Data General

Customer Documentation

ECLIPSE MV/9500™ System Principles of Operation

Data General

Customer Documentation

ECLIPSE MV/9500™ System Principles of Operation Supplement

ECLIPSE MV/9500™ System Principles of Operation Supplement

014-001855-00

Ordering No. 014-001855

Printed in the United States of America
Rev. 00, March, 1990

Notice

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

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

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.

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

2 Fixed–Point Computing

6 Queue Management

7 Graphics Management

8 Device Management

9 Memory and System Management

10 ECLIPSE 16-Bit Programming

A Register Fields

B Fault and Status Codes

C Reserved Memory Locations

D Load Control Store Instruction

E Standard I/O Device Codes

F Context Block Formats

G Instruction Execution Times

Glossary

Figures

Figure

Figure

Tables

Table

Table

Table

Table

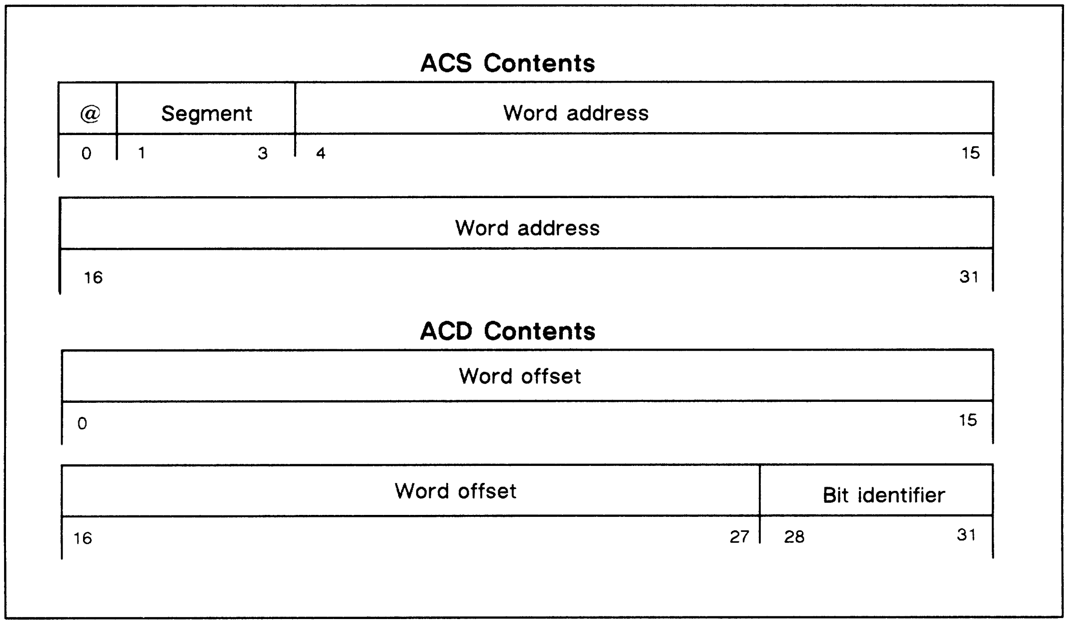


Figure 1-10 *Bit pointer format*

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.

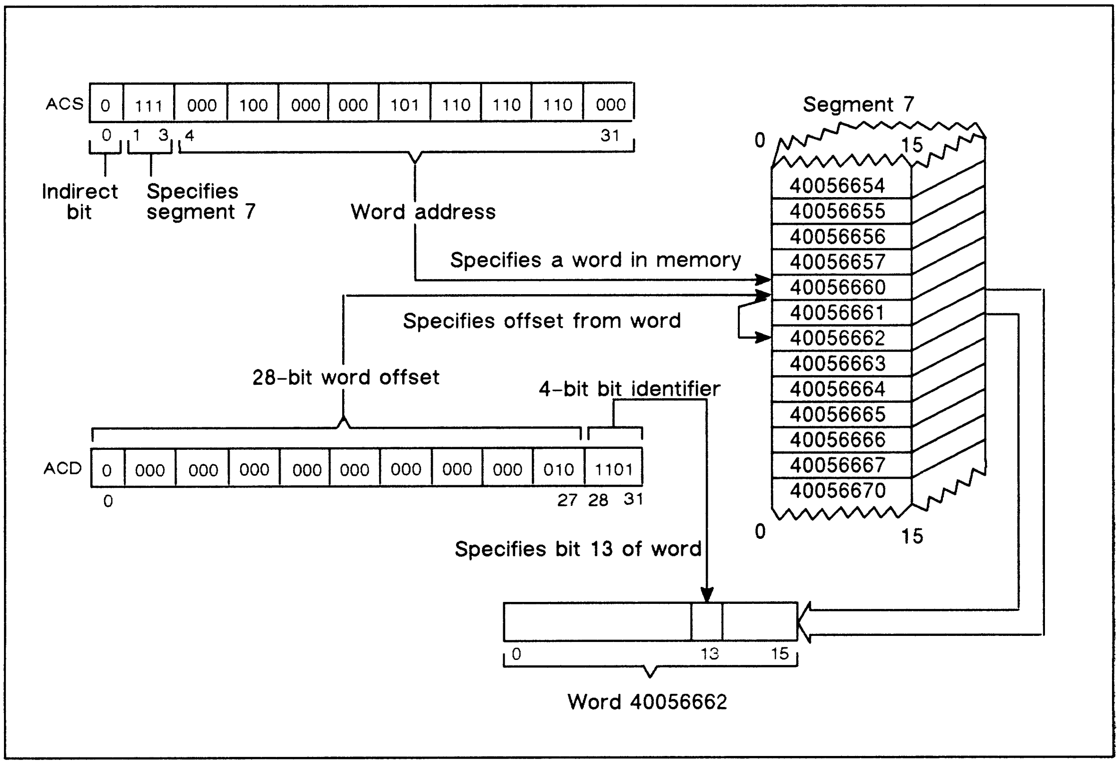


Figure 1-11 *Bit addressing*

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.

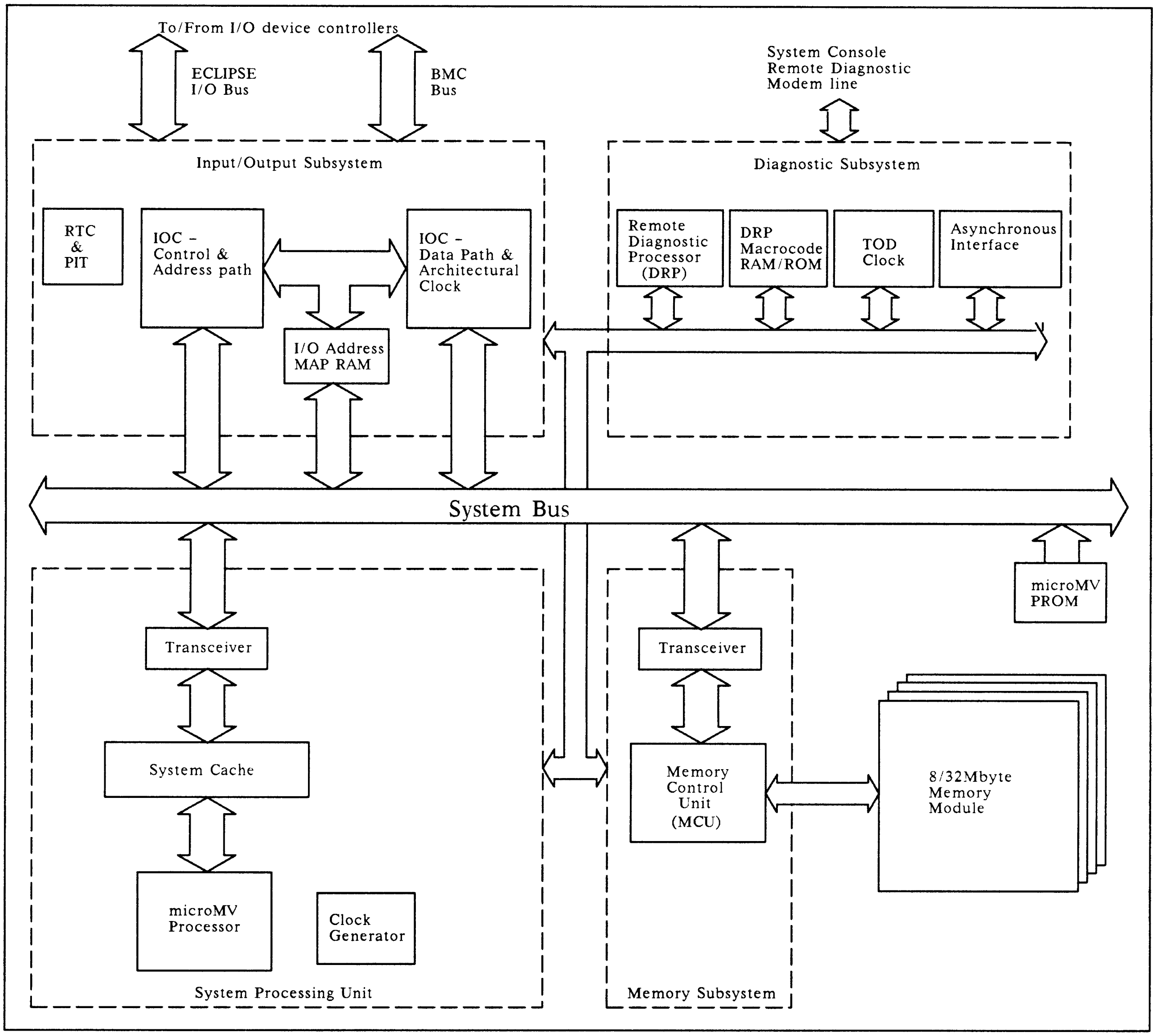


Figure 1-12 System Board Organization

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.

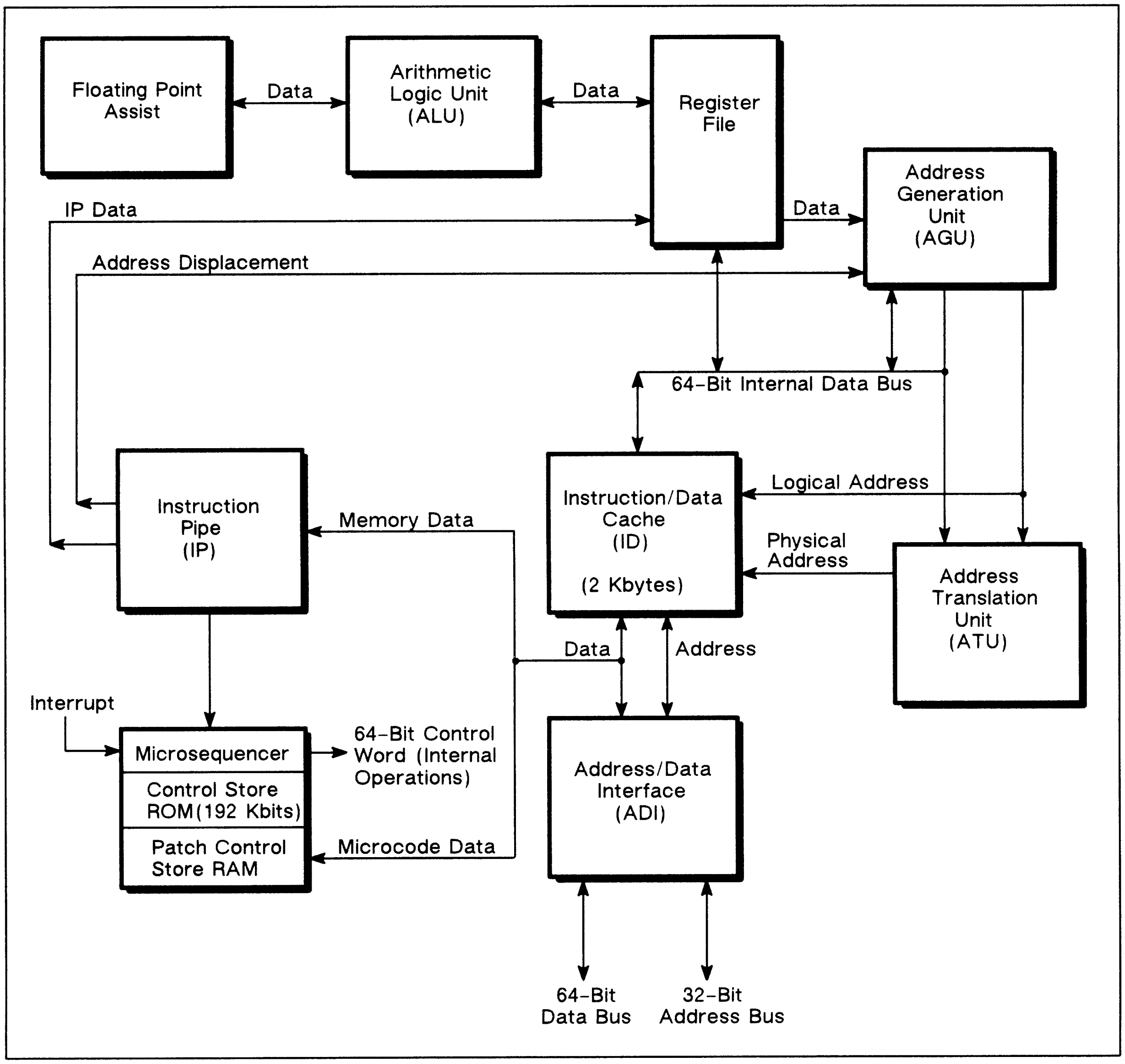
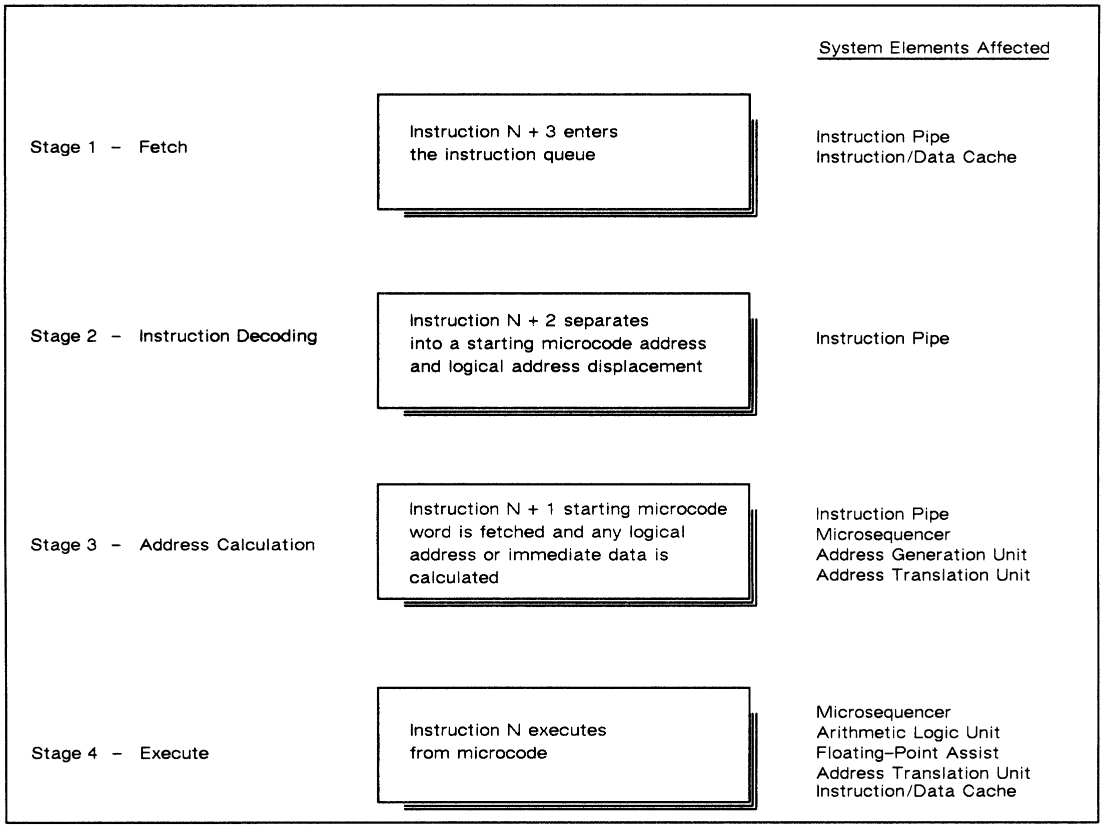


Figure 1-13 *Major Elements of the microMV Processor*

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 subseqent instruction (N+2), and first stage fetches another instruction (N+3), putting it into the instruction queue.



		System Elements Affected
Stage 1 – Fetch	Instruction N + 3 enters the instruction queue	Instruction Pipe Instruction/Data Cache
Stage 2 – Instruction Decoding	Instruction N + 2 separates into a starting microcode address and logical address displacement	Instruction Pipe
Stage 3 – Address Calculation	Instruction N + 1 starting microcode word is fetched and any logical address or immediate data is calculated	Instruction Pipe Microsequencer Address Generation Unit Address Translation Unit
Stage 4 – Execute	Instruction N executes from microcode	Microsequencer Arithmetic Logic Unit Floating-Point Assist Address Translation Unit Instruction/Data Cache

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 resisters, 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 accessses 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 architectual clocks, refer to Chapter 8.

PIT and RTC

The system clocks are software accessable 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.

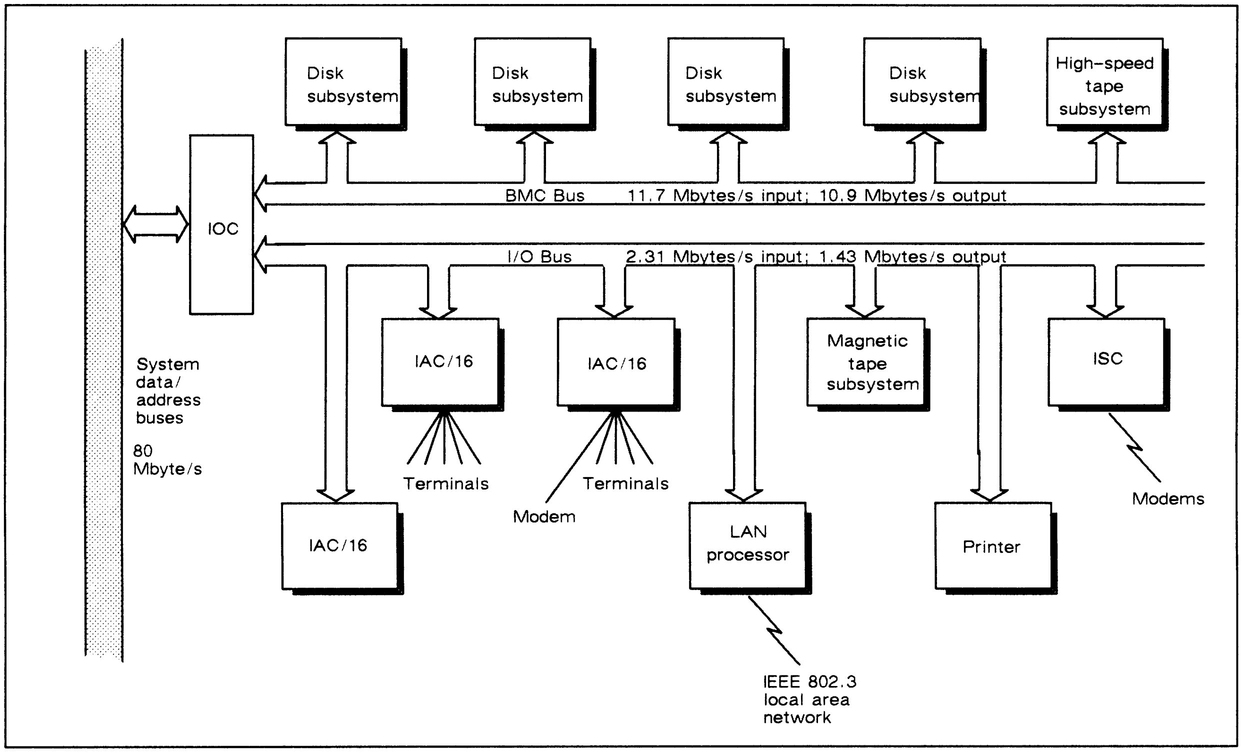

Disk subsystem
Disk subsystem
Disk subsystem
Disk subsystem
High-speed tape subsystem
IOC
BMC Bus 11.7 Mbytes/s input; 10.9 Mbytes/s output
I/O Bus 2.31 Mbytes/s input; 1.43 Mbytes/s output
IAC/16
IAC/16
Magnetic tape subsystem
ISC
System data/ address buses
80 Mbyte/s
Terminals
Terminals
Modem
Modems
IAC/16
LAN processor
Printer
IEEE 802.3 local area network

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.
- Multicommunications 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 facilites:

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

8

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.

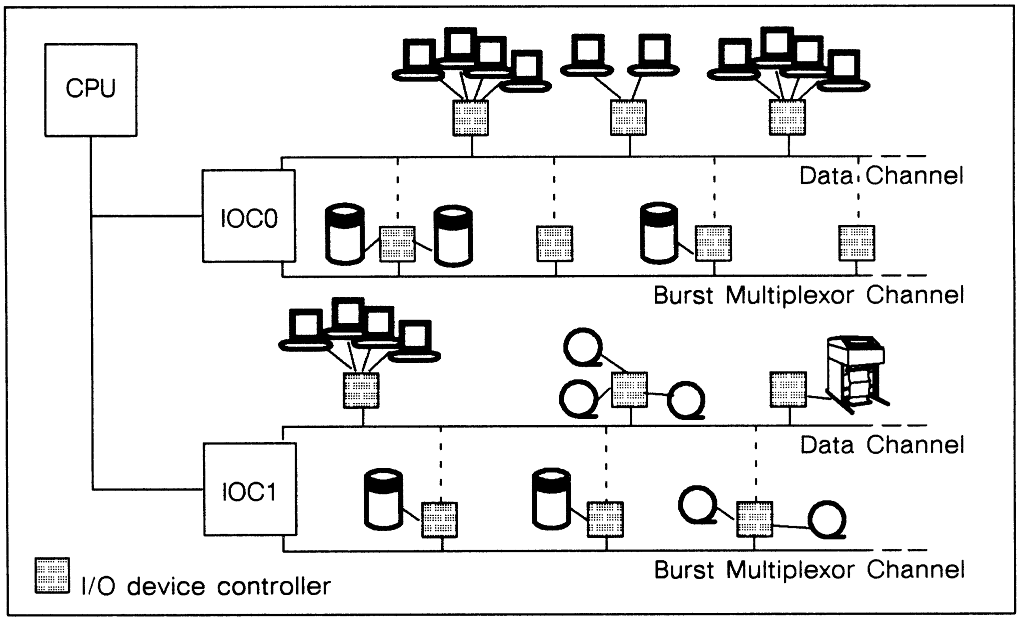


Figure 8-1 *An ECLIPSE MV/Family system with dual IOCs*

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
CIO, CIOI	Returns BMC/DCH status or loads map slots from accumulators (½ slot at a time).
IORST *	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).
WLMP	Loads BMC/DCH map slots from memory (as a block of doublewords).
LPHY	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_8, 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.

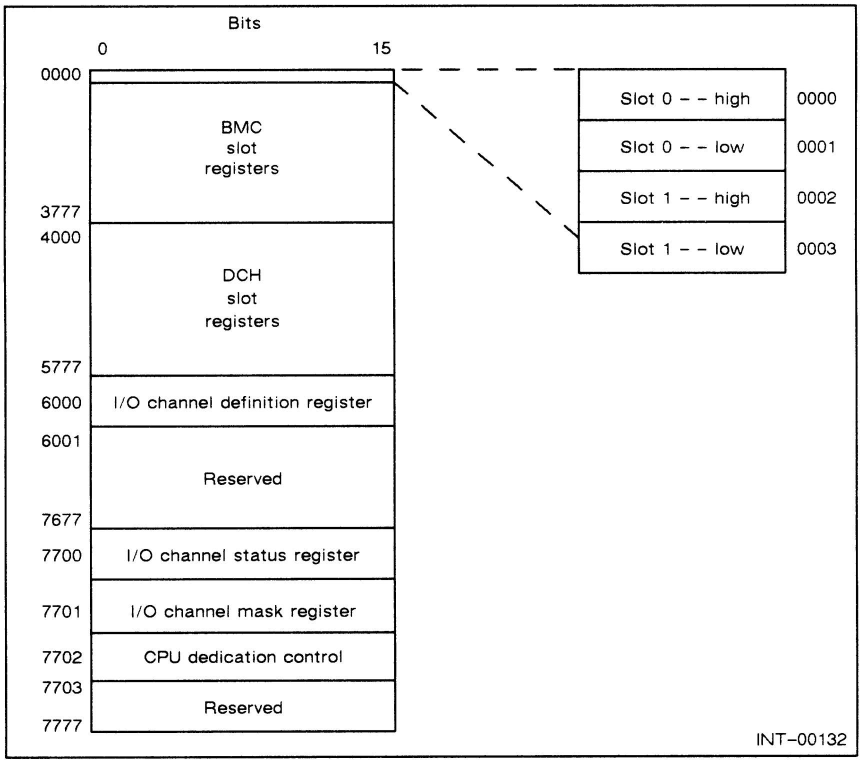


Figure 8–8 *DCH/BMC registers*

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 ($0000–3776_8$ and $4000–5776_8$, 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 ($0000–3777_8$ and $4001–5777_8$, 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_8) 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_8) 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_8) 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_8) 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

Device Code	77_8
Assembler Mnemonic	CPU
Priority Mask Bit	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:

f=omitted	Leaves ION unchanged.
f=**S**	Sets ION to 1.
f=**C**	Sets ION to 0.
f=**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_8. 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_8
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_8 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.

f=omitted	Leaves Busy and Done flags unchanged.
f=S	Sets the Busy flag to 1 and the Done and interrupt request flags to 0; begins the counting cycle.
f=C	Sets the Busy and Done flags and the interrupt request flag to 0; stops the counting cycle.
f=P	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*[f] ac*,PIT	Places the PIT counter value into the accumulator.
DOA*[f] ac*,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**

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

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:	
	Command (octal)	**Name**
	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_8). For the Size command (012_8), 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_8)

  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_8 through 003_8)
- Select SCP Powerdown Mode (command 002_8)
- Disable SCP Powerdown Mode (command 003_8)
- Set Block (command 004_8)

  The Set Block command specifies to the SCP the address of the interface block (bits 24–31).

- Enable All ERCC Error Reporting (command 005_8)

  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_8)
- Mask Soft ERCC (command 007_8)

  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_8)

  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_8)

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

- Size (command 012_8)

  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_8, $0A02_{16}$
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: **Bit(s)** **Contents** 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_8)

  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_8 into its error log, and posts an interrupt.

- Get Boot Clock (command 014_8)

  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_8)

  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_8 into its error log, and posts an interrupt.

- Get GMT Offset (command 016_8)

  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_8)

- Initiate MI call (command 020_8)

  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_8, 1000_{16}	
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_8, 022_8)

  These commands are reserved for Data General Field Service.

- Set Auto Restart Address (command 023_8)

  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_8, 1300_{16}
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_8 through 176_8)

- Enter Diagnostic Sequence (command 177_8)

  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 *ac*; otherwise unused.
Busy flag	Set to 1
Carry	Unchanged
Done flag	Set to 0
Overflow	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**

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

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: **Code Definition** **(octal)** 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) **ERCC Extended Status** The four-word extended status for ERCC codes 053, 054, and 055 is **Word Contents** 0 Status code (053, 054, or 055) 1 Status **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
	Code Definition **(octal)** 161-207 Reserved
000001	SCP reset. The SCP is reset and must be reinitialized with the **DOBS** *ac*,**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: **Sub-code** **Definition** 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_8	**Device Code**	4_8
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_8) 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.

f=omitted	Busy and Done flags unchanged.
f=S	Sets the Busy flag to 1 and the Done flag to 0.
f=C	Sets the Busy and Done flags to 0.
f=P	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
IORST	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.
HALT	Halt. Halts the processor and notifies the SCP that it is stopped.
SSPT	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 (JPSTOP, JPFLUSH, JPSTATUS) 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 (MSKO).
PRTSEL	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	JPLCS, JPSTART	Not stopped. The processor to receive the request is currently running.
2	CINTR, JPLCS, JPSTART JPSTATUS, JPSTOP	Nonexistent processor. The processor ID specifies a value for a processor that does not exist.
3	JPLCS	LCS error.
4	CINTR, JPLCS, JPSTART JPSTATUS, JPSTOP	Processor failure. The processor to receive the request is not working.
5	JPFLUSH	No JP state block.
6	CINTR	Illegal option.
7	JPSTOP	Processor not running.

End of Chapter

I/O Channel Definition Register Format

The I/O channel definition register (6000_8) 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_8 indicates 1 Mbyte; 7_8 indicates 2 Mbytes. NOTE: *Systems which contain 64 Mbytes or more of physical memory return 377_8 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_8 = 32 Mbytes, 255_8 = 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
Short Format	
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:

Bit 0	Bit 31	Type
0	0	Short format only
0	1	Long format
1	0	Emulation format

Word in Block	Contents or Function
14-15	LAR, address that caused page fault.
16-17	Current execution register/Valid flag

Extensions for Long Context Block*		**Extensions for Emulation Context Block***	
18-19	General Register 0	18-19	AC1
20-21	General Register 1	20-21	ION, ATU on
22-23	General Register 2	22-23	Current execution register/Valid flag
24-25	General Register 3	24-25	Wide Stack Pointer
26-27	Current LAR	26-27	PSR:0000
28-29	DBUS Register	28-29	AC0
30-31	General Register A	30-31	Pagefault Code
32-33	General Register B	32-33	AC2
34-35	Microstack level 0	34-35	AC3
36-37	Microstack level 1	36-37	CARRY, PC
38-39	Microstack level 2	38-39	CARRY, Next PC
40-41	Microstack level 3	40-41	Scratch memory register 0
42-43	Pipe Flags	42-43	Scratch memory register 1
44-45	Current ring of execution/ALU State	44-45	Scratch memory register 2
46-47	ATU State/W/L	46-47	Scratch memory register 3
48-49	Next PC	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 (μs)
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 * levels
All levels, 32-bit instructions	1.072 + 0.201 * 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 (μs)
If any of the following faults occur	
Stack overflow/underflow	
Wide stack, type 0, 2, 4	2.211 + any indirection
Wide stack, type 1	2.613 + any indirection
Wide stack, type 3	2.402 + any indirection
Narrow stack	1.139 + any indirection
Fixed point fault	2.278 + any stack/indirection
Protection fault, typical	3.618 + any stack/indirection

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

Mnemonic	Timing (μs)*
WLMP	0.335 + 0.268/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	WEDIT	24.120 + sum of sub-op execution
	times		times
DADI	5.762	DADI	6.298
DAPS	4.221 (w/o add)	DAPS	4.489 (w/o add)
	5.561 (with add)		5.762 (with add)
DAPT	4.221 (w/o add)	DAPT	4.489 (w/o add)
	5.561 (with add)		5.762 (with add)
DAPU	5.226	DAPU	5.427
DASI	6.767 (type 4)	DASI	7.236 (type 4)
	8.040 (type 5)		8.576 (type 5)
DDTK	10.251 (branch taken)	DDTK	10.251 (branch taken)
	8.375 (branch not taken)		8.442 (branch not taken)
DEND	7.035	DEND	6.700
DICI	5.360	DICI	5.561
	+ 3.417/character inserted		+ 3.287/character inserted
DIMC	6.968	DIMC	7.102
	+ 1.608/character inserted		+ 1.548/character inserted
	+ 1.608 if j in stack		+ 1.348 if j in stack
DINC	6.365	DINC	6.432
DINS	6.700 (S=0)	DINS	6.767 (S=0)
	7.169 (S=1)		7.169 (S=1)
DINT	6.700 (T=0)	DINT	6.767 (T=0)
	7.169 (T=1)		7.169 (T=1)
DMVA	7.571	DMVA	7.705
	+ 4.690/character moved		+ 4.550/character moved
	+ 1.608 if j in stack		+ 1.348 if j in stack
DMVC	7.169	DMVC	7.303
	+ 3.752/character moved		+ 3.612/character moved
	+ 1.608 if j in stack		+ 1.348 if j in stack
DMVF	5.226	DMVF	5.494
	+ 10.787/digit moved		+ 10.587/digit moved
	+ 1.608 if j in stack		+ 1.308 if j in stack
DMVN	5.695	DMVN	5.494
	+ 8.576/digit moved		+ 8.446/digit moved
	+ 1.608 if j in stack		+ 1.348 if j in stack
DMVO	12.797/digit zero	DMVO	12.864/digit zero
	13.400/digit non zero		13.467/digit non zero
DMVS	5.226	DMVS	5.494
	+ 9.648/digit moved		+ 9.448/digit moved
	+ 1.608 if j in stack		+ 1.348 if j in stack
DNDF	4.221 (T=1)	DNDF	4.556 (T=1)
	7.303 (T=0)		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)
ADC	0.201 + 0.067 if Skip
ADC#	0.134 + 0.067 if Skip
ADCNS	0.067
ADD	0.201 + 0.067 if Skip
ADD#	0.134 + 0.067 if Skip
ADDNS	0.067
ADDI	0.067
ADI	0.067
ANC	0.067
AND	0.201 + 0.067 if Skip
AND#	0.134 + 0.067 if Skip
ANDNS	0.067
ANDI	0.067
BAM	0.201 (out of range) 0.603 + 0.335/word (number of words < 4) 0.737 + 0.335/word (number of words >= 4)
BKPT	4.355 (non-XCTed) 11.524 (XCTed)
BLM	0.134 (out of range) 0.536 + 0.268/word (# of words < 4) 0.737 + 0.268/word (# of words >= 4 and overlap of 2 or 3 words) 1.005 + 0.034 * n + (0.067 + 0.268 * m) (# words >= 4 and source + 1 = destination) n = # of words DIV 4 m = # of words MOD 4 0.737 + 0.084 * n + (0.067 + 0.268 * m) + 0.335/word of alignment required (# words >= 4)
BTO	0.603 (ACS=ACD) (1) + 0.067 if ACS< >ACD
BTZ	0.603 (ACS=ACD) (1) + 0.067 if ACS< >ACD
CLM	0.536 (ACS< >ACD, ACS < LOW) + 0.067 if ACS > HIGH + 0.067 if skip + 0.067 if ACS=ACD
CMP	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)
CMT	0.871 + 0.469/byte (AC2=AC3) 0.938 + 0536/byte (AC2<>AC3)
CMV	0.536 (destination length = 0) 0.871 + 0.201/byte (source length = 0) + 0.067 if AC1 negative
CMV (continued)	1.005 + 0.402/byte (destination descending or length(destination) < 16) + 0.067 if AC1 negative 1.072 + 0.402/byte (source descending) 0.871 + 0402/byte (length(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)
COB	1.809 (typical) 0.134 (all zeros)
COM	0.201 + 0.067 if Skip
COM#	0.134 + 0.067 if Skip
COMNS	0.067
CRYTC	0.067
CRYTO	0.067
CRYTZ	0.067
CTR	0.804 + 0.469/byte (Xlate-move) 0.804 + 0.670/byte (Xlate-compare)
CVWN	0.201
DAD	0.335
DEQUE	3.484 (deque the Head) + 0.067 if mid element + 0.201 if tail of queue 2.278 (empty queue) 3.350 (one element queue)
DERR	0.536
DHXL	0.536 0.268 (ACD & ACD + 1 = 0) 0.469 (ACD + 1 = 0)
DHXR	0.536 0.268 (ACD & ACD + 1 = 0)
DIV	2.747
DIV	0.402 (dividend = 0)
DIVS	2.747
DIVS	0.469 (dividend = 0)
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	FSGT	0.134 + 0.067 if Skip
	abs|FPACD| > abs|FPACS|)	FSLE	0.134 + 0.067 if Skip
	1.005 (SIGNs. EXPs of both are equal and	FSLT	0.134 + 0.067 if Skip
	abs|FPACD| < abs|FPACS|)	FSMD	0.536 + 0.067 if opposite signs
	1.005 (SIGNs. EXPs. MANTs of both are	FSMS	0.536 + 0.067 if opposite signs
	equal)	FSND	0.201 + 0.067 if Skip
FDD	0.436 (FPACD=0)	FSNE	0.134 + 0.067 if Skip
	8.576 (average) + 0.804 (rounding)	FSNER	0.201 + 0.067 if Skip
FDMD	0.503 (FPACD=0)	FSNM	0.201 + 0.067 if Skip
	8.643 (average) + 0.804 (rounding)	FSNO	0.201 + 0.067 if Skip
FDMS	0.536 (FPACD=0)	FSNOD	0.268 + 0.067 if Skip
	2.412 (average) + 0.469 (rounding)		0.201 (OVF = 1)
FDS	0.469 (FPACD=0)	FSNU	0.201 + 0.067 if Skip
	2.345 (average) + 0.469 (rounding)	FSNUD	0.268 + 0.067 if Skip
FEXP	0.402 (average)		0.201 (UNF = 1)
	0.268 (FPACD=0)	FSNUO	0.268 + 0.067 if Skip
FFAS	0.871		0.201 (UNF = 1)
FFMD	0.804	FSS	0.335 (opposite signs)
FHLV	0.433 (average)		+ 0.067 if same signs and FPACD <=
FINT	0.268 (exponent <= 40)		FPACS
	0.469 + 0.067 * N (40 < exponent < 48)	FSST	0.201
	n = exponent-40	FSTD	3.216
	0.536 + 0.067 * N (exponent >=48)	FSTS	3.149
FLAS	0.603 + 0.067 if negative	FTD	0.268
FLDD	0.067	FTE	0.268
FLDS	0.067	FXTD	0.067
FLMD	0.536 + 0.067 if negative	FXTE	0.134
FLST	0.268	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 (μs)	Mnemonic	Timing (μs)
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	LNDSZ	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)
		LNSTA	0.067 (1)
	3.149 + indirection + 0.268/argument	LNSUB	0.134
	Inter-ring	LOB	0.335 min
LCPID	2.144		1.340 max
LDA	0.067	LPEF	0.134
LDAF	0.067	LPEFB	0.201
LDASB	0.067	LPHY	1.474 (one level)
LDASL	0.067		2.278 (two level)
LDASP	0.067	LPSHJ	0.201
LDATS	0.067	LPSR	0.067
LDB	0.335 (1)	LPTE	1.340 (one level)
LDSP	0.737		2.077 (two level)
	0.268 (ACD < Lower)	LRB	0.536 min (ACS=ACD)
	0.335 (ACD > Upper)		1.541 max (ACS=ACD)
	0.603 value = -1		+ 0.067 if ACS<>ACD
LEF	0.067	LSBRA	13.936
LFAMD	0.402 + 0.067 if opposite signs	LSBRS	13.601
LFAMS	0.402 + 0.067 if opposite signs	LSH	0.268
LFDMD	0.503 (FPACD=0)		0.201 (ACD=0)
	8.643 (average) + 0.804 (rounding)	LSTB	0.067 (1)
LFDMS	0.536 (FPACD=0)	LWADD	0.134
	2.412 (average) + 0.469 (rounding)	LWADI	0.201 (1)
LFLDD	0.067	LWDIV	2.546 (average)
LFLDS	0.067		0.335 (Dividend = 0)
LFLST	0.201	LWDO	0.268 (for termination)
LFMMD	0.469 (FPACD=0)		0.603 (no termination)
	0.402 (FPACS=0)	LWDSZ	0.201 + 0.134 if Skip
	2.245 (average) + 0.234 (rounding)	LWISZ	0.201 + 0.134 if Skip
LFMMS	0.469 (FPACD=0)	LWLDA	0.067
	0.402 (FPACS=0)	LWMUL	0.603
	0.838 (average) + 0.067 (rounding)		0.201 (destination or source = 0)
LFSMD	0.536 + 0.067 if opposite signs	LWSBI	0.201 (1)
LFSST	0.201	LWSTA	0.067 (1)
LFSTD	0.067	LWSUB	0.134
LFSTS	0.067	MOV	0.201 + 0.067 if Skip
LJMP	0.201	MOV#	0.134 + 0.067 if Skip
LJSR	0.201	MOVNS	0.067
LLDB	0.067	MSP	0.335
LLEF	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.

(continued)

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

Mnemonic	Timing (μs)	Mnemonic	Timing (μs)
MUL	0.603	SMRF	1.809
	0.536 (AC0=0)	SNB	0.603 + 0.067 if Skip (ACS=ACD)
	0.268 (AC1=0)		+ 0.067 if ACS<>ACD
	0.335 (AC2=0)	SNOVR	0.134 + 0.067 if Skip
MULS	0.603	SPSR	0.067
	0.536 (AC0=0)	SPTE	2.814
	0.268 (AC1=0)	SSPT	0.134
	0.335 (AC2=0)	STA	0.067 (1)
NADD	0.067	STAFP	0.067
NADDI	0.067	STASB	0.335
NADI	0.067	STASL	0.134
NBStc	1.407 + 0.737 * (n-1) match found	STASP	0.067
	n= number of elements before match	STATS	0.067 (1)
	1.139 + 0.737 * (n-1) No match	STB	0.335 (1)
	n= number of elements in queue	SUB	0.201 + 0.067 if Skip
NCLID	2.010	SUB#	0.134 + 0.067 if Skip
NDIV	1.608	SUBNS	0.067
	0.469 (dividend = 0)	SZB	0.603 + 0.067 if Skip (ACS=ACD)
NEG	0.201 + 0.067 if Skip		+0.067 if ACS<>ACD
NEG#	0.134 + 0.067 if Skip	SZBO	0.737 (ACS=ACD)
NEGNS	0.067		+ 0.067 if ACS<>ACD
NFStc	1.407 + 0.737*(n-1) match found		+ 0.335 if Skip fault or Impis
	n= number of elements before match	VBP	0.402
	1.139 + 0.737*(n-1) No match		0.335 (ring < AC1 ring)
	n= number of elements in queue		0.268 (ring < CRE)
NLDAI	0.067	VWP	0.402
NMUL	0.603		0.335 (ring < AC1 ring)
	0.201 (destination = 0)		0.268 (ring < CRE) + 5.695 + 0.268/level
	0.268 (source = 0)		of indirection
NNEG	0.067	WADC	0.067
NSALA	0.201 + 0.067 if Skip	WADD	0.067
NSALM	0.335 + 0.067 if Skip	WADDI	0.067
NSANA	0.134 + 0.067 if Skip	WADI	0.067
NSANM	0.335 + 0.067 if Skip	WANC	0.067
	0.134 (immidiate = 0)	WAND	0.067
NSBI	0.067	WANDI	0.067
NSUB	0.067	WASH	0.335 (shift right or count = 0)
PATU	0.402		+ 0.067 if shift left
PBX	25.728	WASHI	0.335 (shift right or count = 0)
POP	0.335 + 0.067/word		+ 0.067 if shift left
POPB	0.603	WBLM	0.402 + 0.268/word (descending)
POPJ	0.536		0.469 + 0.268/word (ascending)
PSH	0.402 + 0.067/word		(# of words < 4)
PSHJ	0.469		0.670 + 0.268/word (# of words >= 4
PSHR	0.402		and overlap of 2 or 3 words)
RSTR	0.670		0.938 + 0.034 * n + (0.067 + 0.268 * m)
RTN	0.670		(# words >= 4 and source + 1 =
RTOD	0.737		destination)
SAVE	1.005		n = # of words DIV 4
SAVZ	1.005		m = # of words MOD 4
SBI	0.067		0.670 + 0.084 * n + (0.067 + 0.268 * m)
SEX	0.067		(# words >= 4) + 0.335/word of
SGE	0.134 + 0.067 if Skip		alignment required
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)
WBP	0.402 0.335 (ring < AC1 ring) 0.268 (ring < CRE)
WBR	0.201
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
WBTO	0.536 (ACS+ACD) (1) +0.067 if ACS<>ACD
WBTZ	0.536 (ACS=ACD) (1) + 0.067 if ACS<>ACD
WCLM	0.536 (ACS<>ACD, ACS < LOW) + 0.067 if ACS > HIGH + 0.067 if skip + 0.067 if ACS=ACD
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)
WCMT	0.871 + 0.469/byte (AC2=AC3) 0.938 + 0.536/byte (AC2<>AC3)
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 + 0402/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)
WCOB	3.417 0.134 (all zeros)
WCOM	0.067
WCST	0.536 + 0.469/byte + 0.067 if negative + 0.067 delimiter is last character in string
WCTR	0.804 + 0.670/byte (xlate - compare) 0.804 + 0.469/byte (xlate - move)
WDIV	2.546 (average)
WDIVS	2.747 (average)
WDPOP	22.244 (same ring, short context block) + 0.804 if long context block +1.139 if different ring
WFFAD	0.737
WFLAD	0.536 +0.067 if negative 0.201 (zero)
WFPOP	0.871
WFPSH	0.938
WFStc	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
WHLV	0.201
WINC	0.067
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)
			0.603 (no termination)
	3.149 + indirection + 0.268/argument inter-ring	XWDSZ	0.201 + 0.134 if Skip
		XWISZ	0.201 + 0.134 if Skip
XCH	0.201	XWLDA	0.067
XCT	8.643	XWMUL	0.603
XFAMD	0.402 + 0.067 if opposite signs		0.201 (destination or source = 0)
XFAMS	0.402 + 0.067 if opposite signs	XWSBI	0.201 (1)
XFDMD	0.503 (FPACD=0)	XWSTA	0.067 (1)
	8.643 (average) + 0.804 (rounding)	XWSUB	0.134
XFDMS	0.536 (FPACD=0)	ZEX	0.067
	2.412 (average) + 0.469 (rounding)		

(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).

ECLIPSE MV/9500™
System
Principles of Operation
014-001371 plus
014-001855-00

Cut here and insert in binder spine pocket

DataGeneral
Data General Corporation, Westboro, Massachusetts 01580