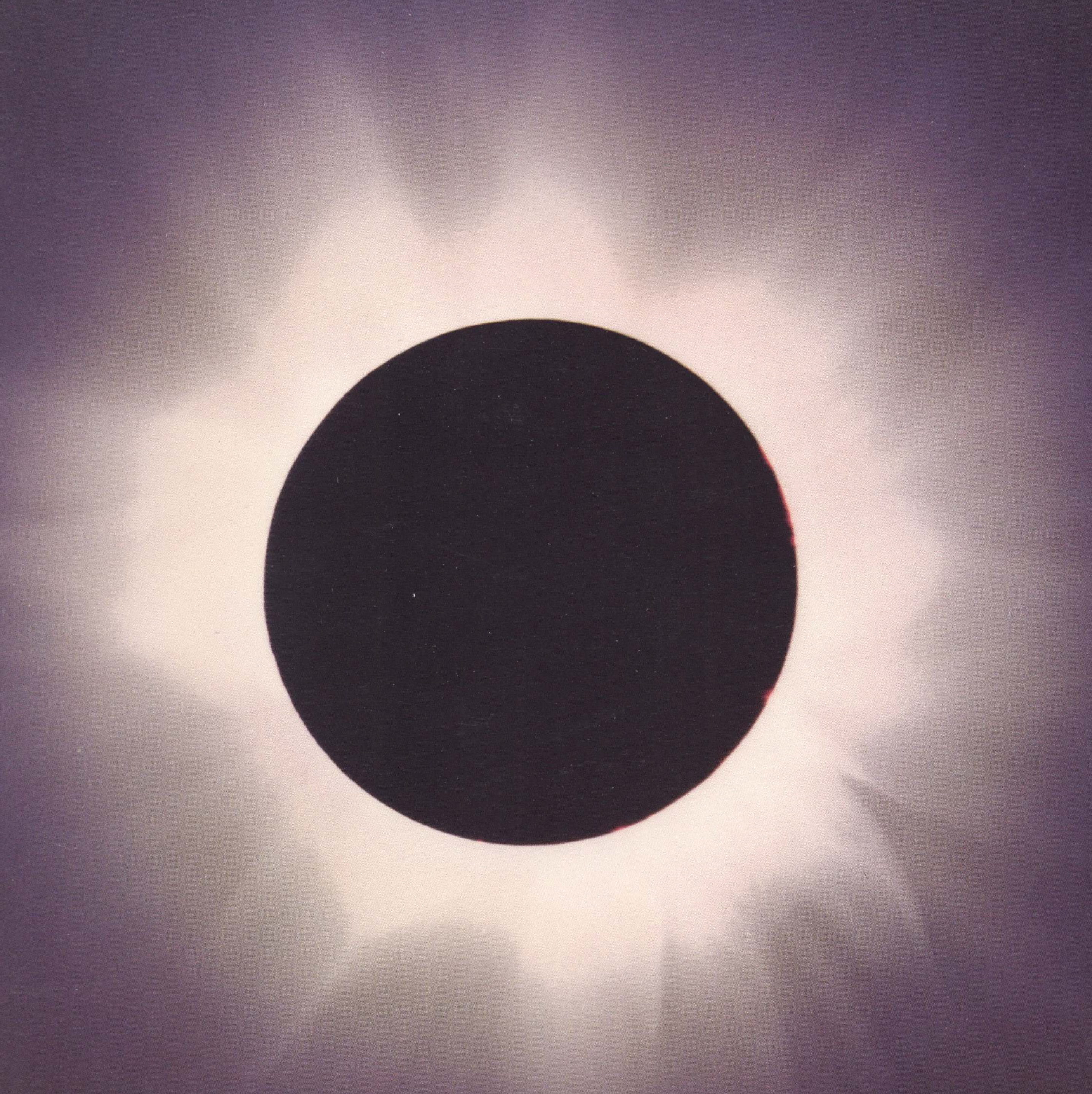
ECLIPSE MV/4000™ System



Functional Characteristics

Data General

ECLIPSE MV/4000™ System Functional Characteristics

014-000736-00

NOTICE

DGC reserves the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases consult DGC to determine whether any such changes have been made.

THE TERMS AND CONDITIONS GOVERNING THE SALE OF DGC HARDWARE PRODUCTS AND THE LICENSING OF DGC SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN DGC AND ITS CUSTOMERS. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS DOCUMENT INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING CAPACITY, RESPONSE-TIME PERFORMANCE, SUITABILITY FOR USE OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY DGC FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY OF DGC WHATSOEVER.

ECLIPSE MV/4000^{TM} System
Functional Characteristics
014-000736

Revision History:

Original Release - December 1982

Ordering No. 014-000736

Printed in the United States of America
Revision 00, December 1982

Preface

This manual addresses the assembly language programmers familiar with the *Principles of Operation, 32-Bit ECLIPSE® Systems* manual. For ease of use, the manual maps by chapters to the *Principles of Operation, 32-Bit ECLIPSE® Systems* manual.

The Organization of This Manual

The contents of each chapter and appendix of this manual are as follows:

Chapter 1, *Technical Summary*, explains the system components and functions that are available on the ECLIPSE MV/4000™ computer.

Chapter 2, *Fixed-Point Instruction Summary*, summarizes fixed-point formats and instructions.

Chapter 3, *Floating-Point Instruction Summary*, summarizes floating-point formats and instructions.

Chapter 4, *Stack Management Instruction Summary*, summarizes the wide stack instructions.

Chapter 5, *Program Flow Management*, explains program flow, interrupt handling, and fault handling.

Chapter 6, *Queue Management Instruction Summary*, summarizes the queue instructions.

Chapter 7, *Device Management*, explains the MV/4000 I/O devices and applicable instructions.

Chapter 8, *Memory and System Management*, presents the MV/4000 privileged instructions and related information for the operating system designer.

Chapter 9, *C/350 Programming*, explains ECLIPSE C/350 programming compatibility.

Appendix A, *Instruction Summary*, lists the unique MV/4000 instruction set alphabetically.

Appendix B, *Instruction Execution Times*, presents the typical execution time for each MV/4000 instruction.

Appendix C, *Register Fields*, presents tabular data for the various programmer-accessible registers.

Appendix D, *Reserved Memory Locations and Context Block Format*, lists the reserved memory locations for page zero, and shows the format for the context block.

Appendix E, *Standard I/O Device Codes*, lists standard Data General device codes.

Appendix F, *Fault Codes*, is a tabulation of the contents of Accumulator 1 for protection and nonprotection faults.

Appendix G, *Load Control Store Instruction*, presents the operation and format for this instruction.

Related Manuals

Other manuals useful in conjunction with the MV/4000 computer system are as follows:

Principles of Operation, 32-Bit ECLIPSE® Systems, Programmer's Reference Series (DGC No. 014-000704)

ECLIPSE MV/4000^{TM}*, Product Summary* (DGC No. 014-000708)

Intelligent Asynchronous Controller, Programmer's Reference Series (DGC No. 014-000703)

ECLIPSE® MV/Family Instruction Reference Booklet (DGC No. 014-000702)

Data General Communications Subsystems, Product Summary Series (DGC No. 014-000635)

Programmer's Reference Manual — Peripherals (DGC No. 015-000021)

Learning to Use AOS/VS (DGC No. 069-000031)

AOS/VS Macroassembler Reference Manual (DGC No. 093-000242)

AOS/VS Programmer's Manual (DGC No. 093-000241)

Conventions and Abbreviations

This manual uses the following conventions and abbreviations:

[] The square brackets indicate an optional argument. Omit the square brackets when you include an optional argument with an Assembler statement.

UPPERCASE and/or **Bold** Uppercase or bold characters indicate a literal argument in an Assembler statement. When you include a literal argument with an Assembler statement, use the exact form.

lowercase and/or *Italic* Lowercase or italic characters indicate a variable argument in an Assembler statement. When you include the argument with an Assembler statement, substitute a literal value for the variable argument.

ac The ac abbreviation indicates a fixed-point accumulator.

acs The acs abbreviation indicates a source fixed-point accumulator.

acd The acd abbreviation indicates a destination fixed-point accumulator.

fac The fac abbreviation indicates a floating-point accumulator.

facs The facs abbreviation indicates a source floating-point accumulator.

facd The facd abbreviation indicates a destination floating-point accumulator.

Table of Contents

1 Technical Summary

System Overview 1-1
Central Processing Unit 1-2
Instruction Processor 1-2
Arithmetic Processor 1-3
Address Translator 1-3
Memory System 1-3
I/O System 1-4
I/O Transfers 1-4
Communications Controllers 1-4
Universal Power Supply Controller 1-5
System Control Program 1-5
C/350 Compatibility 1-6
Registers 1-6
Initialization 1-6
Power Up 1-6
System Microcode Loads 1-7
IORST Instruction 1-7

2 Fixed-Point Instruction Summary

Fixed-Point Data Formats 2-1
Fixed-Point Instructions 2-2
Processor Status Register 2-8
Decimal/Byte Operations 2-9

3 Floating-Point Instruction Summary

Floating-Point Data Formats 3-1
Floating-Point Instructions 3-2
Floating-Point Status Register 3-5

4 Stack Management Instruction Summary

5 Program Flow Management

Program Counter 5-1
Address Space 5-2
Interrupts 5-2

Interrupt Sequence 5-2
Interrupting an Instruction 5-4
Program Flow Instructions 5-9
Fault Handling 5-11
Privileged Faults 5-11
Nonprivileged Faults 5-11

6 Queue Management Instruction Summary

7 Device Management

General I/O Instructions 7-1
Central Processor 7-3
Device Flags 7-3
CPU Instructions 7-4
Programmable Interval Timer 7-7
Device Flags 7-8
PIT Instructions 7-8
Real-Time Clock 7-9
Device Flags 7-10
RTC Instructions 7-10
Primary Asynchronous Line Input/Output 7-11
Device Flags 7-12
TTI/TTO Instructions 7-12
System Control Program 7-13
Device Flags 7-13
SCP Instructions 7-14
Data Channel/Burst Multiplexor Channel 7-19
DCH/BMC Maps 7-19
DCH/BMC Registers 7-20
DCH/BMC Map Instructions 7-23
Universal Power Supply Controller 7-26
Device Flags 7-26
UPSC Instructions 7-26

8 Memory and System Management

Address Translator 8-1
Referenced and Modified Bits 8-3
Protection Validation 8-3
Memory/System Management Instructions 8-4
Privileged Faults 8-4
Page Faults 8-4
Address Protection Faults 8-6
Reserved Memory 8-6

9 C/350 Programming

Registers 9-1
Instruction Compatibility 9-2
Program Flow 9-6
Fault Handling 9-6
Reserved Memory 9-6
CPU Identification 9-6

A Instruction Summary

B Instruction Execution Times

C Register Fields

Program Counter C-1
Processor Status Register C-2
Floating-Point Status Register C-3
Segment Base Registers C-4
DCH/BMC Status Registers C-5
CPU Identification C-6
  LCPID and ECLID Instructions C-6
  NCLID Instruction C-7

D Reserved Memory Locations and Context Block Format

Reserved Memory Locations D-1
  Page Zero Locations for Segment 0 D-1
  Page Zero Locations for Segments 1 through 7 D-3
Context Block Format D-4

E Standard I/O Device Codes

F Fault Codes

Protection Faults F-1
Page Faults F-2
Stack Faults F-2
Decimal/ASCII Faults F-3

G Load Control Store Instruction

Microcode File Format G-2
Microcode Block Format G-4
LCS Implementation G-4
Microcode Blocks G-5
Error Return G-8
Kernel Functionality G-10

H Programming Considerations

Current Page of Execution H-1
Double-Word Alignment H-1

Illustrations

Figure	Caption	Page
1-1	The ECLIPSE/4000 system	1-2
5.1	Interrupt sequence	5-3
5.2	Noninterruptible instruction interrupt sequence	5-5
5.3	Restartable instruction interrupt sequence	5-6
5.4	Resumable instruction interrupt sequence	5-8
7.1	DCH/BMC registers	7-21
8.1	Page fault sequence	8-5
G.1	General formats for microcode files	G-3
G.2	General form for microcode blocks	G-5

Tables

Table	Caption	Page
2.1	Fixed-point precision conversion	2-2
2.2	Fixed-point data movement instructions	2-2
2.3	Fixed-point addition instructions	2-3
2.4	Fixed-point subtraction instructions	2-3
2.5	Fixed-point multiplication instructions	2-4
2.6	Fixed-point division instructions	2-4
2.7	Initializing carry instructions	2-4
2.8	Fixed-point skip on condition instructions	2-5
2.9	Fixed-point increment or decrement word and skip instructions	2-6
2.10	Logical instructions	2-6
2.11	Logical shift instructions	2-7
2.12	Fixed-point logical skip instructions	2-7
2.13	PSR manipulation instructions	2-8
2.14	Fixed-point byte movement instructions	2-9
2.15	Fixed-point to floating-point conversion and store instructions	2-9
2.16	Load effective word and byte address instructions	2-9
2.17	Edit subprogram instructions	2-10
2.18	BCD arithmetic instructions	2-10
2.19	Hex shift instructions	2-10
3.1	Floating-point addition instructions	3-2
3.2	Floating-point subtraction instructions	3-2
3.3	Floating-point multiplication instructions	3-2
3.4	Floating-point division instructions	3-3
3.5	Floating-point skip on condition instructions	3-3
3.6	Floating-point binary conversion instructions	3-4
3.7	Floating-point decimal conversion instructions	3-4
3.8	Floating-point data movement instructions	3-4
3.9	FPSR instructions	3-5

Table	Caption	Page
4.1	Wide stack register instructions	4-1
4.2	Wide stack double-word access instructions	4-2
4.3	Wide stack return block instructions	4-2
4.4	Multiword wide stack instructions	4-3
5.1	Restartable or resumable instructions	5-4
5.2	State of PSR bits 2 and 3	5-7
5.3	Execute accumulator instruction	5-9
5.4	Jump instructions	5-9
5.5	Skip instructions	5-9
5.6	Subroutine instructions	5-10
5.7	Segment transfer instructions	5-10
5.8	Sequence of subroutine instructions	5-10
6.1	Queue instructions	6-1
7.1	General I/O instructions	7-2
7.2	Device flags for general devices	7-2
7.3	Device flags for skip instructions	7-2
7.4	I/O instructions for CPU	7-4
7.5	I/O instructions for PIT	7-8
7.6	I/O instructions for RTC	7-10
7.7	I/O instructions for TTI and TTO	7-12
7.8	SCP instructions	7-14
7.9	Device map registers 0000-7777	7-20
7.10	DCH/BMC map instructions	7-23
7.11	I/O instructions for UPSC	7-26
7.12	UPSC fault codes	7-31
8.1	Memory/system management instructions	8-4
9.1	C/350 fixed-point computing instructions	9-3
9.2	C/350 floating-point computing instructions	9-4
9.3	C/350 program float management instructions	9-5
9.4	C/350 stack management instructions	9-5
D.1	Page zero locations for segment 0	D-2
D.2	Page zero locations for segments 1 through 7	D-3
D.3	Context block format	D-4
E.1	Standard I/O device codes	E-1
F.1	Protection fault codes	F-1
F.2	Page fault codes	F-2
F-3	Stack fault codes	F-2
F-4	Decimal/ASCII faults	F-3

Chapter 1
Technical Summary

The ECLIPSE MV/4000™ computer system is a general purpose 32-bit data processing system that supports the complete 32-bit instruction set as presented in the *Principles of Operation, 32-Bit ECLIPSE® Systems* manual. In addition, the ECLIPSE MV/4000 computer system retains substantial hardware and software compatibility with 16-bit ECLIPSE systems. (However, kernel 16-bit operating system instructions (e.g., SYC, VCT, and LMP) are not supported.)

The MV/4000 system operates in the manner described in the *Principles of Operation, 32-Bit ECLIPSE® Systems,* manual.

This chapter describes the physical MV/4000 system, and initial processor conditions.

System Overview

The physical MV/4000 system (see Figure 1.1) incorporates four main systems:

- The *central processing unit,* which consists of: the instruction processor for decoding and executing instructions; the arithmetic processor for manipulation of data; and the address translator for logical to physical address translation.
- The *memory system,* which consists of: a memory controller and up to four memory modules of 0.5 Mbyte to 2 Mbytes each.
- The *input/output system,* which consists of: an integrated burst multiplexor channel/data channel/and programmed I/O controller; and a complement of standard Data General peripherals.
- The *system control program,* which is a micro-coded soft system console that performs diagnostic and operator-controlled functions.

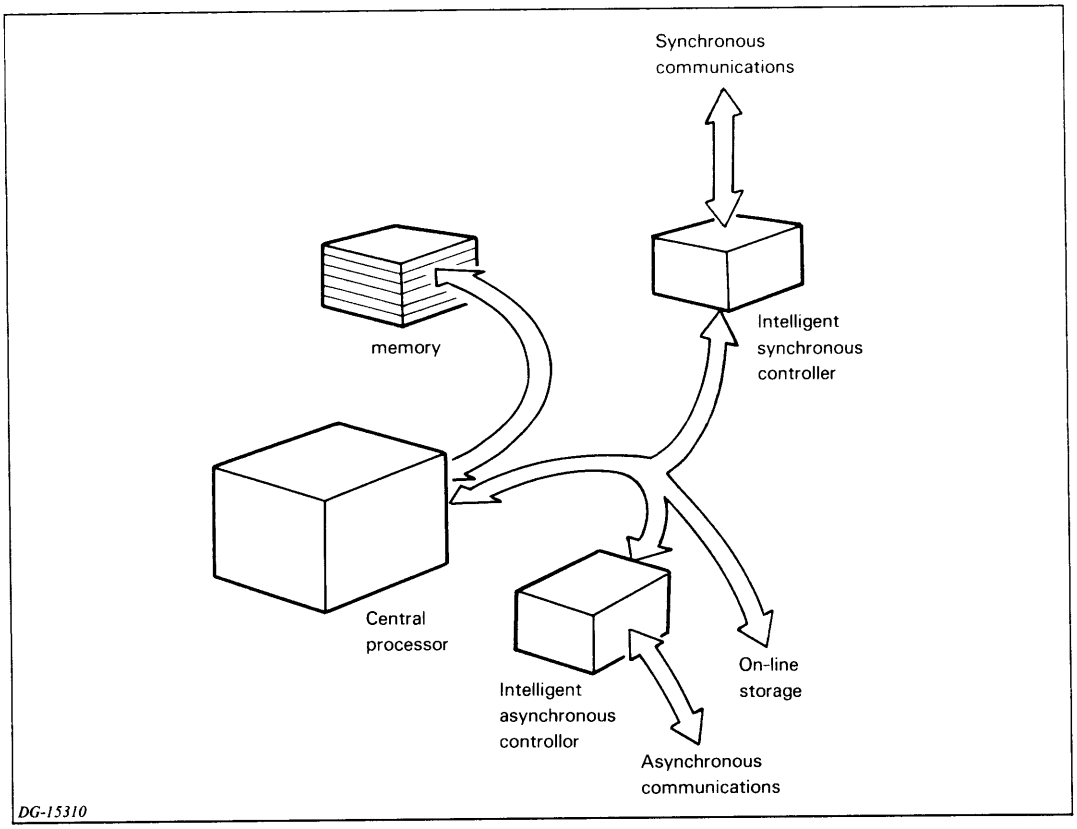

Synchronous communications
memory
Intelligent synchronous controller
Central processor
On-line storage
Intelligent asynchronous controllor
Asynchronous communications
DG-15310

Figure 1.1 The ECLIPSE MV/4000 system

Central Processing Unit

The central processing unit (CPU) of the MV/4000 system consists of a pipelined instruction processor, a high-speed arithmetic processor, and an address translator.

Instruction Processor

The instruction processor decodes instructions for execution. The instruction processor executes instructions in four steps:

1. It fetches an instruction from memory.
2. It parses the instruction opcode to obtain the starting address of the microcode routine, and collects operand information.
3. It sets aside the parsed information to wait for execution (while it parses a new instruction).
4. It initiates the microinstruction execution.

This four-stage sequence allows four instructions to be in the pipeline at any one time (one 16-bit instruction per step).

Arithmetic Processor

The arithmetic processor manipulates floating-point numbers, fixed-point quantities, and addresses.

The MV/4000 system contains four 32-bit fixed-point accumulators. The ECLIPSE C/350 16-bit fixed-point accumulators correspond to bits 16 through 31 of the MV/4000 system accumulators.

The program counter (PC) is 31 bits wide. Bits 1 through 3 specify the current segment of execution, and bits 4 through 31 specify an address in the segment.

Four floating-point accumulators, each 64 bits wide, contain the sign, the exponent, and the mantissa of any single- or double-precision floating-point operand. These four registers are identical to the C/350 floating-point registers. The MV/4000 system floating-point status register (FPSR) is 64 bits wide.

Four 32-bit registers govern the MV/4000 wide stack: the wide stack pointer (WSP), the wide frame pointer (WFP), the wide stack limit (WSL), and the wide stack base (WSB). Maintaining the stack in hardware speeds up stack management operations.

Address Translator

The MV/4000 computer has 4 Gbytes of logical memory and from 0.5 Mbyte to 8 Mbytes of physical memory. Because the logical address space is so much larger than the physical address space, the MV/4000 computer uses a *demand-page* system whereby it can store units of logical memory called *pages* on disk until needed by a process. When a process refers to a page (2 Kbytes) on disk, it moves the page to physical memory for manipulation. In addition to the page-swapping mechanism, this system also contains an address translator to convert the logical address into a physical address in memory.

The address translator also controls two memory management bits for each page: the *modified bit,* and the *referenced bit.* The operating system uses these bits during *page faults.*

The address translator performs all hardware checks required by the protection system. These checks include access validation, page validation, ring crossing validation, and others. If any of the checks fails, the address translator initiates a protection fault to the operating system. For more information about the types of protection checks, refer to the *Principles of Operation, 32-Bit ECLIPSE® Systems* manual.

Memory System

The MV/4000 memory system can support up to four dynamic random access memory (RAM) modules of up to 2 Mbytes each. Each 2 Mbyte memory module contains 512K double words, where each double word is 4 bytes long.

The 1-Mbyte and 2-Mbyte memory modules consist of two independent planes, each containing 0.5 Mbyte or 1 Mbyte of double words. Each plane contains every other double word. For instance with a 2-Mbyte memory module, plane 0 contains the double words 0-1, 4-5; plane 1 contains the double words 2-3, 6-7; and so on. This arrangement allows memory operations to consecutive double words to overlap.

The MV/4000 computer transfers data at a rate of 13.3 Mbytes/s.

The ECLIPSE MV/4000 memory system provides for error detection and correction with every double word read from memory or accessed during a memory refresh operation. The memory system detects memory errors with the error checking and correction (ERCC) logic when reading data from memory. (The memory controller calculates and appends seven ERCC bits to each double word it sends to a memory module.) Each time the controller reads a double word from a memory module, it checks the ERCC code. If it detects an error, it corrects the single-bit error before transmitting the data through the CPU or the I/O port. The system control program can log all ERCC errors.

When the memory controller performs the refresh operations (required by the dynamic random access memory (RAM) modules), the memory controller also checks for memory errors. (This operation is called *sniffing.*) Sniffing verifies all memory locations, correcting a single-bit error in memory even if that memory location is not being used by a program. This prevents an unused area of memory from collecting single-bit errors, and also prevents intermittent single-bit errors from becoming uncorrectable multiple-bit errors. The system control program can log all sniffing errors.

I/O System

The MV/4000 I/O system is electrically compatible and program compatible with the ECLIPSE C/350. This means that the MV/4000 computer supports the full family of standard Data General peripherals with high-speed burst multiplexor channel (BMC) I/O, data channel I/O (DCH), and programmed I/O (PIO).

I/O Transfers

Both the BMC and the data channel transfer data to and from the system memory directly, using the data path resources of the MV/4000 CPU.

> NOTE: *Since the MV/4000 performs the BMC (and DCH) operations in the MV/4000 microcode, CPU operations must halt while the BMC (or DCH) transfers data.*

- The BMC transfers blocks of data to and from memory at a rate of up to 5.0 Mbytes/s on output and up to 5.0 Mbytes/s on input.
- The data channel operates at rates up to 1.25 Mbytes/s on output and 2.5 Mbytes/s on input.

The programmed I/O system operates with a process transferring words or parts of words between the accumulators and I/O devices. These transfers are instrumental in setting up the parameters of the transfers for the higher speed channels. The MV/4000 computer executes most C/350 programmed I/O instructions exactly as the ECLIPSE C/350.

> NOTE: *The MV/4000 computer processes the I/O instructions for device codes 3 and 5 like other external devices (and not as internal ECLIPSE C/350 devices).*

Communications Controllers

Two processors control the asynchronous and synchronous communications. The intelligent asynchronous controller (IAC) handles asynchronous communications and the intelligent synchronous controller (ISC) handles synchronous communications. (The ISC can handle either asynchronous or synchronous communications.)

Intelligent Asynchronous Controller

The IAC is a 16-bit processor connected to the MV/4000 computer, which features standard facilities such as accumulators, stacks, a standard I/O bus, an ECLIPSE C/350 instruction subset, a priority interrupt system, etc. The MV/4000 computer with four IACs supports up to 64 asynchronous lines.

Communication between the MV/4000 central processor and the IAC is necessary to coordinate their operation. For example, the IAC must be able to signal the host when it has completed a task or needs more information. The IAC memory allocation and protection unit and two groups of special instructions provide the MV/4000 computer and the IAC with the necessary ability to communicate.

For further information, refer to the *Intelligent Asynchronous Controller* manual.

Intelligent Synchronous Controller

The ISC is a 16-bit processor connected to the MV/4000 computer, which features standard facilities such as accumulators, stacks, a standard I/O bus, an ECLIPSE instruction subset, a priority interrupt system, etc. The ISC handles two asynchronous or synchronous communications lines.

Communications between the MV/4000 central processor and the ISC is necessary to coordinate their operation. For example, the ISC must be able to signal the host when it has completed a task or needs more information. The ISC memory allocation and protection unit and two groups of special instructions provide the MV/4000 computer and the ISC with the necessary ability to communicate.

Universal Power Supply Controller

The universal power supply controller (UPSC) is a microprocessor- controlled power system that performs diagnostic functions. The UPSC performs a power-up diagnostic self test, monitors the system power, and reports failures, problems, and status to the MV/4000 computer. The UPSC is programmable and responds to a request for status or if allowed to, can generate an interrupt request.

For further information, see the Device Management chapter.

System Control Program

The system control program (SCP) is a soft system console that also performs diagnostic functions. The MV/4000 simulates the SCP operations in the MV/4000 microcode. That is, when the SCP is to function, the MV/4000 computer temporarily halts the CPU operations and performs the SCP function.

As a soft system console, the SCP performs system control functions under operator control. It permits the operator to load or examine and modify main memory and the processor state.

The SCP operator's terminal gives the operator control over the MV/4000 system by sending commands to the system and providing direct responses and reports.

For further information, see the Device Management chapter.

C/350 Compatibility

The MV/4000 computer will fully support the instruction mnemonics and binary opcodes of most instructions implemented on the ECLIPSE C/350. This means that most programs that execute on the C/350 computer will also execute on the MV/4000 computer without recompiling or reassembling.

Note that you can use C/350 instructions that manipulate data between accumulators (without referring to memory) in MV/4000-system-specific programs without modification.

The *Principles of Operation, 32-Bit ECLIPSE® Systems* manual describes the compatibility of C/350 instructions, data types, and formats.

Appendix A contains a complete functional listing of the MV/4000 unique instructions.

Registers

The MV/4000 system implements the following registers, which the *Principles of Operation, 32-Bit ECLIPSE® systems* manual describes in detail:

- Four 64-bit floating-point accumulators.
- Four 32-bit fixed-point accumulators.
- One 32-bit processor status register.
- One 64-bit floating-point status register.
- Four 32-bit stack management registers.
- One 31-bit program counter.
- Eight 32-bit segment base registers.

Initialization

The processor assumes the physical mode upon power-up, a system reset, or the execution the *IORST* instruction with the following conditions applying.

Power Up

When the processor first powers up (and before the system microcode loads), the following actions occur:

- The processor performs a power-up test.
- The processor initializes all of memory (ignoring irrelevant ERCC errors).
- The processor performs a system reset.

  The system reset clears the registers and disables the logical address translation -- equating logical addresses to physical addresses.
- The processor performs an I/O reset.

  The processor disables DCH mapping and the contents of the DCH and BMC maps and are undefined.

The remaining actions depend on the position of the front panel lock switch. If the switch is locked, the processor automatically boots from the device specified by the front panel switches. If the front panel lock switch is not locked, the processor executes the kernel microcode. While executing the kernel microcode:

- The MV/4000 recognizes the NOVA®/800 instruction set (basic NOVA without auto-increment/decrement).
- The kernel soft console is similar to the NOVA/4 console.
- The DCH and BMC operate in the unmapped mode at 2.5 Mbytes/s.
- Except for **LCS** (070077_8) and **NCLID** (064077_8) instructions, the 16-bit and 32-bit ECLIPSE instructions are not available.

The microcode file can be loaded into memory with this kernel instruction set. The **LCS** instruction can then load the microcode from memory.

System Microcode Loads

After the processor loads the system microcode (following the power-up sequence or after a system reset), the following actions occur:

- The processor disables logical address translation.
- The values of the referenced and modified bits are indeterminate.
- The processor sets the processor status register (PSR) and bits 0 through 8 of the floating-point status register (FPSR) to 0.
- The processor disables error reporting.
- The processor halts.
- The processor initializes the I/O devices.

IORST Instruction

After the execution of the *IORST* instruction, the following actions occur:

- The processor disables logical address translation.
- The processor sets the PSR and bits 0 through 8 of the FPSR to 0.
- The processor disables error reporting.
- The processor disables data channel maps.

When in physical mode, effective address translation works the same way as it does when logical address translation is enabled. However, because the logical address space exceeds the physical address space, the processor truncates a number of the logical address's 31 most significant bits before referring to memory. The number of bits truncated is dependent upon the amount of physical memory available. The maximum length of the word address formed from this procedure will be 22 bits for 8 Mbytes of physical memory.

Chapter 2

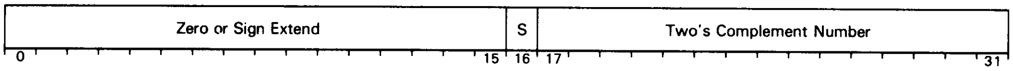
Fixed-Point Instruction Summary

This chapter summarizes the data formats and instructions for fixed-point and decimal/byte operations, and the processor status register. For further information refer to the *Principles of Operation, 32-Bit ECLIPSE® Systems* manual.

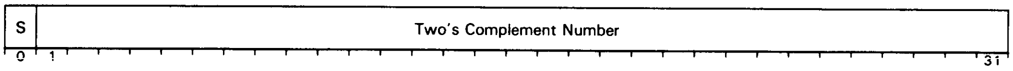
Fixed-Point Data Formats

The fixed-point accumulator formats for the 16- and 32-bit two's complement numbers, and for the 16- and 32-bit logical numbers are:

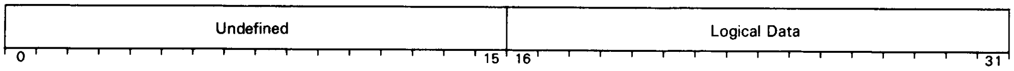
16-Bit Fixed-Point Two's Complement Format



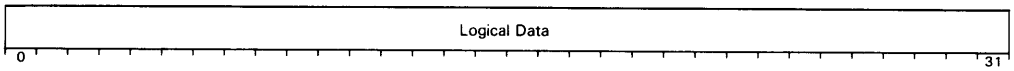
32-Bit Fixed-Point Two's Complement Format



16-Bit Fixed-Point Logical Format



32-Bit Fixed-Point Logical Format



Fixed-Point Instructions

Tables 2.1 through 2.12 list the fixed-point instructions.

Instruction	Operation
CVWN	Convert from 32-bit to 16-bit
SEX	Sign extend 16-bits to 32-bits
ZEX	Zero extend 16-bits to 32-bits

Table 2.1 Fixed-point precision conversion

Instruction	Operation
LDATS	Load accumulator with double word addressed by WSP
LNLDA	Narrow load accumulator
LNSTA	Narrow store accumulator
LWLDA	Wide load accumulator
LWSTA	Wide store accumulator
MOV *	Move and skip
NLDAI	Narrow load immediate
STATS	Store accumulator into double word addressed by WSP
WBLM	Wide block move
WLDAI	Wide load with wide immediate
WMOV	Wide move
WPOP	Wide pop accumulators
WPSH	Wide push accumulators
WXCH	Wide exchange accumulators
XCH *	Exchange accumulators
XNLDA	Narrow load accumulator
XNSTA	Narrow store accumulator
XWLDA	Wide load accumulator
XWSTA	Wide store accumulator

Table 2.2 Fixed-point data movement instructions

* ECLIPSE C/350 compatible instruction

Instruction	Operation
ADC *	Add complement and skip
ADD *	Add and skip
ADDI *	Extended add immediate
ADI *	Add immediate
INC*	Increment and skip
LNADD	Narrow add memory word to accumulator
LNADI	Narrow add immediate
LWADD	Wide add memory word to accumulator
LWADI	Wide add immediate
NADD	Narrow add
NADDI	Narrow extended add immediate
NADI	Narrow add immediate
WADC	Wide add complement
WADD	Wide add
WADDI	Wide add with wide immediate
WADI	Wide add immediate
WINC	Wide increment (no skip)
WNADI	Wide add with narrow immediate
XNADD	Narrow add accumulator to memory word
XNADI	Narrow add immediate
XWADD	Wide add memory word to accumulator
XWADI	Wide add immediate

Table 2.3 Fixed-point addition instructions

* ECLIPSE C/350 compatible instruction

Instruction	Operation
LNSBI	Narrow subtract immediate
LNSUB	Narrow subtract memory word
LWSBI	Wide subtract immediate
LWSUB	Wide subtract memory word
NSBI	Narrow subtract immediate
NSUB	Narrow subtract
SBI *	Subtract immediate
SUB *	Subtract and skip
WSBI	Wide subtract immediate
WSUB	Wide subtract
XNSBI	Narrow subtract immediate
XNSUB	Narrow subtract memory word
XWSBI	Wide subtract immediate
XWSUB	Wide subtract memory word

Table 2.4 Fixed-point subtraction instructions

* ECLIPSE C/350 compatible instruction

Instruction	Operation
LNMUL	Wide multiply memory word
LWMUL	Wide multiply memory word
MUL *	Unsigned multiply
MULS *	Signed multiply
NMUL	Narrow sign extend multiply
WMUL	Wide multiply
WMULS	Wide signed multiply
XNMUL	Narrow multiply memory word
XWMUL	Wide multiply memory word

Table 2.5 Fixed-point multiplication instructions

* ECLIPSE C/350 compatible instruction

Instruction	Operation
DIV *	Unsigned divide
DIVS *	Signed divide
DIVX *	Sign extend and divide
HLV *	Halve (AC/2)
LNDIV	Narrow divide memory word
LWDIV	Wide divide memory word
NDIV	Narrow sign extend divide
WDIV	Wide divide
WDIVS	Wide signed divide
WHLV	Wide halve
XNDIV	Narrow divide memory word
XWDIV	Wide divide memory word

Table 2.6 Fixed-point division instructions

* ECLIPSE C/350 compatible instruction

Instruction	Operation
ADC*	Add complement with optional CARRY initialization
ADD*	Add with optional CARRY initialization
AND*	AND with optional CARRY initialization
COM*	One's complement with optional CARRY initialization
CRYTC	Complement CARRY
CRYTO	Set CARRY to 1
CRYTZ	Set CARRY to 0
INC *	Increment with optional CARRY initialization
MOV *	Move with optional CARRY initialization
NEG*	Negate with optional CARRY initialization
SUB *	Subtract with optional CARRY initialization

Table 2.7 Initializing carry instructions

* ECLIPSE C/350 compatible instruction

Instruction	Operation
ADC*	Add complement with optional skip
ADD*	Add with optional skip
INC *	Increment with optional skip
MOV *	Move with optional skip
NSALA	Narrow skip on all bits set in accumulator
NSALM	Narrow skip on all bits set in memory location
NSANA	Narrow skip on any bit set in accumulator
NSANM	Narrow skip on any bit set in memory location
SGE *	Skip if ACS greater than or equal to ACD
SGT *	Skip if ACS greater than ACD
SNOVR	Skip on OVR reset
SUB *	Subtract with optional skip
WCLM	Wide compare to limits and skip
WSALA	Wide skip on all bits set in accumulator
WSALM	Wide skip on all bits set in double word memory location
WSANA	Wide skip on any bit set in accumulator
WSANM	Wide skip on any bit set in double word memory location
WSEQ	Wide skip if ACS equal to ACD
WSEQI	Wide skip if equal to immediate
WSGE	Wide signed skip if ACS greater than or equal to ACD
WSGT	Wide signed skip if ACS greater than ACD
WSGTI	Wide skip if AC greater than immediate
WSKBO	Wide skip on AC bit set to 1
WSKBZ	Wide skip on AC bit set to 0
WSLE	Wide signed skip if ACS less than or equal to ACD
WSLEI	Wide skip if AC less than or equal to immediate
WSLT	Wide signed skip if ACS less than ACD
WSNB	Wide skip on addressed bit set to 1
WSNE	Wide skip if ACS not equal to ACD
WSNEI	Wide skip if AC not equal to immediate
WSZB	Wide skip on addressed bit set to 0
WSZBO	Wide skip on addressed bit set to 0 and set bit to 1
WUGTI	Wide unsigned skip if AC greater than immediate
WULEI	Wide unsigned skip if AC less than or equal to immediate
WUSGE	Wide unsigned skip if ACS greater than or equal to ACD
WUSGT	Wide unsigned skip if ACS greater than ACD

Table 2.8 Fixed-point skip on condition instructions

* ECLIPSE C/350 compatible instruction

Instruction	Operation
DSZTS	Decrement the double word addressed by WSP (skip if 0)
INC *	Increment and skip
ISZTS	Increment the double word addressed by WSP (skip if 0)
LNDSZ	Narrow decrement and skip if 0
LNISZ	Narrow increment and skip if 0
LWDSZ	Wide decrement and skip if 0
LWISZ	Wide increment and skip if 0
XNDSZ	Narrow decrement and skip if 0
XNISZ	Narrow increment and skip if 0
XWDSZ	Wide decrement and skip if 0
XWISZ	Wide increment and skip if 0

Table 2.9 Fixed-point increment or decrement word and skip instructions

* ECLIPSE C/350 compatible instruction

Instruction	Operation
ANC *	AND with complemented source
AND *	AND
ANDI *	AND immediate
COM *	Complement
IOR *	Inclusive OR
IORI *	Inclusive OR immediate
LOB *	Locate lead bit
LRB *	Locate and reset lead bit
NEG *	Negate
NNEG	Narrow negate
WANC	Wide AND with complemented source
WAND	Wide AND
WANDI	Wide AND immediate
WBTO	Wide set bit to 1
WBTZ	Wide set bit to 0
WCOB	Wide count bits
WCOM	Wide complement (one's complement)
WIOR	Wide inclusive OR
WIORI	Wide inclusive OR immediate
WLOB	Wide locate lead bit
WLRB	Wide locate and reset lead bit
WLSN	Wide load sign
WNEG	Wide negate
WXOR	Wide exclusive OR
WXORI	Wide exclusive OR immediate
XOR *	Exclusive OR
XORI *	Exclusive OR immediate

Table 2.10 Logical instructions

* ECLIPSE C/350 compatible instruction

Instruction	Operation
AND *	Logical AND with optional shift
COM *	Logical one's complement with optional shift
DLSH *	Double logical shift
LSH *	Logical shift
NEG *	Logical negate with optional shift
WLSH	Wide logical shift
WLSHI	Wide logical shift immediate
WLSI	Wide logical shift left immediate

Table 2.11 Logical shift instructions

* ECLIPSE C/350 compatible instruction

Instruction	Operation
AND *	AND with optional skip
COM *	One's complement with optional skip
NEG *	Negate with optional skip
WSNB	Wide skip on nonzero bit
WSZB	Wide skip on 0 bit
WSZBO	Wide skip on 0 bit and set bit to 1

Table 2.12 Fixed-point logical skip instructions

* ECLIPSE C/350 compatible instruction

Processor Status Register

Table 2.13 lists the PSR manipulation instructions. The format for the PSR is:

OVK	OVR	IRES	IXCT	Reserved	Software Reserved
0	1	2	3	4 13	14 15

Argument Count
16 31

Bits 4 through 15 of the PSR are set to zero in a return block. When the PSR is loaded or restored, bits 4 through 15 are ignored.

The argument count appears as the second word of the PSR in a wide return block.

Instruction	Operation
FXTD	Disable fixed-point trap (resets OVK and disables trap)
FXTE	Enable fixed-point trap (sets OVK and enables trap)
LCALL	Call subroutine
LPSR	Load PSR into AC0
SPSR	Store PSR from AC0
WPOPB	Wide pop block
WRSTR	Wide restore
WDPOP	Wide pop context block
WRTN	Wide return
WSAVR	Wide save and set OVK to 0
WSAVS	Wide save and set OVK to 1
WSSVR	Wide special save and set OVK to 0
WSSVS	Wide special save and set OVK to 1
XCALL	Call subroutine
XVCT	I/O vector interrupt

Table 2.13 PSR manipulation instructions

* ECLIPSE C/350 compatible instruction

Decimal/Byte Operations

Tables 2.14 through 2.19 list the decimal/byte instructions.

Instruction	Operation
LLDB	Load byte
LSTB	Store byte
WCMT	Wide character move until true
WCMV	Wide character move
WCTR	Wide character translate and compare
WEDIT	Convert and insert string of decimal or ASCII characters
WLDB	Wide load byte
WSTB	Wide store byte
XLDB	Load byte
XSTB	Store byte

Table 2.14 Fixed-point byte movement instructions

Instruction	Operation
WLDI	Convert a decimal and load into FPAC
WLDIX	Convert a decimal, extend, and load it into four FPACs
WSTI	Convert FPAC data and load into memory
WSTIX	Convert the four FPACs and load into memory

Table 2.15 Fixed-point to floating-point conversion and store instructions

Instruction	Operation
LLEF	Load effective address
LLEFB	Load effective byte address
LPEF	Push address
LPEFB	Push byte address
WMOVR	Wide move right (convert byte pointer to word pointer)
XLEF	Load effective address
XLEFB	Load effective byte address
XPEF	Push effective address
XPEFB	Push effective byte address

Table 2.16 Load effective word and byte address instructions

Instruction	Operation
DADI	Add signed integer to destination indicator
DAPS	Add signed integer to opcode pointer if sign flag is 0
DAPT	Add signed integer to opcode pointer if trigger is 1
DAPU	Add signed integer to opcode pointer
DASI	Add signed integer to source indicator
DDTK	Decrement a word in the stack by one and jump if word is nonzero
DEND	End edit subprogram
DICI	Insert characters immediate
DIMC	Insert character j times
DINC	Insert character once
DINS	Insert character a or character b depending on sign flag
DINT	Insert character a or character b depending on trigger
DMVA	Move j alphabetical characters
DMVC	Move j characters
DMVF	Move j float
DMVN	Move j numerics
DMVO	Move digit with overpunch
DMVS	Move numeric with zero suppression
DNDF	End float
DSSO	Set sign flag to 1
DSSZ	Set sign flag to 0
DSTK	Store in stack
DSTO	Set trigger to 1
DSTZ	Set trigger to 0

Table 2.17 Edit subprogram instructions

Instruction	Operation
DAD *	Add two unsigned BCD numbers in two accumulators
DSB *	Subtract two unsigned BCD numbers in two accumulators

Table 2.18 BCD arithmetic instructions

* ECLIPSE C/350 compatible instruction

Instruction	Operation
DHXL *	Double hex shift left
DHXR *	Double hex shift right
HXL *	Hex shift left
HXR *	Hex shift right

Table 2.19 Hex shift instructions

* ECLIPSE C/350 compatible instruction

Chapter 3
Floating-Point Instruction Summary

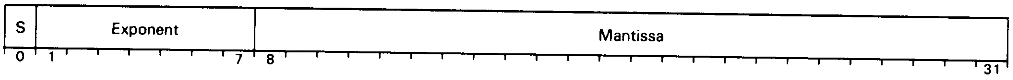
This chapter summarizes the floating-point data formats, floating-point instructions, and the floating-point status register. For further information, refer to the *Principles of Operation, 32-Bit ECLIPSE® Systems* manual.

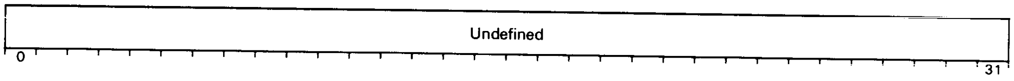
Floating-Point Data Formats

The floating-point accumulator formats for single and double precision floating-point numbers are:

Single Precision

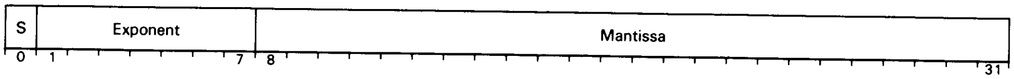
Double Word



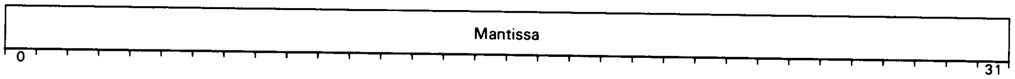


Double Precision

Double Word 0



Double Word 1



Floating-Point Instructions

Tables 3.1 through 3.8 list the floating-point instructions.

Instruction	Operation
FAD *	Add double (FPAC to FPAC)
FAS *	Add single (FPAC to FPAC)
LFAMD	Add double (memory to FPAC)
LFAMS	Add single (memory to FPAC)
XFAMD	Add double (memory to FPAC)
XFAMS	Add single (memory to FPAC)

Table 3.1 Floating-point addition instructions

* ECLIPSE C/350 compatible instruction

Instruction	Operation
FSD *	Subtract double (FPAC from FPAC)
FSS *	Subtract single (FPAC from FPAC)
LFSMD	Subtract double (memory from FPAC)
LFSMS	Subtract single (memory from FPAC)
XFSMD	Subtract double (memory from FPAC)
XFSMS	Subtract single (memory from FPAC)

Table 3.2 Floating-point subtraction instructions

* ECLIPSE C/350 compatible instruction

Instruction	Operation
FMD *	Multiply double (FPAC by FPAC)
FMS *	Multiply single (FPAC by FPAC)
LFMMD	Multiply double (FPAC by memory)
LFMMS	Multiply single (FPAC by memory)
XFMMD	Multiply double (FPAC by memory)
XFMMS	Multiply single (FPAC by memory)

Table 3.3 Floating-point multiplication instructions

* ECLIPSE C/350 compatible instruction

Instruction	Operation
FDD *	Divide double (FPAC by FPAC)
FDS *	Divide single (FPAC by FPAC)
FHLV *	Halve (FPAC/2)
LFDMD	Divide double (FPAC by memory)
LFDMS	Divide single (FPAC by memory)
XFDMD	Divide double (FPAC by memory)
XFDMS	Divide single (FPAC by memory)

Table 3.4 Floating-point division instructions

* ECLIPSE C/350 compatible instruction

Instruction	Operation
FCMP *	Compare two floating-point numbers (set N and Z)
FSEQ *	Skip on 0 (Z = 1)
FSGE *	Skip on greater than or equal to 0 (N = 0)
FSGT *	Skip on greater than 0 (N and Z = 0)
FSLE *	Skip on less than or equal to 0 (N and Z = 1)
FSLT *	Skip on less than 0 (N = 1)
FSND *	Skip on no 0 divide (DVZ = 0)
FSNE *	Skip on nonzero (Z = 0)
FSNER *	Skip on no error (ANY = 0)
FSNM *	Skip on no mantissa overflow (MOF = 0)
FSNO *	Skip on no overflow (OVF = 0)
FSNOD *	Skip on no overflow and no 0 divide (OVF and DVZ = 0)
FSNU *	Skip on no underflow (UNF = 0)
FSNUD *	Skip on no underflow and no 0 divide (UNF and DVZ = 0)
FSNUO *	Skip on no underflow and no overflow (UNF and OVF = 0)

Table 3.5 Floating-point skip on condition instructions

* ECLIPSE C/350 compatible instruction

Instruction	Operation
FEXP *	Load exponent (AC0 17-23 to FPAC 1-7)
FAB *	Compute absolute value (set sign of FPAC to 0)
FFAS *	Fix to AC (FPAC to AC)
FINT *	Integerize (FPAC)
FLAS *	Float from AC (AC to FPAC)
FNEG *	Negate
FNOM *	Normalize (FPAC)
FRDS	Floating-point round double to single
FRH *	Read high word (FPAC 0-15 to AC0 16-31)
FSCAL *	Scale floating point
WFFAD	Wide fix from FPAC
WFLAD	Wide float from AC

Table 3.6 Floating-point binary conversion instructions

* ECLIPSE C/350 compatible instruction

Instruction	Operation
WLDI	Convert a decimal and load into FPAC
WLDIX	Convert a decimal, extend, and load it into four FPACs
WSTI	Convert FPAC data and load into memory
WSTIX	Convert the four FPACs and load into memory

Table 3.7 Floating-point decimal conversion instructions

Instruction	Operation
FMOV *	Move floating point (FPAC to FPAC)
LFLDD	Load floating-point double
LFLDS	Load floating-point single
LFSTD	Store floating-point double
LFSTS	Store floating-point single
WFPOP	Wide floating-point pop
WFPSH	Wide floating-point push
XFLDD	Load floating-point double
XFLDS	Load floating-point single
XFSTD	Store floating-point double
XFSTS	Store floating-point single

Table 3.8 Floating-point data movement instructions

* ECLIPSE C/350 compatible instruction

Floating-Point Status Register

Table 3.9 lists the FPSR manipulation instructions. The format for the FPSR is:

ANY	OVF	UNF	DVZ	MOF	TE	Z	N	RND	Res	0	0	FPMOD
0	1	2	3	4	5	6	7	8	9	10	11	12 15

Reserved
0 15

0	Floating-Point Program Counter
0	1 15

Floating-Point Program Counter
0 15

Instruction	Operation
FCLE *	Clear errors (FPSR)
FTD *	Floating-point trap disable (resets TE)
FTE *	Floating-point trap enable (sets TE)
LFLST	Load FPSR
LFSST	Store FPSR
WFPSH	Push floating-point state
WFPOP	Pop floating-point state

Table 3.9 FPSR instructions

* ECLIPSE C/350 compatible instruction

Chapter 4
Stack Management Instruction Summary

This chapter summarizes the instructions that affect the wide stack. For further information, refer to the *Principles of Operation, 32-Bit ECLIPSE® Systems* manual.

Table 4.1 lists the wide stack register instructions; Table 4.2 lists the instructions that access the wide stack; Table 4.3 lists the instructions that push or pop wide stack return blocks; and Table 4.4 lists the instructions that push or pop one or more double words onto the wide stack. Table 4.4 also lists the number of words that the instructions require beyond the wide stack limit for a stack fault return block.

Instruction	Action
LDAFP	Load accumulator with the WFP register contents
LDASB	Load accumulator with the WSB register contents
LDASL	Load accumulator with the WSL register contents
LDASP	Load accumulator with the WSP register contents
STAFP	Store accumulator in the WFP register
STASB	Store accumulator in the WSB register
STASL	Store accumulator in the WSL register
STASP	Store accumulator in the WSP register
WMSP	Wide modify WSP register

Table 4.1 Wide stack register instructions

Instruction	Action
DSZTS	Decrement the double word addressed by WSP (skip if 0)
ISZTS	Increment the double word addressed by WSP (skip if 0)
LDATS	Load accumulator with double word addressed by WSP
LPEF	Push address
LPEFB	Push byte address
LPSHJ	Push jump to subroutine (pop with WPOPJ)
STATS	Store accumulator into double word addressed by WSP
WFPOP	Wide floating-point pop
WFPSH	Wide floating-point push
WPOP	Wide pop accumulators (push with WPSH)
WPOPJ	Wide pop PC and jump (push with LPSHJ or XPSHJ)
WPSH	Wide push accumulators (pop with WPOP)
XPEF	Push address
XPEFB	Push byte address
XPSHJ	Push jump to subroutine (pop with WPOPJ)

Table 4.2 Wide stack double-word access instructions

Instruction	Action
BKPT	Breakpoint handler (return from breakpoint handler with PBX)
LCALL	Call subroutine (return from call with WRTN)
PBX	Pop block and execute (return from breakpoint handler)
WPOPB	Wide pop block
WRSTR	Wide restore from an interrupt
WRTN	Wide return via wide save (WSAVR, WSAVS, WSSVR, and WSSVS)
WSAVR	Wide save/reset overflow mask (used with LCALL and XCALL)
WSAVS	Wide save/set overflow mask (used with LCALL and XCALL)
WSSVR	Wide special save/reset overflow mask (used with LJSR & XJSR)
WSSVS	Wide special save/set overflow mask (used with LJSR & XJSR)
WXOP	Extended operation (return with WPOPB; used to expand instruction set)
XCALL	Call subroutine (return from call with WRTN)

Table 4.3 Wide stack return block instructions

Instruction	Description	Double Words	
		Pushed or (Popped)	Required Beyond WSL for Stack Fault
ADD, etc.	Arithmetic with OVK enabled	0	11
FAD, etc.	Arithmetic with TE enabled	0	11
BKPT	Breakpoint handler	6	11
LCALL	Subroutine call	1	6
LPEF	Push address	1	6
LPEFB	Push byte address	1	6
LPSHJ	Push jump	1	6
PBX	Pop block and execute	(6)	5
WEDIT	Wide edit	16	27
WFPOP	Wide floating-point pop	(10)	5
WFPSH	Wide floating-point push	10	15
WPOP	Wide pop accumulators	(1-4)	5
WPOPB	Wide pop block	(6)	5
WPOPJ	Wide pop PC and jump	(1)	5
WPSH	Wide push accumulators	1-4	9
WRSTR	Wide restore	(10)	5
WRTN	Wide return	(6)	5
WSAVR	Wide save/reset OVK	5	10
WSAVS	Wide save/set OVK	5	10
WSSVR	Wide special save/reset OVK	6	11
WSSVS	Wide special save/set OVK	6	11
WXOP	Extended operation	6	11
XCALL	Subroutine call	1	6
XPEF	Push address	1	5
XPEFB	Push byte address	1	5
XPSHJ	Push jump to subroutine	1	5
XVCT	Vector on I/O interrupt	6	11

Table 4.4 Multiword wide stack instructions

Chapter 5
Program Flow Management

Program flow management consists of program flow, interrupt handling, and fault handling.

Program flow management occurs with the MV/4000 address translator enabled, or without the MV/4000 address translator enabled (physical mode). The *Principles of Operation, 32-Bit ECLIPSE® Systems* manual describes program flow management for these conditions.

This chapter presents the program counter, address space available, the sequence of events upon an interrupt, a listing of program flow instructions, and a summary of fault handling.

Program Counter

The program counter (PC), which specifies the logical address of the instruction, controls the sequence of executing instructions. Address wraparound occurs within the current segment since only bits 4 through 31 take part in incrementing the PC.

To address the next instruction (for normal program flow), the processor either increments the PC or forces an address into the PC. The processor increments the PC by:

- One when executing a one-word instruction (such as **NADI**)
- Two when executing a two-word instruction (such as **NADDI**)
- Three when executing a three-word instruction (such as **LNADI**)
- Four when executing a four-word instruction (such as **LCALL**)

When the processor forces an address into the PC, the processor clears the instruction processor pipeline and initiates a different program sequence. Any of the following events alter the normal program sequence:

- Executing the **XCT** instruction
- Executing a jump instruction
- Executing a skip instruction

- Executing a subroutine call or return instruction
- Detecting a fault
- Detecting an I/O interrupt request

Program flow is further described in the *Principles of Operation, 32-Bit ECLIPSE® Systems* manual.

Address Space

MV/4000 main memory physical address space can range from 0.5 Mbyte to 8 Mbytes.

The address translator has a 4-Gbyte logical address space, divided into eight segments of 512 Mbytes each. The *Principles of Operation, 32-Bit ECLIPSE® Systems* manual describes segmentation and MV/4000 system addressing. The MV/4000 computer uses 31-bit word addresses and 32-bit byte addresses that can refer to all 4 Gbytes of the logical address space.

Interrupts

When an interrupt occurs, the processor disables further interrupts by setting the interrupt on (ION) flag to 0. The state of the address translator determines the actions that follow.

Interrupt Sequence

With the address translator disabled, the processor fetches the contents of physical location 1 and prepares to resolve any indirection. The processor is operating in physical mode and treats this address as the address of the interrupt handler.

With the address translator enabled, the processor fetches the contents of logical location 1 in page zero of segment 0. This location contains the address of the interrupt handler. The processor then determines the current segment of execution. If it is not segment 0, the processor performs a ring crossing to segment 0. Next, the processor must resolve the interrupt handler address.

If the fetched address of the interrupt handler is indirect, the processor resolves it to a final direct address. This address refers to the first instruction of the handler.

The first instruction of the interrupt handler will be one of the following three types:

- An **XVCT** instruction.
- Any other MV/4000-system-specific instruction (Type 1).
- C/350 instructions, **WBR**, and some MV/4000-system-specific memory to accumulator instructions (Type 2).

The flow chart in Figure 5.1 summarizes the interrupt sequence.

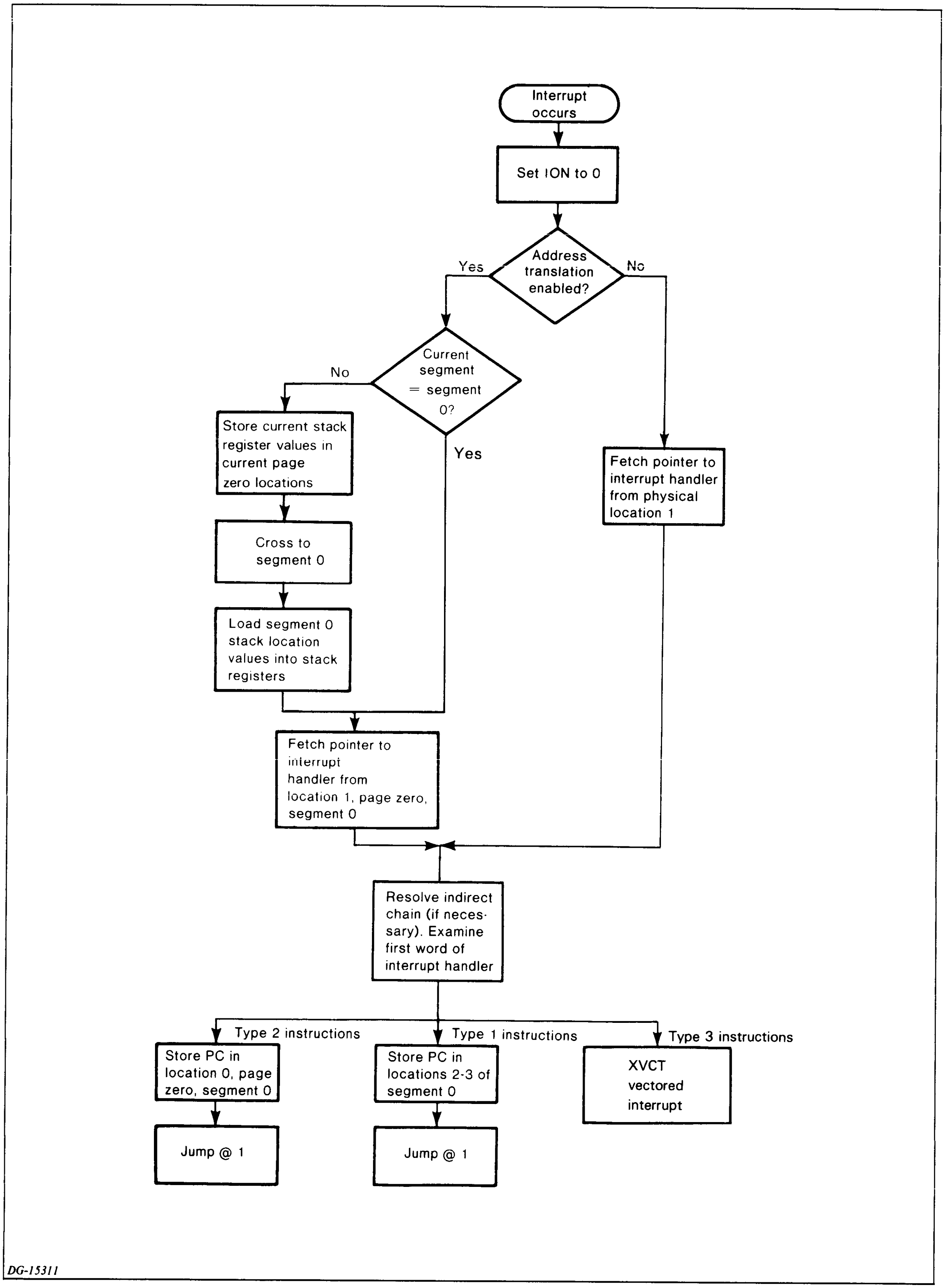

Interrupt occurs
Set ION to 0
Address translation enabled?
Yes
No
Current segment = segment 0?
No
Yes
Store current stack register values in current page zero locations
Cross to segment 0
Load segment 0 stack location values into stack registers
Fetch pointer to interrupt handler from physical location 1
Fetch pointer to interrupt handler from location 1, page zero, segment 0
Resolve indirect chain (if neces-sary). Examine first word of interrupt handler
Type 2 instructions
Type 1 instructions
Type 3 instructions
Store PC in location 0, page zero, segment 0
Store PC in locations 2-3 of segment 0
XVCT vectored interrupt
Jump @ 1
Jump @ 1
DG-15311

Figure 5.1 Interrupt sequence

Interrupting an Instruction

When the processor honors an interrupt, program execution stops. How the processor halts program execution to service the interrupt depends upon the instruction currently executing within the program. The currently executing instruction will be one of the following three kinds:

- A noninterruptible instruction.
- A restartable instruction.
- A resumable instruction.

Refer to Table 5.1 for a listing of restartable or resumable instructions. Any instruction not listed as either restartable or resumable is noninterruptible.

Restartable (From Beginning)		Restartable (With Updated Values)		Resumable	
*FMD	*LSN	*BAM	WBLM	*EDIT	WEDIT
*FMMD	ORFB	*BLM	WCMP	*LDI	WLDI
*FDD	PATU	*CMP	WCMT	*LDIX	WLDIX
*FDMD	RRFB	*CMT	WCMV	*STI	WSTI
LCALL	WDPOP	*CMV	WCTR	*STIX	WSTIX
LFMD	XCALL	*CTR	WLMP	NBStc	WBStc
LFDMD	XFDMD			NFStc	WFStc
LSBRA	XFMMD				
LSBRS					

Table 5.1 Restartable or resumable instructions

* Denotes a C/350 instruction.

Noninterruptible Instructions

If an instruction is noninterruptible, the processor finishes executing that instruction before it services the interrupt (refer to Figure 5.2). Examples of noninterruptible instructions are *Add, Load Accumulator,* and *Complement.*

The processor does not set any bits in the PSR if an interrupt occurs during a noninterruptible instruction.

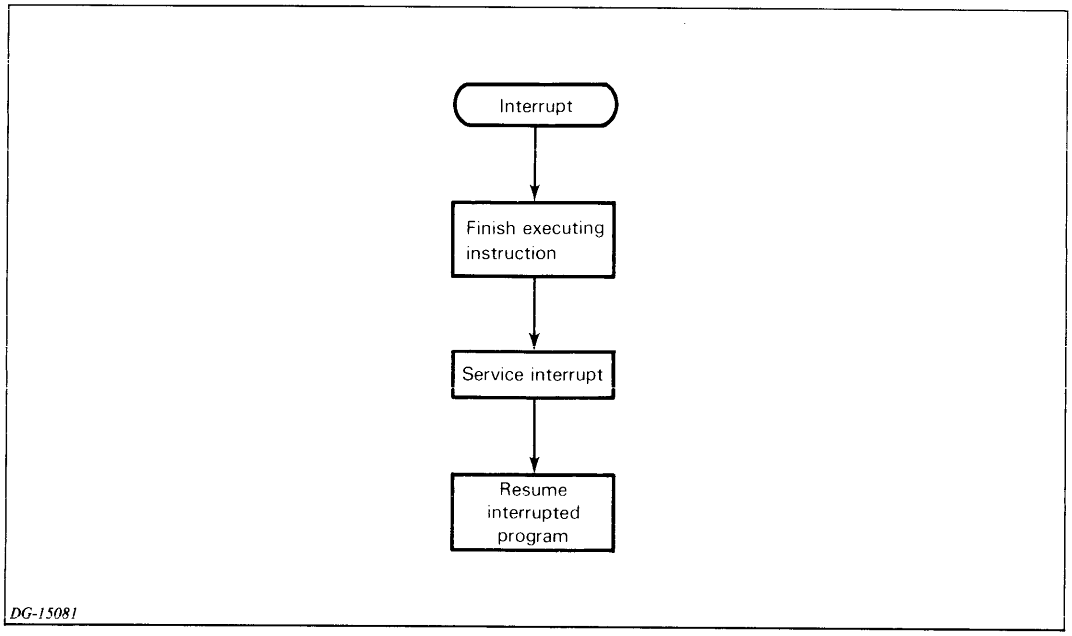


Figure 5.2 Noninterruptible instruction interrupt sequence

Restartable Instructions

If an instruction is restartable, the processor services the interrupt before the instruction finishes. When an interrupt occurs, the processor saves the address of the interrupted instruction in the PC, and then services the interrupt. When servicing is complete, the processor can restart the interrupted instruction in one of the following two ways.

- If the parameters of the restartable instruction *have not changed*, then the processor will restart the instruction from the beginning. That is, if an interrupt occurs during a *Floating-Point Divide* instruction, the processor will restart the instruction from the beginning because the accumulators containing the operands have not changed.
- If the parameters of the interrupted instruction *have changed*, the processor will restart execution with the updated values. This type of instruction (*Block Move,* for example) uses pointers to source and destination locations and updates them after each one-word move. After servicing the interrupt, the processor restarts execution with the current values of the source and destination pointers, not the original values.

Note that the processor sets bit 2 of the PSR to 1 when an interrupt occurs during a restartable instruction.

Figure 5.3 summarizes the interrupt sequence for a restartable instruction.

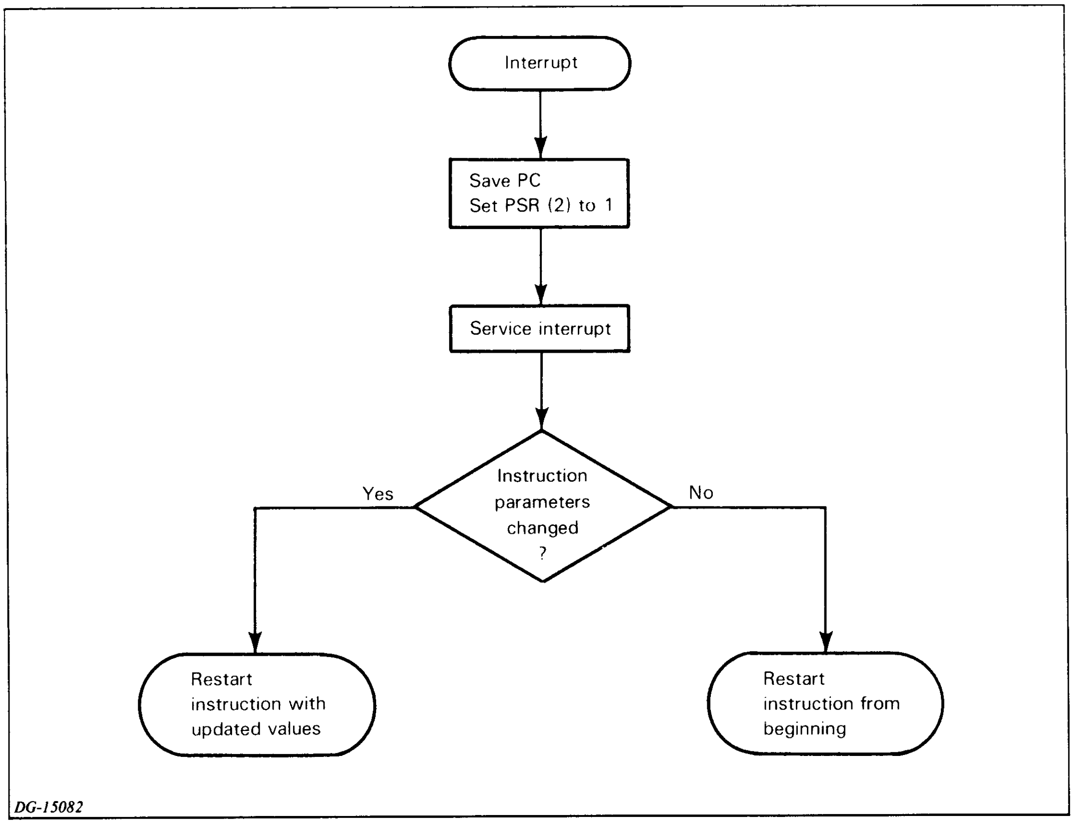

Interrupt
Save PC
Set PSR (2) to 1
Service interrupt
Instruction parameters changed ?
Yes
No
Restart instruction with updated values
Restart instruction from beginning
DG-15082

Figure 5.3 Restartable instruction interrupt sequence

Resumable Instructions

As with restartable instructions, the processor services an interrupt before finishing a resumable instruction. The processor must save a copy of internal processor state if it is to restart a resumable instruction correctly.

The following discussion describes what happens when an interrupt occurs during execution of a resumable instruction.

Before interrupting resumable instructions, you should ensure that:

- You define a stack.
- The interrupt handler uses **WPOPB, WRSTR, WRTN,** or **LPSR** to return to the interrupted program. These instructions restore the PSR when interrupt service completes.

When an interrupt occurs, the processor saves the address of the interrupted instruction, and pushes a copy of all necessary processor information (the microstate block) onto the current stack.

The information needed depends upon the interrupted instruction. If the processor is interrupted during execution of a **WEDIT** instruction, the processor sets bit 2 of the PSR (IRES) to 1. If the processor is interrupted during execution of a resumable or restartable instruction resulting from a **PBX** instruction, the processor sets bit 3 of the PSR (IXCT) to 1.

After pushing the block, the processor checks for stack overflow. If it detects a stack overflow, the processor:

1. Services the interrupt.
2. Returns to the interrupted program.
3. Services the stack fault (if necessary).
4. Resumes the interrupted instruction.

Next, the processor restores the PSR using the appropriate return instruction. If a resumable instruction was interrupted, then the processor tests bits 2 and 3. If either bit contains a 1, the processor examines the microstate block on the current wide stack to determine the type of microinterrupt.

- If the microstate block is valid, the processor resumes executing the interrupted instruction.
- If the block is invalid, the actions taken depend on the interrupted instruction:
  - An MV/4000-system-specific instruction causes a protection fault to occur. Accumulator 1 (AC1) will contain the code 12 to indicate the invalid microstate block.
  - A C/350 floating-point instruction causes a floating-point fault to occur.
  - A C/350 *Decimal/ASCII* instruction causes a narrow decimal/ASCII fault to occur. AC1 will contain the code 5 to indicate the invalid microstate block.

- A *PBX type* instruction. If the interrupted instruction was inserted into the instruction stream, (e.g., PBX), then the processor had set the IXCT flag in the PSR and pushed the op-code of the executing instruction onto the wide stack.

Table 5.2 shows the processor settings of bit 2 of the PSR and bit 3 of the PSR when an interrupt occurs during execution of a resumable instruction. Figure 5.4 summarizes the sequence of events upon the interruption of a resumable instruction.

NOTE: *When an interrupt occurs during a ring crossing, the saved PC points to the first instruction of the called procedure.*

Instruction	PSR Bit 2 (IRES)	PSR Bit 3 (IXCT)
C/350	Unchanged	Unchanged
MV/4000-specific	Function of instruction	Function of instruction

Table 5.2 State of PSR bits 2 and 3

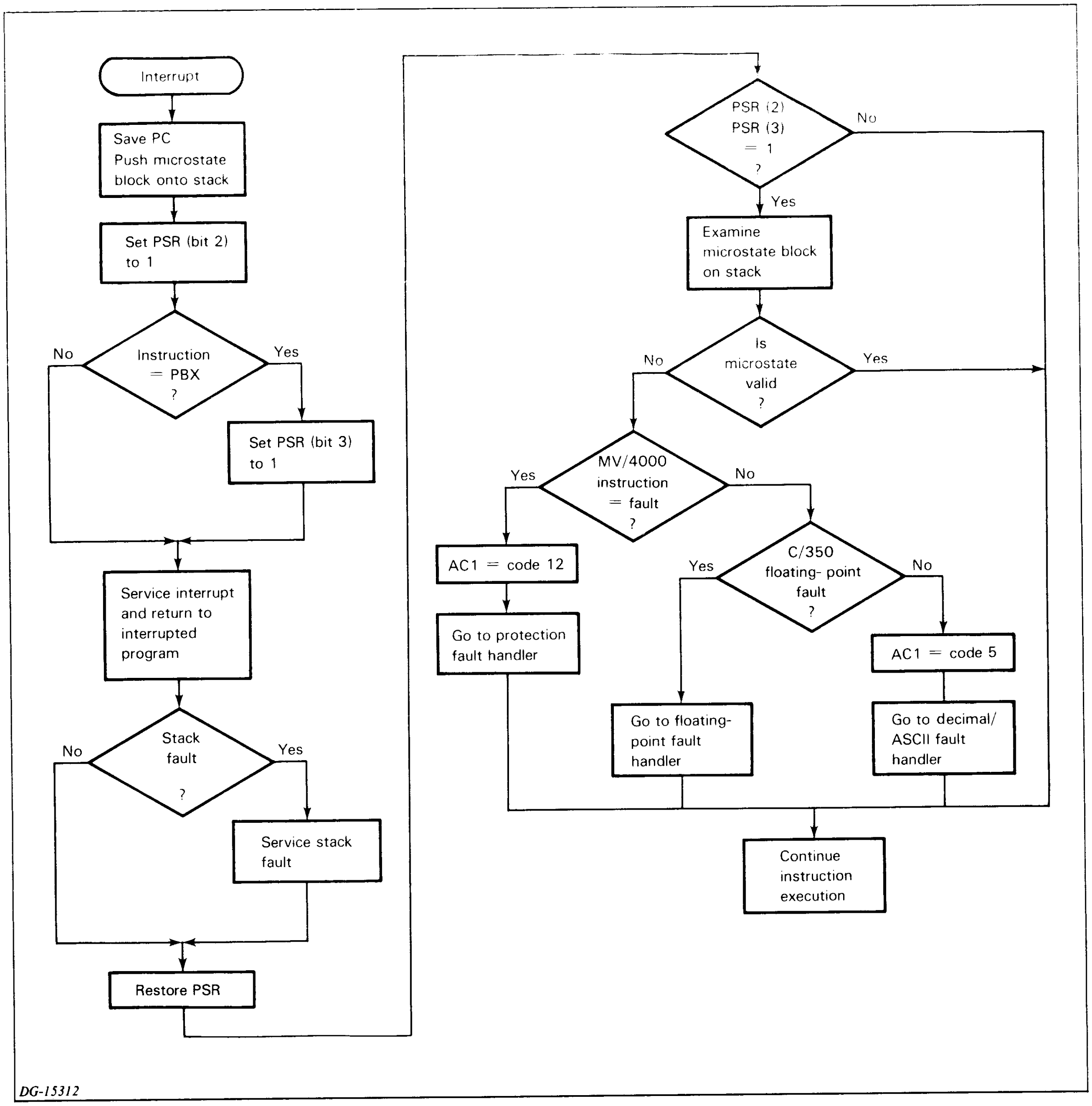

Interrupt
Save PC Push microstate block onto stack
Set PSR (bit 2) to 1
Instruction = PBX ?
No
Yes
Set PSR (bit 3) to 1
Service interrupt and return to interrupted program
Stack fault ?
No
Yes
Service stack fault
Restore PSR
PSR (2) PSR (3) = 1 ?
No
Yes
Examine microstate block on stack
Is microstate valid ?
No
Yes
MV/4000 instruction = fault ?
Yes
No
AC1 = code 12
Go to protection fault handler
C/350 floating- point fault ?
Yes
No
AC1 = code 5
Go to floating-point fault handler
Go to decimal/ ASCII fault handler
Continue instruction execution
DG-15312

Figure 5.4 Resumable instruction interrupt sequence

Program Flow Instructions

Tables 5.3 through 5.8 list the instructions that affect program flow. For further information, refer to the *Principles of Operation, 32-Bit ECLIPSE® Systems* manual.

Instruction	Action
XCT *	Execute bits 16-31 of an accumulator as an instruction.

Table 5.3 Execute accumulator instruction

* ECLIPSE C/350 compatible instruction

Instruction	Action
LDSP	Dispatch
LJMP	Jump (with long displacement)
WBR	Branch (PC relative jump)
XJMP	Jump (with extended displacement)

Table 5.4 Jump instructions

Instruction	Action
FNS *	No skip
FSA *	Skip always
LNDO	Narrow do until greater than
LWDO	Wide do until greater than
XNDO	Narrow do until greater than
XWDO	Wide do until greater than
NBStc	Narrow search queue backward
NFStc	Narrow search queue forward
WBStc	Wide search queue backward
WFStc	Wide search queue forward

Table 5.5 Skip instructions

* ECLIPSE C/350 compatible instruction

Instruction	Action
BKPT	Breakpoint handler
LCALL	Call subroutine
LJSR	Jump to subroutine
LPSHJ	Push jump
PBX	Pop block and execute
WEDIT	Wide edit of alphanumeric
WPOPB	Wide pop block
WPOPJ	Wide pop PC and jump
WRTN	Wide return
WSAVR	Wide save/reset overflow mask
WSAVS	Wide save/set overflow mask
WSSVR	Wide special save/reset overflow mask
WSSVS	Wide special save/set overflow mask
WXOP	Wide extended operation
XCALL	Call subroutine
XJSR	Jump to subroutine
XPSHJ	Push jump

Table 5.6 Subroutine instructions

Instruction	Action
LCALL	Call subroutine
WPOPB	Wide pop block
WRTN	Wide return
XCALL	Call subroutine
WRSTR	Wide restore from an I/O interrupt

Table 5.7 Segment transfer instructions

Call Instruction or Sequence	Segment Crossing Permitted	Associated Save Instruction	Return Instruction
BKPT	no		PBX
LCALL	yes	WSAVR	WRTN
	yes	WSAVS	WRTN
LJSR	no	WSSVR	WRTN
	no	WSSVS	WRTN
LPSHJ	no		WPOPJ
WEDIT	no		DEND
WXOP	no		WPOPB
XCALL	yes	WSAVR	WRTN
	yes	WSAVS	WRTN
XJSR	no	WSSVR	WRTN
	no	WSSVS	WRTN
XPSHJ	no		WPOPJ

Table 5.8 Sequence of subroutine instructions

Fault Handling

While executing an instruction, the processor performs certain checks on the operation and the data. If the processor detects an error, a privileged or nonprivileged fault occurs before execution of the next instruction.

With the address translator enabled, the processor detects the following faults (also refer to the *Principles of Operation, 32-Bit ECLIPSE® Systems* manual):

Fault Generated	**Fault Type**
Protection Violation Fault	Privileged
Page Fault	Privileged
Stack Fault	Nonprivileged
Fixed-Point Overflow	Nonprivileged
Floating-Point Fault	Nonprivileged
Decimal/ASCII Fault	Nonprivileged

Appendix F lists the error codes returned to AC1, and denotes the type of fault generated.

Privileged Faults

The Memory and System Management chapter explains page faults. The *Principles of Operation, 32-Bit ECLIPSE® Systems* manual describes the handling of protection violation faults.

Nonprivileged Faults

The *Principles of Operation, 32-Bit ECLIPSE® Systems* manual describes the handling of nonprivileged faults.

Execution of C/350 instructions does not generate fixed-point faults. Certain C/350 arithmetic instructions (**ADD**, **DIV**, etc.) set the state of the carry bit. If detection of the appropriate fault is desired, it is necessary to set up a subroutine that checks the state of the carry bit upon completion of these instructions. A carry-out from accumulator bit 16 affects the MV/6000 system's carry bit upon execution of these C/350 instructions. The instruction dictionary in the *Principles of Operation, 32-Bit ECLIPSE® Systems* manual describes the C/350 instruction set and which instructions affect the carry bit.

Note that all faults that occur with the execution of C/350 instructions use the narrow stack.

Chapter 6
Queue Management Instruction Summary

This chapter summarizes the queue instructions. For further information, refer to the *Principles of Operation, 32-Bit ECLIPSE® Systems* manual.

Table 6.1 lists the queue instructions.

Instruction	Action
ENQH	Queue towards the head; add a data element to queue
ENQT	Queue towards the tail; add a data element to queue
DEQUE	Dequeue a queue data element; delete a data element
NBStc	Narrow search queue backward; 16-bit test condition
NFStc	Narrow search queue forward; 16-bit test condition
WBStc	Wide search queue backward; 32-bit test condition
WFStc	Wide search queue forward; 32-bit test condition
WMESS	Wide mask, skip and store if equal

Table 6.1 Queue instructions

Chapter 7
Device Management

This chapter summarizes the general I/O instructions, and presents the instructions for the manipulation of the following devices:

- Central Processing Unit
- Programmable Interval Timer
- Real-Time Clock
- Primary Asynchronous Line Input/Output
- System Control Program
- Data Channel and Burst Multiplexor Channel
- Universal Power Supply Controller

Refer to Appendix E for all the device codes, device mnemonics and priority mask bit assignments.

General I/O Instructions

Table 7.1 lists the general I/O instructions; Tables 7.2 and 7.3 list the device flags mnemonics. For further information, refer to the *Principles of Operation, 32-Bit ECLIPSE® Systems* manual.

Instruction	Operation
DIA[*f*] *	Data in A (from A buffer of device)
DIB[*f*] *	Data in B (from B buffer of device)
DIC[*f*] *	Data in C (from C buffer of device)
DOA[*f*] *	Data out A (to A buffer of device)
DOB[*f*] *	Data out B (to B buffer of device)
DOC[*f*] *	Data out C (to C buffer of device)
IORST *	I/O reset
NIO[*f*] *	No I/O transfer (initialize a BUSY/DONE flag)
PIO	Issue a programmed I/O command to a device
SKP*t* *	I/O skip (test a BUSY/DONE flag and skip on condition)

Table 7.1 General I/O instructions

The [*f*] or *t* defines the optional device flag handling.

The * identifies ECLIPSE C/350 compatible instructions.

Assembler Code for *f*	Bits 8 9	I/O		CPU ION
		BUSY	DONE	
(option omitted)	0 0	No effect	No effect	No effect
S	0 1	Set to 1	Set to 0	Set to 1
C	1 0	Set to 0	Set to 0	Set to 0
P	1 1	Pulses a special I/O bus control line		No effect

Table 7.2 Device flags for general devices

Assembler Code for *t*	Bits 8 9	I/O	CPU
BN	0 0	Test for BUSY = 1	Test for ION = 1
BZ	0 1	Test for BUSY = 0	Test for ION = 0
DN	1 0	Test for DONE = 1	Test for power fail = 1
DZ	1 1	Test for DONE = 0	Test for power fail = 0

Table 7.3 Device flags for skip instruction

Central Processor

Device Code
77_8

Assembler Mnemonic
CPU

Priority Mask Bit
None

Device Flags

Device flag commands to the CPU determine whether or not the processor can interrupt the current program with a program interrupt request. When the interrupt enable flag (ION) equals 1, the processor can interrupt the program (once the instruction following the enable has begun). When the interrupt enable flag equals 0, the processor cannot interrupt the program. The CPU interrupt enable flag is controlled by the device flag commands for device 77 as follows:

f=**S** Sets the interrupt enable flag to 1.

f=**C** Sets the interrupt enable flag to 0.

f=**P** The P flag causes an unimplemented instruction interrupt.

The assembler interprets the I/O instructions for the CPU using either the standard or a special I/O instruction format. Referring to Table 7.4, the instruction that initializes the devices and sets the priority mask bits to 0 uses the standard form of

DIC*[f] ac,***CPU**

or the special form of

IORST

The special **IORST** assembler statement equates to the standard assembler statement of

DICC 0,CPU

which sets all the BUSY and DONE flags to 0. You cannot append a device flag (**S**, **C**, or **P**) to the special form of a CPU instruction.

> NOTE: *The assembler detects a fatal format error if you append a device flag to a special CPU instruction.*

CPU Instructions

Table 7.4 lists the I/O instructions that affect the CPU device.

Assembler Statement	Operation
READS *ac* **DIA***[f]* *ac*, **CPU**	Reads console switches (places the contents of the soft console data switch register into an accumulator)
INTA *ac* **DIB***[f]* *ac*,**CPU**	Returns the device code of the interrupting device (interrupt acknowledge)
IORST **DIC***[f]* *ac*,**CPU**	Initializes the I/O system (resets the BUSY and DONE flags and all the priority mask bits to 0; clears certain CPU registers, and disables the DCH mapping and address translator)
MSKO *ac* **DOB***[f]* *ac*,**CPU**	Initializes or changes the priority mask
HALT **DOC***[f]* *ac*,**CPU**	Stops the processor
INTDS **NIOC CPU**	Sets ION flag to 0 (interrupt disable)
INTEN **NIOS CPU**	Sets ION flag to 1 (interrupt enable)
SKP*t* **CPU**	Tests the condition of the ION flag or power fail flag, and when true, it skips the next word in the program

Table 7.4 I/O instructions for CPU

Read Switches
READS *ac*
DIA*[f]* *ac*,CPU

0	1	1	ac	0	0	1	f	1	1	1	1	1	1
0	1	2	3 4	5	6	7	8 9	10	11	12	13	14	15

The *Read Switches* instruction places the contents of the console switches into bits 16 through 31 of the specified accumulator. After the transfer, the instruction sets the ION flag according to the function specified by *[f]*.

> **NOTES:** *The assembler recognizes the special mnemonic* **READS** ac *to be equivalent to* **DIA** ac,**CPU**.

The format of the specified accumulator after instruction execution is:

Undefined	Console Switches
0 15	16 31

Console Switches: 1=ON; 0=OFF

Interrupt Acknowledge
INTA *ac*
DIB*[f]* *ac*,CPU

0	1	1	ac		0	1	1	f		1	1	1	1	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The *Interrupt Acknowledge* instruction places the 6-bit device code of that device requesting an interrupt that is physically closest to the CPU on the I/O bus into bits 26 through 31 of the specified accumulator, setting bits 0 through 25 to 0. After the transfer, the instruction sets the ION flag according to the function specified by *[f]*.

NOTES: *The assembler recognizes the special mnemonic* **INTA** ac *to be equivalent to* **DIB** ac,CPU.

Do not use the **DIBP** *ac*,**CPU** *instruction on a 32-bit processor. The instruction is reserved for (and used as a VCT instruction) on the ECLIPSE C/350.*

The format of the specified accumulator after instruction execution is:

0 - 0		Device Code	
0	25	26	31

Bits	Name	Contents or Function
0-25	0-0	
26-31	Device Code	6-bit device code of highest priority interrupting device.

Reset
IORST
DIC*[f]* *ac*,CPU

0	1	1	ac		1	0	1	f		1	1	1	1	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The *I/O Reset* instruction sends a reset signal to all devices to clear their states. The instruction sets the 16-bit priority mask to 0, disables logical address translation, sets the PSR to 0, sets bits 0 through 9 of FPSR to 0, sets bits in the IOC status register, and sets the ION flag according to the function specified by *[f]*.

If you use the standard form (*DIC[f] ac,CPU*), you must code an accumulator to avoid assembly errors. During execution, the processor ignores the accumulator field and the contents of the accumulator remain unchanged.

NOTE: *The assembler recognizes the special mnemonic* **IORST** *to be equivalent to* **DICC 0,CPU**. *This instruction sets the BUSY and DONE flags as described above and sets the ION flag to 0.*

Mask Out
MSKO *ac*
DOB*[f]* *ac*,CPU

0	1	1	ac		1	0	0	f		1	1	1	1	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The *Mask Out* instruction places the contents of bits 16 through 31 of the specified accumulator in the priority mask. After the transfer, the instruction sets the ION flag according to the function specified by *[f]*. The contents of the specified accumulator remain unchanged. A 1 in a bit position disables interrupt requests for devices that use that bit as a mask.

NOTES: *Masking out a device when interrupts are enabled is not recommended.*

The assembler recognizes the special mnemonic **MSKO** ac *to be equivalent to* **DOB** ac,CPU.

The contents of the specified accumulator is:

Unused		Priority Mask	
0	15	16	31

Halt
HALT
DOC*[f]* *ac*,CPU

0	1	1	ac		1	1	0	f		1	1	1	1	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The *Halt* instruction sets the ION flag according to the function specified by *[f]*, and then stops the processor.

NOTE: *The assembler recognizes the special mnemonic* **HALT** *to be equivalent to* **DOC 0,CPU**.

Interrupt Disable
INTDS
NIOC CPU

0	1	1	0	0	0	0	0	1	0	1	1	1	1	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The *Interrupt Disable* instruction sets the ION flag to 0.

Interrupt Enable
INTEN
NIOS CPU

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1

The *Interrupt Enable* instruction sets the ION flag to 1.

If the instruction changes the state of the ION flag, the CPU allows one more instruction to execute before the first I/O interrupt can occur. However, if the instruction is interruptible, then interrupts can occur as soon as the instruction begins to execute.

CPU Skip
SKP*t* CPU

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	0	0	1	1	1	t		1	1	1	1	1	1

The *CPU Skip* instruction tests the specified flag. If the test condition is true, the processor skips the next sequential word. (Table 7.3 lists the possible test conditions.)

Programmable Interval Timer

Device Code
43_8

Assembler Mnemonic
PIT

Priority Mask Bit
6

The programmable interval timer (PIT) is a CPU-independent time base that you can set to initiate program interrupts at fixed intervals ranging from 100 microseconds to 6.5536 seconds in increments of 100 microseconds. It can also be sampled with I/O instructions at any point in its cycle to determine the time until the next interrupt. You use the PIT in multiprogram operating systems to allocate CPU time to different programs on a "time-slice" basis.

The PIT consists of a 16-bit initial count register and a 16-bit counter. During operation, the processor loads the PIT counter with the contents of the initial count register. The processor then increments the counter at 100-microsecond intervals until the count reaches 177777_8. The PIT then initiates a program interrupt request. At the end of the next 100-microsecond interval, the processor again loads the PIT counter with the contents of the initial count register and the counting process is repeated. A BUSY flag and a DONE flag control the operation of the device.

In order to obtain a particular time interval between program interrupt requests, load the two's complement of the number of 100-microsecond intervals between interrupt requests into the initial count register. When you first start the PIT, the processor immediately loads the count into the counter. At the first 100-microsecond pulse, the processor again loads the count into the counter. This is done to synchronize the program and the counter.

Device Flags

Device flag commands to the PIT determine the starting or stopping of the counting cycle for program interrupts.

f=S Sets the BUSY flag to 1 and the DONE flag and interrupt request flag to 0; begins the counting cycle.

f=C Sets the BUSY and DONE flags and the interrupt request flag to 0; stops the counting cycle.

f=P No effect.

PIT Instructions

Table 7.5 lists the I/O instructions that affect the PIT device.

Assembler Statement	Operation
DIA*[f] ac*,PIT	Reads the counter value into the accumulator
DOA*[f] ac*,PIT	Loads the counter with a value (PIT initializes the counter with the value each time the counter starts or overflows)
IORST	Stops the counting cycle and sets the BUSY and DONE flags, the interrupt mask bit 6 and the counter to 0

Table 7.5 I/O instructions for PIT

Read Count
DIA*[f]* *ac*,PIT

0	1	1	ac		0	0	1	f		1	0	0	0	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The *Read Count* instruction places the value of the PIT counter in bits 16 through 31 of the specified accumulator, destroying the accumulator's previous contents. After the data transfer, the instruction performs the function specified by *[f]*. The format of the specified accumulator is as follows:

Reserved	Count
0 … 19	20 … 31

Bits	Name	Contents or Function
0-19	Reserved	Reserved for future use
20-31	Count	Current value of the PIT counter within one count cycle (two's complement)

Specify Initial Count
DOA*[f]* *ac*,PIT

0	1	1	ac	0	1	0	f	1	0	0	0	1	1
0	1	2	3 4	5	6	7	8 9	10	11	12	13	14	15

The *Specify Initial Count* instruction loads bits 16 through 31 of the specified accumulator into the PIT's initial count register. After the data transfer, the instruction performs the function specified by *[f]*. The contents of the specified accumulator remain unchanged. The format of the accumulator is as follows:

Reserved	Initial Count
0 … 19	20 … 31

Bits	Name	Contents or Function
0-19	Reserved	Reserved for future use
20-31	Initial Count	The number of 100-microsecond intervals between interrupts (two's complement)

Real-Time Clock

Device Code
14_8

Assembler Mnemonic
RTC

Priority Mask Bit
13

The real-time clock (RTC) generates low-frequency I/O interrupts for performing time calculations independent of CPU timing. You can use these interrupts as a time base in programs that require it. The frequency of the clock is program selectable to ac-line frequency, 10, 100, and 1000 Hz. Both a BUSY and a DONE flag control the operation of the device.

Once you start the RTC, the first program interrupt request can come at any time up to the selected clock period. After the first interrupt has occurred, succeeding interrupts come at the clock frequency, provided that the program always sets the BUSY flag to 1 before the clock period expires. After power up or the issuance of an **IORST** instruction, the processor sets the clock to the line frequency. After power up, the line frequency pulses are available immediately, but 5 seconds must elapse before a steady pulse train is available from the clock for other frequencies.

Device Flags

Device flag commands to the RTC determine the enabling or disabling of RTC interrupts.

f=S	Sets the BUSY flag to 1, and the DONE flag and interrupt request flag to 0; enables RTC interrupts.
f=C	Sets the BUSY and DONE flags and the interrupt request flag to 0; disables RTC interrupts.
f=P	No effect.

RTC Instructions

Table 7.6 lists the I/O instructions that affect the RTC device.

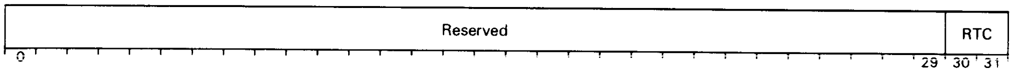
Assembler Statement	Operation
DOA*[f] ac*,**RTC**	Selects a clock frequency with a value from an accumulator
IORST	Disables RTC interrupts and selects the ac-line frequency; also, sets the BUSY and DONE flags and the interrupt mask bit 13 to 0

Table 7.6 I/O instructions for RTC

Select RTC Frequency
DOA*[f] ac*,RTC

0	1	1	ac		0	1	0	f		0	0	1	1	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The *Select RTC Frequency* instruction sets the clock frequency according to bits 30 and 31 of the specified accumulator. The contents of the specified accumulator remain unchanged with bits 0 through 29 ignored. The format of the specified accumulator is as follows:



Bits	Name	Contents or Function
0-29	Reserved	Reserved for future use (set to 0)
30-31	RTC	Selects the clock frequency as follows: **Bits** **Frequency Selected** 00 ac-line frequency 01 10 Hz 10 100 Hz 11 1000 Hz

Primary Asynchronous Line Input/Output

INPUT Device Code	**OUTPUT Device Code**
10_8	11_8
Assembler Mnemonic	**Assembler Mnemonic**
TTI	TTO
Priority Mask Bit	**Priority Mask Bit**
14	15

The asynchronous line controller (ALC) is the communication link between the processor and the master terminal. It supports asynchronous communication at selected rates from 110 to 9600 baud in 7-bit codes with program-generated parity, or 8-bit codes with no parity. You can use one or two stop bits with either format.

Because the asynchronous communications input and output can generate program interrupts independently, each has its own device code and is controlled by its own set of BUSY and DONE flags. The ALC is program-compatible with Data General's Model 4010 controller.

The ALC is set up to transmit and receive 8-bit characters without parity checking. A process may send or receive 7-bit characters with even, odd, or mark parity by using the high-order bit in the 8-bit character (bit 8 in the accumulator) as a parity bit. On transmission, the program that drives the ALC calculates and inserts the correct parity bit. On reception, the program calculates and checks parity on the received character.

There are timing constraints on the *receive* portion of the controller. As the ALC receives each character, it places the character in an input character buffer, sets the DONE flag to 1, and the BUSY flag to 0. If the program controlling the receiver does not transfer the character before receiving the next character, the contents of the input character buffer are overwritten and the previous character is lost. Typically, the intercharacter time at 110 baud is 100 milliseconds, and at 4800 baud the intercharacter time is approximately 2.08 milliseconds.

Device Flags

Device flag commands to the TTI/TTO determine the flag settings and the transmission of an output character.

f=S Sets the BUSY flag to 1 and the DONE flag to 0. When the S flag is used with the TTO device, the ALC transfers the character from the output buffer to the shifter and begins transmission of the character. The ALC sets the BUSY flag to 0 and the DONE flag to 1 when the character passes from the output buffer to the shifter.

f=C Sets the BUSY and DONE flags and the interrupt request flag to 0.

f=P No effect.

TTI/TTO Instructions

Table 7.7 lists the I/O instructions that affect the TTI/TTO device.

Assembler Statement	Operation
DIA*[f] ac*,**TTI**	Reads a character from the device into an accumulator
DOA*[f] ac*,**TTO**	Sends a character from an accumulator to the device
IORST	Sets the BUSY and DONE flags and the interrupt mask bit 14 and 15 to 0

Table 7.7 I/O instructions for TTI and TTO

Read Character Buffer
DIA*[f]* *ac*,**TTI**

0	1	1	ac		0	0	1	f		0	0	1	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The *Read Character Buffer* instruction places the contents of the controller's input buffer in bits 24 through 31 of the specified accumulator. After the data transfer, the instruction sets the controller's BUSY and DONE flags according to the function specified by *[f]*. The format of the specified accumulator is as follows:

Reserved		Character	
0	23	24	31

Bits	Name	Contents or Function
0-23	Reserved	Reserved for future use
24-31	Character	The 8-bit character (or 7-bit character with parity in bit position 8) read from the input buffer

Load Character Buffer
DOA*[f]* *ac*,TTO

0	1	1	ac	0	1	1	f	0	0	1	0	0	1
0	1	2	3 4	5	6	7	8 9	10	11	12	13	14	15

The *Load Character Buffer* instruction loads bits 24 through 31 of the specified accumulator into the controller's output buffer. After the data transfer, the instruction sets the controller's BUSY and DONE flags according to the function specified by *[f]*. The contents of the specified accumulator remain unchanged. The format of the specified accumulator is as follows:

Reserved	Character
0 23	24 31

Bits	Name	Contents or Function
0-23	Reserved	Reserved for future use
24-31	Character	The 8-bit character (or 7-bit character with parity in bit position 8) to be placed in the output buffer

System Control Program

Device Code
45_8

Assembler Mnemonic
SCP

Priority Mask Bit
15

The SCP, as described in the Technical Summary chapter, is a microcoded soft system console within the MV/4000 computer. Through the MV/4000 microcode, the SCP isolates hardware problems.

Device Flags

Device flag commands to the SCP determine the settings of the BUSY and DONE flags.

f=S The SCP BUSY flag is never set for the MV/4000 because each operation completes within the instruction execution.

f=C Sets the DONE flag to 0.

f=P No effect.

NOTE: *For compatibility purposes with other MV/Family processors, the SCP instruction descriptions include the BUSY flag conditionals.*

SCP Instructions

Table 7.8 lists the instructions that provide the CPU communication with the SCP.

Mnemonic	Name	Action
DOBS *ac*,**SCP**	Enable/Disable Error Reporting	Enables/disables CPU error reporting, and performs indicated command
DIBC *ac*,**SCP**	Return SCP Status	Returns the current status of the SCP
SKP*t* **SCP**	Skip Test	Tests the SCP BUSY/DONE flag and skips next instruction if true
NIOC SCP	Clear SCP DONE Flag	Clears SCP DONE flag, but the SCP remains in diagnostic mode
IORST	I/O Reset	Disables CPU error reporting or clears diagnostic mode and clears the flags

Table 7.8 SCP instructions

The **SKP, NIOC,** and **IORST** instructions are described earlier in this chapter. Note that the **NIOC SCP** instruction clears the SCP DONE flag, but does not take the SCP out of diagnostic mode.

Before issuing a **DOBS SCP** instruction, the process should check the SCP BUSY flag. If the BUSY flag is 0, the SCP is ready to accept the next **DOBS SCP** instruction.

Enable/Disable Error Reporting
DOBS *ac*,SCP

0	1	1	ac		1	0	0	0	1	1	0	0	1	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The *Enable/Disable Error Reporting* instruction sets the SCP BUSY flag and uses the contents of the specified accumulator to enable or disable CPU error reporting and to perform the command contained in the command field. Following is the accumulator format:

Reserved		E	Command		Interface Block	
0	15	16	17	23	24	31

Bits	Name	Contents or Function
0-15	Reserved	These bits are reserved for future use
16	E	The E flag enables the SCP error reporting 1 = enable; 0 = disable
17-23	Command	The SCP performs the function defined by these bits **Command (octal)** **Name** 000 No-op 001 Undefined 002 Select SCP Power Down mode 003 Disable SCP Power Down mode 004 Set block 005 Enable All ERCC 006 Undefined 007 Mask Soft ERCC 010 Mask All Sniff 011 Undefined thru 176 Undefined 177 Enter Diagnostic Sequence
24-31	Interface Block	Depending on the command, the CPU or SCP uses bits 24 through 31 as a physical address pointer to a multiword block in page zero

The E flag (or enable command) enables CPU error reporting. When the CPU or SCP wishes to report an error, it will use the page zero address specified by the last set block command as a pointer to a double-word physical address. This address will, in turn, point to a 16-word block that the CPU or SCP may use to report error data. The first word of the block will receive the error code. The remaining 15 words are available for reporting extended error status.

If the SCP should interrupt the CPU, the SCP disables error reporting until the process issues a new enable command.

For instance, under a Data General operating system, the CPU uses the first word of the error block as the SYSLOG code number. Any error that requires extended error status will also cause the entire 16-word block (including the code number) to be logged as the data area of the SYSLOG entry.

The command definitions are as follows:

- Select SCP Power-Down Mode (command 002_8)

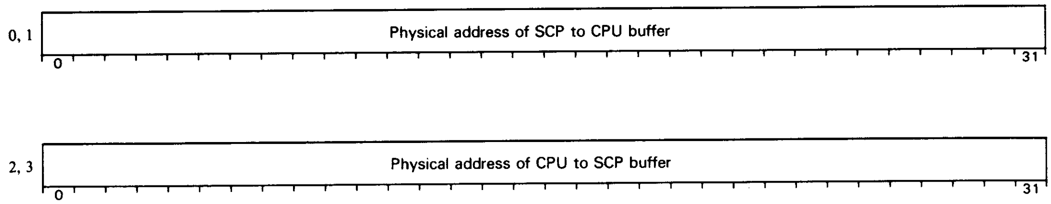
  The SCP does not use the page zero address entered with this command. The command places the SCP in the power-down mode with power fails reported as maskable SCP interrupts.

- Disable SCP Power-Down Mode (command 003_8)

  The command removes the SCP from the power-down mode. The SCP no longer intercepts powerfail interrupts.

- Set block (command 004_8)

  The command specifies to the SCP the address of the SCP/CPU interface block. The address points to a four-word block in page zero. The format of the four-word block is as follows:


0, 1
Physical address of SCP to CPU buffer
0
31
2, 3
Physical address of CPU to SCP buffer
0
31

NOTE: *The SCP restricts word 0 of the four-word block to be in the range of 0 to 377_8.*

A command that requires a CPU/SCP interface block (16-word block) specifies it with the command.

- Enable All ERCC Error Reporting (command 005_8)

  The command enables the SCP to detect and report any memory error.

– Single-bit --	1-bit ERCC error detected during memory read
– Multibit --	2-bit (or more) ERCC error detected during memory read
– Soft-sniff --	1-bit ERCC error detected during memory refresh
– Hard-sniff --	2-bit (or more) ERCC error detected during memory refresh

- Mask Soft ERCC Error Reporting (command 007_8)

  The command disables all of memory from single-bit, soft-sniff, and hard-sniff error reporting; detection and correction remain enabled. The processor reports multibit memory errors.

- Mask All Sniff Error Reporting (command 010_8)

  The command disables all of memory from soft-sniff and hard-sniff error reporting; detection and correction remain enabled. The processor reports single-bit and multibit memory errors.

- Enter Diagnostic Sequence (command 177_8)

  Disable CPU error reporting. The SCP does not use the page zero address entered with this **DOBS SCP** instruction. The SCP uses the previous page zero address as a pointer to the SCP/CPU interface block. The SCP clears its BUSY flag. The SCP remains in diagnostic mode until either a console reset occurs or the process issues another **DOBS SCP** instruction.

  When the process issues the **DOBS SCP** instruction, the SCP first places the contents of bits 16 through 31 of the specified accumulator into word 0 of the *SCP to CPU* buffer. The SCP then reads words 1 through 7 from the *CPU to SCP* buffer, inverts them, and writes them back to their respective locations in the *SCP to CPU* buffer. Upon completion, the SCP transmits a status 0 to the host, sets the DONE flag, and interrupts the CPU.

  NOTE: *The* **NIOC SCP** *will clear the DONE flag, but will not take the SCP out of diagnostic mode.*

Return SCP Status
DIBC *ac*,SCP

0	1	1	ac	0	1	1	1	0	1	0	0	1	0	1
0	1	2	3 4	5	6	7	8	9	10	11	12	13	14	15

> **NOTE:** *The* **DIBC** *ac,* **SCP** *and the* **DIB** *ac,* **SCP** *instructions are equivalent. The* **DIBS** *ac,* **SCP** *instruction is a no-op.*

The *Return SCP Status* instruction clears the SCP DONE flag and returns a code to the specified accumulator denoting the current status of the SCP. Following is the accumulator format:

Reserved	Status
0 15	16 31

Bits	Name	Contents or Function
0-15	Reserved	These bits are reserved for future use
16-31	Status	The codes returned to these bits denote the current status of the SCP as follows: **Code** **Meaning** 000000 Error information is in current error block The CPU error status codes and their definitions returned to the first word (word 0) of the error block are: **Status (octal)** **Definition** 007 Power fail detected 053 Single-bit or soft-sniff ERCC detected 054 Multibit ERCC detected 055 Hard-sniff ERCC detected Extended error status is used for ERCC as follows: **Word** **Contents** 0 Status (53, 54, 55) 1 Bit 13 = other Bit 15 = sniff 2 Physical page number 3 Double-word on module 4 Syndrome bits

Bits	Name	Contents or Function
		000001 SCP reset; the SCP is reset and must be reinitialized with the *DOBS ac,SCP* instruction and a command 4. 000002 Request acknowledge for any undefined command. 000003 SCP-requested function is in error; The SCP reports an unknown error with this code. For instance, if a required SCP/HOST interface block has not been defined, or if an undefined function request is made, or if invalid data is passed to the SCP (through the HOST to SCP buffer), the SCP issues this code. 177777 SCP is in diagnostic sequence.

Data Channel/Burst Multiplexor Channel

The data channel (DCH) provides I/O communication for medium-speed devices and synchronous communications. The burst multiplexor channel (BMC) is a high-speed communications pathway that transfers data directly between main memory and high-speed peripherals. I/O-to-memory transfers for both DCH and BMC always bypass the address translator.

A map controls a DCH or BMC. This map is a series of contiguous map slots, each of which contains a pair of map registers (one even-numbered register and its corresponding odd-numbered register).

DCH/BMC Maps

The MV/4000 computer supports 512 DCH device map slots and 1024 BMC device map slots. The DCH or BMC sends to the processor a logical address with each data transfer. The processor translates the logical address to a physical address using the appropriate map slot for that address.

The device controller performing the data transfer controls the BMC. No program control or CPU interaction is required except to set up the BMC's map table.

NOTE: *Since the MV/4000 computer performs the BMC or DCH operations in the MV/4000 microcode, CPU operations must halt while the BMC or DCH transfers data.*

BMC Address Modes

The BMC operates in either the unmapped (physical) mode or the mapped (logical) mode.

In the unmapped mode, the BMC receives 21-bit addresses from the device controllers, and passes them directly to memory. As the BMC transfers each data word to or from memory, it increments the destination address, causing successive words to move to or from consecutive locations in memory.

If the controller specifies the mapped mode for a data transfer, a 20-bit address is used. The high-order 10 bits of the logical address form a logical page number, which the BMC map translates into a 12-bit physical page number. This page number, combined with the 10 low-order bits from the logical address, forms a 22-bit physical address, which the BMC uses to access memory.

BMC Map

The BMC uses its own map to translate logical page numbers to physical ones. (The SSPT instruction defines the memory locations of the BMC map.) The map contains 1024 map registers, the odd-numbered registers each containing a 10-bit physical page number. The BMC uses the logical page number as an index into the map, and the contents of the selected map register become the high-order 10 bits of the physical address.

Note that when the BMC performs a mapped transfer, it increments the destination address after it moves each data word. If the increment causes an overflow out of the 10 low-order bits, this selects a new map register for subsequent address translation. Depending on the contents of the map table, this could mean that the BMC may not transfer successive words to or from consecutive physical pages in memory.

NOTE: *For each BMC device, the MV/4000 computer contains a one-address translation cache.*

DCH/BMC Registers

The MV/4000 system contains 512 DCH map registers and 1024 BMC map registers. The map registers are numbered from 0 through 7777_8, as explained in Table 7.9 and depicted in Figure 7.1.

Registers (Octal)	Description
0000-3776	Even-numbered registers are the most significant half of the BMC map positions 0-1777
0001-3777	Odd-numbered registers are the least significant half of the BMC map positions 0-1777
4000-5776	Even-numbered registers are the most significant half of the DCH map positions 0-777
4001-5777	Odd-numbered registers are the least significant half of the DCH map positions 0-777
6000	Port Definition Register
6001-7677	Reserved
7700	Port Status Register
7701-7777	Reserved

Table 7.9 Device map registers 0000-7777

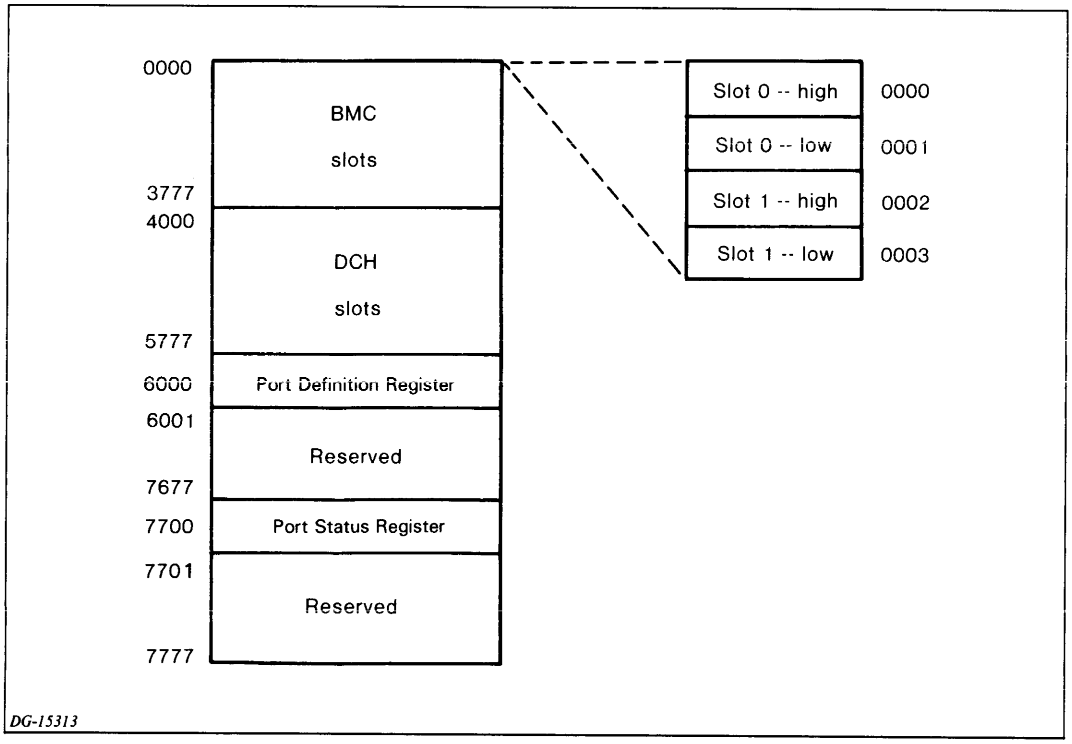
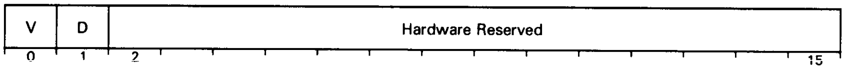


Figure 7.1 DCH/BMC registers

The device map registers and their formats follow:

BMC/DCH Even-Numbered Register Formats

The processor translates the contents of the BMC and DCH even address registers ($0000\text{-}3776_8$, $4000\text{-}5776_8$ respectively) as:



Bits	Name	Contents or Function
0	V	Validity bit; if 1, the processor denies access
1	D	Data bit If 0, the channel transfers data If 1, the channel transfers zeros
2-15	Hardware Reserved	Undefined when read

BMC/DCH Odd-Numbered Register Formats

The processor translates the contents of the BMC and DCH odd address registers (0000-3777_8, 4001-5777_8 respectively) as:

Reserved	Physical Page Number
0 3	4 15

Bits	Name	Contents or Function
0-3	Res	Hardware reserved; reading these bits returns an undefined state
4-15	Physical Page Number	Physical page number associated with the logical page that referred to that particular slot

DCH/BMC Port Definition Register (6000_8)

E	Reserve	BV	DV	Reserve	A	P	Reserved	M	1
0	1 2	3	4	5 6	7	8	9 13	14	15

Bits	Name	Contents or Function
0	E	Error flag; if 1, an error has occurred on the I/O port (0 only when all other error bits are 0)
1,2	Reserve	Bits 1 and 2 are reserved for future use and returned as zero
3	BV	BMC validity error flag; if 1, BMC validity protect error has occurred
4	DV	DCH validity error flag; if 1, DCH validity protect error has occurred
5,6	Reserve	Bits 5 and 6 are reserved for future use and returned as zero
7	A	BMC address error; if 1, the channel has detected an address parity error
8	P	BMC data error; if 1, the channel has detected a data parity error
9-13	Reserved	Bits 9 through 13 are reserved for future use and returned as zero
14	M	DCH mode; if 1, DCH mapping is enabled
15	1	Always set to 1

NOTES: *Setting bit 3, 4, 7, or 8 to a one with the CIO instruction complements these bits.*

The C/350 **IORST** *instruction clears bits 0, 3, 4, 7, 8, and 14.*

Port Status Register Format

The read-only port status register (7700_8) provides status information. The format for the register follows:

DCH/BMC Port Status Register (7700_8)

ERR	Reserved	1	1	Res	INT
0	1 11	12	13	14	15

Bits	Name	Contents or Function
0	ERR	If 1, the port has detected an error indicated by the port definition register
1-11	Reserved	Bits 1 through 11 are reserved for future use
12-13	1,1	Always set to 1
14	Reserved	Bit 14 is reserved for future use and returned as zero
15	INT	Interrupt pending if 1

DCH/BMC Map Instructions

The **CIO**, **CIOI**, or **WLMP** instruction control DCH/BMC map loads and reads. Table 7.10 lists the instructions that affect the DCH and BMC maps.

Assembler Statement	Operation
WLMP	Loads BMC/DCH map slots from memory
CIO, CIOI	Returns BMC/DCH status or loads map registers (1/2 slot) from accumulators
IORST	Clears bits 0, 3, 4, 7, 8, and 14 of the port definition register

Table 7.10 DCH/BMC map instructions

The **CIO**, **CIOI**, and **WLMP** instructions are described in the *Principles of Operation, 32-Bit ECLIPSE® Systems* manual. The **IORST** instruction is presented earlier in this chapter.

Wide Load Map
WLMP

1	0	1	0	0	1	1	1	1	1	1	1	1	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The **WLMP** instruction in conjunction with three accumulators loads successive double words from memory into successive DCH or BMC map slots.

The double word contained in AC0 refers to the first map slot in the specified I/O channel that the **WLMP** instruction will load. AC1 contains a 16-bit unsigned count of the number of map slots in the I/O channel to be loaded. AC2 contains the effective address of the first double word to be loaded into the referenced I/O channel slots.

For each map slot loaded:

AC0 is incremented by one;
AC1 is decremented by one;
AC2 is incremented by two.

Upon completion of the **WLMP** instruction:

AC0 references the map slot following the last slot loaded;
AC1 contains a 0 in the 16 least significant bits;
AC2 contains the address of the word following the last double word loaded.

NOTE: *If AC1 is initially 0, a no-op is performed.*

The accumulator formats for the **WLMP** instruction are as follows:

AC0 contains a double word:

Reserved	0 — 0	Map Slot Number
0 – 17	18 – 20	21 – 31

Bits	Name	Contents or Function
0-17	Reserved	Bits 0 through 17 are reserved for future use
18-20	0 - 0	Must be 0
21-31	Map Slot Number	Indicates which map slot the instruction will load **Number** **Meaning** $0\text{-}1777_8$ Loads a BMC slot $2000\text{-}2777_8$ Loads a DCH slot

AC1 contains a 16-bit unsigned count:

Ignored	Number of Map Slots
0 – 15	16 – 31

Bits	Name	Contents or Function
0-15	Ignored	Bits 0 through 15 are ignored by the **WLMP** instruction
16-31	Number of Map Slots	Unsigned count of the number of map slots, which the **WLMP** instruction will load

AC2 contains an effective address that refers to the first double word that the **WLMP** instruction will load.

The contents of these double words are in the following format:

V	D	Reserved	Physical Page Number
0	1	2 – 19	20 – 31

Bits	Name	Contents or Function
0	V	Valid; set to 0 implies valid; set to 1 implies access denied
1	D	Data; set to 0 implies transfer data; set to 1 implies transfer zeros
2-19	Reserved	Bits 2 through 19 are reserved and must be set to 0
20-31	Physical Page Number	The translation for the logical map slot

The effect of the setting of the V and D bits and the direction of the transfer are:

V	D	Transfer Direction	Action
0	0	From I/O Port	Transfer data
0	1	From I/O Port	Transfer 0s from either DCH or BMC device
1	-	From I/O Port	Transfer aborted -- flag error
0	0	To I/O Port	Transfer data
0	1	To I/O Port	Transfer 0s to memory
1	-	To I/O Port	Transfer aborted -- flag error

NOTE: *From I/O Port implies memory to device; To I/O Port implies device to memory.*

Upon detection of an invalid map entry due to an active device:

For the BMC -- The active BMC requesting device is flagged.
For the DCH -- Bit 4 of the Port Definition Register is set to 1.

WLMP is a privileged and interruptible instruction.

Universal Power Supply Controller

Device Code
4_8

Assembler Mnemonic
UPSC

Priority Mask Bit
13

The universal power supply controller (UPSC) is a daughter board containing a microprocessor. The UPSC performs a power-up diagnostic self test, monitors the system power, and reports failures, problems, and status to the MV/4000 computer.

The UPSC monitors: problems on the power supplies (such as, over-temperature and over-current), AC over-voltage or under-voltage, reed switches for sensing overload on +5V (or the power switch was turned off), battery backup fault, and fan failure.

Device Flags

Device flag commands to the UPSC determine the enabling or disabling of UPSC interrupts.

f=S	Sets the BUSY flag to 1, and the DONE flag to 0.
f=C	Sets the BUSY and DONE flags to 0
f=P	No effect.

UPSC Instructions

Table 7.11 lists the I/O instructions that affect the UPSC device.

Assembler Statement	Operation
DOAS *ac*,**UPSC**	Write data to UPSC
DOAP *ac*,**UPSC**	Request data from USPC
DIA[*f*] *ac*,**UPSC**	Read data from UPSC
IORST	Clears BUSY and DONE flags and interrupt mask bit 13

Table 7.11 I/O instructions for UPSC

Write Data to UPSC
DOAS[*f*] *ac*,UPSC

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	ac		0	1	0	0	1	0	0	0	1	0	0

When the MV/4000 computer issues the DOAS, the UPSC sets the BUSY flag. The UPSC resets the BUSY flag and sets the DONE flag when the UPSC completes the operation.

The *Write data to UPSC* instruction sends the contents of the accumulator to the UPSC. The four registers that can be written on the UPSC are defined as follows:

Register	Name	Contents or Function
0	Control register	Selects reporting mode, power margining, and enable/disable battery backup
1	Power margining register	When the backpanel is jumpered for margining or margining is selected using the control register, the +5V logic, +5V memory, -5V memory, and +12V memory voltages can be increased or decreased
2	Reserved	Reserved for future use
3	Diagnostic test register	Verify the data path between the MV/4000 computer and UPSC or enable battery test

The bit descriptions in the following tables explain the bit function when a bit is set.

Register 0
Control Register

Undefined	0	0	Res	BT	ALT	COMM	BBU	PFM
0 – 7	8	9	10	11	12	13	14	15

Bits	Name	Contents or Function
0-7	Undefined	Bits 0 through 7 are undefined
8,9	Register 0	The control register bits 8 and 9 equal zero
10	Res	Bit 10 is reserved for future use
11	BT	Remove AC power to allow battery testing
12	ALT	Mask out powerfail interrupts. When ALT is 1, powerfail skips (SKPDN and SKPDZ) will always behave as if there is no powerfail
13	COMM	UPSC can interrupt MV/4000 when a fault occurs
14	BBU	Disables the battery backup unit
15	PFM	Enable power margining

Register 1
Power Margining Register

Undefined	0	1	+5LI	+5LD	ALLI	ALLD	+5MI	+5MD
0 – 7	8	9	10	11	12	13	14	15

A voltage is in the nominal state when the corresponding bit is 0. The voltage is margined when the corresponding bit is 1 and the MV/4000 computer is jumpered or programmed for margining.

All percentages are additive. For instance, when bits 12 and 15 are used together, the voltage for +5V memory increases approximately 5 percent; while the -5V memory and +12V memory increase approximately 8 percent.

Bits	Name	Contents or Function
0-7	Undefined	Bits 0 through 7 are undefined
8,9	Register 1	The power margining register bits 8 and 9 equal 01_2
10	+5LI	Increase +5V logic approximately 2.2%
11	+5LD	Decrease +5V logic approximately 5%
12	ALLI	Increase +5V memory, -5V memory, and +12V memory voltages approximately 8%
13	ALLD	Decrease +5V memory, -5V memory, and +12V memory voltages approximately 8%
14	+5MI	Increase +5V memory approximately 2.2%
15	+5MD	Decrease +5V memory approximately 2.2%

Register 3
Diagnostic Test Register

0	0	0	0	0	0	0	0	1	1	0	0	0	0	BTE	COMP
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The UPSC performs the battery test or bit test as specified by bits 14 and 15 of the accumulator. To complete the command, the UPSC requires a second **DOAS** *ac*,**UPSC** instruction.

If UPSC fails to detect the second DOAS instruction, the UPSC will automatically exit the diagnostic test. The UPSC indicates a timeout by setting the DONE flag and the appropriate fault code in the fault code register.

Bits	Name	Contents or Function
0-7	0 — 0	Bits 0 through 7 are reserved and must be zero
8,9	Register 3	The diagnostic test register bits equal 11_2
10-13	0 — 0	Bits 10 through 13 are reserved and must be zero
14	BTE	Battery Test Enable. If the accumulator contains 2_8, the battery test is enabled. You initiate the test with the second DOAS to bit 11 of register 0 (BT) **NOTE:** *The BTE bit must be set before the BT bit.*
15	COMP	Complement. If the accumulator contains 0_8 or 1_8, the UPSC reads the data from the second DOAS (A buffer), complements it if COMP is 1, and then returns the data to the A buffer. The A buffer can then be read with the DIA instruction

Request Data from UPSC
DOAP*[f]* *ac*,UPSC

0	1	1	ac		0	1	0	1	1	0	0	0	1	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The *Request Data From UPSC* instruction uses bits 13 through 15 of the accumulator to request specific information from the UPSC.

0	0	0	0	0	0	0	0	0	0	0	0	0	octal value		
0	1	2	3	4	5	6	7	8	9	10	11	12	13		15

Bits	Name	Contents or Function
0-12	Reserved	
13-15	Octal value	
	0	Read control bits
	1	Read battery backup and margining bits
	2	Read power supply system status
	3	Read fault code register
	4	Read UPSC code revision number

Read data from UPSC
DIA*[f]* *ac*,UPSC

0	1	1	ac		0	0	1	0	0	0	0	0	1	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The *Read Data From UPSC* instruction loads the data from the UPSC A Buffer into the accumulator. The previous *Request Data from UPSC* instruction defines the data read from the A Buffer.

Read Control Bits

0	0	0	0	0	0	0	0	0	0	0	0	ALT	COMM	BBU	PFM
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents or Function
0-11	Reserved	Returned as zero
12	ALT	Power fail is masked out
13	COMM	UPSC can interrupt the MV/4000 computer when a fault occurs
14	BBU	The battery backup unit is disabled
15	PFM	Power margining is enabled

Read Battery Backup and Margining Bits

0	0	0	0	0	0	0	0	BAT	0	+5LI	+5LD	ALLI	ALLD	+5MI	+5MD
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents or Function
0-7	Reserved	
8	BAT	The battery backup is connected and in use. (The bit is cleared if system is not running on batteries, a battery fault occurs, or the BBU flag is set)
9	Reserved	
10	+5LI	Increase +5V logic approximately 2.2%
11	+5LD	Decrease +5V logic approximately 5%
12	ALLI	Increase +5V memory, -5V memory, and +12V memory voltages approximately 8%
13	ALLD	Decrease +5V memory, -5V memory, and +12V memory voltages approximately 8%
14	+5MI	Increase +5V memory approximately 2.2%
15	+5MD	Decrease +5V memory approximately 2.2%

Read Power Supply System Status

0	0	0	0	0	0	0	0	0	0	0	0	PART	FULL	RUN	CHAR
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Bits	Name	Contents or Function
0-11	Reserved	
12	PART	The system is equipped with partial battery backup
13	FULL	The system is equipped with full battery backup
14	RUN	The system is running on the batteries
15	CHAR	The batteries are recharging

Read Fault Code Register

0	0	0	0	0	0	0	0	0	Fault Code	Fault Category
0	1	2	3	4	5	6	7	8	9 12	13 15

Bits	Name	Contents or Function
0-8	Reserved	
9-12	Fault Code	Specifies the fault code for a specific fault category (Table 7.12)
12-15	Fault Category	Specifies the fault categories (in a range of 0 through 7)

The UPSC power system displays a fault category by flashing the MV/4000 front panel LEDs. The power system also loads the fault code into the fault code register. Table 7.12 lists the fault codes by fault category.

NOTE: *Codes not shown are unused.*

Fault Category and Fault Code Bits 9 - 15$_8$	Operation
Category 0	System off or no fault or UPSC software fault
000	System off or no fault
170	Diagnostic mode timeout
Category 1	Environment Fault
011	VNR under-voltage (Voltage Nonregulated Unit)
021	VNR over-voltage
031	Power supply over-temperature
041	Chassis over temperature
Category 2	Fan Failure
002	Blower or multiple fan failure
012	Failure of Fan #1
022	Failure of Fan #2
032	Failure of Fan #3
042	Failure of Fan #4
052	Failure of Fan #5
062	Failure of Fan #6
072	UPSC cannot set fan signals
Category 3	VNR Fault
013	Battery backup fault indicated
Category 4	Power supply fault (includes under-voltages)
004	+5V logic under-voltage
014	+5V logic current not sharing, PS1
024	+5V logic current not sharing, PS2
034	+5V logic current not sharing, PS3
044	+5V memory under-voltage, PS1
054	+5V memory under-voltage, PS2
064	+5V memory under-voltage, PS3
074	+12V memory or +12V under-voltage, PS1
104	+12V memory or +12V under-voltage, PS2
114	+12V memory or +12V under-voltage, PS3
124	-5V memory or -5V under-voltage, PS1
134	-5V memory or -5V under-voltage, PS2
144	-5V memory or -5V under-voltage, PS3
154	under-voltage PS1, voltage unknown
161	under-voltage PS2, voltage unknown
174	under-voltage PS3, voltage unknown
Category 5	Over-voltage fault
005	Over-voltage on +5V, '+5VOV-NOT' low
045	Over-voltage on +5V memory, PS1
055	Over-voltage on +5V memory, PS2
065	Over-voltage on +5V memory, PS3
075	Over-voltage on +12V or +12V memory, PS1
105	Over-voltage on +12V or +12V memory, PS2
115	Over-voltage on +12V or +12V memory, PS3
125	Over-voltage on -5V or -5V memory, PS1
135	Over-voltage on -5V or -5V memory, PS2
145	Over-voltage on -5V or -5V memory, PS3
155	Over-voltage PS1, voltage unknown
165	Over-voltage PS2, voltage unknown
175	Over-voltage PS3, voltage unknown

Table 7.12 UPSC fault codes

(continues)

Fault Category and Fault Code Bits 9 - 15_8	**Operation**
Category 6	Over-current fault
006	Reed switch sense low, +5V output
016	Over-current on +5V, PS1
026	Over-current on +5V, PS2
036	Over-current on +5V, PS3
046	Over-current on +5V memory, PS1
156	Over-current on +5V memory, PS2
166	Over-current on +5V memory, PS3
167	Over-current on +12V or +12V memory, PS1
106	Over-current on +12V or +12V memory, PS2
116	Over-current on +12V or +12V memory, PS3
126	Over-current on -5V or -5V memory, PS1
136	Over-current on -5V or -5V memory, PS2
146	Over-current on -5V or -5V memory, PS3
156	Over-current PS1, voltage unknown
166	Over-current PS2, voltage unknown
167	Over-current PS3, voltage unknown
Category 7	UPSC fault
177	LED lamp test at power-up

Table 7.12 UPSC fault codes (concluded)

Chapter 8

Memory and System Management

This chapter describes the address translator, the memory and system management instructions, the sequence of events initiated by a privileged fault, and the reserved memory.

Address Translator

The CPU *address translator* converts the logical address of a piece of data into a physical address in memory.

To perform the translation, the address translator uses a series of *page tables* containing information about the pages of logical memory. These tables contain one entry for each page. The tables indicate whether or not the page is currently in physical memory, whether or not the page is valid (and the process can access it), and the information needed for logical-to-physical address translation.

To avoid referring to a page table for every memory reference, the address translator maintains a table of address translations and access privileges for 64 recently referenced pages. The hardware checks the address translator's table for entries before referring to a page table in memory.

Page Table Entry Format

V	M	R	W	E	Reserved		Physical Page Address	
0	1	2	3	4	5	12	13	31

Bits	Name	Contents or Function
0	V	Valid access flag. 0 indicates invalid page. 1 indicates valid page.
1	M	Memory resident page. 0 indicates disk-resident page. 1 indicates memory-resident page.
2	R	Read access flag. 0 indicates read access denied. 1 indicates read access.
3	W	Write access flag. 0 indicates write access denied. 1 indicates write access.
4	E	Execute access flag. 0 indicates execute access denied. 1 indicates execute access.
5-12	Reserved	Bits 5 through 12 are reserved for future use.
13-31	Physical Page Address	The physical address of a page in memory.

Because the memory references for a procedure tend to cluster in several pages, a needed page translation is likely to be in the address translator's table of address translations. The address translator updates the entries in this table as execution continues.

Referenced and Modified Bits

The address translator also controls two memory management bits for each page: the *modified bit,* and the *referenced bit.* The operating system uses these bits during page faults.

A page fault occurs when a process refers to a page that is not currently in physical memory. Each time a page fault occurs, the *page fault handler* must transfer a new page from disk to physical memory. This could also mean that the page fault handler might remove a page from physical memory to make room for the new page. The modified bit indicates whether or not the old page is the same as it was when it came into physical memory.

- If the modified bit for the old page is 1, it indicates that it is a modified page, and the page fault handler must save the modified page on the disk before it can bring in the new page.
- If the modified bit is 0, the copy of the old page on disk is still valid, and the page fault handler can move the new page immediately into memory.

The referenced bit helps determine which page in memory the page fault handler can replace with a new page from disk. In general, the page the processor least frequently refers to is the page replaced. The referenced bit allows the operating system to determine the frequency of references to individual pages.

Protection Validation

The address translator performs all protection system hardware checks. These checks include access validation, page validation, segment crossing validation, and others. If any of the checks fails, the address translator initiates a protection fault to the operating system. For more information about the types of protection checks, refer to the *Principles of Operation, 32-Bit ECLIPSE® Systems* manual.

Memory/System Management Instructions

Table 8.1 lists the memory/system management instructions. For further information, refer to the *Principles of Operation, 32-Bit ECLIPSE® Systems* manual. The accumulator formats for the *Load CPU Identification* and *Narrow Load CPU Identification* instructions are listed in Appendix C.

Instruction	Operation
ECLID	Load CPU identification
LCPID	Load CPU identification
LMRF	Load modified and referenced bits
LPHY	Translates logical addresses to physical addresses
LSBRA	Load all segment base registers
LSBRS	Load segment base registers 1 through 7
NCLID	Narrow load CPU identification
ORFB	OR referenced bits
PATU	Purge address translator
RRFB	Reset referenced bits
SMRF	Store modified and referenced bits
WDPOP	Pop context block (return from page fault)

Table 8.1 Memory/system management instructions

Privileged Faults

Upon detection of a privileged fault, the address translator generates either a page or protection fault. The interpretation of the validity and appropriate access bits in a page table entry, coupled with the occurrence of one of the following conditions, initiates a page fault.

- An attempt to refer to a location that is part of the logical address space, but is not part of the physical address space.
- The result of a logical address reference that requires a two-level page table, but is allocated only a one-level page table.

Page Faults

When a page fault occurs, the following actions result:

- If the current segment is not 0, the processor stores the frame pointer and stack pointer in their respective locations in page zero of the current segment, and performs a segment crossing to segment 0.
- The processor uses the contents of locations 32_8 and 33_8 of segment 0 as a base address to store a context block, (the internal state of the machine) in memory, (see Appendix D for context block structure).
- The processor initializes the segment 0 stack from page zero of segment 0.

- The processor stores the fault code in AC1.

Fault Code	Explanation
0	Multiple ERCC fault
1	Page table depth
2	Page table page fault
3	Reserved
4	Normal object reference

- The processor disables interrupts for one instruction, jumps indirect through locations 30_8 and 31_8 of segment 0, and executes the first instruction of the page fault handler.

NOTE: *If an additional page fault occurs during any of these actions, the processor halts.*

Once the page fault handler corrects the fault (e.g., brings the page into physical memory, or creates a two-level page table), the execution of a **WDPOP** instruction restarts the program. The **WDPOP** instruction restores the processor state from information contained in the context block. Figure 8.1 summarizes the actions taken upon detection of a page fault.

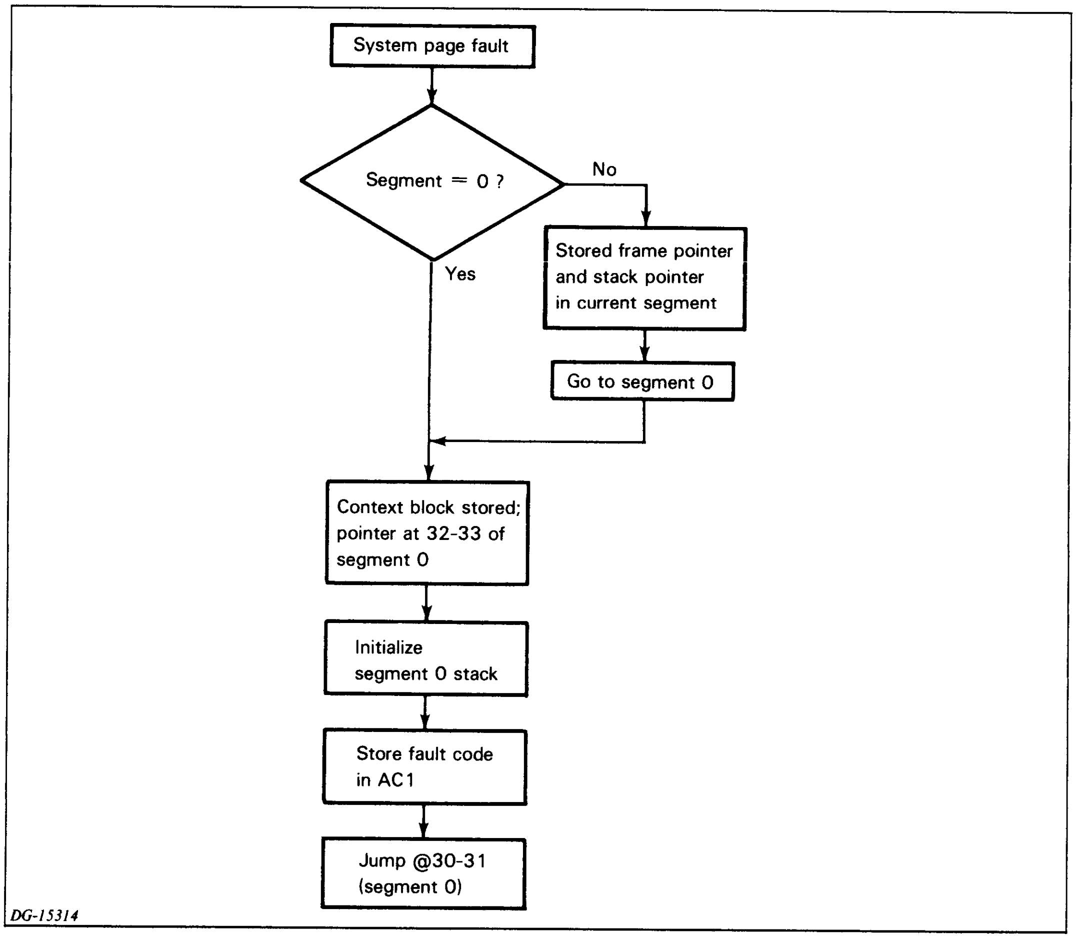

System page fault
Segment = 0 ?
No
Yes
Stored frame pointer and stack pointer in current segment
Go to segment 0
Context block stored; pointer at 32-33 of segment 0
Initialize segment 0 stack
Store fault code in AC1
Jump @30-31 (segment 0)
DG-15314

Figure 8.1 Page fault sequence

Address Protection Faults

With the address translator enabled, the following (in descending order of priority) will produce a protection violation fault:

- Privileged or I/O instruction violation.
- Defer (indirect) address violation.
- Inward reference violation.
- Segment validity violation.
- Page table validity violation.
- Read, write, or execute access violation.
- Segment crossing violation.

When a fault occurs, AC1 receives a code indicating the type of fault (refer to Appendix F). The *Principles of Operation, 32-Bit ECLIPSE® Systems* manual describes the remainder of the protection violation fault procedure.

Reserved Memory

When a privileged/nonprivileged fault occurs, the processor transfers control to an appropriate fault handler. A reserved storage location in page zero of each segment contains the starting address of the fault handler.

The processor interprets page zero locations of segment 0 slightly different from the page zero locations of segments 1 through 7. For instance, segment 0 contains pointers to privileged fault handlers while segments 1 through 7 reserve these locations. Appendix D describes the page zero locations for all the segments.

NOTE: *The first instruction of the protection fault handler executes before the processor honors interrupts.*

In addition, the MV/4000 computer requires that a contiguous 4-Kbyte block of main memory be allocated to the processor for control purposes. The privileged *Store State Pointer* (SSPT) instruction places the base address, for the contiguous block, from AC0 into the state pointer in memory. The operating system then defines the size of the block.

Store State Pointer Instruction
SSPT

1	1	1	0	0	1	1	1	1	1	0	1	1	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The *Store State Pointer* instruction in conjunction with AC0 and AC1 define the state area in memory.

The instruction moves the physical page frame number (of the state area base) in AC0 to the state pointer. (AC0 bits 20 through 31 contain the 12 bits of a physical page frame number.) At the completion of the instruction, AC1 contains the number of consecutive physical pages that the operating system must reserve in memory above this physical page frame. AC0, AC2, and AC3 remain unchanged.

The state area is available for use by the processor as hardware reserved memory. For the MV/4000, the state area contains the 1024 BMC map translations. The operating system considers the area as unusable memory.

If during the course of operation it becomes necessary to move the state area (for example as a result of a hard memory failure within the state area), the operating system must stop operations that may change the contents of the state area, then perform the move, and finally reload the state pointer by re-executing the *Store State Pointer* instruction.

The operating system must execute the *Store State Pointer* instruction at system initialization time, before the address translator is enabled. After execution, the state area is available for use by the processor.

Chapter 9
C/350 Programming

The MV/4000 computer is capable of executing ECLIPSE C/350 16-bit programs with only slight program instruction modification.

This chapter describes the operation of the MV/4000 system when it implements C/350 instructions. In this chapter we explain:

- Register implementation
- C/350 instruction compatibility
- Program flow management
- Fault handling
- Reserved memory
- CPU identification

Registers

The following C/350 registers are implemented on the MV/4000 computer:

- Four 64-bit floating-point accumulators
- Four 16-bit fixed-point accumulators
- One 32-bit floating-point status register
- One 15-bit program counter
- One 1-bit CARRY flag

The four 64-bit MV/4000 floating-point accumulators are identical to the C/350 floating-point accumulators.

The ECLIPSE C/350 16-bit fixed-point accumulators correspond to bits 16 through 31 of the MV/4000 accumulators.

The 32-bit floating-point status register (FPSR) corresponds to bits 0 through 15 and 49 through 63 of the 64-bit MV/4000 FPSR.

The C/350 15-bit program counter (PC) corresponds to bits 17 through 31 of the MV/4000 31-bit PC.

Execution of C/350 instructions does not generate fixed-point faults thereby leaving the processor status register unaffected. Certain C/350 arithmetic instructions (**ADD**, **DIV**, etc.) set the state of the carry bit. If you want detection of the appropriate fault, it is necessary to set up a subroutine that checks the state of the carry bit upon completion of these instructions. A carry from accumulator bit 16 affects the MV/4000 carry bit upon execution of these C/350 instructions. The instruction dictionary of the *Principles of Operation, 32-Bit ECLIPSE® Systems* manual describes the C/350 instruction set and which instructions affect the carry bit.

C/350 instructions function with the narrow stack, and thus use reserved memory locations for stack management without affecting the MV/4000 stack management registers.

Appendix C illustrates the register fields.

Instruction Compatibility

C/350 program flow instructions maintain their limitations of a 64-Kbyte addressing range.

C/350 instructions that load AC3 with the address of the next instruction *(jump to subroutine)*, or push the address of the next instruction onto the narrow stack *(push and jump)*, calculate effective addresses within the lower 64 Kbytes of the present segment.

The MV/4000 system does not support the following C/350 instructions:

- **XOP**, **XOP1** (replaced by **XOP0**).
- Floating-point function instructions (**FCOSD**, **FCOSS**, **FEXPD**, **FEXPS**, **FLOGD**, **FLOGS**, **FSIND**, **FSINS**, **FPLYD**, **FPLYS**, **FSQRD**, **FSQRS**).
- **VCT**, **SYC**, and **LMP**.

Tables 9.1 through 9.4 list the C/350 instructions and the equivalent 32-bit instructions.

C/350 Instruction	C/350 Instruction Action	Equivalent Instruction
BAM	Block add and move	—
BLM	Block move	WBLM
BTO	Set bit to 1	WBTO
BTZ	Set bit to 0	WBTZ
CLM	Compare to limits and skip	WCLM
CMP	Character compare	WCMP
CMT	Character move until true	WCMT
CMV	Character move	WCMV
COB	Count bits	WCOB
CTR	Character translate and compare	WCTR
DSZ	Decrement and skip if 0	XNDSZ *
EDIT	Edit decimal and alphanumeric 16-bit data	WEDIT
EDSZ	Extended decrement and skip if 0	XNDSZ
EISZ	Extended increment and skip if 0	XNISZ
ELDA	Extended load accumulator	XNLDA
ELDB	Extended load byte (from memory to AC)	XLDB
ESTA	Extended store accumulator	XNSTA
ESTB	Extended store byte (right byte of AC to byte in memory)	XSTB
ISZ	Increment and skip if 0	XNISZ *
LDA	Load accumulator	XNLDA *
LDB	Load byte (from memory to AC)	WLDB
LSN	Load sign	WLSN
POP	Pop multiple accumulators	WPOP
PSH	Push multiple accumulators	WPSH
SNB	Skip on nonzero bit	WSNB
SZB	Skip on 0 bit	WSZB
SZBO	Skip on 0 bit and set to 1	WSZBO
STA	Store accumulator	XNSTA *
STB	Store byte (right byte of AC to byte in memory)	WSTB

Table 9.1 C/350 fixed-point computing instructions

* The 32-bit processor equivalent instruction requires two words.

C/350 Instruction	C/350 Instruction Action	Equivalent Instruction
FAMD	Add double (memory to FPAC)	**XFAMD**
FAMS	Add single (memory to FPAC)	**XFAMS**
FDMD	Divide double (FPAC by memory)	**XFDMD**
FDMS	Divide single (FPAC by memory)	**XFDMS**
FFMD	Fix to memory (FPAC to memory)	**WFFAD** *
FLDD	Load floating-point double	**XFLDD**
FLDS	Load floating-point single	**XFLDS**
FLMD	Float from memory	**WFLAD** *
FLST	Load floating-point status register	**LFLST** **
FMMD	Multiply double (FPAC by memory)	**XFMMD**
FMMS	Multiply single (FPAC by memory)	**XFMMS**
FPOP	Pop floating-point state	**WFPOP**
FPSH	Push floating-point state	**WFPSH**
FSMD	Subtract double (memory from FPAC)	**XFSMD**
FSMS	Subtract single (memory from FPAC)	**XFSMS**
FSST	Store floating-point status register	**LFSST** **
FSTD	Store floating-point double	**XFSTD**
FSTS	Store floating-point single	**XFSTS**
LDI	Load integer (memory to FPAC)	**WLDI**
LDIX	Load integer extended (memory to FPAC)	**WLDIX**
STI	Store integer (FPAC to memory)	**WSTI**
STIX	Store integer extended (FPAC to memory)	**WSTIX**

Table 9.2 C/350 floating-point computing instructions

* The **WFFAD** and **WFLAD** instructions use a 32-bit accumulator, while the equivalent C/350 instruction uses two memory words.

** The **LFLST** or **LFSST** instruction is a triple word instruction, while the C/350 instruction is a double-word instruction.

C/350 Instruction	C/350 Instruction Action	Equivalent Instruction
DSPA	Dispatch	**LDSP**
EJMP	Extended jump	**XJMP**
EJSR	Extended jump to subroutine	**XJSR**
ELEF	Extended load effective address	**XLEF**
JMP	Jump	—
JMP ,1	Jump, relative to the program counter	**WBR**
JSR	Jump to subroutine	—
LEF	Load effective address	—
POPB	Pop block and execute (return from **XOP0**)	**WPOPB**
POPJ	Pop PC and jump (return with **PSHJ**)	**WPOPJ**
PSHJ	Push jump (return with **POPJ**)	**XPSHJ**
PSHR	Push return address (pop with **POPJ**)	—
RSTR	Restore (return from **VCT** -- mode E)	**WRSTR** **
RTN	Return	**WRTN** *
SAVE	Save (used with **JSR**)	**WSSVR, WSSVS** *
SAVZ	Save without arguments (used with **JSR**)	**WSSVR, WSSVS** *
XOP0 ***	Extended operation (return with **POPB**)	**WXOP** ***

Table 9.3 C/350 program flow management instructions

* The **WRTN**, **WSSVS**, and **WSSVR** instructions modify the OVK fixed-point overflow mask and use a return block of six double words.

** The **WRSTR** instruction uses the wide stack, and is equivalent to **RSTR**.

*** The **XOP0** and **WXOP** instructions are double-word instructions.

C/350 Instruction	C/350 Instruction Action	Equivalent Instruction
MSP	Modify stack pointer	**WMSP**
POP	Pop multiple accumulators	**WPOP**
POPB	Pop block and execute (return from **XOP0**)	**WPOPB**
POPJ	Pop PC and jump	**WPOPJ**
PSH	Push multiple accumulators	**WPSH**
PSHJ	Push jump	**XPSHJ**
PSHR	Push return address	—
RSTR	Restore	**WRSTR** **
RTN	Return	**WRTN** *
SAVE	Save (used with **JSR**)	**WSSVR, WSSVS** *
SAVZ	Save without arguments (used with **JSR**)	**WSSVR, WSSVS** *
XOP0 ***	Extended operation (return with **POPB**)	**WXOP** ***

Table 9.4 C/350 stack management instructions

* The **WRTN**, **WSSVS**, and **WSSVR** instructions modify the OVK fixed-point overflow mask and use a return block of six double words.

** The **WRSTR** instruction uses the wide stack, and is equivalent to **RSTR**.

*** The **XOP0** and **WXOP** instructions are double-word instructions.

Program Flow

The program counter governs program flow management as described in the Program Flow Management chapter.

For any C/350 program executing on the MV/4000 computer, when the PC contains 77777_8 and increments to refer to the next instruction, the PC does not wrap around to 0. The PC increments to 100000_8, and the processor fetches the next instruction from this location. This will affect certain data movement instructions (e.g., **BAM, BLM, CMT, CMV, CTR, EDIT**). If data movement is backward (descending addresses) and the process attempts a ring crossing, the address translator indicates a protection violation.

The C/350 program flow instructions load bits 17 through 31 of the PC with the address generated by the program flow instruction. Bits 0 and 4 through 16 are set to 0; bits 1 through 3 remain unchanged.

Appendix C illustrates the PC contents.

Fault Handling

The handling of faults is identical to the handling of MV/4000 system nonprivileged faults as described in the *Principles of Operation, 32-Bit ECLIPSE® Systems* manual. Note that all faults that occur with the execution of C/350 instructions use the narrow stack.

Appendix F lists the error codes returned to AC1 upon the occurrence of a decimal/ASCII fault, and denotes the type of fault generated.

Reserved Memory

The MV/4000 computer does not implement C/350 auto-increment and auto-decrement locations 20_8 through 37_8, which the processor reserves for storage of certain system parameters.

CPU Identification

The **ECLID** and **NCLID** instructions return central processor information.

The **NCLID** instruction loads the CPU identification into bits 16 through 31 of three accumulators (AC0, AC1, and AC2). The **NCLID** instruction can execute only with the LEF mode disabled. With the LEF bit enabled, this instruction becomes a LEF instruction.

ECLID

1	1	1	1	1	1	1	1	1	1	0	0	1	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The **ECLID** instruction loads a double word into AC0. The double word has the format:

Model Number	FPU	Microcode Rev	0	0	Memory Size
0 ... 14	15	16 ... 23	24	25	26 ... 31

Bits	Code Name	Contents or Function
0-14	001000100011100	Binary representation of the machine's model number
15	FPU	0 indicates no FPU option 1 indicates FPU option
16-23	Microcode Revision	Current microcode revision
24-25	0	Must be 0
26-31	Memory Size	Amount of physical memory available: 0 indicates 256 Kbytes of memory 1 indicates 512 Kbytes to a maximum of 31, indicating 8 Mbytes

NCLID

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	0	1	0	0	0	0	0	1	1	1	1	1	1

The **NCLID** instruction loads CPU identification into bits 16 through 31 of the three accumulators. Following is the three-word CPU identification.

AC0 has the format:

16–30	31
Model Number	FPU

AC1 has the format:

16	17–23	24–31
1	Reserved	Microcode Revision

AC2 has the format:

16–31
Memory Size

AC	Code	Meaning
0	0010001000111100	Binary representation of the machine's model number
	FPU	0 indicates no FPU option 1 indicates FPU option
1	Microcode Revision	**Bits** **Meaning** 16 Always 1 17-23 Reserved for future use. 24-31 Current microcode revision. (If AC1 contains 177777_8, you should load the microcode)
2	Memory Size	Amount of physical memory available: 0 indicates 32 Kbytes of memory 1 indicates 64 Kbytes and so on

Appendix A
Instruction Summary

The instruction summary lists the machine-specific instructions alphabetically by assembler-recognizable mnemonic, giving the format, data type used, action performed, and location contents before and after instruction execution.

The C/350 compatible instructions are identified with an asterisk (*) located at the beginning of the instruction mnemonic.

The *Principles of Operation, 32-Bit ECLIPSE® Systems* manual presents a summary of instructions standard to all ECLIPSE MV/Family computers.

The following abbreviations are used throughout this summary:

Abbreviation	Meaning
#	Integer
→	Returned to
+	Addition
=	Equality
OR	Logical OR
?	Unpredictable result
&	Ties together two (or more) items to be operated upon as one
ac	Fixed-point accumulator
acs	Source ac
acd	Destination ac
PSR	Processor status register
sp	Narrow stack pointer
fp	Narrow frame pointer
sl	Narrow stack limit
sa	Narrow stack fault address
E	Calculated effective address
(#)page zero	Address in page zero
x	Unknown and soon to be lost
displ.	Displacement
PC	Program counter
ION	Interrupt on flag

NOTE: *For all operations, unless specifically mentioned:*

Before instruction execution:	**Upon instruction completion:**
OVR=x	unchanged
CRY=x	unchanged
overflow=x	unchanged
FPSR bits=x	updated
BUSY,DONE flags=x	unchanged

Instruction Format	Action	Before (Location =)	After (Location =)
ECLID	CPU id→AC0	AC0=x	CPU id
*HALT NOTE: *HALT=DOC 0,CPU*	Stops the processor	ION flag=x	unchanged
*INTA ac NOTE: *INTA ac=DIB ac,CPU*	device code→ac	ac=x ION flag=x	device code unchanged
*INTDS NOTE: *INTDS=NIOC CPU*	0→ION flag	ION flag=x	0
*INTEN NOTE: *INTEN=NIOS CPU*	1→ION flag	ION flag=x	1
*IORST NOTE: *IORST=DICC 0,CPU*	Clear all I/O devices 0→priority mask	ION flag=x BUSY,DONE flags=x	0 0

*SSPT	(AC0)→State Pointer	AC0=base of state pointer
		AC1=x
		AC2=x
		AC3=x
			unchanged
			
			AC1=#pages
			unchanged
			unchanged

*MSKO ac
NOTE: *MSKO ac=DOB ac,CPU*
	ac→priority mask	ac=#
		ION flag=x
		mask=x
			unchanged
			unchanged
			ac

NCLID	CPU id→AC0&AC1&AC2	AC0=x
		AC1=x
		AC2=x
			model number
			microcode rev
			memory size

*READS ac NOTE: *READS ac=DIA ac,CPU*	console switches→ac	ac=x ION flag=x	result unchanged
*SKPt device	If t=true =skip	BUSY,DONE flags=x	unchanged

WLMP	(E)→map slots	AC0=#(1st slot #)
		AC1=#(# slots)
		AC2=E
			last
			0
			lastE+2

Appendix B
Instruction Execution Times

The following data sheets give the average execution times of the instructions supported by the ECLIPSE MV/4000 computer. Times throughout are in microseconds.

The instruction execution times listed assume that:

- Physical memory modules are 1 or 2 Mbytes.
- All logical-to-physical address translations are resident in the address translator.
- There is no DCH or BMC activity.
- The EDIT and WEDIT subopcodes that process commercial numeric data assume a data type of 4 and assume that the source pointer (j) into the data is never moved out of the bounds of the data.

If such is not the case, add the following:

To Every Memory Reference	Add (microseconds)
If logical-to-physical address translation traverses page tables in memory	
For one-level page table	1.2
For two-level page table	1.8
If indirection is specified by the instruction	0.8 per level of indirection
To Any Instruction	**Add (microseconds)**
If any of the following faults occurs	
Stack overflow/underflow Fixed-point fault	8.6
Protection fault	9.0
	15.2

The C/350 compatible instructions are identified with an asterisk (*) following the instruction. Any instruction capable of specifying indirection is identified with a tilde (•).

Mnemonic	Timing (microseconds)
ADC *	0.4 + 0.1 if skip + 0.2 if shift or swap
ADD *	0.4 + 0.1 if skip + 0.2 if shift or swap
ADDI *	0.4
ADI *	0.6
ANC *	0.4 + 0.1 if skip + 0.2 if shift or swap
AND *	0.4 + 0.1 if skip + 0.2 if shift or swap
ANDI *	0.4
BAM *	4.2 + 0.60 (number of words moved) + 0.8 (each level of indirect addressing)
BKPT	6.2 + 0.8 (each level of indirect addressing)
BLM *	4.2 + 0.5 (number of words moved) + 0.8 (each level of indirect addressing)
BTO *	1.6 + 0.2 if indirect + 0.8 for each level of indirect addressing
BTZ *	1.6 + 0.2 if indirect + 0.8 for each level of indirect addressing
CLM *	1.6 (2.6 if ACS=ACD)
CMP *	6.2 + 2.0/byte min. 7.6 + 2.0/byte max.
CMT *	4.0 + 2.8/byte min. 3.6 + 3.6/byte max.
CMV *	8.2 + 0.1/byte min. 10.0 + 1.6/byte max.
COB *	2.8
COM *	0.4 + 0.1 if skip + 0.2 if shift or swap
CRYTC	0.4
CRYTO	0.4
CRYTZ	0.4
CTR *	Translate and move 2.4 + 2.6/byte Translate and compare 4.4 + 4.2/byte
CVWN	0.6
DAD *	1.2
DEQUE	4.0
DHXL *	2.7
DHXR *	2.7
DIV *	5.4
DIVS *	6.2
DIVX *	6.8
DLSH *	3.8
DERR	4.2
DSB *	1.0
DSPA *	3.6 + 0.8 (each level of indirect addressing)
DSZ *	1.2 + 0.1 if skip
DSZTS	1.2 + 0.1 if skip
ECLID	0.4

Mnemonic	Timing (microseconds)
EDIT *	4.0 + sum of sub-op execution times that are processed
	DADI 4.6
	DAPS 2.6 (w/o add) 5.0 (with add)
	DAPT 2.6 (w/o add) 5.0 (with add)
	DAPU 4.8
	DASI 5.2 (type 4) 6.6 (type 5)
	DDTK 9.4
	DEND 3.8
	DICI 4.0 + 2.6 per char. insert
	DIMC 6.4 + 1.4 per char. insert + 1.6 if parameter j is located in the narrow stack
	DINC 4.6
	DINS 4.8
	DINT 4.4
	DMVA 5.6 + 3.2 per char. moved + 1.6 if parameter j is located in the narrow stack
	DMVC 5.6 + 2.8 per char. moved + 1.6 if parameter j is located in the narrow stack
	DMVF 5.8 + 5.0 per digit moved + 1.6 if parameter j is located in the narrow stack
	DMVN 5.4 + 4.2 per digit moved + 1.6 if parameter j is located in the narrow stack

Mnemonic	Timing (microseconds)
EDIT* (cont.)	DMVO 9.0 DMVS 6.4 + 4.6 per digit moved + 1.6 if parameter j is located in the narrow stack DNDF 5.2 DSSO 2.6 DSSZ 2.6 DSTK 7.6 DSTO 2.6 DSTZ 2.6
EDSZ *	1.2 + 0.1 if skip ~
EISZ *	1.2 + 0.1 if skip ~
EJMP *	1.0 ~
EJSR *	1.2 ~
ELDA *	0.8 ~
ELDB *	1.2
ELEF *	0.8 ~
ENQH	5.6
ENQT	5.6
ESTA *	0.6 ~
ESTB *	1.0
FAB *	1.0
FAD *	8.4 (FPSR Bit 8=0) 8.8 (FPSR Bit 8=1)
FAMD *	9.2 (FPSR Bit 8=0) 9.6 (FPSR Bit 8=1)
FAMS *	6.8 (FPSR Bit 8=0) 7.0 (FPSR Bit 8=1)
FAS *	6.0 (FPSR Bit 8=0) 6.2 (FPSR Bit 8=1)
FCLE *	1.0
FCMP *	3.0
FDD *	29.8 (FPSR bit 8=0) 37.2 (FPSR bit 8=1)
FDMD *	30.6 (FPSR bit 8=0) 38.0 (FPSR bit 8=1)
FDMS *	11.0 (FPSR bit 8=0) 14.2 (FPSR bit 8=1)
FDS *	10.2 (FPSR bit 8=0) 13.4 (FPSR bit 8=1)
FEXP *	1.6
FFAS *	4.2
FFMD *	5.2 ~
FHLV *	3.8 (FPSR bit 8=0) 4.4 (FPSR bit 8=1)
FINT *	5.4
FLAS *	4.2
FLDD *	2.8 ~
FLDS *	2.6 ~

Mnemonic	Timing (microseconds)
FLMD *	8.0 ∿
FLST *	3.0
FMD *	32.2 (FPSR bit 8=0) 33.0 (FPSR bit 8=1)
FMMD *	33.0 (FPSR bit 8=0) ∿ 33.8 (FPSR bit 8=1) ∿
FMMS *	11.6 (FPSR bit 8=0) ∿ 12.2 (FPSR bit 8=1) ∿
FMOV *	2.2
FMS *	10.8 (FPSR bit 8=0) 11.4 (FPSR bit 8=1)
FNEG *	1.4
FNOM *	7.0
FNS *	0.4
FPOP *	12.0
FPSH *	10.6
FRDS *	3.4
FRH *	1.0
FSA *	0.4
FSCAL *	6.0
FSD *	8.8 (FPSR bit 8=0) 9.2 (FPSR bit 8=1)
FSEQ *	0.4 + 0.1 if skip
FSGE *	0.4 + 0.1 if skip
FSGT *	0.8
FSLE *	0.8
FSLT *	0.4 + 0.1 if skip
FSMD *	9.4 (FPSR bit 8=0) ∿ 9.8 (FPSR bit 8=1) ∿
FSMS *	7.0 (FPSR bit 8=0) ∿ 7.2 (FPSR bit 8=1) ∿
FSND *	0.6
FSNE *	0.4 + 0.1 if skip
FSNER *	0.6
FSNM *	0.6
FSNO *	0.6
FSNOD *	0.6
FSNU *	0.6
FSNUD *	0.6
FSNUO *	0.6
FSS *	6.4 (FPSR bit 8=0) 6.6 (FPSR bit 8=1)
FSST *	3.4 ∿
FSTD *	2.2 ∿
FSTS *	1.6 ∿
FTD *	0.8

Mnemonic	Timing (microseconds)
FTE *	0.8
FXTD	0.8
FXTE	1.0
HLV *	0.60
HXL *	0.95
HXR *	0.95
INC *	0.4 + 0.1 if skip + 0.2 if shift or swap
IOR *	0.4
IORI *	0.6
ISZ *	1.2 + 0.1 if skip ~
ISZTS	1.2 + 0.1 if skip ~
JMP *	1.0 ~
JSR *	1.2 ~
LCALL	4.4 (no indirect, no ring crossing, and no gate checking) ~ 5.6 (indirect, but no ring crossing, and no gate checking) 13.8 + 0.8 (arg_count) (no indirect, but with ring crossing, and gate checking) 15.6 + 0.8(arg_count) (one indirect, with ring crossing, and gate checking) NOTE: All indirections past the first indirection require an additional 0.8 for each indirection.
LCPID	0.4
LDA *	0.8 ~
LDAFP	0.4
LDASB	0.4
LDASL	0.4
LDASP	0.4
LDATS	0.8
LDB *	1.0
LDI *	27.2 (Type 4, length 7)
LDIX *	117.8(Type 4, length 31)
LDSP	3.4 ~
LEF *	0.8 ~
LFAMD	9.2 (FPSR Bit 8=0) ~ 9.6 (FPSR Bit 8=1) ~
LFAMS	6.8 (FPSR Bit 8=0) ~ 7.0 (FPSR Bit 8=1) ~
LFDMD	30.6 (FPSR bit 8=0) 38.0 (FPSR bit 8=1)
LFDMS	11.0 (FPSR bit 8=0) 14.2 (FPSR bit 8=1)
LFLDD	2.8 ~
LFLDS	2.6 ~
LFLST	3.0 ~
LFMMD	33.0 (FPSR Bit 8=0) ~ 33.8 (FPSR Bit 8=1) ~
LFMMS	11.6 (FPSR Bit 8=0) ~ 12.2 (FPSR Bit 8=1) ~
LFSMD	9.4 (FPSR Bit 8=0) ~ 9.8 (FPSR Bit 8=1) ~
LFSMS	7.0 (FPSR Bit 8=0) ~ 7.2 (FPSR Bit 8=1) ~

Mnemonic	Timing (microseconds)
LFSST	3.6 ∿
LFSTD	2.2 ∿
LFSTS	1.6 ∿
LJMP	1.0 ∿
LJSR	1.0 ∿
LLDB	1.2
LLEF	0.6 ∿
LLEFB	0.6
LMRF	3.0
LNADD	1.2 ∿
LNADI	1.2
LNDIV	7.0 ∿
LNDO	2.8 (no termination) 4.6 (for termination)
LNDSZ	1.2 + 0.1 if skip
LNISZ	1.2 + 0.1 if skip
LNLDA	0.8 ∿
LNMUL	5.6 ∿
LNSBI	1.2
LNSTA	0.6 ∿
LNSUB	1.2 ∿
LOB *	1.0 (if no bit set) 1.2 + 0.2 (if no leading zeros)
LPEF	1.4 ∿
LPEFB	1.4
LPHY	7.2 (valid and 2 level)
LPSHJ	1.8 ∿
LPSR	0.6
LRB *	1.0 + 0.4 (number of leading zeros) acs<>acd 0.8 + 0.2 (number of leading zeros) acs=acd
LSBRA	20.4
LSBRS	20.0
LSH *	2.0
LSN *	4.6 + 1.8/leading zero digit
LSTB	1.0
LWADD	1.2 ∿
LWADI	1.2 ∿
LWDIV	9.6 ∿
LWDO	2.4 (no termination) 4.2 (for termination)
LWDSZ	1.2 + 0.1 if skip ∿
LWISZ	1.2 + 0.1 if skip ∿
LWLDA	0.8 ∿
LWMUL	9.2 ∿
LWSBI	1.2 ∿
LWSTA	0.6 ∿

Mnemonic	Timing (microseconds)
LWSUB	1.2 ∼
MOV *	0.4 + 0.1 if skip + 0.2 if shift or swap
MSP *	1.8
MUL *	5.2
MULS *	5.2
NADD	0.6
NADDI	0.6
NADI	0.8
NBStc	3.6 + 1.4 per search
NCLID	3.0
NDIV	6.4
NEG *	0.4 + 0.1 if skip + 0.2 if shift or swap
NFStc	3.6 + 1.4 per search
NLDAI	0.4
NMUL	5.0
NNEG	0.6
NSALA	0.6 + 0.1 if skip
NSALM	1.0 + 0.1 if skip
NSANA	0.2 + 0.1 if skip
NSANM	1.0 + 0.1 if skip
NSBI	0.8
NSUB	0.6
ORFB	1.2 + 5.2 (count), count = AC0 + 1
PATU	13.6
PBX	11.8 + executed instruction
POP *	2.0 + 0.4 per ac
POPB *	4.2
POPJ *	3.2
PSH *	2.0 + 0.4 per ac
PSHJ *	3.0
PSHR *	2.0
RRFB	0.8 + 2.6 (count) count = AC0 +1
RSTR *	5.2
RTN *	4.4
SAVE *	5.2
SAVZ	5.2
SBI *	0.6
SEX	0.4
SGE *	0.4 + 0.1 if skip
SGT *	0.4 + 0.1 if skip
SMRF	3.8
SNB *	1.6 + 0.2 if indirect
SNOVR	0.6 + 0.1 if skip
SPSR	0.4
STA *	0.6 ∼

Mnemonic	Timing (microseconds)
STAFP	0.4
STASB	0.8
STASL	0.8
STASP	0.4
STATS	0.6
STB *	0.8
STI *	42.0 (Type 4, length 7)
STIX *	161.0 (Type 4, length 31)
SUB *	0.4 + 0.1 if skip + 0.2 if shift or swap
SZB *	1.6 + 0.1 if skip + 0.2 if indirect
SZBO *	2.0 + 0.1 if skip + 0.8 if indirect
WADC	0.4
WADD	0.4
WADDI	0.4
WADI	0.6
WANC	0.4
WAND	0.4
WANDI	0.4
WASH	4.25
WASHI	4.25
WBLM	7.2 + 0.5 (number of words moved) + 0.8 (each level of indirect addressing)
WBR	1.0
WBStc	3.4 + 1.4 per search
WBTO	1.8
WBTZ	1.8
WCLM	1.6 (acs<>acd) 2.6 (acs=acd)
WCMP	7.4 + 2.0/byte min. 8.8 + 2.0/byte max.
WCMT	5.0 + 2.8/byte min. 5.4 + 3.6/byte max.
WCMV	9.6 + 0.1/byte min. 11.4 + 1.6/byte max.
WCOB	5.2
WCOM	0.4
WCST	6.2 + 2.8/byte min. 7.2 + 2.8/byte max.
WCTR	2.4 + 2.6/byte min. 4.4 + 4.2/byte max.
WDIV	9.0
WDIVS	9.4
WDPOP	8.4 (no indirect and restart block size of 1) 15.2 (no indirect and resume block size of 2) 20.2 (no indirect and resume block size of 3) 9.6 (one indirect and restart block size of 1) 16.4 (one indirect and resume block size of 2) 21.4 (one indirect and resume block size of 3) 12.0 (ring crossing and restart block size of 1) 18.8 (ring crossing and resume block size of 2) 23.8 (ring crossing and resume block size of 3) NOTE: All indirections past the first indirection require an additional 0.8 for each indirection.

Mnemonic	Timing (microseconds)	
WEDIT	4.0 + sum of sub-op execution times that are processed	
	DADI	4.6
	DAPS	2.6 (w/o add) 5.0 (with add)
	DAPT	2.6 (w/o add) 5.0 (with add)
	DAPU	4.8
	DASI	5.2 (type 4) 6.6 (type 5)
	DDTK	8.8
	DEND	3.8
	DICI	4.0 + 2.6 per char. insert
	DIMC	6.4 + 1.4 per char. insert + 1.0 if parameter j is located in the wide stack
	DINC	4.6
	DINS	4.8
	DINT	4.4
	DMVA	5.6 + 3.2 per char. moved + 1.0 if parameter j is located in the wide stack
	DMVC	5.6 + 2.8 per char. moved + 1.0 if parameter j is located in the wide stack
	DMVF	5.8 + 5.0 per digit moved + 1.0 if parameter j is located in the wide stack
	DMVN	5.4 + 4.2 per digit moved + 1.0 if parameter j is located in the wide stack
	DMVO	9.0

Mnemonic	Timing (microseconds)
WEDIT (cont.)	DMVS 6.4 + 4.6 per digit moved + 1.0 if parameter j is located in the wide stack
	DNDF 5.2 DSSO 2.6 DSSZ 2.6 DSTK 7.0 DSTO 2.6 DSTZ 2.6
WFFAD	4.4
WFLAD	8.0
WFPOP	12.0
WFPSH	10.2
WFStc	3.4 + 1.4 per search
WHLV	0.6
WINC	0.4
WIOR	0.4
WIORI	0.4
WLDAI	0.4
WLDB	1.0
WLDI	27.2 (Type 4, length 7)
WLDIX	117.8 (Type 4, length 31)
WLDO	2.4 (no termination) 4.2 (for termination)
WLMP	3.2 + 1.6 (Number of BMC slots 2.6 + 1.6 (Number of DCH slots
WLOB	1.0 + 0.4 (number of leading zeros) acs<>acd 0.8 + 0.2 (number of leading zeros) acs=acd
WLRB	1.0 + 0.4 (number of leading zeros) acs<>acd 0.8 + 0.2 (number of leading zeros) acs=acd
WLSH	2.1
WLSHI	2.3
WLSI	0.6 + 2.0 per search
WLSN	4.6 + 1.8/leading zero digit
WMESS	1.8
WMOV	0.4
WMOVR	0.4
WMSP	1.4
WMUL	8.6
WMULS	8.4
WNADI	0.8
WNDO	2.8 (no termination) 4.6 (for termination)
WNEG	0.4
WPOP	1.2 + 0.4 per ac
WPOPB	4.8 intra ring 9.0 cross ring
WPOPJ	1.8
WPSH	1.0 + 0.4 per AC
WRSTR	9.0 intra ring 11.6 cross ring
WRTN	5.2 intra ring 9.2 cross ring

Mnemonic	Timing (microseconds)
WSALA	0.6 + 0.1 if skip
WSALM	1.0 + 0.1 if skip
WSANA	0.4 + 0.1 if skip
WSANM	1.0 + 0.1 if skip
WSAVR	4.8
WSAVS	4.8
WSBI	0.6
WSEQ	0.4 + 0.1 if skip + 0.2 if compare to 0
WSEQI	0.4 + 0.1 if skip
WSGE	0.4 + 0.1 if skip + 0.2 if compare to 0
WSGT	0.4 + 0.1 if skip + 0.2 if compare to 0
WSGTI	0.4 + 0.1 if skip
WSKBO	1.6 min. + 0.1 if skip 1.8 max. + 0.1 if skip
WSKBZ	1.6 min. + 0.1 if skip 1.8 max. + 0.1 if skip
WSLE	0.4 + 0.1 if skip + 0.2 if compare to 0
WSLEI	0.4 + 0.1 if skip
WSLT	0.4 + 0.1 if skip + 0.2 if compare to 0
WSNB	1.8 + 0.1 if skip
WSNE	0.4 + 0.1 if skip + 0.2 if compare to 0
WSNEI	0.4 + 0.1 if skip
WSSVR	5.2
WSSVS	5.2
WSTB	0.8
WSTI	42.0 (Type 4, length 7)
WSTIX	161.0(Type 4, length 31)
WSUB	0.4
WSZB	1.8 + 0.1 if skip
WSZBO	2.2 + 0.1 if skip
WUGTI	0.4 + 0.1 if skip
WULEI	0.4 + 0.1 if skip
WUSGE	0.4 + 0.1 if skip
WUSGT	0.4 + 0.1 if skip + 0.2 if compare to 0
WXCH	0.6
WXOP	7.6 + 0.8(indirect)
WXOR	0.4
WXORI	0.6
XCALL	4.4 (no indirect, no ring crossing, and no gate checking) 5.6 (indirect, but no ring crossing, and no gate checking) 13.8 + 0.8(arg_count) (no indirect, but with ring crossing, and gate checking) 15.6 + 0.8(arg_count) (one indirect, with ring crossing, and gate checking) NOTE: All indirections past the first indirection require an additional 0.8 for each indirection.
XCH *	0.6

Mnemonic	Timing (microseconds)
XCT *	3.0 + executed instruction
XFAMD	9.2 (FPSR bit 8=0) ∿ 9.6 (FPSR bit 8=1) ∿
XFAMS	6.8 (FPSR bit 8=0) ∿ 7.0 (FPSR bit 8=1) ∿
XFDMD	30.6 (FPSR bit 8=0) ∿ 38.0 (FPSR bit 8=1) ∿
XFDMS	11.0 (FPSR bit 8=0) ∿ 14.2 (FPSR bit 8=1) ∿
XFLDD	2.8 ∿
XFLDS	2.6 ∿
XFMMD	33.0 (FPSR bit 8=0) ∿ 33.8 (FPSR bit 8=1) ∿
XFMMS	11.6 (FPSR bit 8=0) ∿ 12.2 (FPSR bit 8=1) ∿
XFSMD	9.4 (FPSR bit 8=0) ∿ 9.8 (FPSR bit 8=1) ∿
XFSMS	7.0 (FPSR bit 8=0) ∿ 7.2 (FPSR bit 8=1) ∿
XFSTD	2.2 ∿
XFSTS	1.6 ∿
XJMP	1.0 ∿
XJSR	1.0 ∿
XLDB	1.4 ∿ 1.6 if absolute mode ∿
XLEF	0.6 ∿
XLEFB	0.8 1.6 if absolute mode
XNADD	1.2 ∿
XNADI	1.2 ∿
XNDIV	7.0 ∿
XNDO	2.8 (no termination) 4.6 (for termination)
XNDSZ	1.2 + 0.1 if skip ∿
XNISZ	1.2 + 0.1 if skip ∿
XNLDA	0.8 ∿
XNMUL	5.6 ∿
XNSBI	1.2 ∿
XNSTA	0.6 ∿
XNSUB	1.2 ∿
XOP0 *	6.4 +0.8 (indirect)
XOR *	0.4
XORI *	0.4
XPEF	1.4 ∿
XPEFB	1.6 2.6 if absolute mode

Mnemonic	Timing (microseconds)
XPSHJ	1.8 ~
XSTB	1.0
XVCT	22.4 + 7.0 base level + 0.8 (indirect) + 2.0 (from interrupt)
XWADD	1.2 ~
XWADI	1.2 ~
XWDIV	9.6 ~
XWDO	2.4 (no termination) 4.2 (for termination)
XWDSZ	1.2 + 0.1 if skip ~
XWISZ	1.2 + 0.1 if skip ~
XWLDA	0.8 ~
XWMUL	9.2 ~
XWSBI	1.2 ~
XWSTA	0.6 ~
XWSUB	1.2 ~
ZEX	0.4

The following data sheets give the average execution times of the floating-point instructions affected by the *optional hardware floating-point instruction set accelerator.*

Mnemonic	Timing (microseconds)
FAB *	1.0
FAD *	2.2 (FPSR Bit 8=0) 2.6 (FPSR Bit 8=1)
FAMD *	3.4 (FPSR Bit 8=0) 3.8 (FPSR Bit 8=1)
FAMS *	2.6 (FPSR Bit 8=0) 3.0 (FPSR Bit 8=1)
FAS *	2.2 (FPSR Bit 8=0) 2.6 (FPSR Bit 8=1)
FCLE *	2.2
FCMP *	2.0
FDD *	11.8 (FPSR bit 8=0) 12.6 (FPSR bit 8=1)
FDMD *	13.0 (FPSR bit 8=0) 13.8 (FPSR bit 8=1)
FDMS *	7.4 (FPSR bit 8=0) 8.2 (FPSR bit 8=1)
FDS *	7.0 (FPSR bit 8=0) 7.8 (FPSR bit 8=1)
FEXP *	2.2
FFAS *	3.0
FFMD *	2.6 ~
FHLV *	1.4 (FPSR bit 8=0) 1.6 (FPSR bit 8=1)
FINT *	1.8
FLAS *	2.2
FLDD *	2.2 ~
FLDS *	1.4 ~
FLMD *	2.8 ~
FLST *	3.0 4.2 (FPSR bit TE=0)
FMD *	11.2 (FPSR bit 8=0) 11.6 (FPSR bit 8=1)
FMMD *	12.4 (FPSR bit 8=0) ~ 12.8 (FPSR bit 8=1) ~
FMMS *	6.8 (FPSR bit 8=0) ~ 7.2 (FPSR bit 8=1) ~
FMOV *	1.0
FMS *	6.4 (FPSR bit 8=0) 6.8 (FPSR bit 8=1)
FNEG *	1.0
FNOM *	1.6
FNS *	0.4
FPOP *	19.0 20.2 (FPSR bit TE=0)

Mnemonic	Timing (microseconds)
FPSH *	13.2
FRDS *	1.4
FRH *	1.2
FSA *	0.4
FSCAL *	3.0
FSD *	2.2 (FPSR bit 8=0) 2.6 (FPSR bit 8=1)
FSEQ *	1.2
FSGE *	1.2
FSGT *	1.2
FSLE *	1.2
FSLT *	1.2
FSMD *	3.4 (FPSR bit 8=0) ~ 3.8 (FPSR bit 8=1) ~
FSMS *	2.6 (FPSR bit 8=0) ~ 3.0 (FPSR bit 8=1) ~
FSND *	1.2
FSNE *	1.2
FSNER *	1.2
FSNM *	1.2
FSNO *	1.2
FSNOD *	1.2
FSNU *	1.2
FSNUD *	1.2
FSNUO *	1.2
FSS *	2.2 (FPSR bit 8=0) 2.6 (FPSR bit 8=1)
FSST *	3.4 ~
FSTD *	2.6 ~
FSTS *	1.6 ~
FTD *	0.4
FTE *	1.2
LFAMD	3.4 (FPSR Bit 8=0) ~ 3.8 (FPSR Bit 8=1) ~
LFAMS	2.6 (FPSR Bit 8=0) ~ 3.0 (FPSR Bit 8=1) ~
LFDMD	13.0 (FPSR bit 8=0) 13.8 (FPSR bit 8=1)
LFDMS	7.4 (FPSR bit 8=0) 8.2 (FPSR bit 8=1)
LFLDD	2.2 ~
LFLDS	1.4 ~
LFLST	3.0 ~ 4.2 (FPSR bit TE=0)
LFMMD	12.4 (FPSR Bit 8=0) ~ 12.8 (FPSR Bit 8=1) ~
LFMMS	6.8 (FPSR Bit 8=0) ~ 7.2 (FPSR Bit 8=1) ~

Mnemonic	Timing (microseconds)
LFSMD	3.4 (FPSR Bit 8=0) ∿ 3.8 (FPSR Bit 8=1) ∿
LFSMS	2.6 (FPSR Bit 8=0) ∿ 3.0 (FPSR Bit 8=1) ∿
LFSST	3.6 ∿
LFSTD	2.6 ∿
LFSTS	1.6 ∿
WFFAD	3.0
WFLAD	2.6
WFPOP	18.2 19.4 (FPSR bit TE=0)
WFPSH	12.4
XFAMD	3.4 (FPSR bit 8=0) ∿ 3.8 (FPSR bit 8=1) ∿
XFAMS	2.6 (FPSR bit 8=0) ∿ 3.0 (FPSR bit 8=1) ∿
XFDMD	13.0 (FPSR bit 8=0) ∿ 13.8 (FPSR bit 8=1) ∿
XFDMS	7.4 (FPSR bit 8=0) ∿ 8.2 (FPSR bit 8=1) ∿
XFLDD	2.2 ∿
XFLDS	1.4 ∿
XFMMD	12.4 (FPSR bit 8=0) ∿ 12.8 (FPSR bit 8=1) ∿
XFMMS	6.8 (FPSR bit 8=0) ∿ 7.2 (FPSR bit 8=1) ∿
XFSMD	3.4 (FPSR bit 8=0) ∿ 3.8 (FPSR bit 8=1) ∿
XFSMS	2.6 (FPSR bit 8=0) ∿ 3.0 (FPSR bit 8=1) ∿
XFSTD	2.6 ∿
XFSTS	1.6 ∿

Appendix C
Register Fields

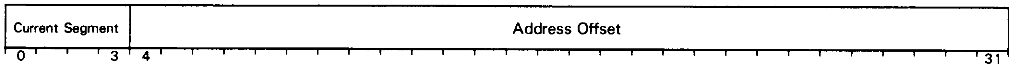
In this appendix, we present the formats for the programmer-accessible registers available on the MV/4000 computer for both MV/4000-system-specific and C/350 compatible formats.

Register	Purpose
Program Counter	Contains the logical address of the currently executing instruction
Processor Status Register	Contains information pertaining to fixed-point computations
Floating-Point Status Register	Contains information pertaining to floating-point computations
Segment Base Registers	Contain information pertaining to MV/4000 logical address translation
DCH/BMC Status Registers	Contain information pertaining to data channel and burst multiplexor channel maps
CPU Identification	Accumulators contain information pertaining to the CPU

Program Counter

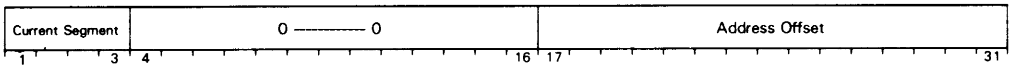
The 31-bit PC contains the logical address of the currently executing instruction; the formats follow:

PC Format for Execution of MV/4000-System-Specific Programs



Bits	Name	Contents or Function
1-3	Current Segment	The current segment of program execution
4-31	Address Offset	The 28-bit address of the currently executing instruction

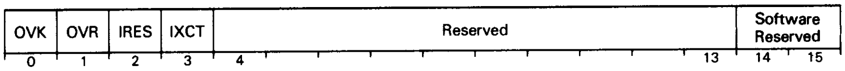
PC Format Altered by C/350 Program Flow Instructions

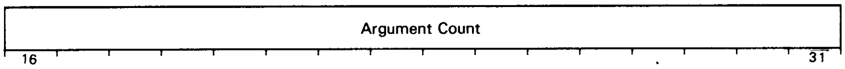


Bits	Name	Contents or Function
1-3	Current Segment	The current segment of program execution
4-16	0-0	Set to 0 by instruction
17-31	Address Offset	The 15-bit address formed by the program flow instruction

Processor Status Register

Only MV/4000-system-specific instructions affect the 32-bit PSR. The format of the PSR follows:





Bits	Name	Contents or Function
0	OVK	Overflow Mask 0 indicates no fixed-point overflow trap 1 indicates trap on OVR set to 1
1	OVR	Fixed-point overflow indicator; set to 1 when calculating a two's complement number that does not fit in the specified location or register, or when attempting to divide by 0 If OVK equals 1, then the setting of OVR to 1 results in a fixed-point overflow fault
2	IRES	Micro-interrupt resume flag; set to 1 when the processor receives an I/O interrupt request while executing a resumable interruptible instruction (such as, WEDIT instruction)
3	IXCT	Interrupt execute flag; set to 1 when the processor receives an I/O interrupt request while executing an instruction that was inserted into the instruction stream (such as, a PBX instruction)
4-13	Reserved	Bits 4 through 13 are reserved for future use
14-15	Software Reserved	Bits 14 and 15 are software reserved in the return block
16-31	Argument Count	Bits 16 through 31 contain the number of arguments to pass with the LCALL or XCALL

NOTE: *Any instruction that loads the OVK and OVR bits as part of its execution, will not cause an overflow fault even if both are set to 1.*

For all C/350 instructions, *overflow* equals 0, thereby leaving OVR unchanged.

Floating-Point Status Register

MV/4000-system-specific and C/350 instructions affect the 64-bit FPSR. The FPSR format follows.

NOTE: *When the C/350* **FLST** *and* **FSST** *instructions write to or read from the FPSR, the instructions ignore bits 16 through 48.*

ANY	OVF	UNF	DVZ	MOF	TE	Z	N	RND	Res	0	0	FPMOD
0	1	2	3	4	5	6	7	8	9	10	11	12 15

Reserved (All 0)
0 15

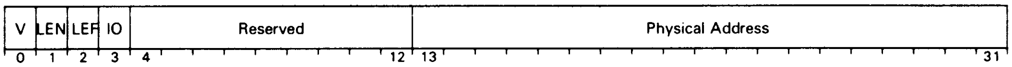
0	Floating-Point Program Counter (Bits 1-15)
0	1 15

Floating-Point Program Counter (Bits 16-31)
0 15

Bits	Name	Contents or Function
0	ANY	Indicates the setting to 1 of any of bits 1 through 4
1	OVF	Exponent overflow indicator
2	UNF	Exponent underflow indicator
3	DVZ	Divide by 0
4	MOF	Mantissa overflow
5	TE	Trap enable; if set to 1, setting of any of bits 1 through 4 will result in a floating-point fault
6	Z	Zero bit
7	N	Negative bit
8	RND	Floating-point rounding mode.
9-11	Reserved	Bits 9 through 11 are reserved for future use and must be set to 0
12-15	FPMOD	Floating-point model; should be set to 0111
16-31	Reserved	Bits 16 through 31 are reserved for future use; these should be set to 0
32	0	Should be set to 0
33-63	Floating-Point Program Counter	Floating-point program counter. In the event of a floating-point fault, this is the address of the first floating-point instruction that caused the fault

Segment Base Registers

The 32-bit segment base registers (SBR) contain information for the MV/4000-system-specific logical address translation mechanism and for I/O protection. The format follows:



Bits	Name	Contents or Function
0	V	Segment validity bit -- indicates the process' ability to refer to a segment 0 indicates an invalid SBR 1 indicates a valid SBR
1	LEN	Length bit -- indicates the maximum range of the logical memory address 0 indicates a one-level page table 1 indicates a two-level page table
2	LEF	LEF enable -- indicates whether the processor will operate in LEF or I/O mode 0 indicates I/O mode 1 indicates LEF mode
3	IO	I/O enable -- indicates if an I/O protection violation will occur upon an execution of an I/O instruction 0 indicates protection violation will occur 1 indicates the I/O instruction will execute
4-12	Reserved	Bits 4 through 12 are reserved for future use
13-31	Physical Address	Identifies the physical page address in memory of the indicated page table

DCH/BMC Status Registers

The port definition register (6000_8) provides status information. The format for the register follows:

DCH/BMC Port Definition Register (6000_8)

E	Reserve	BV	DV	Reserve	A	P	Res	0-0	M	1
0	1 2	3	4	5 6	7	8	9	10 13	14	15

Bits	Name	Contents or Function
0	E	Error flag; if 1, an error has occurred on the I/O port (0 only when all other error bits are 0)
1,2	Reserved	Bits 1 and 2 are reserved for future use and returned as zero
3	BV	BMC validity error flag; if 1, BMC validity protect error has occurred
4	DV	DCH validity error flag; if 1, DCH validity protect error has occurred
5,6	Reserved	Bits 5 and 6 are reserved for future use and returned as zero
7	A	BMC address error; if 1, the channel has detected an address parity error
8	P	BMC data error; if 1, the channel has detected a data parity error
9-13	Reserved	Bits 9 through 13 are reserved for future use and returned as zero
14	M	DCH mode; if 1, DCH mapping is enabled
15	1	Always set to 1

NOTES: *Setting bit 3, 4, 7, or 8 to a one, with the CIO instruction complements these bits.*

The C/350 **IORST** *instruction clears bits 0, 3, 4, 7, 8, and 14.*

The read-only port status register (7700_8) provides status information. The format for the register follows:

DCH/BMC Port Status Register (7700_8)

ERR	Reserved	Res	1	1	Res	INT
0	1 10	11	12	13	14	15

Bits	Name	Contents or Function
0	ERR	If 1, the port has detected an error indicated by the port definition register
1-11	Reserved	Bits 1 through 11 are reserved for future use
12-13	1,1	Always set to 1
14	Reserved	Bit 14 is reserved for future use and returned as zero
15	INT	Interrupt pending if 1

CPU Identification

The three Load CPU Identification instructions return the information shown below to the specified accumulators.

LCPID and ECLID Instructions

The **LCPID** and **ECLID** instructions load a 32-bit double word into AC0.

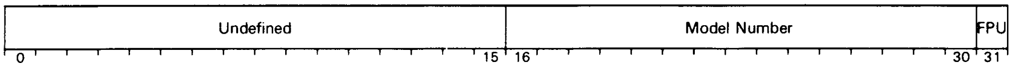
Model Number	FPU	Microcode Rev	0	0	Memory Size
0 14	15	16 23	24	25	26 31

Bits	Name	Contents or Function
0-14	001000100011100	Model Number; the binary value of the model number allocated to the processor
15	FPU	0 indicates no FPU option 1 indicates FPU option
16,23	Microcode Rev	Current microcode revision
24-25	0	Set to 0
26-31	Memory Size	Amount of physical memory available: A 0 indicates 256 Kbytes A 1 indicates 512 Kbytes to a maximum of 31, indicating 8 Mbytes

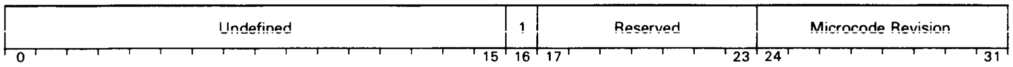
NCLID Instruction

The NCLID instruction loads the result into the low-order 16 bits of the three accumulators.

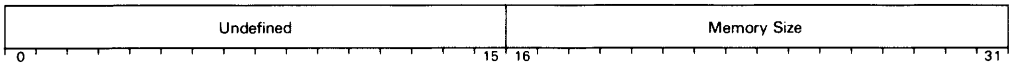
Returned in AC0:



Returned in AC1:



Returned in AC2:



AC#	Name	Contents or Function
0	Model Number	Binary representation of the machine's model number (001000100011100_2)
	FPU	0 indicates no FPU option 1 indicates FPU option
1	Microcode Revision	**Bits** **Meaning** 16 Always set to 1 17-23 Reserved for future use 24-31 Current microcode revision
2	Memory Size	Amount of physical memory available: A 0 indicates 32 Kbytes A 1 indicates 64 Kbytes; etc.

NOTE: *If AC1 contains 177777_8, you should load the microcode.*

Appendix D
Reserved Memory Locations and Context Block Format

This appendix describes the reserved memory locations (see Tables D.1 and D.2), and the context block formats (see Table D.3).

Reserved Memory Locations

The processor reserves memory locations 0 through 47_8 of page zero (locations 0 through 377_8) of each segment for storage of certain parameters and fault handler addresses. The processor translates these locations as shown in Tables D.1 and D.2.

Some of the pointers are 16 bits long, which means that they can only refer to locations in the first 64 Kbytes of the segment containing the pointer. If the pointer is indirect, all pointers in the indirect chain can also only refer to the first 64 Kbytes of the segment.

Page Zero Locations for Segment 0

When an interrupt occurs, segment 0 locations 0 through 47_8 have the meanings listed in Table D.1.

With the address translator enabled, the processor interprets all locations as logical.

Word	Name	Contents or Function
0	Interrupt Level	Level of interrupt processing; 0 indicates base-level processing; non-zero indicates intermediate-level processing
1	I/O Handler	Address of the I/O interrupt handler; indirectable
2-3	I/O Return Address	Address of the I/O interrupt return (word 2 contains the high order; word 3 contains the low order)
4	Vector Stack Pointer	Low-order 16 bits of vector stack pointer, base, and frame pointer; high-order bits are zeroes
5	Current I/O Mask	Current interrupt priority mask
6	Vector Stack Limit	Low-order 16 bits of vector stack limit
7	Vector Stack Fault Address	Address of the vector stack fault handler; indirectable
10-11	Breakpoint Address	Address of the breakpoint handler; indirectable
12-13	**WXOP** Origin Address	Address of the beginning of the MV/4000 extended operations table; indirectable
14	MV/4000 Stack Fault Address	Address of the MV/4000 stack fault handler; indirectable
15-17	Reserved	Reserved
20-21	WFP	MV/4000 frame pointer; nonindirectable
22-23	WSP	MV/4000 stack pointer; nonindirectable
24-25	WSL	MV/4000 stack limit; nonindirectable
26-27	WSB	MV/4000 stack base; nonindirectable
30-31	MV/4000 Page Fault Handler	Address of the MV/4000 page fault handler; indirectable
32-33	Context Block Pointer	Address of the base of context block save area; indirectable
34-35	WGP	Wide gate pointer; address of the gate array; nonindirectable
36	Protection Fault Handler Address	Address of the protection fault handler; indirectable
37	Fixed-Point Fault Handler Address	Address of the fixed-point fault handler; indirectable
40	Stack Pointer	Address of the top of the C/350 stack; nonindirectable
41	Frame Pointer	Address of the start of the current C/350 frame minus 1; nonindirectable
42	Stack Limit	Address of the last normally usable location in the C/350 stack
43	C/350 Stack Fault Address	Address of the C/350 stack fault handler; indirectable
44	**XOP0** Origin Address	Address of the beginning of the C/350 extended operations table
45	Floating-Point Fault Address	Address of the floating-point fault handler; indirectable
46	Decimal/ASCII Fault Handler	Address of the Decimal/ASCII fault handler; indirectable
47	DERR Error Handler	Address of the DERR error/trap handler; nonindirectable

Table D.1 Page zero locations for segment 0

Page Zero Locations for Segments 1 through 7

Table D.2 shows the page zero locations for segments 1 through 7 with the address translator enabled.

Word	Name	Contents or Function
0-7	Reserved	Reserved
10-11	MV/4000 Breakpoint Address	Address of the MV/4000 breakpoint handler; indirectable
12-13	**WXOP** Origin Address	Address of the beginning of the MV/4000 extended operations table; indirectable
14	MV/4000 Stack Fault Address	Address of the MV/4000 stack fault handler; indirectable
15-17	Reserved	Reserved
20-21	WFP	MV/4000 frame pointer; nonindirectable
22-23	WSP	MV/4000 stack pointer; nonindirectable
24-25	WSL	MV/4000 stack limit; nonindirectable
26-27	WSB	MV/4000 stack base; nonindirectable
30-33	Reserved	Reserved
34-35	WGP	Wide gate pointer; address of the gate array; nonindirectable
36	Reserved	Reserved
37	Fixed-Point Fault Handler Address	Address of the fixed-point fault handler; indirectable
40	Stack Pointer	Address of the top of the C/350 stack; nonindirectable
41	Frame Pointer	Address of the start of the current C/350 frame minus 1; nonindirectable
42	Stack Limit	Address of the last normally usable location in the C/350 stack
43	C/350 Stack Fault Address	Address of the C/350 stack fault handler; indirectable
44	**XOP0** Origin Address	Address of the beginning of the C/350 extended operations table
45	Floating-Point Fault Address	Address of the floating-point fault handler; indirectable
46	Decimal/ASCII Fault Handler	Address of the Decimal/ASCII fault handler; indirectable
47	DERR Error Handler	Address of the DERR error/trap handler; nonindirectable

Table D.2 Page zero locations for segments 1 through 7

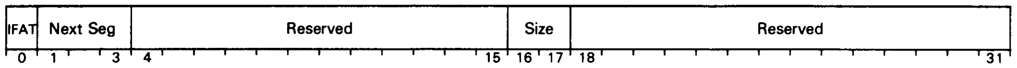
Context Block Format

The context block can be from 15 to 43 double words long. Table D.3 shows the format of the context block.

Words in Block	Contents
0-1	PSR, argument count is zero
2-3	AC0
4-5	AC1
6-7	AC2
8-9	AC3
10-11	CARRY, PC of offending (i.e., executing) instruction
12-13	STATE1 — Doubleword containing segment of next instruction to be executed in bits 1through 3. Bits 16 and 17 contain the context block size. (see below)
14-15	LAR — Address that caused the page fault
16-17	PBXED_OPCODE
18-19	GR0
20-21	GR1
22-23	GR2
24-25	GR3
26-27	MDR
28-29	IR
30-31	STATE2
32-33	Number of micro stack entries
34-35	Contents of micro stack (up to 17 double words)
36-37	GR4
38-39	GR5
40-41	GR6
42-43	GR7
44-45	QREG
46-47	TREG
48-49	STATE3

Table D.3 Context block format

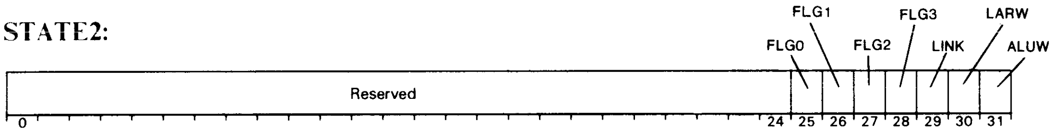
The double word in 12-13 (STATE1) contains the segment of the next instruction to be executed. The processor uses it to resolve on a microcycle basis the segment in which the instruction is actually executing. Since most instructions cannot cross segment boundaries, this double word reflects the same segment as the program counter of the executing instruction. **STATE1:**



Size defines the context block size:

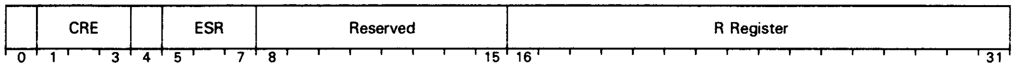
Bit 16	17	Meaning
1	1	Block size 1 (9 double words)
0	0	Block size 2 (18 - 34 double words)
0	1	Block size 3 (25 - 41 double words)
1	0	Block size 4 (reserved)

STATE2:



FLG0 = FLAG0
FLG1 = FLAG1
FLG2 = FLAG2
FLG3 = FLAG3
LARW = LAR_WIDTH
ALUW = ALU_WIDTH

STATE3:



For instructions/operations that can cross inward segment boundaries (LCALL, XCALL, and processor-initiated calls for interrupts, protection faults, etc.), the processor changes the segment field to reflect the inner segment before the processor makes any modification to that inner segment's wide stack or its page zero parameters.

For instructions/operations that can cross outward segment boundaries (WRTN, WRSTR, WPOPB, and processor-initiated returns from interrupts, protection faults, etc.), the segment field reflects the inner segment until the processor makes all modifications to that inner segment's wide stack and its page zero parameters. The processor then changes the segment field to reflect the outer segment before the processor makes any modifications to the outer segment's wide stack or its page zero parameters.

All other words in the context block contain information used by the microcode and other internal systems. The context block does not save the floating-point state. To save this information, use a *Push Floating-Point State* instruction.

Appendix E
Standard I/O Device Codes

Octal Device Codes	Mnem	Priority Mask Bit	Device Name	Octal Device Codes	Mnem	Priority Mask Bit	Device Name
00	--	--	Reserved	40			
01				41			
02				42			
03		--	Reserved	43	PIT	6	Programmable interval timer
04	UPSC	13	Universal Power Supply Controller	44			
05		--	Reserved	45	SCP	14	System control program
06	MCAT	12	Multiprocessor adapter transmitter	46	MCAT1	12	Second multiprocessor transmitter
07	MCAR	12	Multiprocessor adapter receiver	47	MCAR1	12	Second multiprocessor receiver
10	TTI	14	TTY input	50	IAC1	11	Intelligent asynchronous controller 1
11	TTO	15	TTY output	51	IAC2	11	IAC2
12				52	IAC3	11	IAC3
13				53	IAC4	11	IAC4
14	RTC	13	Real-time clock	54	IAC5	11	IAC5
15				55	IAC6	11	IAC6
16				56	IAC7	11	IAC7
17	LPT	12	Line printer	57	LPT1	12	Second Line Printer
20				60			
21				61			
22	MTB	10	Magnetic tape	62	MTB1	10	Second magnetic tape
23				63			
24				64			
25				65	IAC	11	Host to IAC interface
26	DKB	9	Fixed-head DG/Disk	66	DKB1	9	Second fixed-head DG/Disk
27	DPF	7	DG/Disk storage subsystem	67	DPF1	7	Second DG/Disk storage subsystem
30				70			
31				71			
32				72			
33	DKP	7	Moving head disk	73	DKP1	7	Second moving head disk
34	ISC	4	Intelligent synchronous controller	74			
35				75			
36				76			DCU to host interface
37				77	CPU	--	CPU and console functions

Table E.1 Standard I/O device codes

Appendix F
Fault Codes

Tables F.1 through F.3 contain an explanation of the fault codes returned in AC1 for protection, page, stack, and decimal/ASCII faults.

Protection Faults

Table F.1 lists the meanings of the codes returned in AC1 when an MV/4000 address translator protection fault occurs.

AC1 Code (octal)	Meaning
0	Read violation
1	Write violation
2	Execute violation
3	Validity bit protection (SBR or PTE)
4	Inward address reference
5	Defer (indirect) violation
6	Illegal gate -- out of bounds or gate bracket access violation
7	Outward call
10	Inward return
11	Privileged instruction violation
12	I/O protection violation
14	Invalid microinterrupt return block

Table F.1 Protection fault codes

Page Faults

Table F.2 lists the page fault codes that the processor stores in AC1.

AC1 Code	Meaning
0	Multiple ERCC FAULT
1	Page table depth
2	Page table page fault
3	Reserved
4	Normal object reference

Table F.2 Page fault codes

Stack Faults

Table F.3 lists the meanings of the wide stack fault codes. The processor does not return an error code for a narrow stack fault.

AC1 Codes	Meaning
000000	Oveflow on every stack operation other than **SAVE**, **WMSP** or ring crossing
000001	Underflow or overflow would occur if the instruction were executed — **WMSP**, **WSSVR**, **WSSVS**, **WSAVR**, **WSAVS** (PC in return block refers to the instruction that caused the stack fault)
000002	Too many arguments on a cross ring call
000003	Stack underflow
000004	Overflow due to a return block pushed as a result of a microinterrupt or fault

Table F.3 Stack fault codes

Decimal/ASCII Faults

Table F.4 lists the decimal/ASCII faults. The first and second columns give the code that appears in AC1 when either a C/350 or MV/4000 computer fault occurs. The third column lists the instruction that caused the fault, while the last column describes the conditions that could cause the fault.

Code Returned		Faulting Instruction	Meaning
C/350	MV/4000		
000000	100000	**EDIT, WEDIT**	An invalid digit or alphabetic character encountered during execution of one of the following subopcodes: **DMVA, DMVF, DMVN, DMVO, DMVS**
000001	100001	**LDIX, STIX**	Invalid data type (7)
		EDIT, WEDIT	Invalid data type (6 or 7)
000002	100002	**EDIT, WEDIT**	**DMVA** or **DMVC** subopcode with source data type 5; AC2 contains the data size and precision
000003	100003	**EDIT, WEDIT**	Invalid opcode; AC2 contains the data size and precision
000004	100004	**WLDI, WSTI, WSTIX**	Number too large to convert to specified data type
000006	100006	**WLSN, WLDI, WLDIX, EDIT, WEDIT**	Invalid sign code for this data type
000007	100007	**WLSN, WLDI, WLDIX**	Invalid digit

Table F.4 Decimal/ASCII faults

Appendix G
Load Control Store Instruction

This appendix presents the operation and format for the *Load Control Store* instruction and its associated microcode file.

> **WARNING:** *The* **Load Control Store** *instruction changes various parts of the machine's internal state. This instruction is intended for diagnostic and special system applications.*

Load Control Store
LCS
NIO 2,CPU

0	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The *Load Control Store* instruction loads and verifies the soft internal states of the machine (microstore, decode rams, scratch pad, etc.). In conjunction with bits 16 through 31 of three accumulators (AC0, AC1, AC2), the **LCS** instruction performs a load and verify, or verify only, using the contents of a microcode file.

AC0 contains the load and verify, or verify only, argument, and the destination code; AC1 contains the bit length of the code data; and AC2 contains a pointer to the first block of data.

> **NOTE:** *The* **LCS** *instruction loads a maximum of 16K words with each instruction. Therefore, it may be necessary to use multiple* **LCS** *instructions.*
>
> *This instruction is noninterruptible.*

The call sequence for the **LCS** instruction is:

LCS
error return
normal return

The formats for the three accumulators are as follows:

AC0

Unused	L/V	Destination Code
0 15	16	17 31

AC1

Unused	Bit Length
0 15	16 31

AC2

Unused	Pointer
0 15	16 31

AC #	Contents	Meaning
0	L/V	Load/verify option 0 implies load and verify 1 implies verify only
	Destination Code	Code for where the data is to be loaded.
1	Bit Length	Bit length of code data
2	Pointer	Pointer to first block of data (nonindirectable)

The steps for LOAD and VERIFY are:

1. Parse microcode file blocks: Load Code blocks, fill Fill blocks, ignore Revision blocks, print Comment blocks.

   Repeat this sequence until an End block is encountered.

2. Verify Code blocks that were loaded in step 1, ignore Fill, Comment, and Revision blocks.

If an End block is encountered, the **LCS** instruction is completed.

The sequence of events for the VERIFY ONLY is step 2 of the Load and Verify.

Microcode File Format

The microcode file format contains data for use in various parts of the machine's state. The microcode format is a block-oriented format (arranged into packets or blocks) that contains a description of the size of the block and the type of data it contains.

The general format for each microcode file is shown in Figure G.1.

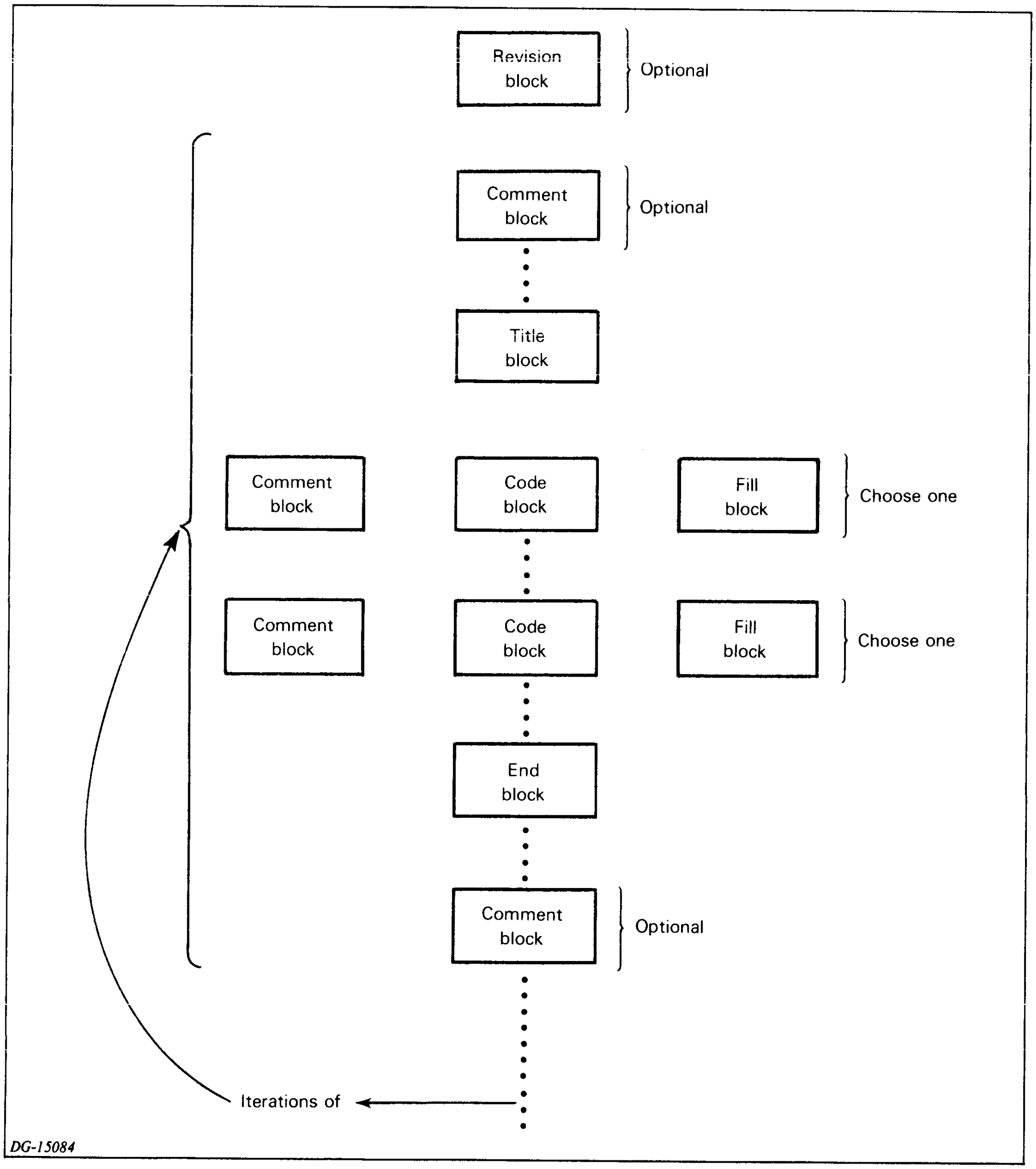


Figure G.1 General formats for microcode files

Microcode Block Format

Each microcode file must begin with a Title block and finish with an End block (Title/End block pair). Fill and Code blocks must be placed between the Title/End block pair. The Revision block preceeds the first Title block. Comment blocks may appear anywhere within the microcode file.

Title blocks contain data pertaining to the code word's bit length, and the destination code. The program issuing the LCS instruction should use the data from the Title block as the data for AC0 and AC1.

End blocks contain the necessary data to either continue execution or terminate the LCS instruction.

Code blocks contain code words and the starting location for storing each code word. Code blocks must appear between a Title/End block pair.

Fill blocks contain code words for use as background filler and the locations to receive this data. Fill blocks must appear between a Title/End block pair.

Comment blocks contain data that may be output to the system console (or ignored). Comment blocks may appear anywhere within the microcode file structure. If the Comment block appears within the Title/End block pair (internal), the data is output to the system console; if the Comment block appears outside the Title/End block pair (external), the program issuing the LCS instruction decides whether to output or ignore the data.

Revision blocks contain the target CPU model number and the microcode major and minor revision numbers. Revision blocks should appear as the first block of the microcode file. The program issuing the LCS instruction determines whether the Revision blocks are ignored or output to the system console.

LCS Implementation

The LCS instruction performs the following functions:

- Recognizes Code blocks, and loads the data contained into the proper destination addresses.
- Recognizes internal Comment blocks, and prints the text string on the system console.
- Recognizes Fill blocks, and performs a fill operation of the proper destination.
- Recognizes End blocks, and performs a Verify operation upon the previously loaded data.
- Recognizes any of five error conditions (see Error Return) and returns the proper error code to AC0.

NOTE: *The* LCS *instruction operates on Code, Comment, Fill, and End blocks as described above. The program issuing the* LCS *instruction must parse out, and set up, the information from the Title, Revision, and any external Comment blocks.*

Microcode Blocks

The general form of each microcode block is shown in Figure G.2.

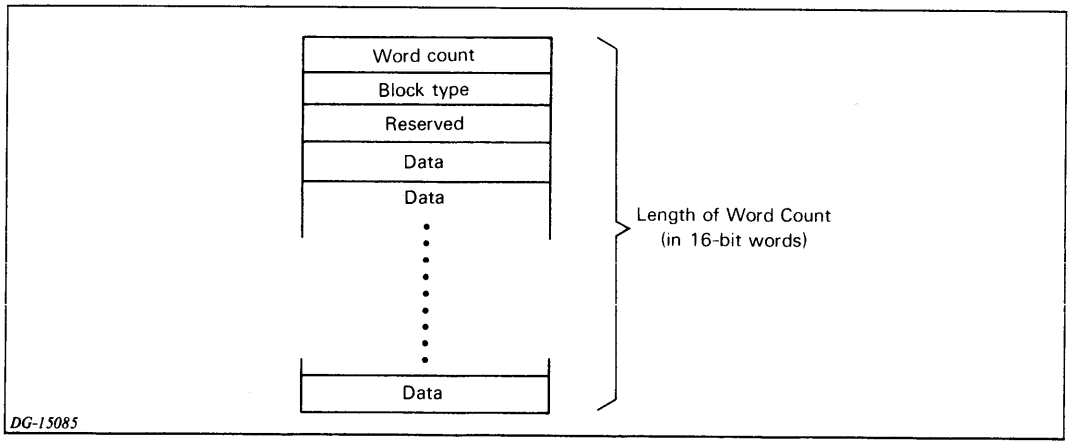


Figure G.2 General form for microcode blocks

The first word of each block is the Word Count (the number of 16-bit words in the microcode block).

The second word of each block is the Block Type (Title, End, Code, Fill, Comment, and Revision) indicating the type of data contained in the block.

The third word is Reserved for future use.

The remaining words contain the Data pertaining to the block type.

The formats for the specific blocks are:

TITLE

Format:

Word Count	7
Block Type	0
Reserved	
Data Word 1	Code word's bit length
Data Word 2	Reserved for future use
Data Word 3	Reserved for future use
Data Word 4	Destination (code for where the data is to be loaded). Only positive, nonzero, 16-bit integers, in the range 1 through 77777_8 are accepted by the processor.

The data from the first Title block is used by the program issuing the **LCS** instruction. For example:

AC0 ← Data Word 4 (Destination)

AC1 ← Data Word 1 (Code word's bit length)

END

Format:

Word Count 5
Block Type 1
Reserved
Data Word 1 Control word

Bits	**Meaning**
0-12	Reserved
13	Destination completion indicator 0 indicates more code of this destination may follow 1 indicates no more code
14	Switch from PROM to RAM Control Store 0 indicates to stay in current mode 1 indicates switch to RAM
15	Start designator 0 indicates start Host (and continue SCP) 1 indicates start Master (SCP); Data Word 2 must be an address

Data Word 2 Address that is to be started:

NOTE: *If this is -1 (177777_8), continue execution with the* LCS *normal/error return.*

The following chart summarizes the combined actions of Data Word 1 (bit 15) and Data Word 2:

Data Word 2 Contains	**Data Word 1 (bit 15)**	
	0	**1**
-1	Continue Host at LCS normal/error return	Illegal
Address	Start Host at this address; continue Master	Start Master at this address; Host remains halted

CODE

Format:

Word Count	Variable
Block Type	2
Reserved	
Data Word 1	Location for storing the first code word in this block
Data Word 2 to N+1	First code word of the block
Data Word N+2 to 2N+1	Code word for the next sequential address
Data Word 2N+2 to 3N+1	Code word for the next sequential address

.

.

.

Until end of block

> **NOTE:** *Code data is in a word-aligned format: N is the number of 16-bit words that contain one code word [N = (word-bit-length + 15)/16]*

FILL

Format:

Word Count	N+5 [N=(word-bit-length + 15)/16]
Block Type	3
Reserved	
Data Word 1	Starting location for storing code word
Data Word 2	Ending location for storing code word
Data Word 3 to N+2	Code word to be used as background filler

The Fill block allows a method to "background fill" certain destinations of the machine; e.g., zero-fill the control store to induce parity errors if an uninitialized location is erroneously entered during execution.

> **NOTE:** *The Fill functionality may also be accomplished via code blocks of the appropriate data.*

COMMENT

Format:

Word Count	Variable
Block Type	4
Reserved	
Data Word 1	String length The length of the ASCII string (terminating NULL(s) are not counted). An odd string length indicates one terminating NULL; an even string length indicates two terminating NULLs.
Data Word 2 to X+2	ASCII string (packed right to left) terminated by a NULL. [X = (String length + 1)/2]

REVISION

Format:

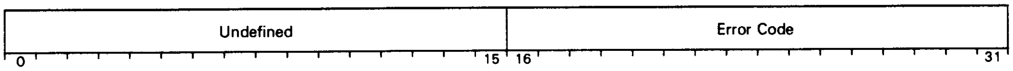
Word Count	6
Block Type	5
Reserved	
Data Word 1	Target CPU model number
Data Word 2	Microcode major revision number
Data Word 3	Microcode minor revision number

Error Return

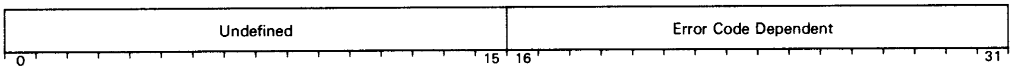
Upon encountering an error, the three accumulators (AC0, AC1, AC2) will contain an indication of the problem.

The formats for the accumulators are:

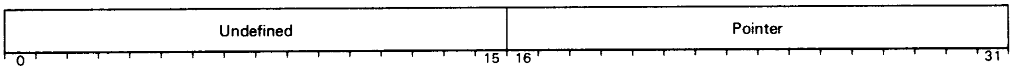
AC0



AC1



AC2



AC #	Contents	Meaning
0	Error Code	Code returned denoting type of error (defined below) **Code Error** 1 Verify error 2 Illegal code word length 3 Unexpected block type 4 Illegal block length 5 Unknown destination
1	Error Code Dependent	If unspecified AC1 is left unchanged
2	Pointer	Pointer to erroneous block NOTE: *If an error occurs because of initial erroneous information in either AC0 or AC1, then AC2 is left unchanged.*

Error codes returned to AC0:

Code	Meaning	Definition (AC1 Contents) (Possible Cause)
1	Verify error	Indicates that the data was not received properly by the destination. (AC1 will contain the code word location that is in error) (Possible hardware problem)
2	Illegal code word length	Code word bit length does not agree with length of code data as specified by the destination word in the same Title block. (AC1 is unchanged) (Possible attempt to load the wrong model microcode)
3	Unexpected block type	Block type other than allowable types (Code, Fill, End, Revision, or Comment) (AC1 is unchanged) (Possible missing block, or out of sequence) NOTE: *If any Title blocks are encountered between the Title/End block pair, the unexpected block type error will be returned.*
4	Illegal block length	Block length is in error (AC1 is unchanged) (Block length of less than four was specified, or the code block did not contain an integral number of code words) For example: If the code word bit length is 80, then the length of all code blocks must be 4+N*(80+15)/16. N = number of code words per code block 16 = number of bits per word 4 = number of words at the beginning of each code block For this example, all code blocks must be of length 4+5*N
5	Unknown destination	Unknown location for loading of code word (AC1 is unchanged) (Possible attempt to load an incorrect model machine microcode file)

Kernel Functionality

The kernel is the minimum set of microcode necessary for the machine to function properly. With the kernel instruction set (including the **LCS** instruction) the processor may read in target microcode from an I/O device (using the kernel I/O instructions) and then load this microcode into the control store using the **LCS** instruction.

Since there is a 16K-word limit to the amount of data that may be loaded with a single **LCS** instruction, it may take several iterations of accessing the I/O device and executing the **LCS** instruction to completely change the machine from the kernel to the target.

> NOTE: *Since the* LCS *instruction must return to the host after completion, the kernel instruction set must exist (in working order) after each execution of the* LCS *instruction.*

Appendix H
Programming Considerations

This appendix lists the machine specific programming/performance considerations.

Current Page of Execution

Writing to the current page of execution -- such as writing to the next word in the instruction stream -- flushes the queue in the instruction prefetcher (within the instruction pipeline).

Double-Word Alignment

The MV/4000 system operates more efficiently if double words are aligned on double-word boundaries.

Index

Within the index, the letter "f" following a page entry indicates "and the following page"; the letters "ff" following a page entry indicate "and the following pages". The letter "t" following a page entry indicates that a table resides on the page.

16-bit
  fixed-point
    accumulators 1-3
    logical format 2-1
    two's complement format 2-1
  initial count register 7-7
  priority mask 7-5ff
32-bit
  fixed-point
  accumulators 1-3
  logical format 2-1
  two's complement format 2-1
4010 controller, model 7-11

A

AC1, error code in 5-7, 5-11, F-2
Access
  bits 8-5
  privileges 8-1
  validation 1-3, 8-3
  violation 8-6
Access, random memory 1-3
Accumulator instruction, execute 5-9t
Accumulators,
  fixed-point 1-3, 1-6, 9-1f
  floating-point 1-3, 1-6, 9-1
Acknowledge, interrupt 7-5ff
Ac-line frequency 7-9f
Addition instructions,
  fixed-point 2-3t
  floating-point 3-2t, 9-4t
Address
  error, BMC 7-22, C-5
  modes, BMC 7-19
  offset C-1
  protection fault 8-4ff
  range, C/350 9-2
  space,
    logical 5-2, 8-5
    physical 5-2, 8-5
  translation 1-6, 7-5ff, 7-19, 8-1, C-4
  translator 1-3, 1-6, 5-2, 5-11, 7-19, 8-1ff,
      B-1, D-1
  translator protection fault F-1
  wraparound 5-1, 9-6
Address,
  byte 5-2
  fault handler D-1
  logical 1-3, 5-1, 7-19, 8-1, C-1
  page zero 7-16f
  physical 1-3, 1-6, 7-15f, 7-19, 8-1
  word 5-2
Addressing,
  C/350 9-2
  indirect 5-2, D-1
ALC device 7-12
ANY flag 3-5, C-2
Area, state 8-8
Argument count 2-8, C-2
Arithmetic processor 1-3
Asynchronous
  communications 1-5, 7-1, 7-11ff
  line controller 1-5, 7-11, E-1
Auto-increment and auto-decrement, C/350 9-6
Available, physical memory 9-6ff

B

Base registers, segment C-1, C-4
Base, time 7-7, 7-9
Battery backup, UPSC 7-27, 7-29f
BCD arithmetic instructions 2-10t
Binary conversion instructions, floating-point 3-4t
Bit(s),
  access 8-5
  BMC validity 7-21
  carry 2-4, 5-11, 9-2
  memory management 1-3, 8-3
  modified 1-3, 8-3
  priority mask 7-1, E-1
  referenced 1-3, 8-3
  stop 7-12
Block
  format, context D-4f
  instructions, wide stack return 4-2t
Block,
  context 8-5f, D-4f
  data 7-16f
  microstate 5-7

- BMC
  - address modes 7-19
  - error 7-22, C-5
  - map 7-19
    - instructions 7-23ff
    - loads/reads 7-23
    - slots 7-24
    - table 7-19
  - modes 7-19
  - registers 7-20ff
  - status register 7-22, C-5
  - transfer flag 7-22, C-5
  - transfer rate 1-4
  - validity 7-21f, C-5
- Buffer,
  - load character 7-13
  - read character 7-12
  - TTI input character 7-11f
- Burst multiplexor channel (see BMC)
- BUSY flag 7-2f, 7-5ff, 7-15ff
- Byte
  - address 5-2
  - address instructions, load effective word and 2-9
  - movement instructions, fixed-point 2-9

C

- C/350
  - addressing range 9-2
  - auto-increment and auto-decrement 9-6
  - compatibility 1-6
  - compatible
    - formats C-1
    - instructions A-1f, B-1ff
  - equivalent instruction 9-4ff
  - fixed-point computing instructions 9-3t
  - floating-point computing instructions 9-4t
  - instruction set 5-11, 9-1ff
  - program flow management instructions 9-5t
  - programming 9-1ff
  - stack 5-11, 9-2, 9-6
- Calculation, time 7-9
- Carry bit 2-4, 5-11, 9-2
- Carry-out 9-2
- CDR device E-1
- Central processor 1-2ff, 7-1, 7-3ff
- Chain, indirect D-1
- Channel,
  - data 1-4, 7-1, 7-19ff
  - I/O 7-24
- Character buffer,
  - input and output 7-11ff
  - load 7-13
  - read 7-12
- Checking, parity 7-11
- Checks, hardware 1-3, 8-3
- CIO and CIOI 7-23f
- Clock frequency 7-9ff
- Clock, real-time 7-1, 7-9f
- Code(s),
  - device 7-1, 7-5, 7-12, E-1
  - error 5-7, 5-11
  - fault F-1ff
  - in AC1, error 5-7
  - protection fault F-1t
  - register, UPSC fault 7-29ff
  - SCP error 7-15f
  - stack fault F-2
  - standard I/O device E-1
- Command field 7-15
- Command, SCP 7-15f
- Communications,
  - asynchronous 1-5, 7-1, 7-11ff
  - controllers 1-4
  - I/O 7-19
  - synchronous 1-5, 7-19
- Compatibility, C/350 1-6, 9-2ff, A-1f, B-1ff, C-1
- Complement, two's 2-1, 7-8
- Computing,
  - fixed-point 2-1ff, 9-3t
  - floating-point 3-1ff, 9-4t
- Console
  - reset 7-16f
  - switches 7-4
- Console, soft system 1-5
- Context block 8-5f, D-4ff
- Control
  - processor, system (see SCP)
  - register, UPSC 7-27
  - store, load G-1ff
- Controller(s),
  - asynchronous line 1-5, 7-11, E-1
  - communications 1-4
  - device 7-19
  - intelligent synchronous 1-5, E-1
  - model 4010 7-11
  - synchronous line 1-5, 7-19
  - universal power supply 1-5, 7-1, 7-26ff, E-1
- Conversion and store instructions, fixed-point to floating-point 2-9
- Count register, PIT 7-7f
- Count,
  - argument 2-8, C-2
  - specify C/350 BMC map word 7-9
- Counter,
  - floating-point program 3-5, C-2
  - PIT 7-8f
  - program 1-3, 1-6, 5-1f, 5-7, 9-1f, 9-6, C-1f
- Counting cycle 7-8
- CPU
  - device 7-3ff, E-1
  - error reporting 7-15
  - identification 8-4, 9-6ff, C-1, C-6f
  - instructions 7-4ff
  - skip 7-7, A-2
- CPU, I/O instructions for 7-4t
- Crossing,
  - ring 5-2, 5-7, 5-10t, 9-6
  - segment 8-3, 8-5

Current microcode revision 9-6ff, C-7
Cycle, counting 7-8

D

Data
  block 7-16f
  channel 1-4, 7-1, 7-19ff
    transfer rate 1-4
  error, BMC 7-22, C-5
  format,
    fixed-point 2-1
    floating-point 3-1
  movement instructions,
    fixed-point 2-2, 9-3t
    floating-point 3-4t, 9-4t
  path between MV/4000 and UPSC, verify the 7-27
DCH 1-4, 7-19, 7-24
  map
    instructions 7-23ff
    loads/reads 7-23
    slots 7-24
  mode 7-22, C-5
  registers 7-20ff
  status register 7-22, C-5
  validity error flag 7-22, C-5
DCH/BMC
  maps 7-19f
  status register 7-21ff
Decimal conversion instructions, floating-point 3-4t
Decimal/ASCII fault 5-7, 5-11, F-3
Decimal/byte instructions 2-9
Decoder, instruction 1-2f
Defer violation 8-6
Definition register, port 7-21f, C-5
Demand paging 1-3
Depth, page table 8-5
Device
  codes 7-1, 7-5, 7-12, E-1
  controller 7-19
  flags,
    general 7-2t
    CPU 7-3ff
    PIT 7-8
    RTC 7-10
    SCP 7-13
    TTI 7-12
    TTO 7-12
    UPSC 7-26ff
  instructions, TTO 7-13
  management 7-1ff
  map registers 7-20t, 7-21
  mnemonics 7-1, E-1
  name E-1
Device,
  ALC 7-12
  CDR E-1
  CPU 7-3ff, E-1
  DCH 1-5
  IAC 1-5, E-1
  ISC 1-4f, E-1
  PIT 7-7, E-1
  PLT E-1
  RTC 7-9f, E-1
  SCP 1-5, 7-13, E-1
  TTI 7-11, E-1
  TTO 7-11, E-1
  UPSC 1-5, 7-26ff, E-1
Diagnostic
  functions, SCP 1-5
  mode 7-15ff
  sequence 7-15ff
  test 7-16f
    register, UPSC 7-27f
Diagnostic, power-up 1-5, 7-26ff
Disable CPU error reporting 7-16f
Disable, interrupt 7-6
Division instructions,
  fixed-point 2-4t
  floating-point 3-3t, 9-4t
DONE flag 7-2f, 7-5ff, 7-15ff, C-5
Double
  precision floating-point 3-1
  words 1-3
DVZ flag 3-5, C-2

E

ECLID 8-4, 9-6, A-2, C-6
EDIT subprogram instructions 2-10t
Enable
  flag, interrupt 7-3
  SCP command 7-15f
Enable, interrupt 7-7
Enable/disable error reporting instruction 7-15
Enter diagnostic sequence 7-15ff
Entry format, page table 8-2f
Entry, page table 8-5
Equivalent instruction, C/350 9-4ff
ERCC error 1-4, 7-18f, F-2
Error
  code in AC1 5-7, 5-11, F-2
  code, SCP 7-15f
  flag, BMC and DCH validity 7-22, C-5
  logging 7-18f
  reporting, CPU 7-15
  status, extended 7-15f
Error,
  BMC 7-22, C-5
  ERCC 1-4, 7-18f, F-2
Errors, isolate hardware 7-13
Execute
  access violation 8-6
  accumulator instruction 5-9t
Execution times, instruction B-1ff
Exponent 1-3, 3-1
Extended error status 7-15f

F

Failure faults, power 1-5, 7-26ff
Fault
code register, UPSC 7-29ff
codes F-1ff
handler 8-3, D-1
sequence, page 8-6
Fault(s),
address
protection 8-4ff
translator protection F-1
decimal/ASCII 5-7, 5-11, F-3
fixed-point 5-11, 9-2, B-1, C-2
floating-point 5-7, 5-11, B-1, C-2
handling 5-11, 8-8
nonprivileged 5-11
page 1-3, 5-11, 8-3, 8-5f, F-2
privileged 5-11, 8-4ff
protection 1-3, 5-7, 8-3, F-1
protection violation 5-11, 8-6
stack 5-7, 5-11, F-2
Faults, power failure 1-5, 7-26ff
Fixed-point
accumulators 1-3, 1-6, 9-1f
addition instructions 2-3t
byte movement instructions 2-9
computing 2-1ff
computing instructions, C/350 9-3t
data format 2-1
data movement instructions 2-2, 9-3t
division instructions 2-4t
fault 5-11, 9-2, B-1
increment or decrement word and skip instructions 2-6t
instructions 2-2ff
logical
formats 2-1
skip instructions 2-7t
multiplication instructions 2-4t
overflow fault 5-11
precision conversion instructions 2-2
skip on condition instructions 2-5t
subtraction instructions 2-3t
to floating-point conversion and store instructions 2-9
two's complement formats 2-1
Flag(s),
ANY 3-5, C-2
BMC
transfer 7-22, C-5
validity error 7-22, C-5
BUSY 7-2f, 7-5ff, 7-15ff
CARRY 2-4, 9-2, D-4f
CPU device 7-3ff
DCH validity error 7-22, C-5
DONE 7-2f, 7-5ff, 7-15ff, C-5
DVZ 3-5, C-2
interrupt
enable 7-3, 7-7
on 7-2
request 7-3, 7-8ff
ION 5-2, 7-2, 7-5ff, 9-6
IRES 2-8, 5-6f, C-2
IXCT 2-8, 5-6f, C-2
MOF 3-5, C-2
N 3-5, C-2
OVF 3-5, C-2
OVK 2-8, C-2
OVR 2-8, C-2
PIT device 7-8
RND 3-5, C-2
RTC device 7-10
SCP device 7-13
TE 3-5, C-2
TTI device 7-12
TTO device 7-12
UNF 3-5, C-2
Z 3-5, C-2
Flags for
general devices, device 7-2t
skip instruction, device 7-2t
Flags, UPSC device 7-26ff
Floating-point
accumulators 1-3, 1-6, 9-1
addition instructions 3-2t, 9-4t
binary conversion instructions 3-4t
computing 3-1ff, 9-4t
conversion and store instructions, fixed-point to 2-9
data
format 3-1
movement instructions 3-4t, 9-4t
decimal conversion instructions 3-4t
division instructions 3-3t, 9-4t
fault 5-7, 5-11, B-1
function instructions 9-4
instructions 3-2ff
model identification C-2
multiplication instructions 3-2t, 9-4t
precision 3-1
program counter 3-5, C-2
skip on condition instructions 3-3t
state D-4f
status register 1-3, 1-6, 3-5t, 9-1f, C-1f
subtraction instructions 3-2t, 9-4t
unit 9-6ff, C-6f
Flow management instructions, C/350 program 9-5t
Flow, program 5-1ff, 9-6
Format(s),
C/350 compatible C-1
context block D-4f
fixed-point data 2-1
floating-point data 3-1
page table entry 8-2f
register C-1ff
FPMOD 3-5, C-2
FPSR 1-3, 3-5t, 7-5ff, 9-2, C-1
FPSR instructions 3-5t, 9-4t
Frame
number, page 8-8
pointer, wide 1-3, 8-5f

Frequency,
ac-line 7-9f
clock 7-9ff
select RTC 7-10f
Function instructions, floating-point 9-4
Functions,
SCP diagnostic 1-5
system control 1-5

G

General
devices, device flags for 7-2t
I/O instructions 7-1f

H

HALT instruction 7-6, A-2
Halt, processor 8-5
Handler,
address D-1
fault 5-11, 8-8
interrupt 5-2, 5-7
page fault 8-3, 8-6
protection fault 8-8
Hardware
checks 1-3, 8-3
errors, isolate 7-13
reserved memory 8-8
Hex shift instructions 2-10t

I

IAC device 1-5, E-1
Identification,
CPU 8-4, 9-6ff, C-1, C-6f
floating-point model C-2
Increment or decrement word and skip instructions, fixed-point 2-6t
Indirect
address 5-2, D-1
violation 8-6
Indirection B-1
Initial
count register, PIT 7-7ff
map register instruction, specify 7-22, C-5
Initialization 1-6
Initialize CARRY instructions 2-4t
Input character buffer, TTI 7-11f
Instruction
compatibility 9-2ff
decoder 1-2f
execution times B-1ff
processor 1-2f
set, C/350 5-11
summary A-1f
violation 8-6
Instruction(s),
BCD arithmetic 2-10t
BMC map 7-23ff
C/350
compatible A-1f, B-1ff
equivalent 9-4ff
fixed-point computing 9-3t
floating-point computing 9-4t
program flow management 9-5t
stack management 9-6
CPU 7-4ff
identification 9-6, C-6f
skip 7-7
DCH map 7-23ff
decimal/byte 2-9
device flags for skip 7-2t
enable/disable error reporting 7-15
execute accumulator 5-9t
fixed-point 2-2ff
addition 2-3t
byte movement 2-9
data movement 2-2, 9-3t
division 2-4t
increment or decrement word and skip 2-6t
logical skip 2-7t
multiplication 2-4t
precision conversion 2-2
skip on condition 2-5t
subtraction 2-3t
to floating-point conversion and store 2-9
floating-point 3-2ff
addition 3-2t, 9-4t
binary conversion 3-4t
data movement 3-4t, 9-4t
decimal conversion 3-4t
division 3-3t, 9-4t
function 9-4
multiplication 3-2t, 9-4t
skip on condition 3-3t
subtraction 3-2t, 9-4t
FPSR 3-5t, 9-4t
general I/O 7-1f
HALT 7-6, A-2
hex shift 2-10t
initializing CARRY 2-4t
INTA 7-5ff, 7-23, A-2, C-5, E-1
INTDS 7-6, A-2
INTEN 7-7, A-2
interrupt
acknowledge 7-5ff, 7-23, A-2, C-5, E-1
disable 7-6
enable 7-7
interrupting an 5-4ff
IORST 7-26ff, A-2
jump 5-9t
load
character buffer 7-13
CPU identification 8-4, C-6
effective word and byte address 2-9
logical 2-6t
logical shift 2-7t
mask out 7-6
memory/system management 8-4
MSKO 7-6, A-2
narrow load CPU identification 8-4

NCLID A-2
NIO 7-2
noninterruptible 5-4
PBX 5-7
PIT 7-8ff
privileged 7-24
program flow 5-9ff
PSR manipulation 2-8
queue 6-1t
read
character buffer 7-12
count 7-8f
data from UPSC 7-29
switches 7-4
READS 7-4, A-2
request data from UPSC 7-29
reset 7-5ff, 7-10f, 7-14ff, 7-23, A-2, C-5
restartable 5-5
resumable 5-6ff
return SCP status 7-18f
RTC 7-10f
SCP 7-14ff
segment transfer 5-10t
select RTC frequency 7-10f
skip 5-9t
specify
initial count 7-9
initial map register 7-22, C-5
SSPT 7-19, 8-8, A-2
store state pointer 7-19, 8-8, A-2
subroutine 5-10t, 5-10t
TTI 7-12f
TTO device 7-13
type 1 or 2 5-2
UPSC 7-26ff
wide stack 4-1ff
WLMP A-2
write data to UPSC 7-26ff
Instructions for
CPU, I/O 7-4t
RTC, I/O 7-10t
TTI and TTO, I/O 7-12t
INTA instruction 7-5ff, 7-23, A-2, C-5, E-1
INTDS instruction 7-6, A-2
Intelligent
asynchronous controller 1-5, 7-11, E-1
synchronous controller 1-5, E-1
INTEN instruction 7-7, A-2
Internal processor state 5-7f
Interrupt(s) 5-2, 5-6f, 8-8, 9-6, C-6
acknowledge instruction 7-5ff, 7-23, A-2, C-5, E-1
disable instruction 7-6
enable
flag 7-3, 7-7
instruction 7-7
handler 5-2, 5-7
on flag 5-2, 7-2, 7-5ff, 9-6
request 7-6
request flag 7-3, 7-8ff
sequence 5-2f
system 7-5ff
Interrupting an instruction 5-4ff
Interrupts,
I/O 7-9
program 7-7, 7-12
RTC 7-10
SCP 7-16f
UPSC power fail 7-27, 7-29
Interval timer, programmable 7-1, 7-7ff
Inward reference violation 8-6
I/O
channel 1-4, 7-24
communication 7-19
device codes, standard E-1
instruction violation 8-6
instructions for
CPU 7-4t
RTC 7-10t
TTI and TTO 7-12t
instructions, general 7-1f
interrupts 7-9
port 7-24
protection C-4
reset 1-6
system 1-4
transfers 1-4
IOC status register 7-5ff, 7-23, C-5
ION flag 5-2, 7-2, 7-5ff, 9-6
IORST 1-6, 7-5ff, 7-10f, 7-14ff, 7-23, C-5
IORST instructions 7-26ff, A-2
IRES flag 2-8, 5-6f, C-2
ISC device 1-4f, E-1
Isolate hardware errors 7-13
IXCT flag 2-8, 5-6f, C-2

J

Jump instructions 5-9t

L

LCPID 8-4, C-6
LCS G-1ff
LEDs, UPSC 7-30ff
LEF mode C-4
Line frequency 7-10
Line, asynchronous 7-11ff
Load
character buffer instruction 7-13
control store G-1ff
CPU identification instruction 8-4, C-6
effective word and byte address instructions 2-9
map, wide 7-24f
Loads/reads, BMC map or DCH map 7-23
Location 1, physical 5-2
Locations for
segment 0, page zero D-1
segments 0 through 7, page zero D-1
Locations, reserved memory 8-8, 9-2, D-1ff
Logging, error 7-18f

Logical
address 1-3, 5-1, 7-19, 8-1, C-1
space 5-2, 8-5
translation 1-6
format 2-1
instructions 2-6t
location 1 5-2
memory 1-3
mode, BMC 7-19
page 7-19
shift instructions 2-7t
skip instructions, fixed-point 2-7t

M

Main memory 7-19
Management
bits, memory 8-3
instructions,
C/350 program flow 9-5t
C/350 stack 9-6
registers, stack 9-2
Management,
device 7-1ff
memory and system 8-1ff
program flow 5-1ff
queue 6-1
stack 4-1ff
Mantissa 1-3, 3-1
Map
instructions, BMC and DCH 7-23ff
registers 7-19ff
registers, device 7-21
slots, BMC and DCH 7-19, 7-24
table, BMC 7-19
Map,
BMC 7-19
wide load 7-24
Mapped transfer 7-19
Maps, DCH/BMC 7-19f
Margining register, UPSC power 7-27, 7-29f
Mask
bit assignments 7-1, E-1
out instruction 7-6
Mask, interrupt 7-5ff
Memory
access, random 1-3
allocation and protection, IAC and ISC 1-5
and system management 8-1ff
available, physical 9-6ff
locations, reserved 8-8, 9-2, D-1ff
management bits 1-3, 8-3
modules 1-3
sniffing 1-4
system 1-3
Memory,
hardware reserved 8-8
logical 1-3
main 7-19
physical 1-3, 8-3, 8-6, 9-6ff, C-6
reserved 8-8f, 9-6
Memory/system management instructions 8-4
Microcode 9-6ff, C-6f, D-4f
Microcode, system 1-6
Microinterrupt 5-7
Microstate block 5-7
Mnemonic, device 7-1, E-1
Mode,
BMC logical and physical 7-19
DCH 7-22, C-5
diagnostic 7-15ff
LEF C-4
physical 1-6, 5-1f
SCP power down 7-15ff
Model
4010 controller 7-11
identification, floating-point C-2
number 9-6ff, C-6f
Modes, BMC address 7-19
Modified bits 1-3, 8-3
Module, memory 1-3
MOF flag 3-5, C-2
MSKO instruction 7-6, A-2
Multiplication instructions,
fixed-point 2-4t
floating-point 3-2t, 9-4t
Multiprogram operating systems 7-7

N

N flag 3-5, C-2
Name, device E-1
Narrow
load CPU identification instruction 8-4
stack 5-11, 9-2, 9-6
NCLID 8-4, 9-6ff, A-2, C-6
NCPID 9-6ff
NIO instruction 7-2
Noninterruptible instruction 5-4
Nonprivileged faults 5-11, 8-8
No-op 7-24

O

Octal device code E-1
Odd-numbered register formats, BMC or DCH 7-21
Offset, address C-1
One-level page table B-1
Operating systems, multiprogram 7-7
Operations, decimal/byte 2-9
Operator's terminal 1-5
Output character buffer, TTO 7-12f
Overflow fault C-2
Overflow, stack 5-7, B-1
Overview, system 1-1ff
OVF flag 3-5, C-2
OVK flag 2-8, C-2
OVR flag 2-8, C-2

P

Page
  fault 1-3, 5-11, 8-3, 8-5f, F-2
  frame number 8-8
  number 7-19, 7-21
  table
    entry format 8-2f
    validity violation 8-6
  table, one-level or two-level 8-6, B-1
  tables 8-1, B-1
  validation 1-3, 8-3
  zero 5-2, 7-16f, 8-5, 8-8, D-1
    address 7-16f
    locations for segments 0 through 7 D-1
Page-swapping 1-3
Paging, demand 1-3
Parity checking 7-11
Path between MV/4000 and UPSC, verify the data 7-27
PBX instruction 5-7
PC 1-3, 1-6, 5-1f, 5-7, 9-1f, 9-6, C-1f, D-4f
Physical
  address 1-3, 1-6, 7-15f, 7-19, 8-1
  address space 5-2, 8-5
  location 1 5-2
  memory 1-3, 8-3, 8-6, 9-6ff, C-6
  memory available 9-6ff
  mode 1-6, 5-1f
  mode, BMC 7-19
  page 7-19
  page number 7-19, 7-21
Pipeline 1-2f, 5-1
PIT
  counter 7-8f
  device 7-7, E-1
  initial count register 7-7ff
  instructions 7-8ff
PLT device E-1
Pointer D-1, D-4f
Pointer,
  state 8-8
  wide frame and stack 1-3, 8-5f
Port
  definition register 7-21f, C-5
  status register 7-21, 7-23, C-5
Port, I/O 7-24
Power
  and status, system 1-5, 7-26ff
  down mode, SCP 7-15ff
  fail 7-16f
  fail interrupts, UPSC 7-27, 7-29
  failure faults 1-5, 7-26ff
  margining register, UPSC 7-27, 7-29f
  supply controller, universal 1-5, 7-1, 7-26ff, E-1
  supply status, UPSC 7-30
  up 1-6, 7-10
Power-up diagnostic 1-5, 7-26ff
Precision
  conversion instructions, fixed-point 2-2
  floating-point, double or single 3-1
Primary asynchronous line 7-1, 7-11
Priority mask 7-1, 7-5ff, E-1
Privileged
  faults 5-11, 8-4ff, 8-8
  instruction 7-24
  instruction violation 8-6
Privileges, access 8-1
Processor
  halts 8-5
  information 5-7
  initialization 1-6
  state 5-7, 8-5f
  status register 1-6, 2-8, 5-6f, 9-2, C-1f
Processor,
  arithmetic 1-3
  central 1-2ff, 7-1, 7-3ff
  instruction 1-2f
  stop 7-6
Program
  counter 1-3, 1-6, 5-1f, 5-7, 9-1f, 9-6, C-1f
  counter, floating-point 3-5, C-2
  flow 9-6
    instructions 5-9ff
    management 5-1ff
    management instructions, C/350 9-5t
  interrupt request 7-3
  interrupts 7-7, 7-12
Program, system control 1-5, 7-1, 7-13ff
Programmable interval timer 7-1, 7-7ff, E-1
Programming, C/350 9-1ff
Protection
  fault 1-3, 5-7, 8-3, F-1
    codes F-1t
    handler 8-8
  fault,
    address 8-4ff
    address translator F-1
  system hardware checks 8-3
  validation 8-3
  violation 5-11, 8-6, 9-6
Protection,
  I/O C-4
  logical address translation 1-6
PSR 2-8, 5-6f, 7-5ff, 9-2, C-1f, D-4f
PSR manipulation instructions 2-8

Q

Queue
  instructions 6-1t
  management 6-1

R

RAM modules 1-3
Random memory access 1-3
Range, C/350 addressing 9-2

Read
  access violation 8-6
  character buffer instruction 7-12
  count instruction 7-8f
  data from UPSC instruction 7-29
  switches instruction 7-4
READS instruction 7-4, A-2
Real-time clock 7-1, 7-9f
Reference violation, inward 8-6
Referenced bits 1-3, 8-3
Register
  fields, summary of C-1ff
  formats, BMC and DCH 7-21
  instruction, wide stack 4-1t
Register(s),
  16-bit initial count 7-7
  BMC 7-20ff
  BMC status 7-22, C-5
  C/350 9-1f
  DCH 7-20ff
  DCH status 7-22, C-5
  DCH/BMC status 7-21ff, C-5
  device map 7-20t, 7-21
  floating-point status 1-3, 1-6, 3-5t, 9-1f, C-1f
  I/O channel status 7-24
  IOC status 7-5ff, 7-23, C-5
  PIT initial count 7-9
  port
    definition 7-21f, C-5
    status 7-21, 7-23
  processor status 1-6, 2-8, 5-6f, 9-2, C-1f
  segment base 1-6, C-1, C-4
  specify initial map 7-22, C-5
  stack management 1-6, 9-2
  UPSC
    control 7-27
    diagnostic test 7-27f
    fault code 7-29ff
    power margining 7-27, 7-29f
  wide stack 4-1t
Reporting, CPU error 7-15
Request
  data from UPSC instruction 7-29
  flag, interrupt 7-8, 7-10
Request, interrupt 7-3, 7-6, 7-8f
Reserved
  memory 8-8f, 9-6
  memory locations 8-8, 9-2, D-1ff
  memory, hardware 8-8
Reset instruction 7-5ff, 7-10f, 7-14ff, 7-23, A-2, C-5
Reset,
  console 7-16f
  I/O 1-6
  system 1-6
Restartable instruction 5-5
Resumable instruction 5-6ff
Return
  block instructions, wide stack 4-2t
  SCP status instruction 7-18f
Revision, microcode 9-6ff, C-6
Ring
  crossing 5-2, 5-7, 5-10t, 9-6
  crossing validation 1-3
RND flag 3-5, C-2
RTC device 7-9f, E-1

S

SBR 1-6, C-1, C-4
SCP
  commands 7-15ff
  device 1-5, 7-13, E-1
  error code 7-15f
  instructions 7-14ff
  interrupts 7-16f
  power down mode, select 7-15ff
  status instruction, return 7-18f
Segment
  crossing 8-5
  crossing validation 8-3
  crossing violation 8-6
Segment(s) 1-3, 5-1f, 8-5, 8-8, C-1
  0 5-2, 8-5f, 8-8, D-1
  0 through 7, page zero locations for D-1
  base registers 1-6, C-1, C-4
  crossing (see ring crossing)
  transfer instructions 5-10t
  validity violation 8-6
  zero (see segment 0)
Segmentation 5-2
Select
  RTC frequency instruction 7-10f
  SCP power down mode 7-15ff
Sequence of subroutine instructions 5-10t
Sequence,
  diagnostic 7-18f
  enter diagnostic 7-15ff
  interrupt 5-2f
  page fault 8-6
Single precision floating-point 3-1
Size, physical memory 9-6ff
Skip
  instructions 5-9t
  instructions, fixed-point increment or decrement word and 2-6t
  on condition instructions,
    fixed-point 2-5t
    floating-point 3-3t
Skip, CPU 7-7, A-2
Slots, BMC and DCH map 7-19, 7-24
Sniffing, memory 1-4
Soft system console 1-5
Space, logical and physical address 5-2, 8-5
Specify initial
  count instruction 7-9
  map register instruction 7-22, C-5

SSPT instruction 7-19, 8-8, A-2
Stack
  access instructions, wide 4-2t, 4-4
  base, wide 1-3
  fault 5-7, 5-11
  fault codes F-2
  instructions, wide 4-1ff
  limit, wide 1-3
  management 4-1ff
    instructions, C/350 9-6
    registers 1-6, 9-2
  overflow 5-7, B-1
  pointer, wide 1-3, 8-5f
  register instructions, wide 4-1t
  return block instructions, wide 4-2t
  underflow B-1
Stack,
  narrow 5-11, 9-2, 9-6
  wide 1-3, 5-7, 8-5f
Standard I/O device codes E-1
State
  area 8-8
  pointer 8-8
  pointer instruction, store 7-19, 8-8
State,
  floating-point D-4f
  processor 5-7, 8-5f
Status
  field 8-5
  register,
    DCH/BMC 7-21ff, C-5
    floating-point 1-3, 1-6, 3-5t, 9-1f, C-1f
    I/O channel 7-24
    IOC 7-5ff, 7-23, C-5
    port 7-21, 7-23, C-5
    processor 1-6, 2-8, 5-6f, 9-2, C-1f
Status,
  extended error 7-15f
  return SCP 7-18f
  system power and 1-5, 7-26ff
  UPSC power supply 7-30
Stop
  bits 7-12
  processor 7-6
Storage location, reserved 8-8
Store state pointer instruction 7-19, 8-8
Store, load control G-1ff
Subprogram instructions, EDIT 2-10t
Subroutine
  instructions 5-10t, 5-10t
  instructions, sequence of 5-10t
Subtraction instructions,
  fixed-point 2-3t
  floating-point 3-2t, 9-4t
Summary of register fields C-1ff
Summary,
  instruction A-1f
  technical 1-1ff
Switches, console 7-4
Synchronous
  communications 1-5, 7-19
  controller, intelligent 1-5, E-1
  line controller 1-5
System
  control
    functions 1-5
    program 1-5, 7-1, 7-13ff
  hardware checks, protection 8-3
  management, memory and 8-1ff
  microcode 1-6
  overview 1-1ff
  power and status 1-5, 7-26ff
  reset 1-6
System,
  interrupt 7-5ff
  I/O 1-4
  memory 1-3
  multiprogram operating 7-7

T

Table
  entry format, page 8-2f
  entry, page 8-5
  validity violation, page 8-6
Table,
  BMC map 7-19
  one-level or two-level page 8-6, B-1
Tables, page 8-1, B-1
TE flag 3-5, C-2
Technical summary 1-1ff
Terminal, operator's 1-5
Test register, UPSC diagnostic 7-27f
Test, diagnostic 7-16f
Time
  base 7-7, 7-9
  calculations 7-9
Timeout, UPSC 7-27
Timer, programmable interval 7-1, 7-7ff, E-1
Times, instruction execution B-1ff
Time-slice 7-7
Transfer
  flag, BMC 7-22, C-5
  instructions, segment 5-10t
  rates 1-4
Transfer,
  I/O 1-4
  mapped 7-19
Translation, address 7-5ff, 7-19, 8-1, C-4
Translator protection fault, address F-1
Translator, address 1-3, 1-6, 5-2, 5-11, 7-19, 8-1ff,
    B-1, D-1
TTI
  and TTO buffer 7-12
  device 7-11, E-1
  device flags 7-12
  instructions 7-12f

TTO
  device 7-11, 7-13, E-1
  device flags 7-12
Two-level page table 8-6, B-1
Two's complement 2-1, 7-8
Type 1 or 2 instruction 5-2

U

Underflow, stack B-1
UNF flag 3-5, C-2
Unit,
  central processing 7-1
  floating-point 9-6ff, C-6f
Universal power supply controller 1-5, 7-1, 7-26ff, E-1
UPSC
  battery backup 7-27, 7-29f
  control register 7-27
  device 1-5, 7-26ff, E-1
  device flags 7-26ff
  diagnostic test register 7-27f
  fault code register 7-29ff
  instructions 7-26ff
  LEDs 7-30ff
  power fail interrupts 7-27, 7-29
  power margining register 7-27, 7-29f
  power supply status 7-30
  timeout 7-27
UPSC, verify the data path between MV/4000 and 7-27

V

Validation,
  access 1-3, 8-3
  page 1-3, 8-3
  protection 8-3
  ring crossing 1-3
  segment crossing 8-3
Validity 8-5
  bit, BMC 7-21
  error flag, BMC and DCH 7-22, C-5
Verify the data path between MV/4000 and UPSC 7-27
Violation fault, protection 5-11, 8-6
Violation,
  execute access 8-6
  indirect 8-6
  inward reference 8-6
  I/O instruction 8-6
  page table validity 8-6
  privileged instruction 8-6
  protection 9-6
  read access 8-6
  segment
    crossing 8-6
    validity 8-6
  write access 8-6

W

WDPOP 8-6
Wide
  frame pointer 1-3, 8-5f
  load map 7-24f
  stack 1-3, 5-7, 8-5f
    access instructions 4-2t
    base 1-3
    instructions 4-1ff
    limit 1-3
    pointer 1-3, 8-5f
    return block instructions 4-2t
WLMP 7-23ff
WLMP instruction A-2
Word address 5-2
Word, double 1-3
Wraparound, address 5-1, 9-6
Write
  access violation 8-6
  data to UPSC instructions 7-26ff

X

XOP instruction 9-2
XOP0 instruction 9-2
XOP1 instruction 9-2

Z

Z flag 3-5, C-2
Zero address, page 7-16f
Zero,
  page 5-2, 7-16f, 8-5, 8-8, D-1
  segment (see segment 0)

Data General

014-000736-00

Data General Corporation, Westboro, MA 01580