABLISH, CLEAR, RESTORE, or STATUS is not recognized by the MPSLPFM.

APU number IS NOT MARKED OFF

An APU specified is not marked off; the writable control store (WCS) of that APU is not loaded.

CANNOT WRITE SUCCESSFULLY TO THE WCS OF APU number.

After transferring the microcode to the WCS, it is read back from the WCS. The latter does not match the microcode actually transferred - this is an indication of hardware malfunction.

WCS OF APU number IS LOADED SUCCESSFULLY

The microcode from the image file for the APU has been transferred successfully to the WCS of that APU.

I/O ERROR xxxx * LU l FOR APU number

While reading a record from the image file logical unit (lul) for the APU, a nonzero status code (xxxx) is obtained. The MPSLPFM will clear the WCS of that APU.

LOAD ERROR - IMPROPER IMAGE FILE FOR APU number

The image file specified for the APU does not have a proper header record in the loader information block. MPSLPFM will clear the WCS of that APU. An image file must be built using the WCSLINK.

SYNTAX ERROR IN RESTORE/CLEAR - IGNORED

The syntax of the message is not correct. The function requested in the message is not performed.

NO IMAGE FILE IS SPECIFIED FOR THE APU number

A request to RESTORE the WCS of an APU has been made, but no image file has been specified for that APU. The WCS of that APU is cleared.

APU number IS NOT EQUIPPED WITH WCS

A request to load the WCS of an APU has been encountered through a RESTORE or CLEAR message or through the START command or the ESTABLISH message, but the APU is not equipped with a WCS.

WCS OF APU number IS CLEARED.

The APU number is not associated with an image file, but the APU is equipped with WCS, or a CLEAR message is sent to the MPSLPFM for that APU or an error was encountered in reading the image file for that APU. The WCS of the APU is filled with illegal instruction traps.

APU number IS MARKED ON

or

APU number IS MARKED OFF

After loading the WCS of an APU, the APU is marked ON or OFF. This is also done even if errors were encountered during the process of loading the WCS.

SVC 13 ERROR STATUS xxxx FUNCTION nn OPTION mm FOR APU number.

The SVC 13 function 'nn' option 'mm' for the APU returned a nonzero status code of 'xxxx'.

MISMATCH IN MODEL NUMBERS FOR APU number, IMAGE FILE fd FOR model-number.

The image file is not TARGETed for an APU (3200APU). Nevertheless, the microcode is transferred to the WCS of the indicated APU.

SIZE OF MICROCODE IN fd IS LARGER THAN WCS SIZE (size) OF APU number

The microcode in the image file does not fit in the WCS of the APU.

SUCCESSIVE POWER FAILS, CANNOT PROCEED; RESTART MPSLPFM

A power fail occurred when processing an earlier power fail interrupt. MPSLPFM cannot proceed under these circumstances. It should be cancelled, removed, and then reloaded and restarted. If an attempt is made to continue MPSLPFM, it would terminate with an end of task code of 1.

POWER FAIL WHEN PROCESSING A MESSAGE, MESSAGE IGNORED

When processing a message sent to MPSLPFM, a power fail occurs. The message processing is discontinued and MPSLPFM proceeds with power fail/restore.

TRANSFER OF MICROCODE TERMINATED

Due to a reason already explained by an earlier message, the transfer of microcode to the WCS of that processor could not be continued.

SIZE OF THE WCS OF APU "number" COULD NOT BE DETERMINED

The APU is equipped with WCS, but a read/write to the WCS location X'800' could not be performed. This is an indication of a hardware malfunction. MPSLPFM will assume that this APU is not equipped with WCS.

THAT APU IS ASSUMED NOT TO HAVE WCS

Due to a reason already indicated by an earlier message, the APU is assumed not to be equipped with WCS.

COULD NOT START APU number FOR TASK EXECUTION

After obtaining the control privileges of an APU, an attempt to start the APU for task execution failed.

COULD NOT MARK THAT APU AS REQUESTED

Due to a reason indicated by an earlier message, an attempt to mark that APU as requested (ON/OFF) was unsuccessful.

COULD NOT RELEASE CONTROL RIGHTS OF THAT APU

After transferring the microcode to the WCS of the APU, an attempt to release the control rights of that failed.

SVC 6 ERROR IN ASSIGN LPU, STATUS = xxxx

An attempt to assign a logical processing unit (LPU) for MPSLPFM failed with status code xxxx.

NO FREE LPU IS AVAILABLE

There was no logical processing unit (LPU) available for MPSLPFM to use for loading the WCS of the APU.

RESCHEDULE TO APU number FAILED

MPSLPFM was unable to reschedule itself to the specified APU.

APPENDIX G
EXAMPLE OF DUMP IMAGE

```
  A DUMP FROM   950 TO   970 :
950 : 006E00F0          L       SR,LENGTH
951 : 00146040          S       LOC,SR,LOC
952 : 006200B0          L       MR2,YSI
953 : 000AA001          AI      YDI,MR2,'01'
954 : 00600080          L       MRO,YD
955 : 000AA003          AI      YDI,MR2,'03'
956 : 00610080          L       MR1,YD
957 : 000AA002          AI      YDI,MR2,'02'
958 : 001C5000          S       NULL,YS,MRO
959 : 0E4095C0          BF      C,'95C'
95A : 00EF8004          LI      FLR,'04',JAM
95B : 0E009790          B       '979'
95C : 001C4010          S       NULL,YD,MR1
95D : 0E409600          BF      C,'960'
95E : 00EF8001          LI      FLR,'01',JAM
95F : 0E009790          B       '979'
960 : 00630080          L       MR3,YD
961 : 00620090          L       MR2,YS
962 : 00660020          L       MAR,MR2
963 : BE0296B0          LINK    '96B',DR2
964 : 04690080          L       ARSYS,ARSYD
965 : 00660030          L       MAR,MR3
966 : BE0296B0          LINK    '96B',DR2
967 : 044C5080          X       NULL,ARSYS,ARSYD
968 : 0E789740          BF      C+V+G+L,'974'
969 : 0008C001          AI      YD,YD,'01'
96A : 0E0495C0          B       '95C',IVJE
96B : 006C0E60          SRL     NULL,MAR
96C : 0C409710          BT      C,'971'
96D : 046884FF          EXBI    ARSYD,'FF'
96E : 04284050          N       ARSYD,ARSYD,MDR
96F : 04680480          EXB     ARSYD,ARSYD
970 : 0E010000          RETN
```

```
  EXAMPLE MICROCODE ROUTINES FOR  APPENDIX H                    PAGE    1   07:58:43      07/03/79

PROG= EXAMPLE      ASSEMBLED BY MICROCAL II  (32-BIT)

                                    1           SCRAT
                                    2           CROSS


  0000                              4           ORG   '800'
                                    5   *
                                    6   *  THE ECS ENTRY POINT VECTORS FOLLOW
                                    7   *
  0800   177C 8200                 8           BALD  '208'(WMDR)        ECS ENTRY POINT  -0-
  0801   177C 8200                 9           BALD  '208'(WMDR)        ECS ENTRY POINT  -1-
  0802   13FE 5400                10           BALA  FIND(NULL)         ECS ENTRY POINT  -2-
  0803   17FE 5C00                11           BALD  FSQR(NULL)         ECS ENTRY POINT  -3-
  0804   177C 8200                12           BALD  '208'(WMDR)        ECS ENTRY POINT  -4-
  0805   177C 8200                13           BALD  '208'(WMDR)        ECS ENTRY POINT  -5-
  0806   177C 8200                14           BALD  '208'(WMDR)        ECS ENTRY POINT  -6-
  0807   177C 8200                15           BALD  '208'(WMDR)        ECS ENTRY POINT  -7-
  0808   177C 8200                16           BALD  '208'(WMDR)        ECS ENTRY POINT  -8-
  0809   177C 8200                17           BALD  '208'(WMDR)        ECS ENTRY POINT  -9-
  080A   177C 8200                18           BALD  '208'(WMDR)        ECS ENTRY POINT  -A-
  080B   177C 8200                19           BALD  '208'(WMDR)        ECS ENTRY POINT  -B-
  080C   177C 8200                20           BALD  '208'(WMDR)        ECS ENTRY POINT  -C-
  080D   177C 8200                21           BALD  '208'(WMDR)        ECS ENTRY POINT  -D-
  080E   177C 8200                22           BALD  '208'(WMDR)        ECS ENTRY POINT  -E-
  080F   177C 8200                23           BALD  '208'(WMDR)        ECS ENTRY POINT  -F-
```

FIND A IN B

```
                                25  *
                                26  *
                                27  *  MATCH A VARIABLE LENGTH BYTE STRING "A" AGAINST A
                                28  *  LONGER BYTE STRING "B". USE AN ECS INSTRUCTION
                                29  *  TO CALL THE ROUTINE.
                                30  *
                                31  *        (R2)   = START ADDRESS OF A
                                32  *        (R2+1) = END ADDRESS OF A
                                33  *        (R2+2) = START ADDRESS OF B
                                34  *        (R2+3) = END ADDRESS OF B


                                36  *  MICRO REGISTER ASSIGNMENTS
                                37  *
0000 0018                       38  ASTRT    EQU   '18'              YS
0000 0010                       39  AEND     EQU   '10'              MR0
0000 0019                       40  BSTRT    EQU   '19'              YD
0000 0011                       41  BEND     EQU   '11'              MR1
0000 0012                       42  A1       EQU   '12'              MR2
0000 0013                       43  B1       EQU   '13'              MR3
0000 0014                       44  BYTEA    EQU   '14'              MR4
0000 0015                       45  BYTEB    EQU   '15'              MR5


0810                            47           ORG   '950'


0950   2A5F 1E80                49  FIND     L     A1,YSI            COLLECT R2 FIELD
0951   33D2 1001                50           AI    YDI,A1,1          POINT TO R2+1
0952   2A1F 1C80                51           L     AEND,YD           COLLECT END ADDRESS OF A
                                52  *                                START ADRS OF A IS (YS)
0953   33D2 1003                53           AI    YDI,A1,3          POINT TO R2+3
0954   2A3F 1C80                54           L     BEND,YD           COLLECT END ADRS OF B
0955   33D2 1002                55           AI    YDI,A1,2          START ADRS OF B IS (YD)
```

FIND A IN B

```
                        56  *
                        57  *
                        58  *
0956    23F0 0C5A       59          SX      NULL,AEND,ASTRT,LOOP1,C COMPARE ASTRT=AEND
                        60  *
                        61  *                                  TRANSFER IF END ADRS EQUAL TO
                        62  *                                  OR GREATER THAN START (NO CARRY)
                        63  *
                        64  *                                  FALL THROUGH IF START ADDRESS
                        65  *                                  OF A GREATER THAN END ADDRESS
0957    33BD 5FF0       66          NI      PSW,PSW,'FF0'       CLEAR CONDITION CODE
0958    33BD 7004       67          OI      PSW,PSW,'004'       THEN SET V FLAG
0959    13FA 5AC0       68          BAL     EXIT1(NULL)
                        69  *
                        70  *
095A    23F1 0CDE       71  LOOP1   SX      NULL,BEND,BSTRT,LOOP1A,C COMPARE BSTRT=BEND
                        72  *
                        73  *                                  TRANSFER IF BEND EQUAL TO OR
                        74  *                                  GREATER THAN BSTRT.
                        75  *                                  IF FALL THROUGH, REACHED END
095B    33BD 5FF0       76  FINISHB  NI      PSW,PSW,'FF0'       OF B BEFORE A MATCH WAS MADE
095C    33BD 7001       77          OI      PSW,PSW,'001'       SET L FLAG IN CONDITION CODE
095D    13FA 5AC0       78          BAL     EXIT1(NULL)         AND EXIT
                        79  *
095E    2A7F 1C80       80  LOOP1A  L       B1,BSTRT            MOVE START ADDRESS
095F    2A5F 1C00       81          L       A1,ASTRT            INTO WORK REGISTERS
                        82  *
0960    2B9F 190B       83  LOOP2   L       MAR,A1,DR1          FETCH BYTE FROM STRING A
0961    2A9F 1D80       84          L       BYTEA,RMDR          COPY BYTE TO BYTE A
0962    2B9F 1988       85          L       MAR,B1,DR1          FETCH BYTE FROM STRING B
0963    2A8F 1D80       86          L       BYTEB,RMDR          COPY BYTE TO BYTEB
                        87  *
0964    2BF4 6A80       88          X       NULL,BYTEA,BYTEB    COMPARE THE TWO BYTES
0965    13E2 59C0       89          BALZ    MATCH(NULL)         BRANCH IF EQUAL
                        90  *                                  NO MATCH, INCREMENT BSTRT (YD)
0966    233F 3C9A       91          AINCX BSTRT,NULL,BSTRT,LOOP1 AND LOOP
```

FIND A IN B

```
                          92  *
                          93  *
0967   3252 1001          94  MATCH    AI     A1,A1,1              INCREMENT WORK ADDRESSES
0968   3273 1001          95           AI     B1,B1,1
0969   23F0 096E          96           SX     NULL,AEND,A1,MATCH1,C COMPARE PRESENT ADDRESS
                          97  *                                    OF A TO AEND. TRANSFER IF
                          98  *                                    NOT DONE, ELSE FALL THRU
096A   33BD 5FF0          99           NI     PSW,PSW,'FF0'        COMPLETE MATCH, CLEAR
                         100  *                                    CONDITION CODE AND EXIT
096B   325F 1002         101  EXIT1    LI     A1,2
096C   2B52 1000         102           A      CLOC,A1,ILOC         INCREMENT LOC BY TWO
096D   2BFF 1F92         103           L      NULL,NULL,IRD        FETCH & EXECUTE NEXT
                         104  *                                    USER INSTRUCTION.
                         105  *
096E   23F1 09E0         106  MATCH1   SX     NULL,BEND,B1,LOOP2,C COMPARE PRESENT ADDRESS
                         107  *                                    OF B TO BEND. TRANSFER IF
096F   13FA 56C0         108           BAL    FINISHB(NULL)        NOT DONE, ELSE GO TO FINISHB.
```

FLOATING POINT SQUARE ROOT

```
0000 0970                    110  FSQR     EQU   *
0970   CA9F 1C00              111           RRE   MR4,YS              CHECK THE SIGN
0971   2A9F 1A00              112           L     MR4,MR4             OF THE ARGUMENT.
0972   17E2 50C0              113           BALNZ SQRT.ST(NULL)
0973   323F 1002              114           LI    MR1,2               INCREMENT LOC
0974   2B51 1D00              115           A     CLOC,MR1,ILOC       X=0.0  GO HOME
0975   CBF8 2C00              116           LE    YS,YS               SET CONDITION CODE.
0976   CBFF 0FB2              117           RCC   NULL,NULL,E,IRD     SET CONDITION CODE.
0977   3674 59A2              118  SQRT.ST  NI    MR3,MR4,DSQRT.SP,I  (MR3)=EX000000
0978   3694 599B              119           NI    MR4,MR4,FRACT,I     00123456
0979   3294 9004              120           SLLI  MR4,MR4,4           GET F IN SCALE 16**7
                              121  *
                              122  *GET STARTING VALUE Y0*=Y0*4*16**7=4*16**7(A+.5*F-B/(C+F)).
                              123  *
097A   2AF4 1A00              124           A     MR7,MR4,MR4         2*F
097B   36F7 199D              125           AI    MR7,MR7,A.DSQRT,I   2*F+A
097C   36D4 199F              126           AI    MR6,MR4,C.DSQRT,I   C+F
097D   361F 199E              127           LI    MR0,B.DSQRT,I       LOAD B
097E   2A3F 1F80              128           L     MR1,NULL            PREPARE TO DIVIDE
097F   2A11 FB00              129           D     MR0,MR1,MR6         B/(C+F)
0980   2AF7 0880              130           S     MR7,MR7,MR1         (MR7)=STARTING VALUE
                              131  *
                              132  *NOW MAKE 2 ITERATIONS YK=.5*(Y+F/Y).FIRST:
                              133  *
0981   2A1F 1A00              134           L     MR0,MR4             LOAD F:
0982   2A3F 1F80              135           L     MR1,NULL            .
0983   2A11 FB80              136           D     MR0,MR1,MR7         F/Y0
0984   2AF7 1880              137           A     MR7,MR7,MR1         Y0+F/Y0
0985   32F7 8001              138           SRLI  MR7,MR7,1           .5*(Y0+F/Y0)=Y1
                              139  *
                              140  *SECOND  ITERATION:
                              141  *
0986   2ABF 1F80              142           L     MR5,NULL            PREPARE F
0987   2A95 FB80              143           D     MR4,MR5,MR7         F/Y1
0988   2AF7 1A80              144           A     MR7,MR7,MR5         8*16**7*Y2
0989   32F7 1040              145           AI    MR7,MR7,'040'       ROUND IT
```

FLOATING POINT SQUARE ROOT

```
                          146  *
                          147  *NOW COMPUTE SQRT(16**EX).IF EX=2E THEN Y2 IS SHIFTED 1BIT LEFT AND
                          148  *EXPONENT=E-1.OTHERWISE Y2 IS SHIFTED 1BIT RIGHT
                          149  *
098A   37F3 599C          150           NI     NULL,MR3,ONE.LG,I    IF EXPONENT ODD OR EVEN
098B   13E2 6440          151           BALZ   SQRT.EV(NULL)
                          152  *
098C   3673 099C          153  SQRT.ODD SI     MR3,MR3,ONE.LG,I     DO SO
098D   3273 8001          154           SRLI   MR3,MR3,1            32+E
098E   32F7 8001          155           SRLI   MR7,MR7,1            Y2*4*16**7
098F   3673 19A0          156           AI     MR3,MR3,DSQRT.S7,I   ADD 39*16**6
0990   13FA 6580          157           BAL    SQRT.NR(NULL)        GO TO NORMALIZE Y2
                          158  *
0991   3273 8001          159  SQRT.EV  SRLI   MR3,MR3,1            32+E1
0992   3673 19A1          160           AI     MR3,MR3,DSQRT.S6,I   ADD 38*16**6
0993   32F7 9001          161           SLLI   MR7,MR7,1            Y2*16**8
0994   17F2 6580          162           BALNC  SQRT.NR(NULL)        SKIP IF NO CARRY
0995   3273 1001          163           AI     MR3,MR3,1            TRANSFER CARRY BACK
                          164  *
0996   CBFF 8980          165  SQRT.NR  LW     NULL,MR3             NORMALIZE (MR3,MR7)
0997   CBF8 2B80          166           LE     YS,MR7
0998   323F 1002          167           LI     MR1,2                SET THE RETURN ADDRESS.
0999   2B51 1D00          168           A      CLOC,MR1,ILOC
099A   CBFF 0FB2          169           RCC    NULL,NULL,E,IRD      SET CC AND LEAVE.-
                          170  *
                          171  * CONSTANTS
                          172  *
099B   00FF FFFF          173  FRACT    DC     '00FFFFFF'
099C   0100 0000          174  ONE.LG   DC     '01000000'
099D   2529 8CC1          175  A.DSQRT  DC     '25298CC1'           .580661*4*16**7
099E   0058 897F          176  B.DSQRT  DC     '0058897F'           .086462*4*16**6
099F   02CD C982          177  C.DSQRT  DC     '02CDC982'           .175241*16**7
09A0   2700 0000          178  DSQRT.S7 DC     '27000000'           39*16**6
09A1   2600 0000          179  DSQRT.S6 DC     '26000000'           38*16**6
09A2   7F00 0000          180  DSQRT.SP DC     '7F000000'
09A3                      181           END
```

FLOATING POINT SQUARE ROOT

ASSEMBLED BY MICROCAL II (32BIT) NO ASSEMBLY ERRORS

A.DSQRT	0000 099D	125										
A1	0000 0012	49	50	53	55	81	83	94	94	96	101	102
AEND	0000 0010	51	59	96								
ASTRT	0000 0018	59	81									
B.DSQRT	0000 099E	127										
B1	0000 0013	80	85	95	95	106						
BEND	0000 0011	54	71	106								
BSTRT	0000 0019	71	80	91	91							
BYTEA	0000 0014	84	88									
BYTEB	0000 0015	86	88									
C.DSQRT	0000 099F	126										
DSQRT.S6	0000 09A1	160										
DSQRT.S7	0000 09A0	156										
DSQRT.SP	0000 09A2	118										
EXIT1	0000 096B	68	78									
FIND	0000 0950	10										
FINISHB	0000 095B	108										
FRACT	0000 099B	119										
FSQR	0000 0970	11										
LOOP1	0000 095A	59	91									
LOOP1A	0000 095E	71										
LOOP2	0000 0960	106										
MATCH	0000 0967	89										
MATCH1	0000 096E	96										
ONE.LG	0000 099C	150	153									
SQRT.EV	0000 0991	151										
SQRT.NR	0000 0996	157	162									
SQRT.ODD	0000 098C											
SQRT.ST	0000 0977	113										

INDEX

PERKIN-ELMER
Computer Systems Division

D O C U M E N T A T I O N C H A N G E N O T I C E

The purpose of this documentation change notice (DCN) is to provide a quick and efficient way of making technical changes to software manuals before they are formally updated or revised.

The manual affected by these changes is:

48-096 F01 R00 PERKIN-ELMER SERIES 3200 WRITABLE CONTROL STORE (WCS) SUPPORT PROGRAMS REFERENCE MANUAL

- Page 3-6

  In the description of the Link command sequence shown for building MPSLPFM as a task, the Link OPTION command should be removed, and the following note should be added after the Link sequence:

  NOTE

  The following Link options are embedded in the MPSLPFM object file:

```
DTASK
RESIDENT
NROLL
PRIORITY=(11,11)
ACPRIVILEGE
CONTROL
APMAPPING
APCONTROL
WORK=(X2000,X2000)
```

- Page 6-6

  In the example at the bottom of the page, the following:

  "Establish associations for all APUs and mark them ON:"

should be changed to:

"Establish associations for all APUs and mark their queues ON:"

- Page 6-7

In the first example, the following:

"Reestablish the associations of APU numbers 3 and 5, and mark them ON:"

should be changed to:

"Reestablish the associations of APU numbers 3 and 5, and mark their queues ON:"

- Page 6-8

In the second paragraph on page 6-8, the following sentences:

"If the option ON is specified in the above messages, the APUs for which this option is specified is marked ON after its WCS has been loaded. Otherwise, the APU is left marked OFF. The option ON(OFF) specified in the message applies to all the APUs appearing in the message."

should be changed to:

"If the option ON/OFF is specified in the above messages, the APU queue to which the APU is assigned is marked ON/OFF after its WCS has been loaded. Otherwise, the APU queue state is preserved. The option ON/OFF specified in the message applies to all the APUs appearing in the message.

- Page 6-8

The sentence in the third paragraph of section 6.4 that reads:

"However, the APUs should be marked off before sending the message to the MPSLPFM."

should be changed to:

"The APUs should be enabled before sending the message to the MPSLPFM."

- Page F-3

The SVC13 message (second from the bottom of the page) should be changed to read:

"SVC13 ERROR STATUS X'xx' FUNCTION X'nn' OPTION X'mm' FOR APU number.

The SVC13 function 'nn' option 'mm' for the APU returned a nonzero status code of 'xx'; all numbers are hexadecimal."

- Page F-5

The following messages and explanations should be added to Appendix F :

NO MEMORY TO KEEP CPU WCS FOR AUTO POWER FAIL RESTART

When loading a WCS image into the CPU, MPSLPFM detected that there was not enough dynamic memory available to store that image internally.

The program is normally linked with enough memory to store a CPU WCS image of 2K words, so that on power fail restart there is no need to access files containing WCS. This allows the WCS to be loaded before the rest of the tasks are restarted by the operating system.

COULD NOT SET QUEUE STATUS

After transferring the microcode to the WCS of the APU, an attempt to set or restore the status of the applicable APU queue failed.

PERKIN-ELMER

PUBLICATION COMMENT FORM

We try to make our publications easy to understand and free of errors. Our users are an integral source of information for improving future revisions. Please use this postage paid form to send us comments, corrections, suggestions, *etc.*

1. Publication number _____

2. Title of publication _____

3. Describe, providing page numbers, any technical errors you found. Attach additional sheet if neccessary.

_____

_____

_____

4. Was the publication easy to understand? If no, why not?

_____

5. Were illustrations adequate? _____

_____

_____

6. What additions or deletions would you suggest? _____

_____

_____

7. Other comments: _____

_____

_____

From _____ Date _____

Position/Title _____

Company _____

Address _____

_____

_____

STAPLE

STAPLE

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 22 OCEANPORT, N.J.

POSTAGE WILL BE PAID BY ADDRESSEE

PERKIN-ELMER

Data Systems Group
106 Apple Street
Tinton Falls, NJ 07724

ATTN:
TECHNICAL SYSTEMS PUBLICATIONS DEPT.

FOLD

FOLD

STAPLE

STAPLE

6433-1