

# **ECLIPSE MV/9500<sup>™</sup> System**

## **Principles of Operation**



**ECLIPSE MV/9500<sup>™</sup> System  
Principles of Operation  
Supplement**





# **ECLIPSE MV/9500™ System**

## **Principles of Operation**

### **Supplement**

014-001855-00

Ordering No. 014-001855  
Copyright © Data General Corporation, 1990  
All Rights Reserved  
Printed in the United States of America  
Rev. 00, March, 1990

# Notice

DATA GENERAL CORPORATION (DGC) HAS PREPARED THIS DOCUMENT FOR USE BY DGC PERSONNEL, CUSTOMERS, AND PROSPECTIVE CUSTOMERS. THE INFORMATION CONTAINED HEREIN SHALL NOT BE REPRODUCED IN WHOLE OR IN PART WITHOUT DGC'S PRIOR WRITTEN APPROVAL.

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.

IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS DOCUMENT OR THE INFORMATION CONTAINED IN IT, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY OF SUCH DAMAGES.

CEO, DASHER, DATAPREP, DESKTOP GENERATION, ECLIPSE, ECLIPSE MV/4000, ECLIPSE MV/6000, ECLIPSE MV/8000, GENAP, INFOS, microNOVA, NOVA, PRESENT, PROXI, SWAT, and TRENDVIEW are U.S. registered trademarks of Data General Corporation; and AOSMAGIC, AOS/VSMAGIC, AROSE/PC, ArrayPlus, AViiON, BaseLink, BusiGEN, BusiPEN, BusiTEXT, CEO Connection, CEO Connection/LAN, CEO Drawing Board, CEO DXA, CEO Light, CEO MAILI, CEO Object Office, CEO PXA, CEO Wordview, CEOwrite, COBOL/SMART, COMPUCALC, CSMAGIC, DASHER/One, DASHER/286, DASHER/386, DASHER/LN, DATA GENERAL/One, DESKTOP/UX, DG/500, DG/AROSE, DGConnect, DG/DBUS, DG/Fontstyles, DG/GATE, DG/GEO, DG/HEO, DG/L, DG/LIBRARY, DG/UX, DG/XAP, ECLIPSE MV/1000, ECLIPSE MV/1400, ECLIPSE MV/2000, ECLIPSE MV/2500, ECLIPSE MV/7800, ECLIPSE MV/9500, ECLIPSE MV/10000, ECLIPSE MV/15000, ECLIPSE MV/18000, ECLIPSE MV/20000, ECLIPSE MV/40000, FORMA-TEXT, GATEKEEPER, GDC/1000, GDC/2400, microECLIPSE, microMV, MV/UX, PC Liaison, RASS, REV-UP, SLATE, SPARE MAIL, SUPPORT MANAGER, TEO, TEO/3D, TEO/Electronics, TURBO/4, UNITE, WALKABOUT, WALKABOUT/SX, and XODIAC are trademarks of Data General Corporation.

ECLIPSE MV/9500™ System Principles of Operation Supplement  
014-001855-00

Revision History:

Original Release - March 1990

# Updating Instructions

The complete documentation for ECLIPSE MV/9500™ Systems Principles of Operation consists of the two-volume set, *ECLIPSE® MV/Family (32-Bit) Systems Principles of Operation* (014-001371) and *ECLIPSE® MV/Family (32-Bit) Systems Instruction Dictionary* (014-001372), and this supplement (014-001855).

To create your copy of the ECLIPSE MV/9500 specific manual, please remove the following pages from the *ECLIPSE® MV/Family (32-Bit) Systems Principles of Operation* and *ECLIPSE® MV/Family (32-Bit) Systems Instruction Dictionary* and insert the supplement pages as follows:

Remove (from 014-001371)	Insert (from 014-001855)
—	Title/Notice (before Title/Notice of 014-001371)
Contents	Contents
Figures	Figures
Tables	Tables
1-17/1-18	1-17 through 1-30
8-1/8-2	8-1/8-2
8-9/8-10	8-9/8-10
8-21/8-22	8-21/8-22
8-25 through 8-28	8-25 through 8-28
8-35/8-36	8-35/8-36
8-59/8-60	8-59/8-60
8-71 through 8-80	8-71 through 8-80
8-93	8-93
A-7/A-8	A-7 through A-9
—	Appendix E
—	Appendix F
—	Appendix G
Index	Index
Front Cover	New Front Cover for 014-001371 (at end of supplement)
Back Cover	New Back Cover for 014-001371 (at end of supplement)
Remove (from 014-001372)	Insert (from 014-001855)
None	None



---

# Contents

## 1 System Overview

Functional Capabilities .....	1-2
Registers .....	1-2
Fixed-Point Computation .....	1-2
Floating-Point Computation .....	1-3
Stack Management .....	1-5
Program Flow Management .....	1-5
Queue Management .....	1-6
Graphics Management .....	1-6
Device Management .....	1-6
System Management .....	1-7
Memory Management .....	1-7
ECLIPSE 16-Bit Compatible Instructions .....	1-8
Accessing Memory .....	1-8
Current Segment .....	1-9
Other Segments .....	1-9
Memory Reference Instructions .....	1-10
Address Modes .....	1-11
Operand Access .....	1-13
Protection Capabilities .....	1-18
ECLIPSE MV/9500™ Hardware Summary .....	1-19
Main Systems .....	1-19
Central Processing Unit .....	1-20
Memory System .....	1-25
Input/Output System .....	1-25
Power System .....	1-28
Diagnostic Remote Processor .....	1-28
Initialization .....	1-31

## 2 Fixed-Point Computing

Binary Operations .....	2-2
Data Formats .....	2-2
Move Instructions .....	2-3
Arithmetic Instructions .....	2-4
Carry Operations .....	2-7
Shift Instructions .....	2-7
Skip Instructions .....	2-9
Overflow Fault .....	2-10
Processor Status Register .....	2-10
Logical Operations .....	2-12
Data Formats .....	2-12
Logical Instructions .....	2-13
Bit Manipulation .....	2-13
Shift Instructions .....	2-14
Skip Instructions .....	2-14

Decimal and Byte Operations .....	2-15
Data Formats .....	2-15
Move Instructions .....	2-19
Arithmetic Instructions .....	2-20
Shift Instructions .....	2-21
Effective Address Instructions .....	2-21
Skip Instructions .....	2-21
Data Type Faults .....	2-22
Decimal Arithmetic Example .....	2-23
<b>3 Floating-Point Computing</b>	
Data Formats .....	3-2
Conversion Instructions .....	3-3
Move Instructions .....	3-4
Floating-Point Arithmetic Operations .....	3-5
Appending Guard Digits .....	3-5
Aligning the Mantissas .....	3-5
Calculating and Normalizing the Result .....	3-6
Truncating or Rounding the Result .....	3-6
Storing the Result .....	3-6
Arithmetic Instructions .....	3-7
Addition .....	3-7
Subtraction .....	3-7
Multiplication .....	3-8
Division .....	3-8
Skip Instructions .....	3-9
Intrinsic Instruction Set .....	3-10
Faults and Status .....	3-12
<b>4 Stack Management</b>	
Wide Stack Operations .....	4-2
Wide Stack Registers .....	4-3
Wide Stack Base .....	4-3
Wide Stack Limit .....	4-3
Wide Stack Pointer .....	4-4
Wide Frame Pointer .....	4-4
Wide Stack Register Instructions .....	4-4
Wide Stack Data Instructions .....	4-5
Initializing a Wide Stack .....	4-7
Wide Stack Faults .....	4-8
<b>5 Program Flow Management</b>	
Related Instruction Groups .....	5-2
Execute Accumulator .....	5-2
Jump .....	5-2
Skip .....	5-2
Subroutine .....	5-4
Transferring Program Control to Another Segment .....	5-9
Subroutine Call .....	5-9
Subroutine Return .....	5-14
Fault Handling .....	5-16
Fixed-Point Overflow Fault .....	5-17
Floating-Point Faults .....	5-18
Decimal and ASCII Data Faults .....	5-19
Stack Faults .....	5-23

## 6 Queue Management

Building a Queue .....	6-2
Queue Descriptor .....	6-3
Setting Up and Modifying a Queue .....	6-3
Queue Examples .....	6-3
Queue Descriptor of an Empty Queue .....	6-3
Adding a Data Element into an Empty Queue .....	6-4
Adding a Data Element at the Head of a Queue .....	6-4
Adding a Data Element at the Tail of a Queue .....	6-4
Removing a Data Element .....	6-6
Queue Instructions .....	6-7

## 7 Graphics Management

Graphics Instruction Set .....	7-2
Forms .....	7-4
Forms and Bitmaps .....	7-5
Local Origin .....	7-6
Bounding Rectangle .....	7-6
Coordinate System .....	7-7
GIS Data Structures .....	7-9
Form Descriptor .....	7-9
Form Attributes .....	7-13
Operation Mask and Combination Rule .....	7-13
Line Drawing Attributes .....	7-16
Character Drawing Attributes .....	7-17
Character Fonts .....	7-17
Cursor Descriptor .....	7-17
Color Descriptors .....	7-20
Form Cache .....	7-21
Interrupts .....	7-21
Fault Handling .....	7-22
Fixed-Point Overflow .....	7-25

## 8 Device Management

I/O Communication .....	8-2
I/O Access .....	8-3
I/O Registers .....	8-3
Types of Information Transfers .....	8-4
General I/O Instructions .....	8-6
Device Flags .....	8-7
Interrupts .....	8-8
Interrupt Flags .....	8-8
Instruction Interruption .....	8-9
Processor Interrupt Servicing .....	8-10
Vectored Interrupt Processing .....	8-12
Interrupt Service Routines .....	8-17
Data Channel/Burst Multiplexor Channel .....	8-20
Transfer Sequence .....	8-21
Device Maps and Data Transfers .....	8-22
DCH/BMC Registers .....	8-25
Device Controllers .....	8-29
Device Controller Registers .....	8-29
Device Controller Programming .....	8-32
Data Transfer Latency .....	8-33

Integral Devices .....	8-36
Central Processor .....	8-36
Timing Mechanisms .....	8-49
Architectural Clocks .....	8-49
Programmable Interval Timer .....	8-60
Real-Time Clock .....	8-63
Primary Asynchronous Line Input/Output .....	8-65
System Control Processor/Program .....	8-68
Power Supply Controllers .....	8-80
Multiple Central Processing Units .....	8-88
Initialization .....	8-88
Processor State Block .....	8-89
Memory Views .....	8-90
I/O Communication .....	8-91
Multiple I/O Channels .....	8-91
I/O Interrupt Handling .....	8-91
Intra-Processor Communication .....	8-92
Error Codes .....	8-93

## 9 Memory and System Management

Page Access .....	9-2
Segment Access and Address Translation .....	9-2
Segment Base Registers .....	9-2
Page Frames .....	9-4
Pagatables .....	9-4
Address Translation .....	9-6
Page Access .....	9-9
Central Processor Identification .....	9-10
Privileged Faults .....	9-11
Page Faults .....	9-11
Protection Violations .....	9-13
Reserved Memory .....	9-18
Page Zero .....	9-18
State Area .....	9-21

## 10 ECLIPSE 16-Bit Programming

ECLIPSE Registers .....	10-2
ECLIPSE Stack .....	10-5
ECLIPSE Faults and Interrupts .....	10-6
Expanding an ECLIPSE Program .....	10-6
Expanding an ECLIPSE Subroutine .....	10-7
ECLIPSE Instructions .....	10-7
ECLIPSE MV/Family Instruction Compatibility .....	10-8
ECLIPSE Memory Reference Instructions .....	10-8
ECLIPSE Fixed-Point Instructions .....	10-12
ECLIPSE Floating-Point Instructions .....	10-13
ECLIPSE Program Flow Instructions .....	10-15
ECLIPSE Stack Instructions .....	10-16
Program Flow .....	10-17
Fault Handling .....	10-17
Reserved Memory .....	10-18
CPU Identification .....	10-18



## **A Register Fields**

Segment Base Registers .....	A-2
Program Counter .....	A-3
Processor Status Register .....	A-4
Floating-Point Status Register .....	A-5
DCH/BMC Status Registers .....	A-6
CPU Identification .....	A-7
ECLIPSE MV/9500 Supplement .....	A-9

## **B Fault and Status Codes**

Protection Faults .....	B-1
Page Faults .....	B-2
Stack Faults .....	B-2
UPSC Faults .....	B-2
PSC Status and Faults .....	B-4
Decimal/ASCII Faults .....	B-6

## **C Reserved Memory Locations**

## **D Load Control Store Instruction**

Microcode File and Block Format .....	D-3
LCS Implementation .....	D-4
Microcode Blocks .....	D-5
Error Return .....	D-7
Kernel Functions .....	D-8

## **E Standard I/O Device Codes**

## **F Context Block Formats**

## **G Instruction Execution Times**

## **Glossary**

# Figures

## Figure

1-1	ECLIPSE MV/Family functional components .....	1-1
1-2	Fixed-point accumulator .....	1-3
1-3	Floating-point accumulator .....	1-4
1-4	Program counter format .....	1-5
1-5	Logical address space .....	1-7
1-6	Memory reference instruction word addressing formats .....	1-10
1-7	Memory reference instruction byte addressing formats .....	1-11
1-8	Byte pointer format .....	1-15
1-9	Byte addressing .....	1-16
1-10	Bit pointer format .....	1-17
1-11	Bit addressing .....	1-17
1-12	Major elements of the ECLIPSE MV/9500 system .....	1-20
1-13	Four-stage instruction pipeline .....	1-21
1-14	I/O system .....	1-26
1-15	Diagnostic remote processor .....	1-29
2-1	Fixed-point two's-complement data formats .....	2-2
2-2	ECLIPSE compatible shift operations .....	2-8
2-3	Processor status register format .....	2-10
2-4	Fixed-point logical data formats .....	2-12
2-5	Explicit data type indicator .....	2-15
2-6	Packed and unpacked decimal data .....	2-18
2-7	Decimal arithmetic example .....	2-23
3-1	Floating-point data formats .....	3-2
3-2	Intrinsic instruction set format .....	3-10
3-3	Floating-point status register format .....	3-13
4-1	Typical wide stack .....	4-2
4-2	Wide stack management register format .....	4-3
4-3	Sample code for initializing a wide stack .....	4-7
4-4	Example of wide stack operations .....	4-7
5-1	Illegal and legal skip instruction sequences .....	5-3
5-2	DO-loop instruction sequence .....	5-3
5-3	Example of subroutine code for <b>XJSR</b> .....	5-7
5-4	Wide stack operations from <b>XJSR</b> , <b>WSSVS</b> , and <b>XPEF</b> instructions .....	5-8
5-5	Wide stack operations from <b>WRTN</b> instruction .....	5-8
5-6	Gate array format .....	5-10
5-7	<b>XCALL</b> or <b>LCALL</b> effective address .....	5-11
5-8	Validating inward segment crossing sequence .....	5-13
5-9	Wide Return instruction sequence .....	5-15

## Figure

6-1	Data elements with user data .....	6-2
6-2	Format of queue descriptor .....	6-3
6-3	Queue descriptor for an empty queue .....	6-3
6-4	Data element added to an empty queue .....	6-4
6-5	Data element added at head of queue .....	6-4
6-6	Data element added at tail of queue .....	6-5
6-7	Data element removed .....	6-6
7-1	Form data structures .....	7-4
7-2	Windowing with virtual bitmaps .....	7-5
7-3	Use of rectangle list .....	7-6
7-4	Coordinate conversions .....	7-8
7-5	Vertex of contiguous line segments .....	7-16
7-6	Effect of line style .....	7-17
7-7	Types of cursors .....	7-18
7-8	GIS fault sequence .....	7-23
7-9	Overdraw condition parameters .....	7-26
7-10	Overdraw condition parameters for endpoints .....	7-27
8-1	An ECLIPSE MV/Family system with dual IOCs .....	8-2
8-2	General I/O instruction format .....	8-6
8-3	Interrupt sequence .....	8-11
8-4	Vectored interrupt processing sequence .....	8-13
8-5	Sequence of actions to conclude interrupt service .....	8-14
8-6	Vector table .....	8-14
8-7	Device control table (DCT) .....	8-16
8-8	DCH/BMC registers .....	8-25
8-9	CPU/SCP communications sequence .....	8-69
9-1	Segment base register format .....	9-3
9-2	Pagetable entry format .....	9-4
9-3	Indirect and effective logical address formats .....	9-6
9-4	One-level pagetable translation .....	9-7
9-5	Two-level pagetable translation .....	9-8
9-6	Page fault sequence .....	9-12
9-7	Protection violation sequence .....	9-16
10-1	ECLIPSE MV/Family registers with applicable ECLIPSE bits .....	10-4
10-2	ECLIPSE word addressing format .....	10-8
10-3	ECLIPSE effective addressing .....	10-9
10-4	ECLIPSE byte addressing format .....	10-9
10-5	ECLIPSE byte addressing .....	10-10
10-6	ECLIPSE bit addressing format .....	10-10
10-7	<b>BTO, BTZ, SNB, SZB, and SZBO</b> bit addressing .....	10-11
D-1	Microcode file format .....	D-4
D-2	Microcode block format .....	D-5

# Tables

## Table

1-1	Program counter format .....	1-6
1-2	Effective addressing .....	1-13
1-3	Word-oriented data .....	1-14
1-4	Byte data .....	1-14
1-5	Byte pointer contents .....	1-16
1-6	Bit pointer contents .....	1-16
1-7	Faults .....	1-18
2-1	Fixed-point two's-complement formats .....	2-2
2-2	Range of 16- and 32-bit fixed-point numbers (in octal) .....	2-2
2-3	Fixed-point precision conversion .....	2-3
2-4	Fixed-point data movement instructions .....	2-3
2-5	Fixed-point addition instructions .....	2-4
2-6	Fixed-point subtraction instructions .....	2-5
2-7	Fixed-point multiplication instructions .....	2-5
2-8	Fixed-point division instructions .....	2-6
2-9	Fixed-point increment or decrement value and skip instructions .....	2-6
2-10	Carry initializing instructions .....	2-7
2-11	Fixed-point skip on condition instructions .....	2-9
2-12	PSR manipulation instructions .....	2-10
2-13	Processor status register contents .....	2-11
2-14	Logical Instructions .....	2-13
2-15	Bit Instructions .....	2-13
2-16	Logical shift instructions .....	2-14
2-17	Fixed-point logical skip instructions .....	2-14
2-18	Data type indicator description .....	2-16
2-19	Explicit data types .....	2-17
2-20	Sign and number combination for unpacked decimal .....	2-19
2-21	Nonsign-positioned numbers for unpacked decimal .....	2-19
2-22	Fixed-point byte movement instructions .....	2-19
2-23	Fixed-point to floating-point conversion and store instructions .....	2-20
2-24	Edit subprogram instructions .....	2-20
2-25	Arithmetic instructions .....	2-20
2-26	Hex shift instructions .....	2-21
2-27	Load effective address instructions .....	2-21
2-28	Decimal and ASCII fault codes .....	2-22
3-1	Floating-point data formats description .....	3-2
3-2	Floating-point binary conversion instructions .....	3-3
3-3	Floating-point decimal conversion instructions .....	3-3
3-4	Floating-point data movement instructions .....	3-4
3-5	Floating-point addition instructions .....	3-7
3-6	Floating-point subtraction instructions .....	3-7
3-7	Floating-point multiplication instructions .....	3-8
3-8	Floating-point division instructions .....	3-9

# Table

3-9	Floating-point skip on condition instructions .....	3-9
3-10	Intrinsic instruction set format description .....	3-11
3-11	Floating-point intrinsic instructions .....	3-11
3-12	Floating-point status register instructions .....	3-12
3-13	Floating-point status register format description .....	3-13
4-1	Wide stack management register format description .....	4-3
4-2	Wide stack register instructions .....	4-5
4-3	Wide stack doubleword access instructions .....	4-5
4-4	Wide stack return block instructions .....	4-6
4-5	Standard wide return block .....	4-6
4-6	Instructions affecting the wide stack .....	4-8
5-1	Jump instructions .....	5-2
5-2	Skip instructions .....	5-3
5-3	Standard wide return block .....	5-5
5-4	Subroutine instructions .....	5-5
5-5	Sequence of subroutine instructions .....	5-5
5-6	Segment program control transfer instructions .....	5-9
5-7	Gate array format description .....	5-10
5-8	<b>XCALL</b> or <b>LCALL</b> effective address format description .....	5-11
5-9	Faults .....	5-16
5-10	Fixed-point fault return block .....	5-17
5-11	Wide floating-point fault return block .....	5-18
5-12	Narrow floating-point fault return block .....	5-19
5-13	Decimal and ASCII fault codes .....	5-20
5-14	Wide return block for decimal data fault (type 1) .....	5-21
5-15	Wide return block for ASCII data fault (type 2) .....	5-21
5-16	Wide return block for ASCII data fault (type 3) .....	5-21
5-17	Narrow return block for decimal data fault (type 1) .....	5-22
5-18	Narrow return block for ASCII data fault (type 2) .....	5-22
5-19	Narrow return block for ASCII data fault (type 3) .....	5-22
5-20	Wide stack fault return block .....	5-23
5-21	Wide stack fault codes .....	5-24
5-22	Narrow stack fault return block .....	5-25
6-1	Queue instructions .....	6-7
7-1	GIS instructions .....	7-3
7-2	Form descriptor contents .....	7-10
7-3	Rectangle descriptor contents .....	7-13
7-4	Form attributes .....	7-14
7-5	Combination rules .....	7-15
7-6	Cross-hair cursor descriptor .....	7-19
7-7	Image cursor descriptor .....	7-19
8-1	General I/O instructions .....	8-6
8-2	General I/O instruction format description .....	8-6
8-3	Device flag controls for general devices .....	8-7
8-4	Device flag tests for skip instruction .....	8-7
8-5	Vector table contents .....	8-15
8-6	Device control table contents .....	8-16
8-7	I/O instructions for DCH/BMC maps .....	8-22
8-8	I/O registers .....	8-25

# Table

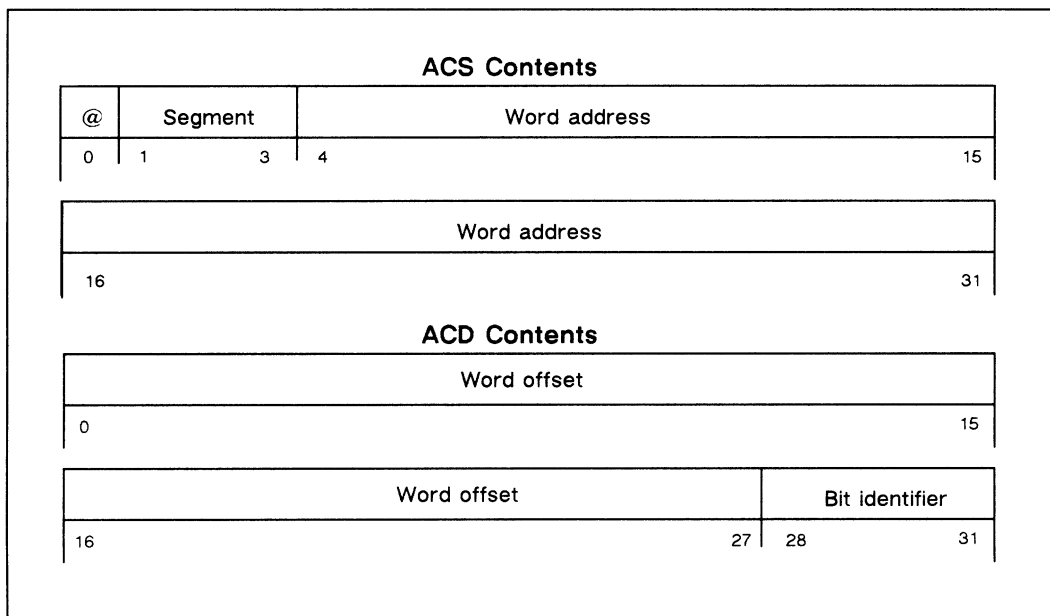
8-9	Word/block counter values .....	8-31
8-10	I/O instructions for the CPU .....	8-37
8-11	CPU device instructions with I/O channels .....	8-40
8-12	Instructions affecting the alarm clock .....	8-50
8-13	Instructions affecting the time-slice timer .....	8-54
8-14	Instructions affecting the PIT .....	8-60
8-15	Instructions affecting the RTC .....	8-63
8-16	I/O instructions for TTI and TTO .....	8-66
8-17	SCP instructions .....	8-68
8-18	I/O instructions for the power supply controllers .....	8-81
8-19	Multiple-processor instructions .....	8-92
8-20	ECLIPSE MV/Family instructions with multiple-processor functions .....	8-93
8-21	Error values returned to AC1 .....	8-93
9-1	Segment base register format description .....	9-3
9-2	Pagetable entry format description .....	9-5
9-3	Logical address format description .....	9-6
9-4	Instructions that manipulate referenced and modified bits .....	9-10
9-5	System identification instructions .....	9-10
9-6	Priority of protection violation faults .....	9-13
9-7	Protection fault return block .....	9-15
9-8	Protection fault codes .....	9-15
9-9	Page zero locations for segment 0 .....	9-19
9-10	Page zero locations for segments 1 through 7 .....	9-20
10-1	Comparison of ECLIPSE 16-bit and ECLIPSE MV/Family registers .....	10-3
10-2	Standard ECLIPSE (narrow) return block .....	10-5
10-3	Alterations to ECLIPSE subroutines .....	10-7
10-4	ECLIPSE word addressing format description .....	10-9
10-5	ECLIPSE byte addressing format description .....	10-9
10-6	ECLIPSE bit addressing format description .....	10-11
10-7	ECLIPSE fixed-point computing instructions .....	10-12
10-8	ECLIPSE floating-point computing instructions .....	10-13
10-9	ECLIPSE program flow management instructions .....	10-15
10-10	ECLIPSE stack management instructions .....	10-16
A-1	Registers and contents .....	A-1
A-2	Segment base register contents .....	A-2
A-3	Program counter format for ECLIPSE MV/Family programs .....	A-3
A-4	Program counter modified by ECLIPSE 16-bit instructions .....	A-3
A-5	Processor status register contents .....	A-4
A-6	Floating-point status register contents .....	A-5
A-7	I/O channel status register contents .....	A-6
A-8	I/O channel mask register contents .....	A-6
A-9	I/O channel definition register contents .....	A-7
B-1	Protection fault codes .....	B-1
B-2	Page fault codes .....	B-2
B-3	Stack fault codes .....	B-2
B-4	UPSC fault codes .....	B-2
B-5	PSC status and fault codes .....	B-4
B-6	Decimal/ASCII fault codes .....	B-6

**Table**

C-1	Page zero location for segment 0 .....	C-2
C-2	Page zero locations for segments 1 through 7 .....	C-3
D-1	Microcode file format blocks .....	D-3
D-2	Words used in the microcode block format .....	D-5
D-3	Title block format .....	D-5
D-4	End block format .....	D-6
D-5	Combined action of End block data words 1 and 2 .....	D-6
D-6	Code block format .....	D-6
D-7	Fill block format .....	D-7
D-8	Comment block format .....	D-7
D-9	Revision block format .....	D-7
D-10	Error codes returned to AC0 .....	D-8
E-1	Standard I/O device codes .....	E-1
F-1	Context block format .....	F-2
G-1	Adjusting memory references .....	G-1
G-2	Adjusting any instruction .....	G-1
G-3	ECLIPSE MV/9500 instruction execution times .....	G-4

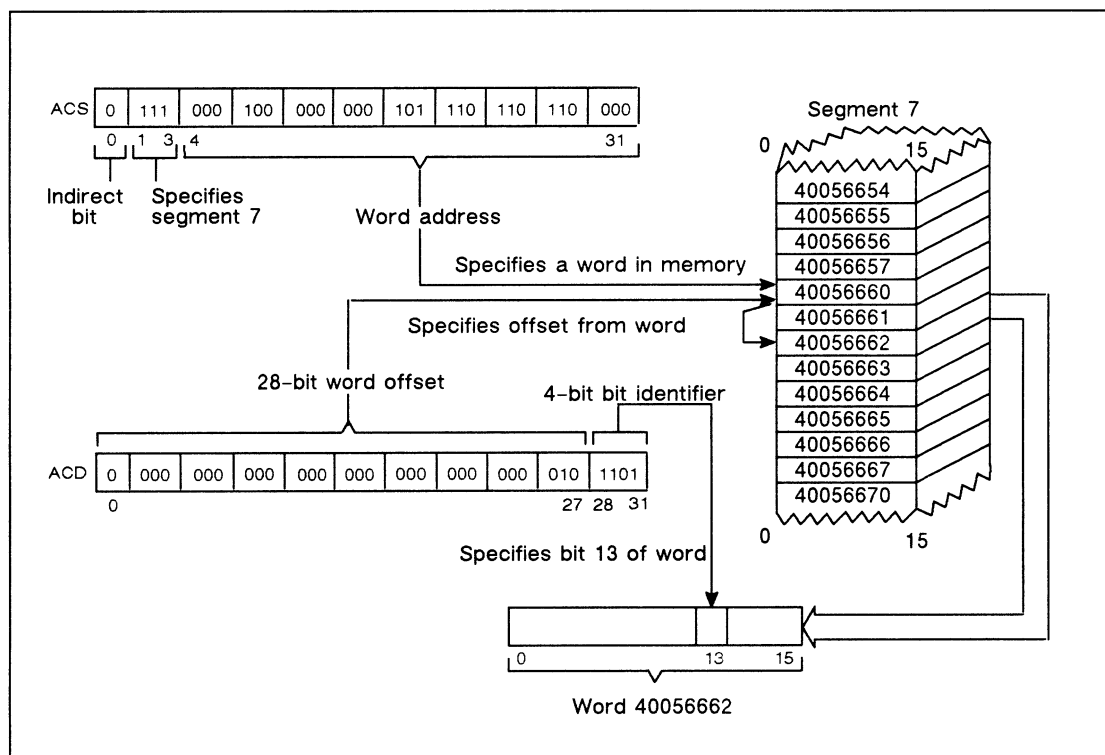






**Figure 1-10** *Bit pointer format*

The processor uses the *acs* accumulator contents to calculate an effective address. If a bit instruction specifies the two accumulators as the same accumulator, then the base word address is zero in the current segment.



**Figure 1-11** *Bit addressing*

## Protection Capabilities

While executing an instruction, the processor checks the validity of a memory reference or an I/O operation (protection violation), a page reference (nonresident page), a stack operation, a computation, and a data format. Table 1–7 lists the validity checks (or faults).

**Table 1–7 Faults**

Fault	Type
Nonresident page	Privileged
Protection violation	Privileged
Stack operation	Nonprivileged
Fixed-point computation	Nonprivileged
Floating-point computation	Nonprivileged
Invalid decimal or ASCII data format	Nonprivileged

If the processor detects an error, a nonprivileged or privileged fault occurs before the next instruction is executed. A nonprivileged fault occurs when the processor detects a computation error. The processor limits I/O access on a per ring basis, and limits memory access using a hierarchical protection mechanism. For instance,

- Before executing an I/O instruction, the processor checks the I/O validity flag in the current segment.
- Before executing a memory reference instruction, the processor checks the validity of the reference.

The processor executes an I/O or memory reference instruction when validity checks permit the access. Otherwise, the processor initiates a protection violation. Thus, an operating system can restrict access to the devices to specific segment(s).

Accessing and changing a protection mechanism requires a privileged instruction (executable only in segment 0) or data access, typically controlled by the operating system. Refer to the chapter, “Program Flow Management,” for further details on servicing a nonprivileged fault.

A privileged fault occurs when an operation is not permitted by the address translation mechanism (page not resident, I/O protection) or by the ring structure (privileged instruction, outward call). Refer to the chapters, “Memory and System Management” and “Program Flow Management,” for further details on servicing a privileged fault.

## ECLIPSE MV/9500™ Hardware Summary

This summary describes the ECLIPSE MV/9500™ computer system and its initial processor conditions

The ECLIPSE MV/9500 computer implements the 32-bit processing power of the ECLIPSE MV/Family architecture and is organized as a single-board assembly as shown in Figure 1-12. This board, referred to as the system board, consists of a microMV processor, memory control unit (MCU), system cache, system clocks, input/output controller, and diagnostic remote processor (DRP). These subsystems are interconnected by the system bus.

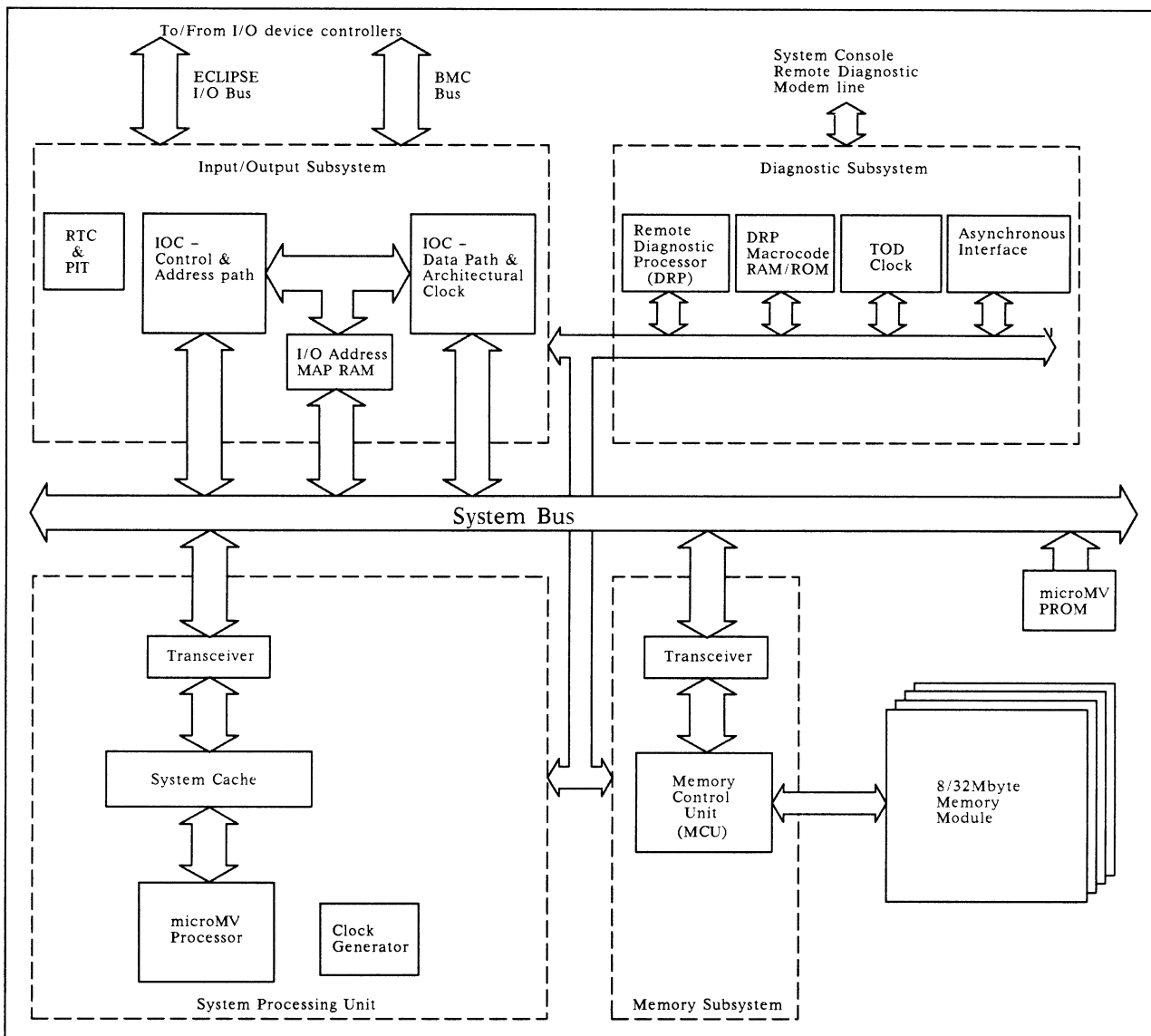


Figure 1-12 System Board Organization

The system processing unit consists of a microMV processor, a system clock generator, a system cache, and a transceiver interface to the system bus.

The input/output subsystem is based on an input/output controller (IOC), an I/O address MAP RAM, a real-time clock (RTC), a programmable interval timer (PIT), and architectural clock circuits.

The diagnostic subsystem consists of a diagnostic remote processor (DRP), DRP macrocode RAM/ROM, a time-of-day (TOD) clock, and an asynchronous interface to a remote diagnostic modem line and system console.

### microMV Processor

The microMV processor is a custom-design, high-density, CMOS VLSI microprocessor chip that provides the processing power for the ECLIPSE MV/9500 computer system. Essentially, the microMV is a microcode-controlled, 32-bit central processing unit (CPU) that processes instructions, generates logical addresses, translates logical addresses to physical addresses, and performs arithmetic and logical data manipulation.

The microMV processor runs at a 15-megahertz rate, giving a single-cycle instruction time of 66.67 nanoseconds. The microMV executes the standard ECLIPSE MV/Family 32-bit instruction set. An internal 64-bit wide data bus provides the data link between microMV elements. A 32-bit wide address bus provides physical memory addressing up to 4.3 gigabytes. An internal 2-kilobyte cache provides fast-access interim storage, direct mapped, and with automatic write-through to system memory. The microMV has four inputs for receiving interrupts. Three of these inputs are maskable; the other is nonmaskable. The microMV also incorporates the time-slice timer portion of the system's architectural clocks. The microMV's instruction set supports system configurations with multiple microMVs operating in parallel.

### Major Processor Elements

As shown in Figure 1-13, the microMV consists of the following:

- A four-stage instruction pipe (IP) for fetching and decoding instructions.
- A microsequencer for controlling execution of microinstructions read from a ROM-based control store. A patch control store RAM provides supplemental storage for updating the control store code.
- An instruction/data cache (IDC) for high-speed, temporary storage of instructions and data.
- An address generation unit (AGU) for logical address generation.
- An address translation unit (ATU) for logical-to-physical address translation.
- An arithmetic logic unit (ALU) for data manipulation.
- Floating-point assist hardware for executing floating-point instructions.
- An address/data interface that provides interface with external buses.

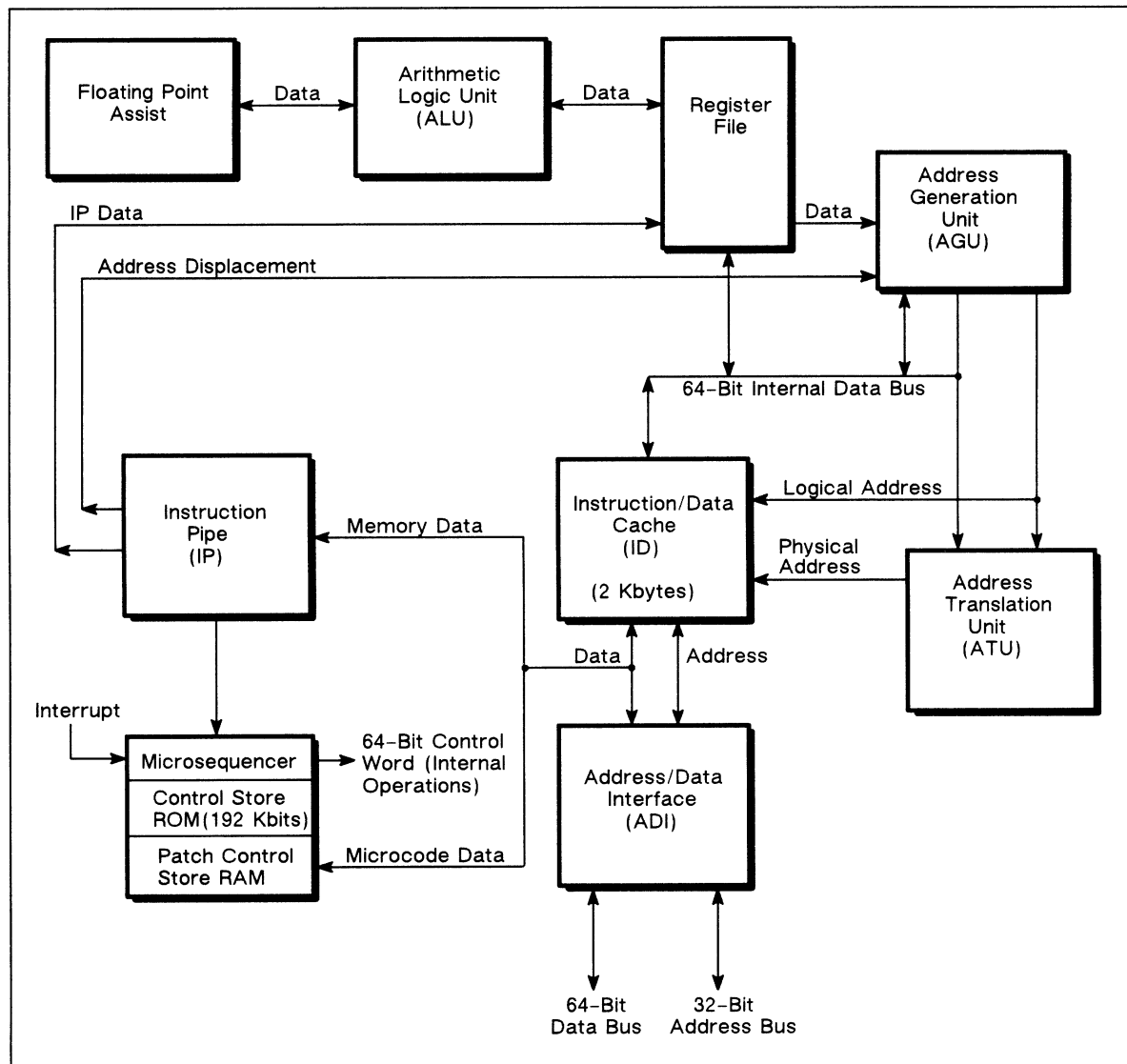
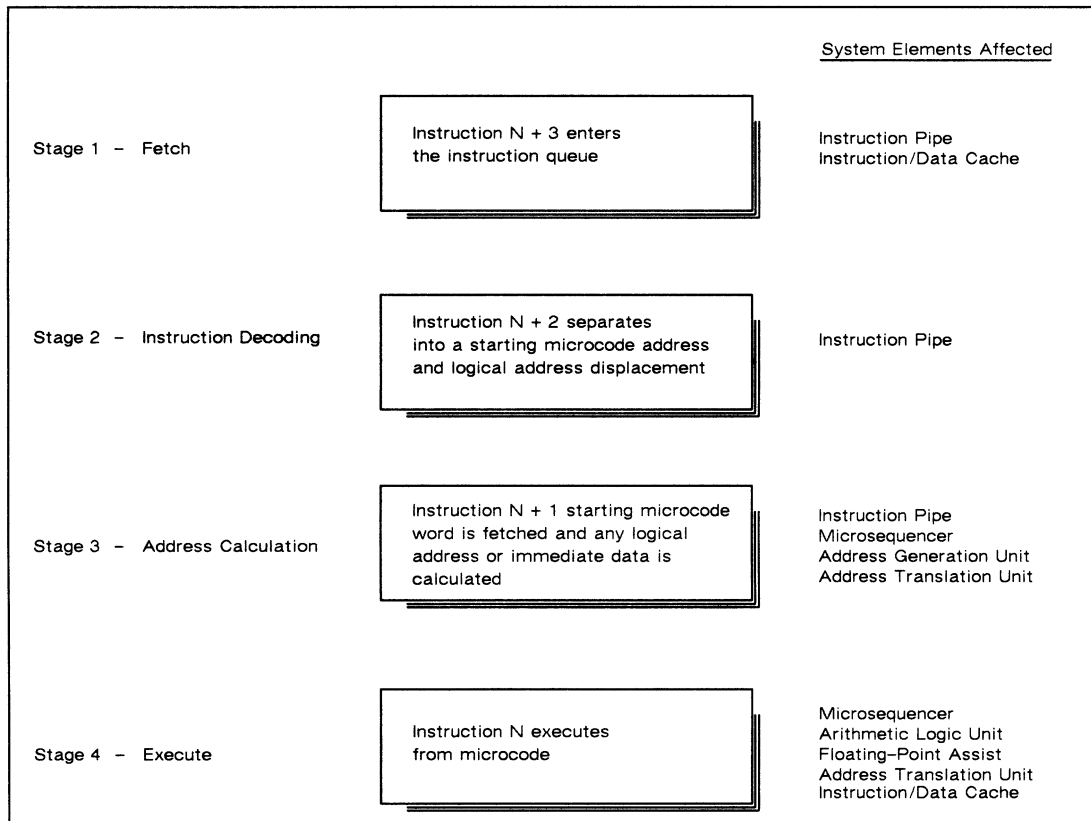


Figure 1-13 Major Elements of the microMV Processor

## Instruction Pipe

The instruction pipe (or pipeline) is the instruction processing facility of the microMV. Essentially, it implements a process called *pipelining* wherein instructions undergo simultaneous decoding and execution. Before an instruction is actually executed, the appropriate starting microcode address is found. The instruction may also contain a logical address to be resolved or immediate data to be extracted. To perform these tasks, the instructions move through the pipeline in various stages of decoding. At the last stage of the pipeline, the instructions execute in a minimum number of CPU cycles, due to the preprocessing which occurs in the previous stages.

Figure 1-14 shows the four stages of the pipe and identifies the processing that occurs in each stage. As the fourth pipeline stage executes an instruction (N), the third stage calculates the effective address for the following instruction (N+1), the second stage decodes the subsequent instruction (N+2), and first stage fetches another instruction (N+3), putting it into the instruction queue.



**Figure 1-14** *Four-stage instruction pipeline*

Looking at the pipe operation starting at the beginning, in stage 1, the pipe prefetches four words from the instruction stream and loads them into input latches. It then feeds each instruction through decode logic and into an instruction queue. The decode logic reformats the instruction into a starting address to the appropriate microroutines. As one instruction passes to the microsequencer for the beginning of the execution phase, others are in various stages of decoding. The instruction queue maintains a queue of the next two sequential instructions after the current instruction.

## Microsequencer

The microsequencer operates from a read-only control store, which provides microcode for driving the instruction processing. The control store is based in ROM and stores 3K microwords, 64-bits wide. An additional storage area (16 by 64), referred to as the patch control store RAM, or patch memory, stores information for updating the microcode stored in ROM. During the start-up booting, the system loads the microcode update information into the patch memory. These microwords are used in place of microcode words that are to be fixed. Loading from a peripheral simplifies microcode changes and updates.

The microMV processor operates at a microinstruction cycle time of 66.67 nanoseconds. As each microinstruction executes, the output of its control fields drives the remaining chip elements to execute the originally fetched, assembly-level instruction and to keep the pipeline operating. If a microword is flagged as being invalid, the system uses code from the patch memory to complete the instruction.

## Address Generation Unit

When memory reference instructions are in the pipeline, the address generation unit (AGU) calculates their effective logical addresses. In turn, it passes the logical addresses to the address translation unit which begins to convert the logical addresses to physical addresses.

When the processor accesses memory, the AGU provides an additional level of pipelining. By computing the logical addresses, the AGU permits the processor to overlap operand fetches with instruction execution, which accelerates the execution process. The AGU also maintains the processor's stack pointers.

## Address Translation Unit

The address translation unit (ATU) performs the logical-to-physical address conversion. To efficiently implement this conversion, the ATU contains its own cache together with protection and pagetable-access logic. The ATU performs all hardware checks required by the protection system. Access, page, and ring-crossing validations are among the types of hardware checks. If any of these checks fail, the address translator initiates a page or protection fault to the operating system.

The ATU cache stores the most recently used address translations along with the associated page-reference bits used by an operating system for virtual memory management. The cache contains 32 address translations for segments 0 through 7, in addition to associated protection information.

As each conversion transpires, the ATU halts the microsequencer if the needed translation information is not present in its cache. When this occurs, the ATU uses a pagetable look-up mechanism to access the physical address. The ATU also sets the appropriate reference/modify bits for the memory operation. Then it loads the address into the ATU cache, and microinstruction execution proceeds.

## Arithmetic Logic Unit

The arithmetic logic unit (ALU) performs the data manipulation required to execute most arithmetic- and logic-class instructions that deal with fixed-point data. It also supplies memory and I/O data to executing programs via its accumulators.

The ALU itself is supported by two other units: a shifter and a multiplier. The shifter performs operations such as shift-and-rotate. The multiplier is a 2-bit string multiplier for multiplication operations.

The ALU contains four 32-bit, fixed-point accumulators; four 64-bit floating point accumulators; the processor status register; the floating point status registers, and several general registers for internal use.

## Floating Point Assist

For floating-point calculations, the microMV performs the calculations using special microcode routines. The floating point assist hardware aids in exponent sign compare, normalization, and error detection.

## Instruction/Data Cache

The internal instruction/data cache provides a high-speed, temporary storage buffer for the processor. Data enters the 2-Kbyte cache from main memory as the program calls for it. The processor then executes the program directly using the cached data. It can read data from the cache within its 66.67-nanosecond cycle time.

Within the cache, memory is organized into 16-byte blocks. Each block directly maps to a block in main memory, overlapping every 2 Kbytes. A tag store contains the corresponding main memory address for each block. The tag store contains a validity flag indicating whether or not the data within the cache block is valid. The tag store monitors access to cached data from within the processor and also monitors accesses from other processors and I/O devices.

The instruction/data cache uses a technique called write through. Whenever data in the cache is modified by the CPU, the corresponding main memory location is also updated. This procedure ensures that other devices on the system bus will read only current data from main memory. If another device updates a main memory location that is also in the data cache, the tag store monitoring the system bus will invalidate the corresponding cache block. When the CPU accesses data within that block, the cache will first read in the updated block from main memory.

Once data is brought into the cache, it remains there until invalidated or overwritten by another block.

### **Address/Data Interface**

The address/data interface (ADI) circuits provide interfacing to externally implemented address/data buses, linking the microMV to other system elements. Memory addresses transfer over a 32-bit, bidirectional address bus, and system data over a 64-bit data bus. The address/data buses are supported by associated control signal lines. The maximum bandwidth for the buses is 60 megabytes per second.

### **Memory Support**

The microMV supports physical memory addressing for 4.3 gigabytes of external memory.

### **Multiple Processor Support**

The microMV implements the ECLIPSE MV/Family instructions supporting multiple processors in a tightly coupled multiprocessor system. These instructions allow for individual control of the processors, status checks, and cross interrupts. Refer to the chapter, "Device Management" for multiple-processor programming information.

### **Input/Output Support**

The microMV implements the ECLIPSE MV/Family I/O instructions supporting communications with peripherals on the I/O channels — either individually on one I/O channel or simultaneously on multiple I/O channels. For further information, refer to the chapter, "Device Management."

### **Architectural Clock Support**

The microMV implements the architectural clock time-slice timer function. This is a count-down timer that causes a time-slice fault to occur when a specified time slice expires.

## **Architectural Clocks**

The term architectural clocks encompasses three clocks: a time-slice timer, an alarm clock, and a boot clock. The time-slice timer, implemented within the microMV, is a count-down timer that causes a time-slice fault to occur when a specified time slice expires. The alarm clock, implemented within the IOC circuits, is a time-of-day (TOD) clock with an alarm for generating an I/O interrupt. The boot clock, implemented by the



DRP, can be read after powerup to provide the time in hours, minutes, seconds, day, date, and year. (Also see PIT and RTC.) For further information on architectural clocks, refer to Chapter 8.

### PIT and RTC

The system clocks are software accessible timers. The system programmer can select from the architectural clocks (see microMV processor) or a combination of the programmable interval timer (PIT) and the real time clock (RTC).

The PIT is an independent time base that is set to initiate program interrupts at fixed intervals. The RTC generates low-frequency I/O interrupts for performing time calculations independent of system timing. A gate array on the system board implements the PIT and RTC.

### System Cache

The system cache is essentially 64-kilobytes of high-speed memory supporting the microMV processor's performance. It provides interim storage for 4K blocks (64-bits wide, two quad-words per cache block) of data copied from system memory. The cache provides fast-access interim storage, direct mapped and with automatic write-through to system memory. A 64-bit wide data bus and a 32-bit wide address bus (with physical addressing up to 4.3 gigabytes) provide the communications link between the cache and the microMV processor.

A cache update mechanism maintains cache coherency with system memory. It also works closely with the microMV processor's internal 2K cache. If another device writes data to a physical memory address, the system cache informs the microMV processor so that it can invalidate its entry. A write to system memory will overwrite the data in the system cache. The system cache has a 1 cycle (66.67 nanoseconds) read/write access time on a cache hit.

### System Memory

The system memory provides the physically addressed memory storage for the computer. It consists of a memory control unit (MCU) on the system board and up to four memory modules that connect to the system board.

#### Memory Control Unit (MCU)

The MCU provides timing, error checking and correction, dynamic RAM refresh and sniffing, and memory control. A single gate array implements the MCU functions. It controls addressing and data accesses to the dynamic memory arrays and also performs the error checking and correction for the 64-bit wide bus data. The ERCC facility corrects both single- and double-bit errors (except for double-bit soft errors).

The MCU can access memory in four ways: quad-word read, quad-word write, block read, and read-modify write.

## Memory Modules

The memory modules are daughter boards that connect to the system board via connectors. The modules consists of arrays of dynamic RAMs controlled by the memory control unit (MCU). Functionally, the memory arrays connect to the system bus through a set of latches and transceivers associated with the MCU. Each memory module provides either 8 or 32 megabytes of physical memory depending on whether 1- or 4-megabit RAM chips are used. Both capacities of modules may be mixed in a system. With a maximum of four memory modules, the total memory capacity per system ranges from 32 to 128 megabytes, depending on the capacity of the RAM chips used. The memory cycle time for read/writes is 333.33 nanoseconds.

## Diagnostic Remote Processor (DRP)

The diagnostic remote processor (DRP) is an 8-bit microprocessor that drives system powerup, initialization, and diagnosis of both hardware and software problems. It also generates the boot clock portion of the system's architectural clocks. The DRP is supported by power-up code, boot clock, and system control program (SCP). The DRP connects to the system bus through the I/O controller.

## System Bus

The system bus is an address/data bus allowing communications between the main subsystems elements. The bus is internal to the system board and consists of a 32-bit, bidirectional address bus and a 64-bit data bus and associated control signals. The bus supports multiple requestors and an address space of up to 4.3 gigabytes. The maximum bandwidth for the bus is 60 megabytes per second.

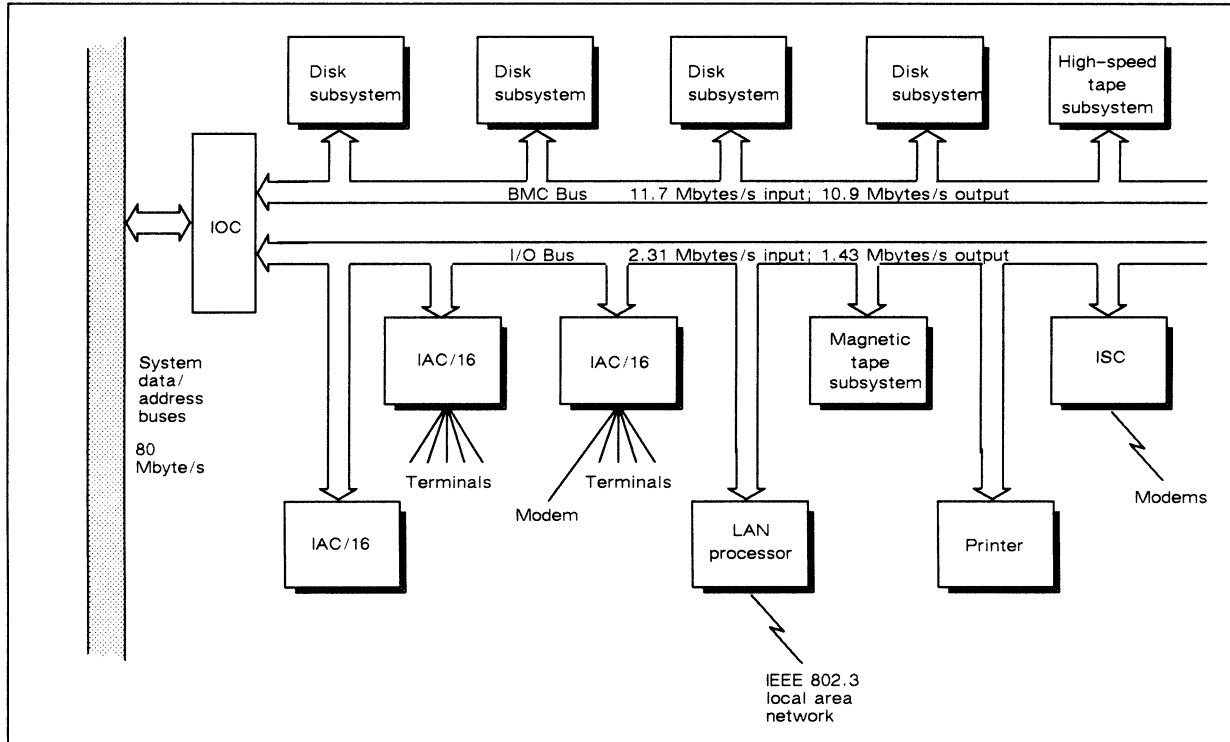
## I/O Controller (IOC)

The I/O controller (IOC) subsystem acts as an intermediate I/O processor, handling data transfers between the I/O device controllers and the microMV or system memory. These communications take place over two separate I/O interface buses: the burst multiplexor channel (BMC) for high-speed block-oriented devices, and the I/O bus for medium-speed data channel devices and low-speed programmed I/O devices. These buses connect to the various I/O controller cards in the card cage. The I/O subsystem also connects directly to the system bus.

Two gate arrays provide basic IOC functions, such as PIO decoding, addressing, parity checking, DCH and BMC bus generation, input/output buffering, data alignment, and upstream map loading. Additional circuits include the I/O map RAM, the alarm clock oscillator, external bus drivers/receivers, and an external PIO pulse decoder.

## Input/Output Subsystem

Figure 1-15 illustrates the input/output (I/O) subsystem for the ECLIPSE MV/9500 computer. The primary responsibility of the I/O subsystem is to handle all communication with the system peripherals. Two buses, the ECLIPSE I/O bus and the burst multiplexor channel (BMC) bus, carry the communicated data.



**Figure 1-15** I/O system

The burst multiplexor channel supports high-speed devices, such as disks and tapes, with direct-memory-access transfers of block-oriented data. The BMC operates at a bandwidth of 11.7 megabytes per second for input from the device and 10.9 megabytes per second for output to the device.

The I/O bus supports medium- and low-speed devices. It provides a data channel facility to support medium-speed devices with direct-memory-access transfers of single words or variable-length blocks of data. The data channel facility of the I/O bus operates at two different bandwidths: normal for transfers within the system chassis, and extended (also referred to as slow data channel) for transfers to an expansion chassis. The normal bandwidth data channel operates at 2.31 megabytes per second for input and 1.43 megabytes per second for output. The slow data channel bandwidth operates at 1.67 megabytes per second for input and 1.0 megabytes per second for output. Data channel controller boards inserted into some slots of the main chassis and into any slot of the first expansion chassis, operate at the extended bandwidth. Data channel controller boards inserted into certain slots of the main chassis and into any slot of the second expansion chassis, operate at normal bandwidth. Refer to the configuration guide for the ECLIPSE MV/9500 system for information on device controller slot assignments.

The I/O bus provides a programmed I/O facility for low-speed transfers of information (instructions, commands, status, or bytes) between a device and an accumulator in the CPU. These transfers are instrumental in setting up the parameters for transfers on the higher speed channels. (Device controllers on the BMC bus also connect to the I/O bus for device setup and data transfer initiation.) The ECLIPSE MV/9500 computer executes all ECLIPSE 16-bit programmed I/O instructions exactly as ECLIPSE 16-bit systems (such as, the ECLIPSE C/350 systems).

For further information, refer to the chapter, "Device Management."

## Communications Controllers

ECLIPSE computers support a wide range of intelligent communications and network controllers that function as full I/O processors. These independent processors include the following:

- Intelligent asynchronous controllers.
- Intelligent synchronous controllers.
- Multicom munications processors.
- Intelligent LAN controllers.
- Intelligent computer-to-PBX (private branch exchange) interfaces.

Each controller is microprocessor based with local memory and industry-standard line interfaces. These I/O processors connect to the I/O bus of the input/output system. They use a cross-interrupt facility for direct communication with the CPU. Data transfers take place to and from main memory via the data channel facility. Refer to the chapter, "Device Management." Also refer to the appropriate communications controller manual for programming information specific to a device.

For information on configuring ECLIPSE I/O controllers into an ECLIPSE MV/9500 computer system, refer to the manual, *Configuring Your ECLIPSE MV/9500™ Computer System*.

## Power System

The power system includes a microprocessor-based controller, called the universal power system controller (UPSC), a voltage nonregulated unit (VNR), a dc-to-dc regulator unit, and optionally, a battery backup unit (BBU).

The UPSC brings up power to the system in a specified sequence and, once power is up, monitors and manages power parameters. Further, to prevent system failures resulting from random ac-line dropout conditions, the UPSC supplies a powerfail/auto-restart facility that allows the system to recover automatically. During power outages, the battery back-up units supply dc power to the CPU chassis and the expansion chassis for up to 2 minutes, giving the operating system time to prepare for an orderly shutdown.

During operations, the UPSCs also check for blower failures and excessive power supply and cabinet temperatures, and can interrupt the CPU to report critical failure conditions.

Refer to the chapter, "Device Management," for programming information for the PSCs.

The system console terminal (typically a basic character display terminal or hard-copy console) connects directly (through pins on the backplane) to an asynchronous communications port on the diagnostic remote processor. The system console normally consists of a keyboard and either a CRT display or a character graphics printer.

The system console terminal is the operator's interface to the system. It provides a convenient way to enter commands that start, stop, and modify the system, as well as receive status and error messages. It is also the interface through which the operator implements the system control program.

Refer to the chapter, "Device Management," for programming information on the primary asynchronous line.

## Control Panel

The control panel contains several mechanically operated switches that control the operation of the computer. In addition to the main on-off power switch, there are other switches for system boot and reset, console reset, and lock. The front panel also has three discrete LEDs that supply status information. For detailed information on the operation of the front panel, refer to *Starting ECLIPSE MV/9500™ Computer Systems*.

## Additional facilities

The ECLIPSE MV/9500 computer system provides the following additional facilities:

### System Control Program

The system control program (SCP) is a ROM-based operating system that runs on both the DRP and the CPU. During full operation, the SCP allows the system operator to load, examine, and modify main memory, and to step through the instructions of a program.

For further information, refer to the manual, *Using the ECLIPSE MV/9500™ System Control Program*. Also refer to the chapter, "Device Management," for programming information on the SCP.

### Error Detection and Logging

Most errors are detected by the CPU. The CPU passes the error information to the DRP through programmed I/O commands. For correctable errors, the CPU notifies the DRP and continues operation. For uncorrectable errors, the CPU notifies the DRP, suspends normal operations, and enters SCP mode. The SCP then attempts to display a message indicating the nature of the error.

When the DRP receives an error message, it records the message in RAM. Each recorded error is time stamped from the time-of-day clock as it is logged. In addition, the DRP keeps track of the frequency of certain correctable memory errors. Through this logging procedure, the operating system tracks correctable errors. The error logging procedure also provides service personnel with important information about both correctable and uncorrectable errors.

### Diagnostic Troubleshooting

The DRP, in combination with diagnostic software, supplies system engineers and service personnel with diagnostic capabilities. Since the DRP has its own separate processor, it can perform certain diagnostic functions independently of the status of the CPUs.

### Remote Diagnostic Communications

In addition to the standard system console interface, the DRP and associated circuits provide a remote access console (RAC) facility. The RAC interface can connect to either a locally installed service console or via a modem to a remote service site. The full set of system console functions is available over the modem, including remote access to the SCP processor.

Another asynchronous port can connect to both a user terminal and an intelligent asynchronous controller, such as an IAC/16. This port enables the modem to communicate with the CPU through a standard IAC connection. It also allows individuals using the modem and the user terminal ports to send messages to each other.

The remote diagnostics facility allows Data General's staff of system engineers to diagnose hardware and software system problems off site.

## System Console

The system console terminal (typically a basic character display terminal or hard-copy console) connects directly to an asynchronous port associated with the DRP on the system processor board.

The system console terminal is the operator's interface to the system. It provides a convenient way to enter commands that start, stop, and modify the system, as well as receive status and error messages. It is also the interface through which the operator implements the system control program.

Refer to the chapter, "Device Management," for programming information on the primary asynchronous line.

## Initialization

The processor assumes the physical mode when the processor first powers up, or after a system reset from either the front panel System Reset switch or the SCP Reset command. When either occurs, the processor

- Disables logical address translation, making logical and physical addresses equal.
- Disables the logical address translation protection system.
- Disables error reporting through the SCP interface.
- Disables interrupts.
- Clears bits 3, 4, 7, 8, and 14 of the I/O channel definition register.

The ECLIPSE I/O bus handles both programmed I/O (PIO) and data channel (DCH) transfers within the system. Programmed I/O transfers 1 byte or word at a time between the CPU's accumulators and I/O devices. These transfers are instrumental in setting up the parameters of the transfers for the higher speed channels. Data channel transfers blocks of data between memory and peripheral devices.

End of Chapter



## Device Management

The processor supports devices that transfer data using a slow-, medium-, or high-speed transfer rate. With a programmed I/O facility, the processor transfers 1 or 2 bytes of data between a device and an accumulator. With a data channel I/O facility, the processor transfers words or blocks of words between a medium-speed device and memory. With a burst multiplexor channel I/O facility, the processor transfers bursts of data between a high-speed device and memory. Note that an MRC and a job processor communicate through message passing. For MRC I/O programming information, contact your Data General sales representative.

For instance, a slow-speed asynchronous line controller transfers data with the programmed I/O facility. Medium-speed devices, such as line printers and magnetic tapes, transfer data with the data channel (DCH) I/O facility. The high-speed disk drives and high-speed magnetic tape drives transfer data with the burst multiplexor channel (BMC) I/O facility.

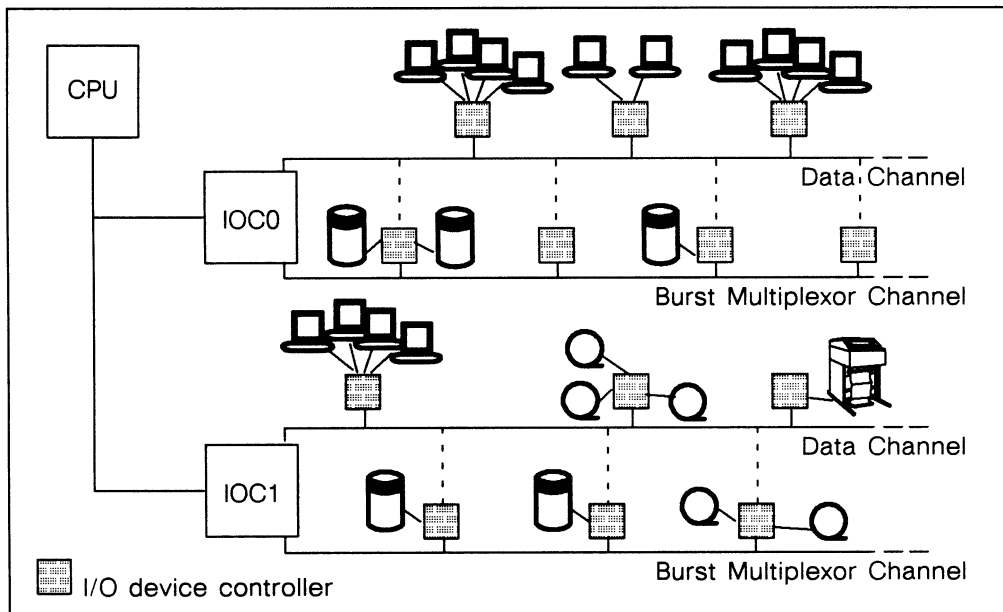
Devices are either external or internal to the computer.

- External devices are those peripherals residing on either the I/O bus or BMC with communications generally handled through a device controller (such as disk drive and magnetic tape units, printers, and terminals).
- Internal devices are those which are integral to the computer and accessible directly by the processor without the necessity of communicating through a device controller (such as the CPU, real-time clock, programmable interval timer, system control processor).

Depending upon the operating system, a device is usually accessed through a system call to an operating system. This chapter presents basic information to assist in reading and writing an interrupt or device handler routine. The chapter first provides general information pertinent to all devices: device access, I/O instructions, and interrupts. It then discusses the data channel and burst multiplexor channel, device controllers in general, and writing device handler routines to support external peripherals. The chapter concludes with information on integral (internal) devices. For descriptions of the structure, functions, signals, and timing characteristics of the I/O buses, refer to the machine-specific interface designer's guide.

## I/O Communication

A peripheral generally consists of one or more devices and a device controller. Communication between the processor and a device is through the device controller using one or more of the I/O facilities (programmed, data channel, burst multiplexor channel). An intermediary for some ECLIPSE MV/Family systems is an I/O channel controller (IOC). An IOC manages access to an I/O channel with each IOC maintaining its own set of buses (generally both a data channel and a burst multiplexor channel). Figure 8-1 shows an ECLIPSE MV/Family system with dual I/O channel controllers.



**Figure 8-1** An ECLIPSE MV/Family system with dual IOCs

The I/O bus is shared by all the device controllers as well as by the computer, while the BMC bus is shared only by the computer and device controllers that incorporate the BMC facility. The I/O bus connects in parallel to each DCH and BMC device controller in the system; the BMC connects in parallel to each device controller that uses the BMC facility. Since these buses are shared they are, by necessity, half-duplex buses with only one operation occurring at any one time. However, an operation can be occurring simultaneously on each bus.

The direction of all I/O transfers is relative to the computer. *Output* refers to information moving from the computer to a device controller; *input* refers to information moving from a device controller to the computer.



once with a **MSKO** instruction using a mask contained in an accumulator. (Each device controller is assigned by its hardware to a bit position in the mask. Mask bit assignments for standard device controllers are given in the machine-specific supplement.) When a **MSKO** instruction executes, each device controllers' interrupt disable flag is set to the value of the assigned bit of the mask (a value of 1 disables the device from posting an interrupt to the processor; a value of 0 enables the posting of an interrupt). Following powerup, or when a reset occurs, all interrupt disable flags are set to 0.

To service an interrupt, the processor first determines the action to take on the currently executing instruction, next redefines the interrupt mask, and finally services the interrupt request. The section, "Processor Interrupt Servicing," explains the processor's actions to transfer program control to the interrupt handler, and then to the interrupt service routine.

## Instruction Interruption

Most instructions that require only a minimum of processor execution time are noninterruptible. For instructions that require more execution time, the processor (if required) interrupts the executing instruction, updates the accumulators, and services the interrupt. If the instruction must continue where it left off (resumable instruction), the processor also sets the processor status register interrupt resume flag (IRES) to 1. After servicing the interrupt, the processor either restarts or resumes the interrupted instruction.

**NOTE:** *Some processors use nonmaskable interrupts (NMIs) to control internal processor events. NMIs are not masked by the state of ION. Thus, any instruction that is interruptable (such as **WBLM**) produces undefined results if the instruction overwrites the executing opcode, regardless of the state of ION. An NMI causes the execution of the instruction to be stopped, and then resumed with updated accumulator values. If the instruction (**WBLM**) overwrites itself, then the opcode may no longer be available when the instruction is resumed.*

ECLIPSE MV/Family processors set the processor status register bit 2 (IRES) to 1 when an interrupt occurs during execution of a resumable instruction. The processors also set PSR bit 3 (IXCT) to 1 if the interrupted instruction was executed by a Pop Block and Execute (**PBX**) instruction.

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

## Processor Interrupt Servicing

To service an interrupt request (Figure 8-3), the processor

1. Sets ION to 0.
2. Determines if the address translation facilities are enabled.  
 If address translation is enabled, the processor continues with step 3.  
 If address translation is disabled, the processor gets the interrupt handler pointer from physical location  $1_8$  and continues with step 7.  
 Refer to the chapter, "Memory and System Management," for information on enabling and disabling the address translation facilities.
3. Stores the current stack register values in the respective page zero locations of the current segment.
4. Changes the current segment of execution to segment 0.
5. Initializes the wide stack using the values from page zero locations of segment 0.
6. Fetches the interrupt handler pointer from word  $1_8$  of reserved page zero memory for segment 0.
7. Resolves the effective address of the interrupt handler.
8. Examines the first word of the interrupt handler, which may be one of the following types:
  - Type 1 — an ECLIPSE MV/Family 32-bit processor instruction.  
 A 32-bit processor instruction contains bit 0 equal to 1 and bits 12-15 equal to  $1001_2$ .
  - Type 2 — an ECLIPSE 16-bit instruction.  
 Instructions other than **XVCT** or type 1 are identified as ECLIPSE 16-bit instructions.
  - Type 3 — a vector interrupt (**XVCT**) instruction.
9. Stores the return address in the following locations (according to instruction type):
  - Type 1 — Logical locations  $2_8$  and  $3_8$  of segment 0.
  - Type 2 — Location 0 of segment 0.
  - Type 3 — The vector stack (as part of the return block) for the **XVCT** instruction, during the vector interrupt processing.
10. Jumps indirectly (according to instruction type) as follows:
  - Type 1 — To the immediate interrupt handler and executes the type 1 instruction as the first instruction of the handler.  
 A jump instruction (**LJMP** or **XJMP**) can be used to jump indirectly through the return address in order to return from the interrupt handler.
  - Type 2 — To the ECLIPSE interrupt handler and executes the type 2 instruction as the first instruction of the ECLIPSE interrupt handler.
  - Type 3 — To the vectored interrupt handler (through word  $7_8$  of page zero for segment 0) and executes the **XVCT** instruction. The next section describes vectored interrupt processing.  
 The last instruction of the vectored interrupt handler should be a wide restore from vector interrupt instruction (**WRSTR**), which pops the wide return block from the vector stack.

## Transfer Sequence

The entire I/O transfer sequence is synchronized by clock signals generated by the respective bus controller (BMC or DCH). For timing information, refer to the machine-specific interface designer's guide. The actual transfer sequence is a two-way communication between the bus controller and the device controller that proceeds as follows.

When a device controller has a word or block of data ready for transfer to memory or wants to receive data from memory, it issues a request to its respective bus controller. If the device controller has bus priority and no other controllers are active on that channel, the bus controller begins the cycle by acknowledging the device controller's request. The acknowledgment signal causes the device controller to send the bus controller the starting memory address and the direction of the transfer. Intelligent device controllers may also send the type of transfer (data or map load), and BMC device controllers may also specify the mode of addressing (physical or logical) for the transfer.

Following the receipt of the address, the data itself is transferred in the specified direction on the appropriate bus.

Each time a data transfer is completed, the interaction between the bus controller and the device controller is over. The device controller carries out any tasks necessary to complete the data transfer, such as transferring the data to the device itself for an output operation.

As each word (or word of a burst) is transferred, the device controller increments its memory address register to point to the next memory location. If the word/block counter is used as a

- Word counter, the device controller also increments the counter as each word (or word of a burst) is transferred.
- Block counter, the device controller increments the counter only when each block has been transferred.

When the word/block counter becomes 0, the device controller typically terminates further transfers, sets its Busy flag to 0 and its Done flag to 1, and initiates a program interrupt request.

If the counter is not yet 0, the device controller continues the operation, issuing another request when it is ready for the next transfer.

## Device Maps and Data Transfers

A device memory allocation and protection (MAP) feature provides allocation of memory for I/O access. The MAP allows physical memory to be allocated in 2-kilobyte blocks (pages). During I/O operations, these pages are selected with a logical address that the MAP translates into a physical address for accessing memory.

The DCH or BMC facility uses a device map in either unmapped or mapped mode.

- In unmapped mode, the processor passes the word address directly to memory, as a physical address. You can use the load physical address (**LPHY**) instruction to translate a logical address to a physical address and store it in an accumulator. (The logical address must point to a current or higher number segment.) Then, send the physical address to the device, using an I/O instruction.
- In mapped mode, the processor uses the device map and the word address to translate the most significant bits of the logical address to a physical page number. The processor then concatenates the physical page number to the 10 least significant bits of the logical address to form the physical address.

Some I/O device controllers can load their own mapping information. Both the DCH and BMC allow device controllers to load their own map locations (slots). This is called “upstream map loading.” (Map slots for device controllers that do not include this provision are loaded by program control.)

Upstream map loading is performed by the device controller requesting access to its respective bus controller. When access is granted, the device controller specifies a map load operation along with a map slot address, and then proceeds with the transfer as previously described. The data transferred to the respective channel bus controller is placed into the I/O map on the respective bus controller rather than being transferred into memory.

Table 8-7 lists the I/O instructions that affect a device map (a DCH map or BMC map).

**NOTE:** *Loading a data channel map from a device while DCH mapping is disabled (bit 14 of the I/O channel definition register is set to 0) will produce undefined results.*

**Table 8-7** I/O instructions for DCH/BMC maps

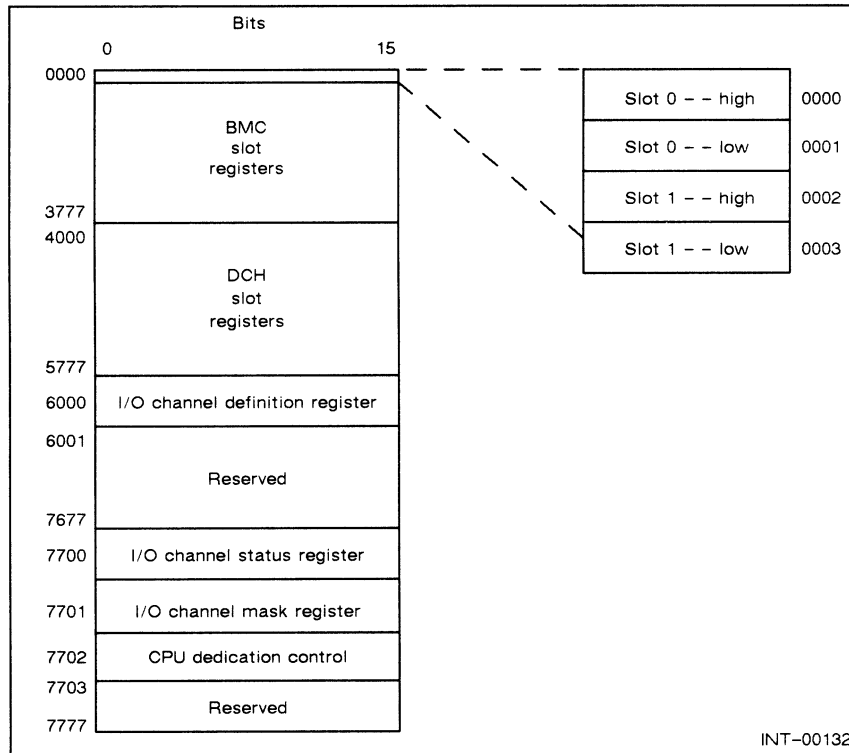
Instruction	Operation
<b>CIO, CIOI</b>	Returns BMC/DCH status or loads map slots from accumulators (1/2 slot at a time).
<b>IORST *</b>	Sends a reset signal to all devices on all I/O channels to clear their states and turns off DCH and BMC mapping (clears bits 3, 4, 7, 8, and 14 of the I/O channel definition register).
<b>WLMP</b>	Loads BMC/DCH map slots from memory (as a block of doublewords).
<b>LPHY</b>	Translates a logical address to a physical address, loading the result into an accumulator (for use in the unmapped mode).

\* ECLIPSE compatible instruction

The **CIO**, **CIOI**, and **WLMP** instructions initiate DCH/BMC map loads and return status information when in mapped mode. Use the **LPHY** instruction for map loads in unmapped mode. The DCH or BMC sets its Busy flag to 1 when a map load or read is in progress. Neither channel has a Done flag, and the channels never cause program interrupts.

## DCH/BMC Registers

ECLIPSE MV/Family systems contain 512 DCH slot registers and 1024 BMC slot registers. These slot registers and the I/O channel registers are numbered from 0 through 7777<sub>8</sub>, as depicted in Figure 8–8 and explained in Table 8–8. The DCH and BMC map registers contain page number and access information. The I/O channel registers contain status and control information which affect DCH and BMC maps and data transfers. The figures and tables that follow describe the formats for each of the registers.



**Figure 8–8** DCH/BMC registers

**Table 8–8** I/O registers

Registers (Octal)	Description
0000–3776	Even-numbered registers are the most significant half of BMC map slots 0–1777.
0001–3777	Odd-numbered registers are the least significant half of BMC map slots 0–1777.
4000–5776	Even-numbered registers are the most significant half of DCH map slots 0–777.
4001–5777	Odd-numbered registers are the least significant half of DCH map slots 0–777.
6000	I/O channel definition register.
6001–7677	Reserved.
7700	I/O channel status register.
7701	I/O channel mask register.
7702	CPU dedication control.
7703–7777	Reserved.

## BMC/DCH Slot Register Formats

The processor translates the contents of the BMC and DCH even slot registers (0000–3776<sub>8</sub> and 4000–5776<sub>8</sub>, respectively) as diagrammed below.

V	D	Reserved								Physical page number				
0	1	2							10	11				15

The processor translates the contents of the BMC and DCH odd slot registers (0000–3777<sub>8</sub> and 4001–5777<sub>8</sub>, respectively) as diagrammed below.

Physical page number													
0													15

Bits	Name	Contents or Function
0	V	Map validity bit If 0, processor allows access to page. If 1, processor denies access to page.
1	D	Data bit If 0, the channel transfers data to device or memory. If 1, the channel transfers zeros to device or memory.
2–10	Reserved	Reserved for use by the hardware. Write to with zeros; reading these bits returns an undefined state.
11–15 (odd) 0–15 (even)	Physical Page Number	The physical page number that the map uses when translating from a logical to a physical address.

## I/O Channel Definition Register Format

The I/O channel definition register (6000<sub>8</sub>) provides error and status information.

ICE	Reserved		BVE	DVE	DCH	BMC	BAP	BDP	R	I/O Channel			DME	1
0	1	2	3	4	5	6	7	8	9	10		13	14	15

**NOTE:** Writing a 1 into bits 3, 4, 7, or 8 complements these bits. The **IORST** instruction clears bits 3, 4, 7, 8, and 14.

Bits	Name	Contents or Function
0	ICE	I/O channel error flag; if 1, an error has occurred on the I/O channel (0 only when all other error bits are 0 — read-only bit).
1, 2	Reserved	Reserved for future use and returned as zero.
3	BVE	BMC validity error flag; if 1, BMC address validity protect error has occurred.
4	DVE	DCH validity error flag; if 1, DCH address validity protect error has occurred.
5	DCH	DCH transfer flag; if 1, a DCH transaction is in progress (read-only bit).
6	BMC	BMC transfer flag; if 1, a BMC transfer is in progress (read-only bit).
7	BAP	BMC address error; if 1, the channel has detected an address parity error.
8	BDP	BMC data error; if 1, the channel has detected a data parity error.
9	R	Reserved and returned as zero.
10–13	I/O Channel	I/O channel number.
14	DME	DCH mode; if 1, DCH mapping is enabled.
15	1	Always set to 1.

## I/O Channel Status Register Format

The read-only I/O channel status register (7700<sub>8</sub>) provides I/O channel status information.

ERR	Reserved							DTO	MPE	1	1	CMB	INT
0	1						9	10	11	12	13	14	15

Bits	Name	Contents or Function
0	ERR	Error; if 1, the I/O channel has detected an error. This bit is set to 1 if any error-indicating bit in the IOC status register is set to 1.
1–9	Reserved	Bits 1 through 9 are reserved for future use.
10	DTO	DCH time-out error; if 1, a DCH read-modify-write operation time-out error has occurred.
11	MPE	Map parity error; if 1, a map parity error has occurred.
12	1	Always set to 1, indicating extended DCH map slots and operations are supported.
13	1	Always set to 1.
14	CMB	Current state of the mask bit for this I/O channel (refer to the I/O channel mask register format description).
15	INT	Interrupt pending; if 1, the channel is attempting to interrupt the CPU.

## I/O Channel Mask Register Format

The write-only I/O channel mask register (7701<sub>8</sub>) specifies a mask flag for each channel. When an I/O channel mask flag is set to 1, the processor ignores all interrupt requests from devices on that channel.

NOTE: *The ECLIPSE MV/9500 system supports one I/O channel.*

The Interrupt Acknowledge instruction (**INTA**) using an I/O channel number from 0 through 6 returns the device code of the highest priority interrupting device on that channel, with its Done flag set to 1. For channel 7, the **INTA** instruction returns the device code of the highest priority interrupting device on the highest priority channel, regardless of the state of the I/O channel mask register flags.

An I/O Channel Reset instruction (**PRTRST**) sets the mask bit to 0 for a single channel (0 through 6) or for all channels (7).

NOTE: *A CIO read to the I/O channel mask register produces undefined results.*

The format of the I/O channel mask register is as diagrammed below.

Reserved							C0	C1	C2	C3	C4	C5	C6	Res.
0						7	8	9	10	11	12	13	14	15

Bits	Name	Contents or Function
0–7	Reserved	Reserved for future use; should be set to 0.
8	C0	I/O channel 0 mask *
9	C1	I/O channel 1 mask *
10	C2	I/O channel 2 mask *
11	C3	I/O channel 3 mask *
12	C4	I/O channel 4 mask *
13	C5	I/O channel 5 mask *
14	C6	I/O channel 6 mask *
15	Reserved	Reserved and set to 0.

\* If 1, prevents all devices connected to the indicated I/O channel from interrupting the processor. A system reset sets C0 to zero and sets C1 through C6 to 1.

## CPU Dedication Control Register Format

Each I/O channel contains a 16-bit register (7702<sub>8</sub>) that controls which processor is to receive I/O interrupts. This register is applicable only to multiple-processor configurations; systems that support only a single processor ignore the contents of this register. This read/write register is available only while a multiple-processor system is in dedicated mode. An attempt to read or write this register while the system is in any mode other than dedicated may cause unpredictable results (refer to the section, "Multiple Central Processing Units," for more information on operating modes).

The format of the CPU dedication control register is as diagrammed below.

Reserved				CPU	
0		7		8	15

Bits	Name	Contents or Function
0-7	Reserved	Reserved for future use; must be set to 0.
8-15	CPU	Processor number to which all NOVA type interrupts (except cross interrupts) will be directed. Applicable value for ECLIPSE MV/9500 system is 0. On a system reset, this number is set to the value of the initial processor. Upon execution of an IORST instruction, this number is set to the value of the processor that issued the IORST instruction.



## Data Channel and Burst Multiplexor Channel

Time constraints may also be encountered when transferring data via the data channel or burst multiplexor channel. When a device needs service, it makes a request. More than one device, however, may be waiting to access the DCH or BMC at any one time. Consequently, a significant delay may occur between the time when a device requests access to the channel and the time when the transfer actually occurs. This delay or latency also includes the time required to complete transfers to or from any higher priority devices that are also requesting channel access.

The length of DCH or BMC latency depends on the number of DCH or BMC devices operating in the system at a higher priority and the frequency of their use.

Most devices operate under fixed time constraints. For devices such as disk drives, diskette drives, or magnetic tape transports, if data is not read or written at the correct instant, the controller will have to wait for another revolution of the disk, or in the case of tape, reverse tape direction and perform the operation again. (Since BMC device controllers transfer data in bursts, the time constraints are related to the size of the data burst.)

Consequently, on input, such devices must be allowed to write a word (or burst of words) into memory before the next word (or burst) is assembled by the device controller. On output, the device controller must be able to read the data from memory before the surface is positioned under the write head. In either case, if the latency time is too long, data cannot be properly transferred. Most devices operating under either DCH or BMC control set an error flag (data late) when this happens, so that the service routine can take appropriate action to recover from the error, if possible. Most magnetic media device controllers also provide extra levels of data buffering or larger data buffers that usually eliminate data late issues. With these controllers, no data is moved to memory or written to the device until the device controller's buffers are full.

The maximum allowable latency of a device is the longest time the device can wait for a transfer. During system configuration, DCH or BMC priorities should be assigned to devices on the same basis as programmed I/O devices, that is, by considering the following parameters:

- The maximum allowable latency of the device. A device with a short allowable latency should usually receive a higher priority than one with a long allowable latency.
- The recovery time of a device (how long before it can repeat a transfer that failed because of excessive latency) if the device can recover.
- The cost of losing data from the device if the device cannot recover.

**NOTE:** *Device controllers that have the potential for using large portions of the data channel bandwidth and effectively locking out controllers of lower priority should be placed in lower priority positions. These controllers generally include local area networks (LANs), intelligent synchronous controllers (ISCs), network bus adapters (NBAs), and graphics display controllers (GDCs).*

DCH latency might be improved by less frequent use of I/O instructions. In addition, there is an upper limit on the number of DCH transfers per second that an I/O channel can support. In cases where this limit is exceeded, one solution is to reduce the number of devices using the data channel at the same time. Refer to the machine-specific supplement for DCH and BMC bandwidths.

BMC latency might be improved by decreasing the size of the data burst from the BMC device controllers. Reducing the size of the data burst means a BMC device controller must request BMC bus controller service more often and the controller must buffer fewer words before requesting service.

## Integral Devices

The following sections of this chapter describe instructions for manipulation of these integral devices:

- Central processing unit
- Timing mechanisms (architectural clocks or programmable interval timer and real-time clock) \*
- Primary asynchronous line input/output
- System control processor (or program)
- Data channel and burst multiplexor channel
- Universal power supply controller †
- Power supply controller †

\* *The ECLIPSE MV/9500 system supports either the architectural clocks or the programmable interval timer and real-time clock. The LCS instruction uses the contents of AC3 to determine which timing mechanism microcode to load. Refer to the appendix, "Load Control Store Instruction."*

† *The ECLIPSE MV/9500 system supports the universal power supply controller (UPSC).*

The "Standard I/O Device Codes" appendix lists device codes, device mnemonics, and priority mask bit assignments.

## Central Processor

<b>Device Code</b>	77 <sub>8</sub>
<b>Assembler Mnemonic</b>	CPU
<b>Priority Mask Bit</b>	None

The central processor (CPU) is considered an internal device with control and status flags and is accessible using I/O instructions. The control flag is the interrupt on flag; the status flag is the powerfail flag (refer to the section, "General I/O Instructions"). The I/O instructions to the CPU may use either the standard I/O instruction form, or a special CPU-specific form. Some CPU-specific instructions may be interpreted differently from their standard I/O instruction equivalent. For information on ECLIPSE MV/Family systems that may support more than one central processor, refer to the section, "Multiple Central Processing Units."

## Device Flag Control

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 on flag (ION) equals 1, the processor can interrupt the program (once the instruction following the enable has begun). The processor cannot interrupt the program when the interrupt on flag equals 0. The CPU interrupt on flag is controlled by the device flag commands as follows:

<i>f</i> =omitted	Leaves ION unchanged.
<i>f</i> =S	Sets ION to 1.
<i>f</i> =C	Sets ION to 0.
<i>f</i> =P	Has no effect.

## Boot Clock

Each ECLIPSE MV/Family system contains one boot clock. Access to the boot clock is provided by the SCP interface on device code 45<sub>g</sub>. The boot clock can be read after powerup to get the initial time of day. The boot clock parameters are machine-dependent, but in general, the clock

- Is powered by a battery during a powerfail condition.
- Provides at least 1-second resolution (even when on battery).
- Returns at least hours, minutes, seconds, day, date, and year.

Refer to the SCP section of the machine-specific supplement for further information on boot clock support.

## Programmable Interval Timer

Device Code	43 <sub>8</sub>
Assembler Mnemonic	PIT
Priority Mask Bit	6 or 11 (See machine-specific appendix, "Standard I/O Device Codes")

The programmable interval timer (PIT) is a CPU-independent time base that is set to initiate program interrupts at fixed intervals ranging from 100 microseconds to 6.5536 seconds in increments of 100 microseconds. The PIT can also be sampled with I/O instructions at any point in its cycle to determine the time that will elapse before the next interrupt. 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 goes from 177777<sub>8</sub> to 0. If interrupts are enabled, the PIT then initiates a program interrupt request. Since the counter continues incrementing after reaching 0, it is necessary to reload the PIT counter with the initial value to restart the count. A Busy flag and a Done flag control the operation of the device.

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 Flag Control

Device flag commands to the PIT start or stop the counting cycle for program interrupts.

<i>f</i> =omitted	Leaves Busy and Done flags unchanged.
<i>f</i> =S	Sets the Busy flag to 1 and the Done and interrupt request flags to 0; begins the counting cycle.
<i>f</i> =C	Sets the Busy and Done flags and the interrupt request flag to 0; stops the counting cycle.
<i>f</i> =P	Has no effect.

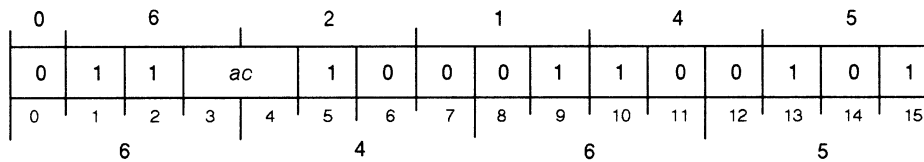
### PIT Instructions

Table 8-14 lists the I/O instructions that affect the PIT device.

**Table 8-14** *Instructions affecting the PIT*

Assembler Statement	Function
DIA [ <i>f</i> ] <i>ac</i> ,PIT	Places the PIT counter value into the accumulator.
DOA [ <i>f</i> ] <i>ac</i> ,PIT	Loads the initial count register with the value in the accumulator.
IORST	Stops the counting cycle and sets the Busy and Done flags, the interrupt mask bit, and the counter to 0.

## Enable/Disable Error Reporting

DOBS *ac*, SCP

Function: SCP error reporting → enable/disable  
 Perform command  
 1 → Busy  
 0 → Done

Parameters: *ac* = command → unchanged

The Enable/Disable Error Reporting instruction sets the SCP Busy flag, clears the Done flag, and uses the contents of the specified accumulator to enable or disable CPU error reporting and to perform the command (function) contained in the command field.

### Arguments

*ac*(16–31)

Before execution, contains function word as follows:

Bits	Contents or Function
16	This bit enables or disables the SCP error reporting. 1 = enable; 0 = disable
17–23	Command. The SCP performs the function defined by these bits: <b>Command</b> (octal) <b>Name</b> 000          No-op 001–003      Reserved 004          Set block 005          Enable all ERCC 006          Reserved 007          Mask soft ERCC 010          Mask all sniff error reporting 011          Disable all ERCC 012          Size 013          Set boot clock time 014          Get boot clock time 015          Set GMT offset 016          Get GMT offset 017          Reserved (used by Data General's AOS/VS) 020          Initiate MI call 021, 022      Reserved (used by Data General's Field Service) 023          Set Auto-Restart Address 024–176      Undefined 177          Enter diagnostic sequence
24–31	Interface Block. The CPU or SCP uses the contents of these bits dependent upon the command in bits 17–23. For any command requiring a CPU/SCP buffer, this value is a physical address pointer to a multiple word block in page zero (in the range of 0 to 377 <sub>8</sub> ). For the Size command (012 <sub>8</sub> ), this value is a type indicator.

The following explains the enable/disable SCP error reporting bit and the SCP commands.

The enable/disable bit (bit 16) enables or disables CPU error reporting. When the CPU or SCP reports an error, it uses the page zero address specified by the interface block (bits 24–31) as a pointer to a doubleword physical address. This address in turn points to a 16-word

block that the CPU or SCP can use to report error data. The first word of the block receives the error code. The remaining 15 words are available for reporting extended error status information. (The actual buffer lengths are specified by the functions that use them.)

If the SCP interrupts the CPU, the SCP disables error reporting until the CPU issues a new enable command.

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

The allocated and implemented commands (bits 17–23) are defined as follows:

- No-op (command 000<sub>8</sub>)  
The No-op command enables or disables SCP-to-CPU logging only; no other function is specified. The command enables ERCC error reporting, but does not change the current ERCC reporting mode.
- Reserved (commands 001<sub>8</sub> through 003<sub>8</sub>)
- Select SCP Powerdown Mode (command 002<sub>8</sub>)
- Disable SCP Powerdown Mode (command 003<sub>8</sub>)
- Set Block (command 004<sub>8</sub>)  
The Set Block command specifies to the SCP the address of the interface block (bits 24–31).
- Enable All ERCC Error Reporting (command 005<sub>8</sub>)  
The Enable All ERCC Error Reporting command enables the SCP to detect and report any memory error to the CPU. This function requires an SCP-to-CPU buffer of five words. The SCP detects the following ERCC errors:  

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

NOTE: *After a system reset or power restoration, reporting of ERCC codes is turned off until an Enable All ERCC Error Reporting command is issued.*
- Reserved (command 006<sub>8</sub>)
- Mask Soft ERCC (command 007<sub>8</sub>)  
The Mask Soft ERCC command disables the reporting of all one-bit correctable ERCC errors. Only two-bit correctable errors and all uncorrectable errors are reported.

- Mask All Sniff Error Reporting (command 010<sub>8</sub>)

The Mask All Sniff Error Reporting command disables the reporting of all soft-sniff and hard-sniff errors occurring during memory refresh.

- Disable All ERCC Error Reporting (command 011<sub>8</sub>)

The Disable All ERCC Error Reporting command tells the SCP to disable all memory error reporting and detection.

- Size (command 012<sub>8</sub>)

The Size command returns various system parameters, such as memory size, microcode revision, and node number. The Size command uses bits 24–31 of the accumulator to contain the value for the type of information being requested. The SCP requires that the Size command disable error reporting (bit 16 equals 0). Since the Size command uses word 0 of the SCP-to-CPU buffer to contain certain status data, the setting of word 0 to 0 has no meaning for this command. However, when the buffer is full, the SCP writes a status word with a value of 0 into its B register.

The types currently specified in bits 24–31 are listed below and described in the following tables along with the contents of their CPU-to-SCP and SCP-to-CPU buffers.

**Type # Returns Information on**

1	Reserved
2	Memory
3	Reserved

Buffer	Address Offset (Octal)	Value
CPU-to-SCP	+0,1	Physical page number which must be at the beginning of a module
DOB register		5002 <sub>8</sub> . 0A02 <sub>16</sub>
SCP-to-CPU	+0	Protocol word (< >0 when buffer is valid)
	+1	Error code
	+2,3	Number of pages
	+4,5	Indicates memory type. Bit 0 of offset 4 defines whether standard main memory or special memory. The offset breaks down as follows:
		<b>Bit(s) Contents</b>
		0 0 Indicates standard main memory
		1 Indicates special memory*
		1–9 Reserved
		10–15 Slot number (encoded)
		16–31 Memory Type
	+6,7	Status of memory area
		0 = Believed to be good
		Nonzero = undefined
DIB register		4 if command completed successfully
		3 if command not completed

\* Special memory is not implemented on the ECLIPSE MV/9500 computer system.

- Set Boot Clock (command 013<sub>8</sub>)

The Set Boot Clock command disables error reporting and sends new boot clock values to the SCP. This command uses seven words of the CPU-to-SCP buffer as follows (words one through six must be binary coded decimal values):

Word Offset Contents	
0	Nonzero protocol word
1	Seconds
2	Minutes
3	Hours
4	Days
5	Months
6	Years

Upon completion, the SCP sets word 0 of the SCP-to-CPU buffer to 0, returns a status value to its B register, and sets the Done flag. A status value of 2 indicates an acknowledgement; a value of 3 means the SCP could not perform this function (no further action is taken by the SCP). Note that if the boot clock is changed through the SCP-CLI, the SCP places the value 0 into its B register, enters code 206<sub>8</sub> into its error log, and posts an interrupt.

- Get Boot Clock (command 014<sub>8</sub>)

The Get Boot Clock command disables error reporting and requests the boot clock time from the SCP. The SCP writes the appropriate information into the SCP-to-CPU buffer as follows (words one through six are binary coded decimal values):

Word Offset Contents	
0	1 (sub-code indicating this buffer contains boot clock data)
1	Seconds
2	Minutes
3	Hours
4	Days
5	Months
6	Years

Upon completion, the SCP returns a status value to its B register, and sets the Done flag. A status value of 4 indicates that the requested information is now in the buffer; a value of 3 means the SCP could not perform this function (no further action is taken by the SCP).

- Set GMT Offset (command 015<sub>8</sub>)

The Set GMT Offset command disables error reporting and sends new Greenwich Mean Time values to the SCP. This command uses eight words of the CPU-to-SCP buffer as follows (words one through six must be binary coded decimal values):



Word Offset Contents	
0	Nonzero protocol word
1	Seconds
2	Minutes
3	Hours
4	Days
5	Months
6	Years
7	Offset sign (0 = positive. 1 = negative)

Upon completion, the SCP sets word 0 of the SCP-to-CPU buffer to 0, returns a status value to its B register, and sets the Done flag. A status value of 2 indicates an acknowledgement; a value of 3 means the SCP could not perform this function (no further action is taken by the SCP). Note that if the GMT offset is changed through the SCP-CLI, the SCP places the value 0 into its B register, enters code 207<sub>8</sub> into its error log, and posts an interrupt.

- Get GMT Offset (command 016<sub>8</sub>)

The Get GMT Offset command disables error reporting and requests the Greenwich Mean Time from the SCP. The SCP writes the appropriate information into the SCP-to-CPU buffer as follows (words one through six are binary coded decimal values):

Word Offset Contents	
0	2 (sub-code indicating the buffer contains GMT offset data)
1	Seconds
2	Minutes
3	Hours
4	Days
5	Months
6	Years
7	Offset sign (0 = positive. 1 = negative)

Upon completion, the SCP returns a status value to its B register, and sets the Done flag. A status value of 4 indicates that the requested information is now in the buffer; a value of 3 means the SCP could not perform this function (no further action is taken by the SCP).

- Undefined (command 017<sub>8</sub>)
- Initiate MI call (command 020<sub>8</sub>)

The Initiate Machine Initiated (MI) Call command initiates a call to the Remote Assistance Center (RAC) without implying that an operating system panic has occurred. The command requires operating system support, otherwise it functions as a NOP. The command's buffer requirements and address offsets are as follows:

Buffer	Address Offset (octal)	Value	Description
CPU-to-SCP	+0	Non-zero "protocol" word	
	+1	Number of words to be transferred (n)	
	+2	Word 1	
	+3	Word 2	
	+4	Word 3	
	...		
	...		
	+(n+1)	Word n	
DOB register		10000 <sub>8</sub> , 1000 <sub>16</sub>	
SCP-to-CPU		Not used. Word 0 of CPU-to-SCP buffer reset to 0.	
DIB register		2 if command completed successfully 3 if command not completed	

- Reserved (commands 021<sub>8</sub>, 022<sub>8</sub>)

These commands are reserved for Data General Field Service.

- Set Auto Restart Address (command 023<sub>8</sub>)

The Set Auto Restart Address command allows the operating system to select whether the SCP should restart program execution after a hardware-detected fatal error and at what address the restart should occur. The restart would follow an automatic hardware-initiated reboot of the system. The command requires compatible hardware and the AOS/VS II operating system. Auto-restart should be disabled during a normal power-up operation. If a hard error occurs with the auto-restart function disabled, the SCP halts the CPU and jumps to the SCP CLI. The command's buffer requirements and address offsets are as follows:

Buffer	Address Offset (octal)	Value
CPU-to-SCP	+0	Non-zero "protocol" word
	+1,2	Auto-Reboot physical word address (32 bits). If the address is -1; disables auto restart.
DOB register		11400 <sub>8</sub> , 1300 <sub>16</sub>
SCP-to-CPU		Not used. Word 0 of CPU-to-SCP buffer reset to 0.
DIB register		2 if command completed successfully 3 if command not completed

- Undefined (commands 024<sub>8</sub> through 176<sub>8</sub>)
- Enter Diagnostic Sequence (command 177<sub>8</sub>)

The Enter Diagnostic Sequence command disables CPU error reporting and all previously enabled functions. The SCP does not use the interface block address entered with this **DOBS SCP** instruction. Instead, the SCP uses the previous 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 CPU issues another **DOBS SCP** instruction.

When the CPU issues the second **DOBS SCP** instruction, the SCP first places the contents of bits 16–31 of the specified accumulator into word 0 of the SCP-to-CPU buffer. The SCP then reads words 1–7 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 returns a status value of 0 to its B register, sets the Done flag, and interrupts the CPU.

NOTES: *The following will also clear the SCP diagnostic mode: an **IORST** instruction, or the SCP-CLI commands, RESET, START, and BOOT.*

*The SCP-to-CPU interface block address is lost when diagnostic mode is terminated.*

### Registers, Flags and Stacks

AC0–AC3	Can be individually specified as <i>ac</i> ; otherwise unused.
Busy flag	Set to 1
Carry	Unchanged
Done flag	Set to 0
<i>Overflow</i>	Unaffected
PC	PC + 1
PSR	Unchanged
Stack	Unchanged

### Related Instructions

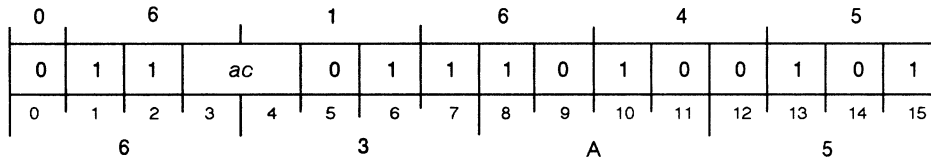
None

### Exceptions

Issuing the **IORST** instruction causes single-bit errors to be checked and corrected, but not reported.

## Return SCP Status

DIBC *ac*,SCP



Function: Status code → *ac*

0 → Busy

0 → Done

Parameters: *ac* = ? → status code

The Return SCP Status instruction clears the SCP Busy and Done flags and returns a code to the specified accumulator denoting the current status of the SCP. The CPU expects all information except status information to be passed using the SCP-to-CPU buffer. If this buffer contains information, the CPU does not expect this data to be valid until after it issues the **DIB** instruction.

### Arguments

*ac*(16–31)

After execution, contains codes denoting current status of SCP as follows:

Status (octal)	Meaning																								
000000	Log information is in current SCP-to-CPU buffer. SCP logging is disabled. The first word of the log buffer (word 0) is interpreted as a log code; the use of the remaining 15 words is dependent on the log code (extended status). The status codes and their definitions are: <b>Code Definition (octal)</b> 007 Powerfail detected 050 Power restore detected 053 Single-bit ERCC error detected (see ERCC extended status) 054 Multiple-bit ERCC error detected (see ERCC extended status) 055 Sniff or I/O detected multiple-bit ERCC error (see ERCC Extended Status) <b>ERCC Extended Status</b> The four-word extended status for ERCC codes 053, 054, and 055 is <table> <tr> <th>Word</th><th>Contents</th></tr> <tr> <td>0</td><td>Status code (053, 054, or 055)</td></tr> <tr> <td>1</td><td>Status <table> <tr> <th>Bits</th><th>Definition</th></tr> <tr> <td>0–11</td><td>Reserved</td></tr> <tr> <td>12</td><td>CPU access</td></tr> <tr> <td>13</td><td>I/O access</td></tr> <tr> <td>14</td><td>Reserved</td></tr> <tr> <td>15</td><td>Sniff access</td></tr> </table></td></tr> <tr> <td>2</td><td>Physical page number</td></tr> <tr> <td>3</td><td>Doubleword on module</td></tr> <tr> <td>4</td><td>Syndrome bits</td></tr> </table>	Word	Contents	0	Status code (053, 054, or 055)	1	Status <table> <tr> <th>Bits</th><th>Definition</th></tr> <tr> <td>0–11</td><td>Reserved</td></tr> <tr> <td>12</td><td>CPU access</td></tr> <tr> <td>13</td><td>I/O access</td></tr> <tr> <td>14</td><td>Reserved</td></tr> <tr> <td>15</td><td>Sniff access</td></tr> </table>	Bits	Definition	0–11	Reserved	12	CPU access	13	I/O access	14	Reserved	15	Sniff access	2	Physical page number	3	Doubleword on module	4	Syndrome bits
Word	Contents																								
0	Status code (053, 054, or 055)																								
1	Status <table> <tr> <th>Bits</th><th>Definition</th></tr> <tr> <td>0–11</td><td>Reserved</td></tr> <tr> <td>12</td><td>CPU access</td></tr> <tr> <td>13</td><td>I/O access</td></tr> <tr> <td>14</td><td>Reserved</td></tr> <tr> <td>15</td><td>Sniff access</td></tr> </table>	Bits	Definition	0–11	Reserved	12	CPU access	13	I/O access	14	Reserved	15	Sniff access												
Bits	Definition																								
0–11	Reserved																								
12	CPU access																								
13	I/O access																								
14	Reserved																								
15	Sniff access																								
2	Physical page number																								
3	Doubleword on module																								
4	Syndrome bits																								
140	SCP error logging enabled																								
141	SCP error logging disabled																								
142	Processor halted																								
143	SCP BOOT command has been executed																								
144	Power down																								
145	Power up																								
146	Reserved																								
147	Reserved																								
150	Battery backup complete																								
153	Microsequencer parity error																								
154	System cache parity error																								
155	System cache to Bank Controller parity error																								
156	IOC parity error																								
157	Sbus time-out																								
160	Sbus parity error																								

## Exceptions

Status (octal)	Meaning
<b>Code Definition (octal)</b> 161-207      Reserved	
000001	SCP reset. The SCP is reset and must be reinitialized with the DOBS <i>ac</i> , SCP instruction and a command 4. (All previously enabled functions have been disabled, and the SCP-to-CPU interface block address has been lost.)
000002	SCP function request acknowledge. This acknowledgment indicates to the CPU that the SCP has completed a requested function.
000003	SCP requested function is in error. The SCP reports an unknown error with this code. The SCP issues this code if the required SCP/CPU interface block has not been defined, an undefined function request is made, or invalid data is passed to the SCP (through the CPU-to-SCP buffer).
000004	Data requested is in buffer. The SCP has placed the requested information into the SCP-to-CPU buffer. The sub-code values returned to word 0 of the buffer indicate the contents of the buffer:
<b>Sub-code                      Definition</b>	
001	Boot clock data
002	GMT offset data
177777	SCP is in diagnostic sequence.

## Registers, Flags, and Stacks

AC0-AC3      Can be individually specified as *ac*; otherwise unused.

Busy, Done flags Set to 0

Carry      Unchanged

*Overflow*      Unaffected

PC      PC + 1

PSR      Unchanged

Stack      Unchanged

## Related Instructions

**DIB *ac*, SCP**      The **DIB *ac*, SCP** and the **DIBC *ac*, SCP** instructions perform identical functions. The **DIBS *ac*, SCP** instruction is a no-op.

## Exceptions

Issuing the I/O Reset instruction (**IORST**) causes double-bit errors to be checked and corrected, but not reported to the CPU.

## Power Supply Controllers

Universal Power Supply Controller		Power Supply Controller	
Device Code	4 <sub>8</sub>	Device Code	4 <sub>8</sub>
Assembler Mnemonic	UPSC	Assembler Mnemonic	PSC
Priority Mask Bit	13	Priority Mask Bit	13

The ECLIPSE MV/9500 System supports the universal power supply controller (UPSC).

The power supply controllers for the ECLIPSE MV/Family systems contain an on-board microprocessor which perform functions such as: a power-up diagnostic self-test, monitoring the system power, and reporting failures, problems, and status information. These controllers are either the Power Supply Controller (PSC) or the Universal Power Supply Controller (UPSC) — refer to the machine-specific supplement for which type of controller your machine supports.

Both types of controllers provide powerup and powerdown sequencing, the transfer to battery operation, I/O operations with ECLIPSE MV/Family systems, and output voltage margining. A major difference between controllers is the communications pathway — the CPU communicates directly with the UPSC; the CPU-to-PSC communication generally is through an intermediate processor (usually the diagnostic remote processor), though this is transparent to your program.

Both controllers use the same device code (4<sub>8</sub>) and the same priority mask bit (13). In multiple-I/O channel configurations, the CPU uses device code 4 on the primary I/O controller (IOC0) as the gateway to their power supply controller.

In addition, the controllers monitor the following:

- problems with the power supplies (such as overtemperature and overcurrent conditions);
- ac overvoltages and undervoltages (PSC only);
- overloads on the output voltages;
- state of the power switch;
- battery backup faults;
- fan or blower failures.

### Device Flag Control

Device flag commands to the power supply controllers determine the enabling or disabling of controller interrupts.

<i>f</i> =omitted	Busy and Done flags unchanged.
<i>f</i> =S	Sets the Busy flag to 1 and the Done flag to 0.
<i>f</i> =C	Sets the Busy and Done flags to 0.
<i>f</i> =P	Sets the Busy flag to 1 and the Done flag to 0.

**Table 8–20** *ECLIPSE MV/Family instructions with multiple-processor functions*

Mnemonic	Description
<b>IORST</b>	I/O Reset. In addition to its normal functions, this instruction also clears all pending cross interrupts, and redirects all IOC traffic to the processor that issued the IORST instruction.
<b>HALT</b>	Halt. Halts the processor and notifies the SCP that it is stopped.
<b>SSPT</b>	Store State Pointer. Regardless of the number of processors, there is only one state pointer per system.  It is recommended that the state area not be moved, as the system maintains the clocks (RTC and PIT) and multiple-processor synchronization within the state area. If the state area is moved, multiple-processor operations will be disrupted; cross interrupts and any pending messages ( <b>JPSTOP</b> , <b>JPFLUSH</b> , <b>JPSTATUS</b> ) will be lost. The RTC and PIT values will also become invalid. Since the current mask of IOC0 is stored in this area, IOC0 will have to be re-masked out ( <b>MSKO</b> ).
<b>PRTSEL</b>	I/O Channel Select. Changes the default I/O channel on ALL processors.

## Error Codes

Conditions that produce errors during execution of multiple-processor instructions return a value to AC1. Table 8–21 lists these errors.

**Table 8–21** *Error values returned to AC1*

Value	Instruction	Description
1	<b>JPLCS</b> , <b>JPSTART</b>	Not stopped. The processor to receive the request is currently running.
2	<b>CINTR</b> , <b>JPLCS</b> , <b>JPSTART</b> <b>JPSTATUS</b> , <b>JPSTOP</b>	Nonexistent processor. The processor ID specifies a value for a processor that does not exist.
3	<b>JPLCS</b>	LCS error.
4	<b>CINTR</b> , <b>JPLCS</b> , <b>JPSTART</b> <b>JPSTATUS</b> , <b>JPSTOP</b>	Processor failure. The processor to receive the request is not working.
5	<b>JPFLUSH</b>	No JP state block.
6	<b>CINTR</b>	Illegal option.
7	<b>JPSTOP</b>	Processor not running.

End of Chapter





## I/O Channel Definition Register Format

The I/O channel definition register (6000<sub>8</sub>) provides status information. The format for this register is diagrammed below and described in Table A-9.

ICE	Reserved		BVE	DVE	DCH	BMC	BAP	BDP	DIS	I/O Channel			DME	1
0	1	2	3	4	5	6	7	8	9	10		13	14	15

**Table A-9** I/O channel definition register contents

Bits	Name	Contents or Function
0	ICE *	I/O channel error flag. 1 Error occurred on I/O channel . 0 Only when all other error bits are 0.
1, 2	Reserved	Reserved for future use and returned as zero.
3	BVE *†	BMC validity error flag; if 1, BMC address validity protect error has occurred.
4	DVE *†	DCH validity error flag; if 1, DCH address validity protect error has occurred.
5	DCH	DCH transfer flag; if 1, a DCH transaction is in progress (read-only bit).
6	BMC	BMC transfer flag; if 1, a BMC transfer is in progress (read-only bit).
7	BAP *†	BMC address error; if 1, the channel has detected an address parity error.
8	BDP *†	BMC data error; if 1, the channel has detected a data parity error.
9	DIS *	Disable block transfer; if 1, disables BMC block transfers to and from I/O memory port.
10-13	I/O channel	I/O channel number.
14	DME *	DCH mode; if 1, DCH mapping is enabled.
15	1	Always set to 1.

\* The **IORST** instruction clears these bits.

† Writing to these bits with a 1 complements them.

## CPU Identification

The three Load CPU Identification instructions — **LCPID**, **ECLID**, and **NCLID** — return processor information to specified accumulators. The accumulator formats and descriptions follow.

### LCPID and ECLID Instructions

The **LCPID** and **ECLID** instructions load a doubleword into AC0.

Model Number			
0			15
Microcode Revision		Memory Size	
16	23	24	31

Bits	Name	Contents or Function
0-15	Model Number	Binary value of processor's allocated model number.
16-23	Microcode Revision	Current microcode revision.
24-31	Memory Size	Amount of physical memory available (measured in increments of 256-Kbyte modules with an origin of 0). Note that the actual memory size in bytes is equal to: $(AC0(24-31) + 1) * 262144_{10}$ . For example, 3 <sub>8</sub> indicates 1 Mbyte; 7 <sub>8</sub> indicates 2 Mbytes. NOTE: Systems which contain 64 Mbytes or more of physical memory return 37 <sub>8</sub> to bits 24-31 of AC0.

**NCLID Instruction**

The **NCLID** instruction loads its result into the low-order 16 bits of accumulators AC0 through AC2. Bits 0 through 15 of each accumulator are undefined.

**AC0**

Model Number	
16	31

Bits	Name	Contents or Function
16–31	Model Number	Binary value of processor's allocated model number.

**AC1**

1	Reserved		Microcode Revision	
16	17	23	24	31

If AC1 contains  $177777_8$ , load microcode.

Bits	Name	Contents or Function
16	1	Always set to 1.
17–23	Reserved	Reserved for future use and returned as zeros.
24–31	Microcode Revision	Current microcode revision.

**AC2**

Memory Size	
16	31

Bits	Name	Contents or Function
16–31	Memory Size	Amount of physical memory available measured in 32-Kbyte modules with an origin of 0). Note that the actual memory size in bytes is equal to: $(AC0(16-31) + 1) * 32768_{10}$ . For example, $32_8$ indicates 1 Mbyte; $64_8$ indicates 2 Mbytes. NOTE: Systems which contain 2 Gbytes or more of physical memory return $177777_8$ to bits 16–31 of AC0.

## ECLIPSE MV/9500 Supplement

This section supplements the “Register Fields” appendix by providing machine-specific information on ECLIPSE MV/9500 computer systems. ECLIPSE MV/9500 system supports all bits previously described and diagrammed in the “Register Fields” appendix with the exception of those discussed in this supplement.

### DCH/BMC Status Registers

The ECLIPSE MV/9500 system differs in the implementation of some bits in the following registers:

I/O Channel Definition Register — bit 9 is reserved and returned as 0.

I/O Channel Mask Register — bits 8 through 10 (C0, C1, C2) are the only I/O channel bits defined.

### CPU Identification

LCPID and ECLID instructions are formatted as described earlier. Listed below are the bits implemented on the ECLIPSE MV/9500 system.

Bits	Contents		
0-15	System model number	Model #	
		(octal)	(hexadecimal)
	ECLIPSE MV/9500	020774	21FC
16-23	Microcode revision		
24-31	Memory size (physical memory available in increments of 256 Kbytes — 0 = 256 Kbytes, 177 <sub>8</sub> = 32 Mbytes, 255 <sub>8</sub> = 64 Mbytes)		

The NCLID instruction loads AC1 and AC2 as formatted and described earlier. The ECLIPSE MV/9500 system implements the following bits in AC0.

Bits	Contents		
0-15	Undefined		
16-31	System model number	Model #	
		(octal)	(hexadecimal)
	ECLIPSE MV/9500	020774	21FC

End of Appendix



# E

## Standard I/O Device Codes

This appendix lists the standard input/output devices with their codes, mnemonics, and mask bits as supported by Data General's Macroassembler (MASM). In Table E-1, codes listed as "available for use" may be assigned to any additional devices. For compatibility with other ECLIPSE MV/Family systems, codes indicated as "reserved" should not be assigned to devices.

**Table E-1** *Standard I/O device codes*

Octal Code	Device Mnemonic	Priority Mask Bit	Description
00	—	—	Reserved
01	—	—	Available for use
02	—	—	Available for use
03	—	—	Reserved
04	PSC	13	Power supply controller *
05	—	—	Reserved
06	MCAT	12	Multiprocessor adapter transmitter
07	MCAR	12	Multiprocessor adapter receiver
10	TTI	14	TTY (primary asynchronous line) input *
11	TTO	15	TTY (primary asynchronous line) output *
12	—	—	Reserved
13	—	—	Reserved
14	RTC	13	Real-time clock *
15	—	—	Reserved
16	—	—	Reserved
17	LPT	12	Line printer
20	—	—	Reserved
21	—	—	Reserved
22	MTB	10	Magnetic tape
23	MTJ	10	Magnetic tape
24	DPJ	—	Disk subsystem (host interface protocol)
25	—	—	Reserved
26	DKB	9	Fixed-head DG/disk
27	DPF	7	DG/disk storage subsystem
30	IAC13	11	Intelligent asynchronous controller (IAC) 13
31	IAC14	11	IAC14
32	IAC15	11	IAC15
33	DKP	7	Moving-head disk
34	ISC	4	Intelligent synchronous controller
35	—	—	Reserved
36	—	—	Reserved
37	—	—	Reserved

\* These devices are supported on the primary IOC only. The device codes are reserved on any additional IOCs.

(continued)

**Table E-1** *Standard I/O device codes (concluded)*

Octal Code	Device Mnemonic	Priority Mask Bit	Description
40	DCU	4	Data control unit
41	DCU1	4	Second data control unit
42	—	—	Reserved
43	PIT	11	Programmable interval timer *
44	—	—	Reserved
45	SCP	15	Diagnostic remote processor *
46	MCAT1	12	Second multiprocessor transmitter
47	MCAR1	12	Second multiprocessor receiver
50	IAC1	11	IAC1
51	IAC2	11	IAC2
52	IAC3	11	IAC3
53	IAC4	11	IAC4
54	IAC5	11	IAC5
55	IAC6	11	IAC6
56	IAC7	11	IAC7
57	LPT1	12	Second line printer
60	—	—	Reserved
61	—	—	Reserved
62	MTB1	10	Second magnetic tape
63	MTJ1	10	Second magnetic tape
64	DPJ1	—	Disk subsystem (host interface protocol)
65	IAC	11/5	Host-to-IAC interface
66	DKB1	9	Second fixed-head DG/disk
67	DPF1	7	Second DG/disk storage subsystem
70	IAC8	11	IAC8
71	IAC9	11	IAC9
72	IAC10	11	IAC10
73	IAC11	11	IAC11
74	IAC12	11	IAC12
75	—	—	Reserved
76	—	—	Reserved
77	CPU	—	CPU and console functions *

\* These devices are supported on the primary IOC only. The device codes are reserved on any additional IOCs.

End of Appendix

---

## Context Block Formats

This appendix describes the formats of the three types of context blocks in the ECLIPSE MV/9500 computer: short, long, and emulation. The type is indicated in bits 0 and 31 of doubleword 12–13 of the short context block.

- The short context block uses words 0 through 17. This is the basic block saved on a fault during execution of a restartable instruction.
- The long context block uses words 0 through 17 plus an extension provide by words 18 through 49. This is the basic block saved on a fault during execution of a resumable instruction.
- The emulation context block uses words 0 through 17 plus an extension provide by words 18 through 59. This is the basic block saved on a fault during execution of an emulated instruction.

The context block contains information used by the microcode and other internal systems.

NOTE: *The context block does not contain the complete floating-point state. To save this information, use a Push Floating-Point State instruction.*

*The processor requires that the context block save area, the pointer to the save area, and all indirect chains accessed align on doubleword boundaries. If the first word of a long context block is not quad-aligned, an additional double-word is written after the short format to ensure that the subsequent words comprising the extended formats are quad-aligned.*

Table F-1 shows the format of the context blocks.

**Table F-1** *Context block format*

Word in Block	Contents or Function												
<b>Short Format</b>													
0	PSR												
1	0												
2-3	AC0												
4-5	AC1												
6-7	AC2												
8-9	AC3												
10-11	CARRY, PC of offending (executing) instruction.												
12-13	Current ring of execution/Context block type. Context block type defined as follows:												
	<table><tr><th>Bit 0</th><th>Bit 31</th><th>Type</th></tr><tr><td>0</td><td>0</td><td>Short format only</td></tr><tr><td>0</td><td>1</td><td>Long format</td></tr><tr><td>1</td><td>0</td><td>Emulation format</td></tr></table>	Bit 0	Bit 31	Type	0	0	Short format only	0	1	Long format	1	0	Emulation format
Bit 0	Bit 31	Type											
0	0	Short format only											
0	1	Long format											
1	0	Emulation format											
14-15	LAR, address that caused page fault.												
16-17	Current execution register/Valid flag												
 <b>Extensions for Long Context Block*</b>													
18-19	General Register 0												
20-21	General Register 1												
22-23	General Register 2												
24-25	General Register 3												
26-27	Current LAR												
28-29	DBUS Register												
30-31	General Register A												
32-33	General Register B												
34-35	Microstack level 0												
36-37	Microstack level 1												
38-39	Microstack level 2												
40-41	Microstack level 3												
42-43	Pipe Flags												
44-45	Current ring of execution/ALU State												
46-47	ATU State/W/L												
48-49	Next PC												
 <b>Extensions for Emulation Context Block*</b>													
18-19	AC1												
20-21	ION, ATU on												
22-23	Current execution register/Valid flag												
24-25	Wide Stack Pointer												
26-27	PSR:0000												
28-29	AC0												
30-31	Pagefault Code												
32-33	AC2												
34-35	AC3												
36-37	CARRY, PC												
38-39	CARRY, Next PC												
40-41	Scratch memory register 0												
42-43	Scratch memory register 1												
44-45	Scratch memory register 2												
46-47	Scratch memory register 3												
48-49	Scratch memory register 4												
50-51	Scratch memory register 5												
52-53	Scratch memory register 6												
54-55	Scratch memory register 7												
56-57	Scratch memory register 8												
58-59	Scratch memory register 9												

\* See NOTE on previous page.

End of Appendix



---

## Instruction Execution Times

This appendix discusses instruction execution times for the ECLIPSE MV/9500 computers. Tables list execution times for each instruction along with the variables that must be taken into account in calculating execution times.

NOTE: *If an instruction is not listed in this appendix, the instruction is not supported by the processor.*

The discussion in this appendix assumes the following:

- All logical-to-physical address translations are resident in the address translation unit.
- No indirection is used in any instruction.
- The page containing the executing instruction does not contain any data. (This would force a flush of all instruction pipelines)
- All data referenced is resident in the microMV internal cache.
- The instruction pipeline is always ready.
- Cycle time is 67 ns.
- Values reflect microcode version 9.70, patch revision 14.
- For all of the floating-point instructions, no normalization or prescaling is performed.
- Where additional time is specified, the addition is cumulative.

The emulated instructions are not directly supported by microMV microcode. Instead they are performed by a combination of microcode and macrocode.

Table G-1 lists the adjustments that must be made for memory references. Table G-2 lists the adjustments required for fault conditions. Table G-3 lists the I/O instructions. Table G-4 lists the emulated instructions. Table G-5 lists the instructions for microMV based instructions (chip-resident microcode) and also shows the adjustments that must be made for situational factors.

**Table G-1** *Adjustments for memory references*

To Every Memory Reference	Add ( $\mu$ s)
If logical-to-physical address translation is not in the address translation unit (ATU) cache,	
1-level page table	0.67
2-level page table	1.072
If indirection is specified	
1-level, 16-bit instruction	0.268
2 or more levels, 16-bit instructions	$0.268 + 0.134 * \text{levels}$
All levels, 32-bit instructions	$1.072 + 0.201 * \text{levels}$
Cache block crossing	
Read	0.134
Write	0.201
Page crossing operation	
Read	0.402
Write	0.469
Data not resident in internal cache, but resident in system cache	0.134
Data not resident in internal cache and not resident in system cache	0.469

**Table G-2** *Adjustments for fault conditions*

To Any Instruction	Add ( $\mu$ s)
If any of the following faults occur	
Stack overflow/underflow	
Wide stack, type 0, 2, 4	$2.211 + \text{any indirection}$
Wide stack, type 1	$2.613 + \text{any indirection}$
Wide stack, type 3	$2.402 + \text{any indirection}$
Narrow stack	$1.139 + \text{any indirection}$
Fixed point fault	$2.278 + \text{any stack/indirection}$
Protection fault, typical	$3.618 + \text{any stack/indirection}$

**Table G-3** *Execution times for I/O instructions*

Mnemonic	Timing ( $\mu$ s)*
WLMP	$0.335 + 0.268/\text{slot}$ (minimum instruction)
XVCT	4.280 (base level, in-line)
	3.35 (intermediate level, inline)

\* I/O instruction times are minimum average times. These instructions vary considerably in the amount of time they may take to execute (such as waiting for bus access or being interrupted). Though all ECLIPSE MV/Family I/O instructions are supported on ECLIPSE MV/9500 series systems, we list minimum execution times for only two I/O instructions, WLMP and XVCT.

Table G-4 Execution times for emulated instructions

Mnemonic	Timing (μs)	Mnemonic	Timing (μs)
EDIT	26.934 + sum of sub-op execution times	WEDIT	24.120 + sum of sub-op execution times
DADI	5.762	DADI	6.298
DAPS	4.221 (w/o add) 5.561 (with add)	DAPS	4.489 (w/o add) 5.762 (with add)
DAPT	4.221 (w/o add) 5.561 (with add)	DAPT	4.489 (w/o add) 5.762 (with add)
DAPU	5.226	DAPU	5.427
DASI	6.767 (type 4) 8.040 (type 5)	DASI	7.236 (type 4) 8.576 (type 5)
DDTK	10.251 (branch taken) 8.375 (branch not taken)	DDTK	10.251 (branch taken) 8.442 (branch not taken)
DEND	7.035	DEND	6.700
DICI	5.360 + 3.417/character inserted	DICI	5.561 + 3.287/character inserted
DIMC	6.968 + 1.608/character inserted + 1.608 if j in stack	DIMC	7.102 + 1.548/character inserted + 1.348 if j in stack
DINC	6.365	DINC	6.432
DINS	6.700 (S=0) 7.169 (S=1)	DINS	6.767 (S=0) 7.169 (S=1)
DINT	6.700 (T=0) 7.169 (T=1)	DINT	6.767 (T=0) 7.169 (T=1)
DMVA	7.571 + 4.690/character moved + 1.608 if j in stack	DMVA	7.705 + 4.550/character moved + 1.348 if j in stack
DMVC	7.169 + 3.752/character moved + 1.608 if j in stack	DMVC	7.303 + 3.612/character moved + 1.348 if j in stack
DMVF	5.226 + 10.787/digit moved + 1.608 if j in stack	DMVF	5.494 + 10.587/digit moved + 1.308 if j in stack
DMVN	5.695 + 8.576/digit moved + 1.608 if j in stack	DMVN	5.494 + 8.446/digit moved + 1.348 if j in stack
DMVO	12.797/digit zero 13.400/digit non zero	DMVO	12.864/digit zero 13.467/digit non zero
DMVS	5.226 + 9.648/digit moved + 1.608 if j in stack	DMVS	5.494 + 9.448/digit moved + 1.348 if j in stack
DNDF	4.221 (T=1) 7.303 (T=0)	DNDF	4.556 (T=1) 7.370 (T=0)
DSSO	3.551	DSSO	3.484
DSSZ	3.551	DSSZ	3.484
DSTK	8.174	DSTK	8.040
DSTO	3.551	DSTO	3.484
DSTZ	3.551	DSTZ	3.484
LDI	5.866 (type 4, length=7)	WLDI	5.865 (type 4, length=7)
LDIX	1243.9 (type 4, length=31)	WLDIX	1260.274 (type 4, length=31)
LSN	26.106 (type 4, length=7)	WLSN	27.302 (type 4, length=7)
STI	7.567 (type 4, length=7)	WSTI	7.566 (type 4, length=7)
STIX	303.344 (type 4, length=31)	WSTIX	295.738 (type 4, length=31)
WDCMP	112.138 (type 4, length=7)		
WDDEC	33.603 min (type 4, length=7) 112.138 max (type 4, length=7)		
WDINC	33.271 min (type 4, length=7) 49.490 max (type 4, length=7)		
WDMOV	67.268 (type 4, length=7)		

Table G-5 Execution times for microMV based instructions

Mnemonic	Timing (μs)	Mnemonic	Timing (μs)
ADC	0.201 + 0.067 if Skip	CMV (continued)	
ADC#	0.134 + 0.067 if Skip		1.005 + 0.402/byte (destination descending or length(destination) < 16) + 0.067 if AC1 negative
ADCNS	0.067		1.072 + 0.402/byte (source descending)
ADD	0.201 + 0.067 if Skip		0.871 + 0.402/byte (length(source) < length(destination))
ADD#	0.134 + 0.067 if Skip		+ 0.134 + 0.201/byte, if padding with blanks.
ADDNS	0.067		both ascending, length(both) > 16, length(source) >= length(destination), word aligned
ADDI	0.067		2.211 + 0.017/byte + 0.402 * n (one word apart)
ADI	0.067		n = number of bytes MOD 4
ANC	0.067		2.278 + 0.017/byte + 0.402 * n (two words apart)
AND	0.201 + 0.067 if Skip		1.675 + 0.402/byte (three words apart)
AND#	0.134 + 0.067 if Skip		2.010 + 0.042/byte + 0.402 * n (four or more words)
ANDNS	0.067	COB	1.809 (typical)
ANDI	0.067		0.134 (all zeros)
BAM	0.201 (out of range)	COM	0.201 + 0.067 if Skip
	0.603 + 0.335/word (number of words < 4)	COM#	0.134 + 0.067 if Skip
	0.737 + 0.335/word (number of words >= 4)	COMNS	0.067
BKPT	4.355 (non-XCTed)	CRYTC	0.067
	11.524 (XCTed)	CRYTO	0.067
BLM	0.134 (out of range)	CRYTZ	0.067
	0.536 + 0.268/word (# of words < 4)	CTR	0.804 + 0.469/byte (Xlate-move)
	0.737 + 0.268/word (# of words >= 4 and overlap of 2 or 3 words)		0.804 + 0.670/byte (Xlate-compare)
	1.005 + 0.034 * n + (0.067 + 0.268 * m) (# words >= 4 and source + 1 = destination)	CVWN	0.201
	n = # of words DIV 4	DAD	0.335
	m = # of words MOD 4	DEQUE	3.484 (deque the Head)
	0.737 + 0.084 * n + (0.067 + 0.268 * m) + 0.335/word of alignment required (# words >= 4)		+ 0.067 if mid element
BTO	0.603 (ACS=ACD) (1) + 0.067 if ACS<>ACD		+ 0.201 if tail of queue
BTZ	0.603 (ACS=ACD) (1) + 0.067 if ACS<>ACD		2.278 (empty queue)
CLM	0.536 (ACS<>ACD, ACS < LOW) + 0.067 if ACS > HIGH + 0.067 if skip + 0.067 if ACS=ACD		3.350 (one element queue)
CMP	0.737 + 0.469/byte, length(string 1) <= length(string 2)	DERR	0.536
	0.804 + 0.469 * n + 0.268 * m, length(string 1) > length(string 2)	DHXL	0.536
	n = number of bytes in string 2		0.268 (ACD & ACD + 1 = 0)
	m = 4 * (length(string 1) - length(string 2))		0.469 (ACD + 1 = 0)
	+0.067 if opposite signs	DHXR	0.536
	0.737 + 0.268/byte (string 1 blank)		0.268 (ACD & ACD + 1 = 0)
	0.871 + 0.268/byte (string 1 blank)	DIV	2.747
CMT	0.871 + 0.469/byte (AC2=AC3)	DIV	0.402 (dividend = 0)
	0.938 + 0.536/byte (AC2<>AC3)	DIVS	2.747
CMV	0.536 (destination length = 0)	DIVS	0.469 (dividend = 0)
	0.871 + 0.201/byte (source length = 0) + 0.067 if AC1 negative	DIVX	1.608
		DIVX	0.335 (dividend = 0)
		DLSH	0.536 (shift right or count = 0) +0.067 if shift left
			0.335 (ACD & ACD + 1 = 0)
		DSB	0.335

(1) If instruction is followed by an instruction which will perform a memory operation, the second instruction must add 0.067 us to its execution time.

(continued)

Table G-5 Execution times for microMV based instructions

Mnemonic	Timing (μs)	Mnemonic	Timing (μs)
DSPA	0.670	FMD	0.402 (FPACD=0)
	0.268 (ACD < LOWER)		0.335 (FPACS=0)
	0.402 (ACD > UPPER)		2.178 (average) + 0.234 (rounding)
	0.603 value = -1	FMMD	0.469 (FPACD=0)
DSZ	0.201 + 0.134 if Skip		0.402 (FPACS=0)
DSZTS	0.201 + 0.134 if Skip		2.245 (average) + 0.234 (rounding)
ECLID	2.144	FMMS	0.469 (FPACD=0)
EDSZ	0.201 + 0.134 if Skip		0.402 (FPACS=0)
EISZ	0.201 + 0.134 if Skip		0.838 (average) + 0.067 (rounding)
EJMP	0.201	FMOV	0.067
EJSR	0.201	FMS	0.402 (FPACD=0)
ELDA	0.067		0.335 (FPACS=0)
ELDB	0.067		0.771 (average) + 0.067 (rounding)
ELEF	0.067	FNEG	0.268
ENQH	3.484 + 0.201 if new element is Head/Tail	FNOM	0.335 + 2 * (number of digits)
ENQT	3.685 + 0.201 if new element is Head/Tail	FNS	0.134
ESTA	0.067 (1)	FPOP	1.407
ESTB	0.067 (1)	FPSH	1.206
FAB	0.201 + 0.067 if FPACD is negative	FRDS	0.201 (truncation)
FAD	0.335 + 0.067 if opposite signs		0.335 (rounding)
FAMD	0.335 + 0.067 if opposite signs	FRH	0.201
FAMS	0.335 + 0.067 if opposite signs	FSA	0.134 + 0.067 if Skip
FAS	0.268 + 0.067 if opposite signs	FSCAL	2.278
FCLE	0.268	FSEQ	0.134 + 0.067 if Skip
FCMP	0.536 (opposite signs)	FSGE	0.134 + 0.067 if Skip
	0.938 (SIGNs, EXPs of both are equal and abs FPACD  > abs FPACS )	FSGT	0.134 + 0.067 if Skip
	1.005 (SIGNs, EXPs of both are equal and abs FPACD  < abs FPACS )	FSLE	0.134 + 0.067 if Skip
	1.005 (SIGNs, EXPs, MANTs of both are equal)	FSLT	0.134 + 0.067 if Skip
FDD	0.436 (FPACD=0)	FSMD	0.536 + 0.067 if opposite signs
	8.576 (average) + 0.804 (rounding)	FSMS	0.536 + 0.067 if opposite signs
FDMD	0.503 (FPACD=0)	FSND	0.201 + 0.067 if Skip
	8.643 (average) + 0.804 (rounding)	FSNE	0.134 + 0.067 if Skip
FDMS	0.536 (FPACD=0)	FSNER	0.201 + 0.067 if Skip
	2.412 (average) + 0.469 (rounding)	FSNM	0.201 + 0.067 if Skip
FDS	0.469 (FPACD=0)	FSNO	0.201 + 0.067 if Skip
	2.345 (average) + 0.469 (rounding)	FSNOD	0.268 + 0.067 if Skip
FEXP	0.402 (average)		0.201 (OVF = 1)
	0.268 (FPACD=0)	FSNU	0.201 + 0.067 if Skip
FFAS	0.871	FSNUD	0.268 + 0.067 if Skip
FFMD	0.804		0.201 (UNF = 1)
FHLV	0.433 (average)	FSNUO	0.268 + 0.067 if Skip
FINT	0.268 (exponent <= 40)		0.201 (UNF = 1)
	0.469 + 0.067 * N (40 < exponent < 48)	FSS	0.335 (opposite signs)
	n = exponent - 40		+ 0.067 if same signs and FPACD <= FPACS
	0.536 + 0.067 * N (exponent >= 48)	FSST	0.201
FLAS	0.603 + 0.067 if negative	FSTD	3.216
FLDD	0.067	FSTS	3.149
FLDS	0.067	FTD	0.268
FLMD	0.536 + 0.067 if negative	FTE	0.268
FLST	0.268	FXTD	0.067
		FXTE	0.134
		HLV	0.201

(1) If instruction is followed by an instruction which will perform a memory operation, the second instruction must add 0.067 us to its execution time.

(continued)

Table G-5 Execution times for microMV based instructions

Mnemonic	Timing ( $\mu$ s)	Mnemonic	Timing ( $\mu$ s)
HXL	0.201	LLEFB	0.134
HXR	0.268	LMRF	1.273
	0.134 (ACD=0)	LNADD	0.134
INC	0.201 + 0.067 if Skip	LNADI	0.201 (1)
INC#	0.134 + 0.067 if Skip	LNDIV	1.541 (average)
INCNS	0.067		0.402 (dividend = 0)
IOR	0.067	LNDO	0.268 (for termination)
IORI	0.067		0.603 (no termination)
ISZ	0.201 + 0.134 if Skip	LNDZ	0.201 + 0.134 if Skip
ISZTS	0.201 + 0.134 if Skip	LNISZ	0.201 + 0.134 if Skip
JM	0.201	LNLDA	0.067
JSR	0.201	LNMUL	0.536
LCALL	0.201 + indirection, Intra-ring (same ring)	LNMUL	0.201 (destination or source = 0)
	2.211 + indirection, Gated Intra-ring (same ring)	LNSBI	0.201 (1)
	3.149 + indirection + 0.268/argument	LNSTA	0.067 (1)
	Inter-ring	LNSUB	0.134
LCPID	2.144	LOB	0.335 min
LDA	0.067		1.340 max
LDAF	0.067	LPEF	0.134
LDASB	0.067	LPEFB	0.201
LDASL	0.067	LPHY	1.474 (one level)
LDASP	0.067		2.278 (two level)
LDATS	0.067	LPSHJ	0.201
LDB	0.335 (1)	LPSR	0.067
LDSP	0.737	LPTE	1.340 (one level)
	0.268 (ACD < Lower)		2.077 (two level)
	0.335 (ACD > Upper)	LRB	0.536 min (ACS=ACD)
	0.603 value = -1		1.541 max (ACS=ACD)
LEF	0.067		+ 0.067 if ACS > ACD
LFAMD	0.402 + 0.067 if opposite signs	LSBRA	13.936
LFAMS	0.402 + 0.067 if opposite signs	LSBRS	13.601
LFDMD	0.503 (FPACD=0)	LSH	0.268
	8.643 (average) + 0.804 (rounding)		0.201 (ACD=0)
LFDMS	0.536 (FPACD=0)	LSTB	0.067 (1)
	2.412 (average) + 0.469 (rounding)	LWADD	0.134
LFLDD	0.067	LWADI	0.201 (1)
LFLDS	0.067	LWDIV	2.546 (average)
LFLST	0.201		0.335 (Dividend = 0)
LFMMD	0.469 (FPACD=0)	LWDO	0.268 (for termination)
	0.402 (FPACS=0)		0.603 (no termination)
	2.245 (average) + 0.234 (rounding)	LWDSZ	0.201 + 0.134 if Skip
LFMMS	0.469 (FPACD=0)	LWISZ	0.201 + 0.134 if Skip
	0.402 (FPACS=0)	LWLDA	0.067
	0.838 (average) + 0.067 (rounding)	LWMUL	0.603
LFSMD	0.536 + 0.067 if opposite signs		0.201 (destination or source = 0)
LFSST	0.201	LWSBI	0.201 (1)
LFSTD	0.067	LWSTA	0.067 (1)
LFSTS	0.067	LWSUB	0.134
LJMP	0.201	MOV	0.201 + 0.067 if Skip
LJSR	0.201	MOV#	0.134 + 0.067 if Skip
LLDB	0.067	MOVNS	0.067
LLEF	0.067	MSP	0.335

(1) If instruction is followed by an instruction which will perform a memory operation, the second instruction must add 0.067  $\mu$ s to its execution time.

(continued)

Table G-5 Execution times for microMV based instructions

Mnemonic	Timing ( $\mu$ s)	Mnemonic	Timing ( $\mu$ s)
MUL	0.603	SMRF	1.809
	0.536 (AC0=0)	SNB	0.603 + 0.067 if Skip (ACS=ACD) + 0.067 if ACS $\nabla$ ACD
	0.268 (AC1=0)	SNOVR	0.134 + 0.067 if Skip
	0.335 (AC2=0)	SPSR	0.067
MULS	0.603	SPTE	2.814
	0.536 (AC0=0)	SSPT	0.134
	0.268 (AC1=0)	STA	0.067 (1)
	0.335 (AC2=0)	STAFP	0.067
NADD	0.067	STASB	0.335
NADDI	0.067	STASL	0.134
NADI	0.067	STASP	0.067
NBStc	1.407 + 0.737 * (n-1) match found n= number of elements before match 1.139 + 0.737 * (n-1) No match n= number of elements in queue	STATS	0.067 (1)
NCLID	2.010	STB	0.335 (1)
NDIV	1.608	SUB	0.201 + 0.067 if Skip
	0.469 (dividend = 0)	SUB#	0.134 + 0.067 if Skip
NEG	0.201 + 0.067 if Skip	SUBNS	0.067
NEG#	0.134 + 0.067 if Skip	SZB	0.603 + 0.067 if Skip (ACS=ACD) +0.067 if ACS $\nabla$ ACD
NEGNS	0.067	SZBO	0.737 (ACS=ACD) + 0.067 if ACS $\nabla$ ACD + 0.335 if Skip fault or Impis
NFStc	1.407 + 0.737*(n-1) match found n= number of elements before match 1.139 + 0.737*(n-1) No match n= number of elements in queue	VBP	0.402 0.335 (ring < AC1 ring) 0.268 (ring < CRE)
NLDAI	0.067	VWP	0.402 0.335 (ring < AC1 ring) 0.268 (ring < CRE) + 5.695 + 0.268/level of indirection
NMUL	0.603	WADC	0.067
	0.201 (destination = 0)	WADD	0.067
	0.268 (source = 0)	WADDI	0.067
NNEG	0.067	WADI	0.067
NSALA	0.201 + 0.067 if Skip	WANC	0.067
NSALM	0.335 + 0.067 if Skip	WAND	0.067
NSANA	0.134 + 0.067 if Skip	WANDI	0.067
NSANM	0.335 + 0.067 if Skip 0.134 (immediat = 0)	WASH	0.335 (shift right or count = 0) + 0.067 if shift left
NSBI	0.067	WASHI	0.335 (shift right or count = 0) + 0.067 if shift left
NSUB	0.067	WBLM	0.402 + 0.268/word (descending) 0.469 + 0.268/word (ascending) (# of words < 4) 0.670 + 0.268/word (# of words >= 4 and overlap of 2 or 3 words) 0.938 + 0.034 * n + (0.067 + 0.268 * m) (# words >= 4 and source + 1 = destination) n = # of words DIV 4 m = # of words MOD 4 0.670 + 0.084 * n + (0.067 + 0.268 * m) (# words >= 4) + 0.335/word of alignment required
PATU	0.402		
PBX	25.728		
POP	0.335 + 0.067/word		
POPB	0.603		
POPJ	0.536		
PSH	0.402 + 0.067/word		
PSHJ	0.469		
PSHR	0.402		
RSTR	0.670		
RTN	0.670		
RTOD	0.737		
SAVE	1.005		
SAVZ	1.005		
SBI	0.067		
SEX	0.067		
SGE	0.134 + 0.067 if Skip		
SGT	0.134 + 0.067 if Skip		

(1) If instruction is followed by an instruction which will perform a memory operation, the second instruction must add 0.067 us to its execution time.

(continued)

Table G-5 Execution times for microMV based instructions

Mnemonic	Timing (μs)	Mnemonic	Timing (μs)
WBP	0.402 0.335 (ring < AC1 ring) 0.268 (ring < CRE)	WCTR	0.804 + 0.670/byte (xlate - compare) 0.804 + 0.469/byte (xlate - move)
WBR	0.201	WDIV	2.546 (average)
WBStc	1.407 + 0.737 * (n-1) match found n= number of elements before match 1.139 + 0.737 * (n-1) no match n= number of elements in queue	WDIVS	2.747 (average)
WBTO	0.536 (ACS+ACD) (1) +0.067 if ACS<ACD	WDPOP	22.244 (same ring, short context block) + 0.804 if long context block +1.139 if different ring
WBTZ	0.536 (ACS=ACD) (1) + 0.067 if ACS<ACD	WFFAD	0.737
WCLM	0.536 (ACS<ACD, ACS < LOW) + 0.067 if ACS > HIGH + 0.067 if skip + 0.067 if ACS=ACD	WFLAD	0.536 +0.067 if negative 0.201 (zero)
WCMP	0.737 + 0.469/byte length(string 1) <= length(string 2) 0.804 + 0.469 * n + 0.268 * m length(string 1) > length(string 2) n = number of bytes in string 2 m = 4 * (length(string 1) - length(string 2)) + 0.067 if opposite signs 0.737 + 0.268/byte (string 1 blank) 0.871 + 0.268/byte (string 1 blank)	WFPOP	0.871
WCMT	0.871 + 0.469/byte (AC2=AC3) 0.938 + 0.536/byte (AC2<AC3)	WFPSH	0.938
WCMV	0.536 (destination length = 0) 0.871 + 0.201/byte (source length = 0) 1.005 + 0.402/byte (destination descending or len(destination) < 16) + 0.067 if AC1 negative 1.072 + 0.402/byte (source descending) 0.871 + 0.402/byte (len(source) < length(destination)) +0.134 + 0.201/byte if padding with blanks, both ascending, length(both) > 16, length(source) >= length(destination), word aligned 2.211 + 0.017/byte + 0.402 * n (one word apart) n= number of bytes MOD 4 2.278 + 0.017/byte + 0.402 * n (two words apart) 1.675 + 0.402/byte (three words apart) 2.010 + 0.042/byte + 0.402 * n (four or more words)	WFSStc	1.407 + 0.737 * (n-1) match found n= number of elements before match 1.139 + 0.737 * (n-1) no match n= number of elements in queue
WCOB	3.417 0.134 (all zeros)	WHLV	0.201
WCOM	0.067	WINC	0.067
WCST	0.536 + 0.469/byte + 0.067 if negative + 0.067 delimiter is last character in string	WIOR	0.067
		WIORI	0.067
		WLDAI	0.067
		WLDB	0.335 (1)
		WLOB	0.402 min 1.474 max
		WLRB	0.469 (ACS=ACD min) 2.546 (ACS=ACD max) +0.067 if ACS<ACD
		WLSH	0.201 (shift left) + 0.067 if shift right or count=0
		WLSHI	0.201 (shift left) + 0.067 if shift right or count=0
		WLSI	0.067
		WMESS	0.536 (successful) 0.469 (unsuccessful)
		WMOV	0.067
		WMOVR	0.067
		WMSP	0.335
		WMUL	0.536 0.134 (destination=0) 0.201 (source=0)
		WMULS	0.603 0.469 (AC0=0) 0.268 (AC1=0) 0.335 (AC2=0)
		WNADI	0.067
		WNEG	0.067
		WPOP	0.067 + 0.067/word
		WPOPb	0.536 +1.541 if different rings +0.067 if not aligned
		WPOPJ	0.335
		WPSH	0.134/word
		WRSTR	1.273 + 0.871 if different rings
		WRTN	0.536 + 1.551 if different rings + 0.067 if not aligned
		WSALA	0.201 + 0.067 if Skip

(1) If instruction is followed by an instruction which will perform a memory operation, the second instruction must add 0.067 us to its execution time.

(continued)



**Table G-5** Execution times for microMV based instructions

Mnemonic	Timing (μs)	Mnemonic	Timing (μs)
WSALM	0.335 + 0.067 if Skip	XFLDD	0.067
WSANA	0.134 + 0.067 if Skip	XFLDS	0.067
WSANM	0.335 + 0.067 if Skip	XFMMD	0.469 (FPACD=0)
	0.134 (immediate = 0)		0.402 (FPACS=0)
WSAVR	0.469 (non-aligned)		2.245 (average) +0.234 (rounding)
	0.536 (aligned)	XFMMS	0.469 (FPACD=0)
WSAVS	0.469 (aligned or non-aligned, OVK clear)		0.402 (FPACS=0)
	0.536 (non-aligned, OVK set)		0.838 (average) +0.070 (rounding)
WSBI	0.067	XFSMD	0.536 + 0.067 if opposite signs
WSEQ	0.134 + 0.067 if Skip	XFSMS	0.536 + 0.067 if opposite signs
WSEQI	0.134 + 0.067 if Skip	XFSTD	0.067
WSGE	0.134 + 0.067 if Skip	XFSTS	0.067
WSGT	0.134 + 0.067 if Skip	XJMP	0.201
WSGTI	0.134 + 0.067 if Skip	XJSR	0.201
WSKBO	0.134 + 0.067 if Skip	XLDB	0.067
WSKBZ	0.134 + 0.067 if Skip	XLEF	0.067
WSLE	0.134 + 0.067 if Skip	XLEFB	0.134
WSLEI	0.134 + 0.067 if Skip	XNADD	0.134
WSLT	0.134 + 0.067 if Skip	XNADI	0.201 (1)
WSNB	0.603 ACS>ACD	XNDIV	1.541 (average)
	0.536 ACS=ACD + 0.067 if Skip		0.402 (dividend = 0)
WSNE	0.134 + 0.067 if Skip	XNDO	0.268 (termination)
WSNEI	0.134 + 0.067 if Skip		0.603 (no termination)
WSSVR	0.536 (aligned)	XNDSZ	0.201 + 0.134 if Skip
	0.603 (non-aligned)	XNISZ	0.201 + 0.134 if Skip
WSSVS	0.536 (aligned)	XNLDA	0.067
	0.603 (non-aligned)	XNMUL	0.536
WSTB	0.335 (1)		0.201 (source or destination = 0)
WSUB	0.067	XNSBI	0.201 (1)
WSZB	0.603 ACS>ACD	XNSTA	0.067 (1)
	0.536 ACS=ACD + 0.067 if Skip	XNSUB	0.134
WSZBO	0.804 ACS>ACD	XOP0	1.273
	0.737 ACS=ACD + 0.335 Skip fault or IMPIS	XOR	0.067
WUGTI	0.134 + 0.067 if Skip	XORI	0.067
WULEI	0.134 + 0.067 if Skip	XPEF	0.134
WUSGE	0.134 + 0.067 if Skip	XPEFB	0.201
WUSGT	0.134 + 0.067 if Skip	XPSHJ	0.201
WXCH	0.201	XSTB	0.067 (1)
WXOP	1.407	XWADD	0.134
WXOR	0.067	XWADI	0.201 (1)
WXORI	0.067	XWDIV	2.546 (average)
XCALL	0.201 + indirection, Intra-ring (same ring)		0.335 (dividend = 0)
	2.211 + indirection, Gated Intra-ring (same ring)	XWDO	0.268 (termination)
	3.149 + indirection + 0.268/argument inter-ring		0.603 (no termination)
XCH	0.201	XWDSZ	0.201 + 0.134 if Skip
XCT	8.643	XWISZ	0.201 + 0.134 if Skip
XFAMD	0.402 + 0.067 if opposite signs	XWLDA	0.067
XFAMS	0.402 + 0.067 if opposite signs	XWMUL	0.603
XFDMD	0.503 (FPACD=0)		0.201 (destination or source = 0)
	8.643 (average) + 0.804 (rounding)	XWSBI	0.201 (1)
XFDMS	0.536 (FPACD=0)	XWSTA	0.067 (1)
	2.412 (average) + 0.469 (rounding)	XWSUB	0.134
		ZEX	0.067

(1) If instruction is followed by an instruction which will perform a memory operation, the second instruction must add 0.067 us to its execution time.

(concluded)

End of Appendix



---

# Index

Within the index, the page number refers to the first page for an entry (even if the subject spans multiple pages). Instruction mnemonics are printed in boldface type (such as **INTEN**); instruction names are printed with initial capital letters (such as I/O Interrupt Enable).

16-bit. *See* ECLIPSE 16-bit

## A

Absolute addressing 1-11

Access

- I/O 8-3
- memory 1-8
- operand 1-13
- page 9-9
  - validation 9-9
- segment 9-2

Accumulator instruction, execute 5-2

Accumulators

- fixed-point 1-2, 1-3
- floating-point 1-3, 1-4

Adding a data element to queue 6-4

Addition instructions

- fixed-point 2-4
- floating-point 3-7

Address

- absolute 1-11
- base register 1-12
- bit 1-17
  - ECLIPSE 16-bit 10-10, 10-11
- BMC modes 8-23
- byte 1-16
  - ECLIPSE 16-bit 10-9, 10-10
- effective 1-12, 1-13
  - ECLIPSE 16-bit 10-9
  - load, instructions 2-21
  - resolution 1-10
- generator 1-23
- indirect 1-12
- indirect field 1-12
- logical 1-8
  - format 9-6
  - space 1-7, 1-8
- modes 1-11
- relative 1-12
- translation 9-2, 9-6
  - unit 1-23
- word, ECLIPSE 16-bit 10-8, 10-9
- wraparound 5-1

**ALARM** 8-53

Alarm clock 8-50

- instructions 8-50

Alarm instruction, Set 8-53

Aligning floating-point mantissas 3-5

ALU. *See* Arithmetic logic unit

Appending floating-point guard digits 3-5

Architectural clocks 1-24, 8-49

Arithmetic

- decimal 2-15
- floating-point, operations 3-5
- instructions
  - decimal 2-20
  - fixed-point 2-4
  - floating-point 3-7
- logic unit 1-23

ASCII

- fault
  - codes 5-20
  - data 5-19, 5-21, 5-22
  - servicing 5-19
- manipulation. *See* Byte manipulation

Attribute block, graphics 7-13

ATU. *See* Address translation unit

## B

Base register

- address 1-12
- segment 5-11

Base-level interrupt processing 8-12

Binary fixed-point operations 2-2

Bit

- address 1-17
  - ECLIPSE 16-bit 10-10, 10-11
- data 1-16
- instructions
  - fixed-point 2-13
  - modified 9-10
  - referenced 9-10
- least significant 1-2
- manipulation, fixed-point 2-13
- modified 9-9
- most significant 1-2
- pattern 1-2
- pointer 1-16
  - format 1-17
- referenced 9-9
- reserved 1-2

- Bitmap, graphics
  - and forms 7-5
  - physical, definition 7-5
  - virtual
    - definition 7-5
    - windowing with 7-5
- BKPT** 5-6
- Block
  - counter, device controller 8-31
  - diagrams
    - microMV, 1-24
    - system, 1-19
  - standard return
    - narrow 10-5
    - wide 4-6, 5-5
- BMC
  - address modes 8-23
  - maps 8-23
    - instructions 8-22
  - registers 8-25
    - slot 8-26
  - status registers A-9
  - see also* Burst multiplexor channel
- Boot clock 8-59
- Bounding rectangle, graphics 7-6
- Breakpoint
  - handler, return from 5-6
  - instruction 5-6
- Building a queue 6-2
- Burst multiplexor channel 1-6, 1-27, 8-20
  - bandwidths 1-27
  - burst counter, 8-32
  - control, 8-4
  - I/O 8-1
  - latency, 8-35
- Busy flags 8-7
- Byte
  - address 1-16
    - ECLIPSE 16-bit 10-9, 10-10
  - data 1-14
  - decimal data formats 2-15
  - decimal operations 2-15
  - instructions
    - move 2-19
    - skip 2-21
    - swap 2-8
  - manipulation 2-15
  - pointer 1-14, 1-16
    - format 1-15

## C

- Cache
  - data 1-23
  - form 7-21
  - tag 7-21

- Call subroutine
  - instructions 5-6
  - sequence 5-9
- Carry
  - ECLIPSE 16-bit corresponding bits 10-3
  - fixed-point
    - initializing instructions 2-7
  - operations 2-7
  - flag 2-7
- Central processor 1-20, 8-36
  - device control flags 8-36
  - identification 9-10
    - ECLIPSE 16-bit 10-18
  - instructions 8-37
    - Halt 8-45
    - I/O 8-37
    - identification 9-10
  - multiple units 8-88
- Channel
  - burst multiplexor 8-20
  - data 8-20
  - I/O
    - controllers 8-2
    - instruction
      - Reset 8-41
      - Select 8-39
    - register
      - definition 8-26
      - mask 8-27
      - status 8-27
- Character, graphics
  - color 7-14, 7-17
  - control 7-14, 7-17
  - drawing attributes 7-17
  - fonts 7-17
- CIO** 8-22
- CIOI** 8-22
- Clippable area, graphics 7-25
- Clock
  - alarm 8-50
    - instructions 8-50
  - architectural 8-49
  - boot 8-59
  - real-time 8-63
    - device flag control 8-63
    - instructions 8-63
  - system 1-30
  - time-of-day 8-50
  - time-slice timer 8-54
- Codes, fault
  - and status B-1
  - decimal/ASCII 5-20
  - I/O devices E-1
  - multiple central processing units 8-93
  - wide stack 5-24

- Color, graphics
  - character 7-14, 7-17
  - descriptors 7-20
  - line 7-14, 7-16
  - style 7-14, 7-16
- Combination rule, graphics 7-13, 7-14, 7-15
- Communications controllers 1-28
- Computation
  - fixed-point 1-2, 2-1
  - floating-point 1-3, 3-1
- Context block formats F-1
- Contiguous line segments, graphics 7-16
- Control register, CPU dedication 8-28
- Control panel, 1-29
- Control Store, Load instruction D-1
- Controller
  - device 8-2, 8-29
  - registers 8-29
  - I/O channel 8-2
- Conversion instructions
  - fixed-point 2-3
  - fixed-point to floating-point and store 2-20
  - floating-point 3-3
    - binary 3-3
    - decimal 3-3
- Coordinate system, graphics 7-7
- Coordinates, graphics
  - conversion 7-8
  - physical 7-7
  - user 7-7
  - virtual 7-7
- Counter, program 5-1
- CPU
  - dedication control register 8-28
  - identification A-9
  - multiple units. *See* Multiple central processing units
  - Skip instruction 8-48
  - see also* Central processor
- Current ring of execution 5-9
- Current segment 1-9
- Cursor descriptor 7-17
  - cross-hair 7-18, 7-19
  - image 7-18, 7-19

## D

- Data
  - bit 1-16
  - byte 1-14

- cache 1-23
- channel 1-27, 8-20
  - bandwidths 1-27
  - I/O 1-6, 8-1
  - control, 8-4
  - latency, 8-35
- doubleword 1-14
- element, queue 6-1
- fault
  - ASCII 5-21, 5-22
  - decimal 5-21, 5-22
  - decimal/ASCII 5-19
  - servicing 5-19
  - type 2-22
- formats
  - byte and decimal 2-15
  - fixed-point 2-2, 2-12
  - floating-point 3-2
  - type indicator 2-15
- register 8-31
- structures, graphics 7-9
- transfer
  - I/O 8-22
  - initialization 8-3
  - latency 8-33
- type
  - decimal and byte 2-17
  - indicator format 2-16
- wide stack instructions 4-5
- word 1-14
- word-oriented 1-14
- DCH
  - maps 8-23
    - instructions 8-22
  - registers 8-25
    - slot 8-26
  - status registers A-9
  - see also* Data channel
- Decimal
  - arithmetic 2-15
  - example 2-23
  - data
    - fault 5-21, 5-22
    - packed format 2-18
    - unpacked format 2-18
  - instructions
    - arithmetic 2-20
    - shift 2-21
  - packed string 2-16
  - unpacked 2-16
- Decimal and byte
  - data
    - formats 2-15
    - types 2-17
  - instructions, move 2-19
  - operations 2-15

- Decimal/ASCII, fault
  - codes 2-22, 5-20
  - data 2-22, 5-19
  - servicing 5-19
- ECLIPSE 16-bit 10-6, 10-17
- Decrement and skip instructions, fixed-point 2-6
- Demand paging 1-8, 9-9
- DEQUE** 6-6, 6-7
- Destination segment 1-9
- Device
  - access, I/O 8-2
  - codes E-1
  - controller, 8-2, 8-29
    - block counter, 8-31
    - data transfer latency, 8-33
    - flag, status, 8-30
    - programming, 8-32
  - register
    - BMC burst counter, 8-32
    - control, 8-30
    - data, 8-31
    - memory address, 8-32
    - status, 8-30
  - registers, 8-29
  - word counter, 8-31
- external, definition 8-2
- flag, control
  - central processor 8-36
  - power supply controller 8-80
  - programmable interval timer 8-60
  - real-time clock 8-63
  - SCP 8-68
  - TTY 8-65
- I/O
  - control table 8-16
  - flags 8-7
  - integral 8-36
- internal, definition 8-1
- management 1-6, 8-1
- maps, I/O 8-22
- timing 8-49
- Diagnostic
  - remote
    - communications 1-29
    - processor 1-28
    - troubleshooting 1-29
- Disable instruction, I/O Interrupt 8-46
- Displacement field 1-11
- Division instructions
  - fixed-point 2-6
  - floating-point 3-8, 3-9
- Done flags 8-7
- DO-loop, instructions 5-3
  - example 5-3

- Doubleword
  - data 1-14
  - definition 1-2
- Drawing attributes, graphics
  - character 7-17
  - line 7-16
- DRP. *See* Diagnostic remote processor

## E

- ECLID** 10-18, A-9
- ECLIPSE 16-bit
  - addressing
    - bit 10-10, 10-11
    - byte 10-9, 10-10
    - effective 10-9
    - word 10-8, 10-9
  - CPU identification 10-18
  - compatible instructions 1-8
  - corresponding bits
    - Carry 10-3
    - fixed-point accumulators 10-2
    - floating-point
      - accumulators 10-2
      - status register 10-2
    - processor status register 10-2
    - program counter 10-3
    - registers 10-3, 10-4
    - wide stack registers 10-2
  - fault 10-6
    - decimal/ASCII 10-6, 10-17
    - floating-point 10-6
    - handling 10-17
  - floating-point, numerical algorithms 10-14
  - instructions 10-7
    - ECLIPSE MV/Family compatibility 10-8
    - fixed-point 10-12
    - floating-point 10-13
    - memory reference 10-8
    - program flow 10-15
    - stack 10-16
  - interrupts 10-6
    - I/O 10-6
  - page zero memory 10-18
  - program
    - expansion 10-6
    - flow 10-17
  - programming 10-1
  - registers 10-2
  - reserved memory 10-18
  - stack 10-5
  - subroutine expansion 10-7
- Edit
  - instruction, wide 5-7
  - subprogram instructions 2-20

- Effective address 1-12, 1-13
  - ECLIPSE 16-bit 10-9
  - instructions 2-21
  - resolution 1-10
- Enable instruction, I/O Interrupt 8-47
- Enable/Disable Error Reporting instruction, SCP 8-71
- ENQH** 6-7
- ENQT** 6-5, 6-7
- Erasing graphics objects 7-15
- ERCC, 1-25
- Error
  - checking and detection, 1-25
  - logging, 1-29
- Execute accumulator instruction 5-2
- Execution, instruction times G-1
- Expanding an ECLIPSE 16-bit
  - program 10-6
  - subroutine 10-7
- External device definition 8-1

## F

- Fault 1-18
  - ASCII, data 5-21
  - codes
    - and status B-1
    - decimal/ASCII 2-22, 5-20
    - protection 9-15
    - wide stack 5-24
  - data
    - ASCII 5-21, 5-22
    - decimal 5-21, 5-22
  - decimal/ASCII
    - data 2-22, 5-19
    - servicing 5-19
  - ECLIPSE 16-bit 10-6, 10-17
  - ECLIPSE 16-bit 10-6
    - fixed-point
      - overflow 2-10
      - servicing 5-17
    - flag, overflow 1-3
    - floating-point 3-12, 5-18
      - ECLIPSE 16-bit 10-6
      - overflow 1-4
      - servicing 5-18
      - underflow 1-4
    - graphics, handling 7-22, 7-23
      - sequence 7-22, 7-23
    - handler
      - protection, user 9-17
      - stack
        - narrow 5-25
        - wide 5-24
    - handling 5-16
      - ECLIPSE 16-bit 10-17

- instructions, Set Time-Slice Handler 8-58
- mask
  - floating-point 5-18
  - overflow 1-3
- nonprivileged, sequence 5-16
- overflow
  - fixed-point 5-17
  - narrow stack 5-24
- page 9-11, 9-12
- privileged 9-11
- protection 9-13
- return block
  - narrow 5-22
  - wide 5-21
- stack 5-23
  - narrow 5-24
    - operations 5-24
    - sequence 5-25
  - wide 4-8
    - operations 5-23
    - overflow 5-23
    - sequence 5-23
    - underflow 5-23
- types 5-16
- Final interrupt processing 8-14
- Fixed-point
  - accumulators 1-2, 1-3
    - ECLIPSE 16-bit corresponding bits 10-2
  - binary operations 2-2
  - bit manipulation 2-13
  - Carry operations 2-7
  - computation 1-2, 2-1
  - data formats 2-2, 2-12
  - fault
    - overflow 2-10, 5-17
    - servicing 5-17
  - graphics, overflow 7-25
  - instructions
    - addition 2-4
    - arithmetic 2-4
    - bit 2-13
    - Carry initializing 2-7
    - conversion to floating-point 2-20
    - decrement and skip 2-6
    - division 2-6
    - ECLIPSE 16-bit 10-12
    - increment and skip 2-6
    - logical 2-13
      - shift 2-14
    - skip 2-14
    - move 2-3
    - multiplication 2-5
    - precision conversion 2-3
    - shift 2-7, 2-14
    - skip 2-9, 2-14
    - skip on condition 2-9
    - subtraction 2-5
    - swap byte 2-8

- Fixed-point (continued)
    - logical operations 2-12
    - processor status register 2-10
    - two's complement format 2-2
  - Flag
    - Busy 8-7
    - Carry 2-7
    - device control
      - central processor 8-36
      - power supply controller 8-80
      - programmable interval timer 8-60
      - real-time clock 8-63
      - SCP 8-68
      - TTY 8-65
  - Flag (continued)
    - Done 8-7
    - IXCT 5-6
    - interrupt 8-7, 8-8
      - disable, device controllers 8-18
    - I/O devices 8-7
    - OVR 5-17
    - powerfail 8-7
    - status, device controller 8-30
  - Floating-point
    - accumulators 1-3, 1-4
      - ECLIPSE 16-bit corresponding bits 10-2
    - aligning the mantissas 3-5
    - appending guard digits 3-5
    - arithmetic operations 3-5
    - calculating and normalizing the result 3-6
    - computation 1-3, 3-1
    - data formats 3-2
    - exponent 3-2
    - fault 5-18
      - and status 3-12
      - ECLIPSE 16-bit 10-6
      - mask 5-18
      - return block
        - narrow 5-19
        - wide 5-18
      - servicing 5-18
    - instructions
    - hardware assist, 1-23
      - addition 3-7
      - arithmetic 3-7
      - conversion 3-3
        - binary 3-3
        - decimal 3-3
      - from fixed-point and store 2-20
      - division 3-8, 3-9
      - ECLIPSE 16-bit 10-13
      - intrinsics 3-10, 3-11
      - move 3-4
      - multiplication 3-8
      - skip 3-9
      - skip on condition 3-9
      - status register 3-12
      - subtraction 3-7
    - mantissa 3-2
    - numerical algorithms, ECLIPSE 16-bit 10-14
    - registers 1-3
    - result
      - storing 3-6
      - truncating or rounding 3-6
    - sign bit 3-2
    - status register 1-3, 1-4, 3-12, 5-18, G-3
    - unit 1-24
      - ECLIPSE 16-bit corresponding bits 10-2
  - Form
    - and bitmaps, graphics 7-5
    - attributes 7-13, 7-14
    - cache 7-21
      - miss 7-21
      - tag 7-21
    - descriptor 7-9
      - contents 7-10, 7-11
      - definition 7-3
    - graphics 7-4
      - data structures 7-4
      - ID definition 7-4
      - ID, user's 7-21
    - mask 7-12
  - FPSR 3-13
    - see also* Floating-point status register
  - FPU. *See* Floating-point unit
  - Frame pointer, narrow 10-5
  - Front panel 1-30
  - FTD** 5-18
  - FTE** 5-18
  - Functional capabilities 1-2
  - FXTD** 5-17
  - FXTE** 5-17
- ## G
- Gate
    - array 5-9
    - format 5-10
    - bracket 5-11
  - General I/O instructions 8-6
  - GIS. *See* Graphics instruction set
  - Glossary Glossary-1
  - Graphics
    - attribute block 7-13
    - bitmap
      - physical 7-5
      - virtual 7-5
    - bounding rectangle 7-6
    - character
      - color 7-14, 7-17
      - control 7-14, 7-17
      - drawing attributes 7-17
      - fonts 7-17
    - clippable area 7-25



- Graphics (continued)
  - color
    - descriptors 7-20
    - line 7-14
  - combination rule 7-13
  - coordinate system 7-7
  - coordinates
    - conversion 7-8
    - physical 7-7
    - user 7-7
    - virtual 7-7
  - cursor descriptor 7-17
    - cross-hair 7-18, 7-19
    - image 7-18, 7-19
  - data structures 7-9
  - erasing objects 7-15
- Graphics (continued)
  - fault
    - fixed-point overflow 7-25
    - handling 7-22
    - overdraw condition 7-25
  - form 7-4
    - and bitmaps 7-5
    - attributes 7-13, 7-14
    - cache 7-21
    - data structures 7-4
    - descriptor 7-9
      - contents 7-10
    - ID 7-4
    - mask 7-12
  - instruction set 1-6
  - instructions 7-1
    - format 7-2
    - interrupts 7-21
  - line
    - color 7-16
    - control 7-14, 7-16
    - drawing attributes 7-16
    - style 7-14, 7-16
  - local origin 7-6
  - management 1-6, 7-1
  - operation mask 7-13, 7-14
  - palette sharing 7-12
  - rectangle
    - descriptor 7-6, 7-12
      - contents 7-13
    - list 7-6, 7-12
  - tiling 7-12
  - undrawing objects 7-15
  - user's form ID 7-21
  - windowing 7-5
- Guard digit, floating-point 3-5

## H

**HALT** 8-45  
 Halt instruction 8-45

**Handler**  
 breakpoint, return from 5-6  
 fault
 

- protection, user 9-17

 stack
 

- narrow 5-25
- wide 5-24

**Hardware summary** 1-19  
**Hex shift instructions** 2-21

## I

**I/O**  
 access 8-3
 

- segment base register 8-3

 burst multiplexor channel 1-6, 8-1  
 bus 1-25, 1-27, 8-2  
 channel 8-20
 

- controllers 1-25, 1-26, 8-2
  - multiple 8-39, 8-41
- multiple central processing units 8-91

 register
 

- definition 8-26
- mask 8-27
- status 8-27

 communication 8-2
 

- multiple central processing units 8-91

 data channel 1-6, 8-1  
 data transfers 8-4, 8-22
 

- control 8-4

 device
 

- access 8-2
- codes E-1
- control table 8-16
- flags 8-7
- integral 8-36
- maps 8-22

 instructions
 

- burst multiplexor channel, maps 8-22
- central processor 8-37
- Channel Reset 8-41
- Channel Select 8-39
- data channel, maps 8-22
- general 8-6

 Interrupt
 

- Acknowledge 8-42
- Disable 8-46
- Enable 8-47
- Mask Out 8-44
- Reset 8-43

 interrupt 8-8
 

- ECLIPSE 16-bit 10-6
- multiple central processing units 8-91
- servicing 8-10, 8-11
- vectored 8-12

 primary asynchronous line 8-65  
 programmed 1-6, 8-4  
 registers 8-3, 8-25

- I/O (continued)
    - Skip 8-8
      - instruction 8-17
      - system 1-24, 1-27
      - validity flag 1-18
      - transfer sequence 8-21
      - vector table 8-14, 8-15
  - Identification, central processor 9-10
    - ECLIPSE 16-bit 10-18
    - instructions 9-10
  - IIS. *See* Intrinsic instructions
  - INTA 8-17, 8-42
  - INTDS 8-46
  - INTEN 8-18, 8-47
  - Increment and skip instructions, fixed-point 2-6
  - Index field 1-11
  - Indirect, address 1-12
    - field 1-12
    - protection violation 1-12
  - Indivisible instructions, multiple central processing units 8-90
  - Information transfer, types 8-4
  - Initial processor 8-88
  - Initialization
    - multiple central processing units 8-88
    - powerup 1-28
    - processor 1-30
    - wide stack 4-7
  - Input/output subsystem, 1-26
  - Instructions
    - compatibility, ECLIPSE MV/Family 10-8
    - ECLIPSE 16-bit compatibility 1-8
    - execution times G-1
    - interruption 8-9
    - interrupts, graphics 7-21
    - memory reference 1-8, 1-10
    - pipeline 1-21
    - processor 1-22
      - status register 2-10
    - resumable 8-9
    - unimplemented 9-17
  - Integral devices, I/O 8-36
  - Intermediate-level interrupt processing 8-14
  - Internal device definition 8-1
  - Interrupt
    - Acknowledge 8-17
    - disable flag, device controllers 8-18
    - ECLIPSE 16-bit 10-6
    - flag 8-7, 8-8
    - graphics instructions 7-21
    - I/O 8-8
      - ECLIPSE 16-bit 10-6
    - processing
      - base-level 8-12
      - final 8-14
      - intermediate-level 8-14
      - servicing 8-10, 8-11
      - vectored 8-12
    - instructions 8-9
    - I/O
      - Acknowledge 8-42
      - Disable 8-46
      - Enable 8-47
    - multiple central processing units 8-91
    - priority 8-18
      - handler 8-19
    - mask 8-19
    - service routines 8-17
  - Interval timer, programmable 8-60
  - Intra-processor communication, multiple central processing units 8-92
  - Intrinsic instructions 3-10, 3-11
    - format 3-10, 3-11
  - Inward segment crossing sequence 5-11, 5-13
  - IOC. *See* I/O channel controllers
  - ION. *See* Interrupt flag
  - IORST 8-43
  - IXCT flag 5-6
- ## J
- Jump, instructions 5-2
    - to subroutine 5-6
- ## L
- LCALL 5-6, 5-9, 5-10, 5-11, 5-12
  - LCPID A-9
  - LCS. *See* Load Control Store instruction
  - Line, graphics
    - color 7-14, 7-16
    - contiguous segments 7-16
    - control 7-14, 7-16
    - drawing attributes 7-16
    - style 7-14, 7-16, 7-17
  - Links, queue 6-2
    - backward 6-2
    - forward 6-2
  - LJSR 5-6
  - Load
    - Character Buffer instruction 8-67
    - Control Store instruction D-1
    - effective address instructions 2-21
    - map, from I/O device 8-24
  - Local origin, graphics 7-6

- Logical
  - address 1-8
    - format 9-6
    - space 1-7, 1-8
  - fixed-point
    - instructions 2-13
      - shift 2-14
      - skip 2-14
    - operations 2-12
  - memory 9-2

**LPHY** 8-22

**LPTE** 9-4

**LSBRA** 9-2

**LSBRS** 9-2

## M

Main systems 1-20

Mantissa, aligning floating-point 3-5

MAP. *See* Memory allocation and protection

Map, loading, from I/O device 8-24

Mask

- interrupt, priority 8-19
- Out, instruction 8-8, 8-44

Memory

- accessing 1-8
- address register, device controller 8-32
- addressing, 1-24
- allocation and protection 8-22
- control unit 1-25
- cycle time, 1-26
- logical 9-2
- management 1-7, 9-1
- modules 1-26
- page zero 9-18
  - ECLIPSE 16-bit 10-18
- locations
  - segment 0 9-19
  - segment 1 through 7 9-20
- physical 9-2
- reference
  - instructions 1-8, 1-10
    - byte addressing format 1-10, 1-11
    - ECLIPSE 16-bit 10-8
    - word addressing format 1-10
  - validity 1-18
- reserved 9-18
  - ECLIPSE 16-bit 10-18
- locations C-1
- segment 1-5
  - see also* Segment
- state area 9-21
- support, microMV, 1-24
- system 1-25
- views, multiple central processing units 8-90

microMV

- block diagram, 1-21
- memory support, 1-24
- multiprocessor support, 1-24
- processor, 1-20

Microsequencer 1-22

Modified bit 9-9

- instructions 9-10

Move instructions

- byte 2-19
- decimal and byte 2-19
- fixed-point 2-3
- floating-point 3-4

**MSKO** 8-8, 8-18, 8-19, 8-44

Multiple central processing units 8-88

- error codes 8-93
- I/O

- communication 8-91
- interrupt handling 8-91
- initialization 8-88
- instructions 8-92
  - indivisible 8-90
  - serializable 8-90
  - uninterruptible 8-90
- intra-processor communication 8-92
- memory views 8-90
- multiple I/O channels 8-91
- operation 1-24

Multiplication instructions

- fixed-point 2-5
- floating-point 3-8

## N

Narrow

- frame pointer 10-5
- return block 10-5
  - fault 5-22
  - floating-point 5-19
- stack 10-5
  - definition 1-5, 4-1
  - fault
    - handler 5-25
    - operations 5-24
    - overflow 5-24
    - return block 5-25
    - sequence 5-25
  - limit 10-5
  - pointer 10-5

**NCLID** 10-18, A-9

Nonprivileged fault, sequence 5-16

## O

One-level pagetable 9-6, 9-7

Operand access 1-13

Operation mask, graphics 7-13, 7-14

OVK mask 5-17

- OVR flag 5-17
- Overdraw condition, graphics 7-25, 7-26, 7-27
- Overflow, fault
  - fixed-point 2-10, 5-17
  - graphics 7-25
  - flag 1-3
  - floating-point 1-4
  - mask 1-3
  - stack
    - narrow 5-24
    - wide 4-8, 5-23
    - disabling 4-8

## P

- Packed decimal
  - data format 2-18
  - string 2-16
- Page
  - access 9-2, 9-9
  - validation 9-9
  - definition 1-8
  - fault 9-11
  - sequence 9-12
  - frames 9-4
  - protocols 1-9
  - zero
    - ECLIPSE 16-bit 10-18
    - memory 9-18
    - locations C-1
    - segment 0 9-19
    - segment 1 through 7 9-20
- Pagetable 9-4
  - entry 9-4, 9-5
  - one-level 9-6, 9-7
  - two-level 9-6, 9-8
- Paging, demand 9-9
- Palette sharing, graphics 7-12
- PBX** 5-6, 8-9
- PC. *See* Program counter
- Physical
  - bitmap, graphics, definition 7-5
  - memory 9-2
- Pipeline, instruction 1-21
- PIO** 8-6, 8-17
- PIT. *See* Programmable interval timer
- Pointers, trojan horse 5-14
- Power supply controller 1-28, 8-80
  - device flag control 8-80
  - instructions 8-81
    - Read Data From U/PSC 8-86
    - Request Data From U/PSC 8-85
    - Write Data to U/PSC 8-82
  - interface 1-30
- Power system 1-28
- Powerfail flag 8-7, 8-48

- Powerup 1-28
- Priority interrupt 8-18
  - handler 8-19
  - mask 8-19
- Primary asynchronous line 8-65
- Privileged faults 9-11
- Processor
  - central 8-36
  - initial 8-88
  - initialization 1-30
  - instruction 1-22
  - interrupt servicing 8-10
  - microMV 1-20
  - state block, multiple central processing units 8-89
  - status register 1-2, 1-3, 5-17, 8-9
    - ECLIPSE 16-bit corresponding bits 10-2
    - fixed-point 2-10
    - instructions 2-10
- Program
  - control
    - segment transfer instructions 5-9
    - transferring to another segment 5-9, 5-10
  - counter 1-5, 5-1
    - ECLIPSE 16-bit corresponding bits 10-3
    - format 1-5, 1-6
  - expansion, ECLIPSE 16-bit 10-6
  - flow
    - ECLIPSE 16-bit 10-17
    - instructions 10-15
    - management 1-5, 5-1
    - related instructions 5-2
- Programmable interval timer 1-20, 8-60
  - device flag control 8-60
  - instructions 8-60
    - Read Count 8-61
    - Specify Initial Count 8-62
- Programmed I/O 1-6, 8-1
  - control 8-4
  - instruction 8-6, 8-17
  - latency 8-33
- Protection
  - capabilities 1-18
  - faults 1-18
  - violation 9-13
    - codes 9-15
    - handler, user 9-17
    - indirect address 1-12
    - priorities 9-13
    - return block 9-15
    - sequence 9-13, 9-16
- Protocols
  - page 1-9
  - segment 1-9
- PRTRST** 8-41
- PRTSEL** 8-20, 8-39
- PSC 1-28, 8-80

PSR. *See* Processor status register

PTE. *See* Pagetables, entry

## Q

### Queue

- building 6-2
- data element 6-1
- definition 1-6, 6-1
- descriptor 6-3
  - empty queue 6-3
- examples 6-3
- head 6-1
- instructions 6-7
- links 6-2
- management 1-6, 6-1
- setting up and modifying 6-3
- tail 6-1

## R

### Read

- Character Buffer instruction 8-66
- Count instruction, PIT 8-61
- Data From U/PSC instruction 8-86
- Switches instruction 8-38
- Time of Day instruction 8-51
- Time-Slice instruction 8-55

### READS 8-38

### Real-time clock 1-29, 8-63

- device flag control 8-63
- instructions 8-63
  - Select RTC Frequency 8-64

### Rectangle, graphics

- bounding 7-6
- descriptor 7-12
  - contents 7-13
- definition 7-6
- list 7-12
  - definition 7-6
  - use of 7-6

### Referenced bit 9-9

- instructions 9-10

### Register 1-2

- base address 1-12
- BMC 8-25
  - slot 8-26
- CPU dedication control 8-28
- DCH 8-25
  - slot 8-26
- device controller 8-29
  - BMC burst counter, 8-32
  - control, 8-30
  - data, 8-31
  - memory address, 8-32
  - status, 8-30

### ECLIPSE 16-bit 10-2

- corresponding bits 10-3, 10-4
- fields A-1
- floating-point 1-3
  - status 1-3, 1-4, 3-12, 3-13, 5-18
- I/O 8-3, 8-25
  - channel
    - definition 8-26
    - mask 8-27
    - status 8-27
  - least significant bit 1-2
  - most significant bit 1-2
  - processor status 1-2, 1-3, 2-10, 5-17
  - segment base 1-8, 5-11, 9-2, 9-3
  - wide stack 4-3
    - format 4-3
    - instructions 4-4, 4-5
    - management 1-5

### Relative addressing 1-12

### Remote diagnostic communications 1-29

### Removing a data element 6-6

### Request Data From U/PSC instruction 8-85

### Reserved bits 1-2

### Reserved memory 9-18

- ECLIPSE 16-bit 10-18
- locations C-1
- page zero 9-18
- state area 9-21

### Reset instruction

- I/O 8-43
- I/O Channel 8-41

### Resumable instructions 8-9

### Return

- breakpoint handler 5-6
- SCP Status instruction 8-78.1
- subroutine 5-14
  - instructions 5-6
- wide, instruction sequence 5-15

### Return block

- fault
  - narrow 5-22
    - floating-point 5-19
    - stack 5-25
  - protection 9-15
  - wide 5-21
    - floating-point 5-18
    - stack 5-23
    - standard 5-5
- narrow 10-5
  - standard 10-5
- wide
  - stack, instructions 4-6
  - standard 4-6

Ring  
  current, execution 5-9  
  definition 1-7  
RTC. *See* Real-time clock  
**RTOD** 8-51  
**RTS** 8-55

## S

SBR. *See* Segment base registers  
**SCP**  
  device control flag 8-68  
  instructions 8-68  
    Enable/Disable Error Reporting 8-71  
    Return SCP Status 8-78.1  
  protocol 8-68  
  *see also* System control processor/program  
Scale factor. *See* Data type indicator format  
Segment  
  access 9-2  
  base register 1-8, 5-11, 9-2, 9-3  
    I/O access 8-3  
  crossing sequence, inward 5-11, 5-13  
  current 1-9  
  definition 1-5, 1-7  
  destination 1-9  
  instructions, program control transfer 5-9  
  other 1-9  
  protocols 1-9  
  *see also* Gate array  
Select RTC Frequency instruction 8-64  
Serializable instructions, multiple central  
  processing units 8-90  
Set  
  Alarm instruction 8-53  
  Time of Day instruction 8-52  
  Time-Slice Fault Handler instruction 8-58  
  Time-Slice instruction 8-56  
Shift instructions  
  decimal 2-21  
  fixed-point 2-7, 2-14  
Sign extend, fixed-point 2-2  
Size indicator. *See* Data type indicator format  
Skip instructions 5-2, 5-3  
  byte 2-21  
  CPU 8-48  
  example 5-3  
  fixed-point 2-9, 2-14  
  floating-point 3-9  
  on condition  
    fixed-point 2-9  
    floating-point 3-9  
**SKP CPU** 8-48  
**SKPt** 8-8, 8-17  
Specify Initial Count instruction, PIT 8-62

**SPTE** 9-4  
**SSPT** 9-21  
**STOD** 8-52  
**STS** 8-56  
**STSFH** 8-58  
Stack  
  definition 1-5, 4-1  
  ECLIPSE 16-bit 10-5  
  management 1-5, 4-1  
  narrow 10-5  
    definition 4-1  
    fault  
      handler 5-25  
      operations 5-24  
      overflow 5-24  
      return block 5-25  
      sequence 5-25  
  instructions, ECLIPSE 16-bit 10-16  
  limit 10-5  
  pointer 10-5  
  wide  
    base 4-3  
    definition 4-1  
    example 4-7, 5-7, 5-8  
    fault 4-8, 5-23  
      codes 5-24  
      handler 5-24  
      operations 5-23  
      overflow 5-23  
      return block 5-23  
      sequence 5-23  
      underflow 5-23  
    frame pointer 4-4  
    initializing 4-7  
    instructions  
      data 4-5  
      register 4-4, 4-5  
      return block 4-6  
    limit 4-3  
    management registers 1-5  
    parameters 4-2  
    pointer 4-4  
    register 4-3  
      format 4-3  
Standard return block  
  narrow 10-5  
  wide 4-6, 5-5  
State area 9-21  
State block, multiple CPUs 8-89  
Status  
  codes, and fault B-1  
  flags 1-3  
    and register, device controller 8-30  
  register  
    floating-point 1-3, 1-4, 3-12, 3-13, 5-18  
    instructions 3-12  
    processor 1-2, 1-3, 2-10, 5-17  
Store State Pointer instruction 9-21

- Subroutine 5-4
  - call sequence 5-9
  - expansion, ECLIPSE 16-bit 10-7
  - instructions 5-5
    - call 5-6
    - jump to 5-6
    - return from 5-6
    - sequence 5-5
  - return 5-14
- Subtraction instructions
  - fixed-point 2-5
  - floating-point 3-7
- Swap byte instructions, fixed-point 2-8
- System
  - board organization 1-19
  - block diagram 1-19
  - bus 1-26
  - cache 1-25
  - clock 1-30
  - console 1-30
  - control processor 8-68
  - control program 1-29, 8-68
  - hardware summary 1-19
  - management 1-7, 9-1
  - memory 1-25
  - overview 1-1

## T

- Tiling, graphics 7-12
- Time of Day instruction
  - Read 8-51
  - Set 8-52
- Time-of-day clock 8-50
- Time-slice
  - instructions
    - Read 8-55
    - Set 8-56
    - Set Handler 8-58
  - timer 8-54
    - instructions 8-54
- Timer, programmable interval 8-60
- Timing devices 8-49
- Transferring program control, another segment 5-9, 5-10
- Trojan horse pointers 5-14
- TTY
  - control flags 8-65
  - instructions 8-66
    - Character Buffer
      - Load 8-67
      - Read 8-66
  - see also* Primary asynchronous line

- Two's complement
  - format 2-2
  - fixed-point 2-2
  - precision 2-2
- Two-level pagetable 9-6, 9-8
- Type indicator. *See* Data type indicator format

## U

- Underflow, fault
  - floating-point 1-4
  - wide stack 4-8, 5-23
    - disabling 4-8
- Undrawing graphics objects 7-15
- Unimplemented instructions 9-17
- Uninterruptible instructions, multiple CPUs 8-90
- Universal power supply controller 8-80
- Unpacked decimal 2-16
  - data format 2-18
- UPSC 8-80
- User protection fault handler 9-17
- User's form ID, graphics 7-21

## V

- Validity
  - check
    - I/O instruction 1-18
    - memory reference 1-18
  - flag, I/O 1-18
- VBP** 5-14
- Vector table, I/O 8-14, 8-15
- Vectored interrupt processing, I/O 8-12
- Virtual bitmap, graphics
  - definition 7-5
  - windowing with 7-5
- VWP** 5-14

## W

- WASH** 2-7
- WASHI** 2-7
- WEDIT** 5-7
- WFP. *See* Wide frame pointer
- WGCHRBLT** 7-17
- WGLFORM** 7-21
- WGPLINE** 7-16
- WGRDPAL** 7-20
- WGWRPAL** 7-20

- Wide
  - edit instruction 5-7
  - return block, fault 5-21
    - floating-point 5-18
  - return instruction, sequence 5-15
  - stack
    - base 4-3
    - definition 1-5, 4-1
    - example 4-7, 5-7, 5-8
    - fault 4-8
      - codes 5-24
      - handler 5-24
      - operations 5-23
      - overflow 5-23
      - return block 5-23
      - sequence 5-23
      - underflow 5-23
  - frame pointer 4-4
  - initializing 4-7
  - instructions
    - data 4-5
    - register 4-4
    - return block 4-6
  - limit 4-3
  - operations 4-2
  - overflow 4-8
    - disabling 4-8
  - parameters 4-2
  - pointer 4-4
  - register 4-3
    - ECLIPSE 16-bit corresponding bits 10-2
    - format 4-3
    - instructions 4-5
    - management 1-5
  - underflow 4-8
    - disabling 4-8

Windowing with virtual bitmaps, graphics 7-5

**WLMP** 8-22

**WMESS** 6-7

Word

address, ECLIPSE 16-bit 10-8, 10-9

counter, device controller 8-31

data 1-14

definition 1-2

Word-oriented data 1-14

**WPOPJ** 5-6

Write Data to U/PSC instruction 8-82

**WRTN** 5-6, 5-14

sequence 5-15

**WSAVR** 5-6

**WSAVS** 5-6

WSB. *See* Wide stack base

WSL. *See* Wide stack limit

WSP. *See* Wide stack pointer

**WSSVR** 5-6

**WSSVS** 5-6

## X

**XCALL** 5-6, 5-9, 5-10, 5-11, 5-12

**XCT** 5-2

**XJSR** 5-6

**XVCT** 8-12, 8-17

## Z

Zero extend, fixed-point 2-2





ECLIPSE MV/9500™

System

Principles of Operation

014-001371 plus

014-001855-00

Cut here and insert in binder spine pocket



Data General Corporation, Westboro, Massachusetts 01580



014-001855-00