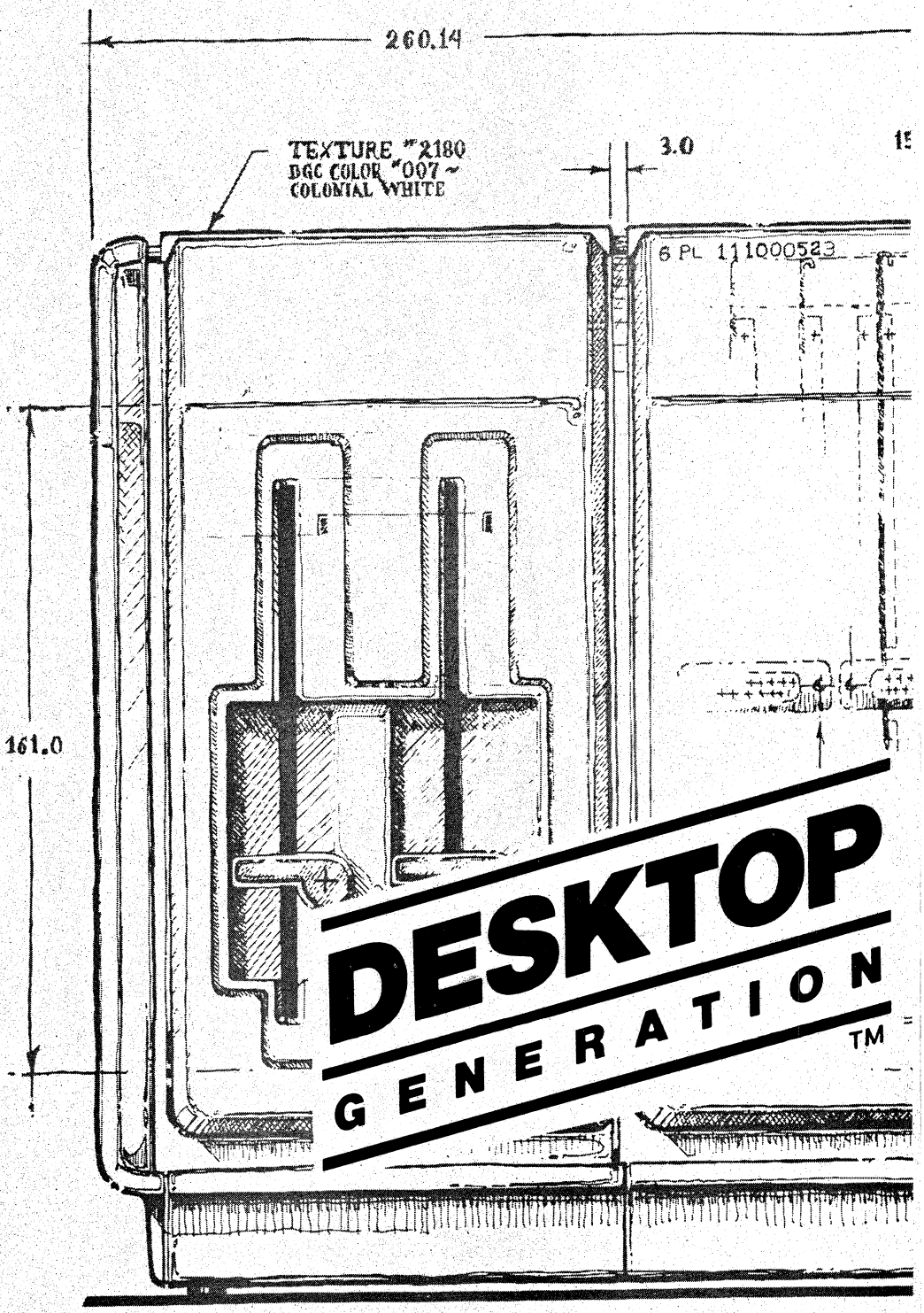
014-000766

Data General

Technical Reference



Model 10 and 10/SP Computer Systems

Notice

Ordering No. 014-000766

Printed in the United States of America
Rev. 00, October 1983

Preface

The technical reference manuals for Desktop Generation™ computers and their peripherals are written for assembly language programmers, systems analysts, and engineers. This set of manuals, together with two companion programmer's references, contains the information you need to: 1) write assembly language software, including I/O subroutines; 2) knowledgeably expand your system; 3) learn how your system operates at the card level; and 4) design custom interfaces.

This manual explains the functional and physical organization of Desktop Generation Model 10 and 10/SP computers. Other technical and programmer's references for the Desktop Generation are listed and summarily described under "Related Manuals" in this preface.

Organization

The manual has three parts: a programming section, a theory of operation section, and a mechanical assemblies section. It also has several appendixes and an index. The chapters in part 1, meant to be read selectively, present the instruction sets for system devices.

- Chapter 1 summarizes the microECLIPSE and 8086 CPU instruction sets and describes in detail those instructions that are unique to the Desktop Generation Model 10 and Model 10/SP CPU. It presents the instructions used to transfer program control between the microECLIPSE processor and the attached 8086 processor.
- Chapter 2 defines the instruction sets and tells you how to program the keyboard and monitor; printer port, real-time clock, programmable interval timer, and diskette controller.
- Chapter 3 describes how to use the firmware console program to do bootstrap loading, assist in debugging programs, and perform system resets.

The chapters in part 2 are also meant to be read selectively. They describe the makeup and operation of the component parts of a standard Desktop Generation Model 10 and Model 10/SP computer system.

- Chapter 4 describes the two-card Desktop Generation Model 10 and Model 10/SP SPU, including the microECLIPSE and 8086 processors, the main memory and MAP unit, the keyboard and monitor interfaces, the diskette interface, the printer port, real-time clock, and programmable interval timer. The chapter also discusses the floating-point option, the power-up self test, and the on-card virtual console.
- Chapter 5 describes the theory of operation of the optional dynamic random-access memory card.
- Chapter 6 discusses the operation of an optional color video interface card.
- Chapter 7 provides a description of the Desktop Generation power supply.

The chapter in part 3 illustrates the mechanical components of the Model 10 and 10/SP computer system.

- Chapter 8 contains a description of the physical modules and their configurations as well as the system cabling scheme.
- Appendix A lists the numbers of available logic schematics and wiring lists.
- Appendix B summarizes the diagnostic assembly language instructions for the diskette.
- Appendix C presents a listing of some typical assembly language code used to manage the 8086 processor from a microECLIPSE program.
- Appendix D explains the format in which words must be written into the monochrome monitor screen buffer in order for the data to be displayed on the screen.
- The index alphabetically lists the concepts and terms in this book and references the pages on which they appear.

A documentation comment form follows the index. It invites you to help Data General improve its publications by commenting on this book.

vorhanden

Related Manuals

A comprehensive documentation set supports all the hardware and software products available for Desktop Generation computers. The hardware-related books listed below fall into three categories: the technical reference series; the user guides for operating, installing, and testing; and the introductory guide for Desktop Generation computers.

The following technical and programmer's references address the needs of assembly language programmers and engineers.

16-bit Real Time ECLIPSE Assembly Language Programming

Global in nature, this book explains the processor-independent concepts, functions, and instruction sets of 16-bit ECLIPSE computers. DGC ordering no. 014-000688.

Model 10 and 10/SP System Console
Programmer's Reference

Describes the organization and alphanumeric and graphic features of the system console. Defines the command sets and includes guidelines for programming the monochrome and optional color monitors at assembly and high-level language levels. DGC ordering no. 014-000770.

Model 20 and 30 Computer Systems
Technical Reference

In addition to the functional and physical organization of Model 20 and 30 computers and their technical specifications, this manual explains their processor-unique concepts, functions, and instruction set features. Also included are guidelines for programming the I/O devices, including the diskette subsystem, and a theory of operation for the basic components of Models 20 and 30. DGC ordering no. 014-000767.

I/O and Interfacing
Technical Reference

Introduces the microI/O bus and describes the I/O interface required to communicate with this bus and its host Desktop Generation computer. Discusses the I/O instruction set and the I/O program interrupt and data channel facilities. Includes a chapter about the 4210 general-purpose interface, useful to those designing a custom I/O interface for their system. DGC ordering no. 014-000774.

For more detailed information about the microI/O bus and Data General integrated circuits used in the I/O interface, refer to *microNOVA Integrated Circuits Data Manual.* DGC ordering no. 014-000074.

Model 6271 Disk Subsystem
Technical Reference

Describes the functional and physical organization of the Model 6271 disk subsystem. Defines the I/O instruction set and provides guidelines for programming the subsystem. DGC ordering no. 014-000768.

Communications Interfaces
Technical Reference

Discusses the functional and physical organization of the asynchronous/ synchronous communications interfaces available for Desktop Generation computers. Defines their I/O instruction sets, offers guidelines for writing assembly language I/O subroutines, and contains theory of operation for each communications card. DGC ordering no. 014-000769.

Sensor I/O
Technical Reference

Defines instruction sets, offers guidelines for writing assembly language I/O subroutines, describes theory of operation at an overview level, and explains how to connect field wiring for the 4222 digital I/O interface, 4223 analog-to-digital interface, 4224 digital-to-analog interface, and 4335 analog subsystem. DGC ordering no. 014-000775.

IEEE-488 Bus Interface
Technical Reference

Provides the information needed to interface, program in assembly language, and troubleshoot this card in a Desktop Generation system. Reviews the contents of the IEEE-488 bus standard, summarizing its commands, messages, and states, and includes a theory of operation. DGC ordering no. 014-000773.

The following books are how-to manuals written for anyone who needs to know how to install, operate, and test a Desktop Generation system.

Installing Model 10 and 10/SP Systems

The first book that a Model 10 or 10/SP owner should read, explains how to unpack and install either system and its optional peripherals. Simple instructions and ample illustrations make the book accessible to any reader. DGC ordering no. 014-000901.

Operating Model 10 and 10/SP Systems

A logical follow-on to Model 10 and 10/SP installation, this guide takes you from powering up the system and its optional peripherals through performing such routine operations as loading paper in a printer and inserting or removing diskettes. Brings you to the point of loading the system software. Amply illustrated and written for users at any level of experience. DGC ordering no. 014-000900.

Testing Model 10 and 10/SP Systems

Follows the installation and operating manuals with instructions for verifying the operation of Model 10 or 10/SP systems and their optional peripherals. Steps you through the power-up test and Customer Diagnostics and explains how to troubleshoot customer-replaceable components. Simple instructions and diagrams make the book accessible to any user. Includes phone numbers for Data General assistance. DGC ordering no. 014-000902.

Installing Model 20 and 30 Systems

The first book a Model 20 or 30 owner should read, explains how to unpack and install either system and its optional peripherals. Accessibly written and illustrated, for users at any level of experience. DGC ordering no. 014-000904.

Operating Model 20 and 30 Systems

Follows Model 20 and 30 installation, leading you from powering up the system and its optional peripherals through performing such routine operations as loading paper in a printer and inserting or removing diskettes. Brings you to the point of loading the system software. The simple instructions and generous illustrations are suitable for any reader. DGC ordering no. 014-000903.

Testing Model 20 and 30 Systems

A follow-on to the installation and operating manuals, explains how to verify the operation of Model 20 or 30 systems and their optional peripherals. Simple instructions and diagrams lead you through the power-up test, Customer Diagnostics, and trouble-shooting of customer-replaceable components. Includes phone numbers for Data General assistance. DGC ordering no. 014-000905.

This last book is a product overview, addressed to all Desktop Generation users.

The Desktop Generation

Introduces the Desktop Generation, summarizing each model of the family, and describes its many hardware and software products, features, and capabilities. Includes a brief history of Data General, a sampling of applications, and an overview of the customer service and support programs available to you as a Desktop Generation user. DGC ordering no. 014-000751.

Conventions

The following conventions are used throughout this manual.

MNEMONIC	Uppercase sans serif letters indicate a signal name or instruction mnemonic. When a signal is active low, it is barred—for example, $\overline{\text{FDCHE}}$.
argument	Italicized lowercase letters mean that a particular instruction takes an argument. In your program, you must replace this symbol with the exact code for the argument you need.
[optional]	Brackets signify an optional argument. If you decide to use this argument, do not include the brackets in your code; they only set off the choice.

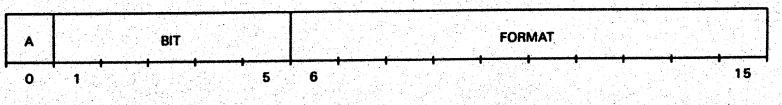
In dialogs between system and user, we use this typeface to show your input:

USER INPUT

and this typeface to show the system's response:

SYSTEM RESPONSE.

In addition, we use the following diagram to show the arrangement of the 16 bits in an instruction. The diagram is always divided into 16 boxes, numbered 0 through 15.



Contents

THREE Mechanical Assemblies

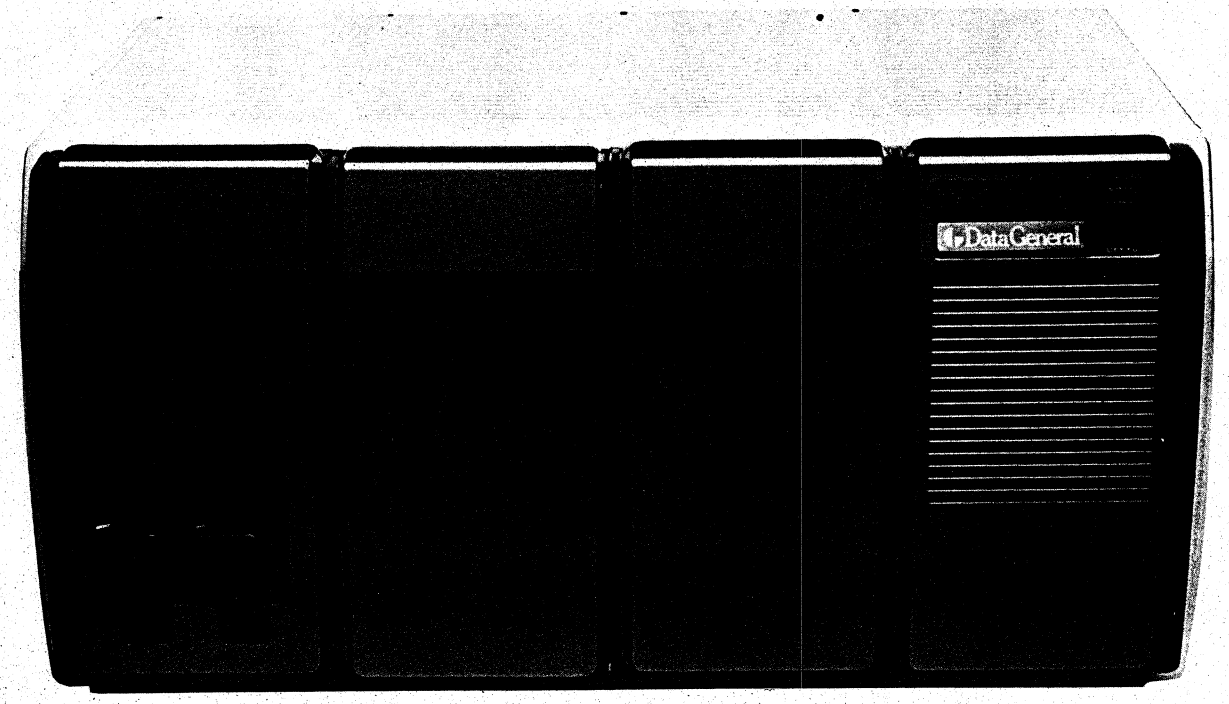
8 Model 10 and Model 10/SP Modules and Configurations

A Related Drawings

B Diskette Diagnostic Commands

C ATP Interface Programming Example

D Memory Formats for Bit-Mapped Screen Buffer



PH-0726

Figure 0-1 Models 10 or 10/SP computer systems

Overview

Model 10 and Model 10/SP systems are dual-processor desktop computers designed for multi-user commercial and technical applications. Used as technical workstations or applications in business, education, science, real-time control, and industrial automation arenas, they offer minicomputer power in a desktop unit.

This chapter provides an overview of the Model 10 and Model 10/SP Desktop Generation computer systems. It discusses their configurations and functional organization and briefly describes their major components, including: the

system processors, their memories, the diskette and Winchester disk subsystems, the power subsystem, the system console (display monitor and keyboard), and the optional cartridge tape subsystem, communications multiplexors, and input/output (I/O) interfaces. System technical specifications conclude the chapter.

Configurations

The Model 10 and Model 10/SP systems are identical to each other except that the Model 10/SP is configured with a firmware floating point feature to support AOS and MP/AOS-SU operating systems. The systems use a common set of modular building blocks, identical in size, that interconnect for ease of installation and system expansion. The set consists of the following modules:

power module

5-slot CPU logic module

diskette module

disk module

5-slot logic expansion module

cartridge tape module

Figure 0-2 shows 3-, 4-, 5-, and 6-module configurations. Minimum configuration systems consist of a system console and a computer unit comprised of the three basic modules: power module, CPU logic module, and diskette module. Other modules can be added to expand upon the system's capabilities.

1. Power module, containing:

   One power supply (if a 3-module system); or
   Two power supplies (if more than 3 modules in system)
   Cooling blower
   Line frequency clock generator card (optional)

2. CPU logic module, containing:

   Two-board system processor unit featuring dual CPUs, basic system memory, and interface logic for system console, serial printer, and diskette subsystem
   256 Kbyte or 512 Kbyte semiconductor memory cards (optional) with byte parity (for a maximum system memory capacity of 768 Kbytes)
   Color monitor interface board (optional)
   One or more (as space permits) I/O interface cards (optional)

3. Diskette drive module, containing:

   One or, optionally, two 5.25-inch diskette drives

4. Disk module, containing:

   Disk controller card
   One 5.25-inch Winchester disk drive

5. Disk expansion unit, consisting of:

   Disk module (less the controller)
   Power module with one power supply and a cooling blower.

6. Cartridge tape module, containing:

   Controller card
   5.25-inch cartridge tape drive
   Power supply

7. 5-slot logic expansion module that accommodates up to five I/O interface cards

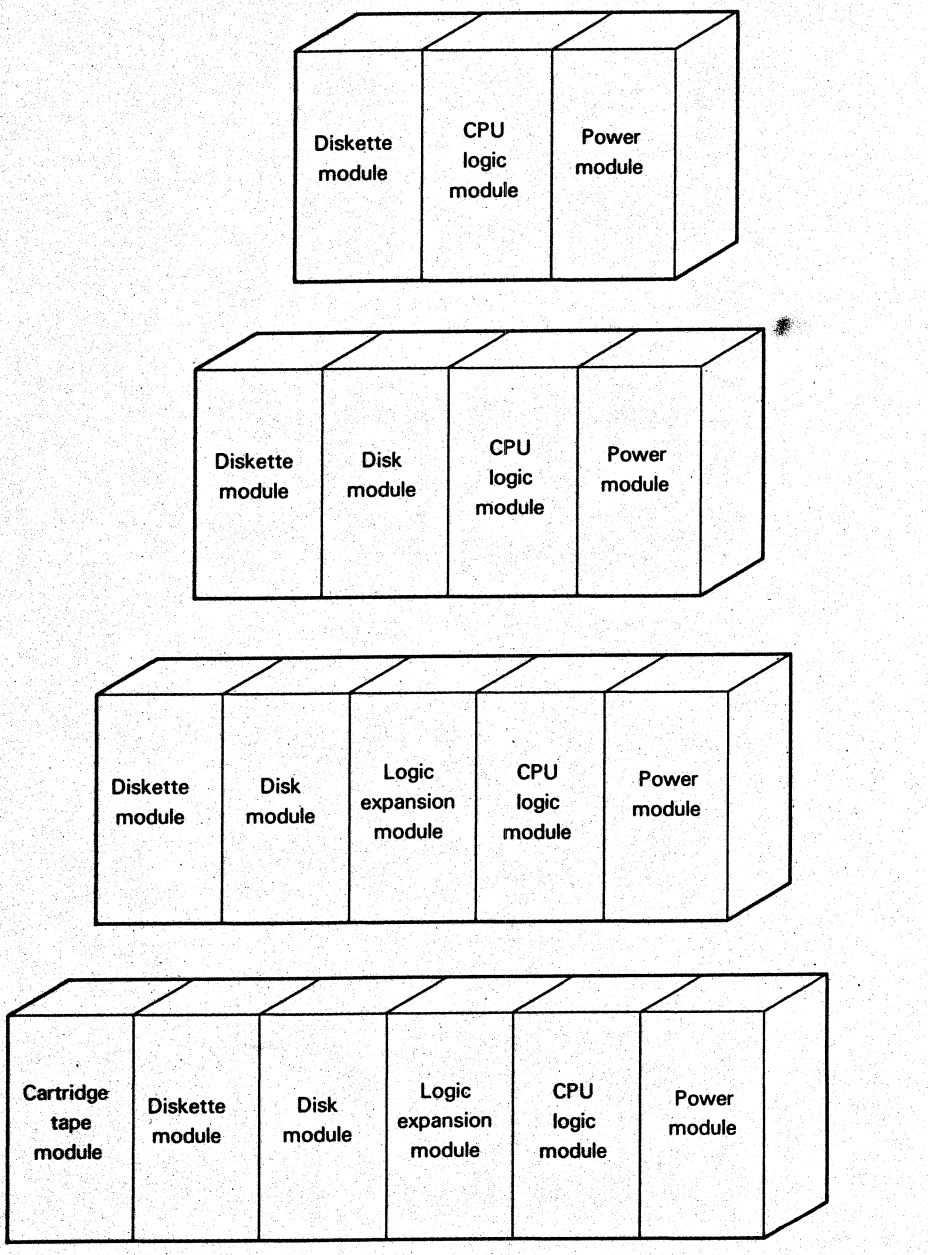


Figure 0-2 Models 10 or 10/SP system configuration

Organization

As shown in Figure 0-3, the two system processors (microECLIPSE and 8086) are organized around two major system buses: the memory bus and the microI/O bus. The memory bus provides a 16-bit wide, memory address/data path (address path expands to 20 bits wide in memory mapped mode) between the system processors and the memories, and the optional firmware floating point unit, when present. The microI/O bus (sometimes called the Microproducts or microNOVA I/O bus) consists of sixteen lines, four of which provide a differentially driven, 2-bit serial data path between the microECLIPSE processor and the I/O subsystems.

Components

The components of the Model 10 and Model 10/SP systems are described below.

System Processing Unit

The two-board System Processing Unit (SPU) resides in slots 1 and 2 of the SPU logic module. These boards contain two CPUs: a microECLIPSE processor and an 8086 microprocessor.

The microECLIPSE processor implements the 16-bit ECLIPSE character instruction set and supports Data General's RDOS, AOS, and MP/AOS-SU operating systems. It also supports bit-mapped graphics for the monochrome monitor and gives direct memory access (DMA) to graphics applications for rapid screen response. The microECLIPSE CPU can be supplied with or without a firmware floating point instruction set.

The 8086 microprocessor implements the Intel 8086 microprocessor instruction set, which supports the the MS-DOS and CP/M-86 operating systems.

The two SPU boards also contain:

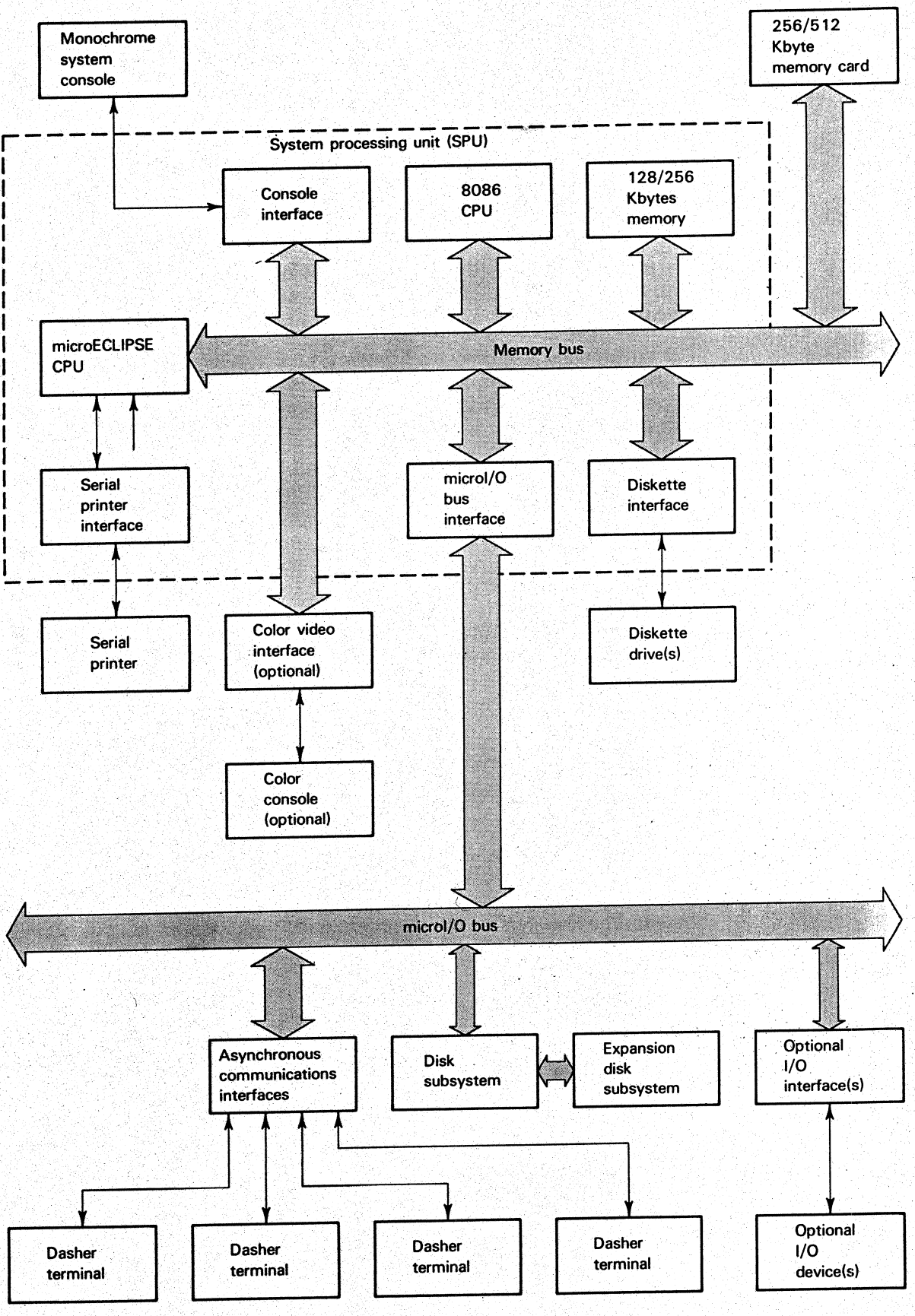
- Basic system memory (128 kbytes or 256 kbytes)
- Memory allocation and protection unit (MAP)
- System console interface for the system console monitor and keyboard
- Diskette interface
- Serial printer interface (EIA RS-232-C compatible)
- Virtual console
- Real-time clock
- Programmable interval timer
- Parity checking logic for memory data
- Power status monitor
- Power-up diagnostics
- Program load logic

The two-board SPU provides basic, low-address RAM for system memory. It can be configured with either 128 Kbytes or 256 Kbytes of storage. The memory

arrays are structured with 64K by 1-bit elements of dynamic MOS random-access memory (RAM). The memory includes byte parity bits for data.

The parity checking logic of the SPU verifies the integrity of system memory. It appends a parity bit to each byte of data written to memory and checks it when it is read. When the parity checking logic is enabled by the program, a parity error generates a program interrupt.

The MAP performs logical-to-physical address translation for the system memory. It also provides the following protection mechanisms: validity, write, I/O and indirection.


Monochrome system console
256/512 Kbyte memory card
System processing unit (SPU)
Console interface
8086 CPU
128/256 Kbytes memory
microECLIPSE CPU
Memory bus
Serial printer interface
micro I/O bus interface
Diskette interface
Serial printer
Color video interface (optional)
Diskette drive(s)
Color console (optional)
micro I/O bus
Asynchronous communications interfaces
Disk subsystem
Expansion disk subsystem
Optional I/O interface(s)
Dasher terminal
Dasher terminal
Dasher terminal
Dasher terminal
Optional I/O device(s)

ID-00715

The system console interface port on the System Processing Unit communicates with the system console monitor and keyboard. For the optional 13-inch color monitor, a separate graphics controller board is used. Additional workstations for multiterminal operation are connected via the asynchronous communications interfaces (see the subsections "I/O Interfaces" and "Multiterminal workstations").

The diskette interface, which resides on the two-board SPU, can be programmed to read and write Data General formatted diskettes (9 sectors per track) or CPM and IBM PC formatted diskettes (8 sectors per track). The controller can also be programmed to read Data General's ENTERPRISE/MPT™ 5.25 inch, formatted diskettes (35 tracks per surface and 10 sectors per track).

The printer interface port on the SPU provides asynchronous communications with a connected serial printer. The interface conforms to EIA RS-232-C specifications.

The virtual console provides a firmware substitute for front panel switches and indicators, allowing the operator at the system console to program load; start, stop and continue program execution; and perform program debugging operations.

The real-time clock and programmable interval timer (PIT) provide time bases for programs that require them. The real-time clock generates low-frequency, I/O interrupts at one of the following program-selectable rates: 10 Hz, 100 Hz, 1,000 Hz, or ac line frequency. The PIT can be programmed to generate I/O interrupts at fixed intervals ranging from 100 microseconds to 6.5536 seconds. The clock rate of the PIT is 10 KHz.

The power status monitor tracks the state of a power status signal supplied by the power supply and initiates a power fail interrupt whenever power falls outside specified limits.

The power-up diagnostic — a thirty- to forty-five second self-test routine — verifies the basic integrity of the system each time power is applied. It checks all of memory, the virtual console, the CPU, the system console interface, the attached processor, the MAP units, the printer port, and the color video interface, if present.

The program-load logic — this feature automatically transfers a low-level bootstrap program from the diskette drive and is initiated after a successful completion of the power-up self-test diagnostics if no disk drive is present in the system. The low-level bootstrap program is then executed to initiate the transfer of subsequent program elements of the operating system.

Diskette Subsystem

The diskette module contains one or, optionally, two drives that reside in the diskette module. Each drive stores and retrieves data from a 5.25 inch, double-sided, low-track density (48 tracks per inch) diskette containing 40 tracks per surface. There can be 8 to 10 sectors per track, depending on the type of diskette.

Each sector stores 512 bytes of data. The diskette interface uses direct memory accessing to transfer 8-bit data bytes between memory and the subsystem.

Disk Subsystem

The optional disk subsystem consists of a disk controller and up to two 5.25-inch, Winchester disk drives with a formatted storage capacity of 15 megabytes per drive.

The controller and one disk drive reside within the disk module of the main computer unit. The second disk drive, when present, resides in an expansion unit, consisting of a disk module and a power module with one power supply assembly. An external device cable connects the second disk drive to the subsystem controller.

Each drive stores and retrieves data from 5.25-inch, double-sided, fixed disk platters, containing 306 tracks per surface — 305 tracks for the user and one for diagnostics. Each track contains 17 sectors and each sector stores 512 bytes of data. The controller provides a sector buffer for data transfers and uses the microI/O bus and data channel facility of the SPU to transfer 16-bit data words between memory and the subsystem.

Video Interface

An optional video interface board can be added for a 13-inch color graphics monitor. The video interface board will reside in the slot next to SPU2 in the CPU logic module.

System Console

The system console consists of a 12-inch monochrome monitor and keyboard. It allows the user to direct the activities of the system. This console connects to interface logic on the SPU. If an optional color video interface board is included, a 13-inch color monitor can be substituted for the monochrome monitor in the system console.

Multiterminal Workstations

Up to four additional terminals can be connected for multiterminal operation. These terminals connect directly to any of the optional asynchronous communications interfaces: Models 4463-Z, 4463-W. Any of the following DASHER™ model terminals may be used: D210, D211, G300, D410, and D460.

Cartridge Tape Subsystem

The optional cartridge tape subsystem consists of a controller, a 1/4 inch magnetic tape cartridge drive, a fan, and a power supply. The subsystem provides a storage capacity of up to 15.4 Mbytes. The controller uses the microI/O bus and standard data channel facility of the CPU to transfer 16-bit data words between memory and the subsystem.

I/O Interfaces

The Model 10 and Model 10/SP support a selection of asynchronous/synchronous communications multiplexors and sensor I/O subsystems. Each card occupies an I/O slot in either the CPU logic module or the 5-slot expansion module and communicates with the CPU via the microI/O bus.

The asynchronous communications facilities provide an EIA-RS232C or 20 MA current-loop line interface, jumper-selectable, for the several DASHER™ display terminals available as additional workstations.

Power Subsystem

The power subsystem consists of one or two, 123-watt (output), power supply assemblies and a cooling blower inside the power module. The power module also houses an optional line frequency clock generator card. One power supply assembly provides dc power to the CPU logic module and diskette subsystem while the second power assembly supports the disk module and optional 5-slot expansion module.

TECHNICAL SPECIFICATIONS

Table 0-1 through Table 0-5 list general specifications as well as mechanical, electrical, and environmental specifications for the Model 10 and Model 10/SP computer systems.

Table 0-1 General specifications

Unit	Description
Model 10 System	Minimum 3-module system contains: CPU logic module which System includes dual-CPU system processing unit and 128 Kbytes of system memory, a diskette module with one 5.25-inch diskette drive, and a power module containing one power supply and a cooling blower. Additions to the minimum configuration can include up to 512 Kbytes of additional memory, a second diskette drive (housed in same module as the first drive), a disk module containing a 15-Mbyte Winchester disk subsystem (this also requires the addition of a second power supply unit in the power supply module), and a cartridge tape module with its own internal power supply. A second disk drive can also be configured and is housed in a separate two-module unit: a disk module and power supply module with one power supply and cooling blower.
Model 10/SP System	This configuration supports AOS and MP/AOS-SU operating systems. Includes the same features as the Model 10 *and* has a firmware floating point option.
System Console	Each system has one 12-inch monochrome display terminal with a keyboard that are used for system operator functions as well as user applications. A 13-inch color monitor is optional.

Table 0-2 General specifications, basic components

Unit		Description
System Processing Unit (SPU)	System location	CPU logic module, slots 1 and 2.
microECLIPSE Central Processor	Instruction set	ECLIPSE instruction set; Model 10/SP includes firmware floating point.
	Instruction length	16 and 32 bits
	Number of accumulators	4 fixed point (2 usable as index registers) 4 floating point
	Accumulator length	Fixed point, 16 bits; floating point, 64 bits
	Priority interrupt levels	16 programmable
	Maximum data channel rates	
	input	267 Kbytes/second
	output	337 Kbytes/second
	Memory allocation and protection unit	Supports up to 2 Mbytes of physical memory with: 4 user maps 1 data channel map write protection validity protection I/O protection indirect protection
8086 Central Processor	Instruction set:	Intel 8086 instruction set
	Accumulators (multifunction registers)	Eight general-purpose 16-bit; accessed as either 8-bit bytes or 16-bit words.
	Interrupts	Single interrupt, initiated by microECLIPSE processor.
	Memory	Uses system memory shared with microECLIPSE processor.
	Input/Output	Serviced by microECLIPSE processor upon request by the 8086 processor.
System Console Interface	Full-duplex, serial keyboard and monitor interface.	
Printer Port	Full-duplex, 9600-baud asynchronous EIA-RS232C interface.	
Real-time Clock	Initiates I/O interrupts at one of the four frequencies: 10 Hz, 100 Hz, 1,000 Hz, and ac line frequency. Frequency is program-selectable.	
Programmable Interval Timer (PIT)	Initiates I/O interrupts at fixed intervals ranging from 100 microseconds to 6.5536 seconds, in 100 microsecond increments. Clock rate is 10 KHz.	
System Memory	System location: CPU logic module. Up to 256 Kbytes on SPU boards (slots 1 and 2). Optional memory board is located in slot 3.	
	Type: semiconductor MOS.	
	Optional board available with either 256 Kbytes or 512 Kbytes with byte parity.	
	Memory configurations: minimum 256 Kbytes; maximum 768 Kbytes.	

Diskette Subsystem	System location: diskette module. Consists of a controller printed circuit board and one or, optionally, two 5.25 inch, double-sided, 48 TPI, diskette drives with a formatted capacity of 368 Kbytes per drive. Supports the following diskette formats: * Data General 9 sectors*/track (read and write) * IBM or CPM 8 sectors*/track (read and write) * Data General ENTERPRISE/MPT 10 sectors*/track (read only)
Power Subsystem	System location: power module. Power supply type: off-line switching converter. Number of power supply assemblies: one for basic 3-module system; two for 4-module or larger system. d.c. power supplied by each power supply: +5V 16.3 amps −5V 0.5 amps +12V 2.5 amps −12V 0.8 amps

**512 bytes per sector*

Table 0-3 Mechanical specifications

Unit	Width	Depth	Height	Weight
Systems				
Basic 3-module system with covers	15.7 in 39.9 cm	13.0 in 33.0 cm	10.7 in 27.2 cm	39.5 lbs 17.9 kg
Four-module system with covers and disk module	20.5 in 52.0 cm	13.0 in 33.0 cm	10.7 in 27.2 cm	52.5 lbs 23.8 kg
Four-module system with covers and logic expansion unit	20.5 in 52.0 cm	13.0 in 33.0 cm	10.7 in 27.2 cm	50.0 lbs 22.7 kg
Five-module system with covers, disk module, logic expansion unit	25.2 in 64.0 cm	13.0 in 33.0 cm	10.7 in 27.2 cm	64.5 lbs 29.3 kg
Six-module system with covers, disk module, logic expansion unit, and cartridge tape module	30.0 in 76.2 cm	13.5 in 34.3 cm	10.7 in 27.2 cm	80.0 lbs 36.3 kg
Modules (without covers)*				
Power module with 1 power supply and cooling blower	4.8 in 12.2 cm	12.8 in 32.5 cm	10.7 in 27.2 cm	13.9 lbs 6.3 kg
Power module with 2 power supplies, cooling blower, and line frequency clock generator card	4.8 in 12.2 cm	12.8 in 32.5 cm	10.7 in 27.2 cm	18.7 lbs 8.5 kg
CPU logic module with model 20 SPU and 1 memory card	4.8 in 12.2 cm	12.8 in 32.5 cm	10.7 in 27.2 cm	8.0 lbs 3.6 kg
Diskette module with controller and 1 drive	4.8 in 12.2 cm	12.8 in 31.8 cm	10.7 in 27.2 cm	11.5 lbs 5.2 kg
Diskette module with controller and 2 drives	4.8 in 12.2 cm	12.8 in 32.5 cm	10.7 in 27.2 cm	14.8 lbs 6.7 kg
Disk module with controller and 1 drive	4.8 in 12.2 cm	12.8 in 32.5 cm	10.7 in 27.2 cm	14.5 lbs 6.6 kg
Logic expansion unit (empty)**	4.8 in 12.2 cm	12.8 in 32.5 cm	10.7 in 27.2 cm	6.5 lbs 3.0 kg
Cartridge tape module with controller and 1 drive	4.8 in 12.2 cm	13.5 in 34.3 cm	10.7 in 27.2 cm	15.5 lbs 7.0 kg
Disk expansion unit with 1 drive and 1 power supply	10.9 in 27.7 cm	12.8 in 32.5 cm	10.7 in 27.2 cm	28.5 lbs 12.9 kg

**Each side cover adds 0.7 inch (1.8 cm) to width.*

***Each circuit card adds 0.75 lb (0.3 kg) weight.*

Table 0-4 Electrical specifications

Power Requirements	
Voltage:	100 volts ac, +/– 10%lb 120 volts ac, +10%, –15%lb 240 volts ac, +10%, –15%
Line frequency:	47-63 Hz
Line current (maximum)	
Unit with 1 power supply*:	3 amperes @ 120 volts ac
Unit with 2 power supplies*:	6 amperes @ 120 volts ac
Power consumption (maximum)	
Unit with 1 power supply*:	180 watts
Unit with 2 power supplies*:	360 watts

**Includes blower unit.*

Table 0-5 Environmental specifications

Temperature ranges	
Operating:	0 to 38 degrees C 32 to 100 degrees F
Storage:	–40 to 65 degrees C –40 to 149 degrees F
Relative humidity	
Operating:	20% to 80%, noncondensing
Storage:	10% to 90%, noncondensing
Altitude	
Operating:	0 to 2450 m 0 to 8000 feet
Storage:	0 to 7620 m 0 to 25000 feet

Part

ONE

Programming

Programming the CPUs

1

At the heart of the Desktop Generation Model 10 and Model 10/SP systems are the two CPU cards in the CPU logic module. These cards contain two system processing units: a Data General microECLIPSE processor and an Intel 8086 processor which access memory, manage data, and control program flow.

This chapter summarizes the logical relationship between the microECLIPSE and 8086 processors and the programming capabilities of each. The summaries include brief descriptions of important programming features such as addressing modes, data formats, program accessible registers and SPU operations and, also, tables of both instruction sets.

The summary descriptions are followed by details of programming the processors that are unique to their use in Model 10 and 10/SP systems. This includes power-up response, microECLIPSE instructions, registers, memory management facilities, a list of reserved memory locations, and the set of instructions used to pass control between the 8086 and the microECLIPSE CPUs that are not described in the *16-Bit Real-time ECLIPSE Assembly Language Programming* manual. The chapter concludes with tables of instruction execution times for both processors.

Figure 1-1 shows the relationship between the microECLIPSE and 8086 processors and the rest of the Model 10 or 10/SP system. The microECLIPSE is the kernal processor: it manages system memory allocation and protection, all input/output operations, and begins running upon power-up or after a system reset.

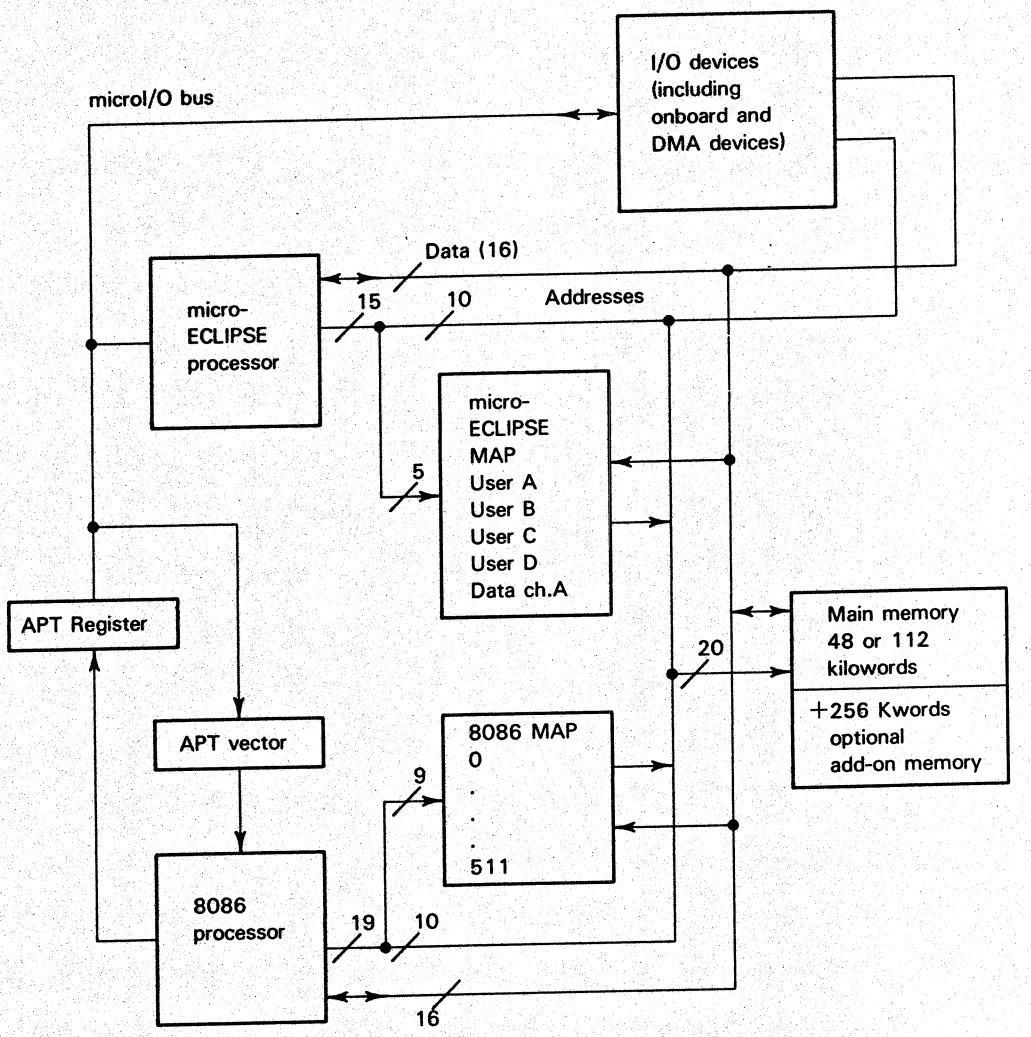


Figure 1-1 Dual-processor organization

The Model 10 and 10/SP Dual-Processor System

The 8086 processor shares main memory with the microECLIPSE CPU. Only one processor can access the memory at a time, so the processors run serially — that is, while one processor runs, the other is idled. Interprocessor communications are handled by a combination of 8086 processor calls (performed by executing any *Out* instruction on the 8086), by 8086 vectored interrupts, and by shared memory mailboxes. Software determines the location and format of the mailboxes.

The operational cycles of the two processors are synchronized so that their memory accesses occur during the same relative timing phase: a memory access for either processor requires 500 ns. The instruction execution times for each processor are listed in the sections on the individual processors.

Since the memory allocation and protection unit (MAP) and all I/O devices are under the control of the microECLIPSE processor, the 8086 must make requests to the microECLIPSE CPU via the memory mailboxes if the 8086 program requires a change of the MAP or an access to an I/O device.

All system I/O interrupts are handled by the microECLIPSE processor. If any interrupt occurs while the 8086 processor is running, the 8086 is paused, and control passes immediately to the microECLIPSE CPU. After the microECLIPSE handles the interrupt, it normally restarts the 8086 processor.

In order to write programs for the Data General microECLIPSE processor, refer to the *16-Bit Real-Time ECLIPSE Assembly Language Programming* (DGC No. 014-000688); for the Intel 8086 processor, you need the *MCS-86 User's Manual* (Intel No. 92-722). These documents detail the instruction sets and architecture of the system processing units.

You will need to supplement these references with specifics in this chapter on operations peculiar to Model 10 and 10/SP systems, such as execution times and the transfer of control between processors.

System Memory Mapping

As shown in Figure 1-1, each processor has its own MAP unit. The microECLIPSE MAP translates 15-bit logical addresses into 20-bit physical addresses and the 8086 MAP translates 19-bit logical addresses into 20-bit physical addresses. (Model 10 and 10/SP systems use only the first 112 kilowords or the first 368 kilowords of the 1024 kilowords of physically addressable space.) The mapping operation translates 1-kiloword logical blocks (called *pages*) to 1-kiloword physical blocks. The blocks do not have to be mapped contiguously, and microECLIPSE and 8086 blocks can be interspersed among each other. For details on each MAP unit, read the sections on the individual processors.

Parity Checking

The Model 10 and parity checking facility, when enabled, detects any byte parity error or errors that might occur during access to main memory. Each memory word consists of 18 bits: 16 data bits (2 bytes) and 2 parity bits (1 for each byte). The parity logic checks the parity bits read from memory. If an error occurs, the parity logic requests an interrupt, when so enabled. The microECLIPSE processor must handle this interrupt. Check the section on that processor for additional information.

microECLIPSE CPU Features

This section summarizes the programming capabilities of the microECLIPSE processor as it is used in a Model 10 or 10/SP system. It includes brief descriptions of important programming features such as addressing modes and data formats, and explains microECLIPSE CPU operations in general terms. Note that the floating point instruction set is an optional feature, included only with Model 10/SP systems.

A microECLIPSE CPU features the standard Data General architecture. This architecture is diagrammed in Figure 1-2.

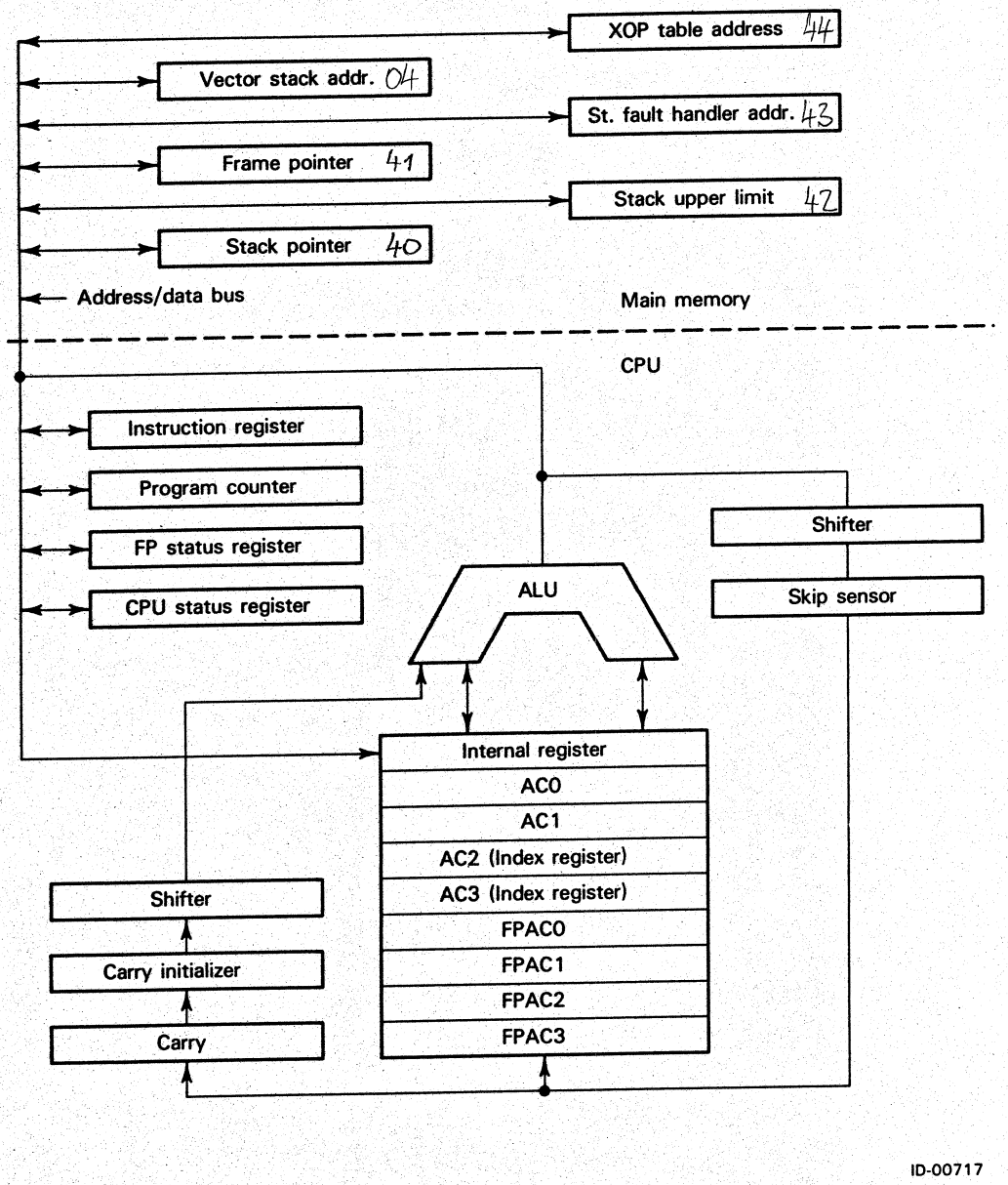


Figure 1-2 ECLIPSE architecture

Addressing

The size of the logical address space of the microECLIPSE CPU is 64 Kbytes. The physical address space of the Model 10 or 10/SP system can be as large as 768 Kbyte. Refer to the section entitled "MAP Operations" for a summary of logical-to-physical address translation.

The microECLIPSE processor has two classes of instructions. *Short class* instructions contain an 8-bit address displacement. *Extended class* instructions contain a 15-bit address displacement. Both classes of instructions use one bit to specify either direct or indirect addressing.

In addition, indirect addressing can be specified by a bit within the contents of

an address. (If bit 0 of an addressed word is one, the addressed word is used as a pointer to another address.)

The microECLIPSE processor permits any number of indirection levels; in mapped mode, however, indirections can be limited to 15 levels. (See the "MAP Operations" section later in this section.)

Addressing Modes

Direct or indirect word addressing in the microECLIPSE CPU can be done in the following modes:

Absolute. The address (before indirection) is the unmodified displacement, that is, it is the page 0 address as shown in Figure 1-3.

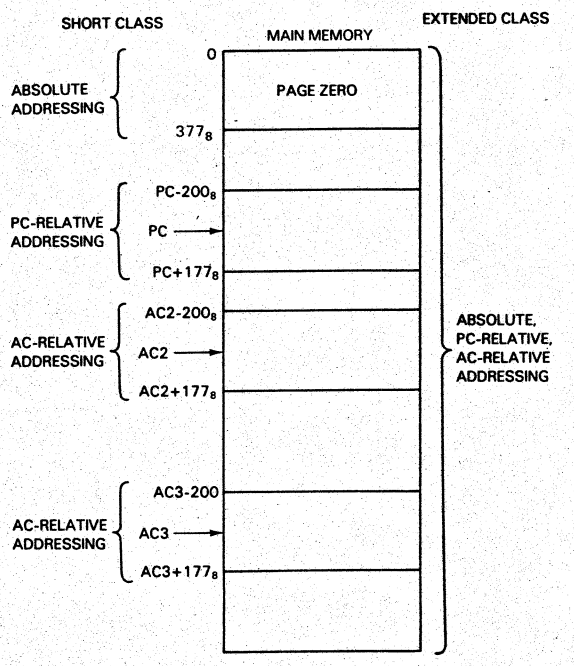


Figure 1-3 Addressing modes

Program Counter Relative. The address (before indirection) is found by adding the displacement to the address of the word containing the displacement, that is, the current instruction.

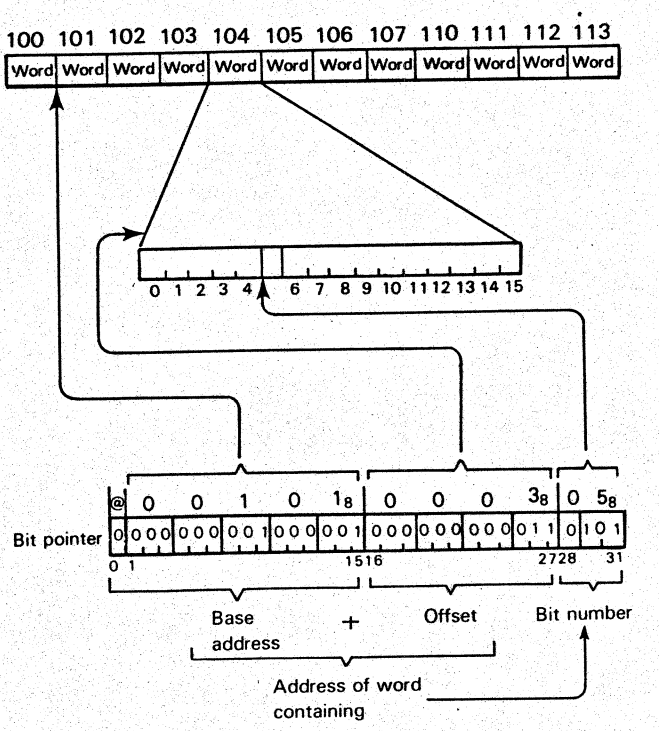
Accumulator Relative. The address (before indirection) is found by adding the displacement to the contents of a specified accumulator (AC2 or AC3).

Figure 1-3 illustrates the accessible memory ranges for the two instruction classes and three addressing modes (direct addressing). Note that absolute addressing mode can be used to access lower page zero, locations 0-377_8, regardless of the current contents of the program counter.

Byte Addressing

A byte in memory is selected by a 16-bit byte pointer. Bits 0-14 of this pointer contain the memory address of a 2-byte word. Bit 15 indicates which byte of the address location will be used. Short class instructions use an accumulator to hold the byte pointer. The pointer is contained in the displacement field of an extended class instruction.

Bit Addressing A bit in memory is selected by a bit pointer. Instructions that require this 32-bit pointer use two accumulators (specified in the instruction) to hold the pointer. Figure 1-4 illustrates the bit addressing process.

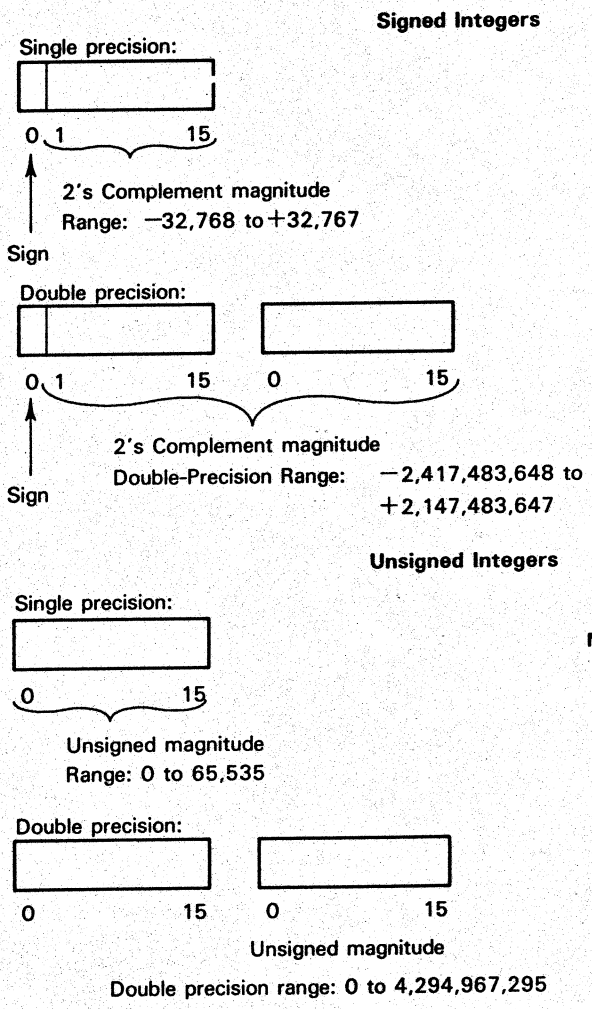


DG-08290

Figure 1-4 Bit pointer

Data Formats

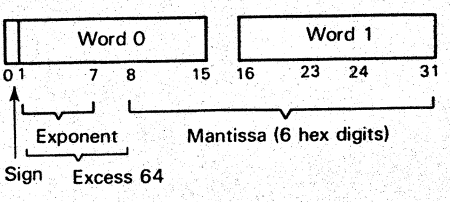
This subsection summarizes integer formats in Figure 1-5 floating point formats in Figure 1-6. Floating-point numbers are normalized at the end of all floating-point mathematic operations.



NOTE *Double precision is used only by MUL and DIV instructions. Refer to their descriptions in 16-bit Real-Time ECLIPSE Assembly Language Programming.*

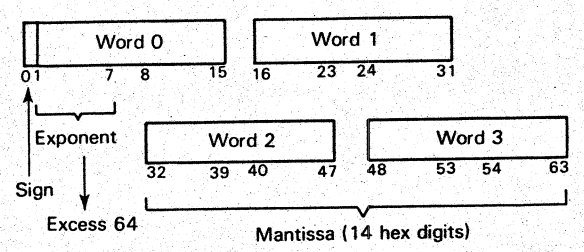
Figure 1-5 Integer formats

Single-precision (2 words)



True value of exponent = (Value in byte 0) - 64

Double-precision (4 words)



True value of exponent = (Value in byte 0) - 64

Range of exponent field: 0 to 127
Range of true value of exponent: -64 to 63

Magnitude of floating-point number:

Mantissa x 16^{4} (true value of exponent)

Normalization: Shift mantissa left 1 hex digit and decrement exponent — repeat until high-order hex digit ≠ 0.

DG-08292

Figure 1-6 Floating-point formats

Arithmetic/Logic Class Operations

The arithmetic/logic class (ALC) instructions include ADC, ADD, AND, COM, INC, MOV, NEG, and SUB. Each instruction performs a group of general functions in addition to the function implied by its name. These general functions are encoded in four fields in the ALC instructions. They are:

Set carry bit (0, 1, complement, or no change),

Shift (right, left, swap),

Skip test,

Load or No Load.

Figure 1-7 illustrates the sequence of operations performed by a general ALC instruction.

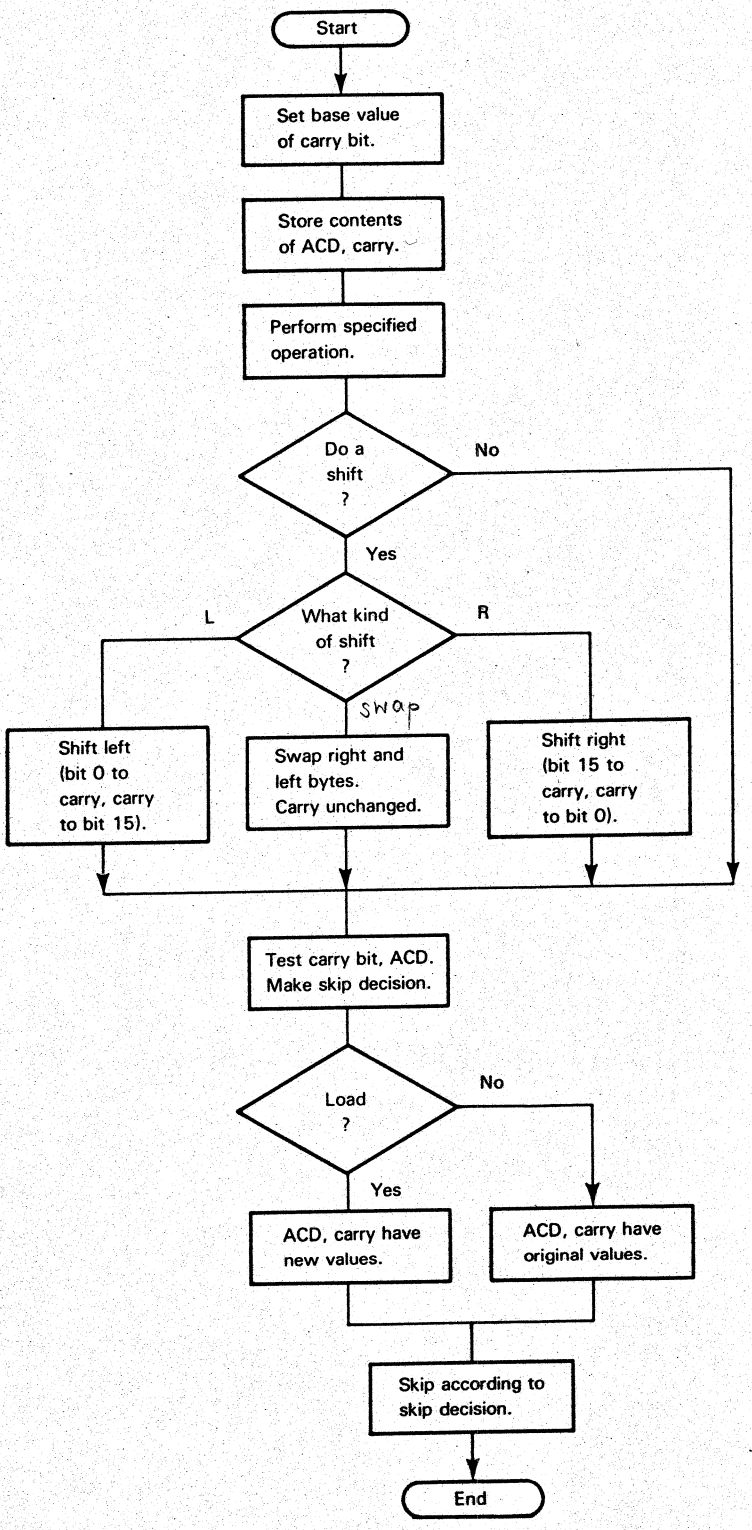


Figure 1-7 ALC instruction operation sequence

Stack Operations

The microECLIPSE processor maintains a last in/first out stack in main memory. Stack operations depend on the contents of four reserved lower page zero locations: the stack pointer, the frame pointer, the stack upper limit, and the stack fault routine pointer. The program must set up the initial contents of the stack pointer, stack upper limit, and stack fault routine pointer. Once this is done, the CPU will update the stack and frame pointers automatically; it will also jump to a user-created stack fault routine, using the stack fault routine pointer, if an instruction causes a stack overflow. A fast and efficient method of changing stacks is also provided so that a priority interrupt handler can make maximum use of the stack feature.

Floating-Point Operations

The floating-point instructions, available with Model 10/SP systems only, allow the manipulation of both single- (32 bits) and double-precision (64 bits) numbers. Single-precision gives 6-7 significant decimal digits, while double-precision gives 15-17. The decimal range of a floating point number is approximately 5.4×10^{-79} to $7.2 \times 10^{+75}$ in either precision.

Four separate 64-bit accumulators (FPACs) are available for floating-point operations. While the first floating-point operand is always in one of the FPACs, the second operand can reside in an FPAC or be fetched from memory. The four FPACs and their associated status bits can be pushed onto or popped off of the stack by one instruction.

After every floating-point operation, the floating-point status register is checked for the following fault conditions.

Overflow. Exponent overflow occurred. (The exponent should be increased by 128; otherwise, the result is correct.)

Underflow. Exponent underflow occurred. This condition is analogous to an exponent overflow.

Divide by Zero. Zero divisor detected; division aborted.

Mantissa Overflow. A bit was shifted out of the high-order end of the mantissa during a FSCAL instruction. Alternatively, the result of a FFAS or FFMD instruction does not fit into the destination.

A fault condition initiates a floating-point trap if the trap enabling bit (5) in the floating-point status register is 1. This trap pushes a return block and causes an indirect jump via location 45_8.

Several floating-point instructions have two forms, one ending in S and another in D. Those ending in S use single-precision floating-point format, while those ending in D use double-precision. The function of the two forms is otherwise identical. Floating-point formats were listed in Figure 1-6.

String Operations

String instructions (CMP, CMT, CMV, CTR) can move strings of bytes from one portion of memory to another, can compare one string of bytes with another such string and can translate a string of bytes from one representation to another. One instruction can search a string of bytes for one or more delimiters.

MAP Operations

This subsection explains the functions of address translation and protection performed by the microECLIPSE MAP unit, and describes the page 31 register. The details of programming the microECLIPSE processor to manage the address translation function are described in "Memory Allocation and Protection" later in this chapter.

Address Translation and Protection

A program can load an address translation map consisting of 32 12-bit words for each of up to four microECLIPSE users, and one Desktop Generation Model 10 data channel. It can also load 512 12-bit words into the 8086 processor's MAP. (The software actually loads 11 bits; the twelfth bit is hardware-derived from the logical *and* of the physical address.)

Each microECLIPSE user's logical address space consists of 32, 1024-word (2-Kbyte) pages. Of the 12 bits in the MAP register, ten of them specify the physical page to which a logical page is mapped; one bit specifies whether that page is write-protected; and one bit specifies whether the page is validity-protected.

The logical address space of the 8086 processor is 512 1024-word pages. The 12 bits in the 8086 MAP registers have the same function as they have in the microECLIPSE user's maps, except that the 8086 MAP does not implement write protection.

The translation process occurs every time a memory reference is made, provided the program enables the MAP by manipulating the contents of the MAP status register. A MAP fault occurs when the program tries to access a validity-protected word or write to a write-protected word. In either case, the state of the processor is saved and the program jumps to the (programmer-supplied) MAP fault handling routine.

Additionally, the program can specify I/O or indirection protection, causing a MAP fault to occur when the processor encounters an I/O instruction or more than 15 levels of address indirection. This specification can also be accomplished by writing to the MAP status register. By the same means, the processor can be instructed to interpret all I/O format instructions as Load Effective Address (LEF) instructions. Finally, whenever the MAP is enabled, the emulator trap facility is enabled. This facility is described in detail further on.

Screen Buffer Write Enable

A program running in the microECLIPSE CPU can use the microECLIPSE MAP to enable direct access to the monochrome monitor screen buffer. This allows direct display of bit-mapped graphics.

Page 31 Register

Unless the MAP is enabled, no address translation occurs. All addresses issued in unmapped mode by the CPU reference locations in the first 64 Kbytes (32 Kwords) of physical memory, that is, locations in physical pages 0-31. If a program operating in the unmapped mode requires access to some other part of memory, the page 31 register can be used to accomplish this.

A *Map Page 31* (DOB, MAP) instruction loads the page 31 register with a 10-bit translation address. This address corresponds to an entry for one page in a user map in the MAP register, but without protection bits. A memory reference addressed to logical page 31 (that is, to octal page 37 — address bits 1-5 are all one) does not access a word in physical page 31. Instead, the reference addresses a word in the physical page specified by the 10 bits in the page 31 register. Thus,

the page 31 register affords a one-page wide "window" on memory to a program running in the unmapped mode. After power up or a system reset, the page 31 register contains octal 37.

Extended Operations

The extended operation (XOP and XOP1) instructions allow the transfer of control to called procedures. An XOP instruction places all relevant return information on the stack and retrieves the address of the called procedure from a user-constructed table of procedure addresses. Control transfers to the procedure after the address has been retrieved.

Emulator Trap

The microECLIPSE CPU has a hardware provision for instruction emulation. If the CPU encounters an undefined instruction while operating in the mapped mode, it automatically makes an indirect jump through location 11_8 — provided that the contents are not zero. This location can contain the indirect address of an emulator routine. If the contents of location 11_8 are zero, an undefined instruction simply results in a NOP (no operation).

Parity Check Operations

When the parity logic detects a memory byte parity error, it requests an interrupt — if the microECLIPSE processor has enabled such interrupts. (If the 8086 processor is running at the time, the interrupt causes it to be idled while control is passed to the microECLIPSE processor.) The microECLIPSE CPU obtains the value of the program counter at the time that the error occurred with a *Read Parity Fault PC* (DIA PAR) instruction. DIA PAR returns the state of the MAP enabled bit (0), the current user map select bits, and bits 1-12 of the logical address contained in the program counter when the fault occurred.

I/O Operations

The microECLIPSE I/O instructions are used to program several system devices and up to 20 external input/output devices connected to the system's microI/O bus. These devices include

- Parity checking logic
- Two MAPs: microECLIPSE and 8086
- Two processors
- System console
- Real-time clock
- Programmable interval timer
- Diskette drive
- Onboard asynchronous interface.

The CPU addresses a controller or system device with the device code in bits 10-15 of an I/O instruction. The basic I/O instruction set controls I/O devices, sets up data channel operations, and passes data to and from these devices. Programming details for system devices are contained in this manual. Similar

information for external I/O devices or interfaces are available in the manuals for these devices.

I/O interrupt control instructions offer the programmer the following selection of I/O control schemes.

Polling (no interrupts). The CPU checks I/O device status under programmed control.

Single-level interrupts (interrupts with no priority system). The CPU services one device at a time in the order determined by the timing of the interrupt and the physical location of the device controller in the system.

Multiple-level interrupts (interrupts with a priority system). The CPU services an interrupt from a selected device in the order just described, but a higher priority device can interrupt a lower priority device's interrupt service routine. The interrupt handler accomplishes this by manipulating the devices' priority mask bits with the MSKO instruction.

If an interrupt-driven operation is selected, the programmer can choose one of the following methods to identify the interrupting device.

Test the device's Busy/Done flags with an *I/O Skip* instruction, or

Place the interrupter's device code in an accumulator with an INTA instruction, or

Identify the interrupting device, save return information, and jump through a table to a device's interrupt handling routine with a VCT instruction.

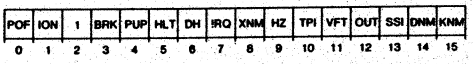
Program-Accessible Registers

Figure 1-8 shows the program-accessible registers of the microECLIPSE processor, their accumulator formats, and the instructions used to access them.

Name **Format**

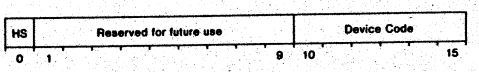
CPU Status

Read with DIS ac*, CPU



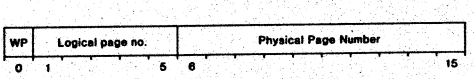
Auto Program Load

Read with DIA ac, APL or READS ac



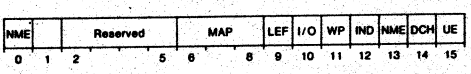
microECLIPSE MAP Address

Write with LMP ac



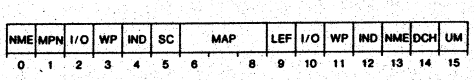
microECLIPSE MAP Status Out

Write with DOA ac, MAP



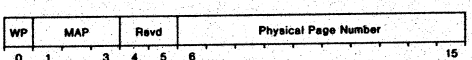
microECLIPSE MAP Status In

Read with DIA ac, MAP



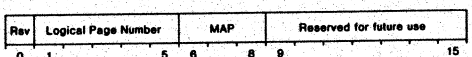
microECLIPSE MAP Page Check

Read with DIC ac, MAP



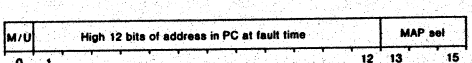
microECLIPSE MAP Initiate Page Check

Write with DOC ac, MAP



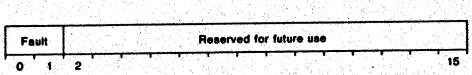
Parity Fault Address

Read with DIA ac, PAR



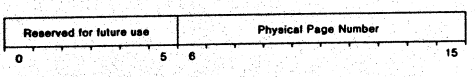
Parity Fault Code

Read with DIB ac, PAR



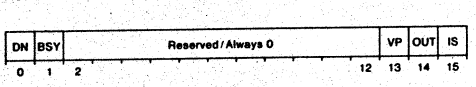
MAP Page 31

Write with DOB ac, MAP



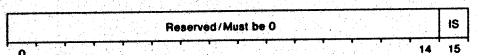
ATP Status In

Read with DIA ac, ATP



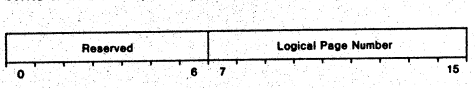
ATP Status Out

Write with DOA ac, ATP



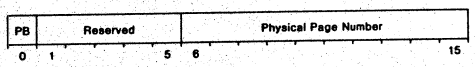
8086 Initiate Page Check

Write with DOC ac, ATP



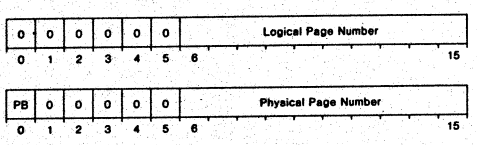
8086 Page Check

Read with DIC ac, ATP



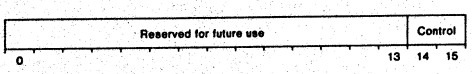
ATP MAP Pair

Load with LMPA (DIB ATP) instruction



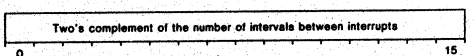
Parity Enable

Write with DOA ac, PAR



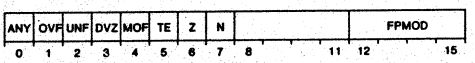
PIT Initial Count

Write with DOA ac, PIT



Floating Point Status (first word)

Read/Write with Status Register Instructions



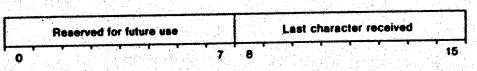
DG-25972

Figure 1-8 Program accessible registers

Name **Format**

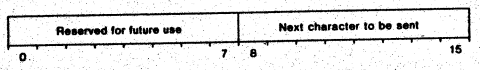
TTI or TTI1 Character Buffer

Read with DIA ac, TTI or DIA ac, TTI1



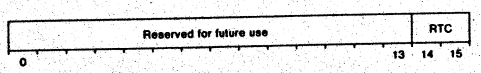
TTO or TTO1 Character Buffer

Write with DOA ac, TTO or DOA ac, TTO1



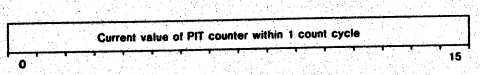
Real Time Clock

Write with DOA ac, RTC



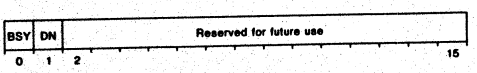
PIT Count

Read with DIA ac, PIT



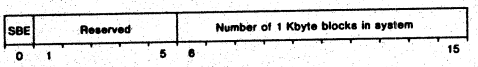
Device Status

Read with DIS ac*, device

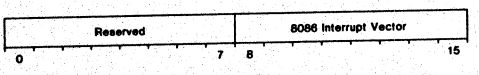


Control Memory Status

Read/write with RHVP, WHVP (AC0 = 14) instructions



ATP Interrupt Vector



NOTE *The format shown for "Device Status" applies to any I/O device, whether part of the CPU (e.g., PAR, TTO) or external to it.*

*This accumulator must not be AC0.

MicroECLIPSE Instructions

This section presents microECLIPSE CPU instructions in table form. The instructions are grouped into the following categories.

Computing instructions

Program flow management

Stack and data management

System management

Device management

Memory management

Computing Instructions

The computing instructions include add (Table 1-1), subtract (Table 1-2), multiply (Table 1-3), divide (Table 1-4), move (Table 1-5), convert (Table 1-6), logic (Table 1-7), status (Table 1-8), and computational skip (Table 1-9) instructions.

Table 1-1 Add instructions

Mnem	Instruction	Action
ADC	Add Complement	Adds an unsigned integer to the logical complement of another unsigned number.
ADD	Add	Adds the contents of one accumulator to the contents of another.
ADDI	Extended Add Immediate	Adds a signed integer in the range of −32,768 to +32,767 to the contents of an accumulator.
ADI	Add Immediate	Adds an unsigned integer in the range of 1 to 4 to the contents.
BAM	Block Add And Move	Moves blocks of memory words from one location to another, adding a constant to each one.
DAD	Decimal Add	Adds together the decimal digits found in bits 12-15 of two accumulators.
FAMS, FAMD	Add (Memory to FPAC)	Adds the floating-point number in memory to the floating-point number in an FPAC.
FAS, FAD	Add (FPAC to FPAC)	Adds the floating-point number in one FPAC to the floating-point number in another FPAC.
INC	Increment	Increments the contents of an accumulator.
ISZ, EISZ	Increment And Skip If Zero	Increments the addressed word, then skips if the incremented value is zero.

Table 1-2 Subtract instructions

Mnem	Instruction	Action
DSB	Decimal Subtract	Subtracts the decimal digit in bits 12-15 of one accumulator from the decimal digit in bits 12-15 of another accumulator.
DSZ, EDSZ	Decrement And Skip If Zero	Decrements the addressed word, then skips if the decremented value is zero.
FSMS, FSMD	Subtract (Memory from FPAC)	Subtracts the floating-point number in memory from the floating-point number in an FPAC.
FSS, FSD	Subtract (FPAC from FPAC)	Subtracts the floating-point number in one FPAC from the floating-point number in another FPAC.
SBI	Subtract Immediate	Subtracts an unsigned integer in the range of 1 to 4 from the contents of an accumulator.
SUB	Subtract	Subtracts the contents of one accumulator from the contents of another.

Table 1-3 Multiply instructions

Mnem	Instruction	Action
FMMS, FMMD	Multiply (Memory by FPAC)	Multiplies the floating-point number in memory by the floating-point number in an FPAC.
FMS, FMD	Multiply (FPAC by FPAC)	Multiplies the floating-point number in one FPAC by the floating-point number in another FPAC.
MUL	Unsigned Multiply	Multiplies the unsigned contents of two accumulators and adds the results to the unsigned contents of a third accumulator.
MULS	Signed Multiply	Multiplies the signed contents of two accumulators and adds the results to the signed contents of a third accumulator.

Table 1-4 Divide instructions

Mnem	Instruction	Action
DIV	Unsigned Divide	Divides the unsigned 32-bit integer in two accumulators by the unsigned contents of a third accumulator.
DIVS	Signed Divide	Divides the signed 32-bit integer in two accumulators by the signed contents of a third accumulator.
DIVX	Sign Extend And Divide	Extends the sign of one accumulator into a second accumulator and performs a *Signed Divide* on the result.
FDMS, FDMD	Divide (FPAC by Memory)	Divides the floating-point number in an FPAC by a floating-point number in memory.
FDS, FDD	Divide (FPAC by FPAC)	Divides the floating-point number in one FPAC by the floating-point number in another FPAC.
FHLV	Halve	Divides the floating-point number in FPAC by 2.
HLV	Halve	Divides the unsigned contents of an accumulator by 2.

Table 1-5 Move instructions

Mnem	Instruction	Action
BAM	Block Add And Move	Moves blocks of memory words from one location to another, adding a constant to each one.
BLM	Block Move	Moves blocks of memory words from one location to another.
CMT	Character Move Until True	Moves a string of bytes from one area of memory to another until a table-specified delimiter character is encountered or the source string is exhausted.
CMV	Character Move	Moves a string of bytes from one area of memory to another under control of the values in the four accumulators.
DHXL	Double Hex Shift Left	Shifts the 32-bit contents of two accumulators left 1 to 4 hex digits, depending on the value of a 2-bit number contained in the instruction.
DHXR	Double Hex Shift Right	Shifts the 32-bit contents of two accumulators right 1 to 4 hex digits, depending on the value of a 2-bit number contained in the instruction.
FEXP	Load Exponent	Places bits 1-7 of AC0 in bits 1-7 of the specified FPAC.
FLDS, FLDD	Load Floating Point	Copies a floating-point number from memory to a specified FPAC.
FMOV	Move Floating Point	Moves the contents of one FPAC to another FPAC.
FRH	Read High Word	Places the high-order 16 bits of an FPAC into AC0.
FSTS, FSTD	Store Floating Point	Copies the contents of a specified FPAC into memory.
HXL	Hex Shift Left	Shifts the contents of an accumulator left 1 to 4 hex digits, depending on the value of a 2-bit number contained in the instruction.
HXR	Hex Shift Right	Shifts the contents of an accumulator right 1 to 4 hex digits, depending on the value of a 2-bit number contained in the instruction.
LDA, ELDA	Load Accumulator	Loads data from memory to an accumulator.
LDB, ELDB	Load Byte	Places a byte of information into an accumulator.
MOV	Move	Moves the contents of an accumulator through the ALU.
POP	Pop Multiple Accumulators	Pops 1 to 4 words off the stack and places them in the indicated accumulators.
POPB	Pop Block	Returns control from a *System Call* routine or an I/O interrupt handler that does not use the stack change facility of the *Vector* instruction.
PSH	Push Multiple Accumulators	Pushes the contents of 1 to 4 accumulators onto the stack.
STA, ESTA	Store Accumulator	Stores data in memory from an accumulator.
STB, ESTB	Store Byte	Stores the right byte of an accumulator in a byte of memory.
XCH	Exchange Accumulators	Exchanges the contents of two accumulators.

Table 1-6 Convert instructions

Mnem	Instruction	Action
CTR	Character Translate	Translates a string of bytes from one data representation to another, and either moves it to another area of memory or compares it to a second string of bytes.
FFAS	Fix To AC	Converts the integer portion of a floating-point number to a signed two's complement integer and places the result in an accumulator.
FFMD	Fix To Memory	Converts the integer portion of a floating-point number to double- precision integer format and stores the result in two memory locations.
FINT	Integerize	Sets the fractional portion of the floating-point number in the specified FPAC to zero and normalizes the result.
FLAS	Float From AC	Converts a signed two's complement number in an accumulator to a single-precision floating-point number.
FLMD	Float From Memory	Converts the contents of two memory locations in integer format to floating-point format and places the result in a specified FPAC.
FNOM	Normalize	Normalizes the floating-point number in FPAC.
FSCAL	Scale	Shifts the mantissa of the floating-point number in FPAC either right or left, depending upon the contents of bits 1-7 of AC0.

Table 1-7 Logic instructions

Mnem	Instruction	Action
ANC	AND With Complemented Source	Forms the logical AND of the contents of one accumulator and the logical complement of the contents of another accumulator.
AND	AND	Forms the logical AND of the contents of two accumulators.
ANDI	AND Immediate	Forms the logical AND of a 16-bit number contained in the instruction and the contents of an accumulator.
BTO	Set Bit To One	Sets the bit addressed by the bit pointer to 1.
BTZ	Set Bit To Zero	Sets the bit addressed by the bit pointer to 0.
CMP	Character Compare	Compares one string of characters in memory to another string.
COB	Count Bits	Counts the number of ones in one accumulator and adds that number to the second accumulator.
COM	Complement	Forms the logical complement of the contents of an accumulator.
DLSH	Double Logical Shift	Shifts the 32-bit contents of two accumulators left or right depending on the contents of a third accumulator.
FAB	Absolute Value	Sets the sign bit of an FPAC to 0.
FCMP	Compare Floating Point	Compares two floating-point numbers and sets the Z and N flags accordingly.

Table 1-7 Logic instructions (Continued)

Mnem	Instruction	Action
FNEG	Negate	Inverts the sign bit of the FPAC.
IOR	Inclusive OR	Forms the logical inclusive OR of the contents of two accumulators.
IORI	Inclusive OR Immediate	Forms the logical inclusive OR of a 16-bit number contained in the instruction and the contents of an accumulator.
LOB	Locate Lead Bit	Counts the number of high-order zeros in one accumulator and adds that number to the second accumulator.
LRB	Locate And Reset Lead Bit	Performs a *Locate Lead Bit* instruction and sets the lead bit to 0.
LSH	Logical Shift	Shifts the contents of an accumulator left or right, depending on the contents of another accumulator.
NEG	Negate	Forms the two's complement of the contents of an accumulator.
XOR	Exclusive OR	Forms the logical exclusive OR of the contents of two accumulators.
XORI	Exclusive OR Immediate	Forms the logical exclusive OR of a 16-bit number contained in the instruction and the contents of an accumulator.

Table 1-8 Status instructions

Mnem	Instruction	Action
DIA MAP	Read MAP Status	Returns the status of the MAP, including the following conditions: last map enabled (by a **DOA**); MAP state (on/off); type of last MAP fault; last map loaded (by a **LMP**; state of LEF, I/O protection, write protection, and indirect protection (on/off); data channel map state; user mode (on/off).
DIS	Data In Status	Returns the status of a specified I/O device.
DIS CPU	Read Processor Status	Returns the status of the processor, including the following conditions: power fail, interrupt on, Break key NMI, power-up (reset) NMI Halt NMI, interrupt pending, external NMI, line frequency, diskette density, ATP validity fault, ATP OUT instruction, single-step NMI, diskette controller NMI, and keyboard NMI. request.
FCLE	Clear Errors	Sets bits 0-4 of the FPSR to 0.
FLST	Load Floating-Point Status	Copies the contents of two specified memory locations to the FPSR.
FSST	Store Floating-Point Status	Copies the contents of the FPSR to two memory locations.

Table 1-9 Computational skip instructions

Mnem	Instruction	Action
CLM	Compare To Limits	Compares a signed integer with two other numbers and skips if first integer is between the other two.
DSZ, EDSZ	Decrement And Skip if Zero	Decrements the addressed word, then skips if the decremented value is zero.
FSEQ	Skip On Zero	Skips the next sequential word if the Z flag of the FPSR is one.
FSGE	Skip On Greater Than or Equal To Zero	Skips the next sequential word if the N flag of the FPSR is zero.
FSGT	Skip On Greater Than Zero	Skips the next sequential word if both the Z and N flags of the FPSR are zero.
FSLE	Skip On Less Than Or Equal To Zero	Skips the next sequential word if either the Z flag or the N flag of the FPSR is one.
FSLT	Skip On Less Than Zero	Skips the next sequential word if the N flag of the FPSR is one.
FSND	Skip On No Zero Divide	Skips the next sequential word if the divide by zero (DVZ) flag of the FPSR is zero.
FSNE	Skip On Nonzero	Skips the next sequential word if the Z flag of the FPSR is zero.
FSNER	Skip On No Error	Skips the next sequential word if bits 1-4 of the FPSR are all zero.
FSNM	Skip On No Mantissa Overflow	Skips the next sequential word if the mantissa overflow (MOF) flag of the FPSR is zero.
FSNO	Skip On No Overflow	Skips the next sequential word if the overflow (OVF) flag of the FPSR is zero.
FSNOD	Skip On No Overflow And No Zero Divide	Skips the next sequential word if both the overflow (OVF) flag and the divide by zero (DVZ) flag of the FPSR are zero.
FSNU	Skip On No Underflow	Skips the next sequential word if the underflow (UNF) flag of the FPSR is zero.
FSNUD	Skip On No Underflow And No Zero Divide	Skips the next sequential word if both the underflow (UNF) flag and the divide by zero (DVZ) flag of the FPSR are zero.
FSNUO	Skip On No Underflow And No Overflow	Skips the next sequential word if both the underflow (UNF) flag and the overflow (OVF) flag of the FPSR are zero.
ISZ, EISZ	Increment And Skip If Zero	Increments the addressed word, then skips if the incremented value is zero.
SGE	Skip If ACS Greater Than Or Equal to ACD	Compares the signed integers in two accumulators and skips if the first is greater than or equal to the second.
SGT	Skip If ACS Greater Than ACD	Compares the signed integers in two accumulators and skips if the first is greater than the second.
SNB	Skip On Nonzero Bit	Skips the next sequential word if the bit addressed by the bit pointer is one.
SZB	Skip On Zero Bit	Skips the next sequential word if the bit addressed by the bit pointer is zero.
SZBO	Skip On Zero Bit And Set To One	Sets the bit addressed by the bit pointer to one and skips the next sequential word if the bit was originally zero.

Program Flow Management

The instructions for program flow management include noncomputational skip (Table 1-10), jump (Table 1-11), subroutine (Table 1-12), interrupt (Table 1-13), and accumulator (Table 1-14) instructions.

Table 1-10 Noncomputational skip instructions

Mnem	Instruction	Action
CLM	Compare To Limits	Compares a signed integer with two other numbers and skips if first integer is between the other two.
FNS	No Skip	No operation.
FSA	Skip Always	Skips the next sequential word.

Table 1-11 Jump instructions

Mnem	Instruction	Action
DSPA	Dispatch	Compares a signed integer with two other numbers and continues sequential execution if the integer is not between the others; otherwise, uses the integer as an index into a table and places indexed value in the program counter.
JMP, EJMP	Jump	Places an effective address in the program counter.
JSR, EJSR	Jump To Subroutine	Increments program counter and stores incremented value in AC3; then places a new address in the program counter.
POPJ	Pop PC and Jump	Pops the top word off the stack and places it in the program counter.
PSHJ	Push PC and Jump	Pushes the address of the next sequential instruction onto the stack, computes the effective address *E*, and places it in the program counter.

Table 1-12 Subroutine instructions

Mnem	Instruction	Action
JSR, EJSR	Jump To Subroutine	Increments program counter and stores incremented value in AC3; then places a new address in the program counter.
PSHR	Push Return Address	Pushes the address of the instruction after the next sequential instruction onto the stack.
RSTR	Restore	Returns control from certain types of I/O interrupts.
RTN	Return	Returns control from subroutines that issue a *Save* instruction at their entry points.
SAVE	Save	Saves the information required by the *Return* instruction.
XOP	Extended Operation	Pushes a return block on the stack, placing the address in the stack of the specified accumulators into AC2 and AC3, and transfers control to 1 of 32 other procedures with the XOP table.
XOP1	Extended Operation	Same as XOP, except that 32 is added to the entry number before entering the XOP table and only 16 table entries can be specified.

Table 1-13 Interrupt instructions

Mnem	Instruction	Action
DSPA	Dispatch	Compares a signed integer with two other numbers and continues sequential execution if the integer is not between the others; otherwise, uses the integer as an index into a table and places indexed value in the program counter.
FTD	Trap Disable	Sets the trap enable flag of the FPSR to zero.
FTE	Trap Enable	Sets the trap enable flag of the FPSR to one.

Table 1-14 Accumulator instructions

Mnem	Instruction	Action
LEF, ELEF	Load Effective Address	Places an effective address in an accumulator.
XCT	Execute	Executes contents of an accumulator as an instruction.

Stack and Data Management

The stack and data management instructions are summarized in Table 1-15.

Table 1-15 Stack instructions

Mnem	Instruction	Action
FPOP	Pop Floating-Point State	Pops an 18-word floating-point return block off the user stack and alters the state of the floating-point unit.
FPSH	Push Floating-Point State	Pushes an 18-word floating-point return block onto the user stack.
MSP	Modify Stack Pointer	Changes the value of the stack pointer and checks for overflow.
POP	Pop Multiple Accumulators	Pops 1 to 4 words off the stack and places them in the indicated accumulators.
POPB	Pop Block	Returns control from a *System Call* routine or an I/O interrupt handler that does not use the stack change facility of the *Vector* instruction.
PSH	Push Multiple Accumulators	Pushes the contents of 1 to 4 accumulators onto the stack.
PSHJ	Push PC Jump	Pushes the address of the next sequential instruction onto the stack, computes the effective address *E*, and places it in the program counter.
PSHR	Push Return Address	Pushes the address of the instruction after the next sequential instruction onto the stack.
RSTR	Restore	Returns control from certain types of I/O interrupts.
RTN	Return	Returns control from subroutines that issue a *Save* instruction at their entry points.
SAVE	Save	Saves the information required by the *Return* instruction.
XOP	Extended Operation	Pushes a return block on the stack, placing the address in the stack of the specified accumulators into AC2 and AC3, and transfers control to 1 of 32 other procedures via the XOP table.
XOP'	Extended Operation	Same as XOP except that 32 is added to the entry number before entering the XOP table and only 16 table entries can be specified.

System Management

The *System Call* instruction can be specified as SYC, SCL (which is equivalent to SYC 0, 1), or SVC (which is equivalent to SYC 0, 0). This instruction turns off the MAP if it is on, pushes a return block onto the stack, and places the address of the System Call handler in the program counter.

Device Management

The instructions for device management include basic I/O (Table 1-16), I/O command flag (Table 1-17), I/O interrupt (Table 1-18), I/O skip flag (Table 1-19), CPU device (Table 1-20), and CPU skip flag (Table 1-21) instructions.

Table 1-16 I/O instructions

Mnem	Instruction	Action
DIA*[f]*	Data In A	Transfers data from the A buffer of an I/O device to an accumulator.
DIB*[f]*	Data In B	Transfers data from the B buffer of an I/O device to an accumulator.
DIC*[f]*	Data In C	Transfers data from the C buffer of an I/O device to an accumulator.
DIS	Data In Status	Returns the status of a specifed I/O device.*
DOA*[f]*	Data Out A	Transfers data from an accumulator to the A buffer of an I/O device.
DOB*[f]*	Data Out B	Transfers data from an accumulator to the B buffer of an I/O device.
DOC*[f]*	Data Out C	Transfers data from an accumulator to the C buffer of an I/O device.
NIO*[f]*	No I/O Transfer	Sets Busy or Done flag. No I/O transfer occurs.

**Refer to the accumulator format of this instruction.*

Table 1-17 I/O command flags

Mnem	Flag Value	Action
[f] omitted	00	Does not alter the Busy and Done flags.
[f] = **S**	01	Starts the device; sets Busy flag to one and Done flag to zero.
[f] = **C**	10	Idles the device; sets Busy flag to zero, and sets Done flag to zero.
[f] = **P**	11	I/O pulse; effect depends upon device.

Table 1-18 I/O interrupt instructions

Mnem	Instruction	Action
INTA (DIB*[f]* CPU)	Interrupt Acknowledge	Returns the device code of an interrupting device.
INTDS (NIOC CPU)	Interrupt Disable	Sets CPU Interrupt On flag to zero.
INTEN (NIOS CPU)	Interrupt Enable	Sets CPU Interrupt On flag to one.
MSKO (DOB*[f]* CPU)	Mask Out	Changes the priority mask.
POPB	Pop Block	Returns control from a *System Call* routine or an I/O interrupt handler that does not use the stack change facility of the *Vector* instruction.

Table 1-18 I/O interrupt instructions (Continued)

Mnem	Instruction	Action
RSTR	Restore	Returns control from I/O interrupts that use the stack change facility of the VCT instruction.
SKP*[t]*	I/O Skip	Skips if the I/O condition t is true.
VCT	Vector On Interrupting Device Code	Identifies highest priority interrupt; passes control through a table to a handler routine for device.
XCT	Execute	Executes contents of an accumulator as an instruction.

Table 1-19 I/O skip flags

Mnem	Flag Value	Action
[t]=**BN**	00	Tests Busy flag for nonzero.
[t]=**BZ**	01	Tests Busy flag for zero.
[t]=**DN**	10	Tests Done flag for nonzero.
[t]=**DZ**	11	Tests Done flag for zero.

Table 1-20 CPU device instructions

Mnem	Instruction	Action
DIS CPU	Read Processor Status	Returns the status of the processor, including the following conditions: power fail, interrupt on, Break key reset, power-up reset, halt instructions, and interrupt request.*
HALT (DOC*[f]* CPU)	Halt	Stops the processor.
INTA (DIB*[f]* CPU)	Interrupt Acknowledge	Returns the device code of an interrupting device.
INTDS (NIOC CPU)	Interrupt Disable	Sets CPU Interrupt On flag to zero.
INTEN (NIOS CPU)	Interrupt Enable	Sets CPU Interrupt On flag to one.
IORST (DICC*[f]* CPU)	Reset	Sets all Busy and Done flags and the priority mask to zero.
MSKO (DOB*[f]* CPU)	Mask Out	Changes the priority mask.
READS (DIA*[f]* CPU)	Read Switches	Places the contents of the virtual console register into an accumulator.**
SKP*[t]* CPU	CPU Skip	Tests the Interrupt On or Power Fail flag and skips the next sequential word if the test condition is true.

**Refer to the accumulator format for this instruction.*

***Refer to a description of this register.*

Table 1-21 CPU skip flags

Mnem	Flag Value	Action
[t]=BN	00	Tests Interrupt On flag for nonzero.
[t]=BZ	01	Tests Interrupt On flag for zero.
[t]=DN	10	Tests Power Fail flag for nonzero.
[t]=DZ	11	Tests Power Fail flag for zero.

Memory Management

The instructions for memory management are summarized in Table 1-22.

Table 1-22 MAP instructions

Mnem	Instruction	Action
DIA MAP	Read Map Status	Reads the status of the current map.
DIC MAP or DIC ATP	Page Check	Provides the identity and some characteristics of the physical page that corresponds to the logical page identified by the immediately preceding *Initiate Page Check* instruction.
DOA MAP	Load Map Status	Defines the parameters of a new map.
DOB MAP	Map Supervisor Page 31	Specifies the physical page corresponding to logical page 31 of unmapped address space.
DOC MAP or DOC ATP	Initiate Page Check	Identifies a logical page; selects map without changing status.
LMP	Load microECLIPSE MAP	Loads successive words from memory into the microECLIPSE MAP, where they are used to define a user or data channel map.
LMPA	Load ATP (8086) Map	Loads successive word pairs from memory into the 8086 MAP, where they are used to define the translation function for the 8086.
NIOP MAP	Map Single Cycle	Maps one memory reference using the last user map, or turns off memory mapping for the microECLIPSE processor.
RHYP	Read Control Memory Status	Loads the contents of the control memory status register into AC1.
WHYP	Write Control Memory Status	Writes the contents of AC1 into the control memory status register. Used with RHYP to enable writing to the monochrome monitor screen buffer.

8086 CPU Features

The 8086 is two processors in one. Data manipulation (that is, instruction execution), status maintenance, and instruction decoding are done within the *execution unit* (EU), while instruction fetching, bus control, memory accessing, and data I/O are done by the *bus interface unit* (BIU), which has its own ALU and registers. Figure 1-9 illustrates the architecture of the 8086 processor.

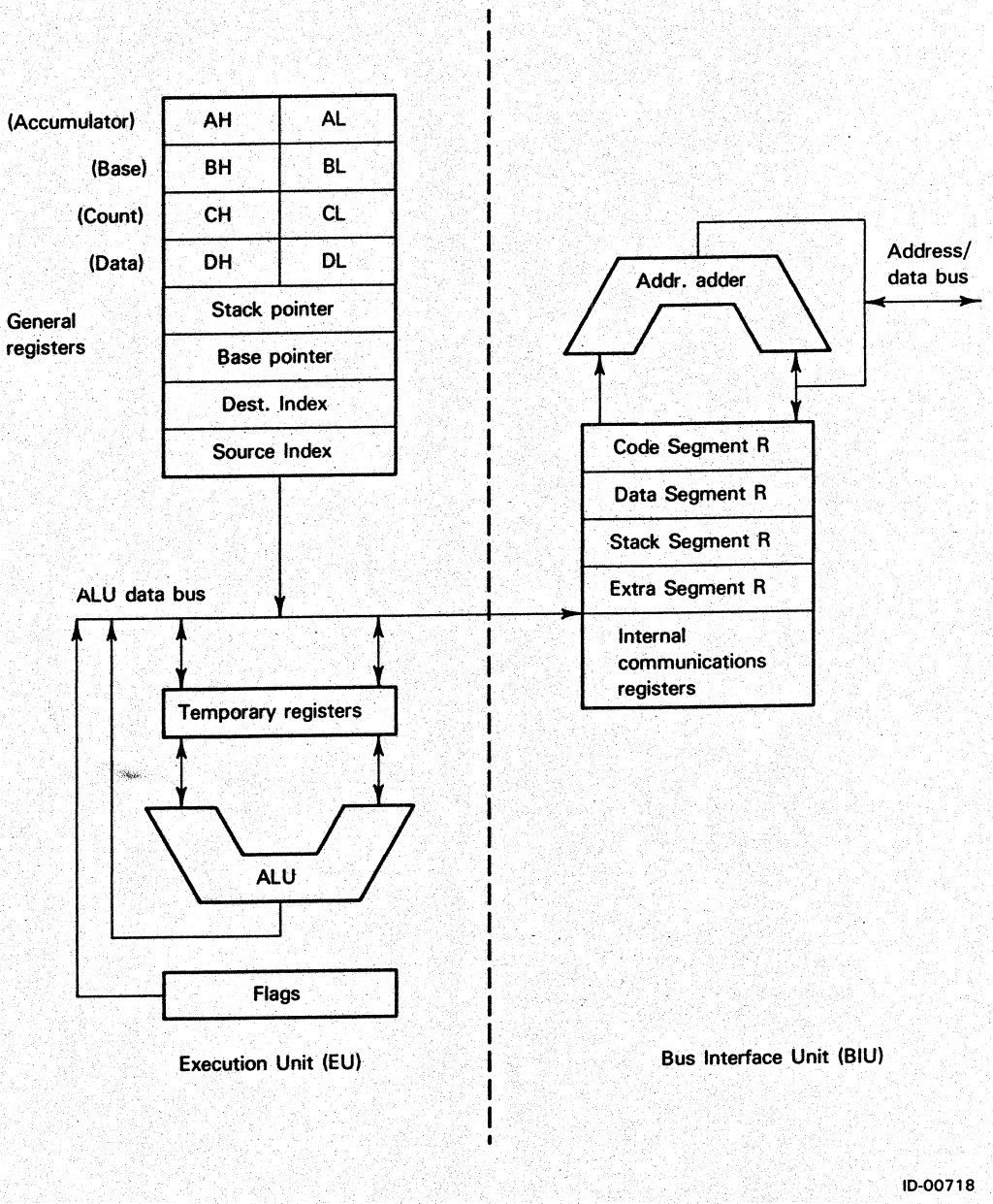


Figure 1-9 Architecture of the 8086 processor

The 8086 processor has two distinct sets of registers: those in the EU, which are used for instruction execution, and those in the BIU, which partition memory into four segments. The memory segments are automatically referenced depending on the operation involved. This feature facilitates relocatable coding.

8086 Memory Segmentation

Memory space in an 8086 system is logically divided into 64-kilobyte segments. The CPU can access four segments at a time, whose starting addresses are stored in BIU registers. The segment registers can be manipulated by the program and are discussed in the context of addressing modes below.

In general, the memory has three types of contents: instructions, (variable) data, and stack items. These different types of contents are stored in different segments (areas) of memory.

8086 Addressing Modes

From the point of view of how it performs *addressing*, the 8086 deals with four kinds of data: register data, immediate data, I/O data, and memory data. Register data and immediate data require no memory reference. Memory mapped I/O data and ordinary data in memory must be located according to their 20-bit physical address. (Non-memory-mapped I/O will be discussed later.)

The physical address is determined by two different sets of calculations: first, the EU calculates an effective address (EA) in one of the five addressing modes (described below), and then the BIU computes a physical address based on the memory segmentation scheme in the 8086. The effective address calculated by the EU is an unsigned 16-bit number that gives the data's memory address displacement in bytes from the beginning of the segment the data is in. The content of the second byte of an instruction tells the EU by what method to calculate the effective address. Figure 1-10 shows in general how the EU goes about the calculation. The item labelled "displacement" in the figure is an 8- or 16-bit number that the EU gets from the instruction.

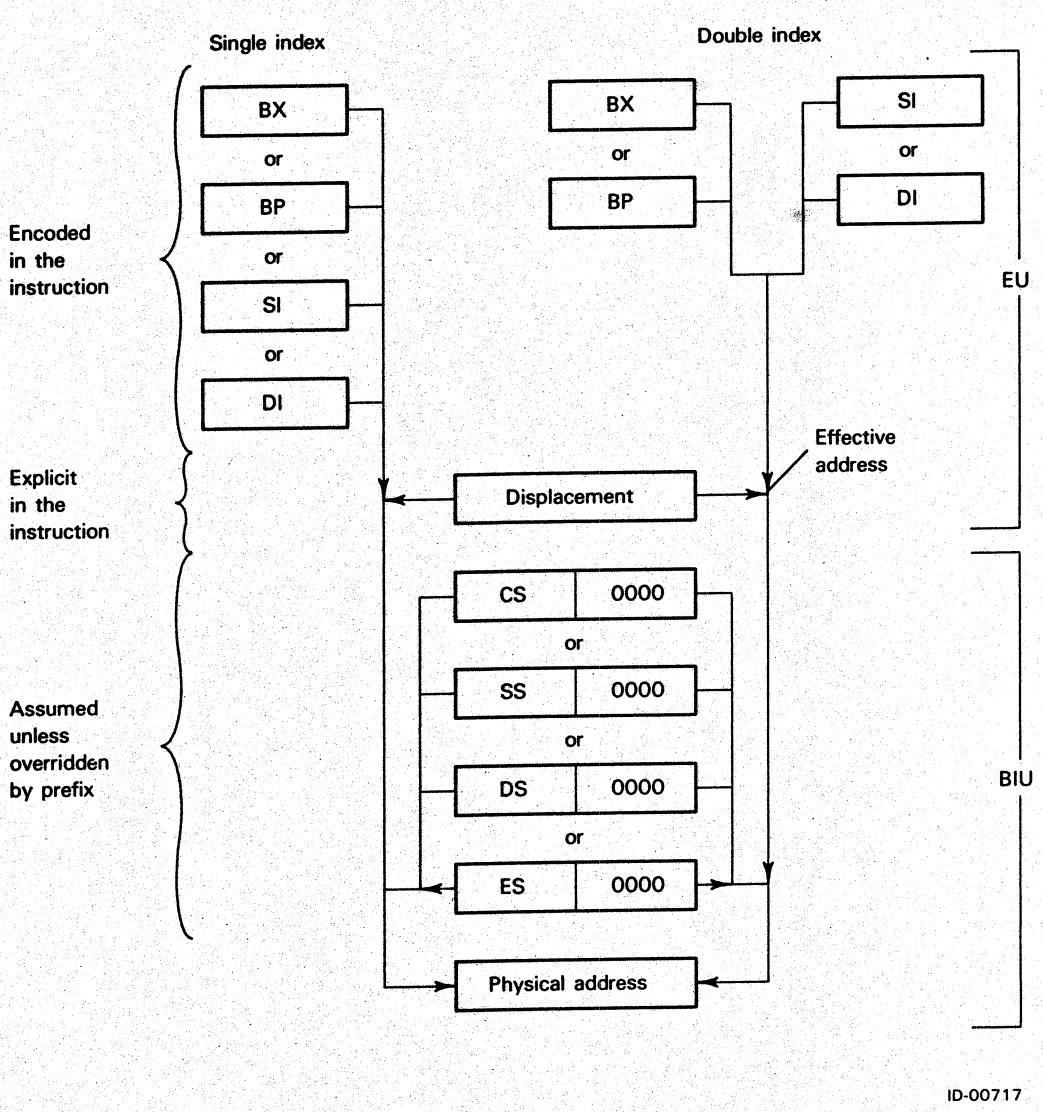


Figure 1-10 8086 memory address computation

The five forms of effective address calculation or addressing modes, are:

1. Direct addressing The effective address is simply the displacement itself.
2. Register indirect addressing The effective address is the contents of the specified base or index register (or in the case of JMP or CALL the contents of any specified register).
3. Based addressing The effective address is the sum of a displacement value and the contents of a base register (BP or BX).
4. Indexed addressing The effective address is the sum of a displacement and the contents of an index register (SI or DI).

5. Based indexed addressing The effective address is the sum of the contents of a based register, an index register, and a displacement.

NOTE *String instructions do not use the addressing modes described above. When a string instruction is executed, SI points to the first byte in the source string and DI points to the first word in the destination string. In repeated operations, the CPU automatically adjusts the contents of SI and DI to obtain the next bytes.*

Segment Registers

After the EU has calculated a 16-bit *logical* effective address, the BIU converts that address into a 20-bit *physical* address. This is done using the contents of the BIU segment registers listed below.

The Code Segment register (CS) contains the address of the code segment base (the location of the first byte of code).

The Data Segment register (DS) contains the address of the data segment base.

The Stack Segment register (SS) points to the stack segment base.

The Extra Segment register (ES) points to the extra segment base.

The BIU, to calculate a physical address, takes a value from one of the segment registers and combines it with the effective address from the EU. (The EA in this context is now referred to as the *offset.*) The BIU shifts the segment base value left four positions and adds the offset to it as shown in Figure 1-11.

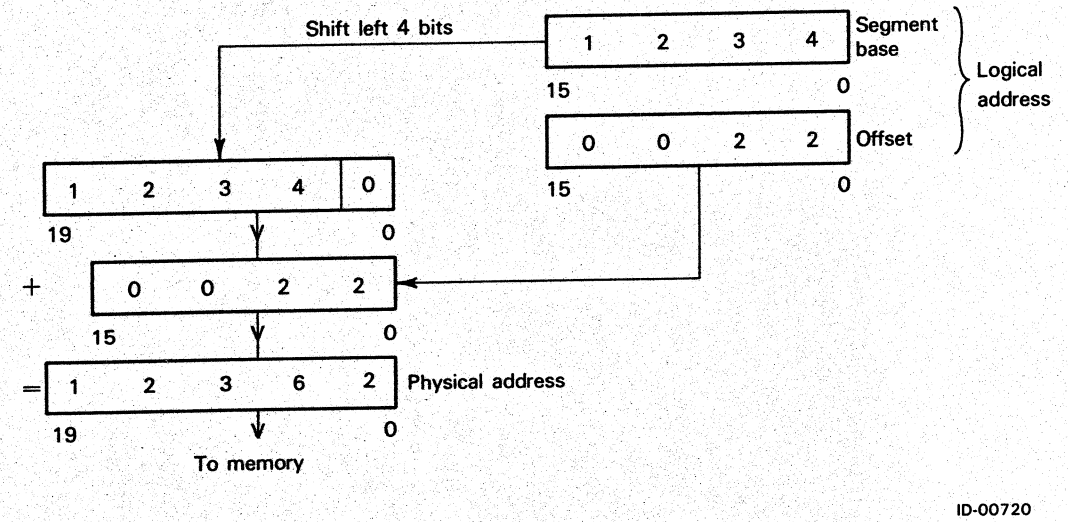


Figure 1-11 8086 physical address generation

Which segment register the BIU takes the base value from depends on the type of reference:

Instructions use the code segment (CS) base and the contents of the instruction pointer (IP) as the offset.

Stack operations use the stack segment (SS) base and the contents of the stack pointer (SP) as the offset.

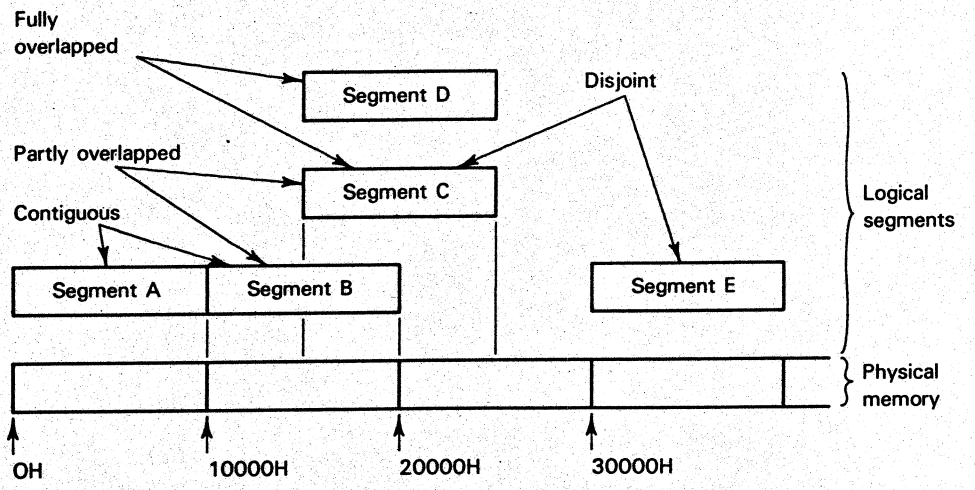
String sources are taken from the data segment (DS) base (or the code, stack, or extra segment base if specified in the instruction) and the contents of the source index (SI) as the offset.

String destinations are taken from the extra segment (ES) base and the destination index (DI) as the offset.

Other variables are taken from the data segment (DS) base (or the code, extra, or stack segment base if specified in the instruction) and the effective address as the offset.

If BP is used as the base register, then the data is taken using the stack segment base (or the code, data, or extra segment base if specified in the instruction) and the effective address as the offset.

Since 64 kilobytes of memory can be accessed using any one of the segment bases, the maximum memory space that is available at any given time is 256 kilobytes. (The result of the algorithm to calculate physical addresses is that sums greater than 64 K "wrap around.") The segments, however, do not have to be mutually exclusive. They can be conjoint or overlapping, as shown in Figure 1-12. The programmer can, of course, access more than 256 kilobytes of memory by writing new values into the segment registers. In order to retain dynamic relocatability, this should not be done within a program segment.



ID-00721

Figure 1-12 8086 segment locations in physical memory

Input/Output

The 8086 processor in a Model 10 or 10/SP system does not perform any I/O operations. Instead, it calls on the microECLIPSE processor to perform data transfers to and from peripheral devices. The 8086 uses one I/O instruction to call the microECLIPSE: OUT. Exchange of control between the 8086 and the microECLIPSE processors is described later in the Model 10 or 10/SP-specific programming section.

Instructions

The 8086 has six different kinds of instructions:

Data transfer instructions

Arithmetic instructions

Bit manipulation instructions

String instructions

Program transfer instructions

Processor control instructions

Data Transfer Instructions

Data transfer instructions move single (as opposed to strings) bytes and words between memory and registers, and between register AL or AX and I/O ports. Stack instructions are included in this group, since they are data-moving instructions, as are load segment instructions, and instructions that move flag contents. Table 1-23 lists these instructions.

Table 1-23 8086 data transfer instructions

Instruction	Function
MOV	Move byte or word
PUSH	Push word onto stack
POP	Pop word off stack
XCHG	Exchange word or byt
XL	Translate byte
IN	Input byte or word (not ud in %tbox()ystems
OUT	Output byte or word; used to pass control to microECLIPSE processor
LEA	Load effective address
LDS	Load pointer using DS
LES	Load pointer using ES
LAHF	Load AH register from flags
SAHF	Store AH register in flags
PUSHF	Push flags onto stack
POPF	Pop flags offxDtack

Arithmetic Instructions

Arithmetic instructions perform the four arithmetic (+, −, x, /) operations on four types of numbers: signed and unsigned binary integers; unsigned, packed decimal and unsigned, unpacked decimal numbers. These are listed in Table 1-24.

Table 1-24 8086 arithmetic instructions

Instruction	Function
ADD	Add byte or word
ADC	Add byte or word with carry
INC	Increment byte or word with carry
AAA	ASCII adjust for addition
DAA	Decimal adjust for addition
SUB	Subtract byte or word
SBB	Subtract byte or word with borrow
DEC	Decrement byte or word by 1
NEG	Negate byte or word
CMP	Compare byte or word
AAS	ASCII adjust for subtraction
DAS	Decimal adjust for subtraction
MUL	Multiply byte or word unsigned
IMUL	Integer multiply byte or word
AAM	ASCII adjust for multiply
DIV	Divide byte or word unsigned
IDIV	Integer divide byte or word
AAD	ASCII adjust for subtraction
CBW	Convert byte or word
CWD	Convert word to double word

Bit Manipulation Instructions

Bit manipulation instructions include logical instructions, and shift and rotate instructions. The logical instructions include one that tests the results of an AND, sets flags, but does not change the operands. Table 1-25 lists these instructions.

Table 1-25 8086 bit manipulation instructions

Instruction	Function
NOT	Logical NOT byte or word
AND	Logical AND byte or word
OR	Logical OR byte or word
XOR	Logical XOR byte or word
TEST	Logical AND of two bytes or words with no change of operands
SHL/SAL	Shift logical/arithmetic left byte or word
SHR/SAR	Shift logical/arithmetic right byte or word
ROL	Rotate left byte or word
ROR	Rotate right byte or word
RCL	Rotate through carry bit left byte or word
RCR	Rotate through carry bit right byte or word

String Instructions

String instructions allow moving, scanning, and comparison of strings in bytes up to 64 kilobytes in length. These instructions are listed in Table 1-26.

Table 1-26 8086 string instructions

Instruction	Function
REP	Repeat
REPE/REPZ	Repeat while equal/zero
REPNE/REPNZ	Repeat while not equal zero
MOVS	Move byte or word string
MOVSB/MOVSW	Alternate (no-operand) coding of move byte or word string
CMPS	Compare byte or word string
SCAS	Scan byte or word string
LODS	Load byte or word string
STOS	Store byte or word string

Program Transfer Instructions

Program transfer instructions include unconditional transfers, conditional transfers, iteration control instructions, and interrupt-related instructions. They are listed in Table 1-27.

Table 1-27 8086 program transfer instructions

Instruction	Function
CALL	Call procedure
RET	Return from procedure
JMPC	Unconditional jump
JA/JNBE	Jump if above/not below nor equal
JAE/JNB	Jump if above or equal/not below
JB/JNAE	Jump if below/not above nor equal
JBE/JNA	Jump if below or equal/not above
JC	Jump if carry flag set
JE/NZ	Jump if equal/zero
JG/JNLE	Jump if greater/not less nor equal
JGE/JNL	Jump if greater or equal/not less
JL/JNGE	Jump if less/not greater nor equal
JLE/JNG	Jump if less or equal/not greater
JNC	Jump if carry flag not set
JNE/JNZ	Jump if not equal/not zero
JNO	Jump if no overflow (overflow flag was not set)
JNP/JPO	Jump if parity flag not set/parity odd
JNS	Jump if sign flag not set
JO	Jump if overflow flag set
JP/JPE	Jump if parity flag set/parity even
JS	Jump if sign flag set
LOOP	Loop
LOOPE/LOOPZ	Loop if equal/zero
LOOPNE/LOOPNZ	Loop if not equal/not zero
JCXZ	Jump if register CX = 0
INT	Interrupt
INTO	Interrupt if overflow bit set
IRET	Interrupt return

Processor Control Instructions

Processor control instructions allow programs to control various CPU functions. They set flags, synchronize the CPU with external events, or cause the CPU to do nothing. They are listed in Table 1-28.

Table 1-28 8086 processor control instructions

Instruction	Function
STC	Set carry flag
CLC	Clear carry flag
CMC	Complement carry flag
STD	Set direction flag
CLD	Clear direction flag
STI	Set interrupt enable flag
CLI	Clear interrupt enable flag
HLT	Halt until interrupt or reset
WAIT	Wait for $\overline{\text{TEST}}$ pin active
ESC	Escape to external processor
LOCK	Lock bus during next operation
NOP	No operation

Several of the operations that can be performed by 8086 instructions are somewhat unusual in that they offer greater computing power than those found in earlier processors. Examples are: multiplication and division of the four different types of numbers, the string instructions, non-destructive bit-testing, byte translation, software-generated interrupts, and multiprocessor coordinating instructions.

Processor Programming

This section describes in detail system management programmable elements that are unique to the Model 10 and Model 10/SP systems. It outlines

System power-up response,

System status registers and how to access them,

Programming the MAPs,

Programming the parity facility,

Passing control between the microECLIPSE and 8086 processors,

Programming bit-mapped graphics

Response to the microECLIPSE HALT instruction, and

System powerfail response,

Instruction execution times.

SPU Power-Up Response

After power is applied to the SPU, the CPU is initialized and enters the Halt state. At this time, all Busy and Done flags are set to zero, all I/O interface registers belonging to system devices and external I/O devices are cleared, and bits 0-5, 9-12, 14, and 15 of the MAP status register are cleared. The CPU status register is cleared except for the Power-up bit (bit 4), which is set to one, and the line frequency and density bits, which are configuration dependent. The

contents of the Page 31 register are set to 37 (octal). The parity fault code and parity enable registers are cleared and the ATP (8086) status register is cleared.

From the Halt state, the CPU enters the virtual console, which clears the Power-up bit and performs a system self-test. The power-up self-test checks system console screen memory, the system console interface (both display and keyboard), the onboard asynchronous interface, all of main system memory, the MAPs, the real-time clock, and the programmable interval timer. It checks a subset of instructions executed from each microECLIPSE user memory space and from unmapped space, and the operation of the microECLIPSE-8086 interface.

If no errors are detected, the following message is displayed on both the system console and the device code 51 device (onboard asynchronous interface):

TESTING...

ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789

BLINK DIM REVERSE UNDERLINE ALL NORMAL

TEST PASSED

The word BLINK will blink, DIM will be dimmed, REVERSE will be displayed in reverse video, UNDERLINE will be underlined, ALL will have all four attributes, and NORMAL will be displayed with none. The blinking will continue for about 2 seconds, stop for 2 seconds, and then begin again.

NOTE *The user should check that this display appears as described, as the test program has no way of detecting that the attributes are actually effected.*

Once the system has successfully completed the self-test, it will follow the flowchart of Figure 1-13 to determine how to program load.

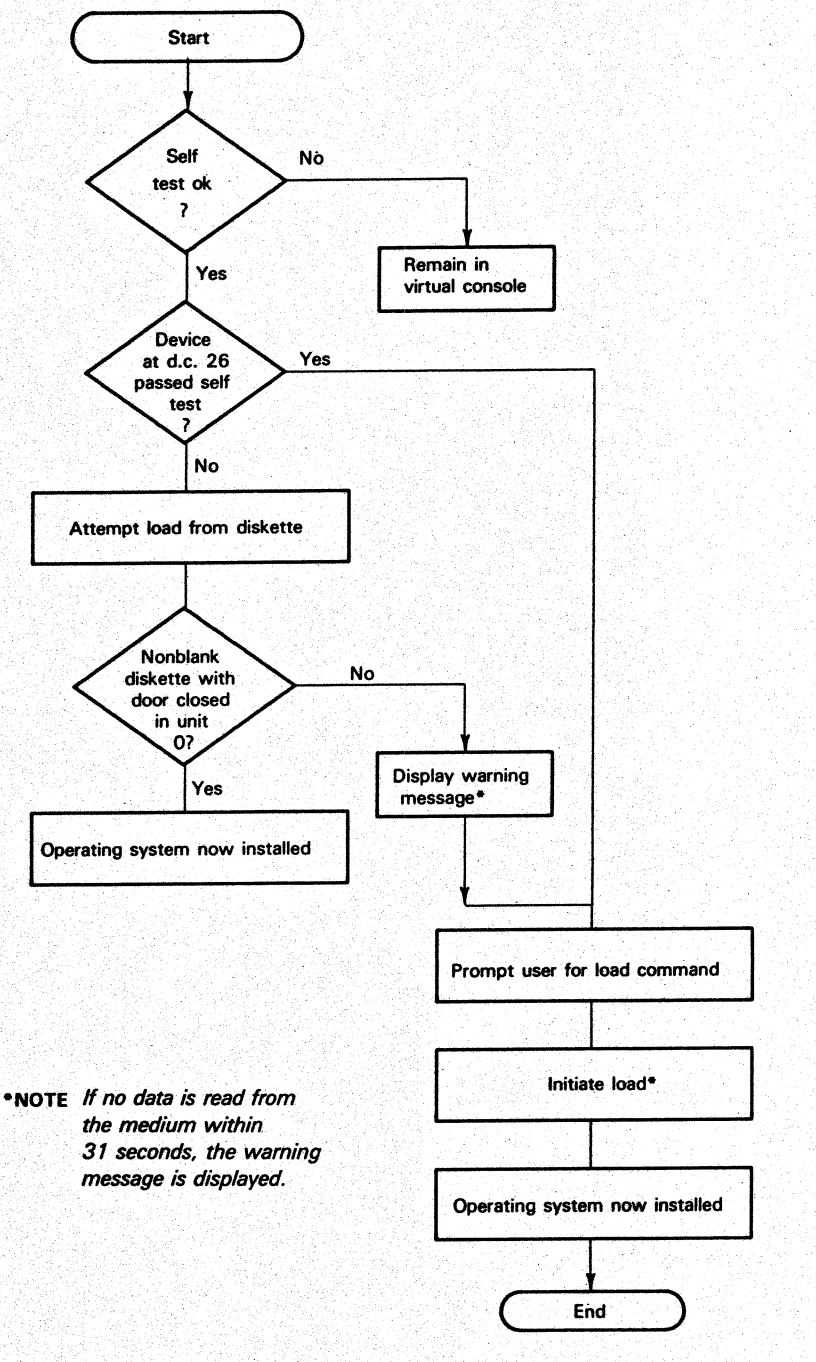


Figure 1-13 Models 10 or 10/SP powerup procedure

If there is no hard disk in the system at device code 26 or if the hard disk at device code 26 fails its self-test, then the Desktop Generation Model 10 system displays

LOADING FROM DISKETTE

and performs an automatic program load from the diskette.

If the program load from the diskette fails because there is no diskette in the drive or because a blank diskette is in the drive, the system displays the warning message

followed by the virtual console prompt, an exclamation point. Note that this same warning message is displayed whenever a program load is attempted from any device (except a diskette drive with no diskette in it) if no data is returned within 30 seconds.

If there is a hard disk in the system, the display message is

SELECT LOAD DEVICE: 20H (FOR DISKETTE) OR 26H (FOR DISK)

!

The user should then enter the appropriate load command to select which device performs the program load.

If no hard disk is contained in the system and the diskette drive has no diskette in it, the message displayed is the same as that above:

SELECT LOAD DEVICE: 20H (FOR DISKETTE) OR 26H (FOR DISK)

!

There is no explicit message warning that there is no diskette in the drive. The user with only a diskette drive should interpret the LOAD DEVICE question as an instruction to insert a diskette.

microECLIPSE CPU Status Register

The CPU status register reports status pertaining to the microECLIPSE processor. A *CPU Status* instruction (DIS) can be used to return the status information to a specified CPU accumulator. The CPU status register definitions are listed below.

CPU Status

DIS CPU

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

Returns the status of the CPU status register and places this data into the specified accumulator.

NOTE *DIS 0 CPU is equivalent to the SKP 0 CPU instruction.*

The information contained in the specified accumulator is in the format:

POF	ION	1	BRK	PUP	HLT	DH	IRQ	XNM	HZ	TPI	VFT	OUT	SSI	DNM	KNM
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Mnem	Bit	Name	Function (If Set to 1)
POF	0	Power Fail	Power Fail flag set.
ION	1	Interrupt On	Interrupt On flag set.
—	2	—	Reserved. (Set to one.)
BRK	3	Break Key Interrupt Enabled	Always 0. Indicates that a nonmaskable interrupt will occur when a Break signal is received from the printer port.
PUP	4	Power Up	Power came up since last **DOAP CPU**
		Reset	used only by the virtual console.
HLT	5	Halt	**HLT** instruction was executed.
			Used only by the virtual console.
DH	6	Halt Dispatch	Always one. Indicates that a **Halt** instruction will cause the processor to enter the virtual console.
IRQ	7	Interrupt	An interrupt is being requested.
XNM	8	External NMI	An external NMI, such as a parity fault interrupt, is pending. This bit is cleared when the CPU status register is read. When taken, the NMI will cause entry into control memory code.
HZ	9	Line frequency	If 1, line frequency is 60 Hz. If 0, line frequency is 50 Hz.
TPI	10	Density	If 1, diskette track density is 48 TPI. If 0 track density is 96 TPI.
VFT	11	ATP validity fault	If 0, indicates that the 8086 processor caused a validity protect fault.
OUT	12	ATP OUT instruction	If 0, indicates that the 8086 processor executed an OUT instruction.
SSI	13	Single-step NMI	Set to 1 in virtual console single-step mode.
DNM	14	Diskette controller NMI	A nonmaskable interrupt was generated by the diskette controler.
KNM	15	Keyboard NMI	A nonmaskable interrut was generated by the keyboard interface.

Program Load Register

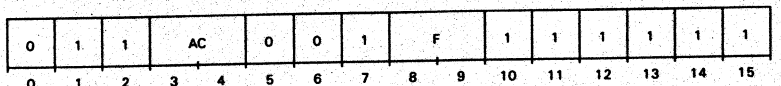
The program load register is also called the virtual console switch register. It indicates the device from which the system has been loaded. After a program load has been performed, the device code of the device loaded from is placed in it. A *reads* instruction returns this device code.

The auto program load register definitions are listed below. For more information on auto program load, refer to Chapter 3, "Virtual Console."

Read Virtual Console Switch Register

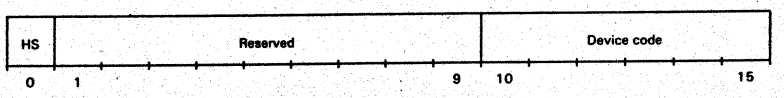
READS *ac*

DIA CPU



Places the contents of the virtual console register into an accumulator.

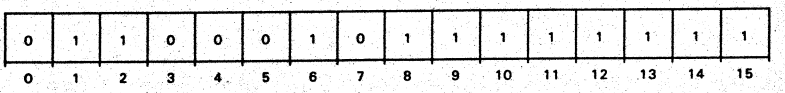
After the transfer, sets the Interrupt On flag according to the function specified by *f*. The format of the specified accumulator after the transfer is:



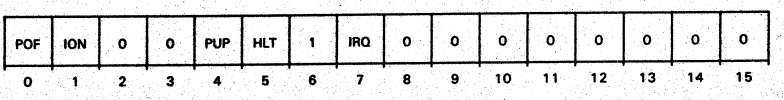
Mnem	Bit	Instruction	Action (If Set to 1)
HS	0	High Speed	Auto load program is loaded from a high-speed device.
—	1-9	—	Reserved.
—	10-15	Device Code	Specifies last device to perform a program load.

CPU Acknowledge

DOAP CPU



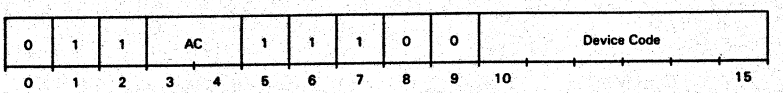
Clears one or more of bits 0-7 in the CPU status register according to the contents of the specified accumulator. The format of the specified accumulator is:



Mnem	Bit	Name	Function (If Set to 1)
POF	0	Power Fail	Clears a power Fail interrupt.
ION	1	Interrupt On	Clears (sets to 0) the Interrupt On flag.
—	2,3	—	Reserved. Must be 0.
PUP	4	Power Up %vs(1)	Clears the Power Up bit.
HLT	5	Halt	Clears the Halt bit.
—	6	—	Reserved. Must be 0.
IRQ	7	Interrupt	Clears the interrupt request bit.
—	8-15	—	Reserved. Must be 0.

Data In Status

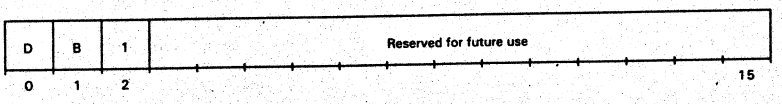
DIS *device*



Returns the status of the addressed device and places this data into the specified accumulator.

The accumulator should be specified as 1, 2, or 3. The DIS instruction uses the same operation code as the SKP instruction. If AC0 is the specified accumulator, then the DIS instruction duplicates the SKP instruction.

The information contained in the specified accumulator is in the following format for I/O devices:



D Device is done if set to one.

B Device is busy if set to one.

Memory Allocation and Protection

The memory allocation and protection (MAP) units provide the necessary hardware to use and control physical memory. In addition, the MAPs provide protection functions to maintain the integrity of a large system.

NOTE *In the following section,* MAP, *in uppercase, refers to a memory allocation and protection unit, whereas* map, *in lower case, refers to a set of memory translation functions used by a MAP unit.*

The MAP units give several users access to the resources of the computer by dividing the available memory space into blocks assigned to each user. By *user* we mean here one of the four microECLIPSE users, a data channel, or an 8086 user. Each time a user accesses memory, the MAP translates the address that the user sees — the logical address — to an address that the memory sees — the physical address. This is all transparent to the user. With software to control the priorities of the MAPs and the CPUs, several users can access the computer without being aware of the presence of the others.

The following definitions will help you understand mapping.

Logical Address. The address used by the user in all programming. The microECLIPSE logical address space is 32,768 words long and is addressed by a 15-bit address. The size of the 8086 logical address space is 524,288 words. An 8086 logical address is 19 bits wide.

Physical Address. The address used by either MAP to address the physical memory. The maximum size of the physical address space in a Model 10 or 10/SP system is 768 kilobytes and it is addressed by a 20-bit address.

Address Translation. The process of translating logical addresses into physical addresses.

Memory Space. The addresses (physical or logical) assigned to a particular user.

Page. 1024 (2000_8) words (2 kilobytes) in memory.

User Map. The set of memory address translation functions defined for a particular microECLIPSE process. They translate logical addresses to physical addresses for every memory reference.

Data Channel Map. The set of address translation functions defined by a prespecified data channel map. They translate logical addresses to physical addresses when a data channel device addresses the memory.

8086 Map. The set of address translation functions defined for the 8086 processor. They translate logical addresses to physical addresses for all 8086 memory references.

Supervisor. The part of the operating system that controls system functions such as the operation of the MAP units.

Translation Function

The primary function of a MAP unit is address translation. A map assigns each logical page of a user (again, in this context, user means either a microECLIPSE user, a microECLIPSE data channel, or the 8086 user) to a corresponding physical page. If a user's map is changed, the address space visible to the user is unchanged, but the map now translates each logical address into a different physical address.

A user's physical pages can be in any order in physical memory. This means that the supervisor can select unused pages for a new user without concern for maintaining any particular arrangement. This also allows a more complete use of the physical memory since no contiguous blocks of memory larger than 2 kilobytes are required.

Memory Sharing Function

The MAPs allow several users to use the same section of physical memory. This is useful if several users want to use a common routine such as trigonometric tables.

Mapped Mode

In mapped mode, the MAP units provide three types of maps:

User maps (microECLIPSE processor running)

Data channel map (microECLIPSE processor running during DMA transfers)

8086 map (8086 processor running)

microECLIPSE User Maps. Each microECLIPSE user requires a separate user map. The microECLIPSE MAP can hold four user maps, but only one can be enabled at any one time. This means that when four users exist, the processor specifies the user map for each and loads them into the MAP. The supervisor can then enable one as needed. If there are more than four users, new user maps must be loaded. This is simplified by the use of the LMP instruction which loads a complete map with one instruction in relatively little time.

Data Channel MAP. The data channel map can access memory without direct control from the user's program. Thus, a data channel can service a user who is not the currently executing user. This allows the I/O activity of one user to be overlapped with the execution of another user. A data channel map can be enabled or disabled at any time.

8086 Map. The 8086 MAP contains 512 map registers, each of which maps one 1-Kword logical page to one physical page. The physical pages can be located anywhere in physical memory and need not be contiguous. The microECLIPSE loads the 8086 MAP with a LMPA instruction, which operates in much the same way as the LMP instruction. The 8086 MAP is automatically enabled when the microECLIPSE transfers control to the 8086.

NOTE *If an instruction changes the current map state and the two maps involved do not contain equivalent mapping, the next instruction will be fetched from and executed in the new map state.*

Unmapped Mode

The microECLIPSE MAP can also operate in unmapped mode. In this mode, no address translation occurs, so that addresses issued by the microECLIPSE CPU reference locations in the first 32 kilobytes of physical memory (locations in physical pages 0 through 32). If, while operating in unmapped mode, the program requires access to some other part of memory, the Page 31 register can be used to accomplish this. Refer to the DOB, MAP instruction in the instruction dictionary.

MAP Address Translation

Figure 1-14 illustrates the address translation performed by the microECLIPSE MAP unit. Each microECLIPSE user's 64-kilobyte logical address space consists of 32 1024-word (2-kilobyte) pages. A program can load an address translation map consisting of 32 12-bit words for up to four users and four data channels. Each 12-bit word in a microECLIPSE map includes 10 bits that specify the physical page and two that indicate a MAP protection code. One map code bit marks the page as write protected or write enabled and the other bit specifies that the page is validity protected or unprotected.

Similarly, a program running in the microECLIPSE processor can load 512 11-bit words into the 8086 MAP. Figure 1-15 illustrates the translation performed by the 8086 MAP unit. Ten bits of the translation word in an 8086 specify the physical page and one is a protection bit. When set to 1, the protection bit and the other ten bits contained in the MAP register for a page prevent that page from being accessed by the 8086 processor. Such an attempt will cause program control to return to the microECLIPSE processor.

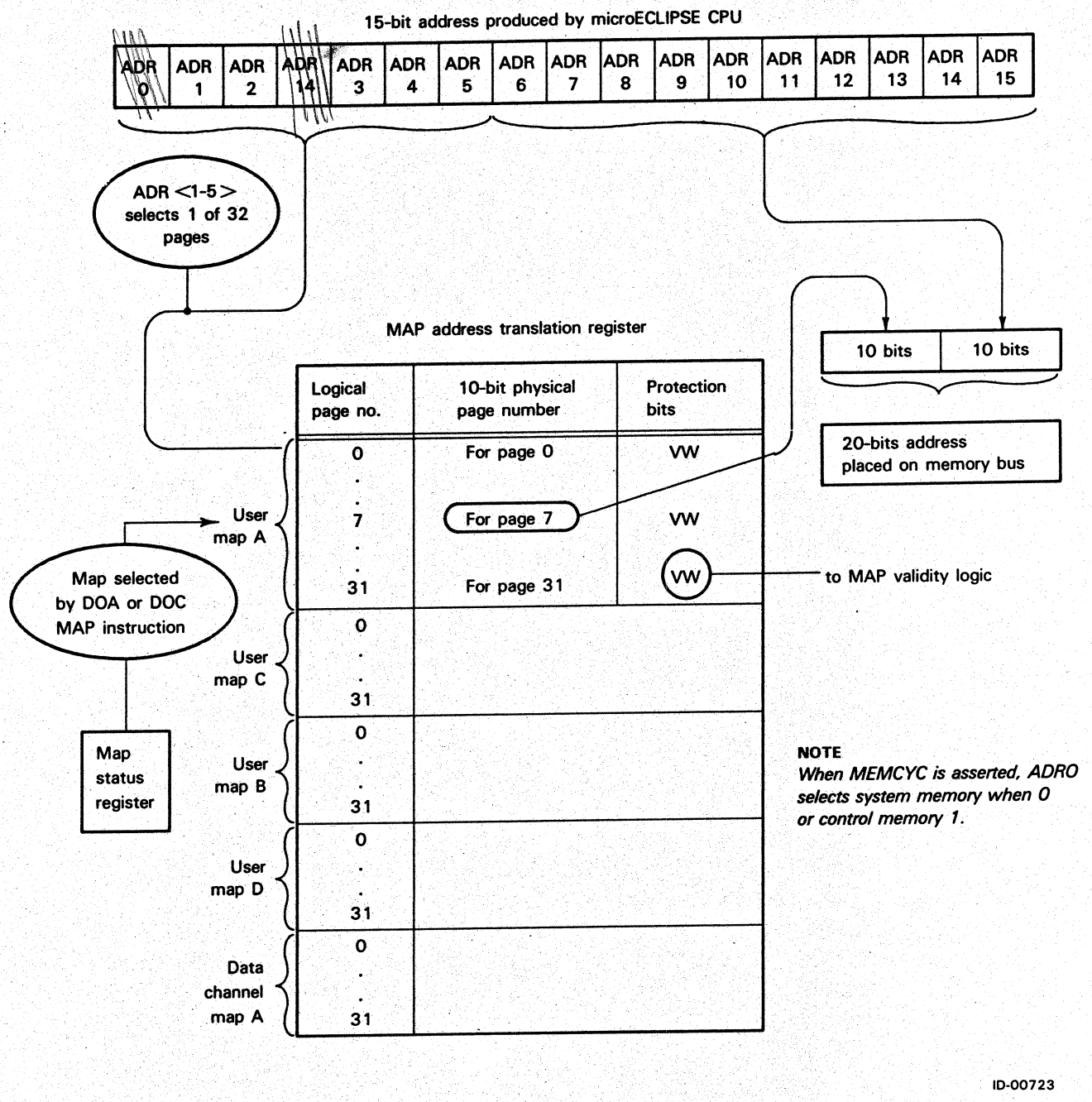


Figure 1-14 microECLIPSE MAP address translation

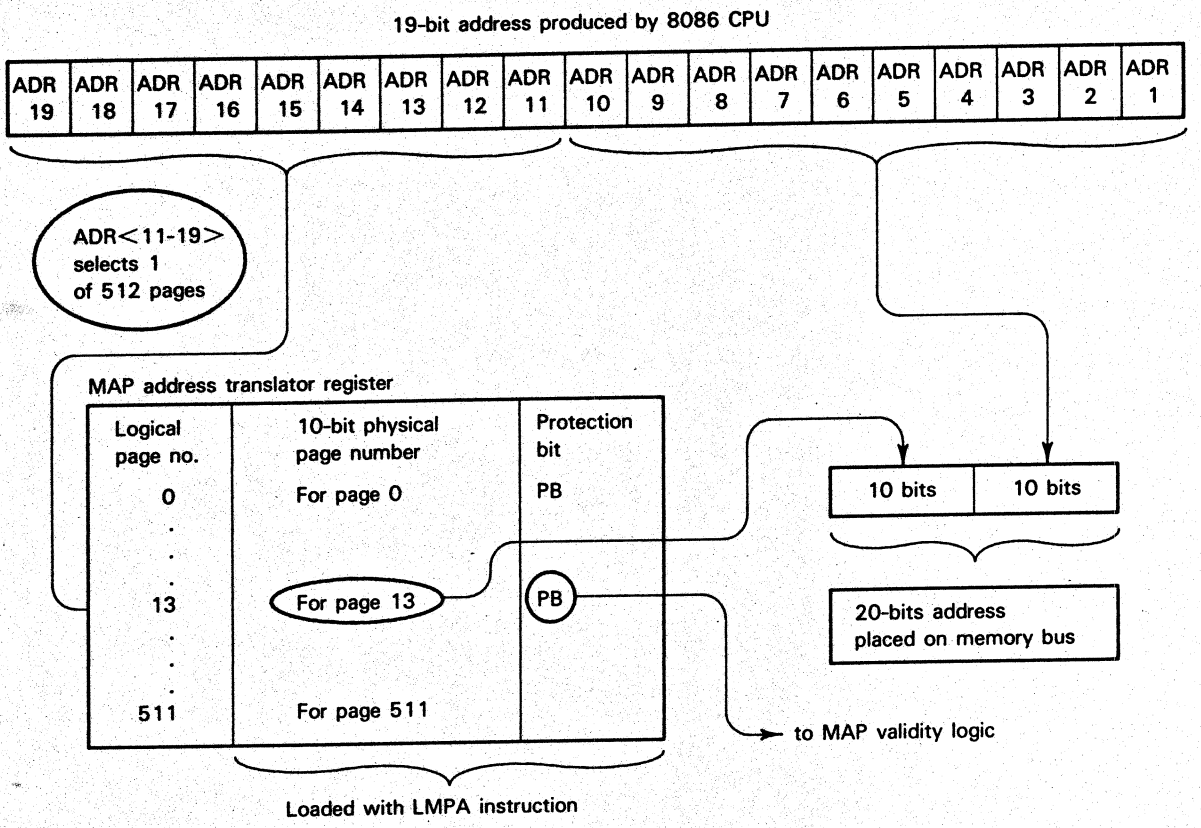


Figure 1-15 8086 MAP address translation

Emulator Trap

The emulator trap allows users to emulate undefined instructions. If an undefined instruction is encountered while operating in the mapped mode, a return block is pushed onto the stack. The program then jumps indirectly through location 11_8. This location can contain the indirect address of an emulator routine.

MAP Protection Capabilities

In addition to address translation, the MAPs provides four types of protection. The MAPs flag the nature of each protection violation and cause MAP protection faults. The four types of MAP protection are:

Validity protection (microECLIPSE and 8086 processors)

Write protection (microECLIPSE processor only)

Indirect protection (microECLIPSE processor only)

I/O protection (microECLIPSE processor only)

Validity Protection. Validity protection protects the memory space of one user — microECLIPSE user, data channel, or 8086 user — from inadvertent access by another user, thereby preserving the integrity and privacy of the user's memory space. When a user's map is specified, the blocks of logical addresses required by the user's program are linked to blocks of physical addresses. The remaining (unused) logical blocks are declared invalid to that user, and any attempt to access them causes a validity protection fault.

Validity protection is always enabled, so the supervisor's responsibility is limited to declaring the appropriate blocks of logical addresses. If a MAP unit attempts to translate an invalid logical address for the user, a MAP protection fault occurs. In this case, the state of the processor is saved and the program jumps to the programmer-supplied MAP fault handling routine. If such a fault occurs while the 8086 is running, the MAP logic generates a nonmaskable interrupt that causes the 8086 processor to pause and the microECLIPSE processor to assume control.

NOTE *No validity traps occur on microECLIPSE MAP single-cycle references, but memory is protected.*

Write Protection. microECLIPSE memory space can be write-protected, but 8086 can only be validity protected. Write protection allows users to read the protected memory locations, but not to write into them. Therefore, the integrity of common areas of memory is protected. An attempt to write into a write protected area of memory will cause a protection fault.

Blocks of logical memory may be write protected when a microECLIPSE map is specified. Write protection can be enabled or disabled at any time by the supervisor. For example, a set of trigonometric functions is stored in a section of memory accessible to all microECLIPSE users. This section should be write protected so that users can read the functions but cannot change them.

Indirect Protection. Indirect protection applies only to the microECLIPSE processor. It allows the supervisor to ensure that the microECLIPSE CPU will not be placed in an indirection loop. If the CPU is in an indirection loop without indirect protection it would be unable to proceed with any further instructions, thus effectively halting the system.

With indirect protection enabled, a chain of 16 indirect references causes a MAP protection fault. Indirect protection can be enabled or disabled at any time by the supervisor.

I/O Protection. I/O protection protects the I/O devices in the system from unauthorized access. If a user with I/O protection enabled attempts to execute an I/O instruction, an I/O protection fault will occur. Enabling I/O protection will prevent execution of the *Load Map* (LMP) instruction. I/O protection can be enabled or disabled at any time. (Recall that the 8086 must pass control to the microECLIPSE processor to perform I/O.)

MAP Protection Faults

When a microECLIPSE user violates one of the enabled types of protection, a protection fault occurs as follows:

The current user map is disabled.

A 5-word return block is pushed onto the system stack. The program counter pushed will point to the word following the instruction which caused the MAP fault. But note that the PC *will not* be valid after a validity protection fault.

Control is transferred to the protection fault handler by an indirect jump through memory location 3 which should contain the fault handler routine address.

The protection fault handler routine may determine the type of fault that occurred, using the *Read Map Status* (DIA MAP) instruction, before taking the appropriate action.

Any attempt to read the contents of a physical address at which memory does not exist will result in undefined data being returned.

Any attempt to write to a nonexistent physical address will have no effect and will not produce an error.

Load Effective Address Mode

The microECLIPSE *Load Effective Address* (LEF) instruction has the same format as I/O instructions. The microECLIPSE MAP has a LEF mode bit which determines whether an I/O format instruction will be interpreted as an I/O or a LEF instruction. When the MAP is enabled and the LEF mode bit is one (LEF mode enabled), all I/O format instructions are interpreted as *Load Effective Address* (LEF) instructions. When the LEF mode bit is zero, all I/O format instructions are interpreted as I/O instructions.

The *Load Effective Address* (LEF) instruction is very useful for loading a constant into an accumulator. In addition, a user operating in the LEF mode is denied access to any I/O devices, because all I/O and LEF instructions are interpreted as LEF instructions in this mode. This means that LEF mode can be used for I/O protection. The *Load Map* (LMP) instruction, however, does not use the I/O format and therefore can still be executed.

Enabling Screen Buffer Access

Monochrome monitor screen buffer access is enabled by the use of the *Read* and *Write Control Memory Status* instructions (RHYP and WHYP). These instructions allow a user program to map the physical pages that contain the screen buffer (pages $1760\text{-}1771_8$) to pages within the program's logical address space. They do so by setting the Enable Buffer Write bit in the control memory status register to 1. Ordinarily, an attempt to map to physical pages greater than 1757 (the top of the optional color monitor's screen buffer) would fail. With the Enable Buffer Write bit set, this mapping becomes possible.

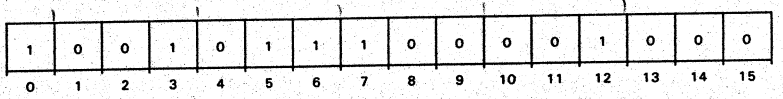
MAP Instructions

The MAP instructions control the actions of the MAPs. They are used by the supervisor program running in the microECLIPSE processor to change the mapping functions or to check the status of the various maps — microECLIPSE user or data channel maps or the 8086 map.

The MAP instructions were shown in Table 1-22. All except *Load Map* (LMP) are in I/O format using the device mnemonic MAP for the microECLIPSE Map or the device mnemonic ATP for the 8086 Map. The MAP instructions are presented below: first instructions to be executed only on the microECLIPSE CPU, then the instructions for the 8086 CPU.

Load microECLIPSE Map

LMP 1 1 3 4 1 0



Loads successive words from memory into the microECLIPSE MAP.

Words are loaded in consecutive, ascending order according to their addresses.

Three accumulators affect the LMP instruction:

AC0 must contain zero for the LMP.

AC1 contains an unsigned integer which is the number of words to be loaded into the MAP.

AC2 contains the address of the first word to be loaded. If bit zero is one, the instruction follows the indirection chain and places the resulting effective address into AC2.

AC3 is ignored and its contents remain unchanged.

For each word loaded, the instruction decrements the number in AC1 by one and increments the address in AC2 by one.

Upon completion of the LMP instruction:

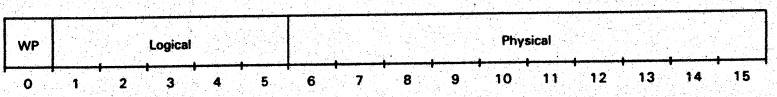
AC0 remains unchanged.

AC1 contains zero.

AC2 contains the address of the word following the last word loaded.

The words loaded into the microECLIPSE MAP define the address translation functions for the various microECLIPSE user and data channel maps. The contents of the MAP field (bits 6-8) of the MAP status register determine which map is affected by the LMP instruction. You can alter this field by using either the *Load Map Status* (DOA *ac*,MAP) or the *Initiate Page Check* (DOC *ac*,MAP) instruction.

The format of each word loaded into the microECLIPSE MAP is



Bits	Name	Contents or Function
0	Write Protect	Write protect for user maps. Map faults may not occur during memory references initiated by data channel.
1-5	Logical	Logical page number
6-15	Physical	Physical page number

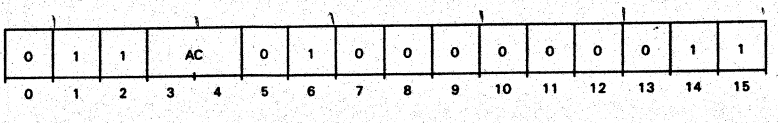
NOTE *To declare a logical page invalid, set the Write Protect bit to one and all of bits 6-15 to one.*

The LMP instruction is interruptible in the same manner as the BAM instruction. If you issue this instruction while in mapped mode, with I/O protection enabled, the map and accumulators are not altered and a MAP fault occurs.

If the LMP instruction alters the translation of the page indicated by the program counter for the next instruction fetch, this causes the instruction to be fetched from the new translation.

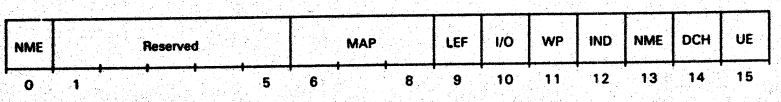
Load microECLIPSE MAP Status

DOA MAP 1



The contents of the specified AC are placed in the microECLIPSE MAP status register. The contents of the AC remain unchanged.

The format of the specified AC is

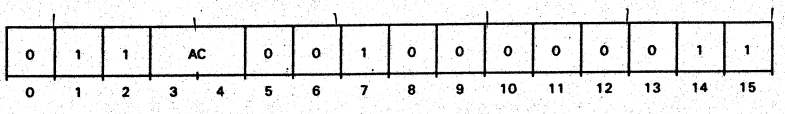


Bits	Name	Contents or Function
0,13	NME	Depending on the bit settings, the next user map enabled will be that for: Bit 0 Bit 13 User Enabled 0 0 A 0 1 B 1 0 C 1 1 D
1-5	—	Reserved for future use.
6-8	MAP	Specifies which map will be loaded by the next **LMP** instruction as follows: Bit 6 Bit 7 Bit 8 User Enabled 0 0 0 A 0 0 1 C 0 1 0 B 0 1 1 D 1 0 0 Data Channel A 1 0 1 Reserved 1 1 0 Reserved 1 1 1 Reserved
9	LEF	If one, the **LEF** instruction will be enabled for the next user.
10	I/O	If one, I/O protection will be enabled for the next user.
11	WP	If one, write protection will be enabled for the next user.
12	IND	If one, indirect protection will be enabled for the next user.
14	DCH Enable	If one, the mapping of data channel addresses will be enabled immediately after this instruction.
15	User Enable	If one, mapping of CPU addresses will commence with the first memory reference after the next indirect reference or return type instruction (**POPB, POPJ, RTN, RSTR**).

NOTE *If the* Load Map Status *LMP instruction sets the User Enable bit to one, the interrupt system is inhibited, and the MAP waits for an indirect reference or a return-type instruction. Either event releases the interrupt system and allows the MAP to begin translating addresses (using the user map specified by bits 0 and 13 of the MAP status register). Address translation resumes after the first level of the next indirect reference; or after a POPB, POPJ, RTN, or RSTR instruction.*

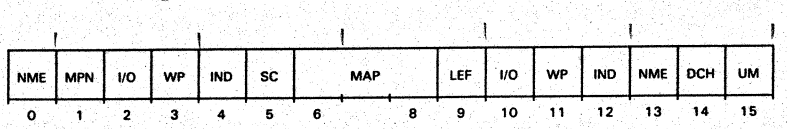
Read microECLIPSE MAP Status

DIA MAP 1 $0\begin{Bmatrix}6\\7\end{Bmatrix}\begin{Bmatrix}0\\4\end{Bmatrix}403$



Reads the status of the current map.

Places the contents of the microECLIPSE MAP status register in the specified AC. The previous contents of the AC are overwritten. The format of the information placed in the specified AC is as follows:

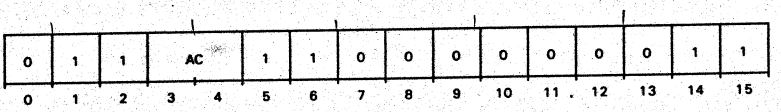


NOTE *The IORST instruction will clear bits 0 to 5 and 13 to 15 of the MAP status register. IORST also turns off the MAP.*

Bits	Name	Contents or Function
0,13	NME	Next MAP enabled. Depending on the bit settings, the last **DOA MAP** instruction enabled: Bit 0 Bit 13 User Enabled 0 0 A 0 1 B 1 0 C 1 1 D
1	MPN	MAP state — 1 indicates mapping enabled.
2	I/O	If one, the last protection fault was an I/O protection fault.
3	WP	If one, the last protection fault was a write protection fault.
4	IND	If one, the last protection fault was an indirect protection fault.
5	SC	If one, the last map reference was a **NIOP MAP** instruction.
6-8	MAP	Specifies which map was loaded by the last **LMP** instruction as follows: Bit 6 Bit 7 Bit 8 User Enabled 0 0 0 A 0 0 1 C 0 1 0 B 0 1 1 D 1 0 0 Data Channel 1 0 1 Reserved 1 1 0 Reserved 1 1 1 Reserved
9	LEF	If one, the **LEF** instruction was enabled for the last user.
10	I/O	If one, I/O protection was enabled for the last user.
11	WP	If one, write protection was enabled for the last user.
12	IND	If one, indirect protection was enabled for the last user.
14	DCH	If one, the mapping of data channel addresses has been enabled.
15	UM	User mode. If one, the last I/O interrupt occurred while in user mode (map enabled).

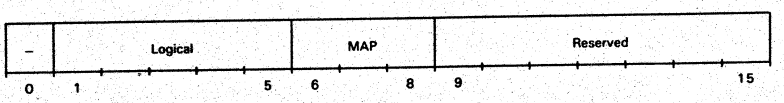
Initiate Page Check

DOC*[f]* *ac*,MAP



The contents of the specified AC are transferred to the MAP feature for later use by the DIC *ac* MAP instruction or the LMP instruction. The contents of the specified AC remain unchanged.

The format of the specified AC for the microECLIPSE MAP page check is:



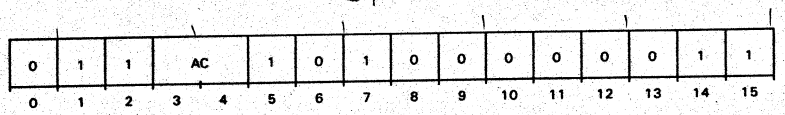
Bits	Name	Contents or Function
0	—	Reserved for future use.
1-5	Logical	Number of the logical page for which the Check Page is requested.
6-8	Map	Specify which map should be used for check as follows:

Bit 6	Bit 7	Bit 8	User Enabled
0	0	0	A
0	0	1	C
0	1	0	B
0	1	1	D
1	0	0	Data Channel
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Reserved

9-15	—	Reserved for future use.

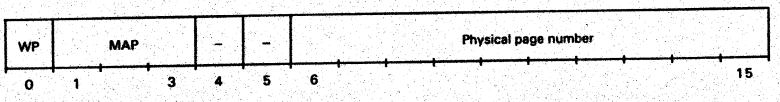
Page Check

DIC*[f]* *ac*,MAP



The number of the physical page which corresponds to the logical page specified by the preceding DOC MAP instruction is placed in bits 6 to 15 of the specified AC. Places additional information about the correspondence in bits 0 or 0-5. The previous contents of the AC are overwritten.

The format of the information placed in the specified AC for the microECLIPSE MAP is



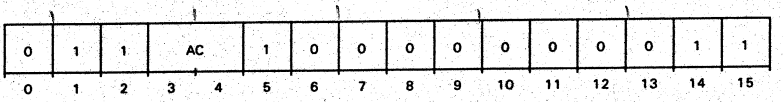
Bits	Name	Contents or Function
0	WP	The write protect bit for the logical page which corresponds to the physical page specified by bits 6-15.
1-3	MAP	The map which was used to perform the translation between logical page number and physical page number as follows:

Bit 1	Bit 2	Bit 3	User Enabled
0	0	0	A
0	0	1	C
0	1	0	B
0	1	1	D
1	0	0	Data Channel
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Reserved

4-5	—	Reserved for future use.
6-15	Physical page	The number of page which corresponds to the logical page given in the preceding **DOC MAP** instruction.

Map Supervisor Page 31

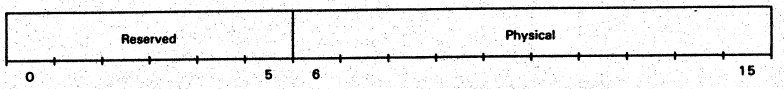
DOB*[f]* *ac*,MAP



Specifies that mapping take place for a single page of an unmapped address space. This instruction is valid for the microECLIPSE processor *only*. Mapping is always done for locations 76000_8 though 77777_8 (logical page 31). This is the only page which can be mapped when in unmapped address space. You can use this instruction to access a page of a user's memory space when in unmapped mode. The MAP supervisor Page 31 instruction can only be used with the MAP off.

Bits 6-15 of the specified AC are transferred to the MAP feature. These bits specify a physical page number to which logical page 31 will be mapped when in the supervisor mode.

The contents of the specified AC remain unchanged. The format of the specified AC is



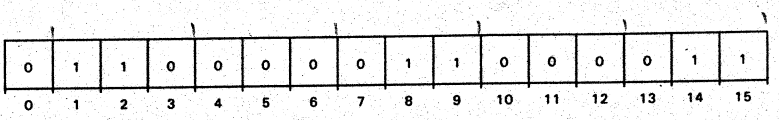
Bits	Name	Contents or Function
0-5	—	Reserved for future use.
6-15	Physical page	Number of physical page to which logical page 31 should be mapped when in supervisor mode.

NOTE *If supervisor page 31 translation is altered while instructions are being fetched through supervisor page 31, instructions will be fetched from the new translation. IORST resets logical address translation to physical address translation.*

Map Single Cycle

Disable User Mode

NIOP MAP 060303



This instruction is valid for the microECLIPSE processor *only*. Its effect depends upon the mode from which it is issued.

NOTE *The interrupt system is disabled from the beginning of the* MAP Single Cycle *instruction until after the next LDA, ELDA, STA, or ESTA instruction.*

From user (mapped) mode. If the LEF mode and I/O protection are disabled, the NIOP instruction turns off the MAP. All subsequent memory references are unmapped until the map is reactivated with a *Load Map Status*, DOA MAP instruction.

From the unmapped mode. The user map is enabled for one memory reference. The first memory reference of the next LDA, ELDA, STA, or ESTA instruction is mapped. After the memory cycle is mapped, the user map is again disabled.

For example, if AC2 contains 405_8 and the following instruction sequence is issued:

```
NIOP MAP ;MAP SINGLE CYCLE

LDA 3,2,2
```

the logical address 407_8 will be mapped using the last enable map (specified by bits 0 and 13 of the MAP status register at the time of the memory reference). The word contained in the corresponding physical location will be placed in AC3.

However, if the following instruction sequence is issued:

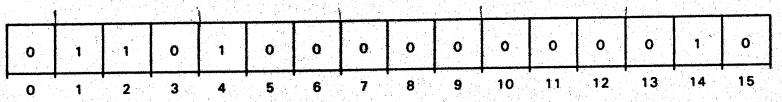
```
NIOP MAP ;MAP SINGLE CYCLE

LDA 3,@2,2
```

the logical address 407_8 will be mapped using the user map for the last enabled user. The contents of the corresponding physical location will be used as the first level of an indirection chain. The next memory cycle, which is the second level of the indirection chain, will not be mapped.

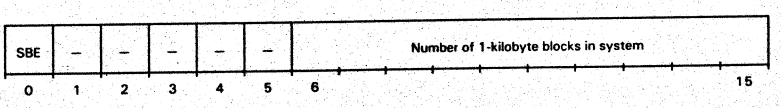
Read Control Memory Status

RHYP



64002

AC0 must contain 14_8 when this instruction is issued. (This selects the status register in control memory.) Loads the contents of the control memory status register into AC1. The format of AC1 will be:

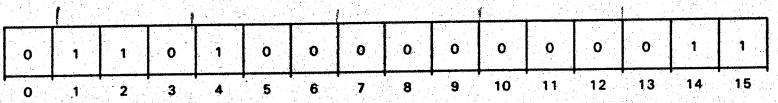


Bits	Name	Contents or Function
0	SBE	When 1, the mapping of pages in logical address space to pages in the monochrome screen buffer (physical pages $1760\text{-}1771_8$) is enabled.
1-5	—	Reserved for future use.
6-15	Number of 1-kilobyte blocks in system	The number of blocks of memory found in system memory by the powerup self test.

Use the *Read Control Memory Status* instruction before a *Write Control Memory Status* instruction in order to preserve bits 6-15 of the status register.

Write Control Memory Status

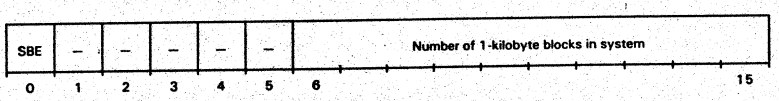
WHYP



64003

Loads the contents of AC1 into the control memory status register.

AC0 must contain 14_8 when this instruction is issued. (This selects the status register in control memory.) The format of AC1 is:



ACCESS TO EMULATOR - PROGRAM

EHYP 64004

Bits	Name	Contents or Function
0	SBE	When 1, enables the mapping of pages in logical address space to pages in the monochrome screen buffer (physical pages $1760\text{-}1771_8$).
1-5	—	Reserved for future use.
6-15	Number of 1-kilobyte blocks in system	The number read from the control memory status register with the preceding RHYP instruction. This should not be changed.

Use this instruction to enable access to the monochrome monitor screen buffer by setting the Screen Buffer Enable bit to 1. Do not change any other bits in the control memory status register. The monitor screen buffer is located in physical memory pages $1760\text{-}1771_8$.

Physical page 1771 is the last (highest) physical page that a user program can access, and this can only be done when bit 0 of the control memory status register is 1. If your program tries to assign a logical page to a physical page number greater than 1771, that logical page is assigned instead to physical page 1771. If bit 0 of the control memory status register is 0, then your program cannot access physical pages higher than 1757 (the last page of the optional color monitor's screen buffer). In this case, attempts to map a logical page to a physical page number greater than 1757 result in the assignment of the logical page to physical page 1757.

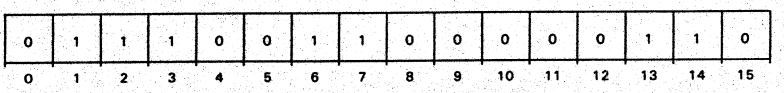
NOTE *Whenever a system reset is performed (for example by execution of an* I/O Reset *instruction) bit 0 of the control memory status register will be reset to 0 and screen buffer memory will no longer be accessible to user memory.*

The following instructions pertain only to the 8086 processor.

Load 8086 Map

LMPA

DIB ATP



Loads the 8086 MAP with the translation for the logical block specified. The logical-to-physical translation, as well as page protection, is specified by a two-word sequence. The first word is the logical address to be translated; the second word is the physical block to which it is mapped, along with its protection status. The two words are called the *translation pair.*

Words are loaded in consecutive, ascending order according to their addresses.

Two accumulators affect the LMPA instruction:

AC1 contains an unsigned integer which is the number of logical block/ physical block word pairs to be loaded into the 8086 map.

AC2 contains the address of the first word to be loaded. If bit zero is one, the instruction follows the indirection chain and places the resulting effective address into AC2.

AC0 and AC3 are ignored and their contents remain unchanged.

For each translation pair loaded, the instruction decrements the number in AC1 by one and increments the address in AC2 by two.

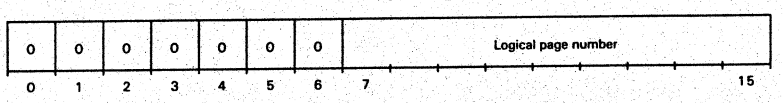
Upon completion of the LMPA instruction:

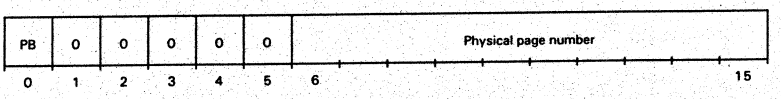
AC1 contains zero.

AC2 contains the address of the word following the last translation pair loaded.

The words loaded into the 8086 MAP define the address translation functions for the 8086 processor. The first word of the translation pair specifies the logical page to be mapped; the second word is the physical page to which it is mapped.

The format of the word pairs loaded into the 8086 MAP is





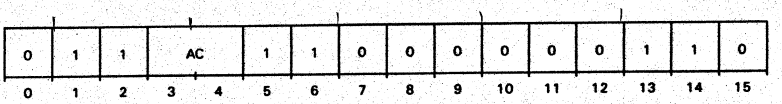
Word 1	Bits	Name	Contents or Function
	0-6	—	Must be zero
	7-15	Logical	The logical page number to be mapped.
Word 2			
	0	Protect bit	Write protects designated page when set to 1.
	1-5	—	Must be zero.
	6-15	Physical	The physical page number to be mapped to.

NOTE *To declare a logical page invalid, set the Protect bit to one and all of bits 6-15 in the physical page word to one.*

NOTE *The LMPA instruction is interruptible in the same way as the LMP instruction. AC2 will point to the next two-word pair to be translated.*

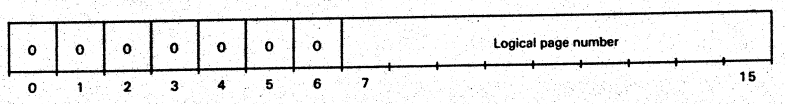
Initiate Page Check

DOC*[f]* *ac*,ATP



The contents of the specified AC are transferred to the MAP feature for later use by the DIC *ac* MAP instruction.

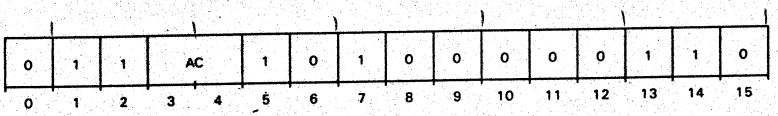
The format of the specified AC for the 8086 MAP page check is:



Bits	Name	Contents or Function
0-5	—	Reserved for future use. Must be zero.
6-15	Logical	Number of the logical page for which the next 8086 Check Page is requested.

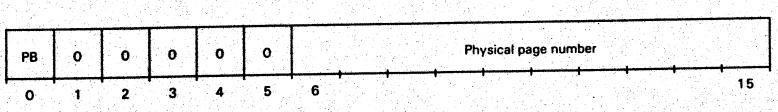
Page Check

DIC*[f]* *ac*,ATP



The number of the physical page which corresponds to the logical page specified by the preceding DOC ATP instruction is placed in bits 6 to 15 of the specified AC. Places additional information about the correspondence in bits 0 or 0-5. The previous contents of the AC are overwritten.

The format of the information placed in the specified AC for the 8086 MAP is



Bits	Name	Contents or Function
0	PB	The protection bit for the physical page specified by bits 6-15. This bit is 0 for no validity protection. If it is 1 and bits 6-15 are 1, the corresponding logical page is validity protected.
1-5	—	Reserved for future use. Will be set to 0.
6-15	Physical	The physical page number mapped to.

NOTE *If all physical page bits including the write protect bit are one, then the logical page is validity protected.*

Programming the MAPs

To manage address translation a memory supervisor routine must maintain information about memory usage. It must keep track of such things as which physical pages are currently allocated to which processes, which pages the various processes can access, which pages processes need to share, which processes currently have map tables stored in the translators.

The supervisor must also maintain a map table in memory or on disk for itself and each process that requires address translation. Normally, the supervisor reserves the user A map table for itself. When setting up the map table for a process, the supervisor should invalidate all pages not needed by the process. This ensures that mistaken references to these unneeded pages do not result in unwanted access to memory used by other processes. In other words, if a process only needs 12 pages (24 Kbytes), then logical pages 13 through 30 should be invalidated. Invalidate a page by setting the write protect bit and the physical page bits to 1 in the map table entry for the page. Note that page 1777 cannot be write-protected without also validity protecting it.

Loading and Enabling Map Tables

Before a process becomes active in a system using address translation, the memory supervisor must make sure the map table for the process is stored in the MAP to be used (microECLIPSE or 8086). Load a map table into the microECLIPSE MAP from memory with the following instruction sequences:

1. *Load MAP Status* instruction (DOA) *ac*,MAP to select the map table to be loaded.
2. *Load Map* instruction (LMP) to start storing the map table.

Load a map table into the 8086 MAP with the LMPA instruction.

The *Load MAP* instructions (LMP and LMPA) are interruptible and resumable.

While a user or DCH map table is being loaded, a user or data channel address translation can occur using another map table.

Before a microECLIPSE user or data channel process can use the new map table, the supervisor must enable the map table together with user or data channel address translation. Do this with another *Load MAP Status* instruction (DOA *ac*,MAP). When a *Load MAP Status* instruction (DOA *ac*,MAP) instruction enables user translation (sets bit 15 to 1 in the translator's status register), then the interrupt system is disabled and the translator waits for an indirect memory reference. After the *first* level of the next indirect reference occurs, the interrupt system is reenabled and address translation starts using the enabled map table.

The supervisor can enable data channel address translation with a *Load MAP Status* instruction (DOA *ac*,MAP). However, it cannot use this instruction to enable a map table for the data channel process. Instead, it must use an I/O instruction for the specific device associated with the data channel process. This instruction is usually one of the instructions used to set the parameters of a data channel transfer for the device.

When a microECLIPSE user or data channel map table is enabled that alters the translation of the logical page indicated by the program counter for the next instruction fetch, then the instruction is fetched using the new translation. Likewise, when a DCH map table is loaded that alters the translation of the logical page indicated by the DCH address register for the next DCH transfer, then this transfer uses the new translation.

NOTE *Unpredictable results happen if a user memory write is done when all the following conditions occur together:*

The write is to a logical page that is different than the logical page currently being translated, *and*

The logical page for the write translates into the same physical page as the logical page currently being translated,
and

The physical address for the write is one or two greater than the current program counter.

NOTE *MAP instructions* can *be executed in mapped mode if I/O protection and LEF mode are disabled for the user. When executed in mapped mode, the* Read Map Status, Initiate Page Check, *and* Page Check *instructions will return the desired information without changing the map. The* Map Single Cycle *instruction will disable the user map after the next memory reference.*

Enabling only LEF mode will convert all I/O instructions (including MAP instructions) to LEF instructions. The *Load Map* instruction, however, does not use the I/O format and therefore can still be executed. Enabling I/O protection will prevent execution of the *Load Map* instruction. Bit 1 of the MAP status register indicates the state of the MAP. If bit 1 is set to one, the MAP is enabled. If bit 1 is set to zero, the MAP is disabled. For further details, refer to the DIA MAP instruction.

Power-up Response

At power up, the user maps (both microECLIPSE and 8086) and the data channel maps are undefined, the MAP is in unmapped mode, and unmapped logical page 31 is mapped to physical page 31. This means that all addresses issued by the CPU reference physical pages 0 to 32. While operating in unmapped mode, page 31 register may be used to access some other part of memory. Refer to the DOB MAP instruction in "Instruction Set" of this section.

After an *I/O Reset* IORST, the MAP is in unmapped mode, the data channel maps are disabled, and unmapped logical page 31 is mapped to physical page 31.

Parity Checking

Model 10 and 10/SP memory logic generates and appends a parity bit to each byte of data written to memory. In addition, the memory logic checks this bit for each byte read from memory. Detection of an incorrect parity bit may, at the user's request, cause an interrupt request.

Programmable Elements

From a programming point of view, the parity checking facility consists of the following major elements:

1. Control Register
2. Fault Code Register
3. Fault PC Register
4. Done Flag

Your program directs the facility by manipulating these major elements, using the following interface instructions together with the START device flag commands:

1. Enable Parity Checking (DOA)
2. Read Parity Fault PC (DIA)
3. Read Parity Fault Code (DIB)

The program also uses *I/O Reset* (IORST) instruction to manipulate the state of some elements.

Programming Summary

The following programming summary provides general facility specifications, the accumulator formats for the three facility instructions, and explains the facility's response to the START, CLEAR, and IOPLS flag commands and the *I/O Reset* instruction (IORST).

Table 1-29 Programming summary: parity checking specifications

Mnemonic	PAR
Device code	02 octal
Priority mask bit	None

Table 1-30 Programming summary: START, CLEAR, IOPLS, and IORST functions

f = S	Sets the Done flag to 0.
f = C	No effect.
f = P	No effect.
IORST	Sets the Done flag to 0. Also sets the parity control bits to zero (disables parity checking).

Registers and Flags

The program-accessible registers of the parity checking facility include the control, fault PC, and fault code registers. Its program-accessible flag is the Done flag.

Control Register. The control register stores the 2-bit code sent to the parity facility by the program to control the parity facility. The program loads this register using a *Enable Parity Checking* instruction (DOA). The program sets the two control bits to 0 using the *I/O Reset* instruction (IORST). Refer to the *Enable Parity Checking* instruction for control details.

Fault PC Register. The fault PC register stores the most significant 12 bits of the logical address contained in the microECLIPSE program counter when a parity error causes a nonmaskable interrupt (NMI). The program reads this register using a *Read Parity Fault Address* instruction (DIA).

Fault Code Register. The fault code register stores a 2-bit code specifying whether a parity error has occurred. The program reads this register using a *Read Parity Fault Code* instruction (DIA). Refer to this instruction for error code details.

Done Flag. The Done flag, when set to 1, initiates an interrupt request to the CPU (unless interrupts are disabled) and specifies that the parity facility has detected a parity error during the last memory access. The program sets the Done flag to 0 using the *I/O Reset* instruction IORST or the Start flag command. A program can test the Done flag using the *I/O Skip* instruction IOSKP addressed to the parity checking facility.

I/O Instruction Set Three programmed I/O instructions enable you to program the parity checking facility, to enable/disable parity checking, and to determine where a parity error occurred. These instructions:

Load the two control bits from a specified CPU accumulator into the control register (DOA).

Read a fault PC value from the fault PC register into a specified CPU accumulator (DIA).

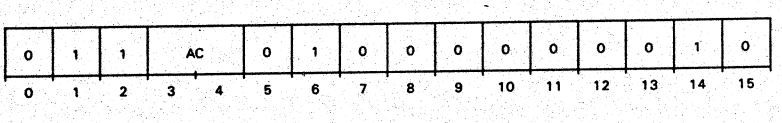
Read a fault code from the fault code register into a specified CPU accumulator (DIB).

Assembly language coding and instruction bit patterns are shown for these instructions. The device code field of the instruction is shown with the standard parity checking facility device code, 02_8. Mnemonics PAR or 02_8 used in the assembly language coding, address the parity checking facility.

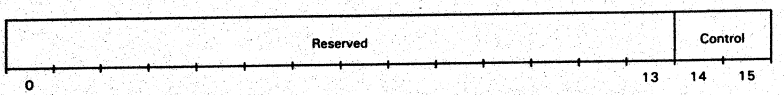
The START device flag command can be specified in the *F* field of programmed I/O instructions. Refer to the "Programming Summary" for the effect that this flag command and the *I/O Reset* (IORST) instruction have on the parity checking facility.

Enable Parity Checking Mode

DOA 2



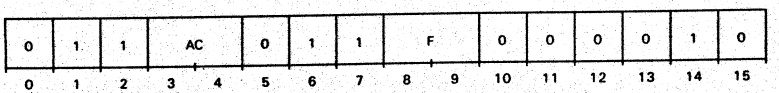
The parity checking facility is enabled according to the setting of bits 14 and 15 of the specified AC. The contents of bits 0 to 13 of the specified AC are not changed. The format of AC is



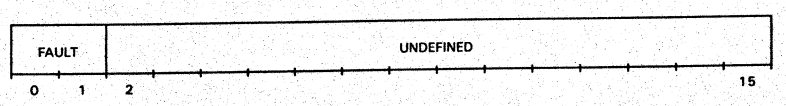
Bits	Name	Contents or Function
0-13	—	Reserved for future use.
14,15	Control	Control the parity checking feature as follows:
		00 Disable checking; write valid (even) parity check field.
		01 For diagnostic purpose only. Do not use this setting. (Disables checking; writes odd parity field.)
		10 Disable checking; write valid (even) parity check field.
		11 Enable parity checking; interrupt on parity error; write even parity check field.

Read Parity Fault Code

DIB 2



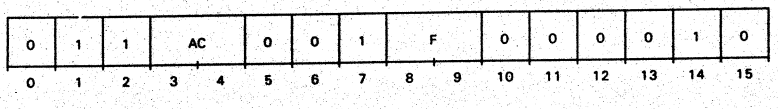
A 2-bit error code is placed in bits 0 and 1 of the specified AC. Bits 2 to 15 of the specified AC are undefined. The format for the specified AC is



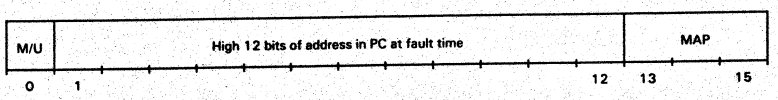
Bits	Name	Contents or Function
0,1	Fault	An error code specifying that a parity error occurred: 00 No error 11 Parity error occurred
2-15	—	Undefined.

Read Parity Fault PC

DIA 2



The 12 most significant bits of the logical address contained in the program counter when the parity error occurred are placed in bits 1 to 12 of the specified AC. Bits 13 to 15 indicate which map was enabled at the time. The previous contents of AC are overwritten. The format of the specified AC is



Bits	Name	Contents or Function
0	M/U	When 1, mapping was enabled when the parity fault occurred. When 0, mapping was disabled.
1-12	—	Most significant 12 bits of address in the PC when the parity fault occurred.
13-15	MAP	Specifies which map was enabled, if the fault occurred while in mapped mode: Bit 13 Bit 14 Bit 15 User Enabled 0 0 0 A 0 0 1 C 0 1 0 B 0 1 1 D

Four additional programmed I/O instructions (two addressed to either the receiver or transmitter section of the parity checking facility and three addressed to the CPU–device code 77_8) allow the CPU to:

1. Alter program flow determined by the state of the Done flag (IOSKP).
2. Issue a START flag command without a programmed input/output transfer (NIO).
3. Identify an interrupting source (INTA).
4. Initialize the interface (IORST).

The latter instructions are fully described in *16-bit Real-Time ECLIPSE Assembly Language Programming*.

Power-up Response

After power up or an IORST instruction, the parity checking facility is disabled and a valid (odd) parity check field is written.

The 8086 Attached Processor

Communication between the microECLIPSE and the 8086 processors is achieved by a Data General standard I/O interface. As explained in the earlier section of this chapter on memory management, the 8086 has its own MAP unit, which is controlled exclusively by the microECLIPSE processor.

Two registers and two flags are used to pass control to the 8086, specify the first address to be used by the 8086 processor, determine its status, and to reset it. The 8086 processor can return control to the microECLIPSE processor by executing an *Out* instruction.

The programmable elements of the attached processor (ATP) interface are:

1. ATP Status Register
2. ATP Interrupt Address Register
3. Busy Flag
4. Done Flag

Your program directs the ATP interface by manipulating these major elements, using the following microECLIPSE interface instructions together with *I/O Skip* instructions and the Start device flag commands:

Set ATP Status (DOA)

Read ATP Status (DIA)

Set ATP Interrupt Address (DOB)

The program also uses the *I/O Reset* (IORST) instruction to manipulate the state of some of the elements.

The program running in the 8086 processor uses the 8086 *Out* instruction as an I/O system call.

Programming Summary

The following programming summary provides general facility specifications, shows the accumulator formats for the three microECLIPSE ATP interface instructions, the formats for the 8086 *Out* instructions, and explains the ATP interface's response to the Start, Clear, and Pulse flag commands and the *I/O Reset* instruction.

Table 1-31 Programming summary: attached processor, general specifications

Mnemonic	ATP
Device code	06 octal
Priority mask bit	None

Table 1-32 Programming summary: Start, Clear, IOPLS, and IORST Functions

f = S	Sets the Interrupt Request bit and Done flag to 0 and the Busy flag to 1. Idles the microECLIPSE processor and starts the 8086 processor.
f = C	Sets the Interrupt Request bit, the Busy flag, and the Done flag to 0. Performs a hard reset of the 8086 processor. Clears the interrupt status in the ATP status register.
f = P	Sets the Busy flag and the Done flag to 0.
IORST	Sets the Interrupt Request bit, the Busy flag, and the Done flag to 0. Performs a hard reset of the 8086 processor. Clears the interrupt status in the ATP status register. Clears any pended 8086 interrupt.

Registers and Flags

The program-accessible registers of the ATP interface include the ATP status register and the ATP interrupt address register. Its program-accessible flags are the Busy flag and the Done flag.

ATP Status Register. The ATP status register stores the Interrupt Request bit sent to the ATP interface which enables the 8086 to request interrupts of the microECLIPSE processor. It also stores the states of the ATP interface's Busy and Done flags and two bits that indicate to the microECLIPSE processor the condition under which it received control. The program loads this register using a *Set ATP Status* instruction (DOA) and reads it using a *Read ATP Status* instruction (DIA). Refer to the descriptions of these instructions for control details.

ATP Interrupt Address Register. The ATP Interrupt Address Register stores the 8-bit interrupt vector sent to the ATP interface by the microECLIPSE processor. The program loads this register using a *Load ATP Interrupt Vector* instruction (DOB). Refer to the description of this instruction for control details.

Busy Flag. The Busy flag, when set to 1, indicates to the microECLIPSE processor that the 8086 processor has been paused due to a pending interrupt and that interrupts are enabled. The 8086 processor either continues execution at the previous PC address or responds to an 8086 interrupt request using the previously written ATP interrupt address as an interrupt vector.

Done Flag. The Done flag, when set to 1, initiates an interrupt request to the microECLIPSE CPU (unless interrupts are disabled, or the Interrupt Request bit in the ATP status register is set to 1). The program sets the Done flag to 0 using either the *I/O Reset* instruction (IORST) the Start flag command, or the Clear flag command. A program can test the Done flag using the *I/O Skip* instruction (SKP) addressed to the ATP interface.

I/O Instruction Set Three programmed I/O instructions enable you to program the ATP interface to enable the ATP to interrupt the microECLIPSE processor, determine ATP status, and set up an ATP interrupt. These instructions

Load the interrupt status bit from a specified accumulator into the ATP register (DOA).

Read five ATP status bits into a specified accumulator (DIA).

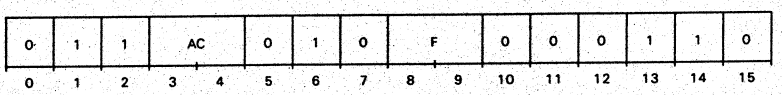
Load an 8-bit interrupt vector into the ATP interrupt address register and pend an 8086 interrupt (DOB).

Assembly language coding and instruction bit patterns are shown for these instructions. The device code field of the instruction is shown with the standard ATP interface device code 06_8. Mnemonics ATP or 06_8 used in assembly language coding address the attached processor (8086) interface.

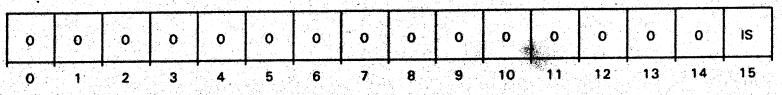
The Start device flag command can be specified in the *f* field of programmed I/O instructions. Refer to the "Programming Summary" for the effect that this flag command and the *I/O Reset* (IORST) instruction have on the ATP interface.

Set ATP Status (microECLIPSE CPU only)

DOA 6



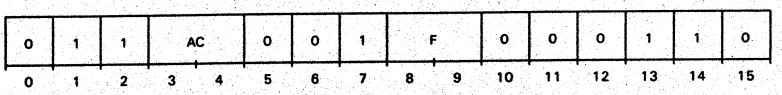
Determines the interrupt status of the ATP (8086) interface. The specified accumulator must be in the following format:



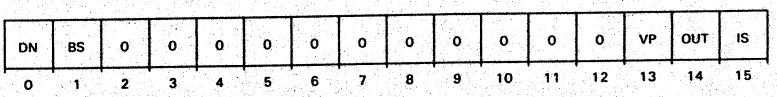
Bits	Name	Contents or Function
0-14	—	Reserved for future use. Must be zero.
15	Interrupt Request	When 0: the ATP interface will issue an interrupt request to the microECLIPSE processor when the ATP Done flag is 1 (due to *Out* instruction or validity fault). When 1: the ATP interface will not issue any interrupt requests.

Read ATP Status (microECLIPSE CPU only)

DIA 6



Places the contents of the ATP (8086) status register in the specified accumulator in the following format:



Bits	Name	Contents or Function
0	Done	Value of the ATP Done flag
1	Busy	Value of the ATP Busy flag
2-12	—	Reserved for future use. Returned as 0.
13	Validity Protect	The 8086 processor returned control to the microECLIPSE processor by accessing a validity protected page. (This is logically equivalent to the 8086 processor running out of memory.)
14	OUT	The 8086 processor returned control to the microECLIPSE processor by executing an *Out* instruction.
15	Interrupt Request	When 0: the ATP interface will issue an interrupt request to the microECLIPSE processor when Done = 1 (due to *Out* instruction or validity fault). When 1: the ATP interface will not issue any interrupt requests.

The status register is cleared by:

An *I/O Reset* (IORST) instruction.

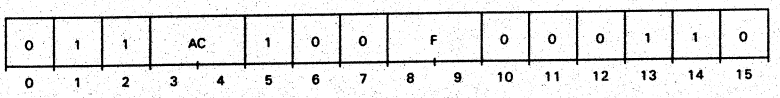
A Clear (C) command.

The OUT and VP bits are cleared by a Start (S) function.

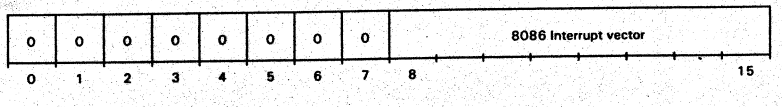
The OUT and VP bits are only valid when the 8086 is done.

Load ATP Interrupt Vector (microECLIPSE CPU only)

DOB 6



Loads an interrupt vector into the ATP interrupt address register from the specified accumulator and pends an 8086 interrupt. When the 8086 processor is started, this interrupt will be taken. The format of the specified accumulator must be as follows:



Bits	Name	Contents or Function
0-7	—	Reserved for future use. Must be zero.
8-15	Interrupt Vector	The interrupt vector for the 8086 processor

NOTE *The pended interrupt can be cleared by resetting the 8086 processor with a Clear command or* I/O Reset *instruction.*

Four additional programmed I/O instructions, two addressed to the ATP interface (device code 6_8) and three addressed to the microECLIPSE CPU (device code 77_8 allow the CPU to:

Alter program flow determined by the state of the Done flag (IOSKP).

Issue a Start or Clear flag command without a programmed input/output transfer (NIO).

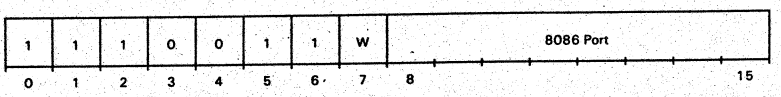
Identify an interrupting source (INTA).

Initialize the interface (IORST).

The latter four instructions are fully described in the *16-Bit Real-Time ECLIPSE Assembly Language Programming.*

Attached Processor System Calls

Fixed Port Out (8086 CPU only)



Where, if W = 0, then

The source is the AL register

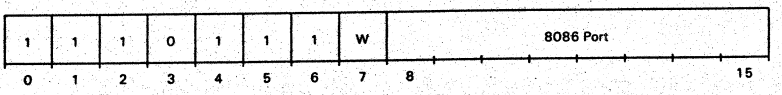
The destination is the port

or if W = 1, then

The source is the AX register

The destination is the port + 1

Variable Port Out (8086 CPU only)



Where, if W = 0, then

The source is the AL register

The destination is the location contained in the DX register

or if W = 1, then

The source is the AX register

The destination is the location contained in the DX register + 1

NOTE *Only bits 0-6 of the above instructions are actually used to determine whether an Out instruction should be executed.*

The execution of any *Out* instruction by the 8086 processor causes the 8086 to be idled at its current address, and the microECLIPSE processor to be started at its last address. The OUT bit in the ATP (8086) status register is set.

Programming the Attached Processor

Once the 8086 processor is started with a Start command, it will run until one of the following conditions is met:

1. A microECLIPSE nonmaskable interrupt is received. The 8086 processor will be held at its current point and the microECLIPSE will handle the NMI. Then, if the system was not stopped or reset by the NMI, the 8086 processor will be restarted at its last location. This action is totally transparent to the user.
2. A microI/O or internal device interrupt is received while interrupts are enabled. The 8086 processor is held at its current position, and the microECLIPSE processor takes control. Since interrupts are enabled, the microECLIPSE interrupt vector sequence will be initiated and the interrupt return address is address immediately following the instruction that issued the most recent Start command to the ATP. The state of the ATP interface at this point will be: Busy = 1, Done = 0. The interrupt handler may then return to that address in microECLIPSE program after the interrupt(s) is handled. The user may then continue the 8086 processor with another Start command.
3. The 8086 processor executes an *Out* instruction. This causes the OUT bit to be set in the ATP status register, causes the ATP Done bit to be set to 1, and holds the 8086 processor idle at the address of the instruction following the OUT. The microECLIPSE processor will be started at the instruction immediately following the last instruction that started the 8086 processor. An interrupt will occur only if a previous DOA ATP instruction had set the interrupt request bit in the ATP status register to 0.
4. The 8086 processor accessed a validity protected page. This causes the VP bit in the ATP status register to be set to 1, causes the Done bit to be set to 1, and idles the 8086. The microECLIPSE processor will be started at the instruction following the last instruction that started the 8086 processor. An interrupt will occur only if a previous DOA ATP instruction had set the interrupt request bit in the ATP status register to 0.

Appendix C presents an example of how the standard I/O interface with the Attached Processor can be used. This example is for information only: it is not intended as a recommended interface.

Programming Bit-Mapped Graphics

Complete instructions for using the downloaded graphics commmand interpreter for programming the bit-mapped graphics interfaces — both the monochrome monitor and the optional color monitor interfaces — are contained in Model 10 and 10/SP System Console Programmer's Manual. A brief summary of the process of programming the monochrome monitor, which involves writing data directly to the screen buffer, is presented here for reference.

Normally, a user program sends characters to be displayed on the monochrome monitor using the I/O instruction DOA TTO or once the graphics command interpreter has been loaded, it uses the graphics commands available.
There is another method for creating graphics displays that is more complicated but executes faster: write data containing the value of each picture element (pixel) directly into the screen buffer. The screen buffer is system memory that the video interface reads to find what it should display. The formats for screen buffer data are presented in Appendix D of this manual.

Both the microECLIPSE and the 8086 processors can write data directly into the screen buffer for the monochrome monitor. The user program must set up the memory map for the processor that is to be used in order to enable this direct

access. *Only the microECLIPSE processor can set up memory maps.* If direct-access bit-mapped graphics is to be performed by the 8086 processor, the 8086 code must execute a call to the microECLIPSE processor to have it set up the 8086 map.

Setting Up the Memory for Bit-Mapped Graphics

Setting up the memory allocation and protection unit (MAP) for bit-mapped graphics involves the following steps:

1. If the 8086 is the controlling processor, set up the mailbox to request the bit-mapped graphics function and execute an *Out* instruction to pass control to the microECLIPSE processor.
2. Set up a load map table in your program space: A microECLIPSE map table will contain 10 words mapping 10 logical pages to physical pages 1760_8 through 1771_8.

   An 8086 map will contain 20 words mapping 10 logical pages to physical pages 1760_8 through 1771_8.
3. Disable I/O protection and LEF mode with a *Load MAP Status* instruction.
4. Execute a *Read Control Memory Status* instruction (RHYP). (AC0 must contain 14_8; status will be returned in AC1.)
5. Set bit 0 in AC1 to 1. Do not change any other bits in AC1.
6. Execute a *Write Control Memory Status* instruction (WHYP). (AC0 must contain 14_8; the contents of AC1 will be written into control memory location 14.)
7. Execute a *Load microECLIPSE* (LMP) or *Load 8086* (LMPA) MAP instruction to load the previously set up map table into the appropriate map.
8. If the 8086 processor requested the setup, return control to it with a *Load ATP Interrupt Vector* (DOB ATP) instruction or an ATP instruction with a START command.

Once your program has executed this instruction sequence, it can write directly into the video screen buffer (in physical pages 1760-1771) by writing to the assigned logical pages.

Halt Instruction

This instruction stops user program execution and returns to the virtual console program. Refer to Chapter 3 "Virtual Console" for more information.

Powerfail/Autorestart

When an abnormal power condition occurs, the power supply asserts a signal that causes the system I/O IC to issue an interrupt request to the CPU. The CPU should respond to the request by entering a user-supplied powerfail interrupt routine. A typical powerfail routine saves the state of the processor, loads a return instruction into physical location 0, issues an **NIOC** 20, and Halts. After the *Halt* instruction, a powerfail-interrupt-clear instruction sequence and an interrupt-return sequence is usually included.

If the power failure condition persists longer than 2 ms, the program and data in the CPU and RAM memory are lost. In this case, when power returns, the virtual console program enters the normal power-up sequence just described.

If power is restored before 2 ms, the following applies.

The *Halt* instruction transfers program control to the virtual console. The virtual console issues a prompt. Resume your program by entering a P command. (Refer to Chapter 3, "Virtual Console.")

NOTE *The CPU Done bit directly reflects the status of the powerfail signal. That is, a CPU Acknowledge* instruction (DOAP) with 177776 in the specified accumulator will clear the interrupt request, but if the powerfail condition persists the CPU Done bit remains set.

Instruction Execution Times

This section discusses instruction execution times for both the microECLIPSE and 8086 processors.

microECLIPSE Instructions

Table 1-33 lists typical execution times for all standard microECLIPSE instructions.

Table 1-29 Instruction execution times

Instruction	Execution Time (microsec.)	CPU Cycles	Notes
ADC	0.50	1	1
ADD	0.50	1	1
ADDI	1.00	2	
ADI	0.50	1	
ANC	0.50	1	
AND	0.50	1	
ANDI	1.00	2	
BAM	6.50 + 2.5/word	13 + 5/word	2,3
BLM	3.50 + 2.0/word	7 + 4/word	2,3
BTO	5.50	11	4
BTZ	4.50	9	4
CLM	3.50	7	1
CMP	8.50 + 7.0/byte	17 + 14/byte	3
CMT	1.50 + 9.0/byte	3 + 18/byte	3
CMV	5.50 + 5.5/byte	11 + 11/byte	3
COB	7.50	15	5
COM	0.50	1	1
CTR	5.50 + 7.0 or 9.5/byte	11 + 14 or 19/byte	6
DAD	8.00	16	
DHXL	7.50	15	5
DHXR	7.00	14	5
DIA,B,C	7.5	15	7
DIS	7.5	15	7
DIV	12.00	24	
DIVS	20.50	41	
DIVX	19.50	39	
DLSH	6.50	13	5
DOA,B,C	6.0	12	7
DSB	8.00	16	
DSPA	6.00	12	2
DSZ	2.00	4	2,1
EDSZ	2.00	4	2,1
EISZ	2.00	4	2,1
EJMP	1.50	3	2
EJSR	1.50	3	2
ELDA	1.00	2	2
ELDB	3.50	7	
ELEF	1.00	2	2

Table 1-29 Instruction execution times (Continued)

Instruction	Execution Time (microsec.)	CPU Cycles	Notes
ESTA	1.00	2	2
ESTB	3.50	7	
FAB	11.00	22	8
FAD	87.00	174	8
FAMD	92.00	184	8,9
FAMS	67.00	134	8,9
FAS	65.00	130	8
FCLE	3.00	6	
FCMP	29.00	58	
FDD	900.00	1800	8,10
FDMD	900.00	1800	8,9,10
FDMS	190.00	380	8,9,10
FDS	190.00	380	8,10
FEXP	13.00	26	8
FFAS	46.00	92	8
FFMD	45.50	90	8,9
FHLV	30.00	60	8,10
FINT	20.50	41	8
FLAS	31.00	62	
FLDD	16.50	32	9
FLDS	15.00	30	9
FLMD	31.00	62	9
FLST	11.50	23	8,9
FMD	266.00	532	8
FMMD	266.00	532	8,9
FMMS	80.50	161	8,9
FMOV	17.00	34	
FMS	80.50	161	8
FNEG	12.50	25	8
FNOM	43.00	86	8
FNS	1.00	2	
FPOP	32.00	64	
FPSH	31.50	63	
FRH	6.00	12	
FSA	1.50	3	
FSCAL	48.00	96	8
FSD	87.00	174	8
FSEQ	5.00	10	
FSGE	4.50	9	
FSGT	5.00	10	
FSLE	5.00	10	
FSLT	4.50	9	
FSMD	92.00	184	8,9
FSMS	67.00	134	8,9
FSND	5.00	10	
FSNE	5.00	10	
FSNER	5.00	10	
FSNM	5.00	10	
FSNO	5.00	10	
FSNOD	5.00	10	
FSNU	5.00	10	
FSNUD	5.00	10	
FSNUO	5.00	10	
FSS	65.00	130	8
FSST	7.50	15	9
FSTD	12.50	25	9
FSTS	9.50	19	9

Table 1-29 Instruction execution times (Continued)

Instruction	Execution Time (microsec.)	CPU Cycles	Notes
FTD	3.00	6	
FTE	5.00	10	8
HALT	6.50	13	
HLV	1.50	3	
HXL	2.00	4	5
HXR	2.00	4	5
INC	0.50	1	1
INTA	7.5	15	7
IOR	1.50	3	
IORI	1.50	3	
IORST	6.0	12	7
ISZ	2.00	4	2,1
JMP	1.50	3	2
JSR	1.50	3	2
LDA	1.00	2	2
LDB	1.50	3	
LEF	1.00	2	2
LOB	4.00	8	5
LRB	5.00	10	5
LSH	8.50	17	5
MOV	0.50	1	1
MSKO	6.0	12	7
MSP	3.00	6	11
MUL	9.50	19	
MULS	9.50	19	
NEG	0.50	1	1
NIO	6.0	12	7
POP	1.50	3	12
POPB	6.50	13	
POPJ	3.00	6	
PSH	3.00	6	11,12
PSHJ	4.50	9	11
PSHR	4.50	9	
RSTR	10.50	21	
RTN	6.00	12	
SAVE	8.00	16	11
SBI	0.50	1	
SGE	0.50	1	1
SGT	0.50	1	1
SKP	1.50	3	1
SNB	4.50	9	1,4
STA	1.00	2	2
STB	1.50	3	
SUB	0.50	1	1
SYC	10.50	21	2
SZB	4.00	8	1,4
SZBO	6.00	12	1,4
XCH	1.50	3	
XCT	2.00	4	13
XOP	20.50	41	
XOP1	21.50	43	
XOR	2.50	5	
XORI	2.50	5	

1. If skip occurs, add 0.50.
2. If indirect chain followed, add 0.50 + (number of indirects - 1)*1.00.
3. For each item moved, add the amount shown.

Table 1-29 Instruction execution times (Continued)

Instruction	Execution Time (microsec.)	CPU Cycles	Notes

4. If ACS < > ACD, add 1.00 + 2 (number of indirects).
5. Execution time is operand-dependent. Time listed is typical.
6. Byte moves require 7.0 microseconds/byte; compares require 9.5 microseconds/byte.
7. Time given for external I/O devices. For internal devices, refer to "Execution times for emulated instructions"
8. This instruction can take a floating point trap, which would add 19 microseconds to the execution time.
9. This instruction does an effective address calculation, which can add 15 microseconds to the execution time.
10. Floating-point divide execution times depend on the number of zero and one bits in the quotient (the more ones contained in the quotient, the longer the execution time)
11. For stack overflow, add 9.00 + (Note 2).
12. For each accumulator pushed/popped, add 0.50.
13. Add instruction execution time.

Table 1-30 lists typical execution times for control-memory-emulated instructions.

Table 1-30 Execution times for emulated instructions

Instruction	Execution Time (microsec.)	CPU Cycles	Notes
DIA ATP	133.0	266	
DIA MAP	2.0	4	1
DIA TTI	128.5	257	
DIC ATP	141.5	283	
DIC MAP	15.0	30	
DOA ATP	132.0	264	
DOAS ATP	187.0	374	
DOA MAP	15.0	30	
DOAS TTO	839.0	1678	2
DOB MAP	15.0	30	
DOC MAP	15.0	30	
MSKO	144.5	289	
INTA	161.5	323	3
IORST	639.5	1279	
LMP	15.0 + 7.5/word	30 + 15/word	
LMPA	44.0 + 78.0/word	88 + 156/word	
NIOS ATP	181.5	363	3
SKP TTI	133.5	267	
SKP TTO	136.5	273	
VCT	166.5 to 254.5	333 to 509	4

[1] *DIA MAP is not an emulated instruction, but is included here for completeness.*
[2] *The time shown is for the transfer of one displayable character. The times for execution of codes for such characters as new lines or formfeeds are longer.*
[3] *The time shown is the maximum that this instruction can take.*
[4] *Vector execution times depend on the mode employed.*

8086 Instructions

Instruction execution times for the 8086 depend upon the type of instruction *and* the type of addressing mode involved. The simplest instructions, those involving simple shifts or bit changes within a flag or accumulator, take only two clock cycles, or 400 ns. Examples are CLI, CMC, CBW, CLC, CLD.

Slightly more complicated operations, such as simple additions or (register-to-register) comparisons, take three cycles, or 600 nanoseconds. Examples are ADC, ADD, CMP.

In the medium execution time range are program transfer instructions like JMP and CALL. With the simplest addressing mode, these take about 4 ms.

At the extreme end of the range are the extended types of arithmetic operations, such as MUL, IMUL, DIV, IDIV. These can take as long as 38 microseconds, plus the time it may take for effective address calculation.

Time required for effective address calculation is as follows:

Displacement only: 1.2 microseconds

Base or index only: 1.0 microseconds

Displacement plus base or index: 1.8 microseconds

Base plus index: 1.4 - 1.6 microseconds

Displacement plus base plus index: 2.2 - 2.4 microseconds

Programming CPU-Resident I/O Devices

2

This section describes the programmable elements of the basic Model 10 and 10/SP resident I/O interfaces, summarizes their specifications, instruction accumulator formats, and the flag commands used to program them; defines the relevant I/O instructions; discusses programming considerations and I/O timing; and explains possible error conditions. The following interfaces are covered:

- The system console interface, including the keyboard (TTI, device code 10) and the monochrome monitor (TTO, device code 11)
- The printer port (TTI1/TTO1, device code 50/51)
- The real-time clock (RTC, device code 14)
- The programmable interval timer (PIT, device code 43)
- The diskette subsystem (DE0, device code 20)

System Console Interface

The system console I/O interface allows a user program to transmit ASCII characters to the system display monitor and receive ASCII characters from the system keyboard using standard I/O instructions. As standard I/O devices, the keyboard is assigned device code 10_8 and the monitor is assigned device code 11_8. The display monitor and the keyboard are both connected to the A connector on the SPU2 PC board.

NOTE *A user program can, instead of using the system monitor's I/O interface, write directly to the monitor's bit mapped memory in order to display text or graphics. For information on how to do this, refer to the section in Chapter 1 titled "Programming Bit-Mapped Graphics."*

Programmable Elements

From a programming point of view, the system console interface consists of the following major elements:

Keyboard Register

Monitor Register

Keyboard Busy Flag

Monitor Busy Flag

Keyboard Done Flag

Monitor Done Flag

Keyboard Interrupt Disable flag

Monitor Interrupt Disable flag

Your program directs activities at the interface level by manipulating these major elements, using the following instructions together with the Start, Clear, and I/O Pulse device flag commands:

Write Character (DOA)

Read Character (DIA)

I/O Skip (IOSKP)

The program also uses two CPU I/O instructions to manipulate the state of some elements: *I/O Reset* (IORST) and *Mask Out* (MSKO).

Programming Summary

The following programming summary provides general interface specifications, shows the accumulator formats for the two system console instructions, and explains its response to the Start, Clear, and Pulse flag commands and the *I/O Reset* instruction (IORST).

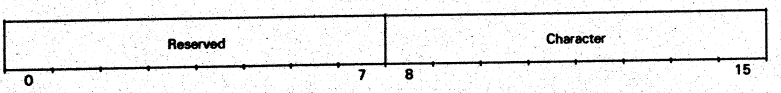
Table 2-1 Programming summary: general interface specifications

Mnemonics	
Receiver	TTI
Transmitter	TTO
Device codes	
Receiver	10 octal
Transmitter	11 octal
Priority mask bits	
Receiver	14
Transmitter	15
Character structure	8 data bits, no parity

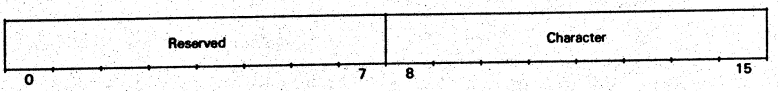
	Programming Summary
Mnemonic	
Receiver	TTI
Transmitter	TTO
Device codes	
Receiver	10_8
Transmitter	11_8
Priority mask bit	
Receiver	14
Transmitter	15
Character structure	8 data bits, no parity

Accumulator Formats

Write Character (DOA)



Read Character (DIA)



Start, Clear, Pulse, and IORST Functions

f=S *Keyboard*
Sets the keyboard Busy flag to 1, the Done flag to 0, and clears the current contents of the keyboard register.
Monitor
Sets the monitor Busy flag to 1, the Done flag to 0, and displays the current contents of the monitor register.

f=C *Keyboard*
Sets the keyboard Busy flag to 1 and the Done flag to 0. Clears the current contents of the keyboard register.
Monitor
Sets the monitor Busy and Done flags to 0. Clears the current contents of the monitor register.

f=P *Keyboard and Monitor*
No effect

IORST Sets the keyboard and monitor Busy, Done, and Interrupt disable flags to 0. Clears the contents of the keyboard and monitor registers.

DG-25973

Figure 2-1 Programming Summary: system console interfaces

Table 2-2 Programming summary: Start, Clear, Pulse, and IORST Functions

f = S	*Keyboard* Sets the keyboard Busy flag to 1, the Done flag to 0, and clears the current contents of the keyboard register. *Monitor* Sets the monitor Busy flag to 1, the Done flag to 0, and displays the current contents of the monitor register.
f = C	*Keyboard* Sets the keyboard Busy flag to 1 and the Done flag to 0. Clears the current contents of the keyboard register. *Monitor* Sets the monitor Busy and Done flags to 0. Clears the current contents of the monitor register.
f = P	*Keyboard and Monitor* No effect
IORST	Sets the keyboard and monitor Busy, Done, and Interrupt disable flags to 0. Clears the contents of the keyboard and monitor registers.

Registers and Flags

The program-accessible registers of the system console include the keyboard and monitor registers. Its program-accessible flags include the Busy, Done, Interrupt Disable, and Break flags.

Keyboard Register

The keyboard register stores an 8-bit ASCII character that corresponds to the key code sent to the SPU by the keyboard. It makes the character available to the program until the receiver overwrites the contents of the register with the next decoded character. The program reads the contents of this register using a *Read Character* instruction (DIA).

Monitor Register

The transmit register stores the 8-bit character sent to the interface by the program for transmission to the monitor. When the program issues a Start command, the monitor hardware encodes the character contained in monitor register and sends it to the video display device. The program loads the monitor register using a *Write Character* instruction (DOA).

Busy Flags

The Busy flag of the keyboard section, when set to 1, causes the keyboard register to be cleared (contents set to 0). The Busy flag will remain set until the program clears it. The program manipulates the monitor Busy flag using the *I/O Reset* instruction (IORST) or the Start and Clear flag commands. The Clear command or the *I/O Reset* instruction sets the Busy flag to 0; the Start flag command sets the Busy flag to 1. A program can test the keyboard Busy flag using the *I/O Skip* instruction (IOSKP) addressed to the keyboard section of the console interface (TTI).

The Busy flag of the monitor section, when set to 1, causes the contents of the monitor register to be performed or displayed on the video display device immediately. As this happens, the Busy flag is automatically cleared and the Done flag is set to 1. The program manipulates the transmitter Busy flag using the *I/O Reset* instruction (IORST) or the Start and Clear flag commands. The *I/O Reset* instruction and Clear flag command sets the Busy flag to 0; the Start flag command sets the Busy flag to 1. A program can test the monitor Busy flag using the *I/O Skip* instruction ($\overline{\text{IOSKP}}$) addressed to the monitor section of the console interface (TTO).

Done Flags The keyboard Done flag, when set to 1, initiates an interrupt request to the CPU (unless interrupts are disabled) and specifies that the interface has a character from the system console available for transfer to the CPU. The program sets the Done flag to 0 using the *I/O Reset* instruction (IORST) or the Start or Clear flag commands. A program can test the Done flag using the *I/O Skip* instruction (IOSKP) addressed to the receiver section of the keyboard section of the console interface (TTI).

The monitor Done flag, when set to 1, initiates an interrupt request to the CPU (unless interrupts are disabled) and specifies that the monitor has performed or displayed the character in the monitor register. The program sets the Done flag to 0 using the *I/O Reset* instruction (IORST) or the Start or Clear flag commands. A program can test the Done flag using the *I/O Skip* instruction (IOSKP) addressed to the monitor section of the console interface (TTO).

Interrupt Disable Flags The keyboard and monitor Interrupt Disable flags enable and disable their respective console interface interrupts. When the flag is set to 0, interrupts are enabled; when it is set to 1, interrupts are disabled. The program manipulates the state of the Interrupt Disable flags with either an *I/O Reset* instruction (IORST) or the CPU's *Mask Out* instruction (MSKO). The *I/O Reset* instruction (IORST) sets the Interrupt Disable flag to 0, enabling interrupts. The *Mask Out* instruction (MSKO) sets the Interrupt Disable flag to 0 or 1, depending upon the logical state of bit positions 14 (keyboard) and 15 (monitor) of the data word being transferred. For more information on the *Mask Out* instruction, refer to its description in *16-bit Real-Time ECLIPSE Assembly Language Programming*.

I/O Instruction Set

Two programmed I/O instructions enable you to program the console interface to perform character transfers to and from the system console. These instructions:

- Load a character from a specified CPU accumulator into the monitor register (DOA).
- Read a character from the keyboard register into a specified CPU accumulator (DIA).

Five additional programmed I/O instructions, two addressed to either the keyboard or monitor section of the printer port and three addressed to the CPU (Device code 77_8) allow the CPU to:

- Alter program flow determined by the state of either section of the console interface — busy, done, or idle (IOSKP).
- Issue a device flag command without a programmed input/output transfer (NIO).
- Enable/disable the console interface interrupt facility (MSKO).
- Identify an interrupting source (INTA).
- Initialize the console interface (IORST).

The latter instructions are fully described in detail in *16-bit Real-Time ECLIPSE Assembly Language Programming*.

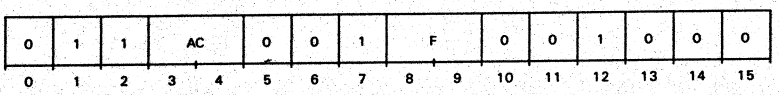
Assembly language coding and instruction bit patterns are shown for each of the two instructions described. The device code field of the instruction is shown

with the standard Model 10 and 10/SP console interface codes: 10_8 for the keyboard section; 11_8 for the monitor section. Mnemonics TTI or 10_8 used in the assembly language coding address the keyboard section of the console interface, while mnemonics TTO or 11_8 address the receiver section.

Two device flag commands, Start (S) and Clear (C), can be specified in the *f* field of programmed I/O instructions. Refer to the "Programming Summary" for the effect these flag commands and the *I/O Reset* (IORST) instruction have on each section of the console interface.

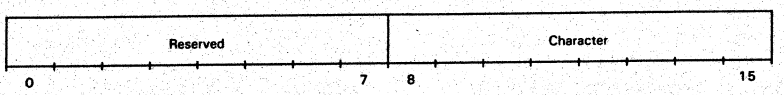
Read Character

DIA *[f]* *ac*,TTI



Reads the character most recently received from the keyboard.

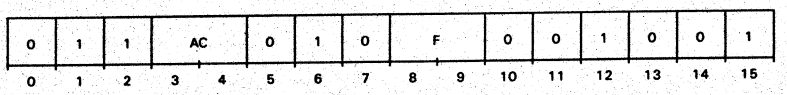
Loads the contents of the keyboard register into bits 8-15 of the specified accumulator. Bits 0-7 of the specified accumulator are set to zero. After the data transfer, the command sets the Busy and Done flags of the console interface keyboard section according to the function specified by *f*. The format of the specified CPU accumulator after the transfer is:



Bit(s)	Name	Function
0-7	—	Reserved (all bits set to 0).
8-15	Character	The character most recently received, right justified in the accumulator.

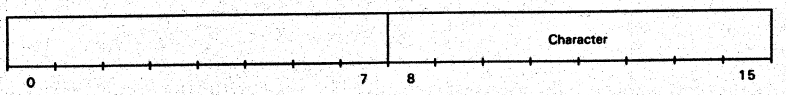
Write Characters

DOA *[f]* *ac*,TTO



Loads the monitor register with the character to be performed or displayed on the video display device.

Bits 8-15 of the specified accumulator are loaded into the monitor register. After the data transfer, the command sets the Busy and Done flags of the console interface section according to the function specified by *f*. The format of the specified CPU accumulator before and after the transfer is:



Bit(s)	Name	Function
0-7	—	Ignored (can be set to 1 or 0).
8-15	Character	The character to be transmitted. *Note:* only the 7 least significant bits of this character are used: the upper bit is ignored.

Programming Guidelines

Programming the console interface involves the following steps:

Writing characters

Reading characters

Servicing interrupts

Writing Characters

Use the following programming sequence to perform or display a character on the video display device. Refer to Figure 2-2.

1. Issue a *Write Character* instruction with a Start flag command (DOAS) using the appropriate accumulator bits to specify the character to be sent over the communications line.

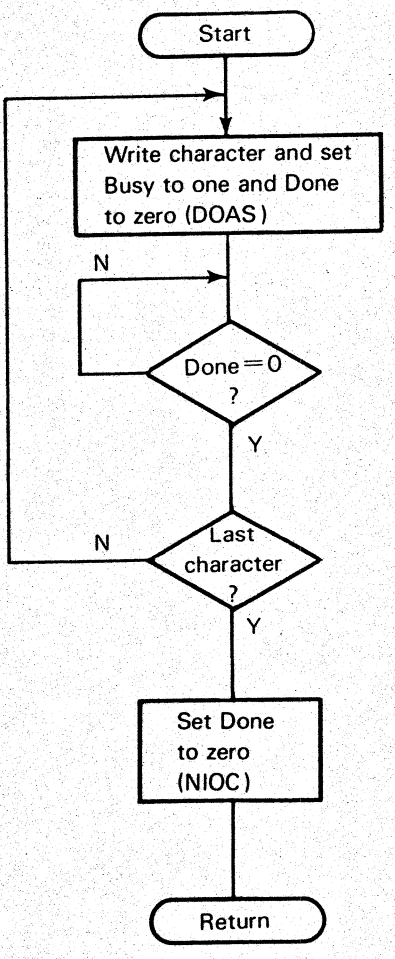


Figure 2-2 Writing characters

Reading Characters

Use the following programming sequence to initiate character reception from the keyboard. Refer to Figure 2-3.

When the receiver has a character for the program (Done flag set to 1), issue a *Read Character* instruction with either a Start (DIAS) or Clear flag command (DIAC) to load the character received from the keyboard via the receive register into a CPU accumulator. The Start flag command initiates another character reception by setting the Busy flag to 1 and the Done flag to 0. The Clear flag command terminates character reception by setting both the Busy and Done flags to 0.

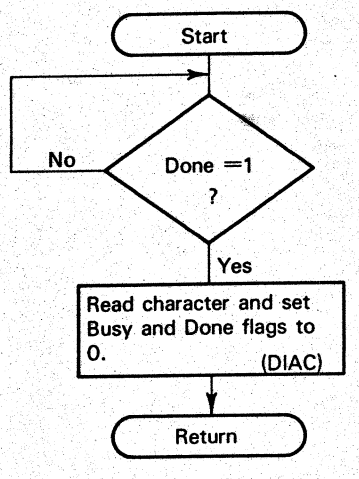


Figure 2-3 Reading characters

I/O Timing

After the keyboard section's Done flag is set to 1, the character in the keyboard register is available to the program until the user enters the next character on the keyboard.

Power-Up Response

After power-up, the keyboard and monitor Busy, Done, and Interrupt Disable flags are set to zero and the monitor and keyboard registers are cleared.

Printer Port

The printer port is an asynchronous communications interface (assigned to microI/O device codes 50/51) that is capable of allowing full-duplex communications between the CPU and a printer or terminal connected to A-connector of the Model 10 and 10/SP SPU-1 card. The interface contains a double-buffered transmitter (TTO1) section and a receiver (TTI1) section that function as separate devices.

NOTE *The Model 10 and 10/SP printer port receives and transmits eight bit data characters without parity. If the device being used with the printer port operates with a data character length of seven bits, configure the device to operate with "mark parity." When receiving data characters from a 7-bit device, software should mask out the parity bit after the character has been loaded into an accumulator. The parity bit is the most significant bit in the character and is contained in bit 8 of the specified accumulator.*

Programmable Elements

From a programming point of view, the printer port communications interface consists of the following major elements:

Receive Register

Transmit Register

Receiver Busy Flag

Transmitter Busy Flag

Receiver Done Flag

Transmitter Done Flag

Receiver Interrupt Disable flag

Transmitter Interrupt Disable flag

Your program directs activities at the interface level by manipulating these major elements, using the following instructions together with the Start and Clear device flag commands:

Write Character (DOA)

Read Character (DIA)

I/O Skip (IOSKP)

The program also uses two CPU I/O instructions to manipulate the state of some elements: *I/O Reset* (IORST) and *Mask Out* (MSKO).

Programming Summary

The following programming summary provides general interface specifications, shows the accumulator formats for the two printer port instructions, and explains its response to the Start, Clear, and Pulse flag commands and the *I/O Reset* instruction (IORST).

Table 2-3 Programming summary: general interface specifications

Mnemonics	
Receiver	TTI1
Transmitter	TTO1
Device codes	
Receiver	50 octal
Transmitter	51 octal
Priority mask bits	
Receiver	14
Transmitter	15
Line speed	9600 baud
Character structure	8 data bits no parity 1 stop bit
Programmed I/O latency	1.04 ms
Modem control signal used	Clear To Send

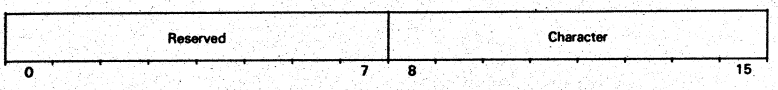
***Table 2-4 Programming summary: Start, Clear, Pulse and IORST** functions*

f = S	*Receiver section* Sets the receiver Busy flag to 1, the Done flag to 0, and initiates a character reception from the communications line. *Transmitter section* Sets the transmitter Busy flag to 1, the Done flag to 0, and initiates a character transmission to the communications line.
f = C	*Receiver section* Sets the receiver Busy flag to 1 and the receiver Done flag to 0. *Transmitter section* Sets the transmitter Busy and Done flags to 0.
f = P	*Receiver section* Sets the Break flag to 0. *Transmitter section* No effect
IORST	Sets the receiver and transmitter Busy, Done, and Interrupt disable flags to 0. Also sets the receiver Break flag to 0.

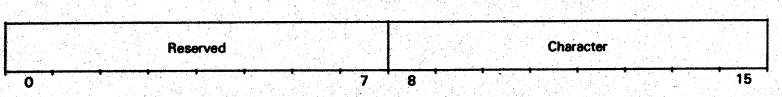
	Programming Summary
Mnemonic	
Receiver	TTI1
Transmitter	TT1O
Device codes	
Receiver	50_8
Transmitter	51_8
Priority mask bit	
Receiver	14
Transmitter	15
Line speed	9600 baud
Character structure	8 data bits no parity 1 stop bit
Programmed I/O latency	1.04 ms
Modem control signal used	Clear to Send

Accumulator Formats

Write Character (DOA)



Read Character (DIA)



Start, Clear, Pulse, and IORST Functions

f=S *Receiver section*
Sets the receiver Busy flag to 1, the Done flag to 0, and initiates a character reception from the communications line.
Transmitter section
Sets the transmitter Busy flag to 1, the Done flag to 0, and initiates a character transmission to the communications line.

f=C *Receiver section*
Sets the receiver Busy flag to 1 and the receiver Done flag to 0.
Transmitter section
Sets the transmitter Busy and Done flags to 0.

f=P *Reciever section*
Sets the Break flag to 0.
Transmitter section
No effect.

IORST Sets the receiver and transmitter Busy, Done, and Interrupt disable flags to 0. Also sets the receiver Break flag to 0.

DG-25975

Figure 2-4 Programming Summary: printer port

Registers and Flags

The program-accessible registers of the printer port include the receive and transmit registers. Its program-accessible flags are the Busy, Done, Interrupt Disable, and Break flags.

Receive Register The receive register stores the 8-bit assembled character received over the communications line in serial form. It makes the character available to the program until the receiver overwrites the contents of the register with the next assembled character. The program reads the contents of this register using a *Read Character* instruction (DIA).

Transmit Register The transmit register stores the 8-bit character sent to the interface by the program for transmission to the communication line. When Clear To Send is asserted by the terminal, the transmitter section disassembles the character contained in this register and sends it in serial form over the communications line. The program loads this register using a *Write Character* instruction (DOA).

Busy Flags The Busy flag of the receiver section, when set to 1, initiates a character reception operation over the asynchronous line to the printer port. This flag remains set until a character has been input from the terminal and fully assembled in the receive register. The program manipulates the receiver Busy flag using the *I/O Reset* instruction (IORST) or the Start and Clear flag commands. The *I/O Reset* instruction sets the Busy flag to 0; the Start and Clear flag command sets the Busy flag to 1. A program can test the receiver Busy flag using the *I/O Skip* instruction (IOSKP) addressed to the receiver section of the printer port (TTI1).

The Busy flag of the transmitter section, when set to 1, initiates a character transmission operation over the asynchronous communications line of the printer port. This flag remains set until a character has been fully disassembled and transferred to the communications line. The program manipulates the transmitter Busy flag using the *I/O Reset* instruction (IORST) or the Start and Clear flag commands. The *I/O Reset* instruction and Clear flag command sets the Busy flag to 0; the Start flag command sets the Busy flag to 1. A program can test the transmitter Busy flag using the *I/O Skip* instruction (IOSKP) addressed to the transmitter section of the printer port (TTO1).

Done Flags The receiver Done flag, when set to 1, initiates an interrupt request to the CPU (unless interrupts are disabled) and specifies that the interface has a character from the communications line available for transfer to the CPU. The program sets the Done flag to 0 using the *I/O Reset* instruction (IORST) or the Start, Clear, or Pulse flag commands. A program can test the Done flag using the *I/O Skip* instruction (IOSKP) addressed to the receiver section of the printer port.

The transmitter Done flag, when set to 1, initiates an interrupt request to the CPU (unless interrupts are disabled) and specifies that the port has completed transfer of the character in the transmit register to the communications line. The program sets the Done flag to 0 using the *I/O Reset* instruction (IORST) or the Start, Clear, or Pulse flag commands. A program can test the Done flag using the *I/O Skip* instruction (IOSKP) addressed to the receiver section of the printer port.

Interrupt Disable Flags The receiver and transmitter Interrupt Disable flags enable and disable their respective printer port interrupts. When the flag is set to 0, port interrupts are enabled; when it is set to 1, port interrupts are

disabled. The program manipulates the state of the Interrupt Disable flags with either an *I/O Reset* instruction (IORST) or the CPU's *Mask Out* instruction (MSKO). The *I/O Reset* instruction (IORST) sets the Interrupt Disable flag to 0, enabling interrupts. The *Mask Out* instruction (MSKO) sets the Interrupt Disable flag to 0 or 1, depending upon the logical state of bit positions 14 (receiver) and 15 (transmitter) of the data word being transferred. For more information on the *Mask Out* instruction, refer to its description in *16-bit Real-Time ECLIPSE Assembly Language Programming*.

I/O Instruction Set

Two programmed I/O instructions enable you to program the printer port to perform character transfers to and from the device connected to it. These instructions:

Load a character from a specified CPU accumulator into the transmit register (DOA).

Read a character from the receiver register into a specified CPU accumulator (DIA).

Five additional programmed I/O instructions, two addressed to either the receiver or transmitter section of the printer port and three addressed to the CPU (Device code 77(8)) allow the CPU to:

Alter program flow determined by the state of the either section of the port — busy, done, or idle (IOSKP).

Issue a device flag command without a programmed input/output transfer (NIO).

Enable/disable the printer port interrupt facility (MSKO).

Identify an interrupting source (INTA).

Initialize the printer port (IORST).

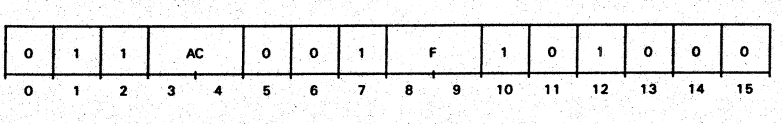
The latter instructions are fully described in detail in *16-bit Real-Time ECLIPSE Assembly Language Programming*.

Assembly language coding and instruction bit patterns are shown for each of the two instructions described. The device code field of the instruction is shown with the standard Model 10 or 10/SP printer port device codes; 50_8 for the receiver section; 51_8 for the transmitter section. Mnemonics TTI1 or 50_8 used in the assembly language coding, address the receiver section of the asynchronous communications interface, while mnemonics TTO1 or 51_8 address the receiver section.

Three device flag commands, Start (S), Clear (C), and Pulse (P), can be specified in the *f* field of programmed I/O instructions. Refer to the Programming Summary for the effect that these flag commands and the *I/O Reset* (IORST) instruction have on each section of the printer port.

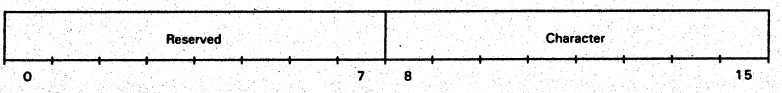
Read Character

DIA*[f]* *ac*,TTI1



Reads the character most recently received by the printer port.

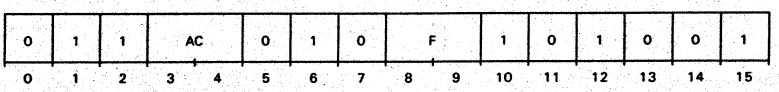
Loads the contents of the receive register into bits 8-15 of the specified accumulator. Bits 0-7 of the specified accumulator is set to zero. After the data transfer, the command sets the Busy and Done flags of the printer port receiver section according to the function specified by *f*. The format of the specified CPU accumulator after the transfer is:



Bit(s)	Name	Function
0-7	—	Reserved (all bits set to 0).
8-15	Character	The character most recently received, right justified in the accumulator.

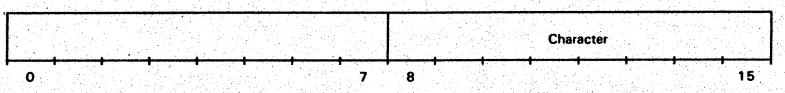
Write Character

DOA*[f]* *ac*,TTO1



Loads the transmit register with the character to be sent to the communications line.

Bits 8-15 of the specified accumulator are loaded into the transmit register. After the data transfer, the command sets the Busy and Done flags, of the printer port interface section, according to the function specified by *f*. The format of the specified CPU accumulator before and after the transfer is:



Bit(s)	Name	Function
0-7	—	Not used (can be set to 1 or 0).
8-15	Character	The character to be transmitted.

Programming Guidelines

Programming the printer port involves the following steps:

Writing characters

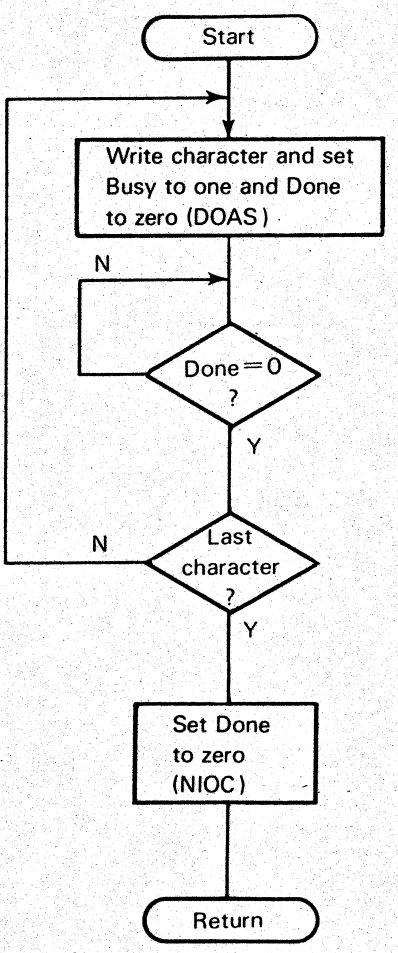
Reading characters

Servicing interrupts

Writing Characters

Use the following programming sequence to send each character over the communications line. Refer to Figure 2-5.

1. Issue an *I/O Skip* instruction (IOSKP) to ensure that the transmitter section's Busy flag is set to 0). If it is a 1, wait until it is 0.
2. Issue a *Write Character* instruction with a Start flag command (DOAS) using the appropriate accumulator bits to specify the character to be sent over the communications line.



DG-09006

Figure 2-5 Writing characters

Reading Characters

Use the following programming sequence to initiate character reception from the communications line of the printer port, and to either continue or terminate character reception. Refer to Figure 2-6.

1. Issue a *No I/O Transfer* instruction with a Start flag command (NIOS) to initiate character reception (sets the receiver section's Busy flag to 1 and Done flag to 0).
2. When the receiver has a character for the program (Done flag set to 1), issue a *Read Character* instruction with either a Start (DIAS) or Clear flag command (DIAC) to load the character received from the communications line via the receive register into a CPU accumulator. The Start flag command initiates another character reception by setting the Busy flag to 1 and the Done flag to 0. The Clear flag command terminates character reception by setting both the Busy and Done flags to 0.

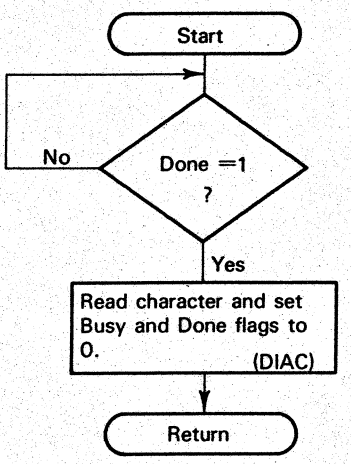


Figure 2-6 Reading characters

I/O Timing

After the receiver section's Done flag sets to 1, the character in the receive register is available to the program for a time interval determined by the transmission rate (baud). To avoid possible data loss when receiving characters from the communications line, the program must respond to the receiver sections Done flag setting to 1 within 1.04 ms.

After the transmitter Done flag sets to one, the program should provide another character to the interface within 1.04 ms.

Characters are transmitted differently depending on whether they are 7-bit characters or 8-bit characters. An 8-bit character is transmitted as a start bit, followed by 8 data bits (least-significant bit first), followed by a stop bit. See Figure 2-7.

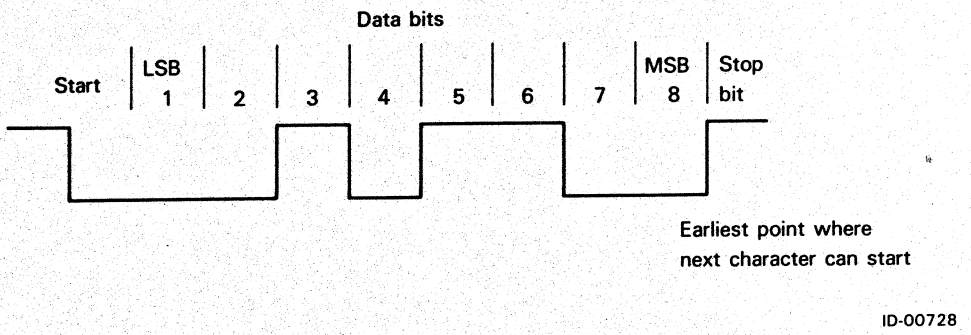


Figure 2-7 Asynchronous transmission of 8-bit character

A 7-bit character is transmitted as a start bit, then 7 data bits (LSB first), a parity bit, and a stop bit. See Figure 2-8.

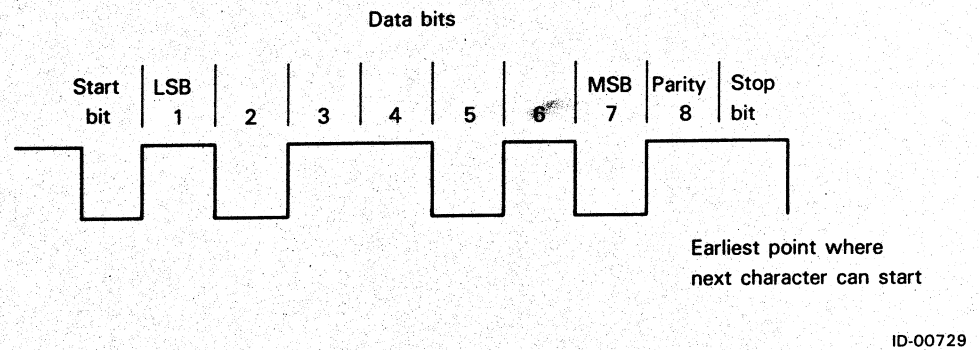


Figure 2-8 Asynchronous transmission of 7-bit character

NOTE *The Model 10 and 10/SP printer port receives and transmits eight-bit data characters without parity. If the terminal device connected to this interface operates with a data character length of seven bits, it should be configured to operate with* mark parity. *If the terminal device connected to the printer port operates with a data character length of eight bits, it should be configured to operate with* no parity.

When receiving data characters from a seven-bit terminal device connected to this port, software should maskout the parity bit position of the character after the character has been loaded into an accumulator. The parity bit position of the character is the most significant bit of the character and will be contained in bit position 8 of the specified accumulator.

Power-Up Response

After power-up, the transmitter and receiver Busy, Done, and Interrupt Disable flags are set to zero.

Real-Time Clock Interface

The real-time clock interface is a programmed I/O interface which provides a programmable selection of precise time bases for the Model 10 and 10/SP computer system.

Programmable Elements

From a programming point of view, the real-time clock interface consists of the following major elements:

Frequency select register

Busy Flag

Done Flag

Interrupt Disable flag

Your program directs interface activities at the interface level by manipulating these major elements, using the following interface instructions together with the Start and Clear device flag commands:

Select Frequency (DOA)

I/O Skip (IOSKP)

The program also uses two CPU I/O instructions to manipulate the state of some elements: *I/O Reset* (IORST) and *Mask Out* (MSKO).

Programming Summary

The following programming summary provides general interface specifications; shows the accumulator formats for the interface instruction; and explains the interface's response to the Start and Clear flag commands and the *I/O Reset* instruction (IORST).

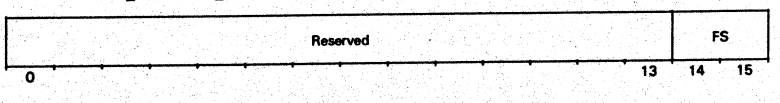
Table 2-5 Programming summary: general interface specifications

Mnemonics	RTC
Device code	14 octal
Priority mask bit	13
Frequencies	10 Hz, 100 Hz, 1000 Hz, and ac line

	Programming Summary
Mnemonic	RTC
Device codes	14
Priority mask bit	13
Frequencies	10 Hz, 100 Hz, 10000 Hz, and ac line

Accumulator Formats

Select Frequency (DOA)



Start, Clear, IOPLS, and IORST Functions

f=S	Sets the Busy flag to 1, the Done flag to 0, enabling real-time-clock interrupts
f=C	Sets the Busy and Done flags to 0, disabling real-time-clock interrupts
f=P	No effect
IORST	Sets the Busy, Done, and Interrupt Disable flags to 0. Also sets the clock frequency bits to 0, selecting ac line clock frequency.

DG-25976

Figure 2-9 Programming Summary: Real-time clock interface

Table 2-6 Programming summary: START, CLEAR, IOPLS, and IORST functions

f = S	Sets the Busy flag to 1, the Done flag to 0, enabling real-time-clock interrupts
f = C	Sets the Busy and Done flags to 0, disabling real-time-clock interrupts
f = P	No effect
IORST	Sets the Busy, Done, and Interrupt Disable flags to 0. Also sets the clock frequency bits to 0, selecting ac line clock frequency.

Register and Flags

The program-accessible register of the real-time clock interface is the 2-bit frequency select register. Its program-accessible flags include the Busy, Done, and Interrupt Disable flags.

Frequency Select Register The frequency select register stores the 2-bit code sent to the interface by the program to select the time base frequency for the real time clock. The program loads this register using a *Select Frequency* instruction (DOA).

Busy Flag The Busy flag, when set to 1, enables the real-time clock to interrupt the CPU (if interrupts are enabled) when each clock period expires. The program manipulates the Busy flag using the *I/O Reset* instruction (IORST) or the Start and Clear flag commands. The *I/O Reset* instruction and Clear flag command

sets the Busy flag to 0, disabling real-time clock interrupts; the Start flag command sets the Busy flag to 1, enabling real-time clock interrupts. A program can test the Busy flag using the *I/O Skip* instruction (IOSKP) addressed to the real-time clock interface (RTC).

Done Flag The Done flag, when set to 1, initiates an interrupt request to the CPU (unless interrupts are disabled) and specifies that a clock period has expired. The program sets the Done flag to 0 using the *I/O Reset* instruction (IORST) or the Start or Clear flag commands. A program can test the Done flag using the *I/O Skip* instruction (IOSKP) addressed to the real-time clock interface.

Interrupt Disable Flag The Interrupt Disable flag also enables and disables real-time clock interface interrupts. When the flag is set to 0, interface interrupts are enabled; when it is set to 1, interface interrupts are disabled. The program manipulates the state of the Interrupt Disable flag with either an *I/O Reset* instruction (IORST) or the CPU's *Mask Out* instruction (MSKO). The *I/O Reset* instruction (IORST) sets the Interrupt Disable flag to 0, enabling interrupts. The *Mask Out* instruction (MSKO) sets the Interrupt Disable flag to 0 or 1, depending upon the logical state of bit position 13 of the data word being transferred. For more information on the *Mask Out* instruction, refer to its description in *16-bit Real-Time ECLIPSE Assembly Language Programming.*

I/O Instruction Set

A single programmed I/O instruction enables you to program the real-time clock interface to select the desired clock frequency. This instruction loads a 2-bit field from a specified CPU accumulator into the frequency select register.

Five additional programmed I/O instructions, two addressed to the real-time clock interface and three addressed to the CPU (Device code 77(8)) allow the CPU to:

Interrogate the state of the interface Busy and Done flags (IOSKP).

Issue a device flag command without a programmed input/output transfer (NIO).

Enable/disable the interface interrupt facility (MSKO).

Identify an interrupting source (INTA).

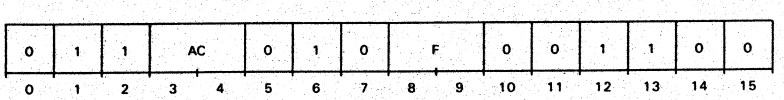
Initialize the interface (IORST).

The latter instructions are fully described in detail in *16-bit Real-Time ECLIPSE Assembly Language Programming.*

Assembly language coding and instruction bit pattern is shown for the instruction described. The device code field of the instruction is shown with the standard real-time clock interface device code, 14_8. Mnemonics RTC or 14_8 used in the assembly language coding, address the real-time clock interface.

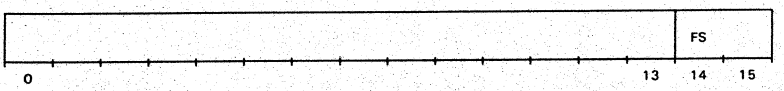
Two device flag commands, Start S and Clear C be specified in the *f* field of programmed I/O instructions. Refer to the "Programming Summary" for the effect that these flag commands and the *I/O Reset* (IORST) instruction have on each section of the real-time clock interface.

Select Frequency DOA[f] ac,RTC



Selects the frequency of the real-time clock.

Bits 14 and 15 of the specified accumulator are loaded into the frequency select register. After the data transfer, the command sets the Busy and Done flags according to the function specified by *f*. The format of the specified CPU accumulator before and after the transfer is:



Bit(s)	Name	Function
0-13	—	Not used
14-15	Frequency select	Selects the desired frequency as follows:

Bits 13	Bits 14	
0	0	Line frequency
0	1	10 Hz
1	0	100 Hz
1	1	1000Hz

Programming Guidelines

Programming the real-time clock interface involves the following steps:

Selecting the clock frequency

Enabling interrupt requests

Servicing interrupt requests

Selecting Clock Frequency

To select the clock frequency, issue a *Select Frequency* instruction (DOA) using the appropriate accumulator bits to specify the frequency at which the interface is to interrupt the CPU. To select the clock frequency and enable the real-time clock interrupts, issue a *Select Frequency* instruction with a Start flag command (DOAS) using the appropriate accumulator bits to specify the frequency at which the interface is to interrupt the CPU.

Enabling Interrupt Requests

To enable real-time clock interrupts, issue a *No I/O Transfer* instruction with a Start flag command NIOS to set the Busy flag to 1 and Done flag to 0.

Servicing Interrupt Requests

When each clock period expires, program (Done flag set to 1), issue a *No I/O Transfer* instruction with either a Start (NIOS) or Clear flag command (NIOC). The Start flag command enables another interrupt request (unless masked out or CPU has interrupts disabled) at the expiration of the current clock period by setting the Busy flag to 1 and the Done flag to 0. The Clear flag command disables subsequent interrupt requests by setting both the Busy and Done flags to 0.

I/O Timing

The first interrupt request initiated by the real-time clock can occur at any time up to the full clock period. If the program responds to the real-time clock interrupt requests before each succeeding clock period expires, all subsequent RTC interrupts will occur at the clock frequency.

Power-Up Response

After power-up or when an *I/O Reset* instruction is performed, the line frequency clock rate is selected, and the Busy, Done, and Interrupt Disable flags are set to zero.

Programmable Interval Timer

The programmable interval timer is a CPU-independent time base which can be programmed to initiate program interrupts at fixed intervals ranging from 1 microsecond to 65.536 seconds in increments of 100 microseconds. It can also be interrogated with programmed I/O instructions at any point in its cycle to determine the time remaining until the next interrupt, or following an interrupt to determine interrupt latency.

Programmable Elements

From a programming point of view, the programmable interval timer consists of the following major elements:

Initial Count Register

Clock Counter

Busy Flag

Done Flag

Interrupt Disable flag

Your program directs interface activities at the timer level by manipulating these major elements, using the following interface instructions together with the Start and Clear device flag commands:

Specify Initial Count (DOA)

Read Count (DIA)

I/O Skip (IOSKP)

The program also uses two CPU I/O instructions to manipulate the state of some elements: *I/O Reset* (IORST) and *Mask Out* (MSKO).

Programming Summary

The following programming summary provides general timer specifications; shows the accumulator formats for the timer instructions; and explains the timer's response to the Start and Clear flag commands and the *I/O Reset* instruction (IORST).

Table 2-7 Programming summary:general timer specifications

Mnemonic	PIT
Device code	43 octal
Priority mask bit	6
Time intervals	100 microseconds to 6.5536 seconds
Count rate	100 microseconds

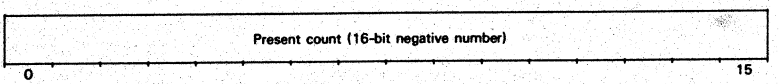
Table 2-8 Programming summary: START, CLEAR, IOPLS, and IORST functions

f = S	Sets the Busy flag to 1, the Done flag to 0, starting or continuing the counting cycle
f = C	Sets the Busy and Done flags to 0, stopping the counting cycle interrupts
f = P	No effect
IORST	Sets the Busy, Done, and Interrupt Disable flags to 0. Also sets the initial count register and timer counter to zeros.

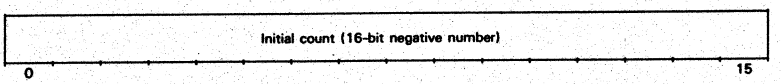
	Programming Summary
Mnemonic	PIT
Device codes	43_8
Priority mask bit	6
Time intervals	100 microseconds to 6.5536 seconds
Count rate	100 microseconds

Accumulator Formats

Read Count (DIA)



Specify Initial Count (DOA)



Start, Clear, IOPLS, and IORST Functions

f=S Sets the Busy flag to 1, the Done flag to 0, starting or continuing the counting cycle

f=C Sets the Busy and Done flags to 0, stopping the counting cycle interrupts

f=P No effect

IORST Sets the Busy, Done, and Interrupt Disable flags to 0. Also sets the initial count register and timer counter to zeros.

DG-25977

Figure 2-10 Programming Summary: programmable interval timer

Table 2-9 Programming Summary: general subsystem specifications

Mnemonic (first drive)	DE0
Mnemonic (second drive)	DE1
Device code	20 octal
Mask bit	7
Capacity per drive:	
DGC standard format	368.64 Kbytes
DGC MPT/100 format	358.40 Kbytes
IBM PC format	327.68 Kbytes
Number of surfaces	2
Heads/Surface	1
Tracks/Head:	
DGC standard format	40
DGC MPT/100 format	35
IBM PC format	40
Sectors/Track:	
DGC standard format	9
DGC MPT/100 format	10
IBM PC format	8
Diskette data transfer rate	250 Kbits/second
Data Channel Latency	62 us
Track Access Time:	
Minimum	21 ms
Average	93 ms (1/3 stroke)
Maximum	249 ms
Recalibrate Time	249 ms max.
Head Load Time	50 ms maximum
Rotational Speed	300 RPM ± 1.5%
Motor On Time	1 second max.
Rotational Latency:	
Average	100 ms
Maximum	200 ms
Data encoding method	Modified frequency modulation (MFM)
Track Density	48 track/inch

Register and Flags

The program-accessible registers of the programmable interval timer include the 16-bit initial count register and the 16-bit counter. Its program-accessible flags include the Busy, Done, and Interrupt Disable flags.

Initial Count Register

The initial count register stores the 16-bit count sent to the interface by the program to specify the number of clock rate intervals between interrupts. The program loads this register using a *Specify Initial Count* instruction (DOA).

Clock Counter

During a timed operation, this counter is first loaded with the count in the initial count register and is then incremented at switch selected time intervals of 100 microseconds. When this counter overflows, the Busy flag sets to 0 and the Done flag sets to 1, thus initiating a timer interrupt request if interrupts are enabled. The program reads the contents of this register using a *Read Count* instruction (DOA).

Busy Flag The Busy flag, when set to 1, starts the counting cycle thus initiating a timed operation. The program manipulates the Busy flag using the *I/O Reset* instruction (IORST) or the Start and Clear flag commands. The *I/O Reset* instruction and Clear flag command sets the Busy flag to 0, stopping the counting cycle; the Start flag command sets the Busy flag to 1, starting the counting cycle interrupts. A program can test the Busy flag using the *I/O Skip* instruction (IOSKP) addressed to the programmable interval timer (PIT).

Done Flag The Done flag, when set to 1, initiates an interrupt request to the CPU (unless interrupts are disabled) and specifies that the time interval has expired. (The count, however, continues.) The program sets the Done flag to 0 using the *I/O Reset* instruction (IORST) or the Start or Clear flag commands. A program can test the Done flag using the *I/O Skip* instruction (IOSKP) addressed to the programmable interval timer.

Interrupt Disable Flag The Interrupt Disable flag enables and disables programmable interval timer interrupts. When the flag is set to 0, timer interrupts are enabled; when it is set to 1, timer interrupts are disabled. The program manipulates the state of the Interrupt Disable flag with either an *I/O Reset* instruction (IORST) or the CPU's *Mask Out* instruction (MSKO). The *I/O Reset* instruction (IORST) sets the Interrupt Disable flag to 0, enabling interrupts. The *Mask Out* instruction (MSKO) sets the Interrupt Disable flag to 0 or 1, depending upon the logical state of bit position 6 of the data word being transferred. For more information on the *Mask Out* instruction, refer to its description in *16-bit Real-Time ECLIPSE Assembly Language Programming.*

I/O Instruction Set

Two programmed I/O instruction enables you to program the programmable interval timer to specify a time interval between interrupts and to determine the time until the next interrupt. These instructions:

Loads the initial count register with the number of clock rate intervals between interrupts. (DOA)

Reads the count remaining before an interrupt is requested, into a specified CPU accumulator (DIA)

Five additional programmed I/O instructions, two addressed to the programmable interval timer and three addressed to the CPU (Device code 77_8) allow the CPU to:

Interrogate the state of the interface Busy and Done flags (IOSKP).

Issue a device flag command without a programmed input/output transfer (NIO).

Enable/disable the interface interrupt facility (MSKO).

Identify an interrupting source (INTA).

Initialize the interface (IORST).

The latter instructions are fully described in detail in *16-bit Real-Time ECLIPSE Assembly Language Programming.*

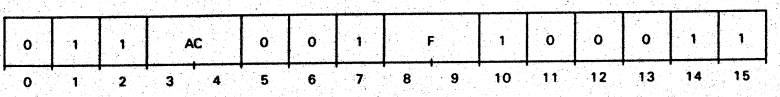
Assembly language coding and instruction bit patterns are shown for each of the two instructions described. The device code field of the instruction is shown with the standard programmable interval timer device code, 43_8. Mnemonics

PIT or 43_8 used in the assembly language coding, address the programmable interval timer.

Two device flag commands, Start (S) and Clear (C), can be specified in the *f* field of programmed I/O instructions. Refer to the "Programming Summary" for the effect that these flag commands and the *I/O Reset* (IORST) instruction have on each section of the programmable interval timer.

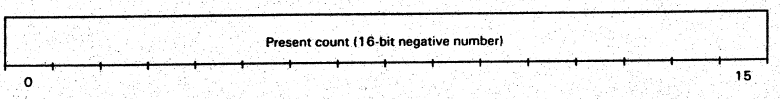
Read Count

DIA*[f]* *ac*,PIT



Reads the count remaining before an interrupt is requested.

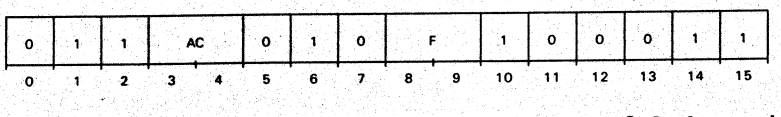
Loads the contents of the counter into bits 0-15 of the specified accumulator. After the data transfer, the command sets the Busy and Done flags according to the function specified by *f*. The format of the specified CPU accumulator after the transfer is:



Bit(s)	Name	Function
0-15	Count	Current value of the PIT counter (2's complement) within one count cycle.

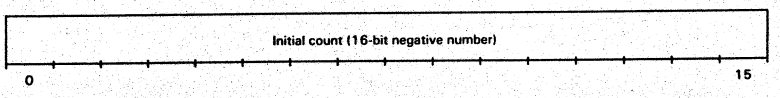
Specify Initial Count

DOA*[f]* *ac*,PIT



Loads the initial count register with the number of clock rate intervals between interrupts.

Bits 0-15 of the specified accumulator are loaded into the initial count register. After the data transfer, the command sets the Busy and Done flags according to the function specified by *f*. The format of the specified CPU accumulator before and after the transfer is:



Bit(s)	Name	Function
0-15	Initial Count	Two's complement of the number of the clock rate intervals between interrupts.

Programming Guidelines

Programming the programmable interval timer involves the following steps:

Specifying a time interval between program interrupts

Starting the counting cycle

Servicing interrupt requests

Selecting Time Interval

To specify a time interval between program interrupts, issue a *Specify Initial Count* instruction (DOA) using the appropriate accumulator bits to specify the time interval between interrupts to the CPU. To specify the time interval and start the counting cycle, issue a *Specify Initial Count* instruction with a Start flag command (DOAS) using the appropriate accumulator bits to specify the time interval between interrupts to the CPU. Note that the number you place in the accumulator determines the time as follows: every 100 microseconds the number specified is decremented by 1. The longest time interval is obtained by loading the accumulator with 0: this causes the contents to be decremented 65,536 times. For the shortest interval, load 1.

Starting the Counting Cycle

To start the timing cycle, issue a *No I/O Transfer* instruction with flag command (NIOS) to set the Busy flag to 1 and Done flag to 0.

Servicing Interrupt Requests

When each time interval expires, the timer sets the Done flag to one, thus initiating an interrupt request if interrupts are enabled. When the interrupt is serviced, issue a *No I/O Transfer* instruction with either a Start (NIOS) or Clear flag command (NIOC). The Start flag command enables another interrupt request (unless masked out or CPU has interrupts disabled) at the expiration of the current time period by leaving the Busy flag set to 1 and setting the Done flag to 0. The Clear flag command disables subsequent interrupt requests by setting both the Busy and Done flags to 0.

I/O Timing

The first interrupt request initiated by the programmable interval timer can occur at any time after the timer is started, up to the full time period. The time between subsequent program interrupt requests will be the value selected by the contents of the initial count register.

Power-Up Response

After power-up or when an *I/O Reset* instruction is performed, Busy, Done, and Interrupt Disable flags are set to zero and the counting cycle is stopped. Also, the initial contents of the count register and the timer counter are set to zero.

Diskette Subsystem

This section describes the programmable elements of the Model 10 and 10/SP diskette subsystem; summarizes its specifications and flag commands and instructions used to program it; defines the relevant I/O instructions; discusses programming considerations and I/O timing; and explains possible error conditions.

Programmable Elements

From a programming point of view, the diskette interface consists of the following major elements:

Command Register

Status Register

Memory Address Register

Word Count Register

Busy Flag

Done Flag

Interrupt Disable flag

IPL Flag

Your program directs subsystem activities at the interface level by manipulating these major elements, using the following interface instructions together with the Start, Clear, and Pulse device flag commands:

Specify Command and Diskette Address (DOA)

Read Diskette Status (DIA)

Load Memory Address Register (DOB)

Read Memory Address Register (DIB)

Load Word Count Register (DOC)

I/O Skip (IOSKP)

The program also uses two CPU I/O instructions to manipulate the state of some elements: *I/O Reset* (IORST) and *Mask Out* (MSKO).

Programming Summary

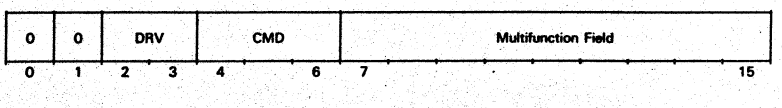
page 2-24

The following programming summary provides general subsystem specifications in Table 2-9; describes diskette formats in Table 2-10; shows the accumulator formats for the five interface instructions in Figure 2-10; and explains the interface's response to the Start, Clear, and Pulse flag commands and the *I/O Reset* instruction (IORST) in Table 2-11.

Table 2-10 Programming summary: diskette formats

DG Standard, 48TPI, 5.25 inch:	
Surfaces/diskette	2 sides per diskette
Surface numbering	1 = label side (right in drive); 0 = opposite side
Tracks/side	40 (48 tracks per inch)
Sectors/track	9
Bytes/sector	512
Total storage capacity	368.64 Kbytes
Byte packing format	High byte/Low byte
IBM PC, 48TPI, 5.25 inch:	
Surfaces/diskette	2
Surface numbering	1 = label side (right in drive); 0 = opposite side
Tracks/side	40 (48 tracks per inch)
Sectors/track	8
Bytes/sector	512
Total storage capacity	327.68 Kbytes
Byte packing format	Low byte/High byte
DG MPT/100, 48TPI, 5.25 inch: (this format is read only)	
Surfaces/diskette	1 or 2
Surface numbering	0 = label side (right in drive); 1 = opposite side
Tracks/side	35 (48 tracks per inch)
Sectors/track	10
Bytes/sector	512
Total storage capacity	358.4 Kbytes
Byte packing format	High byte/Low byte

Specify Command and Diskette Address DOA

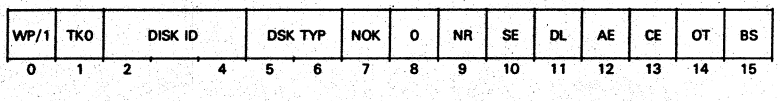


Commands

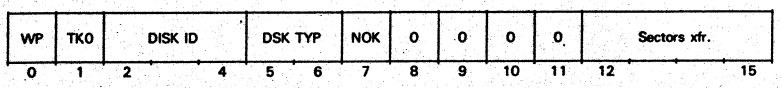
000	Read
001	Read header
010	Write
011	Format track
100	Diagnostic operation
101	Get number of sectors transferred
110	Seek
111	Recalcibrate

Read Diskette Status (DIA)

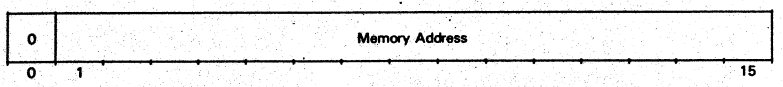
Context: A Get Number of Sectors Transferred was not previously issued.



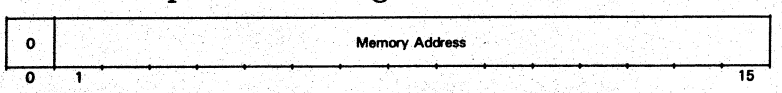
Context: A Get Number of Sectors Transferred was previously issued.



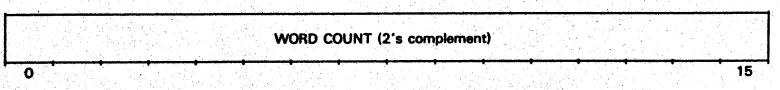
Load Memory Address Register DOB



Read Memory Address Register DIB



Load Word Count (DOC)



DG-25978

Figure 2-11 Programming Summary: accumulator formats

Table 2-11 Programming summary: START, CLEAR, IOPLS, and IORST functions

f = S	Sets the Busy flag to 1, the Done flag to 0, and initiates a command. When issued after an **IORST** instruction, initiates the IPL operation.
f = C	Sets the Busy and Done flags to 0 and initializes the interface. Current track position of the heads is lost. Also sets the operating modes of the drive(s) to their default state as follows: Track density: 48 tracks-per-inch Diskette surfaces: 2 Number of sectors/track: 9 (physical 1-9) Number of words/sector: 256 (512 bytes) Sector addressing: logical (sectors 0-8) Head position 1: label side (right side when installed in drive) 0: opposite side (left side when installed in drive) Byte packing: high byte first, low byte second
f = P	Sets the Done flag to 0.
IORST	Performs all the functions listed under f = C. Also clears the diskette memory address register, sets the Initiate Program Load (IPL) flag to 1, and sets the Interrupt Disable flag to 0.

NOTE *The Pulse flag command should be used to clear program interrupts since the Clear flag command and* ***IORST*** *instruction cause the interface to lose position information for the drives. Thereafter, a Seek command must be issued before a Read or a Write operation can be performed (the interface automatically recalibrates the drive before it processes the Seek command).*

Registers and Flags

The program-accessible registers of the diskette interface include the command, status, memory address, and word count registers. Its program-accessible flags include the Busy, Done, Interrupt Disable, and Initial Program Load (IPL) flags.

Command Register

The command register stores a 3-bit encoded subsystem command and a drive select code. The register also contains a multifunction field that stores the destination track address for a seek operation, the head and starting sector number for a data transfer, the operating mode characteristics for a set mode operation, and a diagnostic operation code to support diagnostic commands. The program loads the command register using a *Specify Command and Diskette Address* instruction (DOA).

Status Register

The status register maintains the following subsystem status information: subsystem and diskette drive identity; ability of the diskette drive to accept commands; positioning and data transfer error flags; presence of drive read/write heads over track 00; and number of sectors transferred. The program can read the contents of the status register at any time by issuing a *Read Diskette Status* instruction (DIA). The program obtains the number of sectors transferred by preceding the *Read Diskette Status* instruction (DIA) with a *Specify Command and Diskette Address* instruction (DOA) specifying a Get Number of Sectors Transferred command.

Memory Address Register

The memory address register is self-incrementing and contains the address of the memory location to be accessed as the source or destination of the next data channel transfer. The program loads and reads the memory address register using a *Load Memory Address Register* instruction (DOB) or *Read Memory Address Register* instruction (DIB) respectively.

The program can also set the memory address register to all zeros by issuing an *I/O Reset* instruction (IORST).

Word Count Register The word count register is self-incrementing and contains the two's complement of the number of words remaining for a data channel transfer. The program loads the word count register using a *Load Word Count Register* instruction (DOC).

Busy Flag The Busy flag, when set to 1, initiates a diskette operation and specifies that the subsystem is performing it. This flag remains set until the operation is completed. The program manipulates the Busy flag using the *I/O Reset* instruction (IORST) or the Start and Clear flag commands. The *I/O Reset* instruction and Clear flag command set the Busy flag to 0; the Start flag command sets the Busy flag to 1. A program can test the Busy flag using the *I/O Skip* instruction (IOSKP) addressed to the diskette subsystem.

Done Flag The Done flag, when set to 1, initiates an interrupt request to the SPU (unless interrupts are disabled) and specifies that the subsystem has completed an operation or that an error condition exists. The program sets the Done flag to 0 using the *I/O Reset* instruction (IORST) or the Start, Clear, or Pulse flag commands. A program can test the Done flag using the *I/O Skip* instruction (IOSKP) addressed to the diskette subsystem.

Interrupt Disable Flag The Interrupt Disable flag enables and disables diskette subsystem interrupts. When the flag is set to 0, subsystem interrupts are enabled; when it is set to 1, subsystem interrupts are disabled. The program manipulates the state of the Interrupt Disable flag with either an *I/O Reset* instruction (IORST) or the CPU's *Mask Out* instruction (MSKO). The *I/O Reset* instruction (IORST) sets the Interrupt Disable flag to 0, enabling interrupts. The *Mask Out* instruction (MSKO) sets the Interrupt Disable flag to 0 or 1, depending upon the logical state of bit position 7 of the data word being transferred. For more information on the *Mask Out* instruction, refer to its description in *16-bit Real-Time ECLIPSE Assembly Language Programming*.

Initial Program Load (IPL) Flag The IPL flag, when set to 1, enables an initial program load sequence to be initiated by a Start flag command. The program manipulates the IPL flag using either the *I/O Reset* instruction (IORST) or the *Specify Command and Diskette Address* instruction (DOA). The *I/O Reset* instruction sets the IPL flag to 1, enabling IPL. (The IPL sequence does not initiate until a Start flag command is performed.) The *Specify Command and Diskette Address* instruction sets the IPL flag to 0, disabling IPL. The IPL sequence will be described in the programming section below.

I/O Instruction Set

Five programmed I/O instructions enable you to program the diskette interface to perform data transfers to and from the diskette subsystem. These instructions:

Select a drive and specify a command and command parameters. Depending on the command, the command parameters specify positioning information for the selected drive's read/write heads; a head and/or sector where the data is to be recorded to or read from; a diagnostic operation; or drive operating mode characteristics (DOA).

Transfer interface and drive status information to a specified CPU accumulator (DIA).

Specify the starting memory address for transferring data between memory and a diskette drive via the data channel (DOB).

Return the address of the memory location to be used as the source or destination of the next data channel transfer to a specified CPU accumulator (DIB).

Specify the number of words (in two's complement) to be transferred between memory and a diskette drive via the data channel (DOC).

Five additional programmed I/O instructions, two addressed to the diskette subsystem and three addressed to the CPU (Device code 77_8) allow the CPU to:

Interrogate the state of the interface — busy, done, or idle (IOSKP).

Issue a device flag command without a programmed input/output transfer (NIO).

Enable/disable the interface interrupt facility (MSKO).

Identify an interrupting source (INTA).

Initialize the subsystem (IORST).

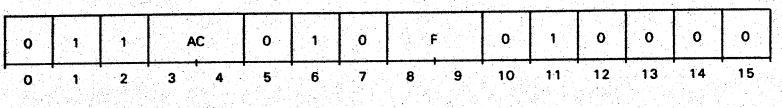
The latter instructions are described in detail in *16-bit Real-Time ECLIPSE Assembly Language Programming*.

Assembly language coding and instruction bit patterns are shown for each of the five instructions described. The device code field of the instruction is shown with the standard Model 10 and 10/SP diskette interface device code, 20_8. Mnemonic DE0, DE1, or 20_8, used in the assembly language coding, address the diskette interface.

Three device flag commands, Start (S), Clear (C) and PULSE (P), can be specified in the *f* field of programmed I/O instructions. Refer to the "Programming Summary" for the effect that these flag commands and the *I/O Reset* (IORST) instruction have on the diskette subsystem.

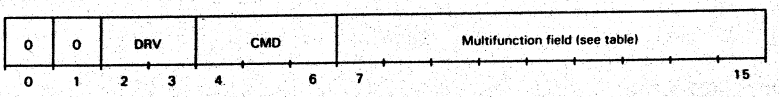
Specify Command and Diskette Address

DOA*[f]* *ac*,MNEMONIC



Loads the diskette interface command register with the command and its parameters.

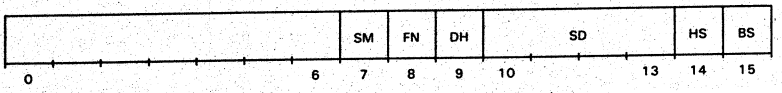
Bits 0-15 of the specified accumulator are loaded into the diskette interface command register. After the data transfer sets the interface Busy and Done flags according to the function specified by *f*. The format of the specified accumulator before and after the transfer is as follows:

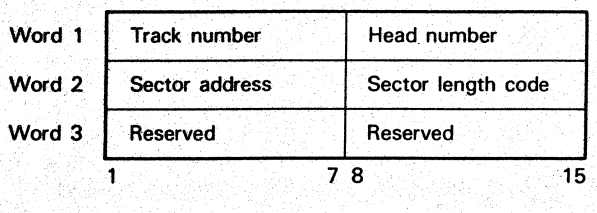


Bit(s)	Name	Function
0-1	—	Reserved (must be set to 0).
2-3	Drive Select	Indicates which diskette drive is to carry out the specified operation, as follows: 00 = drive 0 01 = drive 1 10 = reserved 11 = reserved
4-6	Diskette Command	Specifies the diskette command to be performed, as follows: 000 = Read 001 = Read header 010 = Write 011 = Format track 100 = Diagnostic operation (specified in bits 11-15) 101 = Get number of sectors transferred 110 = Seek 111 = Recalibrate See Table 2-12 for description of diskette commands.
7-15	Multifunction field	Function determined by the contents of the command field, as explained below.
	Track	If Seek command: bits 7-8 are reserved (must be 0) bits 9-15 specify the target track address
	Head/Sector	If Read, Write, or Format track command: bits 7-9 are reserved (must be 0) bit 10 specify the head to select bits 11-15 specify the starting sector address to receive or transmit data (can be any sector address for Read Header or Format Track command)
	Head	If Read Header command: bits 7-9 are reserved (must be 0) bit 10 specify the head to select bits 11-15 are reserved (must be 0)
	Diagnostic	If Diagnostic operation command: bits 7-10 are reserved (must be 0 unless used by the diagnostic command) bits 11-15 specify the diagnostic operation to be performed (see Appendix B)
	Recalibrate/ Set Mode	If Recalibrate command: bit 7 selects command context, where 0 = Recalibrate and 1 = Recalibrate and Set Mode bits 8-15 specify the operating characteristics for the diskette interface when Set Mode is specified (bit 7 = 1) (see "Recalibrate/Set Mode Command" in this section); otherwise ignored

Table 2-12 Diskette commands

Command	Description
Read	Reads and transfers one or more sectors of data to host memory, beginning at the sector specified and continuing until the specified number of words are read. Head boundaries can be crossed where appropriate; that is, a read operation beginning on head 0 can continue on head 1. Read operations beginning on head 1 cannot continue on head 0 and will result in the setting of the Address Error flag.
Read Header	Reads and transfers three words, of the first address field that passes the selected head of the selected drive, to the host memory. (The *Load Word Count Register* instruction (DOC), issued before this command, should specify that three words are to be transferred to memory.) The format of the three words returned to memory are shown in Chapter 2, the Read Header command figure.
Write	Transfers and writes one or more sectors of data from the host memory, beginning at the sector specified and continuing until the specified number of words are written. Head boundaries can be crossed where appropriate; that is, a write operation beginning on head 0 can continue on head 1. Write operations beginning on head 1 cannot continue on head 0 and will result in the setting of the Address Error flag.
Format Track	Formats an entire track by transferring and writing an entire track of data from the host memory. Prior to issuing this command, a buffer in memory must be set up to contain the image of an entire track. (Refer to "Formatting a Diskette" in this chapter.)
Diagnostic	Performs a specified diagnostic operation. (Refer to Appendix B.)
Get Number of Sectors	Transfers a count of the number of sectors transferred during the last diskette drive operation to the host CPU, along with selective Transferred status information of the currently selected diskette drive. The actual transfer does not take place as a result of this command; instead, the information is loaded into a temporary register for transfer to the CPU during a {Read Diskette Status} command that follows this command. (Refer to "Read Diskette Status" under "I/O Instruction Set" in this chapter.)
Seek	Positions the selected diskette drive's read/write heads to a specified track. The diskette interface does not verify that the heads are positioned over the correct track.
Recalibrate	Positions the selected diskette drive's read/write heads to the track 00 position. If track 00 is not found within head 100 steps, the Seek Error bit is set in the status register. When issued with the Set Mode bit set to 1, this command performs the recalibration described above and, in addition, sets the operating modes for the diskette drive(s) as determined by bits 8 through 15 of the specified accumulator. The operating mode bits, shown below and described in the Operating mode bit description table, can be different for each drive connected to the diskette interface.





NOTE *Sector length code* $= 10_{16} = 512$ *bytes/sector*

ID-00620

Figure 2-12 Read Header command: format of words returned

Table 2-13 Operating mode bit descriptions

Bit(s)	Name	Function
7	Set Mode (SM)	Specifies Recalibrate or Recalibrate and Set Mode, as follows: 0 = Recalibrate (bits 8-15 are ignored) 1 = Recalibrate and Set Mode
8	48/96 (FN)	Specifies number of positioning step pulses per track as follows: 0 = one step pulse per track (default value)[1] 1 = two step pulses per track, allows low track density (48 TPI) diskettes to be read on high track density (96 TPI) drives[2]
9	Double Headed Media (DH)	Specifies diskette media as follows: 0 = single sided media 1 = double sided media (default value)[1]
10-13	Sector Description (SD)	Specifies the number of sectors per track, number of words per sector, diskette size, and sector addressing, as follows: 0000 = 8 sectors, 256 words, 5.25" diskette, logical sector addressing 0-7, physical sector addressing 1-8; this option is used to logically access IBM PC formatted diskettes 0001 = 8 sectors, 256 words, 5.25" diskette, logical sector addressing 1-8, physical sector addressing 1-8; this option is used to physically access IBM PC formatted diskettes 0010 = 9 sectors, 256 words, 5.25" diskette, logical sector addressing 0-8, physical sector addressing 1-9; this option is the default value and is used to logically access standard Data General 9-sector formatted diskettes[1] 0011 = 9 sectors, 256 words, 5.25" diskette, logical sector addressing 1-9, physical sector addressing 1-9; this option is used to physically access standard Data General 9-sector formatted diskettes

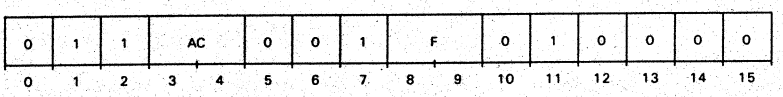
Bit(s)	Name	Function
		0100 = 10 sectors, 256 words, 5.25" diskette, logical sector addressing 0-9, physical sector addressing 0-9; this option is used to physically access Data General MPT/100 10 sector formatted diskettes
		0101-1111 = reserved
14	Head Swap (HS)	Specifies head position as follows: 0 = head 0 on left, head 1 on right (label side of media) (default value)[1] 1 = head 0 on right (label side of media), head 1 on left
15	Byte Swap (BS)	Specifies byte first on/off the diskette as follows; 0 = high order byte (bits 0-7) (default value)[1] 1 = low order byte (bits 8-15)

[1] *(Default value) specifies the value that each diskette drive operating mode is set to on power-up, when an IORST instruction is issued, or a when Clear flag command is specified in any I/O instruction addressed to the diskette interface.*

[2] *If an attempt is made to set the FN bit to 1 on a 48 TPI diskette drive, Address Error will be set in the status register. The recalibration will be performed.*

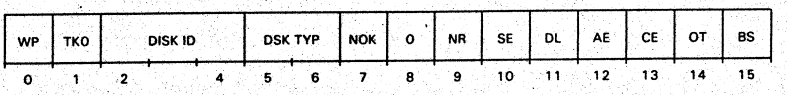
Read Diskette Status

DIA*[f]* *ac*,MNEMONIC



Transfers the status, or selective status along with a number of sectors transferred, of the currently selected diskette drive to the processor.

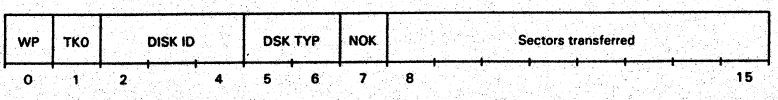
If a command other than *Get Number of Sectors Transferred* was previously issued to the diskette interface, this command loads status information for the currently selected diskette drive into specific bit positions of the specified CPU accumulator. After the data transfer, the command sets the Busy and Done flags according to the function specified by *f*. The format of the specified accumulator after the transfer is:



Bit(s)	Name	Function
0	Write Protect	When set to 1, specifies that the diskette inserted in the selected diskette drive is write-protected.
1	Track 00	Specifies that the selected disk drive has its head positioned over track zero.
2-4	Subsystem Identity	Identifies the diskette subsystem; always set to a value of 2_8 (010).
5-6	Diskette Type	Identifies the diskette drive type and capacity as follows: 00 = double-sided, 48 TPI 01 = reserved 10 = reserved 11 = reserved

Bit(s)	Name	Function
7	Not OK	Specifies that the interface failed its self-test after either a power-up or a self-test diagnostic command. This bit resets when the interface passes its self-test diagnostic.
8	—	Reserved (always set to 0).
9	Not Ready	Specifies that the selected disk drive did not obey the command. This may be because the drive is not up to speed, no diskette is inserted, or the currently selected drive is not present.
10	Seek Error	Specifies that the Track 00 signal failed to assert during a recalibrate operation (or a Seek command in which the interface imbedded a recalibrate operation) and the interface cannot determine the current positioner location. This error is also set if a Seek command is issued with a track address that exceeds the maximum number of tracks of the drive.
11	Data Late	Specifies that data was lost because the data channel facility could not keep up with the demands of the diskette data transfer.
12	Address Error	Specifies one of the following: the interface was unable to find the desired sector a Read or Write command was issued with a head or sector address that exceeds the maximum number of heads or sectors of the drive an attempt was made to transfer a sector past the end of the current cylinder the Bad Sector error flag is set a Recalibrate and Set Mode command attempted to set the 48/96 option bit for a 48 TPI drive to 1.
13	Checkword Error	Specifies that the checkword read from the current sector did not compare with the checkword calculated by the interface during a read operation.
14	Operation Time-out	Indicates that the specified operation failed to complete in a designated amount of time.
15	Bad Sector	Indicates that the last sector accessed found a deleted data mark in the data field. This condition can only be encountered when reading diskette media written on a non-Data General system. Also causes address error to be set.

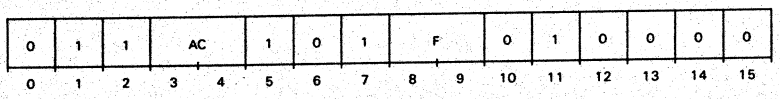
If the previous command issued to the diskette interface was a *Get Number of Sectors Transferred*, this command loads selective status information of the currently selected diskette drive and a sectors transferred count into specific bit positions of the specified CPU accumulator. After the data transfer, the command sets the Busy and Done flags according to the function specified by *f*. The format of the specified CPU accumulator following the transfer is:



Bit(s)	Name	Function
0	Write Protect	When 1, specifies that the diskette inserted in the selected drive is write-protected.
1	Track 00	When 1, specifies the selected disk drive has its head positioned over track zero.
2-4	Subsystem Identity	Identifies the diskette subsystem; always set to 010 (2_8).
5-6	Diskette Type	Identifies the diskette drive type and capacity, as follows: 00 = double-sided, 48 TPI 01 = reserved 10 = reserved 11 = reserved
7	Not OK	When 1, specifies that the interface failed its self-test after either a power-up or a self-test diagnostic command. This bit resets when the interface passes its self-test diagnostic.
8-15	Sectors Transferred	Specifies the number of sectors transferred during the last diskette drive operation that preceded the Get Number of Sectors Transferred command.

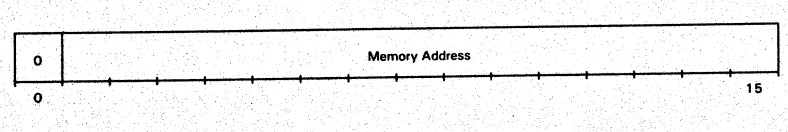
Load Memory Address Register

DOB*[f]* *ac*,MNEMONIC



Loads the diskette interface memory address register with the address of the first memory location to be accessed for a data channel transfer.

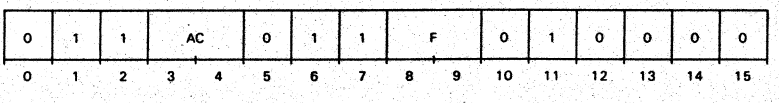
Bits 1-15 of the specified accumulator are loaded into the memory address register. After the data transfer, the command sets the interface Busy and Done flags according to the function specified by *f*. The format of the specified accumulator before and after the transfer is:



Bit(s)	Name	Function
0	—	Reserved (must be set to 0).
1-15	Memory Address	Location in memory to be accessed for the first data channel transfer.

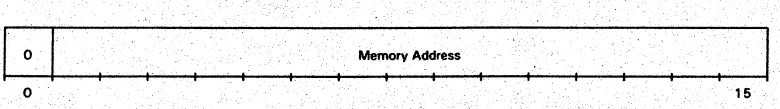
Memory Address Register

DIB*[f]* *ac*,MNEMONIC



Returns to the CPU, the address of the memory location to be accessed for the next data channel transfer.

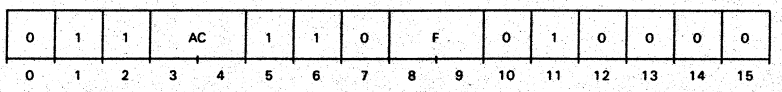
Loads the contents of the memory address register into bits 1-15 of the specified accumulator. Bit 0 of the specified accumulator is set to zero. After the data transfer, the command sets the interface Busy and Done flags according to the function specified by *f*. The format of the specified CPU accumulator after the transfer is:



Bit(s)	Name	Function
0	—	Reserved (always set to 0).
1-15	Memory Address	Location in memory to be accessed for the next data channel transfer.

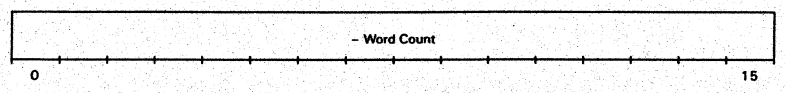
Load Word Count

DOC*[f]* *ac*,MNEMONIC



Loads the word count register with the number of 16-bit words to be transferred during the following read/write operation.

Bits 0-15 of the specified accumulator are loaded into the word count register. After the data transfer, the command sets the interface Busy and Done flags according to the function specified by *f*. The format of the specified accumulator before and after the transfer is:



Bit(s)	Name	Function
0-15	– Word Count	Two's complement of the number of words to be transferred during a data channel transfer.

NOTES *Word count is calculated as the number of sectors times the number of words per sector.*

Only integral numbers of sectors may be transferred.

Programming Guidelines

Factors involved in programming the diskette subsystem are the subject of this discussion. Programming the subsystem involves the following steps:

Initiating an operation, including selecting the drive.

Determining diskette format, including defining the surface position and number of sectors per track.

Setting the diskette drive operating mode, including track density, number of heads, sector description, head swap, and byte swap.

Positioning the drive's read/write heads over the desired track.

Setting up a data transfer, including specifying a starting memory address that will be the source or destination of a block data transfer between memory and the diskette subsystem, specifying the number of words to be transferred, selecting a starting head and sector, and issuing a data transfer command.

The following discussion elaborates on each step. These programming sequences assume that no errors occur during head positioning or data transfer operations. Any error conditions should therefore be corrected by referring to the section, "Error Conditions," below.

Initiating An Operation

The following steps ensure that the subsystem is ready to perform an operation and select a drive.

1. Issue an *I/O Skip* instruction (IOSKP) to ensure that the interface Busy flag is set to 0.
2. Issue a *Specify Command and Diskette Address* instruction (DOA) with no flag command, using the appropriate accumulator bits to select a drive. (Any command can be specified.)

Determining the Diskette Format

Before performing diskette read or write operations, it is necessary that the format of the diskette(s) installed in the drive(s) be determined and the diskette operating mode be set, as required, to handle the diskette format. The format of the diskette installed in a drive can be established by determining the surface position and number of sectors per track, as detailed below.

Use the following programming sequence to determine if head swapping is required. Ensure that the diskette subsystem is ready to perform an operation (explained earlier under "Initiating an Operation").

1. Issue a *Specify Command and Diskette Address* instruction (DOA) with no flag command, using the appropriate accumulator bits to select a drive, specify a Read Header command, and specify a head number.
2. Issue a *Load Memory Address Register* instruction (DOB) with no flag command, using the appropriate accumulator bits to specify the address of the first memory location to receive the diskette header data.
3. Issue a *Load Word Count Register* instruction (DOC) with a Start flag command, using the appropriate accumulator bits to specify three (in two's complement) 16-bit words to be transferred. The Start flag command sets the interface's Busy flag to 1, sets the Done flag to 0, and initiates the Read Header operation.

4. After the Read Header operation is completed on the selected drive, issue a *Read Status* instruction (DIA) with a Pulse flag command and check the error flags. (The Pulse flag command clears the Done flag and interrupt request.) If a Data Late or Checkword Error occurred, try the data transfer again.

5. Interrogate the head number field to determine which surface of the diskette was read. If the head swapping mode was not enabled, the Read Header command in Step 1 selected head 0, and the head number read is 0, the diskette does not need head swapping enabled. (See the description of the Recalibrate command in Table 2-13 for head swap operating mode.) If the head number read is 1, the diskette is in Data General MPT/100™ format for which head swapping and byte swapping must be enabled. Also for MPT/100 formatted diskettes, the sector description option must be set to option 4 (SD field = 0100).

6. The number of sectors per track and the first sector number can be determined by issuing Read commands to see if a particular sector exists. (Refer to "Setting Up Data Transfers" in this section for the steps taken to program a Read operation.) If a Read operation to physical sector 8 completes with no errors but a Read operation to physical sector 9 produces a hard address error, the last sector number is 8 and the diskette in the selected drive is formatted in IBM PC format (8 sectors/track, physically numbered 1 through 8); for this format, the sector description option can be set to either option 0 or option 1 (SD field 0000 or 0001). If a read operation to physical sector 9 completes with no errors, the diskette in the selected drive is formatted in standard Data General format (9 sectors/track, physically numbered 1 through 9); for this format, the sector description option can be set to either option 2 or option 3 (SD field 0010 or 0011).

Setting the Operating Mode

The Set Operating Mode operation is performed when the Set Mode bit of a Recalibrate command is 1. In addition to positioning the selected diskette drive read/write heads at track 00, this operation sets the selected diskette drive's operating mode as specified. This operation is not necessary when the drive is to operate in the default operating modes. The description of the Recalibrate command in Table 2-13 presents the operating modes, their bit positions, their default values, and their functions.

Use the following programming sequence to set the operating mode of the diskette interface. Ensure that the diskette subsystem is ready to perform an operation (explained under "Initiating an Operation").

1. Issue a *Specify Command and Diskette Address* instruction (DOA) with a Start flag, command using the appropriate accumulator bits to select a drive, issue a Recalibrate command (seek to track 00), specify the Set mode operation, and select the settings of the operating mode bits.

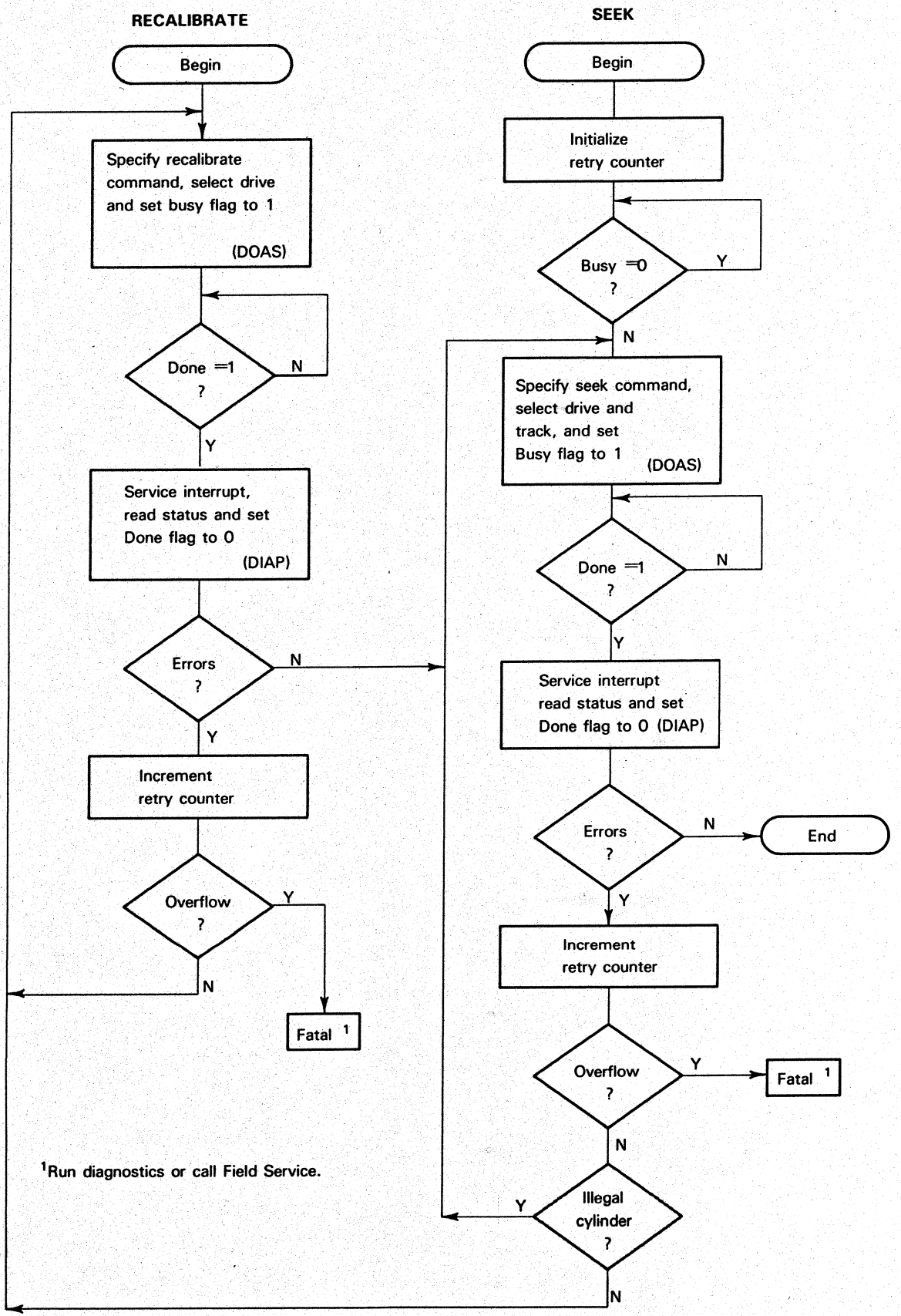
2. After the Recalibrate and Set Operating Mode procedures are completed, issue a *Read Status* instruction (DIA) with a Pulse flag command and check the Seek Error and Operation timeout flags. (The Pulse flag command clears the Done flag and interrupt request.) If a recoverable error occurred, try the operation again.

Positioning the Read/Write Heads

Positioning operations move the read/write heads to the desired track for a data transfer or format operation. When a Recalibrate command is issued, the selected drive positions its heads at track 00. (The drives detect a mechanical reference point to recalibrate on track 00.) When a Seek command is issued, the positioner moves the heads in the direction and number of steps required to arrive at the desired track. It is important to note that the positioner must be recalibrated before the first Seek operation can occur. (Note, too, that the positioner is automatically recalibrated during the first Seek operation that follows an IORST instruction or a Clear flag command.)

Use the following programming sequence to position the heads over a selected track. Refer to Figure 2-13. Make sure the diskette subsystem is ready to perform an operation (explained earlier under "Initiating an Operation").

1. Issue a *Specify Command and Diskette Address* instruction (DOA) with a Start flag command, using the appropriate accumulator to select a drive, and specify either a Recalibrate command (seek to track 00) or a Seek command and track address.
2. After the Recalibrate or Seek operation is completed, issue a *Read Status* instruction (DIA) with a Pulse flag command and check the Seek Error and Operation timeout flags. (The Pulse flag command clears the Done flag and Interrupt request.) If a recoverable error occurred, try the operation again.


RECALIBRATE
Begin
Specify recalibrate command, select drive and set busy flag to 1
(DOAS)
Done =1 ?
N
Y
Service interrupt, read status and set Done flag to 0
(DIAP)
Errors ?
N
Y
Increment retry counter
Overflow ?
Y
N
Fatal 1
SEEK
Begin
Initialize retry counter
Busy =0 ?
Y
N
Specify seek command, select drive and track, and set Busy flag to 1
(DOAS)
Done =1 ?
N
Y
Service interrupt read status and set Done flag to 0 (DIAP)
Errors ?
N
End
Y
Increment retry counter
Overflow ?
Y
Fatal 1
N
Illegal cylinder ?
Y
N
1Run diagnostics or call Field Service.

Setting Up a Data Transfer

Read and write operations can transfer one to 18 (16 in IBM PC format, 20 in Data General MPT/100 format) 512-byte data blocks between memory and the diskette drive. The number of data blocks transferred depends on the word count specified by a preceding *Load Word Count Register* instruction (DOC). Data is transferred through the data channel facility, starting at the memory location specified by a preceding *Load Memory Address Register* instruction (DOB). Data transfers are performed on a demand basis — that is, a data channel transfer occurs each time the diskette interface requires a 16-bit data word — and are double-word buffered to reduce the probability of data-late conditions.

Observe the following precautions before proceeding with a data transfer operation:

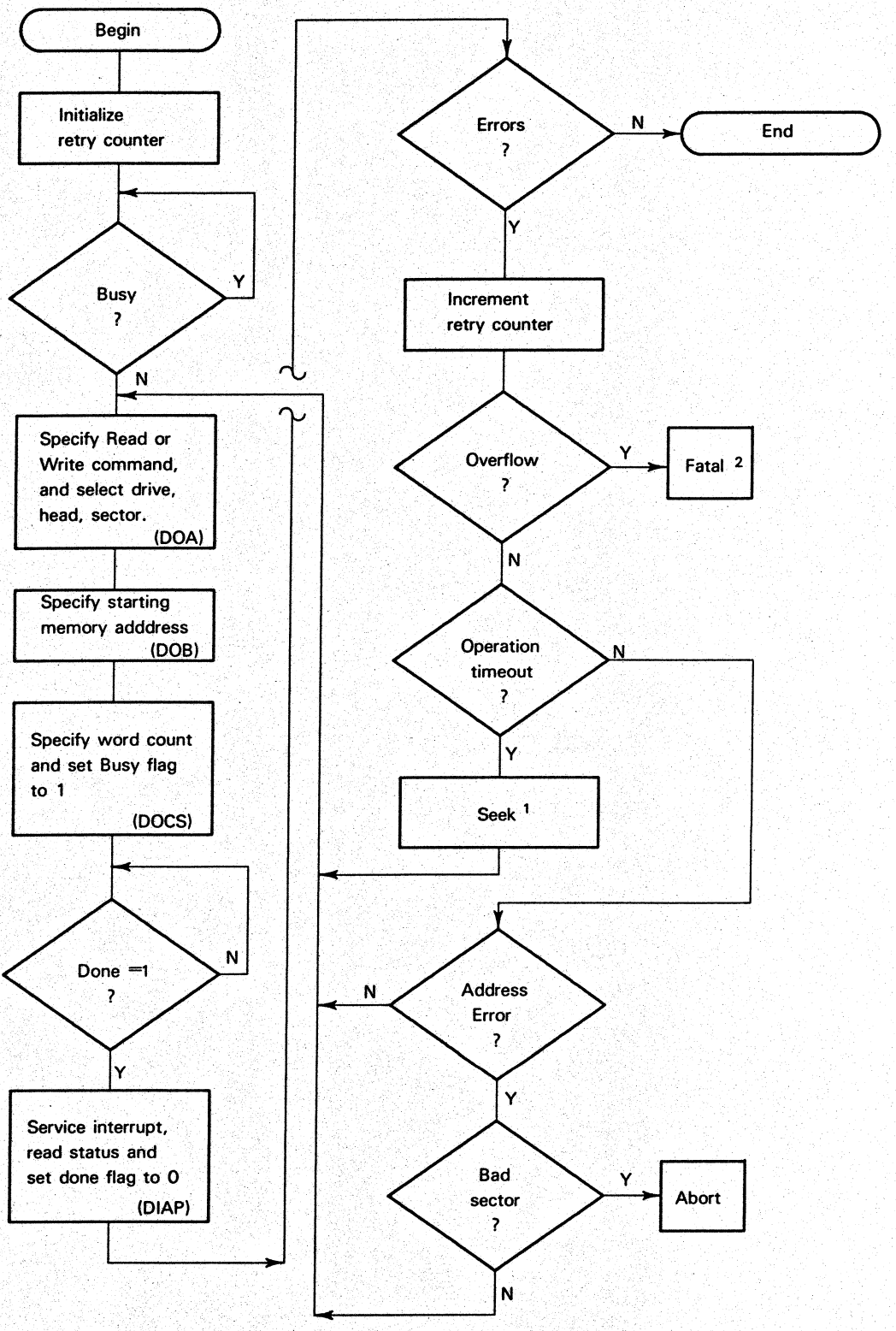
Be sure that the heads are positioned over the desired cylinder. (Refer to "Positioning the Read/Write Heads" in this section.)

Be sure that data is recorded on the diskette before attempting a Read operation.

Do not initiate a multiple-sector transfer that crosses a cylinder boundary (transfers beyond the last sector of head 1).

Continue with the following programming sequence to read data from the diskette. Refer to Figure 2-14. Ensure that the diskette subsystem is ready to perform an operation (explained earlier under "initiating an Operation").

1. Issue a *Specify Command and Diskette Address* instruction (DOA) with no flag command, using the appropriate accumulator bits to select a drive and to specify a Read command, a starting head number, and starting sector number.
2. Issue a *Load Memory Address Register* instruction (DOB) with no flag command, using the appropriate accumulator bits to specify the address of the first memory location to receive the read data block(s).
3. Issue a *Load Word Count Register* instruction (DOC) with a Start flag command, using the appropriate accumulator bits to specify the number (in two's complement) of 16-bit words to be transferred. The Start flag command sets the interface's Busy flag to 1, sets the Done flag and Interrupt request to 0, and initiates the Read operation.
4. After the Read operation is completed on the selected drive, issue a *Read Status* instruction (DIA) with a Pulse flag command and check the error flags. (The Pulse flag command clears the Done flag and Interrupt request.) If a Data Late or Checkword Error occurred, try the data transfer again. If an address error occurred, recalibrate and reposition the positioning mechanism, and retry the transfer.



1 See head positioning figure
2 Run diagnostics for call Field Service.

ID-00614

Figure 2-14 Read or Write Data

Continue with the following programming sequence to write data onto the diskette. Refer to Figure 2-14. Ensure that the diskette subsystem is ready to perform an operation (explained earlier under "Initiating an Operation").

1. Set up the entire data to be written, in a memory buffer.
2. Issue a *Specify Command and Diskette Address* instruction (DOA) with no flag command, using the appropriate accumulator bits to select a drive and to specify a Write command, a starting head number, and starting sector number.
3. Issue a *Load Memory Address Register* instruction (DOB) with no flag command, using the appropriate accumulator bits to specify the address of the first memory location that contains the write data block(s).
4. Issue a *Load Word Count Register* instruction (DOC) with a Start flag command, using the appropriate accumulator bits to specify the number (in two's complement) of 16-bit words to be transferred. The Start flag command sets the interface's Busy flag to 1, sets the Done flag and Interrupt request to 0, and initiates the Write operation.
5. After the Write operation is completed on the selected drive, issue a *Read Status* instruction (DIA) with a Pulse flag command and check the error flags. (The Pulse flag command clears the Done flag and Interrupt request.) If a Data Late error occurred, retry the data transfer. If an address error occurred, recalibrate and reposition the positioning mechanism, and try the transfer again.

Reformatting a Diskette

Data General diskettes are completely formatted before they are shipped. You do not need to reformat a diskette unless problems develop. Data General supplies a stand-alone reformatting program on the Customer Mode Diagnostic diskette (DGC No.). Its operation is described in *Testing Model 10 and 10/SP Systems* (DGC No. 014-000902). The information supplied in this section is intended for those users who may, nevertheless, want to write their own diskette formatting programs.

Model 10 and 10/SP diskettes are *soft sectored:* there are no physical reference points for sectors. Instead, there is a unique address field at the beginning of each sector that identifies the sector's physical address — its track, surface, and sector numbers.

During a formatting operation, the SPU must provide data for the track to be formatted. An entire track must be formatted at once: individual sectors cannot be reformatted. The formatting information includes the track, sector and head addresses, along with the sector length code to be recorded in the address field of each sector.

When a track is reformatted, the previously recorded information is lost. If a particular sector goes bad, the recorded data in the usable sectors of the track must be recovered first, and then the entire diskette track reformatted. Formatting must be performed independently of and before initializing the diskette with a Data General operating system.

The programming directions below explain how you can format your own diskettes in either Data General standard or IBM PC format. First, however, some background information is necessary.

Three diskette formats can be used on a Model 10 and 10/SP diskette subsystem, as indicated in Figure 2-15 through Figure 2-17.

1. Data General standard 9 sector, 512 bytes/sector
2. Data General MPT/100 10 sector, 512 bytes/sector
3. IBM PC 8 sector, 512 bytes/sector

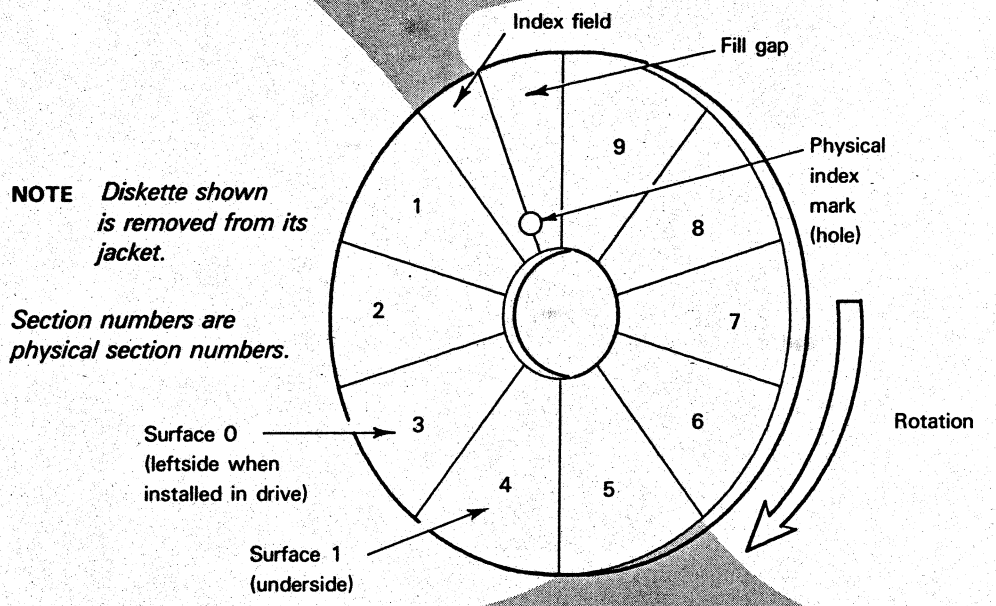
Each diskette has two surfaces, and each surface contains 40 recording tracks. The top surface is labeled 0. One particular track position of each surface constitutes a diskette cylinder. Data General standard formatted diskettes contain nine sectors per track, physically addressed 1 through 9. IBM PC formatted diskettes contain eight sectors per track, physically addressed 1 through 8. MPT/100 formatted diskettes contain ten sectors per track, physically addressed 0 through 9. All three formatted sectors can store up to 512 data bytes.

Note that the diskette interface can be programmed to accept logical instead of physical software sector addresses for standard Data General and IBM PC formatted diskettes. Logical sector addresses for both formats are 0 through 8. When the diskette interface is programmed in this way, it transfers the physical sector number that is one greater than the addressed logical sector.

Surface zero of standard Data General and IBM PC formatted diskettes is located on the left side of the diskette when installed in drive, while surface 1 is located on the right (label) side of the diskette. On Data General MPT/100 formatted diskettes, these positions are reversed.

Diskettes formatted in standard Data General and IBM PC formats include a recorded index field that precedes physical sector 1 of each track. This index field is recorded during the initial formatting and is never written again in normal operation. Diskettes formatted in Data General MPT/100 format do not include the recorded index field. Although you can read diskettes that are formatted in MPT/100 format, you cannot write to these diskettes, nor can you reformat them.

Field	Gap 4A	Sync zone	Index mark	Gap 1
No. of bytes	80	12	4	50
Value in hex	4E's	00's	C2C2C2FC	4E's



NOTE *Diskette shown is removed from its jacket.*

Section numbers are physical section numbers.

Field	Synch zone	Address mark	Cylinder number	Surface number	Sector number	Sector length code
No. of bytes	12	4	1	1	1	1
Value in hex	00's	A1A1A1FE	xx	xx	xx	02

Field	Address field CRC	Gap 2	Sync zone	Data mark	Data field	Data field CRC	Gap 3
No. of bytes	2	22	12	4	512	2	80
Value in hex	xxxx	4E's	00's	A1A1A1FB	xx's	xxxx	4E's

ID-00681

Figure 2-15 Data General standard 9 sector, 512 bytes/sector format

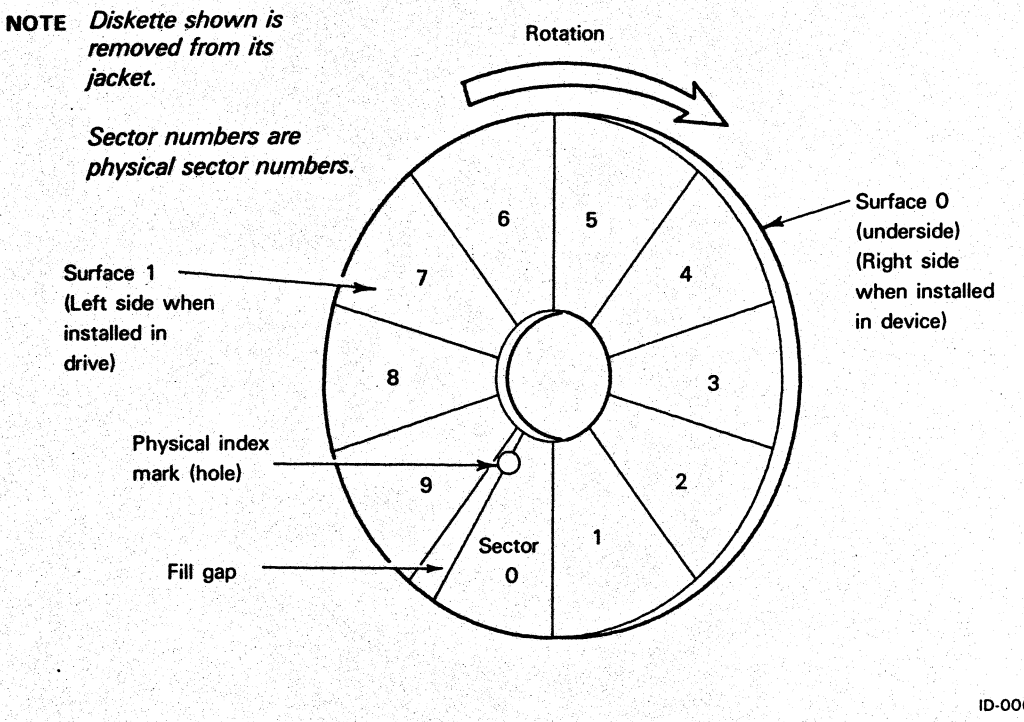
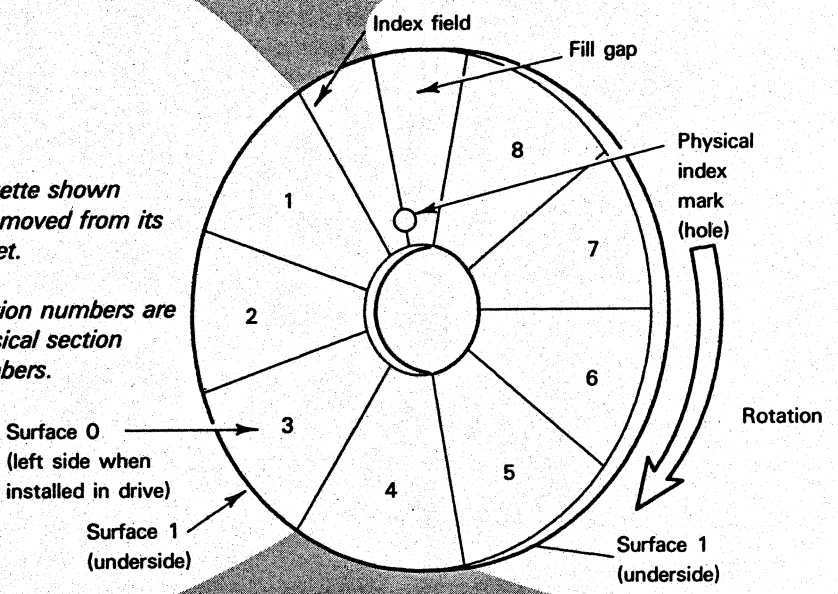


Figure 2-16 Data General MPT/100 10 sector, 512 bytes/sector format

Field	Gap 4A	Sync zone	Index mark	Gap 1
No. of bytes	80	12	4	50
Value in hex	4E's	00's	C2C2C2FC	4E's



NOTE *Diskette shown is removed from its jacket.*

Section numbers are physical section numbers.

Field	Sync zone	Address mark	Cylinder number	Surface number	Sector number	Sector length code
No. of bytes	12	4	1	1	1	1
Value in hex	00's	A1A1A1FE	xx	xx	xx	02

Field	Address field CRC	Gap 2	Sync zone	Data mark	Data field	Data field CRC	Gap 3
No. of bytes	2	22	12	4	512	2	80
Value in hex	xxxx	4E's	00's	A1A1A1FB	xx's	xxxx	4E's

ID-00683

Figure 2-17 IBM PC 8 sector, 512 bytes/sector format

The diskette format delineates an address field and specific data field length in each sector of every data track on the recording surface. The address field of a sector is a coded header that precedes the data block. On a formatted diskette, the address field is recorded at a specific location within the sector, giving the read and write control circuits enough time to initialize and settle before the field is read. Formatted diskette surfaces are necessary for the proper operation of the diskette subsystem.

Special address field marks are written to differentiate header fields from data fields. These headers and their address marks are recorded during the initial formatting and are never written again during normal operation. Formatting must be performed on one complete track at a time; a physically determined index pulse, transmitted from the drive, tells the interface when to begin and end the format operation for each track.

Data field marks specify the beginning of the data field. These marks are written during the initial formatting operation and are also recorded every time a data field is written during normal write operations.

Programming Procedure

A format operation transfers one track of formatting data between memory and the diskette drive. The number of formatting data words transferred depends on the word count specified by a previous *Load Word Count Register* instruction (DOC). Data is transferred through the data channel facility, starting at the memory location specified by a previous *Load Memory Address Register* instruction (DOB). Data transfers are performed on a demand basis — that is, data channel transfer occurs each time the diskette interface requires a 16-bit data word — and are double-word buffered to reduce the probability of data-late conditions.

Remember that a reformatting operation destroys all existing data on a diskette track. Never reformat a track without first backing up the data in its usable sectors.

Before you can format a track(s) on the diskette, you must set up a buffer in memory. The diskette format buffer must contain two words of data for each sector on the track that is to be formatted:

Cylinder number	Head number	word 0
Sector number	02 (= sector size	word 1
High byte	Low byte	

Note that the sector size is always equal to 2 for the standard 512 bytes per sector. If any other value is used, erroneous formatting will occur. The diskette controller on the SPU card will handle all the special address and index marks, the CRC code, and the other control fields, and will fill the data fields with 1s (each bite will contain FF hexadecimal). The above block is repeated 8 times in memory for IBM PC diskettes and 9 times for DG standard diskettes.
Use the following programming sequence to reformat a diskette track. Refer to Figure 2-18. Ensure that the diskette subsystem is ready to perform an operation (explained earlier under "Initiating an Operation").

1. Set up the required format buffer in memory.

2. Position the heads over the track to be formatted by issuing a *Specify Command and Diskette Address* instruction (DOA) with a Start flag command, using the appropriate accumulator to select a drive, and specify either a Recalibrate command (seek to track 00) or a Seek command and track address.

3. After the Recalibrate or Seek operation is completed, issue a *Read Status* instruction (DIA) with a Pulse flag command and check the Seek Error and Operation Timeout flags. (The Pulse flag command clears the Done flag and Interrupt request.) If a recoverable error occurred, try the operation again. Before proceeding, you may want to ensure that the seek operation positioned the heads at the proper track.

4. Issue a *Specify Command and Diskette Address* instruction (DOA) with no flag command, using the appropriate accumulator bits to select a drive and to specify a Format command and head number.

5. Issue a *Load Memory Address Register* instruction (DOB) with no flag command, using the appropriate accumulator bits to specify the starting address of the buffer in memory that contains the format data block.

6. Issue a *Load Word Count Register* instruction (DOC) with a Start flag command, using the appropriate accumulator bits to specify the number (in two's complement) of 16-bit words to be transferred. The Start flag command sets the interface's Busy flag to 1, sets the Done flag and Interrupt request to 0, and initiates the Format operation.

7. After the Format operation is completed on the selected drive, issue a *Read Status* instruction (DIA) with a Pulse flag command and check the error flags. (The Pulse flag command clears the Done flag and Interrupt request.) If a Data Late error occurred, try the data transfer again.

8. If surface 0 was just formatted and surface 1 is to be formatted next, update the head number byte in each sector's address field of the format buffer in memory. Then return to Step 4, this time selecting head 1 and repeat all steps up to this point. Otherwise, proceed to step 9.

9. If another track is to be formatted, update the track and head number bytes in each sector's address field of the format buffer in memory. Then return to Step 2 and repeat all steps up to this point.

After formatting a diskette for the first time, you may want to recalibrate the drive's positioner and step through each track, reading the sectors to ensure that the track, sector, and head numbers were correctly recorded in each address field.

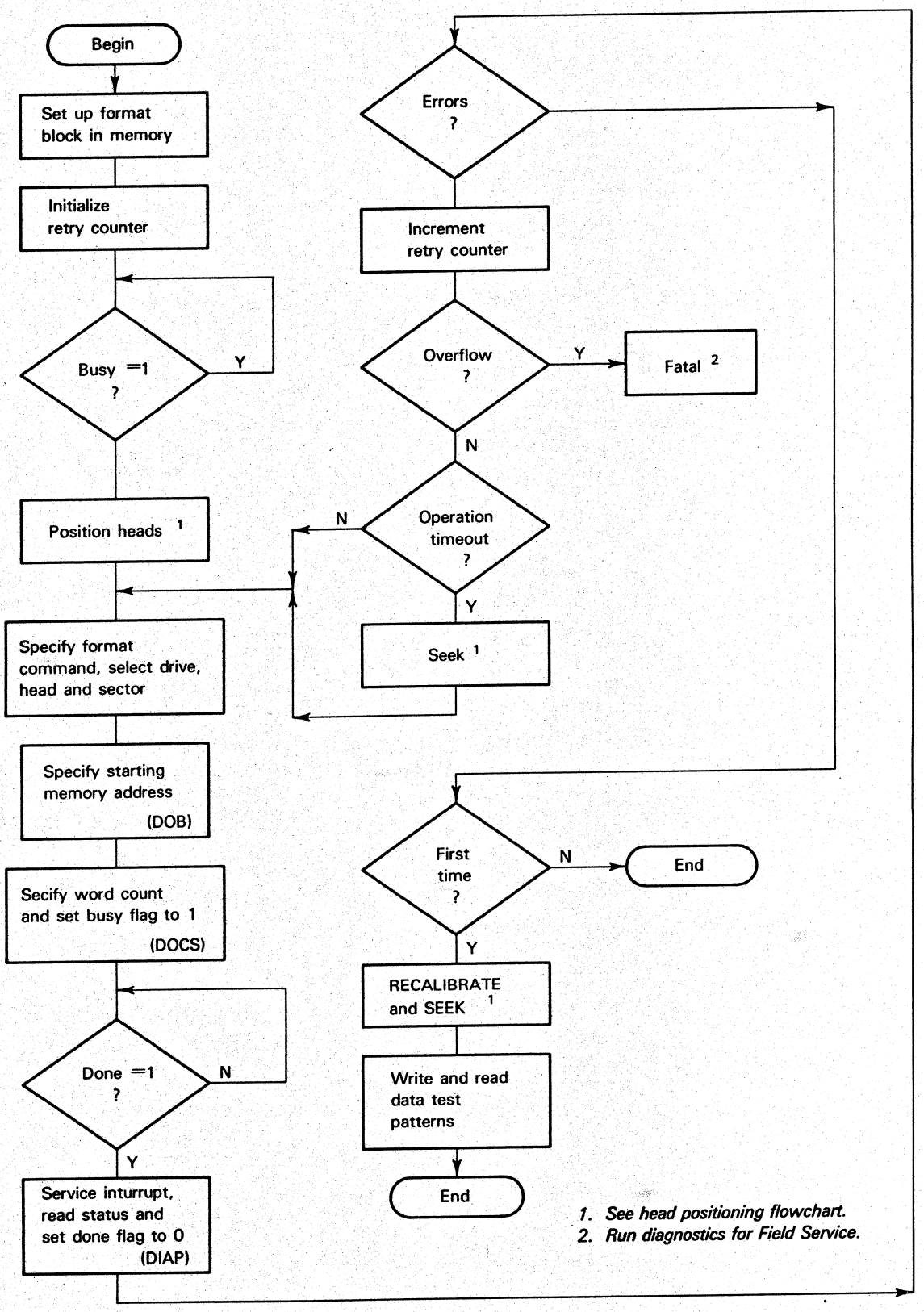


Figure 2-18 Format flowchart

I/O Timing

Several factors determine the time necessary to access and transfer data blocks to or from the diskette drive. These factors include:

Recalibrate or seek time

Head load time

Motor on time

Read or write time

Recalibrate or Seek Time

Read/write heads must be positioned over the proper track before a data transfer can begin. The following factors determine the time necessary to seek a specified track or recalibrate the positioner to select track 00:

The time the SPU takes to issue the Seek command and then process the interrupt when the operation is completed (referred to as SPU overhead time).

The time the controller takes to initiate the Seek operation and issue a program interrupt request (referred to as controller's overhead time and lasting approximately 1 ms).

The time the positioner takes to move the heads to the specified track (6 ms for a minimum seek, 78 ms for an average seek, and 234 ms for a maximum seek).

The time the positioner takes to settle the heads when the destination track is reached (15 milliseconds).

Thus, the total time for a positioning operation is 16 ms plus (6 ms times the number of tracks moved). A recalibrate operation, for example, could take up to 250 ms.

Head Load Time Read/write heads must be loaded onto the surfaces of the diskette before the interface can execute any read or write operation. The heads are loaded when the drive is selected. (After loading, a 50 ms timer delays any use of the heads for reading or writing, allowing them to settle.

Motor On Time The spindle motors of the diskette drive must be energized and up to speed before the interface can execute any commands except the Get Sectors command. The spindle motors deenergize if the interface receives no command within approximately 5 seconds. If the spindle motors are not energized when the interface receives a command, a one-second (maximum) delay allows the spindles to come up to speed before the interface executes the command.

Read or Write Time

The time necessary to read or write information on the diskette depends on its rotational position when the transfer is initiated, as well as the number of sectors to be transferred. Read/write time is determined by the amount of time needed for:

- The SPU to issue the Read/Write command along with a starting memory address and word count, and then process the interrupt when the operation is completed (referred to as the SPU overhead time).
- The interface to initiate the Read/Write operation, data channel operation, and Program Interrupt request (referred to as the interface overhead time, approximately 1 ms).
- The diskette to rotate to the selected sector once the interface initiates the Read/Write operation (variable, 100 ms average at nominal 300 RPM). Note that the interface allows four revolutions before it flags an address error; thus, it could actually take up to 1 second to flag the error.
- The subsystem to transfer the first sector (17.664 ms).
- The subsystem to transfer an additional sector after the first one (each additional transfer takes 20.928 ms).

The minimum total time to transfer a single sector is therefore approximately 18.6 ms excluding the SPU overhead time. For multiple-sector transfers, add 20.9 ms times the number of additional sectors.

The factors that determine the minimum total time for a Format operation are the same as those for a Write operation. Remember that a Format operation writes an entire track from the physical index point.

Error Conditions

This discussion defines the diskette subsystem's error conditions, which are reported by the status register. The error flags specifying the conditions are not valid until an operation has been completed and the Done flag is set to 1.

When an error condition occurs and a visual inspection of the selected diskette drive shows nothing unusual, try duplicating the fault with the same drive. If the fault persists, attempt to duplicate it on the other diskette drive, if present. ("Soft errors" due to airborne particles, random electrical noise, and other external causes are not the fault of the drive.) Diskette subsystem errors divide into power-up and self-test errors, initial selection errors, head positioning errors, and read/write errors. Each type of error is defined below.

Power-Up and Self-Test Errors The Not OK flag sets to 1 if the interface fails its self-test after a power-up sequence or a self-test diagnostic command. This bit resets only when the interface passes its self-test diagnostic.

Initial Selection Errors The Not Ready flag indicates the selected diskette drive did not complete a command. This may be because the drive is not up to speed, no diskette is inserted in it, or the diskette is improperly inserted in the drive. This error can occur at any time, and should therefore be checked after every diskette command.

If the flag asserts to 1 during a Read, Read Header, Write, or Format command, the operation terminates with the Not Ready flag set in the diskette status.

If the Not Ready status flag is 1 when the diskette drive is accessed, check the drive to ensure that (1) the drive is powered up, (2) a diskette is properly installed in the drive, (3) the door is closed, (4) the diskette is rotating, and (5) the drive unit's select jumpers are properly installed.

Head Positioning Errors Seek, Address, and Operation Time-out errors can occur during any head positioning operation and should therefore be checked following the completion of the positioning operation.

The Seek Error flag sets to 1 and the operation terminates, if (1) the Track 00 signal fails to assert during either a Recalibrate or a Seek operation in which the diskette interface imbedded a Recalibrate operation, and the interface cannot determine the current positioner location; or (2) a Seek command is issued with a track address greater than the maximum. To recover head position, reissue the recalibration command and, if successful, continue with the normal operational command sequence.

The Address Error flag sets to 1 if a Recalibrate command, with the Set Mode operation specified, attempts to set the forty/ninety-six operation mode bit to 1 on a 48 track-per-inch diskette drive. The Recalibrate operation completes.

The Operation Time-Out Flag sets to 1 and the operation terminates if the specified operation failed to complete within a reasonable time (approximately 3 seconds).

Read/Write Errors The following errors can occur during any data transfer operation, including a format Track operation, and should therefore be checked following the completion of the data transfer operation.

The Address Error flag sets to 1 and the operation terminates if:

A Read/Write operation was issued with a head or sector address greater than the maximum.

The interface is unable to find the desired sector — that is, it fails to read an address field that compares with the track, head number, and sector number that was specified with the operation's command. This may be due to a Seek fault.

The checkword appended to the address field did not compare with the checkword calculated by the interface while reading the address field. This error also sets the Checkword Error flag to 1.

A multisector read or write operation attempted to transfer a sector past the end of the current cylinder (last sector of track while head 1 is selected).

A Read command finds a deleted data mark recorded in the data field of a sector being read. This condition also sets the Bad Sector flag to 1. It occurs only when reading diskettes written on non-Data General systems.

The Checkword Error flag sets to 1 and the operation terminates immediately if (1) the checkword appended to the address field did not compare with the checkword calculated by the interface while reading the address field (this error also sets the Address Error to 1); or (2) the checkword appended to the data field did not compare with the checkword calculated by the interface while reading the data field during a Read command.

The Data Late flag sets to 1 if the processor's data channel facility fails to respond in time to a data channel request. This means that the interface's data buffer overflowed during a read operation or underflowed during a write operation.) The operation terminates at the end of the current sector, but data is lost.

The Operation Time-Out flag sets to 1 and the operation terminates if the specified operation failed to complete within a reasonable time (approximately 3 seconds).

The Bad Sector flag sets to 1 and the operation terminates if a Read command finds a deleted data mark recorded in the data field of a sector being read. This condition also sets the Address Error flag to 1. It occurs only when reading diskettes written on non-Data General systems.

Power-Up Response and Initial Program Load

Power-Up State

When power is applied to the diskette interface, its memory address and word count registers are cleared; its Busy, Done, and Interrupt Disable flags are set to 0, its Initiate Program Load (IPL) flag is set to 1, and its following status flags are set to 0:

Write protect

Track 00

Seek error

Data late

Address error

Checkword error

Operation time-out

Bad sector

The operating modes for the diskette drives are set as follows:

Track density	48 tracks-per-inch
Diskette surfaces	2
Number of sectors/track	9 (physical 1-9)
Number of words/sector	256 (512 bytes)
Sector addressing	logical (sectors 0-8)
Head position	1 = label side (right side when installed in drive) 0 = opposite side (left side when installed in drive)
Byte packing	high byte first, low byte second

NOTE *The drives are not recalibrated when the subsystem is powered up. Instead, a Recalibrate or Seek command* must *be issued prior to any Read or Write commands. If a Seek instead of a Recalibrate command is issued, the interface automatically performs a recalibrate operation before the seek operation.*

Initial Program Load (IPL)

The Initial Program Load (IPL) feature transfers a single sector, low-level, bootstrap program from diskette to memory. The transfer originates from drive number 0, track 0, head 0, and logical sector 0 (physical sector 1). The IPL sequence transfers the contents of this sector into the first 256 word locations of the memory. The bootstrap program must have been previously recorded on the designated sector of the diskette. If the drive is not ready, the IPL operation waits until it is.

An IPL operation always occurs when the system is first powered up. CPU firmware executes a small loader program that initializes the diskette subsystem and issues an *I/O Reset* instruction (IORST) followed by a Start command. Then the program branches to memory location 377_8 and waits (by looping on a Jump to 377_8 instruction placed in this memory location by the loader program). An operator can initiate an IPL at any time — provided the diskette drive is running and the diskette inserted in drive 0 contains the bootstrap program — by entering virtual console mode and issuing a program load command. (Refer to "Program Load Commands" in Chapter 3.)

The IORST instruction initializes the interface logic, sets the IPL flag to 1, and clears the memory address register. The Start flag command sets the Busy flag to 1, sets the Done flag to 0, and starts the IPL operation. The interface firmware initiates a recalibrate operation on the drive (positions the drive head(s) over track 00) and then starts a read operation, transferring sector 0 of head 0 to memory through the DMA facility of SPU1. When the transfer is completed, the Busy flag is set to 0, the Done flag is set to 1, and a Program Interrupt request is initiated.

The last (256th) word in the bootstrap program contained in the IPL sector on the diskette should contain a Jump instruction to a location in the bootstrap program. When transferred into memory location 377_8, this word will force the CPU to execute the low-level bootstrap program. First the bootstrap program should terminate the diskette Read operation. Next the bootstrap program transfers information from other sectors on the diskette to load the operating system and bring the system on line. The IPL flag is cleared the first time the CPU issues a Specify Command and Diskette Address (DOA) command.

Virtual Console 3

This chapter describes how virtual console is entered; defines virtual console cells; discusses the console commands and their formats, and how typographical errors are corrected.

The virtual console is a program that can aid you in working with the Desktop Generation Model 10 and 10/SP computer system. It allows you to interact with the computer through the terminal that is connected to the system processor unit's (SPU) asynchronous communications port. You enter simple commands on the terminal keyboard to examine or modify any processor register or memory location. A breakpoint feature allows you to stop the execution of a program at selected places for debugging.

NOTE *I/O interrupts are disabled and there is no I/O protection enabled when the virtual console is executing.*

The virtual console resides in read-only memory (ROM) chips on the system processing unit card. The virtual console has access to 2 kilobytes of static read/write memory (RAM), also on the SPU card, for use as a scratchpad. Neither virtual console ROM nor scratchpad RAM is part of the normal address space, so these are transparent to the user.

Upon power up, the virtual console firmware first performs a short self-test routine; then, if the test completes successfully, the virtual console program checks to see if there is a disk unit in the Desktop Generation Model 10 system. If there is, the virtual console displays a message asking the user which device should be used to program load — the disk or the diskette? In response to the user's answer, the virtual console program performs a program load.

If there is no hard disk in the Desktop Generation Model 10 system, the virtual console program will automatically attempt to program load from diskette unit 0. If the program load is unsuccessful, an error message will be displayed.

In addition to power up, the virtual console is entered under the following conditions.

A HALT instruction is executed (if Halt Dispatch has been enabled).

The user presses the Command and Break Keys of the system console.

The program completes execution of an instruction in the one-step mode (see the explanation of single stepping under "Function Commands").

A breakpoint is encountered (see the explanation of breakpoints under "Function Commands").

Once called, the virtual console displays a ! on the terminal. This is the virtual console *prompt*; it tells you that the console is ready to accept a command. The prompt is preceded by a single character that indicates the current state of the memory allocation and protection (MAP) unit:

! The MAP is currently off and no address translation will occur.

A! The MAP is on and user A has been selected.

B! The MAP is on and user B has been selected.

C! The MAP is on and user C has been selected.

D! The MAP is on and user D has been selected.

When a particular user has been selected, the current state of that user's map will be used in all address translations. You can change the MAP or MAP status from within the virtual console. The commands for doing so are presented in the "Additional Commands" subsection of "Function Commands."

Cells

Several virtual console instructions operate on *cells*. A cell is either a memory location (*memory cell*) or an internal register (*internal cell*) such as an accumulator. Each internal register accessible by the virtual console is assigned an internal cell number. Table 3-1 lists these registers and their numbers.

Table 3-1 Internal cells

Number	Cell
0-3	Accumulators AC0 through AC3, respectively.
4	The address of the break instruction at which the program halted, if the virtual console was entered on encountering a break instruction; or the contents of the program counter, if the virtual console was entered in any other way.[1]
5	Carry bit: bit 15 is equal to 0 when carry equals 0; equal to 1 when carry equals 1.
6	CPU (interrupt and NMI) status.[2]
7	System console status word.[3]
10	Virtual console register.
11	MAP status register.[3]

[1] *The virtual console sets bit 0 of this word to 1 when it takes control, no matter what its original value. Bit 0 in cell 4 must be 1 when the user program is recommenced with a* P *command.*

[2] *Refer to the* 16-bit Real-Time ECLIPSE Assembly Language Programming *(DGC No. 014-000688) for the contents of this register.*

[3] *Refer to the programmer's reference listed above for the contents of these registers.*

In order to examine or modify any cell, you must *open* it. Opening a cell causes its address and contents to be printed, in octal, at your terminal.

The *virtual console register,* which corresponds to internal cell number 10 in Table 3-1 can be accessed in user mode with a READS instruction. This cell always contains the device code of the last device from which a program load was effected.

Formats

A virtual console command consists of a single character. Some commands must be preceded by an *argument* which is an octal number. To form a valid number, you may use:

Digits. These must be in the range from 0 through 7. (If the argument is an address, it must be in the range from 0 through 77777.)

Period. The period (.) replaces the value of the last address used.

Signs. + *or* − A + or − sign may be entered after any valid number and must be followed by a valid number. The virtual console program will compute the arithmetic result and enter it in place of the original expression.

Delete or *Rubout.* The Delete Key may be used to delete any single digit. The virtual console displays an underscore character (_) to indicate that the preceding character has been deleted. The Delete Key will *not* delete the +, −, or . symbols. If it is used after a + or a −, it has no effect. If it is used after a

period, it deletes the right-most digit of the last address. The virtual console only retains six digits at any time; therefore, the Delete Key will not resolve all errors.

Examples showing resolution of expressions, when the last address entered was 100 are:

100000_1 will be replaced by 100001.

1000000 will be replaced by 000000. (*The virtual console only retains 16 binary bits.*)

.−3 will be replaced by 75.

.7 will be replaced by 1007.

0−7 will be replaced by 177771.

6+.−3 will be replaced by 103.
(*6+100−3=103.*)

75+_5 will be replaced by 102. (*75+5=102. The + is not deleted.*)

60+._ will be replaced by 70. (*60+10=70. The_ erased the right-most 0 of the last address, 100.*)

Cell Commands

To open a cell, use one of the commands listed in Table 3-2.

Table 3-2 Virtual console cell commands

Command	Function
*n*A	Opens the internal cell specified by *n*.
expr/	Opens the memory location specified by octal number *expr*.
Carriage Return	Closes the current cell and opens the next consecutive cell.
New Line*	Closes the current cell but does not open another.
/	Closes the current cell and opens the memory cell whose address is equal to the contents of the current memory or internal cell; after a New Line, opens location 0.

** Line Feed on non-ANSI standard keyboards.*

In Table 3-2 the term *current cell* refers to the last cell that you opened; the symbol *expr* means that you may type any valid octal number or expression, as explained earlier under "Formats."

When you open a memory cell, the virtual console interprets the address according to the current setting of the user MAP. That is, the number you enter is interpreted as a 15-bit address, then translated into a physical address in accordance with the current state of the MAP. You do not have to type leading zeroes. If, for example, you want to open logical memory location 5, you would type the number 5 followed by a slash (/).

Once you have opened a cell, you may change its contents by typing the octal number or expression whose value is to be placed in the cell. Terminate the expression with a Carriage Return, Line Feed, or New Line. Note that if you press Carriage Return, the next cell will also be opened. This is convenient when you need to enter data into several consecutive locations.

NOTE *If you open a cell and immediately type H or L, the contents of the cell are used as the value of* expr *for that command.*

If you type an expression starting with a + or −, the value of the expression will be added to or subtracted from the current contents of the cell. This result was illustrated under "Formats", and is shown again in the examples that follow. These examples demonstrate the use of /, New Line, and Carriage Return.

```
A! 3A 000003A 000100<CR>  AC3 contains 100.
000004A 000704/000704  024132<NL>
                                  PC contained 704.
                                  Location 704 contains 24132.
A! 5A 000005A 000000 1<NL>
                                  User changed carry bit to 1.
A! 100/ 000100 025037.<NL>  Contents of location 100 (25037) are changed
                            to current address.
A! 100/ 000100 000100<CR>  Above step confirmed.
000101/000101 000503 +1<NL>
                                  Contents of 101 incremented.
A! ./000101 000504
                                  Above step confirmed.
```

NOTE *Internal cell 11 is the last accessible internal cell. Do not use a Carriage Return to close it.*

Function Commands

Table 3-3 lists the virtual console function commands. These commands are explained in detail in the subsections that follow.

Breakpoints and Program Control

The virtual console breakpoint facility allows you to place breakpoints at up to eight locations in your program. When the program encounters the breakpoint during execution, it will enter the virtual console so that you can examine or modify any cells. This can be a great aid in debugging a program, since you can stop your program at points where you think there is a problem and then resume execution with no loss of data.

Table 3-3 Virtual console function commands

Command	Function
expr **B**	Inserts a breakpoint at the memory location specified by octal number *expr.* (If no *expr* is entered, all breakpoints will be displayed along with their assigned numbers.)
n **D**	Deletes breakpoint number *n* where *n* is a number between 0 and 7. (If no *n* is specified, all breakpoints are deleted.)
n **H**	Performs a program load from the data channel device whose device code is *n*.
I	Executes an *I/O Reset* (**IORST**) instruction.
K	Cancels the entire line just typed and prints a question mark (?).
n **L**	Performs a program load from the programmed I/O device whose device code is *n*.
O	Steps through one instruction of the user's program.
P	Starts program execution at the memory location specified by the contents of internal cell number 4. (See Table on "Internal Cells" in this chapter.)
expr **R**	Starts program execution at the memory location specified by octal number *expr.*
U	Changes the user map. Displays a colon (:), after which the user must enter **A, B, C,** or **D** to specify a map. If any other character is entered, the map is turned off.

Setting Breakpoints

To set a breakpoint, type *expr* B. The breakpoint will be set at the address specified by *expr* according to the *current user map.* Breakpoints can be used only in the user map from which the user program will be started.

The virtual console assigns numbers to breakpoints in reverse order — that is, breakpoint 7 is assigned first, then 6, and so on. The unassigned breakpoint with the highest number is always assigned first. For example, if numbers 7 and 5 are assigned, the next will be 6, not 4. To delete a breakpoint you must use the number assigned to it as described in the subsection to come. Typing B with no specified address will cause the virtual console to list all the current breakpoints along with their assigned numbers.

Examples:

A! 423B	Places a breakpoint at address 423 in user map A.
A! B	Requests list of all current breakpoints.
7 75324	Breakpoint 7 is at address 75324.
3423	Breakpoint 3 is at address 423.
A! 623B?	Requests a breakpoint at a valid location . . . But apparently all eight breakpoints are in use. User must delete a breakpoint before setting another.
A!	A prompt always follows a question mark (?).

NOTE *Do not place two breakpoints at the same location, and do not set breakpoints in addresses that are to be executed when the MAP is enabled and I/O protection and/or the Load Effective Address mode is enabled.*

Deleting Breakpoints

Use the D command to delete a breakpoint. Type *n*D to delete breakpoint *n*. This will delete the breakpoint regardless of the current state of the map. If no number *n* is specified, the D command will delete all breakpoints.

Examples of deleting breakpoints are:

A! 3D	Deletes breakpoint number 3.
A! 12D	Only eight breakpoints (numbers 0-7) are valid . . .
?	. . . Any other number is not allowed.

Encountering a Breakpoint

When a breakpoint is encountered during execution of a user program, the virtual console is entered and the address of the instruction at which the breakpoint was set is displayed and placed in internal cell number 4. The instruction at the location of the breakpoint is not yet executed. The virtual console then displays a prompt. The user can now inspect and modify any internal cell or memory location.

Single Stepping

Use the one-step command, O, to single step through a program. The O command, issued while the virtual console is in control, sets a flag that will cause a nonmaskable interrupt (NMI) to occur as soon as the first main (user) program instruction has executed. The virtual console then returns to the main program location specified by the contents of internal cell 4; the main program executes one instruction, and then the virtual console resumes control.

As the instruction is executed, the console will print the address of the following instruction (that is, the contents of the program counter at the time of execution). Pressing and holding down O with the Repeat key is a convenient way of quickly locating the occurrence of skips or branches. After each instruction has been executed, the virtual console resumes control and issues a prompt.

NOTES *The fact that a user breakpoint may have been set for an instruction will have no effect on the execution of the O command. The XCT instruction cannot be single-stepped.*

Resuming Program Execution

The virtual console has two commands that allow you to resume program execution after the console has been entered through a breakpoint, after single-stepping, or by some other means. Typing P restarts program execution at the location specified by the contents of internal cell number 4, which is always the return address (see Table 3-1).

NOTE *When the virtual console is entered through a breakpoint, internal cell 4 contains the address of the location of the breakpoint. This should be the next instruction to be executed in order to resume normal program flow. When the virtual console is entered any other way, internal cell 4 contains the value of the PC + 1; this should also be the next instruction executed in order to resume normal flow. In either case, the P instruction will produce the required result.*

You can also return to a program by typing *expr*R. In this case, program execution resumes at the location specified by *expr*. When the R command is issued, the virtual console inserts all previously specified breakpoints, clears nonmaskable interrupts (NMIs), and resumes program execution at the logical address <*expr*> in the current map. The R command does not cause an I/O or system reset. The number specified by *expr* must be a valid address in user memory. If this argument is not in the user range, or is not supplied, the virtual console will do nothing.

Additional Commands

This section presents the U command for changing user maps, the 11A command for changing the MAP status and the L or H command for causing a program load.

Changing the MAP or MAP Status

To change user maps, use the U command. *When you issue this command, the console will immediately print a colon (:). Now you must type a single character — A, B, C, or D — to change the map to that user. If you type any* other character (including Carriage Return or New Line), the MAP will be turned off. After you type a character, a prompt is displayed that reflects the new state of the MAP.

You can change the MAP status from within the virtual console. To do this, open internal cell 11, the MAP status register, by issuing an 11A command. Type in the new status, then close the cell with a New Line. *Now issue a* U *command, whether or not you want to change maps.* (If you do not want to change maps, simply enter the character for the current map.) The new map status takes effect only after a U command has been issued.

Program Load Commands

Typing *n*L causes the CPU to perform a program load from a programmed I/O device whose device code is equal to the last two digits of the octal number *n*.

Typing *n*H causes the CPU to perform a program load from a data channel device whose device code is equal to the last two digits of *n*. The device code range specified for the *n*L command also applies here. Once a high-speed program load from a data channel device has begun, the virtual console is no longer in control.

After a program load has been performed, the octal number *n* is placed in the virtual console switch register. A READS instruction can then be used to return the 16-bit number entered by the operator with the H or L command.

NOTE *After an automatic program load following initial power up, the READS instruction will return the device code of the boot device in bits 10-15, bits 1-9 will be set to zero, and the state of bit 0 will be undefined. After any program load of whatever type, the state of the accumulators will be indeterminate.*

The I Command

Typing an I on the system console while in the virtual console causes an *I/O Reset* instruction to be executed immediately. All I/O device controllers' flags are cleared as a result (Busy = Done = 0). Refer to the programmer's references for the Model 10 and 10/SP systems and for the device controllers for information on the effects of the *I/O Reset* instruction.

Correcting Errors

This final section explains the use of the Rubout Key and the K command to correct typographical errors. It concludes by describing the conditions under which a virtual console error can occur.

The Rubout Key

You can use the Rubout Key to delete the last character you typed, in which case the virtual console echoes the rubout with an underscore (_). Typing more Rubouts will continue to delete digits from right to left.

If you type any rubouts immediately after opening a cell, the virtual console will delete the right-most digits of the cell's contents as though you had just typed them yourself. You may then type in new values for these digits. Refer to the "Formats" section presented earlier for more information on the Rubout Key.

The K Command

If you wish to cancel an entire line that you have just entered, type a K. In response, the virtual console prints a ? followed by a New Line, and also closes the current cell if it is open. The ? followed by a New Line is also printed if you type a character that the virtual console does not recognize.

Virtual Console Errors

If you attempt to open a nonexistent memory cell, the data displayed as its contents will be meaningless. You can test whether a location exists by entering a new value in the memory cell and then reopening it. If it does not contain the value just entered, the location is nonexistent.

In addition to the case in which an undefined character is typed, the virtual console will type a ? followed by a New Line under the following conditions.

A command to open a nonexistent, internal cell is issued.

An R command is issued without an argument.

A set breakpoint command specifies an invalid address.

A delete breakpoint command specifies a number greater than 7.

A set breakpoint command is issued after all eight breakpoints have been assigned.

A program load command is issued with an illegal device code.

In all of these cases, the command involved will not be executed. In fact, the virtual console will do nothing, and any data just entered will be discarded.

Part

TWO

Theory of Operation

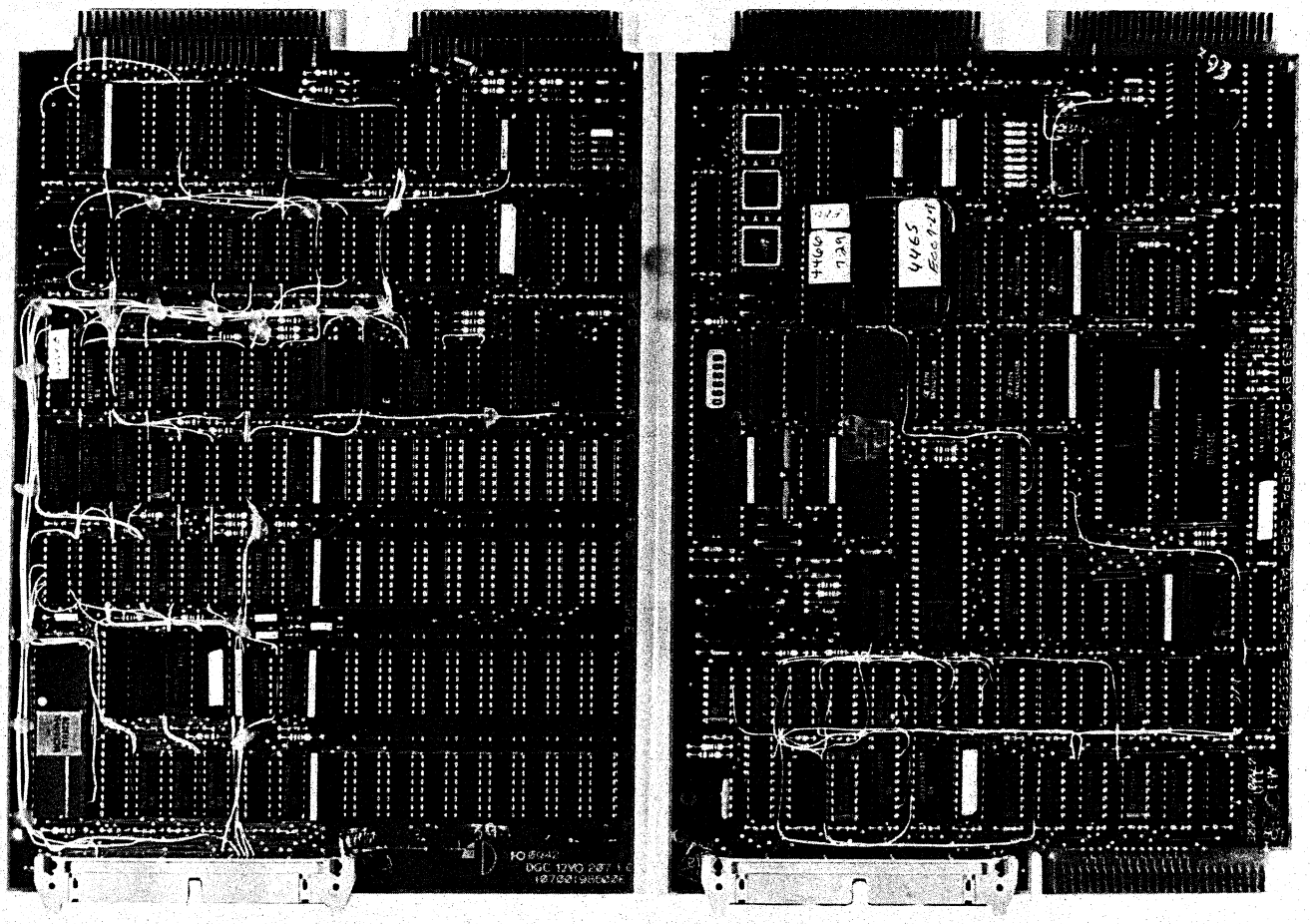


Figure 4-1 Models 10 or 10/SP 2-card SPU

System Processing Unit

4

The Model 10 and 10/SP system processing unit (SPU) is a set of two printed circuit boards that perform the core functions of the computer system. The cards are shown in Figure 4-1.

The SPU consists of the following elements:

- Two 16-bit central processing units (CPUs): a Data General microECLIPSE CPU integrated circuit with one or three external microcode controller chips (XMCs) and an Intel 8086 microprocessor IC, referred to as the attached processor, or ATP.
- A multidevice section based on a microECLIPSE system I/O (SIO) IC, containing a full duplex asynchronous communications interface (called the printer port), a real-time clock, a programmable interval timer, the interface to the microI/O bus, a powerfail monitor, and a SPU status register;
- Bus arbitration and DMA and I/O control logic;
- 16 kilobytes of read-only memory (ROM) that contain a self-test program, a virtual console program, and instruction emulation code (this ROM is supplemented by an additional 6 kilobytes of reserved RAM);
- Two memory allocation and protection units (contained in one physical memory array) — one for each CPU, along with selection logic to route the addresses to the memory;
- 128 or 256 kilobytes of dynamic random-access memory (RAM) and associated parity checking logic;
- A diskette interface that supports one or two minidiskette drives and provides direct memory access (DMA) for diskette data transfers;
- Video and keyboard support logic, including a firmware graphics/ alphanumerics video generator for the standard monochrome monitor of the system console; and
- Decoding logic for memory-mapped I/O control and system timing signal generators.

The SPU boards receive their power and communicate with other printed circuit boards (optional memory, optional color video interface or I/O controllers) in the logic module using their B connectors, which plug into the backpanel printed circuit board of the logic module. They communicate with each other over a short cable connected to their D connectors. The first SPU board, called SPU1, communicates with the printer port over a cable attached to its A connector and with the diskette drive unit over a cable attached to its C connector. The second SPU board, called SPU2, communicates with the system console keyboard and monochrome monitor over a cable attached to its A connector. If the system contains an optional color monitor interface, and if the color monitor is used as the system console, then a cable connects the SPU2 A connector to the color monitor interface, and a cable on the A connector on that interface connects it to the system console keyboard and color monitor.

The next sections present material relative to Data General logic schematics No. 001-003677 and 001-003678. All signals in the figures or text of these sections are listed in Table 4-1 or Table 4-2 at the end of this chapter.

Shown in Figure 4-2, the Model 10 or 10/SP 2-board SPU is organized around two buses, the logical address bus and the system bus. The logical address bus is carried on the foreplane (via the D connectors) and transfers logical addresses generated by the microECLIPSE or the 8086 processor to the MAP RAM and to memory-mapped I/O devices: the diskette and video interfaces and the I/O decode logic.

The system bus is carried on the backplane (via the B connectors) and carries physical addresses and memory control signals to the memory from the MAP RAM and control logic and carries data between the memory and the CPU transceivers.

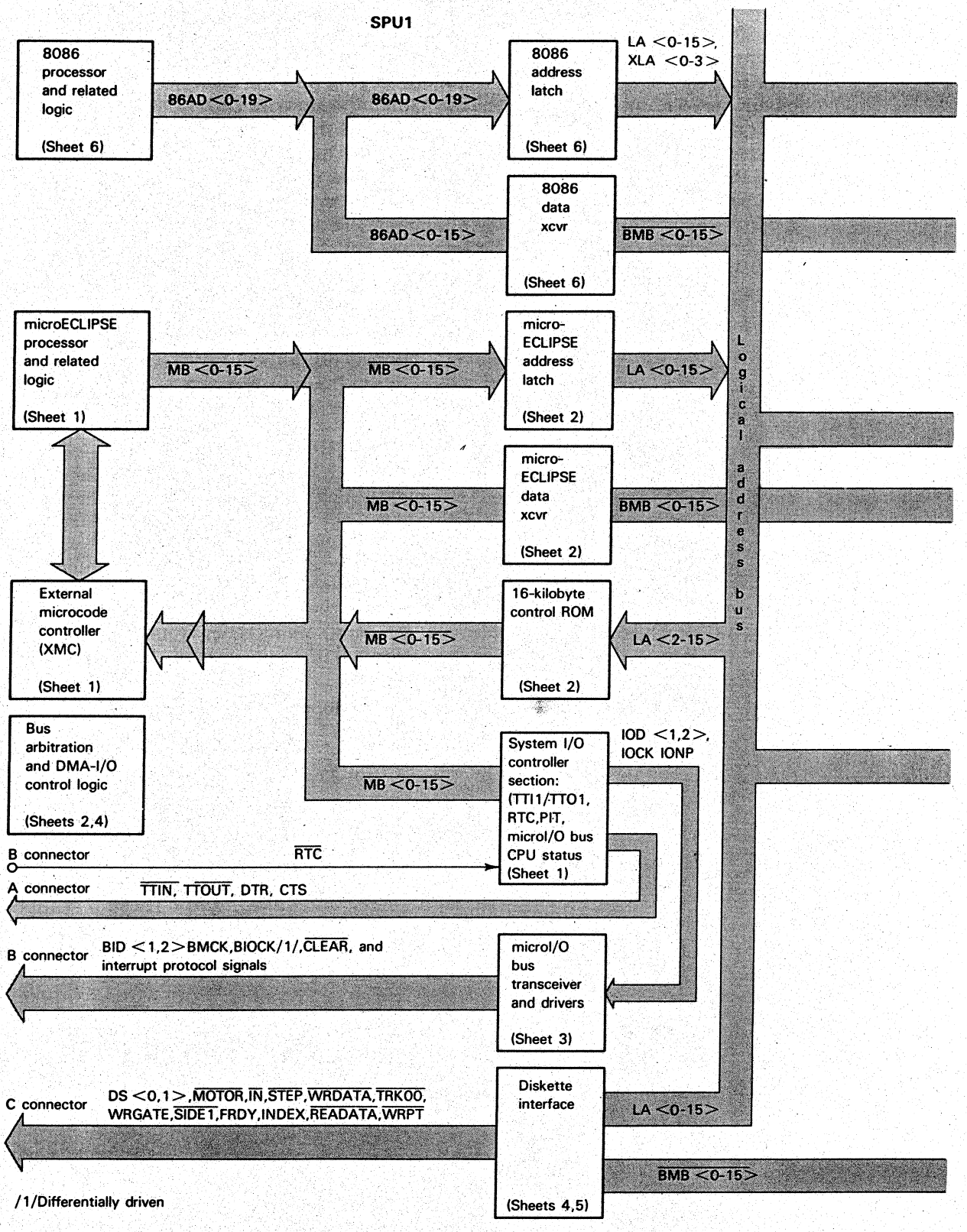


Figure 4-2 Organization or the 2-card SPU

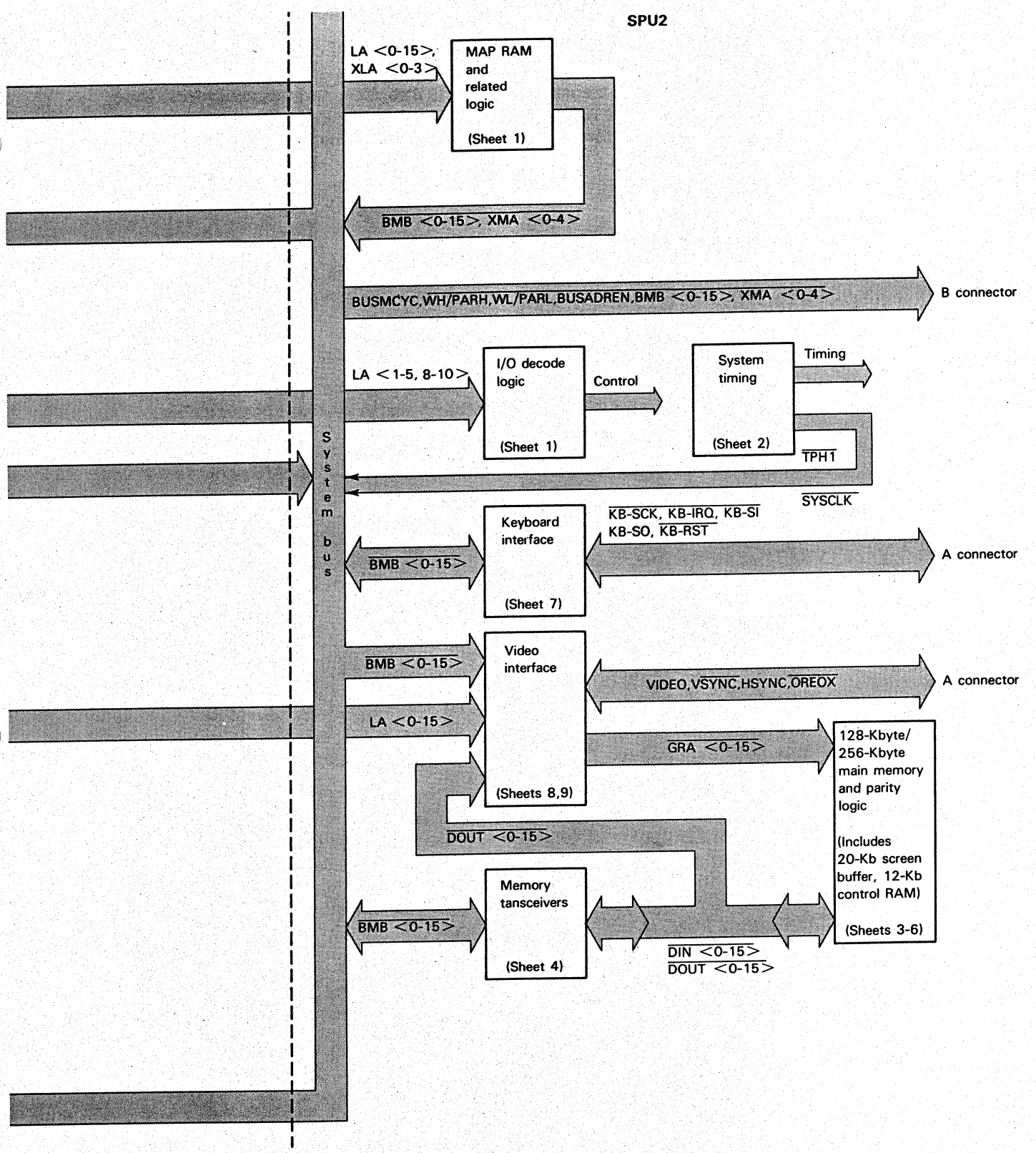

SPU2
LA <0-15>, XLA <0-3>
MAP RAM and related logic
(Sheet 1)
BMB <0-15>, XMA <0-4>
BUSMCYC,WH/PARH,WL/PARL,BUSADREN,BMB <0-15>, XMA <0-4>
B connector
LA <1-5, 8-10>
I/O decode logic
(Sheet 1)
Control
System timing
(Sheet 2)
Timing
TPH1
SYSCLK
System bus
Keyboard interface
(Sheet 7)
KB-SCK, KB-IRQ, KB-SI
KB-SO, KB-RST
BMB <0-15>
A connector
Video interface
(Sheets 8,9)
BMB <0-15>
VIDEO,VSYNC,HSYNC,OREOX
A connector
LA <0-15>
GRA <0-15>
128-Kbyte/ 256-Kbyte main memory and parity logic
(Includes 20-Kb screen buffer, 12-Kb control RAM)
(Sheets 3-6)
DOUT <0-15>
Memory tansceivers
(Sheet 4)
BMB <0-15>
DIN <0-15>
DOUT <0-15>
System bus

Figure 4-2 also shows the physical location of the SPU components on SPU1 and SPU2: the two CPUs and their transceivers, the SIO controller section, control memory, microI/O bus transceiver, diskette interface, and bus arbitration logic on SPU1; and the MAP RAM, onboard memory, console (keyboard and monochrome monitor) interface, and timing and decode logic on SPU2.

CPU Section

The CPU section consists of

two CPUs, microECLIPSE and 8086, each with its own address latch and data transceiver,

control read-only memory (ROM) that contains emulation code for some microECLIPSE instructions plus other CPU-support code,

bus arbitration and DMA and I/O control logic that allocates the system bus between the two CPUs and the other possible system bus controllers.

The microECLIPSE is supported by one or more external microcode controller ICs (XMCs) and by an external CPU status register that is partly contained within the SIO IC.

microECLIPSE CPU and XMCs

The microECLIPSE CPU, resident on SPU1, is a microprogrammed processor that incorporates the full Data General ECLIPSE 16-bit architecture, including four fixed point registers and a MAP status register and, in the Model 10/SP, four floating point accumulators and a floating point status register. Both processors execute the standard ECLIPSE instruction set and the ECLIPSE character instruction set, and the Model 10/SP executes the ECLIPSE floating point instruction set.

The microECLIPSE CPU handles all input/output operations for a Model 10 system and all the memory management. The attached 8086 processor does not perform any direct input/output. The microECLIPSE processor performs I/O for it in response to calls issued from 8086 code.

A kernal of the microinstructions for the ECLIPSE instruction set is executed by the microECLIPSE CPU IC, while most of the remaining instructions are executed by one or more external microcode controller (XMC) ICs. Each XMC communicates with the CPU via a dedicated 8-bit time-multiplexed bus. A few input/output instructions are trapped by code in an XMC and executed by code contained in the control ROM, as will be explained later.

Depending on the model, the microECLIPSE CPU is supported by one of two possible sets of XMCs. The Model 10 computer includes one XMC that contains microcode to help implement the standard ECLIPSE instruction set as well as the

character instruction set. The Model 10/SP computer includes a set of three XMCs that contain the additional microcode for the floating point instruction set.

The CPU uses a 16-bit-wide CPU address/data bus $\overline{MB<0\text{-}15>}$ to communicate with the rest of the system. It communicates directly with the SIO IC and the control memory via this 16-bit CPU bus. The addresses placed on the CPU bus are latched by the microECLIPSE address latch, which places them on the logical address bus LA<0-15>. The most significant bit on the address bus, LA0, is not an ordinary address bit, but is used to specify that the memory access in progress is a main (user) memory access rather than a control memory access. During memory mapped operations, the 15-bit logical addresses, LA<1-15>, are expanded to 20-bit addresses. The data from the CPU is passed to the system bus $\overline{BMB<0\text{-}15>}$ by the microECLIPSE data transceiver.

microECLIPSE CPU *apparent* memory access time is 0.500 microseconds. (Actual CPU memory accesses require only 0.250 microseconds, as will be explained later.) Instruction execution times for instructions contained in the microECLIPSE or XMC microcode range from 0.50 to 34.00 microseconds for fixed-point operations. Floating-point operations require from 1.00 to about 900 microseconds. Instructions that are executed by code in the control memory require from 212 microseconds to about 0.03 seconds.

The microECLIPSE CPU can address up to 64 kilobytes of memory directly and up to 320 kilobytes with a single address translation configuration (five maps set up, each mapping a 64-Kbyte space). It supports write protection, validity protection, indirect protection, and I/O protection. In addition the microECLIPSE CPU

supports direct memory access via data channel,

has two distinct program interrupt facilities,

supports 16 levels of programmed interrupt priorities.

The diskette interface contained on the Model 10 or 10/SP SPU board set provides DMA data transfers directly between main memory and one or two diskette drives at a rate of about 30 kilobytes per second. The diskette DMA controller supplies its own memory control signals and operates independently of either CPU, so that instruction execution continues during data transfers.

The microECLIPSE CPU's data channel facility allows devices connected to a Model 10 and 10/SP system via the microI/O bus to transfer data to and from main memory at speeds of 337 kilobytes per second for output and 267 kilobytes per second for input.

The microECLIPSE CPU has two interrupt facilities: standard and nonmaskable (NMI). The *standard interrupt facility* services interrupt requests based on the following order of priorities:

1. TTO (system console output)
2. TTI (system console input)
3. Diskette
4. ATP
5. Parity
6. Powerfail
7. Programmable interval timer

8. Real-time clock
9. TTI1 (printer port input)
10. TTO1 (printer port output)
11. External microI/O bus devices

The *nonmaskable interrupt* (NMI) facility responds to the following types of interrupts:

Power-up condition

Powerfail condition

Break key

Halt instruction

Parity error

Keyboard interrupt request

Diskette interrupt request

Attached processor (ATP) interrupt request

These interrupts are transparent to the user, except for the Break key and Halt interrupts, which pass control to the virtual console, and the ATP and parity interrupts, which require interrupt handler routines.

The 16 levels of programmed interrupt priority are associated with the standard interrupt facility. They are established by a software priority mask. These levels allow the program to establish interrupt priorities among I/O interfaces. A special vectored interrupt instruction updates the priority mask while saving return information and transferring control.

8086 CPU

The 8086 CPU, also resident on SPU1, appears as an I/O device to the microECLIPSE processor. Referred to as the attached processor, or ATP, it is a 16-bit processor that generates 20-bit byte addresses, 86AD<0-19>, of which 19 bits are used as word addresses. (The high/low-byte bit is 86AD0.) Sixteen of these address bits are time–multiplexed with the 16 data bus bits. The 19 address bits are latched by the 8086 address latch, which places them on the logical address bus, LA<1-15> and XLA<0-4>. The 19-bit logical address is converted by the 8086 MAP to a 20-bit physical address. The sixteen data bits are transferred to the system bus via the 8086 data transceiver.

The 8086 clock rate is 8 megahertz. Since an elementary 8086 instruction executes in 4 clock periods, this yields a minimum instruction execution time of 0.500 microseconds. The minimum 8086 memory access time is likewise 0.500 microseconds. Memory accesses of the 8086 are synchronized with those of the microECLIPSE, as will be explained later.

The 8086 processor pauses whenever an NMI or microECLIPSE *interrupt* occurs. (Interrupts cause the signal 86HOLD). The 8086 itself, however, responds only to one interrupt: that generated by a microECLIPSE DOB ATP instruction. The DOB loads the ATP interrupt vector register with an 8-bit number that is passed to the 8086 processor in response to its assertion of an interrupt acknowledge signal.

The 8086 processor runs or is idle in accordance with the above-mentioned 86HOLD signal generated by the SPU bus arbitration logic. Once running, the 8086 processor continues to run until it encounters an OUT instruction or until the assertion of 86HOLD.

Control Memory

Model 10 or 10/SP control memory consists of 16 kilobytes of read-only memory (ROM) and 12 kilobytes of dynamic random-access memory (RAM). The ROM contains

Emulation code for certain microECLIPSE instructions

Power-up self-test

Virtual console program

Bootstrap loader program

Bit maps for the D1 type terminal emulator program that is present before the D200/D211 emulator program is downloaded

The control RAM contains the downloaded D200/D211 console emulator and graphics code and the scatchpad memory required by the code in the control ROM.

The instructions emulated by code in the control ROM are

- All I/O instructions to the following devices:

  system console, TTI/TTO — device codes 10 and 11

  diskette interface, DEO, — device code 20

  attached (8086) processor, ATP — device code 6 (including *Load Attached Processor MAP,* LMPA)

  system parity logic, PAR — device code 2

- The following instructions to the CPU — device code 77:

  Interrupt Acknowledge, INTA

  Maskout, MSKO

  Vector, VCT

  I/O Reset, IORST

  CPU Status, DIS

- The following microECLIPSE MAP instructions:

  Load MAP Status, DOA MAP

  Initiate Page Check, DOC MAP

The power-up self-test in the control ROM performs a complete test of the functioning of all Model 10 or 10/SP components and prints out a message that confirms the successful completion of each subtest. A complete description of the self-test, its messages, and their meanings is included in *Testing Model 10 and 10/SP Systems.*

Included in the self-test code are routines that initialize various parts of the system, such as the diskette interface and the system console interface.

The functions of the virtual console program contained in the control ROM are described in Chapter 3 of this manual.

Bus Arbitration and DMA and I/O Control Logic

The primary component of the bus arbitration logic is a single programmed array logic (PAL) chip that receives requests from all the potential users of the system (BMB) bus and allocates its use in accordance with the following priority scheme:

Highest priority — the microECLIPSE CPU

Second highest priority — the 8086 CPU

Third priority level — the diskette direct-memory-access (DMA) controller and the system input/output (SIO) IC

The diskette and the SIO IC have the same level of priority.

The arbitration PAL produces signals that indicate to the controllers which may use the bus and which must wait until the bus is granted.

The DMA and I/O control logic, like the bus arbitration logic, is based on a single PAL that produces the control signals necessary for DMA transfers on the system bus and input/output operations for I/O devices resident on the 2-board SPU. Devices that use these control signals are the diskette controllers and the system console keyboard interface.

Multidevice/microI/O Bus Section

The heart of the multidevice/microI/O bus section is the I/O (SIO) integrated circuit chip. This chip contains a:

printer port, which is an asynchronous interface (TTI — device code 50, TTO — device code 51)

real-time clock (device code 14)

programmable interval timer (device code 43)

microI/O bus controller, which forms the interface between the system and devices whose I/O interfaces are not contained on the two-board SPU, such as a mini-Winchester disk or communications multiplexor.

powerfail monitor (device code 0)

half of the CPU status register (device code 77)

The *printer port* is an interface that can be connected to any RS-232C device operating at 9600 baud. It transmits and receives serial asynchronous information. The buffers, registers, and state machine logic for the interface are all contained on the SIO chip.

The terminal output section is equipped with a Clear To Send (CTS) input, which inhibits the shifting out of the serial data when the device is not asserting CTS. The CTS input is only monitored at the start of a TTO cycle — a cycle does not abort if the state of CTS is changed while it is in progress.

The terminal output section is also equipped with a Data Terminal Ready (DTR) signal, which remains asserted as long as the system is powered up.

The terminal input section contains logic that detects the reception of a Break sequence on the asynchronous line. The logic causes the Break key bit in the SPU status register to be set and the $\overline{\text{NMI}}$ signal to be asserted. The nonmaska-

ble interrupt causes the processor to enter the virtual console mode. For information on the Virtual Console, refer to Chapter 3.

The *real-time clock* generates low frequency I/O interrupts for performing time calculations independent of CPU timing. These interrupts can be used as a time base in programs that require one. The interrupt frequency of the clock is program-selectable to ac line frequency, 10 Hz, 100 Hz, or 1000 Hz.

When the CPU acknowledges an interrupt caused by the RTC, the SIO chip returns device code 14. The RTC rate is set according to bits 14 and 15 of the specified accumulator with a DOA RTC instruction.

After power-up or an *I/O Reset* instruction, the real-time clock frequency is automatically set to the ac line frequency.

The *programmable interval timer* (PIT) is a CPU-independent time base that can be programmed to initiate program interrupts at fixed intervals. These intervals range from 100 microseconds to 6.5536 seconds in increments of 100 microseconds. The clock rate of the PIT is 10 KHz. The contents of the PIT counter can also be sampled with I/O instructions at any point in its cycle to determine the time until the next interrupt. The PIT is often used in multiprogram operating systems, where it is used to allocate CPU time to different programs on a "time slice" basis.

The programmable interval timer generates an interrupt when its counter overflows. A DOA PIT instruction loads the counter with the two's complement of the desired count. When the counter overflows, the SIO chip asserts $\overline{\text{SIO INT}}$, but the counter is not stopped. When the CPU acknowledges the interrupt, the SIO chip returns device code 43. The PIT counter/register is double-buffered, so no counts are lost when the register is read.

The *microI/O bus controller* contains all the buffers and logic needed to generate the Data General microI/O bus. This bus is a 16-line communications bus that performs 2-bit serial transfers of data and I/O instructions or status between the microECLIPSE CPU and microI/O device controllers. The SIO chip does the conversion of 16-bit parallel data received from the microECLIPSE CPU into a pair of 8-bit serial signals. It also performs interrupt arbitration.

microI/O bus data transfers occur on two differentially driven bidirectional serial lines (BIO<1,2>) and are synchronized by a differentially driven clock signal (BIOCK). These lines are conditioned and synchronized by a Data General mN629 IOC transceiver.

Two interrupt lines — programmed I/O $\overline{\text{EXTINT}}$ and data channel $\overline{\text{EXT DCHR}}$ — allow I/O interfaces to request processor time. A system reset line ($\overline{\text{CLEAR}}$), two device priority lines (INTPOUT and DCHPOUT), a differentially driven master clock signal (BMCK), and three ground lines comprise the remainder of the bus.

The *powerfail monitor* in the SIO monitors the power status signal PF. The power supply asserts this signal when AC power is interrupted. The monitor generates a power-change interrupt under *one* of two circumstances:

PF goes from negative to positive, indicating that AC power has failed, or

PF goes from positive to negative *and* dc power has remained within specifications.

The second condition indicates that ac power has returned after a momentary interruption. The SIO IC actually detects this condition by noticing that no system reset occurred.

Under this condition, the SIO chip drives $\overline{\text{SIO INT}}$ low to cause a power-change interrupt and sets bit 0 in the SPU status register to 1. The CPU responds to this interrupt with a DIB CPU, interrupt acknowledge, which causes the SIO IC to return device code 0.

The power-change interrupt is cleared by a *CPU Acknowledge* (DOAP CPU) instruction, with bit 0 of the specified accumulator set to 1.

The power monitor generates a power-up, nonmaskable interrupt (NMI) when PF changes from positive to negative and dc power has just risen — that is, when the SIO has received a $\overline{\text{SIO RESET}}$ signal. In this case, the SIO chip sets bit 4 in the CPU status register to 1 and asserts $\overline{\text{NMI}}$. Bit 4 is cleared as just described by a DOAP CPU, with bit 4 of the specified accumulator set to 0.

The *CPU status register* reports on the status of the system by setting bits in the CPU status register as described in Chapter 1 of this manual.

The high-order 8 bits of the CPU status register are contained on the SIO chip, and the low-order 8 bits are contained in a separate buffer. A *Read CPU Status* instruction (DIS *ac*, CPU) places the contents of this register into the specified accumulator. Ten status bits report the occurrences of nonmaskable interrupts: 0, 3-5, 8, and 11-15. These bits can be cleared (set to 0) with a DOAP *ac* CPU instruction, with the corresponding bit in the specified accumulator set to 1. The remaining bits report the status information listed in Chapter 1. These bits cannot be cleared, although the Interrupt On bit becomes 0 when CPU interrupts are disabled.

NOTE *The virtual console clears those bits that cause it to be entered. They are not ordinarily manipulated by the user.*

MAPs and Master Multiplexor

Figure 4-3 shows the organization of the MAP RAM and its associated logic, including the master multiplexor. Note that the two MAPs — microECLIPSE and 8086 — are physically contained in a single array consisting of three 1K-by-4 static RAM ICs. The MAP multiplexor selects appropriate bits of the logical addresses produced by either the microECLIPSE CPU or the 8086 CPU, depending on which processor is running, on whether control memory code is being executed, and on the state of the MAP select bits in the MAP status register. The selected bits, LM<0-9>, address one of the 160 microECLIPSE map registers or one of the 512 8086 map registers.

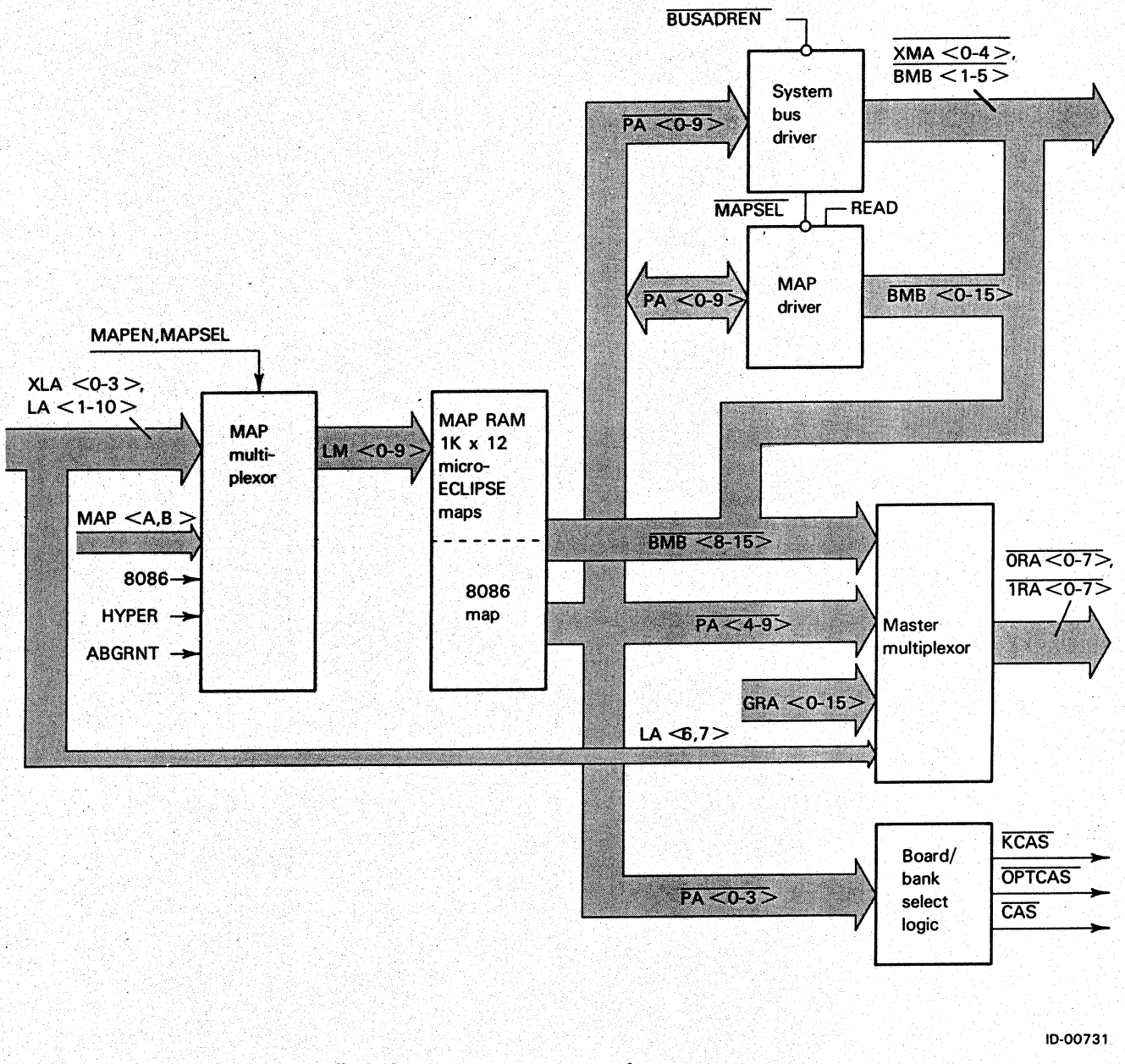


Figure 4-3 Memory allocation and protection section

The 12-bit contents of a MAP register constitute a physical page address. For a CPU memory access to on-board memory, the master multiplexor and the board/bank select logic send the page address (along with two lower-order address bits, LA<6,7>), directly to the main memory to be used as the most significant bits of the physical memory address. For a CPU memory access to optional add-on memory, the master multiplexor does not pass any addresses. Instead, the page address goes directly from the MAP RAM to the system memory bus through the system bus driver.

During MAP load/dump operations, the MAP register to be read or written to is addressed by the selected bits from the MAP multiplexor and the contents to be transferred pass through the MAP driver to or from the system bus. The other functions of the master multiplexor in supplying refresh addresses and addresses for the video screen buffer will be discussed later along with the video interface.

Oncard Memory and Parity Logic

Resident on the Model 10 or 10/SP 2-card SPU is either one or two 128-kilobyte banks of dynamic random-access memory. Each bank consists of 18 64-K by 1 memory chips — one chip for each of 16 data bits plus a chip for a parity bit for each byte of data. The memory chips used have a 120-nanosecond access time and are accessed at a maximum rate of 4 megahertz (once every 250 nanosecond).

The parity logic consists of two 9-bit parity generator ICs together with a PAL device that generates parity signals for the system bus and a parity-fault signal that requests a nonmaskable interrupt from the microECLIPSE CPU. Code that allows control of the parity detection function using I/O instructions is resident in the control memory.

Diskette Interface

The Model 10 or 10/SP diskette interface is based on two large-scale integrated circuit devices, a DMA controller and a diskette controller. These two LSI chips and their associated circuitry are shown in Figure 4-4.

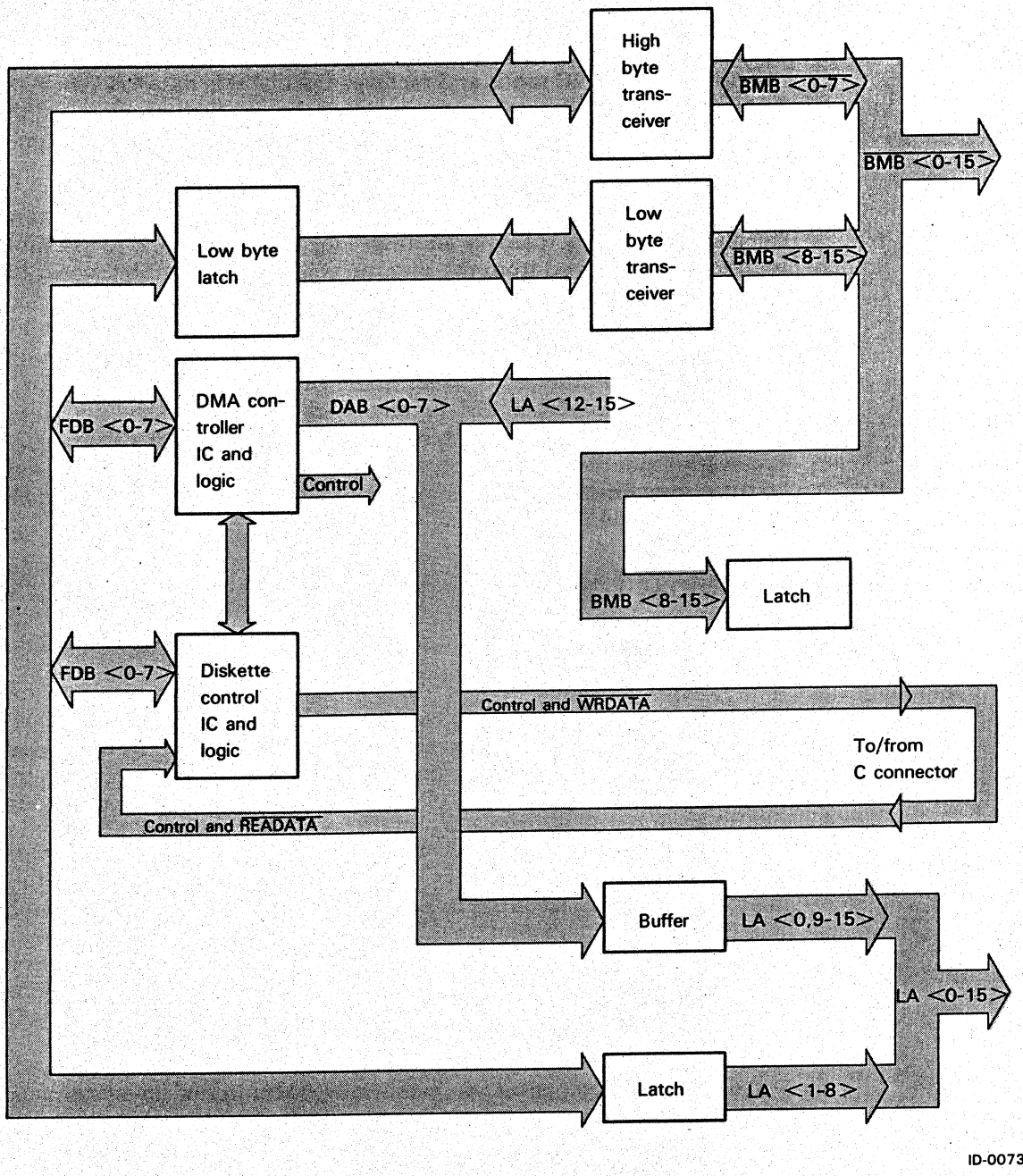


Figure 4-4 Diskette interface

The diskette interface communicates with the rest of the SPU via the system (BMB) bus and the logical address (LA) bus. The diskette control IC and the discrete logic associated with it supply all the signals necessary to control the diskette drive including automatically generated write precompensation. The diskette controller IC sends and receives the diskette data via WRDATA and READATA. A phase-locked loop is included to supply precisely timed pulses to the diskette controller in order to provide accurate data separation on diskette reads.

The DMA controller contains the logic necessary to request the system bus, to calculate and supply memory addresses for DMA transfers, to assemble 8-bit bytes of data received from the diskette controller into 16-bit words, to count the number of words transmitted, to assert an address strobe that clocks the high-order bits of the current memory address into the high-order address latch, and to detect the end of a DMA transfer and request an interrupt to notify the microECLIPSE CPU of this condition. The logic associated with the DMA controller completes each DMA memory transaction by asserting the system bus control signals necessary to read or write into system memory. (DMA write cycles are extended beyond the normal memory write time by one cycle because transfer of a byte of data from the diskette drive takes slightly longer than 1 CPU cycle.)

System Console Interface

The system console interface has two sections: the keyboard interface and the video interface.

The keyboard interface consists of an octal buffer and an 8-bit universal shift/storage register. Serial keyboard code sequences are received and assembled from the system console keyboard by the shift/storage register, which passes them on in parallel on the lower 8 bits of the system bus.

The keyboard interface detects an interrupt request from the keyboard and initiates an NMI to the microECLIPSE CPU. In response to a keyboard NMI, the microECLIPSE passes control to keyboard emulator code in the control memory, which then executes a routine that acknowledges the keyboard interrupt request (so it can send again), decodes the keyboard input, translates it to an ASCII character if it is one, or takes other appropriate action. The emulator code then notifies the CPU if it has a character by requesting a standard I/O interrupt and passes control back to the CPU.

The video interface part of the monochrome monitor of the system console, as an I/O device, is also emulated in code contained in the control memory. When a user program issues an I/O instruction to send a character to the system console, program control passes to control memory. Control memory code then looks up the ASCII character in a bit map table and writes it to screen memory in the appropriate place. The control memory then indicates that it is done by setting the TTI Done flag to 1. In any event, it passes control back to the user program.

The video interface hardware is diagrammed in Figure 4-5. At its heart is a CRT controller IC, which produces horizontal and vertical synchronizing signals and memory addresses. Control memory code initializes and controls the CRT controller by writing data to its internal registers via the BMB bus. The memory addresses the CRT controller produces are either addresses for memory locations in which pixel values (picture element) for the video display are stored or they are refresh addresses for oncard memory, depending upon where in a system timing cycle they occur. This will be explained in the next major section.

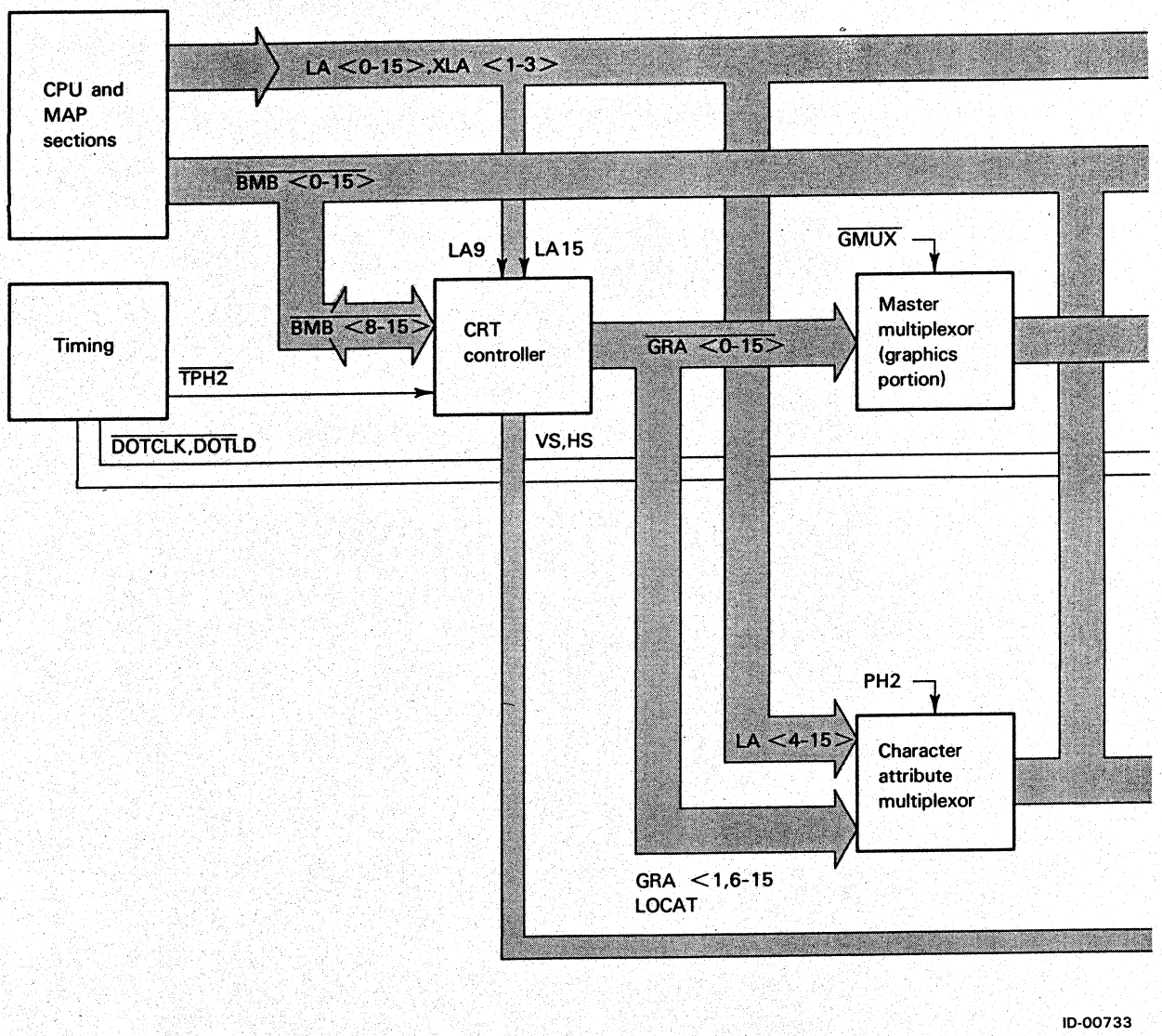
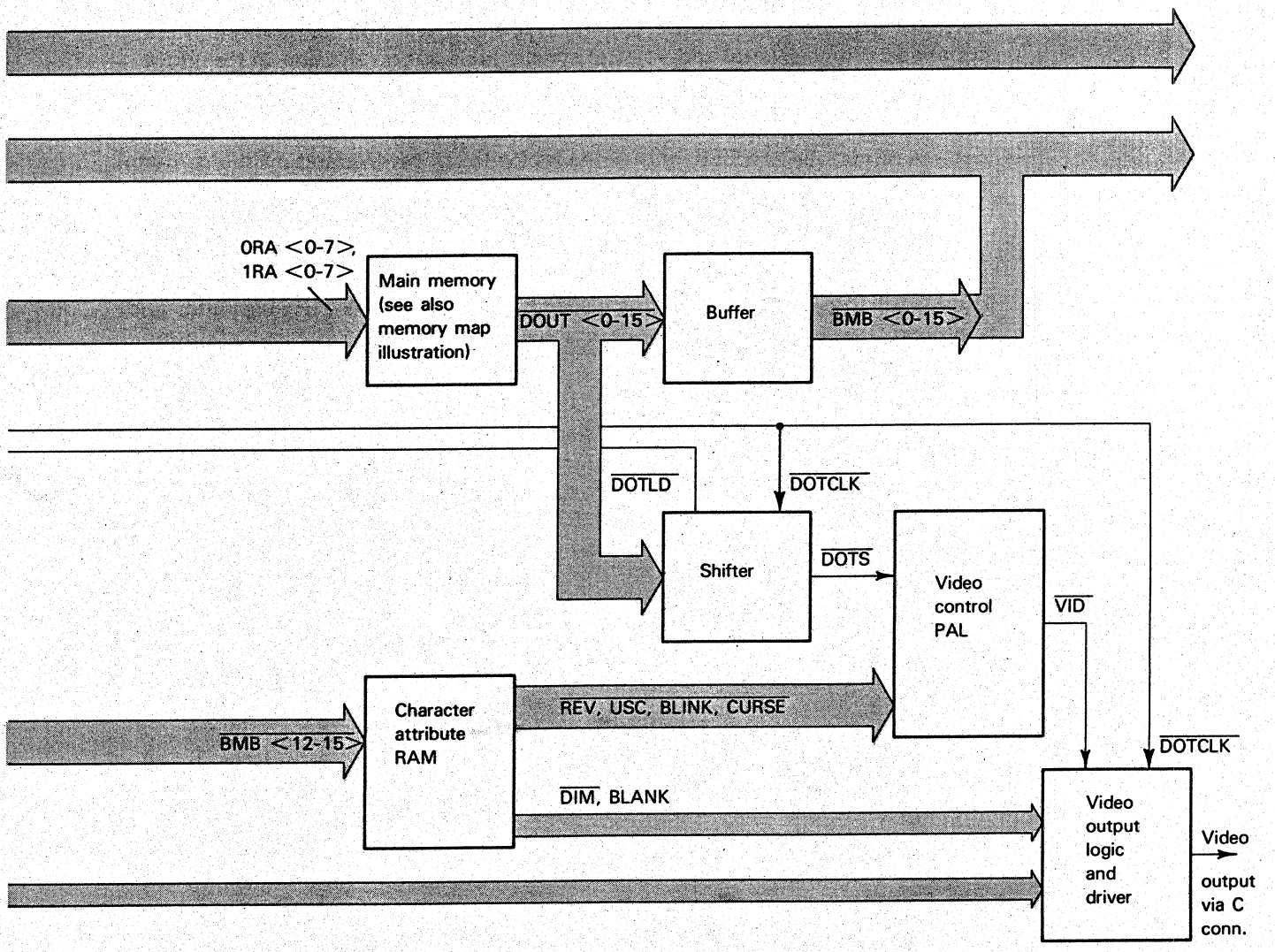


Figure 4-5 Video interface


0RA <0-7>,
1RA <0-7>
Main memory (see also memory map illustration)
DOUT <0-15>
Buffer
BMB <0-15>
DOTLD
DOTCLK
Shifter
DOTS
Video control PAL
VID
BMB <12-15>
Character attribute RAM
REV, USC, BLINK, CURSE
DIM, BLANK
DOTCLK
Video output logic and driver
Video output via C conn.

If the operation in progress is a video screen refresh, then the 16-bit memory address of the current video display word, GRA<0-15>, is passed by the master multiplexor to main memory. The contents of the addressed location, which is located in the 10-kilobyte screen buffer, load the shifter. The shifter, every time it receives the signal $\overline{\text{DOTCLK}}$, shifts out one "dot," or potential pixel.

The character attribute RAM, which is also addressed by (part of) the graphics address, sends control signals to a video control PAL device that conditions the raw dot signal from the shifter. The video driver drives the conditioned video signal and the horizontal and vertical sync signals out on the line to the video monitor.

The character attribute RAM can be loaded under program control. During the load operation, the attribute address to be loaded is determined by the contents of LA<4-15> and the attributes arrive on $\overline{\text{BMB}}$<12-15>.

Decode and Timing Logic

The Model 10 or 10/SP SPU timing circuitry consists of a 48-MHz oscillator and five quad D-type flip-flops, three of which are cascaded in a Johnson counter. These produce all the timing signals necessary to synchronize the video interface and the memory accesses.

Many of the internal devices of the SPU are accessed by means of memory mapped I/O instructions — that is, the devices respond to certain addresses on the local address bus. The component that facilitates this is a registered PAL that decodes logical addresses LA<1-5,8-10> and asserts control signals to the various devices in accordance with the value of these address bits.

The details of system timing and memory mapped control will be discussed in the following section.

Operational Overview

There are three keys to understanding the operation of the 2-board SPU:

In order to obtain the greatest possible speed of I/O operations of on-board devices and at the same time to implement the full ECLIPSE instruction set, the Model 10 and 10/SP system employs code sequences in control memory ROM that emulate certain, mostly I/O-related, instructions.

Accesses to those I/O devices contained on the SPU whose I/O instructions are emulated are *memory mapped*.

There are two types of access to main memory, and they are interleaved: (1) system data accesses and (2) graphics/refresh accesses.

This section discusses these ideas in greater detail and describes the operations of the SPU devices. SPU timing diagrams and signal reference tables conclude the section.

General Principles

When the microECLIPSE CPU encounters an instruction that is to be emulated, it executes a sequence of microcode that saves the program counter and jumps to an emulator entry point in control memory. The control memory then saves the state of the processor and executes a sequence of instructions that manipulate the on-board devices using memory-mapped I/O and other special-purpose instructions. When an emulator routine finishes, the state of

the microECLIPSE is just what it would be if a normal I/O instruction had been executed.

The diskette interface and the system console (monochrome monitor and keyboard) interface, appear to user software to be standard microI/O devices but actually are physically integral parts of the SPU/main-memory system: they interact directly with CPU and memory and do not use the microI/O bus. This direct interaction is accomplished by memory mapping certain I/O and control functions. In order to accomplish these I/O or control functions, the microECLIPSE CPU accesses certain hardware-assigned memory locations.

The monochrome video display is a bit-mapped DMA display. The video screen must be rewritten periodically to refresh the display. Also, main memory, because it comprises dynamic RAM ICs, must be refreshed periodically. The CRT controller IC controls both these functions by accessing main memory in alternate cycles with other SPU devices that access main memory to acquire data: the two CPUs, the SIO controller, and the diskette interface.

Memory Organization and Memory-Mapped I/O

Figure 4-6 shows the organization of memory in a Model 10 or 10/SP computer. Note in particular the existence of the 10-kiloword monochrome monitor screen buffer at the bottom of system memory address space and the 64-kiloword color video screen buffer at the top. The 10-kiloword screen buffer space at the bottom of memory is not normally accessible to user programs. It can be written to directly by means of the programming sequence outlined in Chapter 1 of this manual (setting bit 0 of the control memory status register (location 14) to 1 and mapping user memory space to screen buffer addresses).

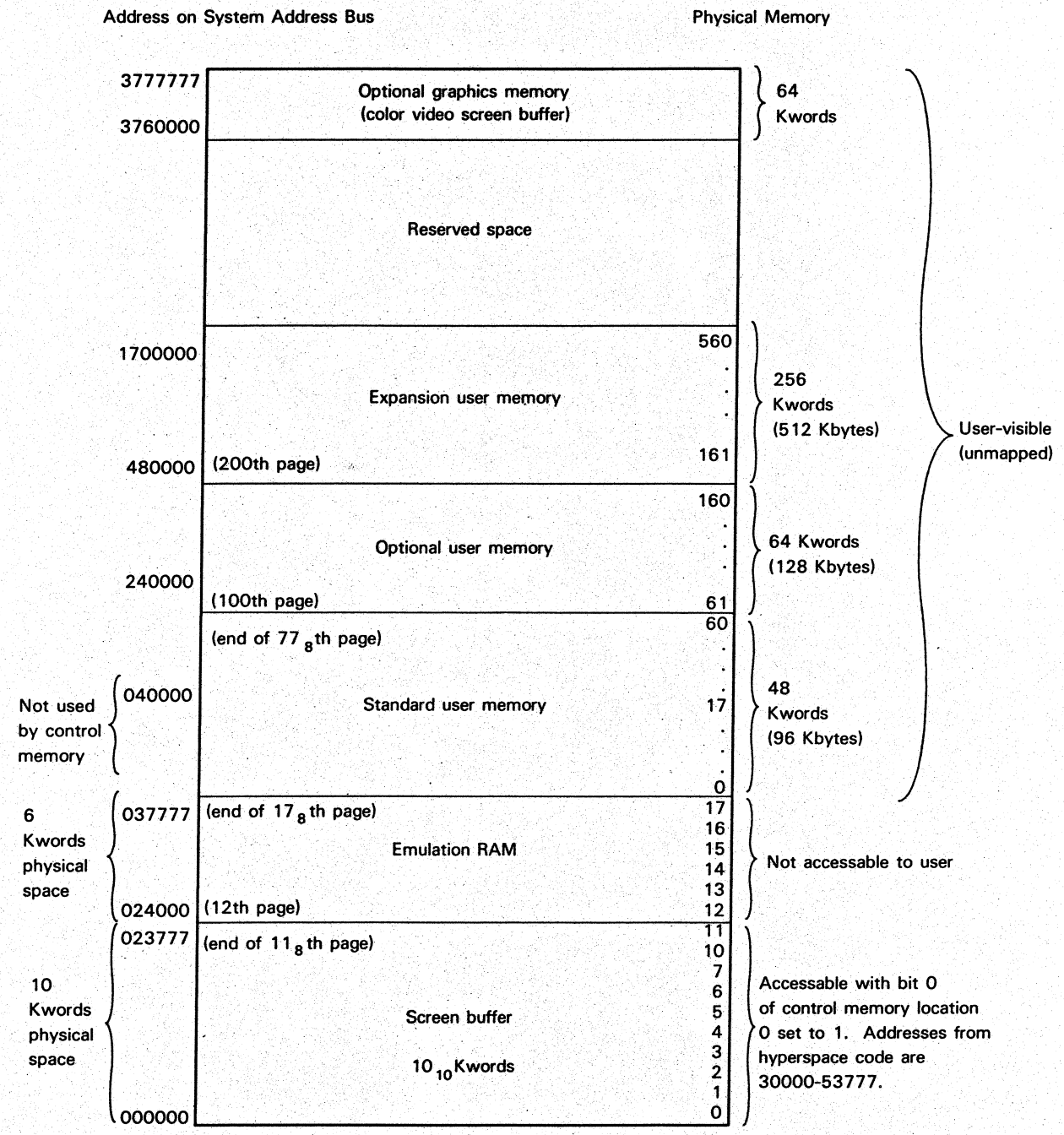


Figure 4-6 System memory organization

Also of note in Figure 4-6 are the 6-kilowords (12 kilobytes) of emulation RAM located in the memory space immediately above screen buffer memory. This emulation RAM contains downloaded D200/D211 keyboard emulation and control memory scratchpad locations. This memory is not accessible to user code. The emulation RAM together with the screen buffer, shown in Figure 4-6, constitute *control RAM* space.

Since control memory occupies 16 kilowords (32 kilobytes) of space, a Model 10 or 10/SP system with 128 kilobytes of memory has 96 kilobytes available for user programs. This is shown in Figure 4-6 with the label "standard user memory." A system with 256 kilobytes of memory has 224 kilobytes available for user programs. Such a system includes the memory labeled "optional user memory" in addition to the standard memory.

If a Model 10 or 10/SP system includes an optional expansion memory board, the amount of memory available to user programs is increased to 736 kilobytes.

The addresses along the left side of Figure 4-6 are the values of the electrical signals that can be observed on the system memory bus. These are *not* the physical addresses that are contained in the MAP. The user-visible physical memory page addresses are shown on the right of Figure 4-6. These are offset by hardware from the electrical-signal addresses by 20_8.

Note that the 16 kilobytes of control *ROM* contained in the SPU do not occupy any system memory address space. All location addresses in this ROM are mapped by the control ROM map into physical address block 0. Similarly, all memory mapped addresses are mapped into physical address block 0. These locations are accessed in the following way.

During a microECLIPSE CPU address phase, the signals MEMCYC and $\overline{\text{MB0}}$ (LA0) determine which sort of operation will occur during the current cycle:

MEMCYC	Address bit 0	Operation
0	0	microI/O bus transaction
0	1	Character attribute and parity control access
1	0	User program memory access
1	1	Control memory access

The microECLIPSE CPU sends MEMCYC and $\overline{\text{MB0}}$ in accordance with the operation it is performing: 00 for an I/O instruction, 10 for an ordinary access to user memory. The microECLIPSE CPU accesses control memory (sends MEMCYC,$\overline{\text{MB0}}$ 11) on power-up and after every nonmaskable interrupt. Subsequent memory accesses will remain within control memory until a program transfer instruction in control memory causes a return to user memory. The microECLIPSE can also perform single read or write operations in control memory with the special instructions RHYP and WHYP.

Character attributes and parity can be controlled by the special instructions RLCL and WLCL, which are similar to RHYP and WHYP. (These instructions are used only by control memory and are documented in Appendix D.) Also, parity checking can be controlled with standard I/O instructions to device code 2.

When code in control memory is executing, that is, when the microECLIPSE is sending $<\text{MEMCYC},\overline{\text{MB0}}> = <1,1>$, then references to certain memory locations control the functions of all the devices whose instructions are emulated in control ROM code. These are:

System console (keyboard and monitor)

Diskette interface

Attached (8086) processor

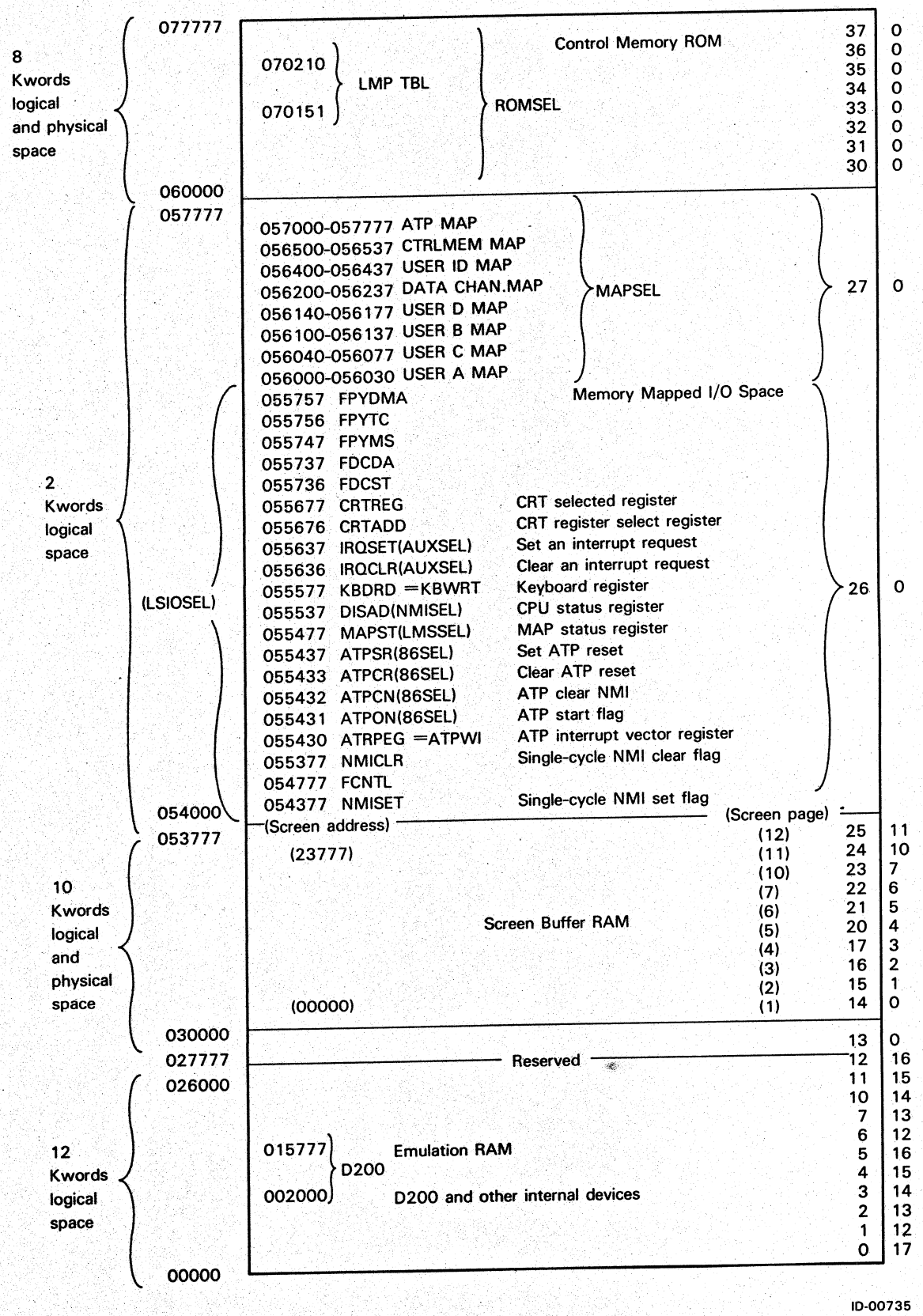
System parity logic

microECLIPSE CPU logic for the following functions:

- INTA
- MSKO
- VCT
- IORST
- DIS

Figure 4-7 shows the memory locations used to perform these functions. This figure also shows how the control ROM map assigns screen buffer space and emulation RAM to physical address space within control memory. The numbers down the right side of the figure are physical addresses; those inside the block are the logical addresses that the control memory uses.

The use of the memory-mapped locations will be described later in the discussion of operation of the various sections of the SPU. In conclusion, note that memory mapping is always enabled on the SPU: when the system first powers up, control ROM code loads an identity map for all users and the data channel, and its own control-ROM map, into the MAP RAM.

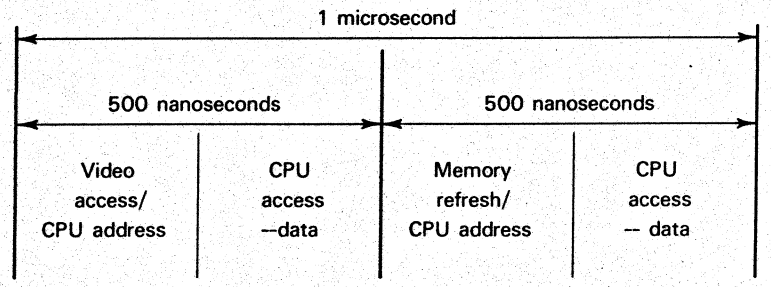


ID-00735

Figure 4-7 Control memory organization

Interleaved CPU/Graphics Memory Access

Figure 4-8 illustrates how CPU (microECLIPSE or 8086) memory accesses occur alternately between video screen accesses and memory-refresh accesses.



ID-00736

Figure 4-8 Interleaved memory accesses

A single read or write to main memory requires 250 nanoseconds. The graphics controller can supply a memory address and the resulting data can be read into the video shift register within this 250-microsecond time, and it needs to do this at least once every microsecond in order to keep the video screen refreshed. The two CPUs and the diskette and SIO DMA controllers, however, require at least twice as much time for an address/data cycle, and the diskette interface even more. To accommodate the requirements of the video interface with those of the rest of the system, video accesses to main memory are alternated with CPU/DMA-controller accesses.

The two types of memory operations are not alternated in a strict address-data, address-data sequence. Instead, a CPU or DMA address phase begins at the same time as a graphics or memory refresh cycle. Because CPU and DMA addresses must be memory mapped, it takes almost the entire 250 nanoseconds for a valid mapped address to appear at the memory pins. The graphics controller, in the meanwhile, has already asserted its address and retrieved its data, or has refreshed main memory. So, 250 nanoseconds after a CPU/DMA address has first been generated, a CPU cycle occurs in which that address, now mapped, is applied to the memory and the data returned.

microECLIPSE CPU Operations

microECLIPSE CPU operations are synchronized by a pair of underlapping clock signals, MPH<1,2>. The underlapping clocks create two distinct timing phases: phase 1 and phase 2. The internal logic on this CPU places addresses on the CPU bus and generates the enabling signal $\overline{\text{ADREN}}$ during phase 1, and it clocks data on the CPU bus in or out in phase 2. During phase 1, the microECLIPSE CPU also generates the control signals that specify the cycle type (memory, or I/O) and the MAP mode (mapped or unmapped).

Phase 1. $\overline{\text{ADREN}}$ latches control signals MEMCYC, MAP, $\overline{\text{WH}}$, and $\overline{\text{WL}}$ and addresses from the CPU bus into the microECLIPSE address latch, which applies the addresses to the logical address bus. The MAP multiplexor selects LA<1-5> to become map address select signals LM<5-9> and, if mapped mode is on,

applies bits MAP<A,B> to the MAP RAM as well (as LM<3,4>). The contents of the selected MAP RAM location, PA<0-9>, are the physical address of the memory location to be accessed. Once PA<0-9> are present, BUSADREN drives the physical address and control signals onto the system (BMB) bus as follows:

Signals PA<0-4> become signals XMA<0-4>

Signals PA<5-9> become signals BMB<1-5>

Signals LA<6-15> become signals BMB<6-15>

Signal LA0 becomes signal BMB0

A short time before BUSADREN returns to its passive (high) state, the CPU-address/Graphics-or-refresh phase (phase 1) ends and phase 2 begins.

Phase 2. As the CPU data phase begins, the master multiplexor sends BMB<8-15> to onboard memory on the row address lines. After the row address strobe, the master multiplexor sends PA<4-9> and LA<6,7> to oncard memory on the column address lines. If the address is an oncard location, a column address strobe occurs. If the address is not an oncard location, no column address strobe occurs. In this case, the memory reference is intended for the optional memory board or for the color monitor screen memory, and these devices will read data or supply it during the data phase.

If the memory operation is a data write, the microECLIPSE CPU places data on the CPU bus during the second half of phase 2, and the CPU bus transceiver drives it onto the system bus. It passes through an internal buffer to onboard memory as signals DIN<0-15>, where RAM-WH and RAM-WL cause it to be written.

If the operation is a data read, the data — whether from onboard memory (DOUT<0-15>) or from external memory — appears on the system bus during the second half of phase 2, and, after it is passed through the CPU transceiver, it is clocked into the microECLIPSE CPU at the end of phase 2.

NOTE *The above descriptions apply for normal microECLIPSE memory read or write operations; however, if the selected physical address in the MAP RAM is write-protected or validity-protected, the memory operation is not completed. Write protection or validity protection inhibits memory write-enable signals. Validity protection causes the microECLIPSE CPU to take a validity protection fault.*

8086 CPU Operation

The 8086 CPU is synchronized by an 8-megahertz clock signal, 8086CLK. A typical 8086 instruction cycle requires four 125-nanosecond phases, or T-periods. Usually the 8086 places an address on its CPU bus (86AD<0-19>) during period T1, and reads data in during T3 or sends data out during T2 and T3. The 8086 control logic on SPU1 forces the 8086 address phase to occur during system phase 1. An 8086 address/data cycle then proceeds in exactly the same manner as a microECLIPSE address/data cycle, with the following exceptions.

1. The 8086 supplies the control signals SYSMEMCYC, SYSADREN, SYSWL, and SYSWH.

2. The MAP multiplexor uses bits XLA<0-3> and LA<1-5> to produce LM<0-9>.

3. The 8086 CPU always runs in mapped mode.

4. If a validity protected physical address is encountered, the 8086 CPU is paused and a microECLIPSE NMI is generated.

If a microECLIPSE NMI or an I/O interrupt request occurs while the 8086 CPU is running, the 8086 will pause and the microECLIPSE will resume execution.

When the 8086 CPU executes an OUT instruction, it asserts M/$\overline{IO}$ and $\overline{86WR}$ (both active low), which cause 86OUT. This results in an NMI to the microECLIPSE and the 8086 CPU pauses.

When the microECLIPSE executes a *Load ATP Interrupt Vector* instruction (DOB ATP), the low order byte in the specified accumulator is loaded into the ATP vector register and a signal called 86INTR is latched high. When the microECLIPSE CPU starts up the 8086 processor (with a Start command to the ATP), the 8086 will respond to the interrupt request by asserting 86INTA. 86INTA enables the output of the vector register to place the interrupt vector on the 86A<0-7> lines. The 8086 processor then transfers program control to the location specified by that interrupt vector.

All of the I/O instructions to the ATP are emulated by code in control ROM, which issues memory-mapped instructions to effect the required actions. In particular, the memory mapped ATP registers are in control memory location 55430, and the Start, Clear, DOB, and DOC ATP functions are controlled by local address bits LA<13-15>.

System I/O Controller Operations

The SIO controller is synchronized by the same timing signals as the microECLIPSE CPU and has the same address/data timing. During programmed I/O operations, involving either internal SIO devices (RTC, PIT) or external devices, the microECLIPSE and the SIO controller communicate over the microECLIPSE CPU bus. The address/data timing is basically the same as it would be for memory operation.

When the SIO controller is performing data channel operations to pass data directly between an I/O device and system memory, it supplies addresses, data, and memory control signals just as the microECLIPSE CPU does; however, it can only do this when granted access to the system bus.

When the SIO controller has or needs a 16-bit word to transfer between itself and memory, it asserts the signal EBREQ. The bus arbitration logic gives control of the system bus to the SIO controller by sending EBLOCK low *if* the microECLIPSE and 8086 processors are not using the bus and a request for the bus from the diskette DMA controller did not precede the SIO request.

The SIO controller, in addition to its internal I/O devices (RTC, PIT), contains one half of the CPU status register. The other half consists of a separate octal latch. Code in the control ROM emulates the CPU read status instruction (DIS) by sending the address 05537 on the local bus. The SPU decode logic asserts NMISEL on detecting this address. NMISEL places the lower half of the CPU status on the microECLIPSE CPU bus. The control ROM code simultaneously reads the upper byte of CPU status from the SIO controller and the lower half from the latch and writes the result into the accumulator specified in the DIS instruction.

Diskette Subsystem Operations

The diskette subsystem logically consists of three major components: the DMA controller IC and its associated logic, the diskette control IC and its logic, and the third part, which is not physically a part of the diskette controller hardware, the code in the control ROM that provides the control signals for other two parts.

All instructions addressed to device code 20 result in control transfer to the diskette emulator routines in the control ROM. These routines fall into two categories: those that control operation of the diskette and those that set up and control DMA operations to exchange data between the diskette and system memory.

The diskette control routines in ROM use memory-mapped location 054777 to initialize the diskette subsystem. This includes turning the diskette drive motor on, selecting the logical drive and diskette side numbers, and selecting byte-transfer order — high byte first or low byte first. Location 054777 is also used to reset the entire subsystem.

The diskette control routines in ROM use two memory-mapped locations to communicate with the diskette controller: access to control location 055737 sets up data transfers between diskette controller and the logical address bus; and access to 055737 sets up status transfers. Once a transfer is initiated by placing one of these addresses on the logical bus, the ROM code programs the diskette controller by writing to its status registers. The ROM contains routines for each of the eight commands possible with the *Specify Command and Diskette Address* instructions, as well as routines for the two types of *Read Diskette Status* instructions.

The DMA controller routines in ROM use three memory-mapped locations:

Access to location 055757 sets up the loading of the memory address register in the DMA IC.

Access to location 055756 sets up the loading of the terminal count register in the DMA IC.

Access to location 055747 sets up the mode of operation of the DMA IC.

The control ROM code loads the memory address register and the terminal count register in two steps; first it loads the low byte, then it loads the high byte.

The DMA/IO PAL device asserts the control signals required to perform the loading of the DMA and diskette control registers. These signals include E, $\overline{\text{FLOPL}}$, and TOMB.

The control ROM code that emulates the *Load Memory Address Register* and *Load Word Count* instructions use short sequences of instructions to load the appropriate DMA IC registers. The actual initiation of DMA transfers occurs in conjunction with the emulation of the DOA instructions that specify diskette reads or writes.

After the control ROM program sets up the DMA controller to perform the DMA operation for a diskette data transfer, the controller automatically issues the sequence of control signals, both to the diskette controller and to the bus arbitration logic and system memory, to perform the transfer:

The first phase of the memory cycle:

1. In a read operation, the diskette controller, having been set up by control codes from the control ROM, reads in serial data from the diskette drive on RD-PLS and accumulates the bits as they arrive in a byte-in buffer.
2. When the diskette controller has a fully assembled byte, it asserts DRQ.
3. The DMA controller in response asserts DMAHRQ.
4. The bus arbitration logic grants the system bus to the DMA controller, if the microECLIPSE and 8086 processors are not using it. DMAHLDA signals this.
5. The DMA controller places the memory address of the location to be accessed on bits DAB<0-7> and FDB<0-7>.
6. The DMA controller asserts DMADSTB to clock DAB<0-7>, which become destination memory address bits LA<1-8>, into the high-order address latch. The DMA controller also sends $\overline{\text{DACK}}$ to the diskette controller. At about the same time $\overline{\text{DMACYC}}$ falls, passing the address latch contents and bits FDB<0-7> to the logical address bus to become LA<0-15>.
7. The DMA controller asserts $\overline{\text{DI/OR}}$, which causes the diskette controller to place its data byte on FDB<0-7>. $\overline{\text{DI/OR}}$ also causes DMARDY to go low, extending the DMA cycle by 500 nanoseconds. At about the same time, the logic associated with the DMA controller generates an address enable signal for the system, which begins the CPU-type memory cycle described earlier; except in this case, one byte rather than one word is written to memory.

The second phase of the memory cycle:

1. The DMA controller asserts $\overline{\text{MEMWRT}}$. The DMA data enable signal, DMADATEN causes the DMA/IO PAL device to assert an output enabling signal to one or the other of the DMA output transceivers, depending on the value of the high-byte bit, HIB.
2. DMARDY goes high, releasing the DMA controller, which releases the system bus.

The entire sequence above takes 3.5 microseconds. The fastest data rate the diskette controller can accommodate is one byte every 32 microseconds. A DMA memory-to-diskette sequence is very similar, but takes 500 nanoseconds less, as the signal $\overline{\text{DI/OW}}$ does not cause an extension of the DMA cycle as did $\overline{\text{DI/OR}}$.

DMA memory-to-diskette transfers read 16-bit words from memory and write 8-bit bytes to the diskette. The DMA controller sets up the address on the local address bus as above, in response to a request for a data byte from the diskette controller. One byte of data from the system (BMB) bus passes through the enabled transceiver (according to the state of $\overline{\text{FLOPH}}$ and $\overline{\text{FLOPL}}$) and is latched into an octal latch by the signal DMADATEN.

After the first byte of data is read into the diskette controller, the state of the high-byte bit changes and the second byte of data passes through the other (newly enabled) transceiver and into the octal latch. Somewhat later the signal DMACYC goes low, which together with $\overline{\text{MEMRD}}$ causes $\overline{\text{DMAWDOE}}$ to enable the output of the latch. The diskette controller then sends this second byte to the diskette.

When the DMA controller has transferred a number of bytes equal to the terminal count, it sends TC to the diskette controller, which causes it to cease transfers.

Keyboard Operations

Two keyboard control signals, $\overline{\text{PUKARD}}$ and $\overline{\text{PUKAWR}}$, from the DMA/IO PAL device set up read and write operations to the keyboard. When no keyboard operation is in progress, these signals are both high. In this case, bits coming from the keyboard will be shifted into the keyboard shift register by the keyboard clock signal, $\overline{\text{KB-SCK}}$.

The keyboard sends the signal KBIRQ high to indicate that it has finished sending a keycode. This causes a CPU NMI that causes the control memory to enter its TTI-emulator routine. The routine executes a read instruction that accesses location 055577. (The read instruction actually reads the keycode from the shift register into an accumulator.) The occurrence of 055577 on the logical address bus during a control memory read operation will send $\overline{\text{PUKARD}}$ low and $\overline{\text{PUKAWR}}$ high, enabling the shifter output and shifting the contents left. The result is that the keyboard receives a logical low signal, which causes it later to clear KBIRQ.

Control memory now writes a word containing the updated status of the keyboard LEDs and bell to "location 055577." This loads the LED word into the shift register, sets $\overline{\text{PUKARD}}$ high and $\overline{\text{PUKAWR}}$ low. When KBIRQ goes low, the LED word will be clocked out to the keyboard at the same time that a new keycode (if there is one) is clocked into the shift register.

Monochrome Monitor Video Operations

The operation of the monochrome monitor as a character display device is completely controlled by emulation code in the control ROM. The DOA TTO emulation routine simply writes a bit pattern for the specified character into the screen buffer.

The control ROM, in addition to TTO emulation code, contains initialization routines that configure and program the CRT controller IC in the video interface. These control memory routines select CRT control registers with memory-mapped address 055676 and write or read these registers with address 055677.

The monochrome video display is controlled by the CRT controller IC and character attribute logic. The sequence of events in a graphics cycle is:

1. The high-low transition of $\overline{\text{TPH2}}$ clocks a graphics address out of the CRT controller. Assume that the least significant bit of the address, LOCAT (GRACYC) is 1.
2. CLK10 enables the output of the graphics multiplexor.
3. $\overline{\text{RAS}}$ strobes into memory the row addresses GRA<8-15>, selected by GMUX high.
4. GMUX goes low, selecting GRA<0-7>.
5. $\overline{\text{KCAS}}$ strobes in the column address GRA<0-7>.

6. With $\overline{\text{DOTLD}}$ low, rising edge of $\overline{\text{DOTCLK}}$ places 16 bits of data from the memory into the video shift register.
7. The falling edge of $\overline{\text{DOTCLK}}$ clocks the first bit in the just loaded word out to the video control PAL. The video PAL and an associated latch and driver condition the video output in accordance with attribute signals they receive from the character attribute RAM. For example, if the attribute of a character is *dim*, the driver will reduce the video drive voltage to make all dots for the character less bright.
8. Just before the eighth bit has been clocked out, the signals $\overline{\text{ATTLD}}$ and $\overline{\text{CATCH}}$ cause a new set of attribute signals to issue from the attribute RAM. In this way the high byte and low byte of a 16-bit word can have separate sets of attributes.
9. While the 16 dots corresponding to one memory read are being clocked out, a CPU memory access (data) phase occurs, and another graphics phase starts. This time LOCAT (GRACYC) will be low, indicating that this is a memory refresh cycle. Bits GRA<8-15> are strobed into memory row addresses by $\overline{\text{RAS}}$ to refresh all the memory locations in the addressed row. No column address strobe is asserted during a memory refresh cycle.

A user program can load the character attribute RAM with a special instruction, WLCL (accumulator format 064001) which sets LA0 to 1 and SYSMEMCYC low. During phase 2 this causes the attribute RAM location addressed by the contents of AC0 (on LA<4-15>) to be loaded with the contents of AC1 (on $\overline{\text{BMB}}$<12-15>). The format for character attribute data is given in Appendix D.

Memory-mapped locations accessed with WLCL instructions and arbitrary data also control the independent blink enables of attribute and cursor video:

20000	Attribute blink enable
20001	Attribute blink disable
20002	Cursor blink enable
20003	Cursor blink disable

MAP Load Operations

microECLIPSE load-map operations are executed in microcode contained on that CPU and its XMCs; ATP (8086) load-map operations are emulated in code contained in the control ROM. In either case, a *Load Map* instruction ultimately causes the decode logic to assert MAPS, which enables write operations to MAP RAM, residing in memory-mapped locations 056000 through 057777. MAPS also selects logical address bits LA<6-15> to become the map register select bits LM<0-9>. The data to be written into the addressed map register appears on $\overline{\text{BMB}}$<0-15>.

Control ROM Operations

Besides the code that emulates internal-device instructions, the control ROM also contains power-up self test routines, a bootstrap loader, and the virtual console program. To perform its functions, the code in control ROM needs, in particular, to be able to set and clear both kinds of interrupts to the microECLIPSE CPU — standard interrupts and NMIs. To do this, the routines access the following memory mapped locations:

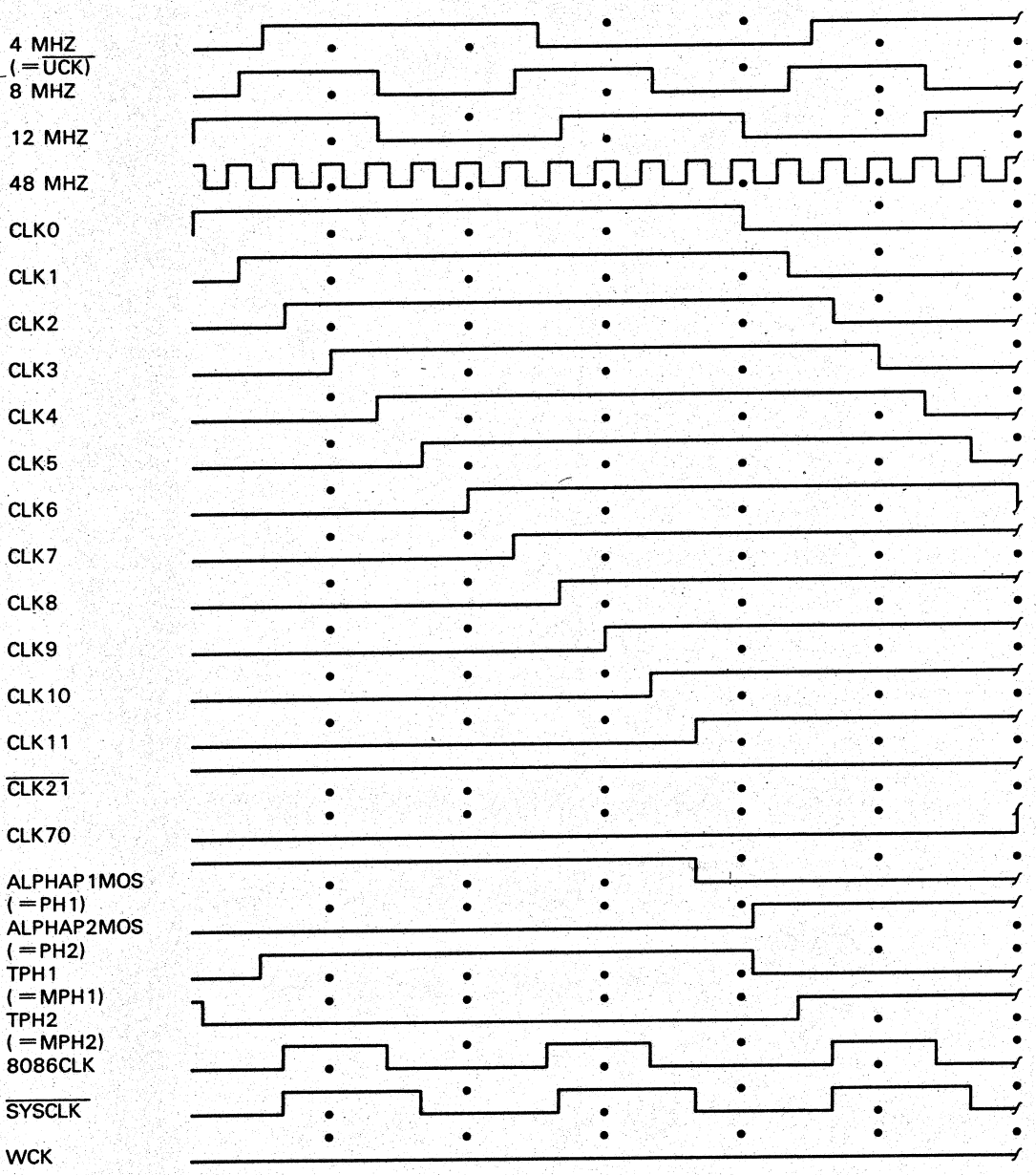
054377	To set an NMI
055377	To clear an NMI
055637	To set a standard interrupt
055636	To clear a standard interrupt

The virtual console program uses the capability to set and clear NMIs to implement the single-step function.

System Timing

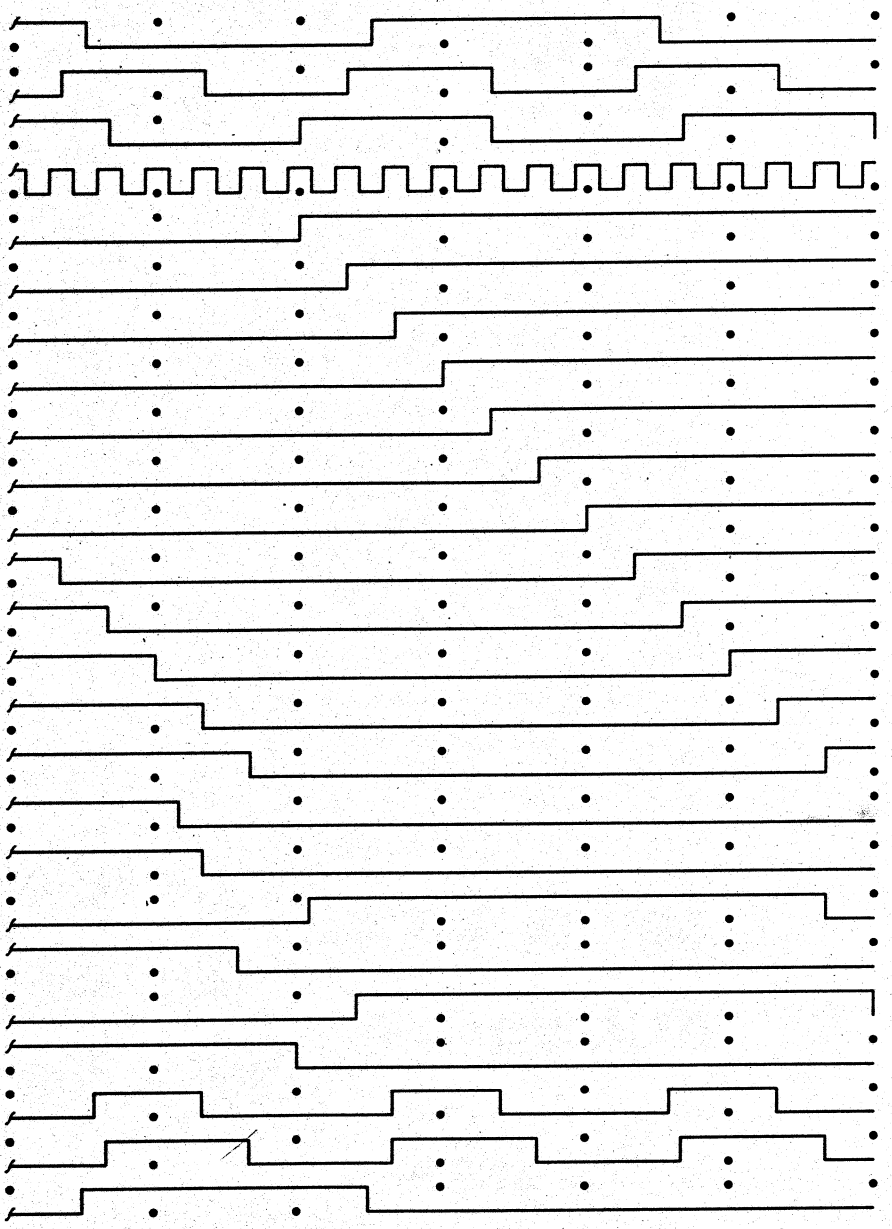
Figure 4-9 shows the most important system timing signals and their relationships to one another.

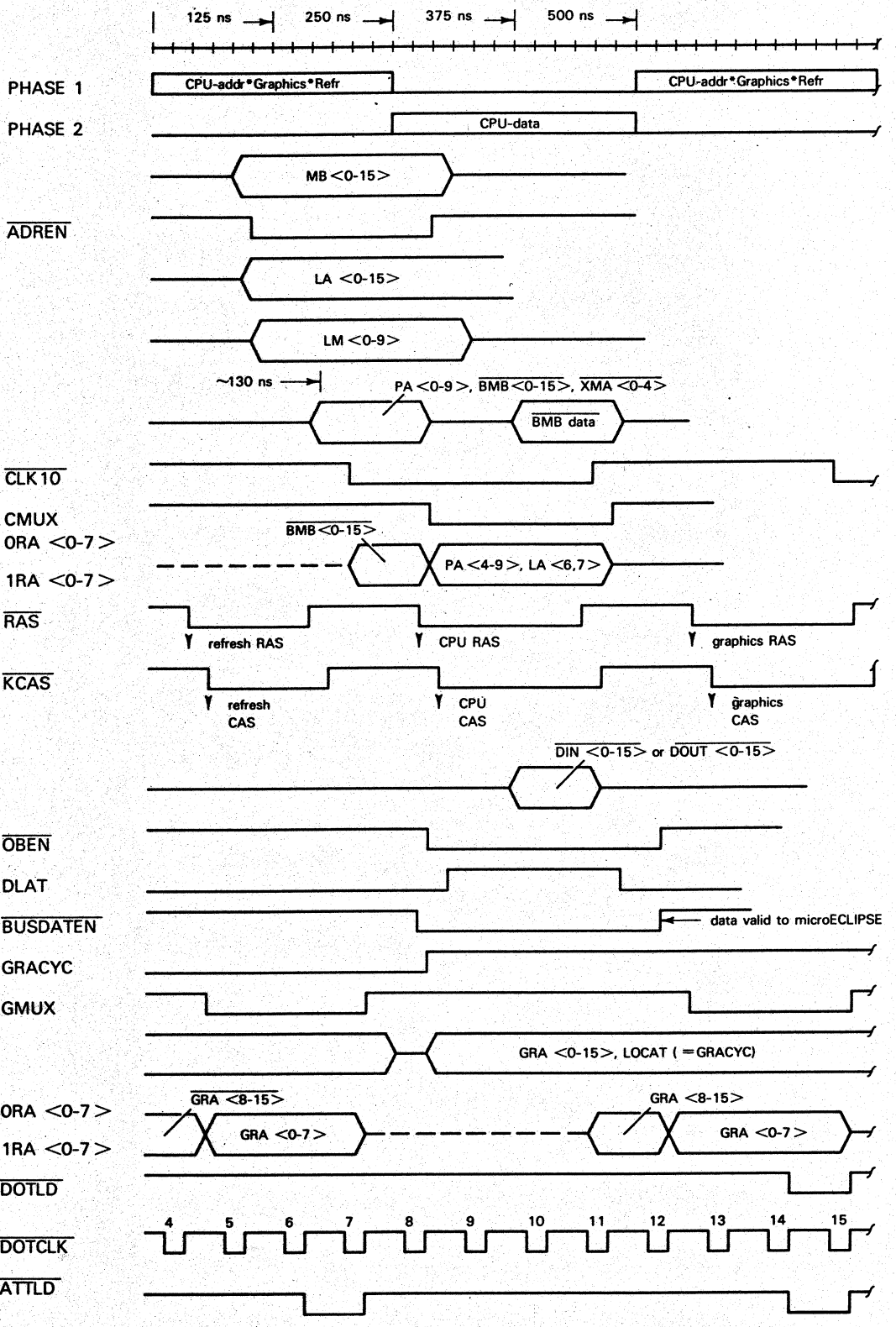
Figure 4-10 shows the main events that occur in CPU-type and graphics/refresh system operations.



ID-00737

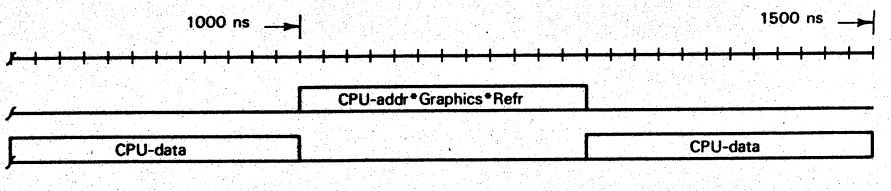
Figure 4-9 System timing signals

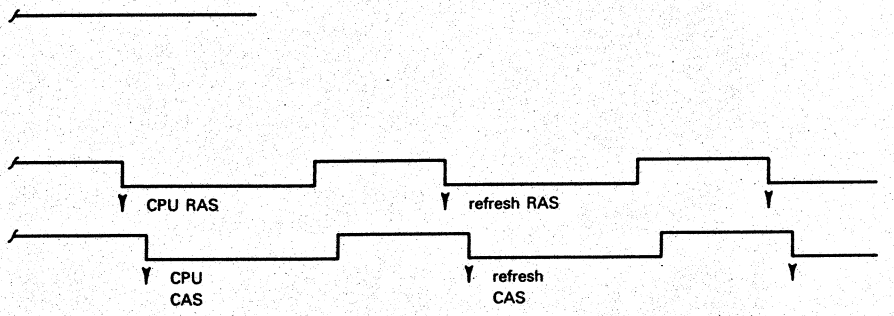


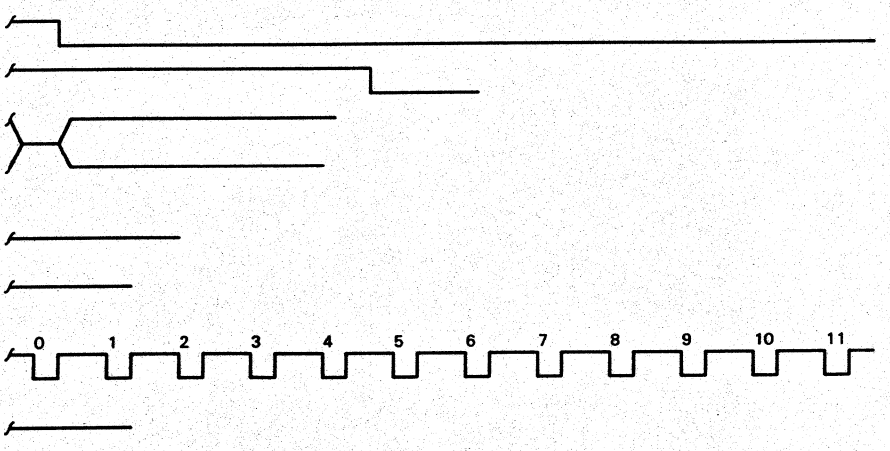


ID-00738

Figure 4-10 System operations


1000 ns
1500 ns
CPU-addr*Graphics*Refr
CPU-data
CPU-data


CPU RAS
refresh RAS
CPU
CAS
refresh
CAS


0
1
2
3
4
5
6
7
8
9
10
11

Signals

Table 4-1 lists all the external signals sourced or sunk by the two SPU boards. Table 4-2 concludes the chapter by listing all the internal signals of the SPU boards that are mentioned in the text or in figures of this book.

Table 4-1 External Signals

Name	Pin(s)	Source	Destination	Description
Memory Bus Signals				
$\overline{\text{BMB0}}$	B50	SPU1	Memory	Address phase: user memory
		Memory (SPU2)	SPU1	Data phase: data word msb
$\overline{\text{BMB<1-15>}}$	B23, 24, 27, 28, 31-34, 37, 38, 41, 42, 45, 46, 49	SPU1	Memory	Address phase: low-order address bits.
		Memory (SPU2)	SPU1	Data phase: low-order data bits
$\overline{\text{XMA<0-4>}}$	B25, 40, 44, 47, 51	SPU2	Memory	Address phase: high-order address bits ($\overline{\text{XMA0}}$ is msb)
$\overline{\text{SYSCLOCK}}$	B52	SPU2	Memory	System clock
$\overline{\text{XNMI}}$	B17	Parity logic (SPU2)	SPU1	Indicates a parity fault
$\overline{\text{TPH1}}$	B43	SPU2	Memory	System timing: indicates beginning of CPU address phase
12MHZ	B10	SPU2	Optional color monitor interface	System timing
BRDY	B18	Memory	SPU1	Data phase: indicates memory is ready to finish cycle
BUSMCYC	B26	SPU2	Memory	Address phase: indicates memory reference (otherwise cycle is I/O)
$\overline{\text{BUSADREN}}$	B48	SPU2	SPU1, memory	Indicates valid address
$\overline{\text{WH}}$ /$\overline{\text{PARH}}$	B30	SPU2	Memory	Address phase: high byte write
		Memory	SPU2	Data phase: high byte data odd parity
$\overline{\text{WL}}$ /$\overline{\text{PARL}}$	B29	SPU2	Memory	Address phase: low byte write
		Memory	SPU2	Data phase: low byte data odd parity
microI/O Bus Signals				
BMCK, $\overline{\text{BMCK}}$	B1, 2	I/O bus interface	I/O bus	Differential system clock for I/O devices
BIO1, $\overline{\text{BIO1}}$	B4, 5	I/O bus interface	I/O bus	Differential bidirectional I/O data

Table 4-1 External Signals (Continued)

Name	Pin(s)	Source	Destination	Description
		I/O bus	I/O bus interface	line used for serial transfer of high byte of data word
BIO2, $\overline{\text{BIO2}}$	B12, 13	I/O bus interface	I/O bus	Differential bidirectional I/O data line used for serial transfer of low byte of data word
		I/O bus	I/O bus interface	
BI/OCLOCK, $\overline{\text{BI/OCLOCK}}$	B15, 16	I/O bus interface	I/O bus	Differential bidirectional clock; synchronizes data transfers
		I/O bus	I/O bus interface	
$\overline{\text{CLEAR}}$	B6	SPU1	I/O bus, SPU2	System reset caused by power up
$\overline{\text{EXTINT}}$	B8	I/O device	SPU1	Interrupt request
INTPOUT	B19	SPU1	I/O device	Interrupt priority line
DCHPOUT	B21	SPU1	I/O device	Data channel priority line
$\overline{\text{EXTDCHR}}$	B9, SIO39	I/O device	SPU	Data channel request
Power Supply Signals				
POWEROK	B22	Power supply	microI/O interface	Indicates that DC power is within specifications
$\overline{\text{PF}}$	B20	Power supply	SIO (SPU1)	Indicates that AC power is not within specifications
$\overline{\text{RTC}}$	B35, SIO42	Real-time clock pcb	SIO (SPU1), video interface (SPU2)	Line frequency
DC Power				
GND	B3, 11, 14, 36,		All cards on backplane	System ground lines
	53, 54, A7		Printer port on SPU1	
	Even pins C18-C50		Diskette interface	
	Even pins A10-A50		Video monitor interface	
+5V	B57, 59, 60, A9	Power supply	SPU1, SPU2 Memory, Printer port	Regulated +5-volt power
−5V	B58, A3	Power supply	microI/O transceiver	Power for differentially driven I/O bus
+12V	B55,	Power supply	SPU clock drivers, printer port	Power for RS-232 communication line
−12V	B39	Power supply	Printer port	Power for RS-232 communications line

Table 4-1 External Signals (Continued)

Name	Pin(s) Source	Destination		Description
Diskette Interface Signals				
$\overline{DS<0,1>}$	C41, 39	SPU1	Diskette drive	Selects diskette drive 0 or 1
$\overline{MOTOR}$	C35(SPU1)	SPU1	Diskette drive	Turns on the diskette drive motor
$\overline{IN}$	C33(SPU1)	SPU1	Diskette drive	Specifies direction of diskette data transfer
$\overline{STEP}$	C31(SPU1)	SPU1	Diskette drive	Steps diskette head in or out
$\overline{WRDATA}$	C29(SPU1)	SPU1	Diskette drive	Serial clock and data bits to the diskette drive
$\overline{WRGATE}$	C27(SPU1)	SPU1	Diskette drive	Enables write data to the diskette drive
$\overline{SIDE1}$	C19(SPU1)	SPU1	Diskette drive	Selects diskette drive head
$\overline{FRDY}$	C17(SPU1)	Diskette drive	SPU1	Indicates diskette drive is ready for transfer
$\overline{INDEX}$	C43(SPU1)	Diskette drive	SPU1	Indicates diskette head is at the beginning of a diskette track
$\overline{TRK00}$	C25(SPU1)	Diskette drive	SPU1	Indicates that diskette head is over track 0
$\overline{WRPT}$	C23(SPU1)	Diskette drive	SPU1	Indicates attempt to access a write protected portion of the diskette
$\overline{READDATA}$	C21(SPU1)	Diskette drive	SPU1	Serial data pulses from the diskette drive
Printer port Signals				
TTIN	A1(SPU1)	Printer port	SPU1	Serial data input
TTOUT	A21(SPU1)	SPU1	Printer port	Serial data output
DTR	A10(SPU1)	SPU1	Printer port	Opens and holds open the communication line
CTS	A2(SPU1)	Printer port	SPU1	Enables data transmission
Keyboard Signals				
$\overline{KB\text{-}SCK}$	A9(SPU2)	Keyboard	SPU2	Clock signal produced by the system console keyboard
$\overline{KB\text{-}IRQ}$	A8(SPU2)	Keyboard	SPU2	Indicates that the keyboard is requesting service
$\overline{KB\text{-}SI}$	A5(SPU2)	Keyboard	SPU2	Serial input of keycode from keyboard

Table 4-1 External Signals (Continued)

Name	Pin(s) Source	Destination	Description	
KB-SO	A7(SPU2)	SPU2	Keyboard	Serial output of LED/bell status from control memory to keyboard
KB-RST	A4(SPU2)	SPU2	Keyboard	Keyboard reset signal
Monitor Signals				
VIDEO	A13(SPU2)	SPU2	Monitor	Conditioned video signal — turns on or off pixel specified by current value of VSYNC and **HSYNC**
VSYNC	A15(SPU2)	SPU2	Monitor	Video monitor vertical sync signal
HSYNC	A17(SPU2)	SPU2	Monitor	Video monitor horizontal sync signal
OREOX	A27(SPU2)	Monitor	SPU2	Indicates that the monochrome video monitor is present in the system
Inter-SPU-Board Signals				
FETCH	CPU9, XMC10, J1-45	microECLISPE CPU	Decode logic	Indicates the memory operation in progress is a microECLIPSE instruction fetch
NMI	CPU45, J1-58	OR-gate (ENMI or XNMI)		Nonmaskable microECLIPSE CPU interrupt
RESET	CPU3, SIO3, XMC7, J1-57	Timing logic	Sections throughout 2-board SPU	Causes microECLIPSE CPU to enter and remain in a halted state until it receives an NMI; causes SIO chip to reset all internal registers and counters, to load the settings of SIO switches into its internal devices, to set the Power Up bit in the CPU status register, then to issue an NMI; puts XMCs in a known idle state; resets the video controller
LA<0-15>	J11-J120	CPU address latches, diskette DMA controller	Control memory, MAP RAMs, decode logic, video interface	Logical addresses generated by either the microECLIPSE CPU or the 8086 CPU
INTR2	J1-55(SPU2)	Decode logic	microECLIPSE CPU (wire-ORed with EXTINT)	Used by control memory programs to request an interrupt

Table 4-1 External Signals (Continued)

Name	Pin(s)	Source	Destination	Description
READY	CPU4, SIO13, XMC9, J1-56		microECLIPSE SPU, SIO, XMC(s) and bus arbitration logic	When high indicates bus transaction in progress will terminate this cycle
MAPEN	J1-54	CPU address latch	MAP multiplexor	Enables MAP logic (latched CPU MAP signal)
VPROT	CPU48	MAP RAM	microECLIPSE CPU	Validity-protection violation
WPROT	CPU2	MAP RAM	microECLIPSE CPU	Write-protection violation
TPH<1,2>	J1-33, J1-34	Timing logic	DMA controller	2-phase system clock: TPH1 is asserted during the CPU-access phase, and TPH2 during the graphics/refresh phase of system operations
8086	J1-60	8086 CPU control logic	8086 address latch, MAP multiplexor	Indicates 8086 processor is running
$\overline{8086CLK}$	J1-37	Timing logic	8086 CPU	Clock signal for 8086 processor
$\overline{8MHZ}$	J1-25	Timing logic	microI/O bus	Master clock for microI/O bus transceiver (also divided down to become 4-MHz master clock for microECLIPSE CPU)
$\overline{CLK4}$	J1-27	Timing logic	DMA and I/O control PAL	Synchronizes the extension of memory cycle for internal-devices accesses and for pended external memory cycles
CLK8	J1-26	Timing logic	8086 bus control logic	Synchronizes 8086 system bus control signals
CLK11, $\overline{CLK11}$	J1-29,J1-30	Timing logic	microECLIPSE CPU timing logic	Source of MPH<1,2>
$\overline{CLK21}$	J1-28	Timing logic	DMA and I/O control PAL	Indicates second cycle of a control memory operation
$\overline{SYSWL}$	J1-41	microECLIPSE CPU address latch or 8086 bus control logic	Memory parity logic and system bus driver	System bus low byte write enable

Table 4-1 External Signals (Continued)

Name	Pin(s)	Source	Destination	Description
$\overline{\text{SYSWH}}$	J1-42	microECLIPSE CPU address latch or 8086 bus control logic	Memory parity logic and system bus driver	System bus high byte write enable
WRITE	J1-22	Decode logic	Main memory control logic, video and diskette interfaces	Indicates that the current operation is a data transfer out (write)
$\overline{\text{SYSADREN}}$	J1-44	microECLIPSE or 8086 CPU (via external logic) or DMA logic	microECLIPSE data transceiver and system bus driver in MAP logic	Indicates the current phase is a CPU or DMA address cycle
SYSMEMCYC	J1-43	microECLIPSE or 8086 address latch	Decode logic	Indicates current operation is a memory (main or control) access
E	J1-38	DMA and I/O control PAL	DMA controller and video interface	Indicates an access to an emulated device: keyboard, video, or diskette interface
$\overline{\text{ABGRNT}}$	J1-39	Bus arbitration PAL	8086 processor control logic	Grants system bus to 8086 processor
$\overline{\text{NMISEL}}$	J1-40	Decode logic PAL	CPU status register, microECLIPSE data bus driver	In phase 2, indicates memory mapped access to microECLIPSE CPU status register
$\overline{\text{ROMSEL}}$	J1-48	Decode logic PAL	Control memory	Indicates memory mapped access to control memory
$\overline{\text{EMI/OSEL}}$	J1-47	Decode logic PAL	DMA and I/O control PAL, DMA logic	Indicates memory mapped access to emulated I/O device instruction
$\overline{\text{86SEL}}$	J1-49	Decode logic PAL	8086 control logic	Indicates memory mapped access to 8086 processor logic
GRACYC	J1-46	Video interface	Timing logic, master multiplexor, diskette drive controller	Indicates current cycle is a graphics or refresh memory access
$\overline{\text{PUKARD}}$	J1-50	DMA and I/O control PAL	System console keyboard interface	When low, enables system console keyboard read
$\overline{\text{PUKAWR}}$	J1-51	DMA and I/O control PAL	System console keyboard interface	When low, enables write from control memory to keyboard of LED/bell status
KBIRQ	J1-59	System monitor keyboard	CPU status register	Indicates that system console keyboard is requesting service

[1] *See also* $\overline{\text{BRDY}}$.

Table 4-2 microECLIPSE CPU Signals[1]

Name	Pin(s)	Description
$\overline{MB}$<0-15>	CPU17-CPU32, SIO17-SIO32, XMC12-XMC29	CPU address/data, bidirectional
$\overline{ADREN}$[2]	CPU16, SIO16, XMC30	Address enable; carried on the system memory bus
$\overline{DATEN}$	CPU13, SIO12	Data enable
$\overline{WH}$, $\overline{WL}$	CPU11, 12, SIO10, 11, XMC31($\overline{WL}$), B29, 30	Write operation to high or low order bytes requested; carried on the system memory bus
ABLOCK	CPU10	Indivisible operation in progress
MEMCYC[3]	CPU15, SIO15, XMC32	Memory cycle is asserted (otherwise I/O cycle); carried on system memory bus
$\overline{INTRQ}$	CPU46, SIO9	I/O or control program interrupt
CR<0-7>	XMC1-XMC3, XMC39-XMC35, CPU42-CPU34	Microinstruction from XMC to CPU
BREQ	CPU47	From bus arbitration logic indicating to the microECLIPSE CPU that another controller needs the system bus
MPH<1,2>	CPU43, 44, SIO43, 44	2-phase non-overlapping clock for CPU, SIO
CPH<1,2>	XMC33, 34	2-phase non-overlapping clock for XMC
SIO Signals[1]		
EBLOCK	SIO47	From bus arbitration prom indicating another device has control of the bus
EBREQ	SIO8	When high, indicates that the SIO controller requires the system bus
ENMI	SIO4	When high, indicates that the SIO is requesting a nonmakable interrupt from the microECLIPSE CPU
$\overline{IOD<1,2>}$	SIO33, SIO34	Serial data to or from the microI/O bus
$\overline{IOCLK}$	SIO36	Synchronizing signal for microI/O bus serial data transfers
$\overline{IONP}$	SIO35	Controls direction of microI/O bus interface transfer: high in, low out
PFAIL	SIO46	Inverted $\overline{PF}$ from microI/O bus
SDO	SIO2	Source of $\overline{TTOUT}$
SDI	SIO41	Inverted $\overline{TTIN}$
CTS	SIO40	Inverted CTS from TTI interface
8086 CPU Signals		
$\overline{86AD}$<0-15>	86P2-86P16, 86P39	CPU address/data, bidirectional
$\overline{86AD}$<16-19>	86P35-86P38	CPU address high-order bits
$\overline{ALE}$	86P25	8086 address latch
$\overline{DEN}$	86P26	8086 data enable
DT/$\overline{R}$	86P27	8086 data transfer direction
$\overline{BHE}$	86P34	Enables data onto the most significant half of the 8086 data bus during the address phase

Table 4-2 microECLIPSE CPU Signals[1] (Continued)

Name	Pin(s)	Description
$\overline{\text{86WR}}$	86P29	8086 write strobe
M/$\overline{\text{I O}}$	86P26	Indicates 8086 memory (as opposed to I/O) reference
86RDY	86P22	Synchronized microECLIPSE READY signal
$\overline{\text{86RESET}}$	86P21	Causes 8086 processor to immediately terminate its present activity
$\overline{\text{86HOLD}}$	86P31	Requests 8086 to relinquish the system bus
$\overline{\text{86HLDA}}$	86P30	Acknowledges that the 8086 processor has relinquished the bus
$\overline{\text{86OUT}}$	—	When low, indicates that the 8086 processor has decoded an **OUT** instruction
86VP	—	Indicates that the 8086 processor has accessed validity-protected memory
$\overline{\text{ATPDOC}}$	—	When low, indicates that the microECLIPSE processor has issued an **ATP DOC** (*Initiate Attached Processor Page Check*) instruction
AREQ	—	Indicates an 8086 processor interrupt condition
Video Interface Signals		
GRA<0,1,6-15>, LOCAT	—	Resident main memory refresh addresses
GRA<2-5>	—	Outputs from CRT controller's internal row address counter; used for most significant screen buffer address bits
READ	CRTC22	Determines direction of data flow into (low) or out (high) of the video interface controller
VS	CRTC40	Determines vertical position of displayed text (source of $\overline{\text{VSYNC}}$)
HS	CRTC39	Determines horizontal position of displayed text (source of **HSYNC**)
$\overline{\text{DOTS}}$	—	Raw dot output of video shift register
$\overline{\text{DOTCLK}}$	—	Clock signal for individual pixels
$\overline{\text{DOTLD}}$	—	When low, loads a byte from the graphics screen buffer into the video shift register
WINK	—	Determines the video blink rate
$\overline{\text{BLINK}}$	—	Character attribute signal: when low, blink current character
$\overline{\text{REV}}$	—	Character attribute signal: when low, show current character in reverse video
$\overline{\text{USC}}$	—	Character attribute signal: when low, underscore current character
$\overline{\text{DIM}}$	—	Character attribute signal: when low, dim current character
CURSOR	CRTC19	Indicates valid cursor address to external display logic
CURSE	—	CURSOR latched by $\overline{\text{DOTCLK}}$
DISPEN	CRTC18	Indicates that video interface controller is providing addresses in the active display area
BLANK	—	DISPEN latched by $\overline{\text{DOTCLK}}$

Table 4-2 microECLIPSE CPU Signals[1] (Continued)

Name	Pin(s)	Description
$\overline{\text{ABEN}}$	—	When low, enables the blink character attribute
$\overline{\text{CBEN}}$	—	When low, enables the cursor
AT2MB	—	Enables attribute RAM read operations
$\overline{\text{RAW}}$	—	When low, enables write attribute RAM write operations
Diskette and DMA Controller Signals		
FDB<0-7>	DMA21-DMA23, DMA26-30	Bidirectional multiplexed memory address(high-order)/data for DMA transfers
DAB<0-7>	DMA32-40	Low-order memory address bits for DMA transfer (DAB<0-3> are also used as register-select inputs)
WDA	FDC30	Serial clock and data bits to the diskette drive
RD-PLS	FDC23	Clock and data bits from the diskette drive
WCK	FDC21	Write clock for the diskette drive (enabled for both read and write)
WG	FDC25	Enables write data to the diskette drive
DRQ	DMA19, FDC14	Indicates that diskette controller is requesting a transfer
DMAHRQ	DMA10	Indicates that DMA controller is requesting control of the system bus
DMAHLDA	DMA7	Indicates that DMA controller has been granted control of the system bus
$\overline{\text{DACK}}$	DMA25, FDC15	Asserted by DMA controller to allow diskette controller to perform a transfer
TC	DMA36, FDC16	Issued by DMA controller to tell diskette controller to terminate transfer operation
HIP1	DMA13, FDC1	Resets the DMA and diskette controllers
$\overline{\text{DI/OR}}$,$\overline{\text{DI/OW}}$	DMA1, DMA2	Control outputs that are gated to become read and write control signals for the diskette controller (also used as control signals for programming the DMA controller)
$\overline{\text{I/OR}}$,$\overline{\text{I/OW}}$	FDC2, FDC3	Gated $\overline{\text{DI/OR}}$ and $\overline{\text{DI/OW}}$.
$\overline{\text{MEMRD}}$	DMA3	Goes low during DMA read cycles, used to read data from the addressed memory location
$\overline{\text{MEMWRT}}$	DMA4	Goes low during DMA write cycles, used to extend memory write accesses.
DMADSTB	DMA8	Clocks the most significant byte of the memory address into a logical address bus latch
NEC765INT	FDC18	Indicates to the diskette controller that the diskette drive needs service
FRDY	FDC35	Indicates to the diskette controller that the diskette drive is ready to send or receive
IDX	FDC17	Indicates to the diskette controller that the diskette drive is at the beginning of a disk track
PS<0,1>	FDC31, FDC32	Write precompensation signals for the diskette drive
RDW	FDC22	Data window signal generated by phase-locked loop

Table 4-2 microECLIPSE CPU Signals[1] (Continued)

Name	Pin(s)	Description
RG	FDC24	Control signal for phase-locked loop
HD	FDC27	Head select signal for diskette drive
SEEK/$\overline{RW}$	FDC39	Low state indicates diskette read/write mode selected;high state indicates seek mode
LCT/DIR	FDC38	Controls current level/direction in read-write/seek modes
FR/STP	FDC37	Resets fault FF/steps head in read-write/seek modes
WP/TS	FDC34	Indicates write protect/two-sided media in read-write/seek modes
FLT/TR0	FDC33	Indicates fault/track 0 condition in read-write/seek modes
DMADATEN	—	Data enable signal for DMA system bus transfers
$\overline{DMACYC}$	—	When low, indicates DMA address phase
$\overline{FCEN}$	—	When low, latches memory-mapped diskette control word on $\overline{BMB}$<08-15>
$\overline{HIB}$	—	When low, indicates current DMA byte is most significant (high) byte
Memory Signals[1]		
$\overline{DIN}$<0-15>	—	Data input to resident main memory
$\overline{DOUT}$<0-15>	—	Data output from resident main memory
$\overline{0POD}$<L,H>	—	Parity input to resident main memory, bank 0
$\overline{1POD}$<L,H>	—	Parity input to resident main memory, bank 1
$\overline{DOPE}$<L,H>	—	Parity output from resident main memory
0RA<0-7>	—	Resident main memory bank 0 row address
1RA<0-7>	—	Resident main memory bank 1 row address
$\overline{KRAS, OPTRAS}$	—	Resident main memory row address strobes
$\overline{KCAS}$	—	Resident main memory kernal (bank 0) column address strobe
$\overline{OPTCAS}$	—	Resident main memory optional (bank 1) column address strobe
$\overline{CAS}$	—	Resident main memory column address timing pulse (OR of $\overline{KCAS}$ and $\overline{OPTCAS}$
$\overline{RAM-WL}$	—	Resident main memory low byte write enable
$\overline{RAM-WH}$	—	Resident main memory high byte write enable
Bus Arbitration Signals		
BREQ	CPU47	Requests microECLIPSE CPU to relinquish bus
HOLD	86P31	Requests 8086 CPU to relinquish bus
HLDA	86P30	Indicates 8086 CPU has relinquished bus
$\overline{86BREQ}$	CPU47	Indicates a pended transfer of control to the 8086 processor
$\overline{EGRANT}$	—	Grants the SIO device control of the system bus
DMAHLDA	—	Gives the DMA controller control of the system bus
EBLOCK	—	Prevents the SIO device from accessing the system bus

Table 4-2 microECLIPSE CPU Signals[1] (Continued)

Name	Pin(s)	Description
System Timing Signals (see figure 4-8)[1]		
CLK<0-11,21,70>	—	Outputs of system clock generator
DMA and I/O Control Signals		
$\overline{\text{FLOPH}}$	—	Enables diskette drive high byte transceiver
$\overline{\text{FLOPL}}$	—	Enables diskette drive low byte transceiver
TOMB	—	Specifies direction of transfer of high and low byte transceivers in diskette interface (high: towards memory)
$\overline{\text{DMA-WH}}$	—	Sources $\overline{\text{SYSWH}}$ during a DMA cycle
$\overline{\text{DMA-WL}}$	—	Sources $\overline{\text{SYSWL}}$ during a DMA cycle
MAP and Master Multiplexor Signals[1]		
LM<0-9>	—	Selected address bits output from MAP multiplexor
PA<0-9>	—	Physical address bits output from MAP RAM
MAP<A,B>	—	Selected user map
MAPSEL	—	During a CPU bus data cycle, indicates an access to the MAP RAM
HYPER	—	When high, enables access to control memory (RAM or ROM)
GMUX	—	When high, selects screen buffer or refresh address to be used for main memory row addresses
CMUX	—	When high, selects physical addresses produced by MAP RAM to be used for main memory row addresses
Parity Signals		
$\overline{\text{PINE}}$	—	When low, enables parity fault interrupts
$\overline{\text{POKE}}$	—	For diagnostic use: when low, causes sense of parity bits written to main memory to be reversed
Memory-Mapped I/O Decode Signals[1]		
$\overline{\text{MAPS}}$	—	When low, indicates memory-mapped access to MAP RAM
$\overline{\text{NMIS}}$	—	When low, indicates memory-mapped access to the CPU status register
$\overline{\text{AUXSEL}}$	—	When low, indicates memory-mapped access to the CPU interrupt flag
$\overline{\text{LMSSEL}}$	—	When low, indicates memory-mapped access to the MAP status register
$\overline{\text{RAMSEL}}$	—	When low, indicates memory-mapped access to control RAM (includes video screen buffer and RAM used by code in control ROM)
Miscellaneous Signals		
SSNMI	—	Set high by control memory to cause a nonmaskable interrupt (used for virtual console single-stepping feature)

[1] *See also* Inter-SPU-Board Signals in Table 1-2.
[2] *See also* $\overline{\text{BUSADREN}}$.
[3] *See also* $\overline{\text{BUSMCYC}}$.

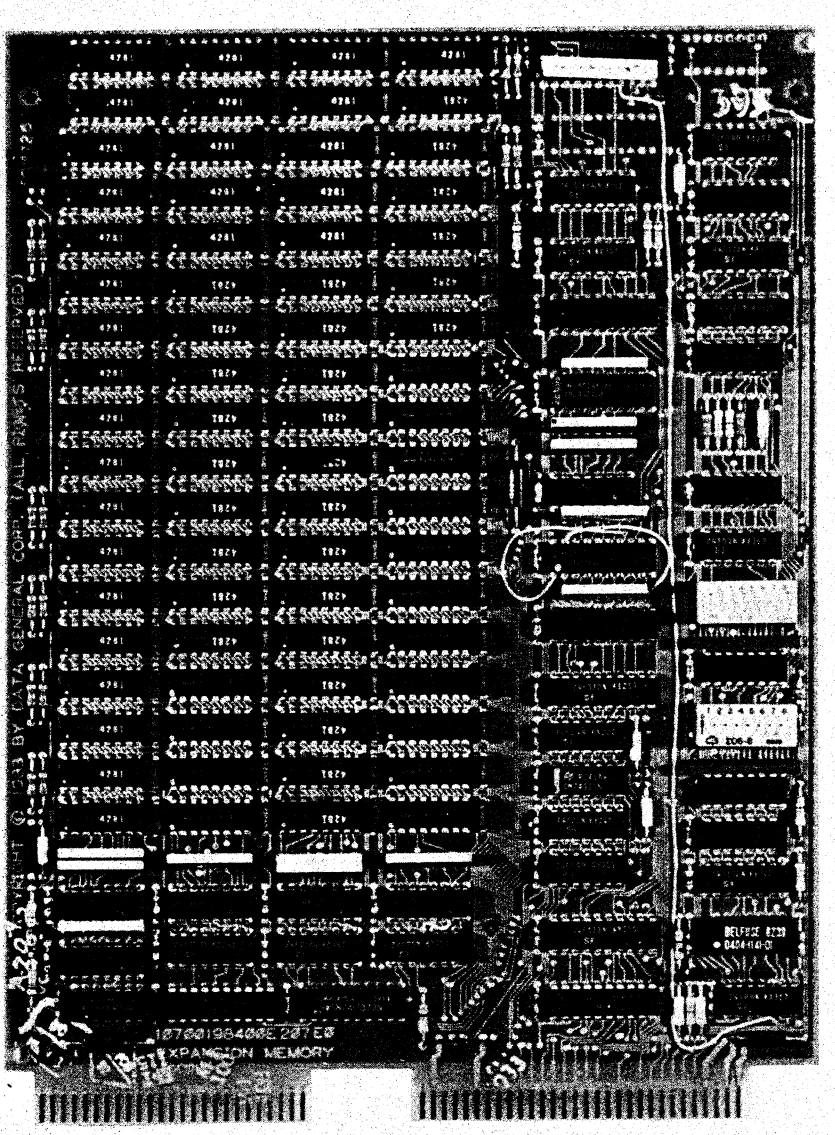


Figure 5-1 Models 10 or 10/SP memory card

Optional Memory Card

5

This chapter describes the theory of operation for the optional memory card.

The optional memory card, shown in Figure 5-1, provides 256 or 512 Kbytes of dynamic MOS random-access memory (RAM) for the Model 10 and 10/SP processor. All memory cards use 64K by 1-bit RAM integrated circuits (ICs). Two parity bits are associated with each 16-bit data word in memory: one parity bit with the low-order byte and another with the high-order byte. Thus, a memory location is 18 bits wide, requiring 18 RAM ICs. The 256-Kbyte memory card contains two banks of RAM ICs (36ICs); the 512 Kbyte memory card contains four banks of RAM ICs (72 ICs).

The optional memory card must be installed in the main system unit's SPU logic module (SLM) cage. The RAM card is positioned in the system's memory space by means of dual-inline-position (DIP) switches. The card has independent controller logic for generating its own timing and refresh operations.

A memory operation either reads the 16-bit word of the addressed memory location or writes the high-order byte, the low-order byte, or both bytes of a word. A parity bit supplied by the CPU is read or written with each byte.

The memory card features a light emitting diode (LED), which lights when a memory reference to an address on the card is in progress.

Table 5-1 summarizes the characteristics of the RAM cards.

Table 5-1 Characteristics of the optional memory card

Memory Type:	Dynamic MOS n-channel RAM
Card Capacity:	256 Kbytes or 512 Kbytes, with parity
Cycle Time	
Read:	500 nanoseconds
Write:	500 nanoseconds

Installation and Jumpering

Table 5-2 gives the power requirements of each memory card.

Table 5-2 Power requirements of each RAM card

Supply Voltage V dc	Current	Power Dissipation (W)
+5	1.9 not selected	9.7
	2.2 selected	11.2

The memory card contains two DIP switches that allow you to position the memories in various blocks of the memory space. Normally, RAM is added to a system from the bottom (address 0) of memory up.

Refer to "Memory Card Tailoring" in *Installing Model 10 and Model 10/SP Systems* (DGC No. 014-000911) for information on switch settings for configuration of the memory card.

Interfacing

The Model 10 and Model 10/SP memory card communicates with the CPUs and data channel controllers through the B edge connector. This connector also provides all power to the memory. (The memories do not use the A connector.) Figure 5-2 shows the RAM card B connector position. Figure 5-3 shows the pin assignments.

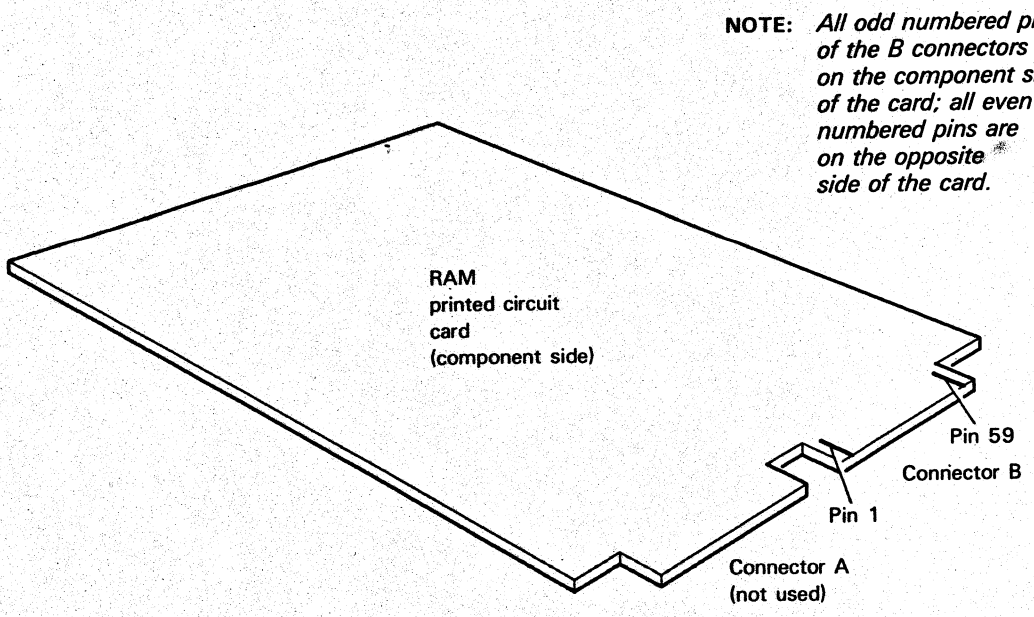


Figure 5-2 Models 10 or 10/SP memory card connector positions

Even	Signal Names		Odd
60	+5V	+5V	59
58	−5V	+5V	57
56	+12V	+12V	55
54	GND	GND	53
52	SYSCLOCK	XMA1	51
50	DATA0	DATA8	49
48	BUSADREN	XMA0	47
46	DATA1	DATA9	45
44	XMA2		43
42	DATA2	DATA10	41
40	XMA3		39
38	DATA3	DATA11	37
36	GND		35
34	DATA4	DATA12	33
32	DATA5	DATA13	31
30	WH/LPARH	WL/PARL	29
28	DATA6	DATA14	27
26	BUSMCYC	XMA4	25
24	DATA7	DATA15	23
22			21
20			19
18			17
16			15
14	GND		13
12		GND	11
10			9
8			7
6			5
4		GND	3
2			1

ID-00625

Figure 5-3 Pin assignments, B connector

Theory of Operation

The main component of the RAM cards is a RAM array containing two or four banks of memory ICs. Each memory bank contains eighteen 64K-by-1-bit RAM ICs to provide 65,536 consecutive 18-bit memory locations. These locations are assigned to ECLIPSE Model 10 and Model 10/SP system memory locations by the card select switches.

Figure 5-4 shows the interconnection of the RAM array and the other components that comprise the RAM cards, as well as the various signals that control the flow of data between the memory bus and the RAM array. The descriptions that follow relate to Data General logic schematic No. 001-003673.

Initiating a Memory Operation

The memory card continually receives data from the processor's memory bus via its B connector. At the start of a memory operation, the bus controller drives $\overline{\text{MCYC}}$ and $\overline{\text{ADREN}}$ low and $\overline{\text{BDATA0}}$ high (**BDATA0** = 0), and sends a 20-bit physical address over the memory bus. The address consists of $\overline{\text{BDATA<1-15>}}$ and $\overline{\text{XMA<0-4>}}$. When $\overline{\text{BDATA0}}$ is high (referencing user memory space), the card/bank select logic of the memory card decodes address bits $\overline{\text{BDATA1}}$ and $\overline{\text{XMA<0-4>}}$. If the card is configured to contain the memory range specified by those bits, the card/bank select logic chooses the appropriate bank and notifies the memory card controller by driving **SELECT** high. On 256-Kbyte memory cards, signals $\overline{\text{XMA<0-2>}}$ select the card, while signal $\overline{\text{XMA3}}$ selects one of the two banks of RAM ICs. On 512-Kbyte memory cards, signals $\overline{\text{XMA<0-1>}}$ select the card, while signals $\overline{\text{XMA<2-3>}}$ select one of the four banks of RAM ICs.

The memory card controller initiates a memory operation as soon as it receives $\overline{\text{ADREN}}$, provided that the addressed location is within the selected memory range and $\overline{\text{MCYC}}$ is low. When the controller drives **ADRLATCH** and **R/WLATCH** high in response to $\overline{\text{ADREN}}$, the bank select logic latches the memory bank select line ($\overline{\text{SEL0}}$, $\overline{\text{SEL1}}$, $\overline{\text{SEL2}}$, or $\overline{\text{SEL3}}$); then the row and column address registers latch the row address ($\overline{\text{XMA4}}$ $\overline{\text{BDATA<1-8>}}$) and the column address ($\overline{\text{BDATA<8-15>}}$), respectively, and send them to the row/column address multiplexor. At this point, the controller begins a read or write operation.

Row and Column Address Selection

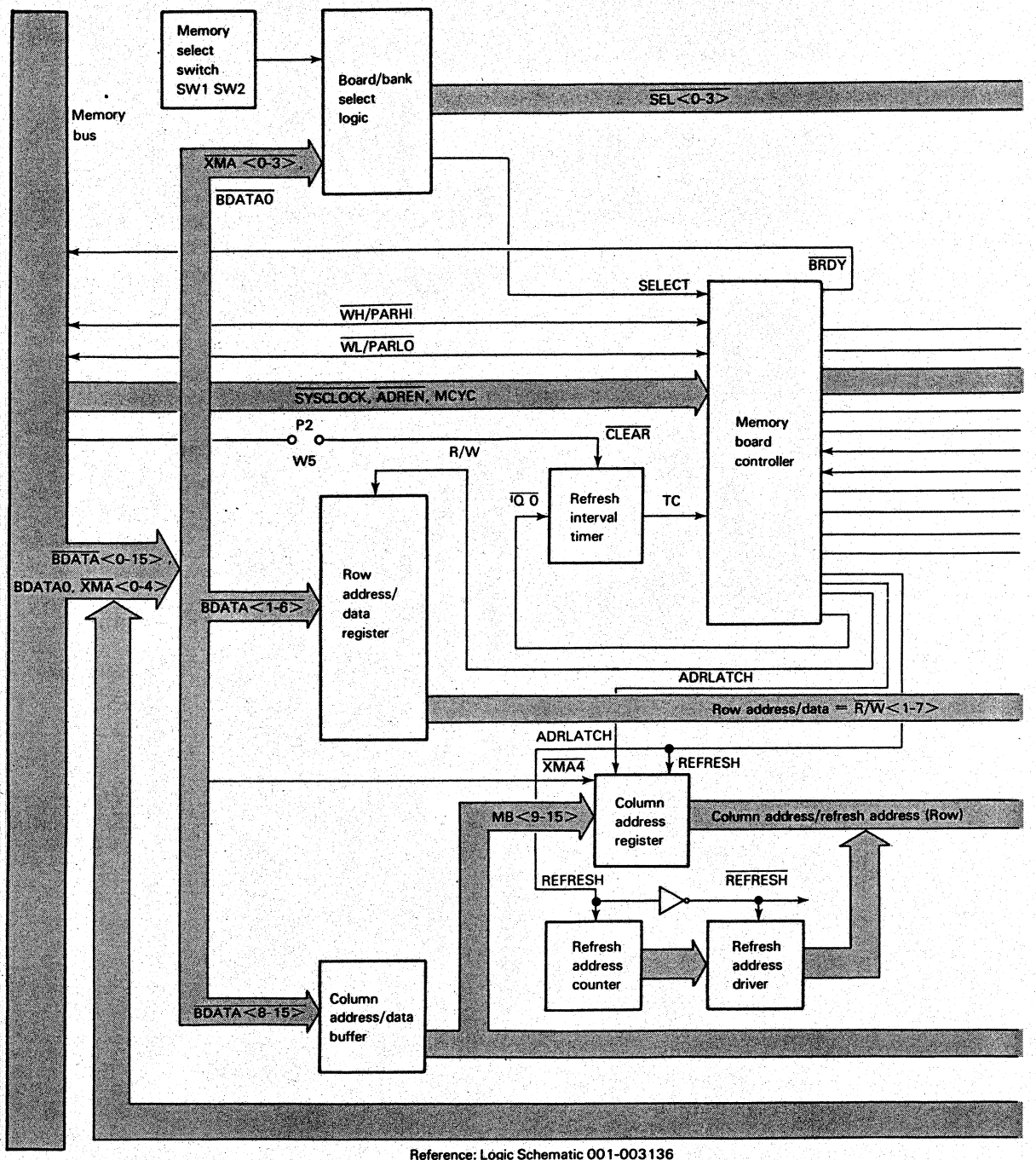
As soon as the address multiplexor receives the address, it sends the row address bits $\overline{\text{XMA 4}}$ and $\overline{\text{BDATA<1-8>}}$ to the RAM array via the $\overline{\text{MAD<1-8>}}$ lines. The controller then asserts **RASEN**. At the same time, the row/column enable logic drives the row address strobe of the selected memory bank ($\overline{\text{RAS0}}$, $\overline{\text{RAS1}}$, $\overline{\text{RAS2}}$, or $\overline{\text{RAS3}}$) low, latching the row address into the appropriate RAMs.

When the controller drives **COLSEL** high, the address multiplexor sends the column address bits ($\overline{BDATA<8\text{-}15>}$) to the RAM array via the $\overline{MAD<0\text{-}8>}$ lines. Then the controller asserts **CASEN**, causing the LED on the memory card to light up. Simultaneously, the row/column enable logic drives the column strobes of all the memory banks ($\overline{CAS0}$, $\overline{CAS1}$, $\overline{CAS2}$, and $\overline{CAS3}$) low, latching the column address into the RAM array. The assertion of **CASEN** causes the LED on the memory card to light up.

Read

If $\overline{WH}$ and $\overline{WL}$ are both high (unasserted) while $\overline{ADREN}$ is driven low during the address phase of the memory operation, the memory board initiates a read operation. After the row and column addresses are multiplexed to the RAM array, the selected RAM bank outputs the contents of the addressed location. The data bits are placed on the read data bus ($\overline{RD<0\text{-}15>}$) and the two parity bits associated with the data are sent to the controller.

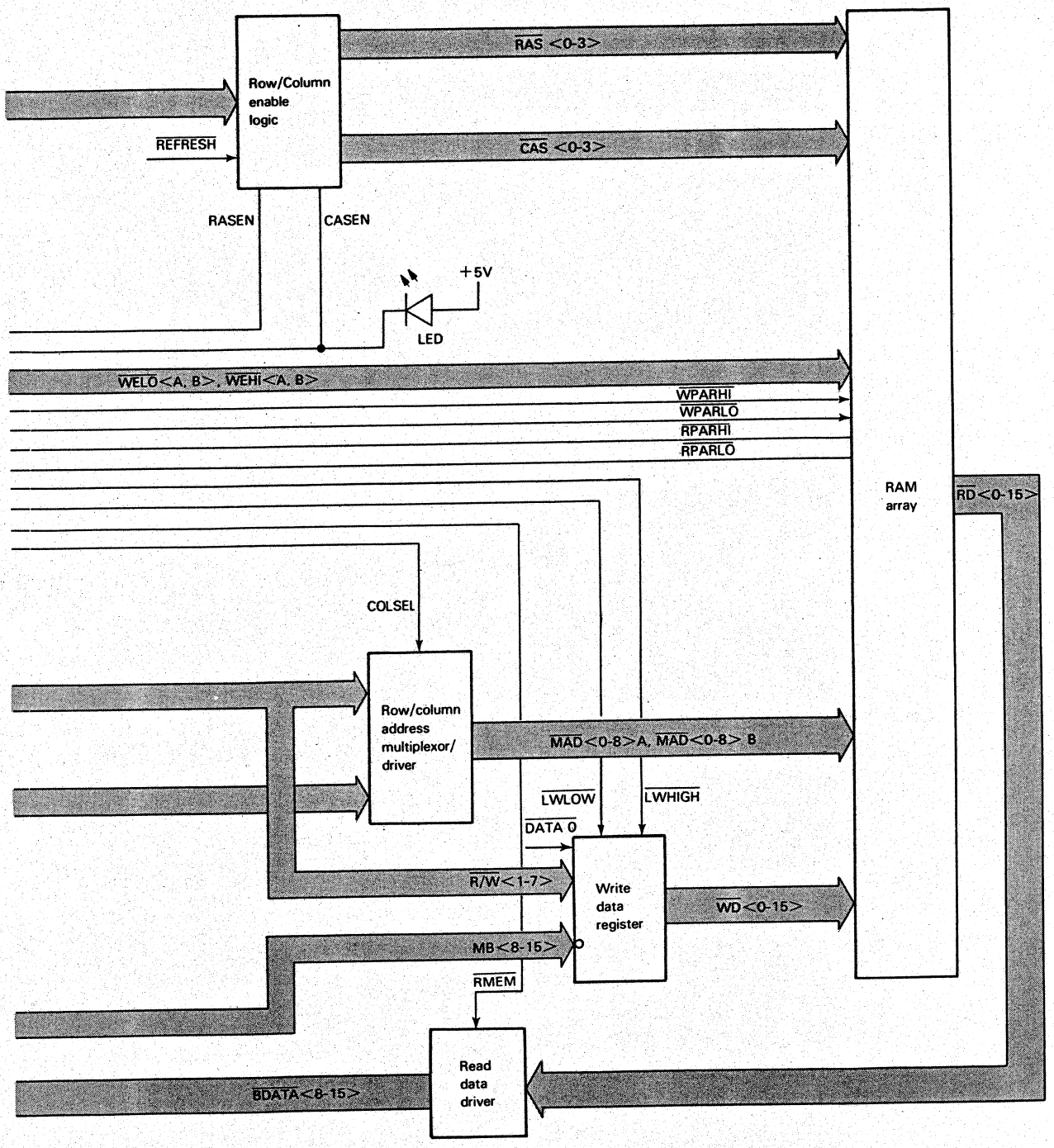
The controller drives $\overline{RMEM}$ low, gating the data onto the memory bus. The controller then transmits the parity bits of the high-order byte and the low-order byte via the $\overline{WH}$ and $\overline{WL}$ lines, respectively. (Note that $\overline{WH}$ and $\overline{WL}$ are multiplexed lines that transmit parity bits during the data phase of the memory bus cycle.) Figure 5-5 shows a read timing diagram.



Reference: Logic Schematic 001-003136

DG-08318

Figure 5-4 Memory card block diagram


Row/Column enable logic
REFRESH
RAS <0-3>
CAS <0-3>
RASEN
CASEN
+5V
LED
WELO<A, B>, WEHI<A, B>
WPARHI
WPARLO
RPARHI
RPARLO
RAM array
RD<0-15>
COLSEL
Row/column address multiplexor/ driver
MAD<0-8>A, MAD<0-8> B
LWLOW
LWHIGH
DATA 0
R/W<1-7>
Write data register
WD<0-15>
MB<8-15>
RMEM
Read data driver
BDATA<8-15>

Write

If $\overline{WH}$ and/or $\overline{WL}$ are low during the address phase while $\overline{ADREN}$ is asserted, the controller initiates a byte or word write operation; signals $\overline{LWHIGH}$ and/or $\overline{LWLOW}$ are driven low, depending on the states of $\overline{WH}$ and $\overline{WL}$. During the data phase of the memory write operation, the write data register receives data from three sources: the column address/data buffer transmits the low-order byte ($\overline{BDATA<8\text{-}15>}$) as $\overline{MB<8\text{-}15>}$; the row address/data register transmits seven bits of the high-order byte ($\overline{BDATA<1\text{-}7>}$) as $\overline{R/W<1\text{-}7>}$; and the card circuitry transmits bit 0 as $\overline{BDATA0}$.

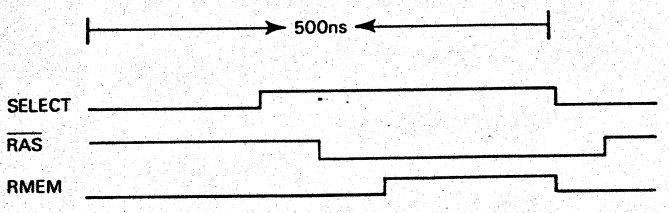
The write data register drives the appropriate byte(s) of data onto the $\overline{WD<0\text{-}15>}$ lines, inverting bits $\overline{MB<8\text{-}15>}$. The controller transmits the parity bit(s) to the RAM array by driving signals $\overline{WPARLHI}$ and/or $\overline{WPARLO}$ high or low, depending on the states of $\overline{WH}$ and $\overline{WL}$. The controller then issues a write enable pulse — $\overline{WELO}$ for low-order bytes and $\overline{WEHI}$ for high-order bytes — which writes the byte or word with its associated parity bit(s) into the addressed RAM location. Figure 5-6 shows a write timing diagram.

Refresh

The refresh interval timer asserts the **TC** signal every 15.6 microseconds. This signal causes the controller to initiate a refresh operation, provided it is not already busy with a read or write operation. If the controller is busy, it waits until the current operation is finished and then immediately starts a refresh operation.

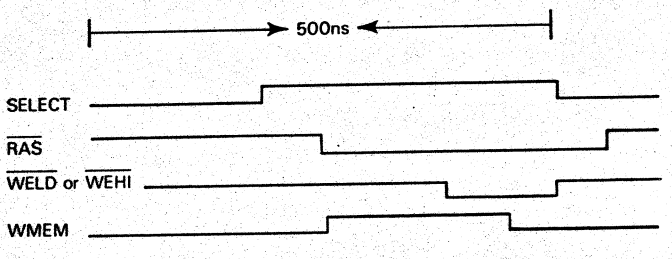
When a refresh operation is initiated, the controller signals the refresh address driver to supply the refresh address; this is the address specified by the refresh address counter to the RAM array through the address multiplexor. The controller asserts $\overline{REFRESH}$, enabling all rows of the RAM array ($\overline{RAS<0\text{-}3>}$). The controller then strobes the refresh address onto $\overline{MAD<0\text{-}8>}$ by driving **COLSEL** high. (Note that although the card logic uses column address lines to drive the refresh address, the RAM array sees the address input as a row address.) The controller does not send any column address because a refresh operation recharges every word of the selected row; thus, no column address is required. After the operation, the controller increments the refresh address counter.

In case of conflict between a memory read/write and a refresh operation, that is, if the **TC** and **SELECT** signals are asserted simultaneously, the controller executes the memory read/write operation first and the refresh operation second. If **SELECT** is asserted during a pended refresh (a refresh preceded by a memory operation), the controller latches the incoming address and drives $\overline{BRDY}$ high to extend the data transfer phase of the CPU cycle by one T period. After the refresh operation is completed, the controller performs a read or write operation to the addressed memory location. This process is known as a pended memory cycle. Figure 5-7 through Figure 5-9 show a normal refresh, a pended refresh, and a pended memory cycle, respectively.



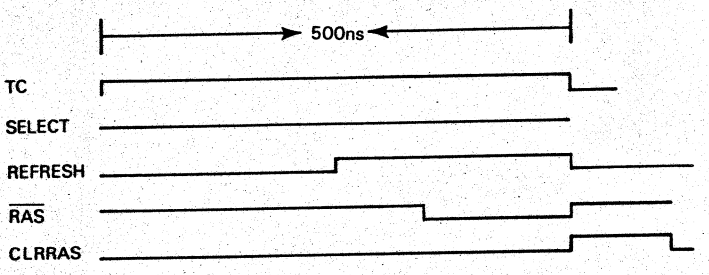
DG-08319

Figure 5-5 Read timing



DG-08320

Figure 5-6 Write timing



DG-08321

Figure 5-7 Refresh timing

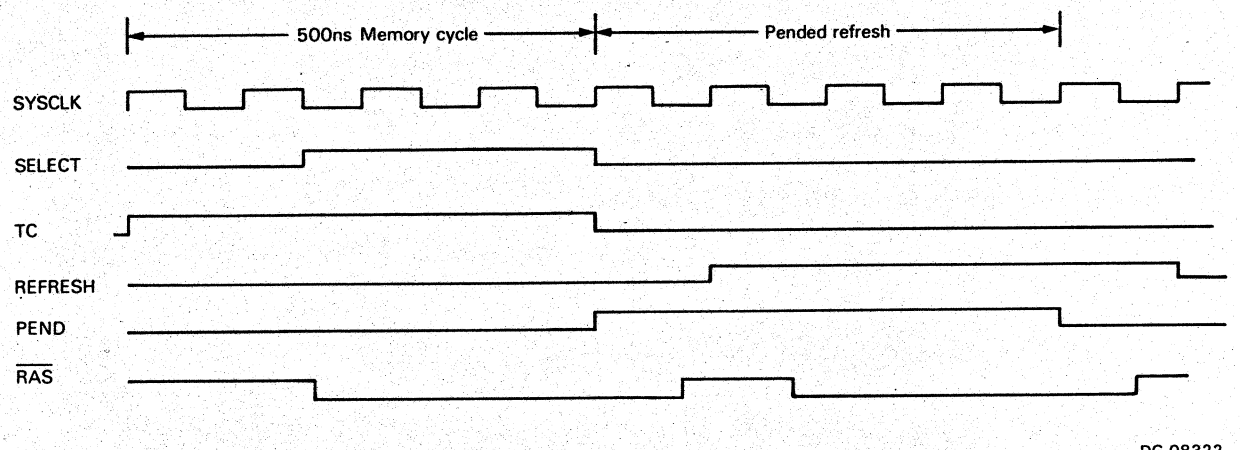


Figure 5-8 Pended refresh timing

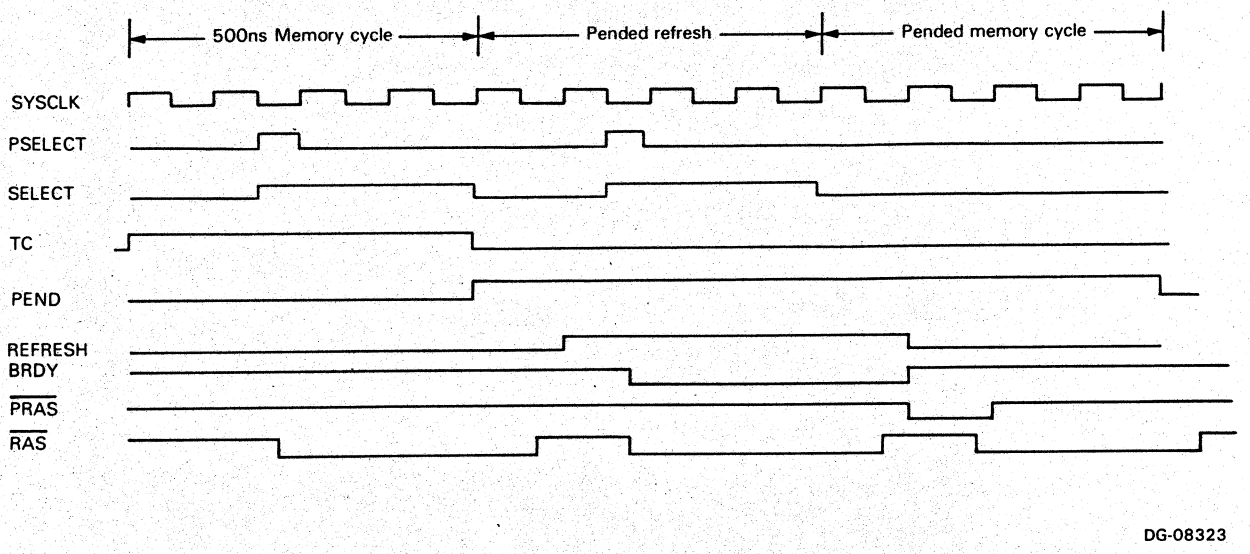


Figure 5-9 Pended memory timing

Optional Video Interface 6

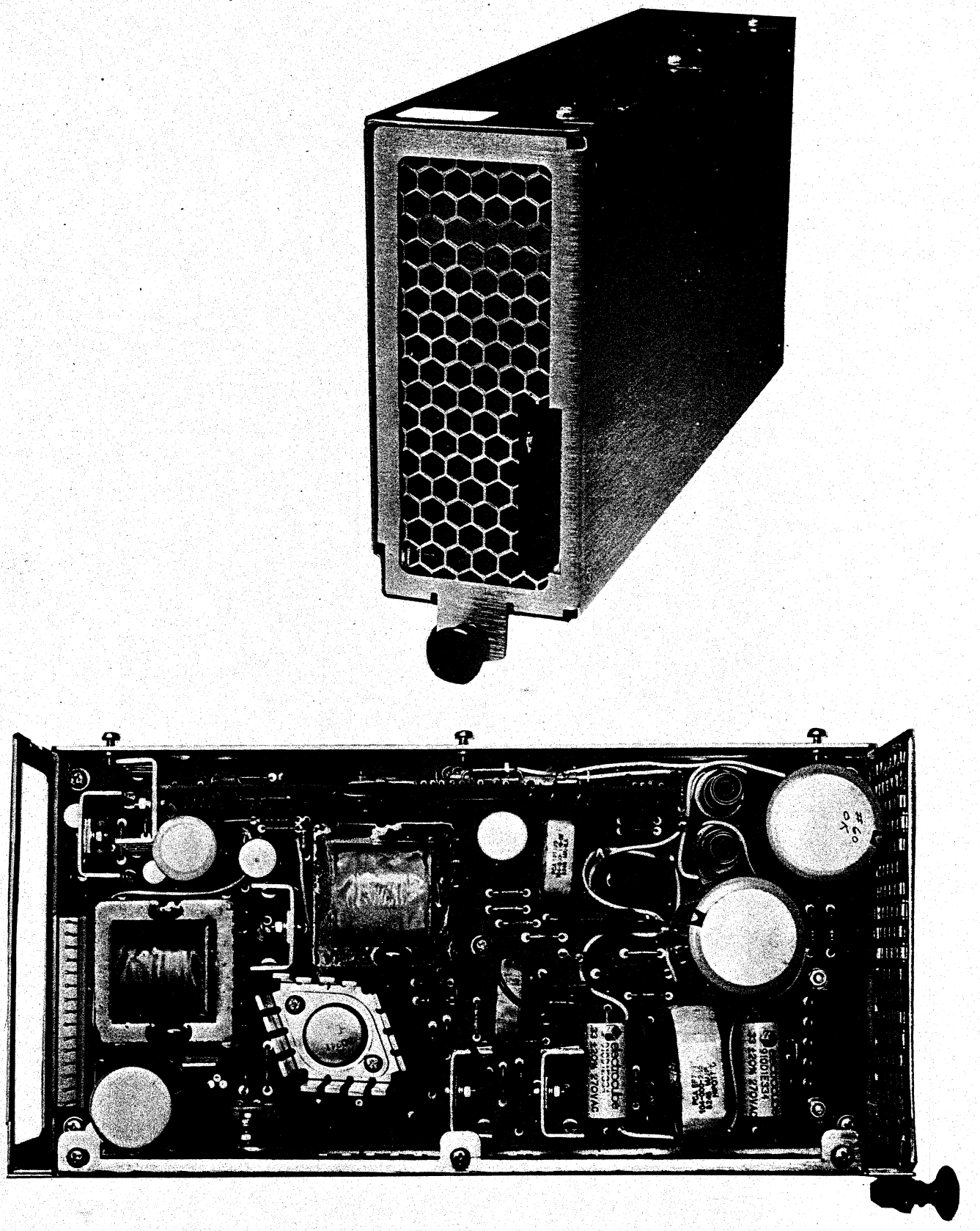
Text to be supplied.

Power Supply Assembly

7

A Model 10 and Model 10/SP power supply assembly consists of a single 4.75-inch by 10.0-inch printed circuit card contained in an EMI-shielded case, as shown in Figure 7-1. The power supply assembly mounts in a compartment within the power supply module (PM). It receives ac power through a line cord, line fuse, interlock, power switch, and internal ac power cable. It provides dc power and status signals to the backplane of the CPU logic module by means of an internal dc power cable.

This chapter describes ac input and dc output specifications of the power supply, explains its theory of operation, and demonstrates its interconnection with the system.



PH-0717, 0718

Figure 7-1 Models 10 or 10/SP power supply

Theory of Operation

The 123-watt power supply is an off-line switching converter that offers high efficiency in a compact package. Portions of the power supply Control and Status circuitry are contained on a daughter-type printed circuit card, vertically mounted and soldered to the power supply card. Two identical power supply assemblies, a main and an auxiliary, can be mounted in a power supply module. The main supply powers the CPU logic module (CLM) and diskette module (FM), while the auxiliary supply powers the logic expansion module (LEM) and disk module (DM). Table 7-1 and Table 7-2 provide the ac input and dc output specifications for each power supply assembly.

Figure 7-2 shows that the supply converts incoming ac voltage to non-regulated high voltage dc, which feeds a high frequency step-down switching power circuit (pulse width modulated). The outputs of the switching power circuit are rectified and filtered to provide the required voltages. All four outputs are regulated and current-limited. The +12, −12V, and −5V outputs are controlled by series pass regulators and are self-protected by the regulator current limits. The +5V output is regulated by a pulse width modulator controller and is current-limited on the primary side of the switching power supply. The +5V output is also protected against overvoltage.

The Control and Status daughter card of the power supply contains the circuitry to perform start-up directly off the nonregulated high voltage line, pulse width modulation, and +5V regulation. This card also contains circuitry that generates two power status signals (+5 OK and $\overline{\text{PWRFAIL}}$) for the processor. +5 OK specifies when the +5V output is above a specified limit, and $\overline{\text{PWRFAIL}}$ specifies when a power loss is imminent.

The Control and Status card also monitors the supply operation for internal undervoltage, overvoltage, and overcurrent conditions. Should one of these faults occur, the power supply will shut down temporarily and then automatically attempt to power up again, provided line power is present. If the fault condition is still present when the power supply attempts to power-up, the power supply detects the fault condition and immediately shuts down.

This section explains the operation of each functional portion of the power supply, as shown in Figure 7-2. Topics include the line rectification, start-up circuit, power section, auxiliary voltage, output, and status circuits. Reference designators such as U1 and T1 refer to logic schematic drawings DGC No. 001-003322 and DGC No. 001-003357. You may find it helpful to refer to these drawings while reading this chapter.

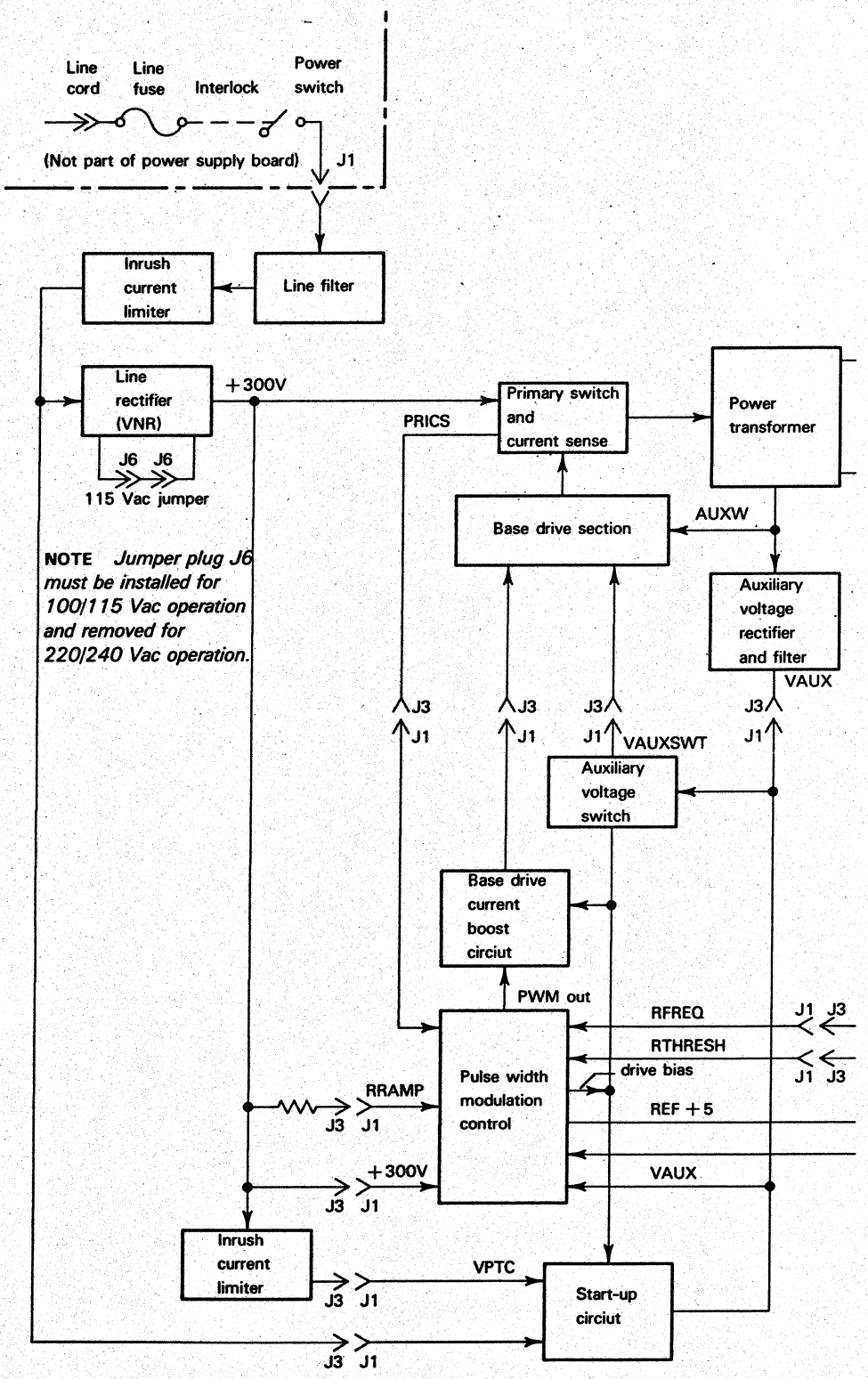
Table 7-1 AC input specifications (each power supply)

Voltage (Vac)	Frequency (Hz)	Input power (Watts)
85 to 115	47 to 63	180
97 to 132	47 to 63	180
187 to 264	47 to 63	180

Table 7-2 DC input specifications (each power supply)

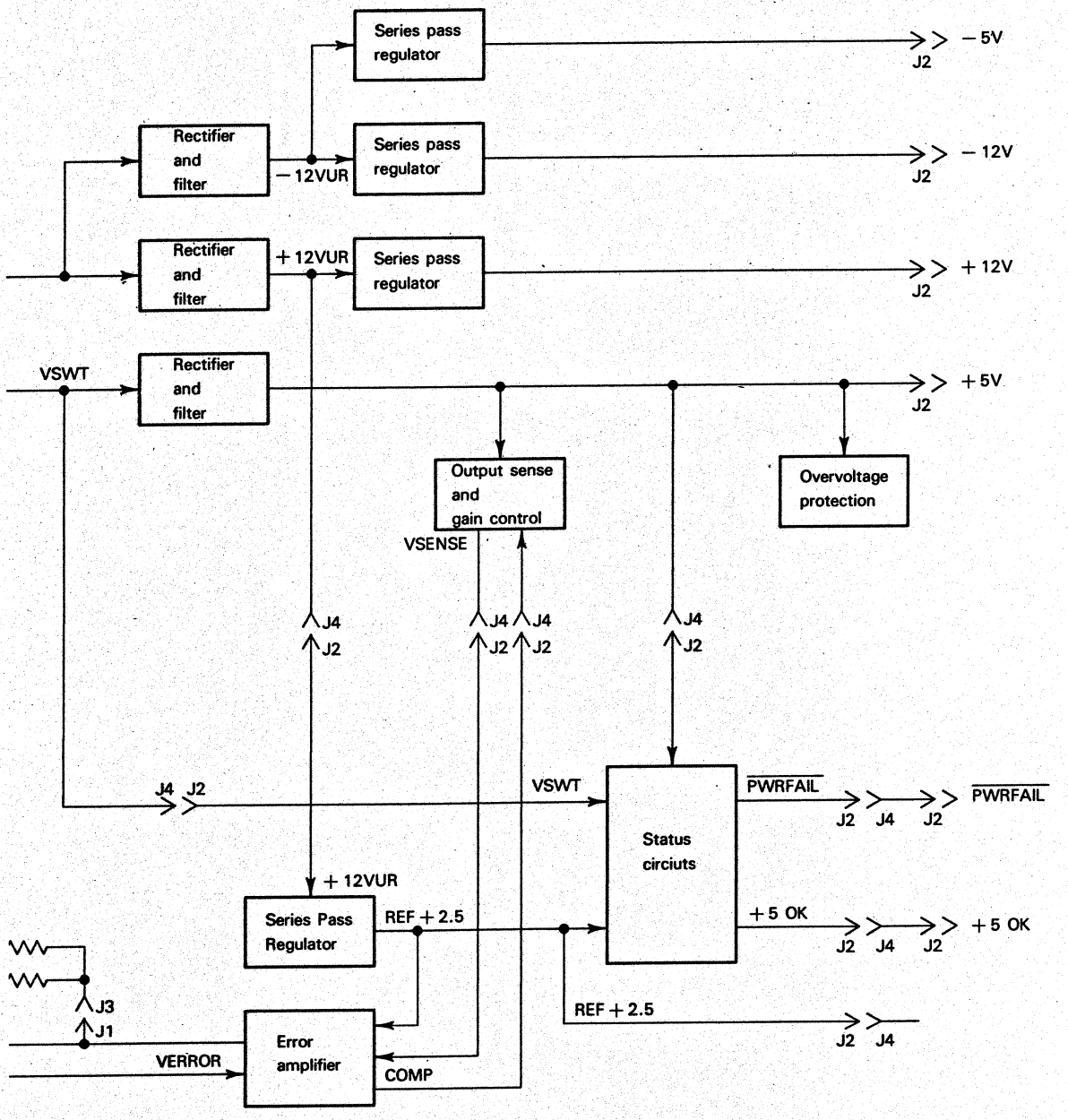
Output	Voltage (Vdc)	Ripple (mVp-p)	Current (Amps)
+5V	5.15 to 5.25	50	2.5 to 16.3
−5V	−4.75 to −5.25	50	0.05 to 0.5
+12V	11.5 to 12.6	50	0.25 to 2.5
−12V	−11.4 to −12.6	50	0.05 to 0.8

Reference: 001-003322 (Main Power Supply Board)



ID-00375

Figure 7-2 Models 10 or 10/SP power supply functional block diagram



Reference: 001-003357 (Control and Status Board)

Line Rectification

The ac line voltage passes through an inrush current limiter (thermistor), and a line filter to the line rectifier, where it converts to a non-regulated, high-dc voltage source. Although this voltage source ranges from 180 to 370 volts with the line voltage, the block diagram and engineering drawings refer to it as 300V. Thermistor RT1 limits the cold start inrush current to an acceptable value.

For 110/120 volt ac or 100 volt ac operation, the line rectifier rectifies and doubles the ac line source by connecting the junction of capacitors C5 and C6 (J6-1) to the neutral side of the power line (J6-3). In this way, the rectifier provides a 300V source. For 220/240 volt ac operation, the line rectifier simply rectifies the ac line source.

NOTE *Jumper plug connecting J6-1 (JMPB) to J6-3 (JMPA) must be installed for 100 Vac or 110/120 Vac operation and removed for 220/240 Vac operation.*

Start-Up Circuit

A start-up circuit, contained on the power supply Control and Status daughter card, provides the power needed to energize the pulse width modulation control section, thus energizing the power drive section. When the power supply receives ac line voltage, the voltage goes both to the line rectifier and to the start-up circuit, where it serves as a control signal. The high-dc voltage (+300V) provides energy (VPTC) for the start-up circuit through an inrush current limiter. The ac control signal turns on Q1, applying VPTC to the auxiliary voltage (VAUX).

VAUX drives the PWM controller integrated circuit (IC) internally, and the controller generates a regulated reference voltage (REF +5) for the power supply control circuitry.

VAUX also drives power to an auxiliary voltage switch (Q4 on the Control and Status card) that turns on when VAUX reaches approximately 25 volts. This switch (Q4) turns on when the PWM controller drive bias output turns on. The resulting voltage (VAUXSWT) drives the base drive section to provide magnetizing current during power supply operation. Also at this time, the PWM controller initiates pulse width modulation and turns off the start-up circuit. At this time VAUX begins to decay toward an undervoltage point. If the PWM controller begins pulse width modulation and the power supply successfully powers up, an auxiliary voltage section supplies VAUX, and VAUX remains above the undervoltage point. If start-up does not occur because the PWM controller fails to begin operation (drive bias output fails to turn on), VAUX, generated by the start-up circuit, will limit at approximately 33V (VR7 conducts, turning on Q2) to prevent damage to the control circuitry.

As stated earlier, when the power drive section turns on, a secondary winding of the power transformer and a rectifier supply the auxiliary voltage (VAUX). If start-up does not occur because of an error condition, VAUX will fall to an undervoltage fault, which initiates a restart attempt. Thermistor RT2 limits the energy dissipated during the period of start-up cycling. The power supply may attempt from 50 to 200 start-up cycles during any fault condition.

Power Section

The power section transforms nonregulated high dc voltage into high frequency, low-voltages. These voltages drive the auxiliary voltage section and the output

section. The power section also regulates the +5V output by means of a feedback loop. This section consists of:

Pulse width modulation section

Base drive section

Power drive section

Error amplifier

Pulse Width Modulation Section

This section consists of a pulse width modulator (PWM) controller IC and associated passive components that govern the operation of the power drive section (and, in turn, of the output section) by controlling the amount of power through the power transformer. It accomplishes this by causing the primary switch to close and open the power path to the power transformer at a typical 50 KHz rate and varying the duty cycle (ratio between the *closed* and *open* times) according to variations on the +5V output. As the ac line voltage decreases, or the +5V output load increases, pulse width modulation increases the *closed* time (drive cycle) to transfer more power to the output section. Similarly, as the ac line voltage *increases* or the +5V output load *decreases,* pulse width modulation decreases the *closed* time to transfer less power to the output section. In this way, the pulse width modulation regulates the + 5V output.

To prevent the power transformer from saturating, pulse width modulation is designed to operate at duty cycles of approximately 40 percent at the lowest line voltage. This section, with the exception of two bias resistors, is contained on the Control and Status daughter card. The PWM controller IC and associated components

provide a low current start-up,

establish the operation frequency of the power drive section,

provide a slow turn-on to prevent output-voltage overshoots by gradually increasing the power drive section *on* time from zero to the nominal time,

provide a +5V regulated reference voltage for the power supply control circuitry,

provide pulse width modulation control to the power drive section by means of the base drive current boost circuit and the base drive section (explained below),

perform pulse-by-pulse current-limiting of the power drive section.

During the drive portion of each power cycle, the output of the PWM controller floats, causing the base drive current boost circuit PWM signal to float. This causes the base drive section to turn off, which turns on the primary switch. When the PWM controller's internally generated ramp voltage reaches the buffered error signal VERROR, or its duty cycle ends, the controller's PWM output goes low, causing the base drive current boost circuit PWM output to go low. The base drive section then turns on, which turns the primary switch off and ends the power drive cycle. The controller also terminates a drive cycle by the same method if the power drive section current (PRICS) reaches an established threshold during a drive cycle (pulse-by-pulse current-limiting).

The PWM controller IC also includes fault-sensing circuitry to monitor

the high voltage bus (+300V) for undervoltage/overvoltage conditions,

the auxiliary voltage bus (VAUX) for an undervoltage condition,

the power drive section for overcurrent conditions (PRICS).

Should one of these faults occur, the PWM controller will shut down, which shuts down the power supply until the start-up circuit recycles. (See "Start-Up Circuit" in this chapter.) The start-up circuit automatically recycles when the PWM controller shuts down, provided line power is present. Of course, if the fault condition is still present when the PWM controller attempts to restart, the controller detects the fault and either fails to start operation or immediately shuts down, depending upon the fault.

Base Drive Section

The base drive section provides the control to turn the primary switch on and off. When the control card signal (PWM) floats, the base drive section causes the primary switch to turn on, closing the +300V path to the power transformer. When the control card signal goes low, the primary switch turns off, which terminates the power drive cycle. During the power drive cycle, AUXW charges capacitor C7 in order to provide a fast turn-off of the primary switch when the drive cycle terminates. Energy supplied by the auxiliary voltage switch (VAUXSWT) provides magnetizing current for the remainder of the power drive off-time.

Power Drive Section

The power drive section consists of the primary switch (composed of Q1, Q2 and associated components) and the power transformer (T1). This section receives nonregulated high dc power from the line rectifier and transforms it into high frequency ac voltage outputs to power the output section, and an auxiliary voltage section. When the power supply is operating, the auxiliary voltage section powers the control circuitry. Pulse width modulation governs the operation of the primary switch, and, in turn, of the output section, by controlling the amount of power through the power transformer.

A current sensing resistor (R8) senses current flow in the power drive path and applies a proportional signal to the PWM controller. The controller monitors this signal to provide pulse-by-pulse current limiting and to shut down the power supply in the event of an over-current fault.

Three secondary windings of the power transformer supply high frequency, low voltage ac to the output and auxiliary voltage sections.

Error Amplifier

The error amplifier compares a sample voltage from the +5V output (VSENSE) with a threshold level established from reference voltage REF+2.5. The amplifier converts the resulting error signal to current that drives an opto-isolator. The collector output of the opto-isolator controls the error voltage (VERROR) applied to a noninverting amplifier contained within the PWM controller IC. A resistor and capacitor located on the power card provides compensation for the error amplifier feedback loop through the COMP signal line by a resistor and capacitor located on the power card.

Output Section

The output section rectifies, filters, and regulates all outputs (except the +5V), senses the +5V output for control of the power drive section (regulates the +5V output), and protects against overvoltage on the +5V output.

In the *rectifying* and *filtering* circuits, transformer-to-inductor-connected rectifiers conduct during the drive cycle, supplying current to an inductor (L4).

During the off portion of the cycle, common line-to-inductor-connected rectifiers commutate current. Thus, current through the inductor remains continuous. Filtering capacitors reject ripples and smooth output to provide a steady dc output voltage.

Three linear series pass voltage regulators provide *regulation* of the + 12V, − 12V, and − 5V outputs, as well as limiting current on the three outputs. The + 12V output also contains a series pass transistor to boost the current handling capacity. This transistor turns on only when output current begins to approach the capacity of the regulator. The + 5V *sensing* circuit provides a sampling voltage (VSENSE) to the control circuitry. The control circuitry compares this sampling voltage with a reference voltage in order to perform the pulse width modulation that governs the operation of the power drive section of the power supply. Signal COMP provides a feedback path to establish gain control for the error amplifier of the control card. This gain is high so that pulse width modulation can regulate the + 5V output to a very close tolerance.

A Zener diode (VR1) provides *overvoltage protection* of the + 5V output. This diode conducts and puts the power supply into foldback limit if the voltage on the + 5V output exceeds 6.2V.

The nonregulated + 12 volt rectifier applies power (+ 12VUR) to a linear series pass regulator on the control card. The resulting + 2.5V (REF + 2.5) provides a reference voltage to the error amplifier and status circuit. The status circuit uses this reference voltage to determine when the + 5V output is above a minimum specified limit.

The 5V secondary winding of the power transformer also applies a high frequency ac voltage (VSWT) to the status circuit of the control card. This voltage, which is monitored by the status circuit, is proportional to the high dc voltage source. When this ac voltage drops below an established threshold (as in an input line voltage failure), the status circuit generates a powerfail signal, warning the system of an imminent power failure.

Auxiliary Voltage Section

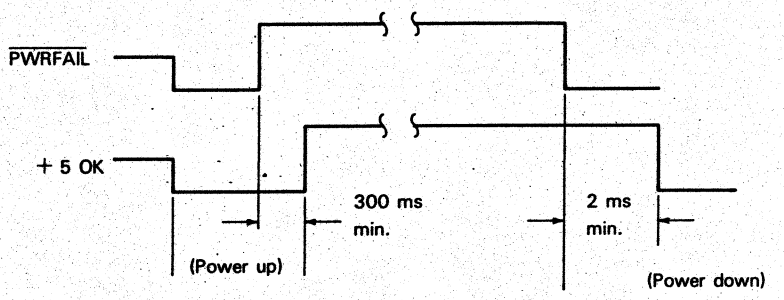
This section consists of a secondary winding of the power transformer, and a rectification network that provides auxiliary voltage (VAUX) to power the control circuitry of the power supply once the supply begins operation. The section also contains an auxiliary voltage switch that provides magnetizing current (VAUXSWT) to the base drive section when the PWM controller becomes operational.

The auxiliary voltage (VAUX) powers the PWM controller on the Control and Status card. Using VAUX, the PWM controller chip develops its internal voltage and generates a regulated reference voltage (REF + 5) for the power supply control circuitry. VAUX also drives an auxiliary voltage switch (Q4 on the control and status card) that turns on when the PWM controller becomes operational. The resulting voltage (VAUXSWT) drives the base drive section to provide magnetizing current during the power drive off-time.

The auxiliary voltage secondary winding of the power transformer also applies a high frequency ac voltage (AUXW) to the base drive section that charges a capacitor during the power drive cycle. This charge provides a fast turn-off of the primary switch when the drive cycle is terminated.

Status Circuits

Status circuits, contained on the Control and Status card, generate two power status signals for use by the processor: + 5 OK signals the processor when the + 5V output is above a specified limit; and PWRFAIL signals that a power loss is imminent. PWRFAIL asserts either when input power to the supply is lost, or when the power drive section shuts down. Timing for the two status signals, during both power up and power down, is presented in Figure 7-3.



ID-00374

Figure 7-3 Status signal timing diagram (power up and power down)

A 2.5 volt reference voltage (REF + 2.5) establishes a threshold voltage for the power ok and power fail status circuits.

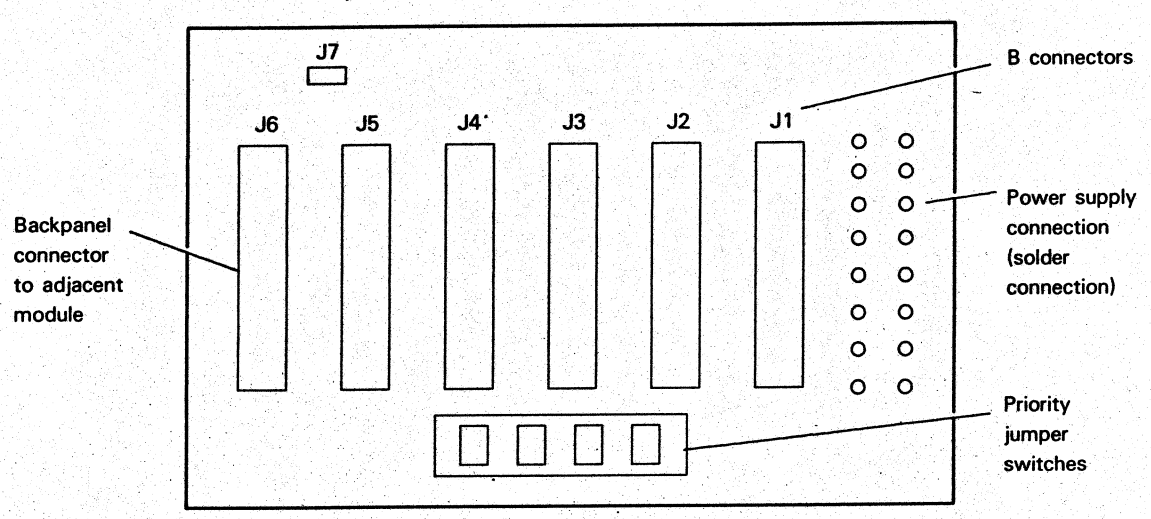
The power ok monitor compares the + 5V output, applied to a voltage divider, with a threshold level established from the reference voltage (REF + 2.5). During power up, an R/C circuit (R12 and C7) delays the assertion of the + 5 OK signal so that the supply voltages can stabilize. Should the + 5V output fall below the specified minimum, the capacitor discharges rapidly through a diode to negate the + 5 OK status signal.

The power fail detector rectifies and monitors the high frequency ac voltage from the 5V secondary winding of the power transformer (VSWT). During power up, PWRFAIL asserts until a capacitor charges to a value that places the monitored voltage level above a threshold level established from the reference voltage (REF + 2.5). This capacitor maintains the monitored voltage level above that threshold level while VSWT is present. During a power failure the capacitor discharges, allowing the monitored voltage level to fall below the established threshold level. The resulting condition causes PWRFAIL to be asserted to the processor.

Interconnection with the System

The power supply connects with the rest of the system through jacks J1 and J2. J1 receives input ac power from the internal ac power cable of the power supply module (PM). This cable plugs into the power supply card through an opening in the front of the case. J2 supplies the four dc voltages and two power status signals to the backpanel of the CPU logic module. One end of this cable plugs into the power supply card through an opening in the rear of the case, and the other end is soldered to the backpanel of the CPU logic module. Table 7-3 through Table 7-5 list each signal received or generated by the power supply card together with the jack location(s) of the signal. Refer to Figure 7-4 or the CLM backpanel logic schematic drawing, DGC No. 001-003344, for the locations of dc voltages and status signals on the backpanel.

Layout of backpanel (viewed from front of module)



Power supply connection

Optional real time clock connections					
GROUND	-	E36			
−5 PS #1	-	E34	RTC	-	E33
+5 PS #1	-	E32	+5 PS #1	-	E31
+5 PS #1	-	E30	+5 PS #1	-	E29
−5 PS #1	-	E23	NO CON.	-	E27
GROUND	-	E25	GROUND	-	E25
GROUND	-	E24	GROUND	-	E23
−12 PS #1	-	E22	NO CON.	-	E21
+12 PS #1	-	E20	+12 PS #1	-	E19
+12 PS #2	-	E18	+12 PS #2	-	E17
−12 PS #2	-	E16	NO CON.	-	E15
GROUND	-	E14	GROUND	-	E13
GROUND	-	E12	GROUND	-	E11
−5 PS #2	-	E10	NO CON.	-	E8
+5 PS #2	-	E8	+5 PS #2	-	E7
+5 PS #2	-	E6	+5 PS #2	-	E5
$\overline{PF1}$	-	E4	POWER OK1	-	E3
$\overline{PF2}$	-	E2	POWER OK2	-	E1

ID-00684

Figure 7-4 CLM backpanel

Table 7-3 ac power input

Signal	Jack Pin
Earth Ground	J1-1
A.C. Neutral	J1-2
A.C. Line	J1-3

NOTE *Jumper plug connecting J6-1 ([JMPB]) to J6-3 ([JMPA]) must be installed for 100 Vac or 110/120 Vac operation; removed for 220/240 Vac operation*

Table 7-4 dc voltage outputs

Signal	Jack Pin
+12 V	J2 pins 1, 2
Common	J2 pins 3, 4, 9, 11
+5 V	J2 pins 5-8
−5 V	J2-10
−12 V	J2-12

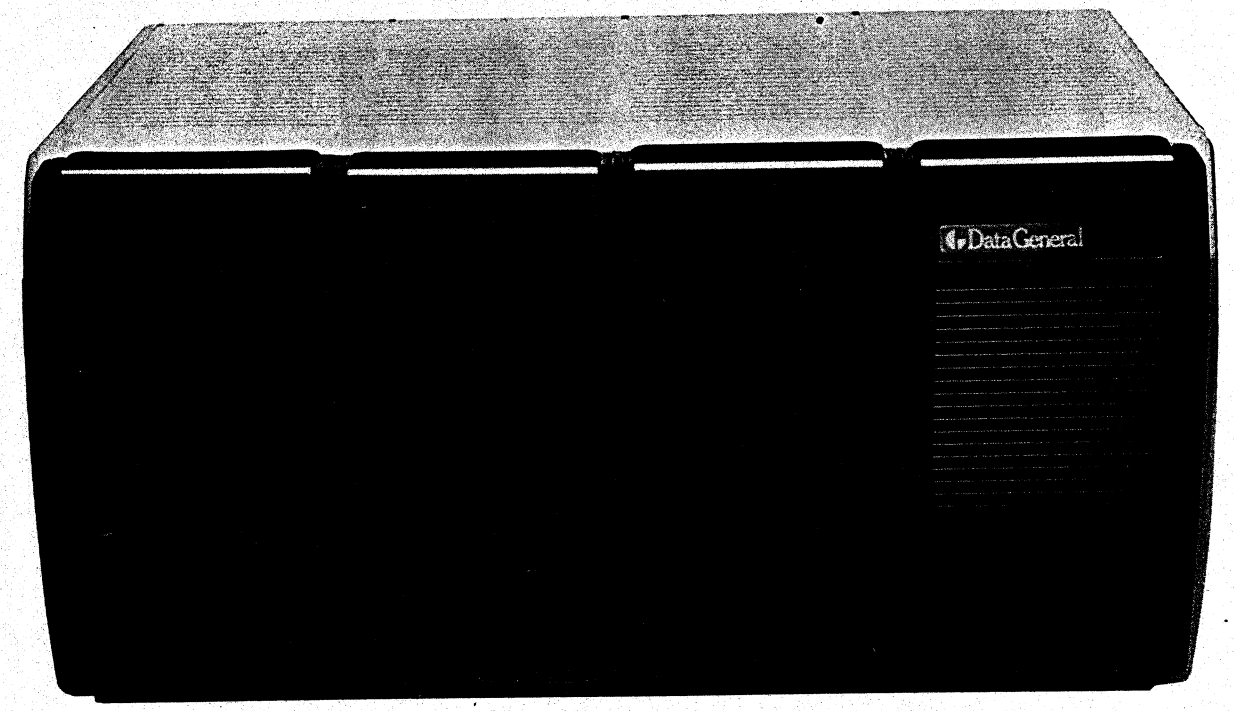
Table 7-5 Power status signals

Signal	Jack Pin
+5 OK	J2-14
$\overline{\text{PWRFAIL}}$	J2-15

Part

THREE

Mechanical Assemblies



PH-0726

Figure 8-1 Models 10 or 10/SP modules

Model 10 and Model 10/SP Modules and Configurations

8

Model 10 and Model 10/SP systems consist of three to six modules horizontally connected to form a single desk- or shelf-top unit. Figure 8-1 demonstrates this modular design. There are six module types, each type accommodating specific components of the system.

This chapter describes the Model 10 and Model 10/SP system modules and their architecture, configurations, and interconnections. The chapter also details system cabling and system expansion potential.

Unit Architecture

The six module types that can comprise Model 10 and Model 10/SP systems are pictured in Figure 8-2 through Figure 8-3 and are described below.

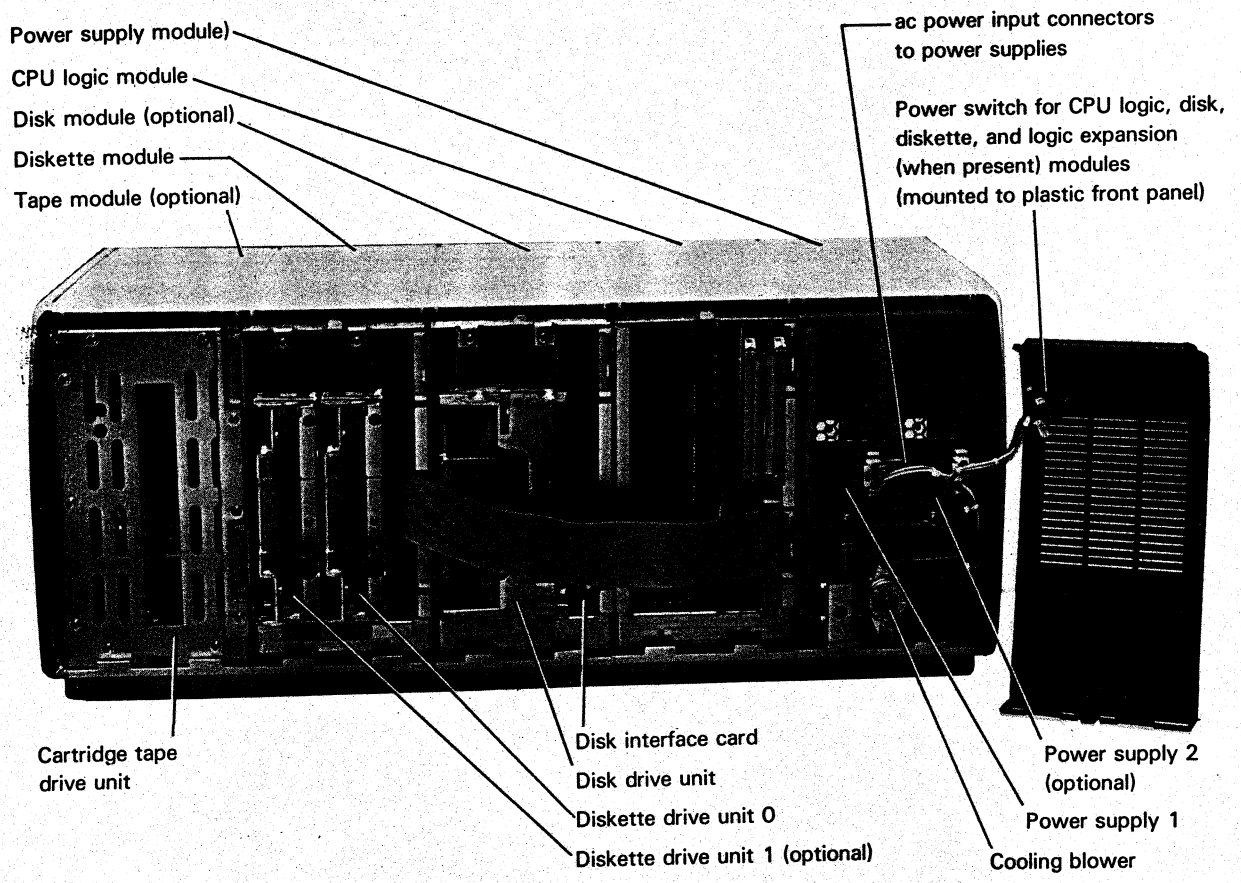


Figure 8-2 Models 10 or 10/SP system modules (front view)

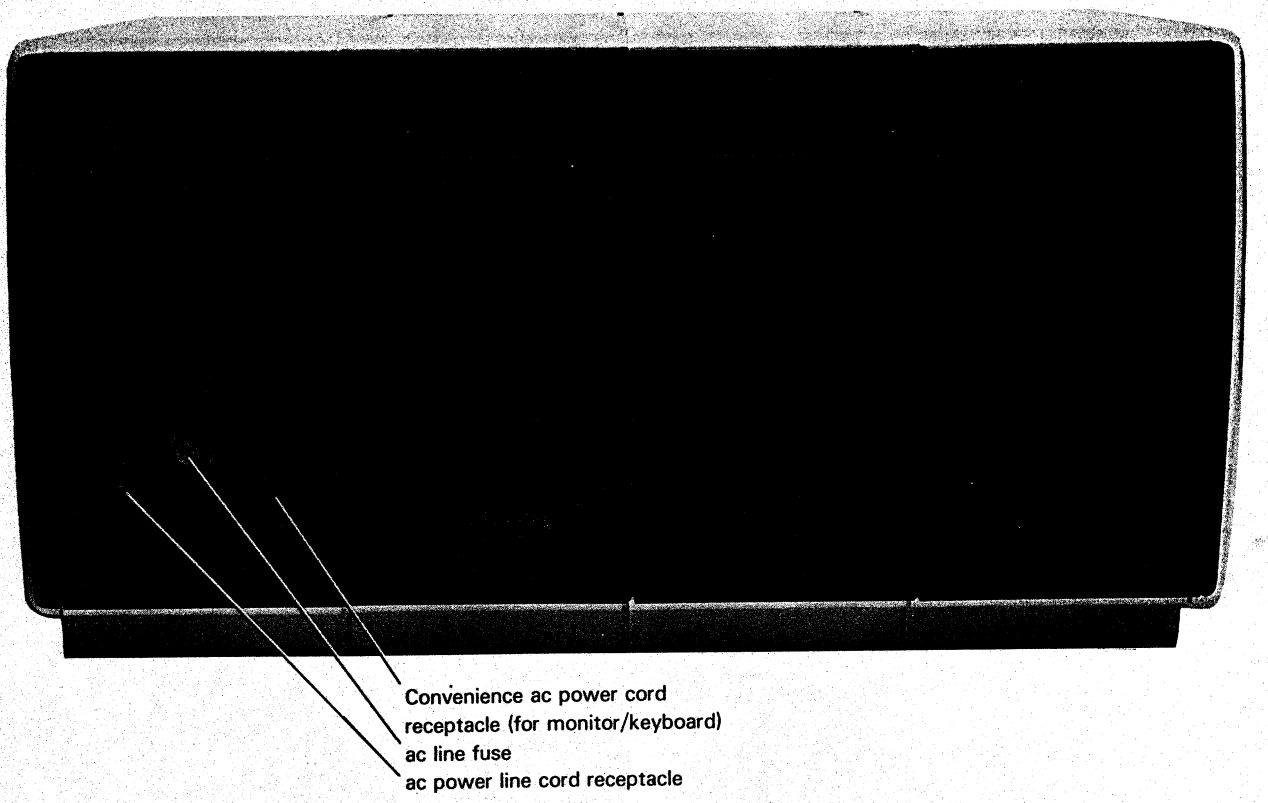


Figure 8-3 Models 10 or 10/SP system modules (rear view)

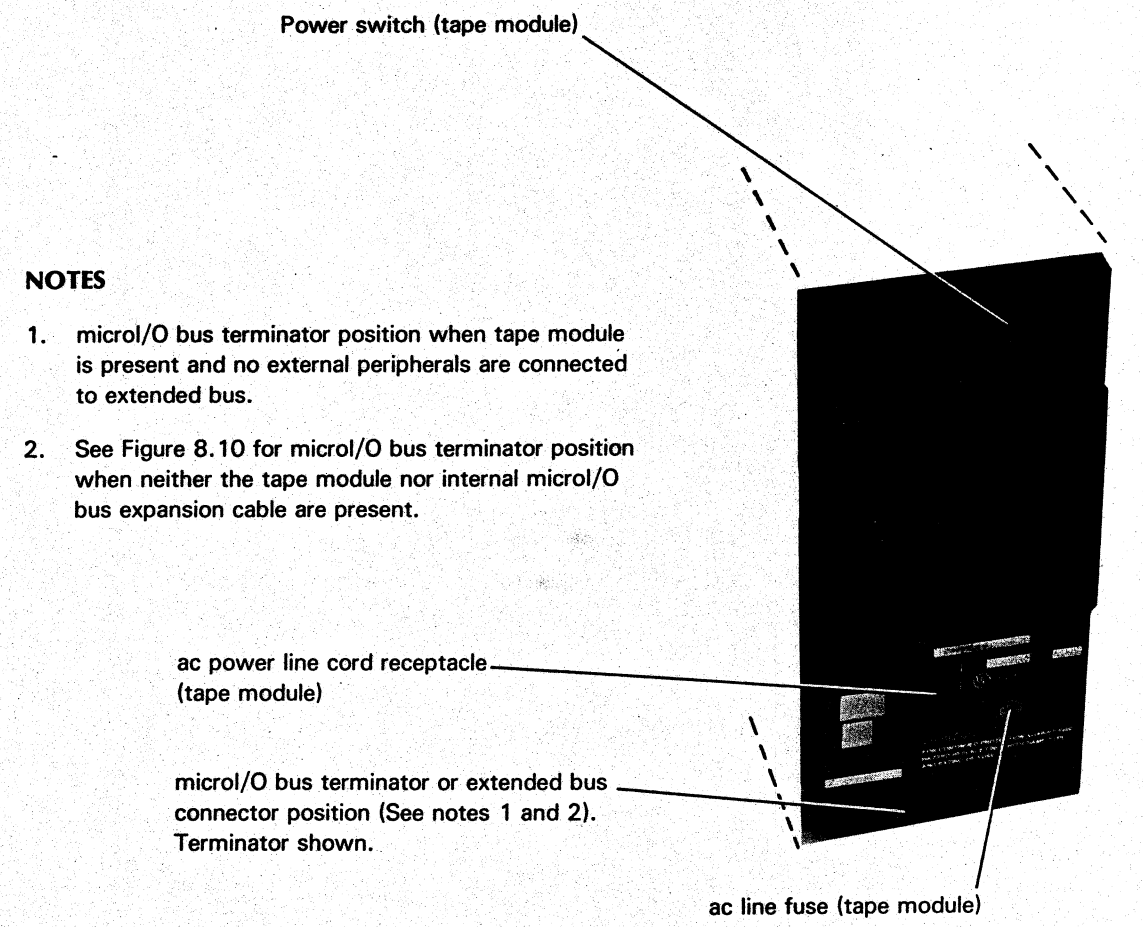


Figure 8-4 Tape module (rear view)

Power Supply Module (PM)

This module contains an ac line connection; a line fuse; a convenience ac outlet; a power on/off switch; a cooling blower; space for an optional line frequency clock card; and space for two power supply assemblies, one standard and one optional. The standard power supply (supply 1) occupies the left-most position; the optional power supply (supply 2) occupies the right-most position. The cooling blower, located at the bottom of this module, draws air, from the outside, through vents in the front and rear panels of the module and the power supply assemblies and forces it through all adjacent modules connected to its left. The air exhausts the modules through air vents in their top panels. Figure 8-5 shows the air flow pattern.

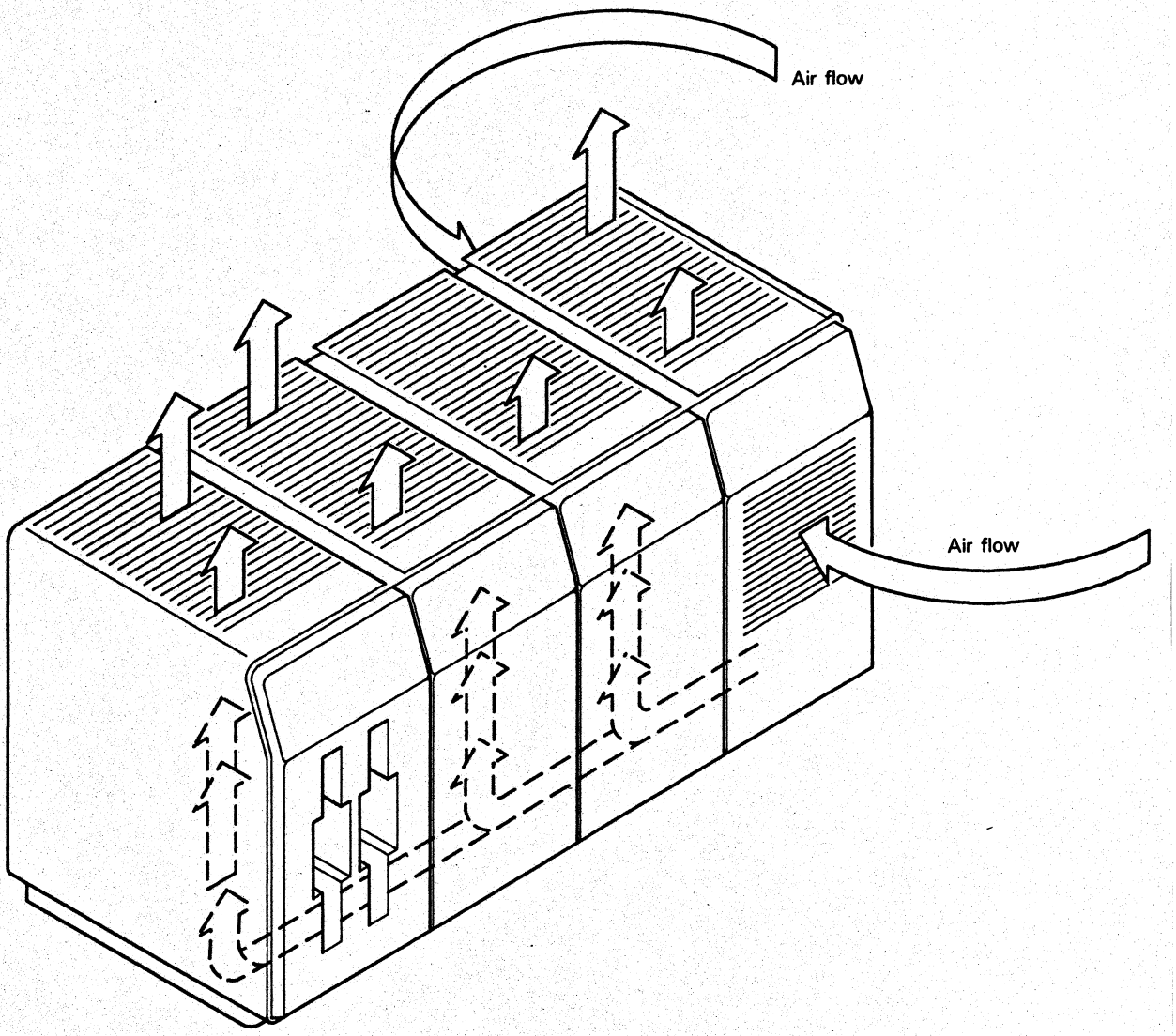


Figure 8-5 Air flow

CPU Logic Module (CLM)

This module accepts up to five 7-inch by 9-inch printed circuit cards including the two system processor cards, the optional memory card, and the optional video interface card. All components contained in the CLM receive their power from power supply 1.

Logic Expansion Module (LEM)

This module accepts up to five 7-inch by 9-inch printed circuit cards. An input/output interface printed circuit card must occupy slot 1 of this module to pass interrupt and data channel priority signals. (See "Backpanel Priority Switches" later in this chapter.) All components contained in the LEM receive their power from power supply 2.

Disk Module (DM)

This module contains a fixed disk drive unit and the disk controller card. The disk controller card can interface to a second disk drive contained in an expansion unit, described below. Both the disk controller card and the disk drive receive their power from power supply 2.

For additional disk storage capacity, an expansion unit consisting of power module and a disk module can be added to the system. When the expansion unit is added to the system, its power module contains only one power supply assembly (supply 1), and its disk module contains only the disk drive. The expansion drive connects to the disk controller card contained in the system disk unit.

Diskette Module (FM)

This module contains space for two diskette drive units, one standard, and one optional. The diskette controller resides on the SPU cards in the CPU logic module and communicates with the diskette drives by way of a cable that passes around the fronts of the modules. All components contained in the FM receive their power from power supply 1.

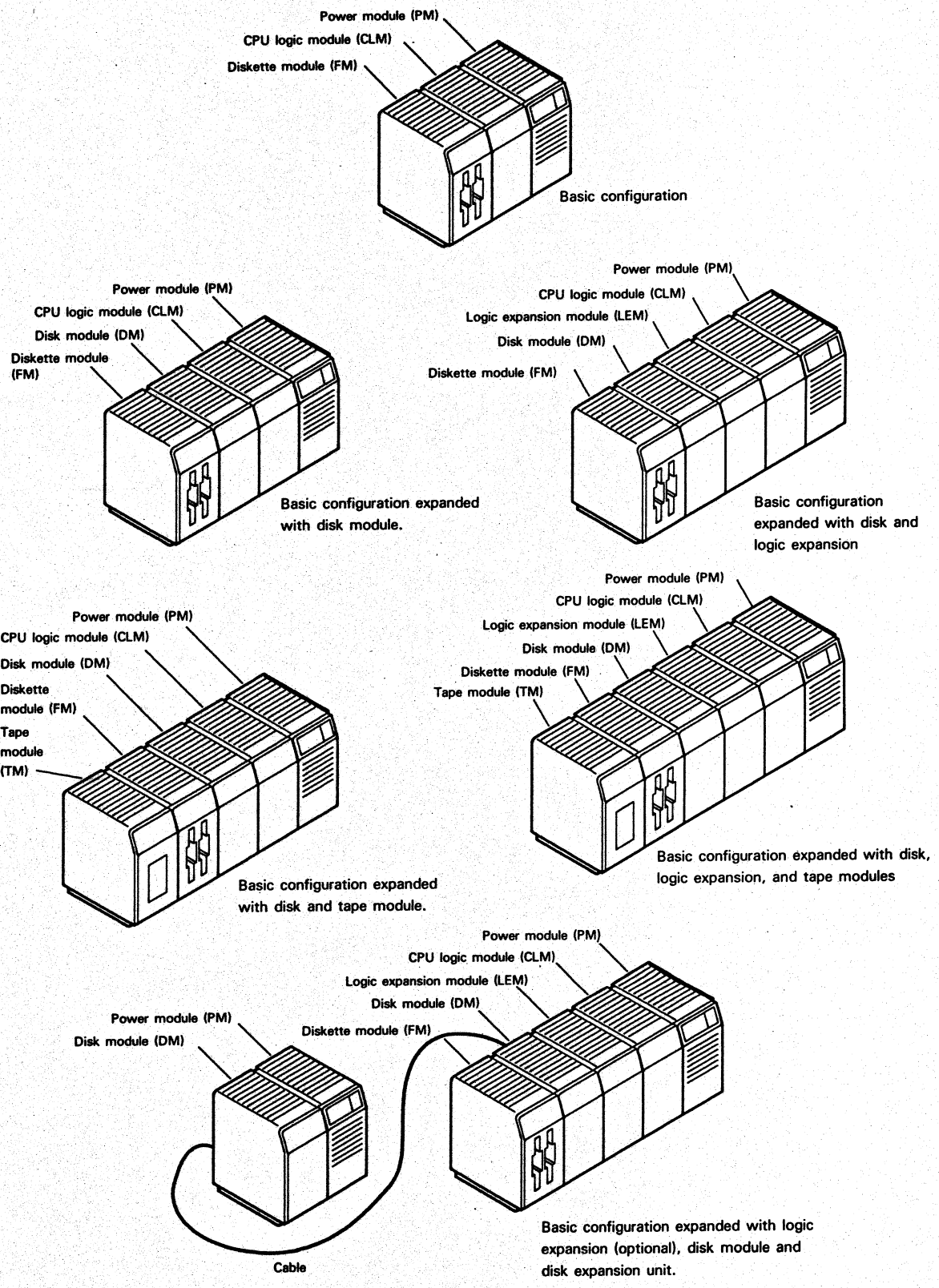
Tape Module (TM)

This module contains a cartridge tape drive unit, and controller card along with its own power supply, cooling fan, power on/off switch, ac line connection, and line fuse. The cartridge tape controller communicates with the system processor unit over the microI/O bus, to which it is connected by way of a flat ribbon cable to the diskette module backpanel.

Configurations

Model 10 and Model 10/SP system units always consist of at least three modules; a power module with one power supply assembly, CPU logic module, and diskette module. They can be expanded to six modules by the addition of a logic expansion module, a disk module, and tape module. The addition of a logic expansion module or a disk module requires that a second power supply assembly be installed in the power module. When a logic expansion module is added to the system, it must be placed to the left of and adjacent to the CLM. When a disk module is added, it must be placed to the right of and adjacent to the FM. Figure 8-6 shows guidelines for configuring a Model 10 or 10/SP system unit. When a tape module is added it must be placed to the left of and adjacent to the diskette module.

The disk expansion unit, when present, always consists of two modules; a power module with one power supply assembly, and a disk module.



ID-00656

Figure 8-6 Models 10 or 10/SP system unit configurations

Module Architecture

Each module consists of a metal cage with a base plate and removable plastic front, rear, and top panels. End modules have a removable plastic end panel. Adjacent cages mechanically interlock with one another. Except for the cages of the power supply, tape module, and expansion disk module, each cage contains its own printed circuit backpanel with connectors that electrically interconnect the backpanels of adjacent modules. The LEM and DM backpanels contain two intermodule connectors, one at each end. The CLM and FM backpanels each contain one intermodule connector, located on the left end of the CLM back-panel and the right end of the FM backpanel. All backpanels also contain one to five connector(s) that accept the B connector of up to five system printed circuit cards. Figure 8-7 shows a typical Model 10 and Model 10/SP module.

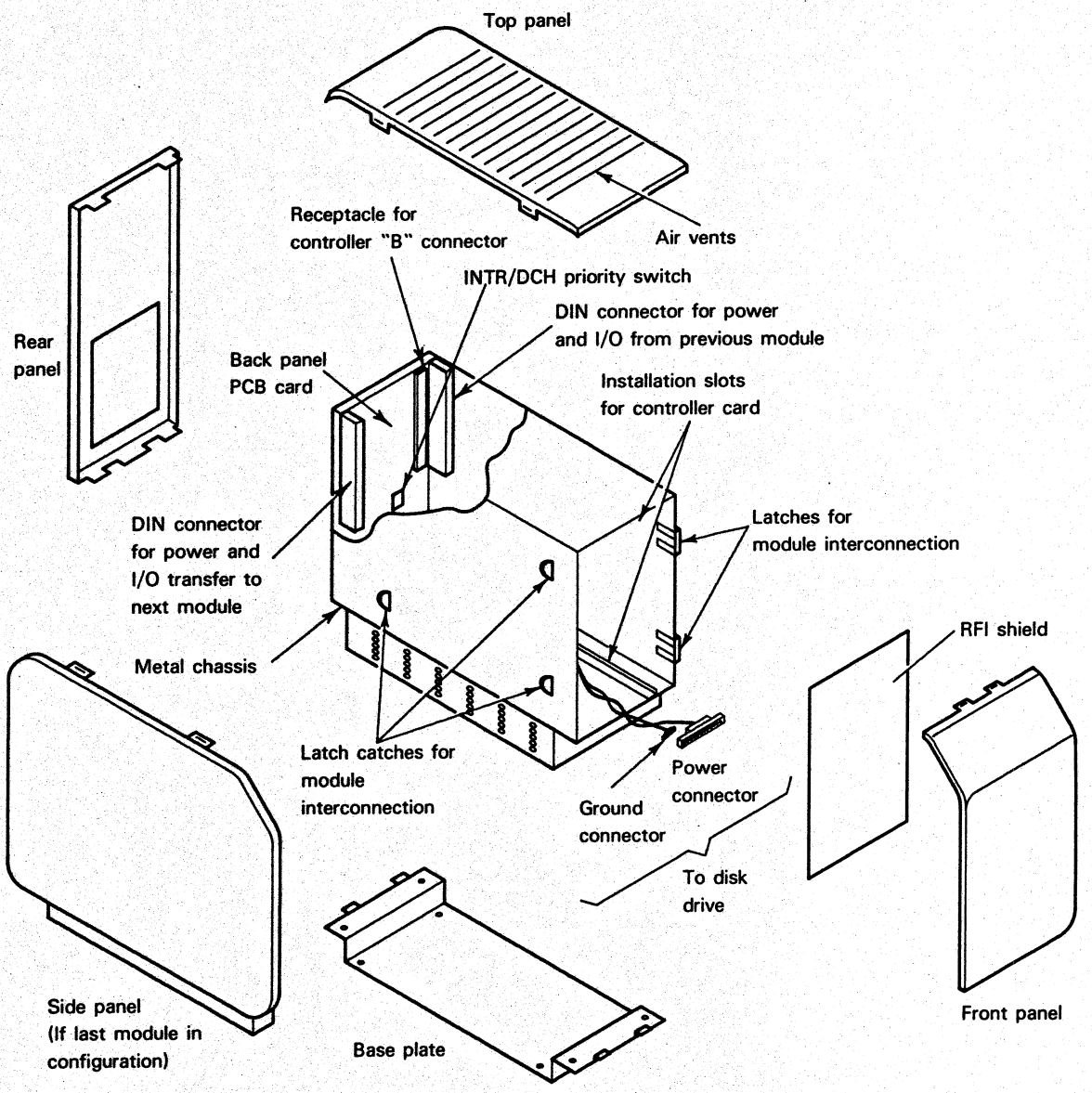


Figure 8-7 Typical Model 10 or 10/SP module

The CLM backpanel has the power supply harness soldered to its right end. On the other end of this harness are two connectors that plug into the standard and optional power supply assemblies as shown in Figure 8-8. The FM and DM backpanels have a dc power cable soldered to them near their left end. On the other end of each cable is a connector that plugs into the rear of the drive unit (refer to "Diskette Power Cable" and "Disk Power Cable" later in this chapter).

The dc power cable for the expansion disk module plugs into a dc load card mounted in the backpanel position of the expansion disk drive module. A dc power cable connects the load card to the power supply assembly of the expansion unit.

System cards (except for the tape controller), diskette or disk drives, and the power supply assemblies are inserted from the front of the module after removing its plastic front cover and RFI shield. The tape controller card inserts from the rear of its module after removing the rear plastic cover.

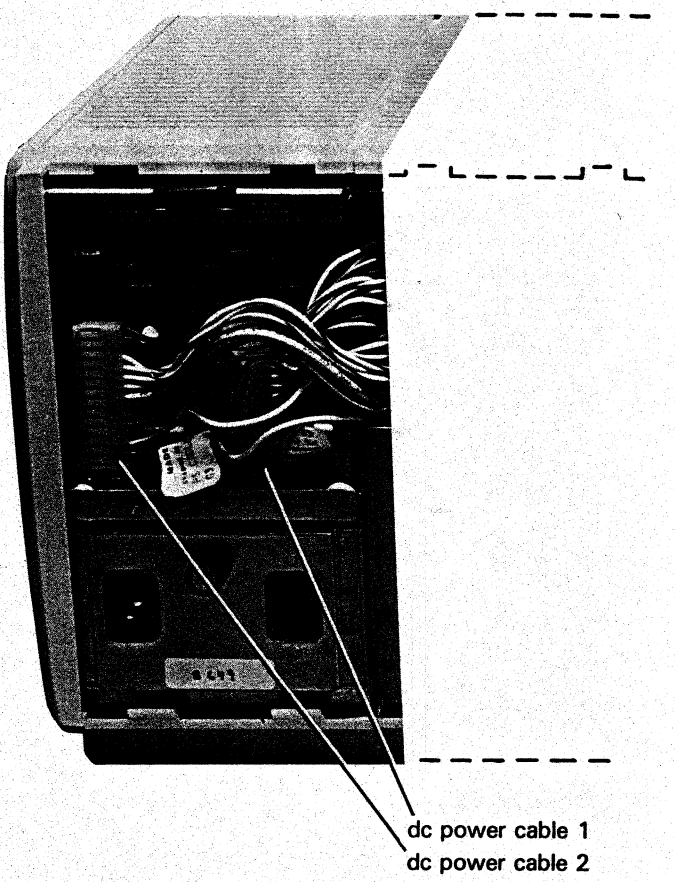


Figure 8-8 dc power cable connections

Power Bus

Two printed circuit power busses extend across the CLM, LEM, FM, and DM backpanels. One power bus on each backpanel connects power to all components contained within that module, and then passes to the next module on the left through connectors that electrically connect adjacent backpanels. The second power bus on each backpanel simply passes to the next module on the left through the connectors that electrically interconnect adjacent backpanels. Both power busses begin on the CLM backpanel. A power supply harness, which connects to the power supply assemblies, is soldered on the right end of the CLM backpanel.

The components in the CPU logic and diskette modules are powered by power bus 1, connected to power supply 1. The components in the logic expansion and disk modules are powered by power bus 2 connected to power supply 2. The expansion disk drive, when present, is powered from power supply 1 of the expansion power supply.

NOTE *When installing additional cards in the CPU logic or logic expansion module, or installing a disk module, or installing an additional diskette drive in the diskette module, be sure that the dc current draw does not exceed the dc current capacity of the respective power supply assembly.*

Memory Bus

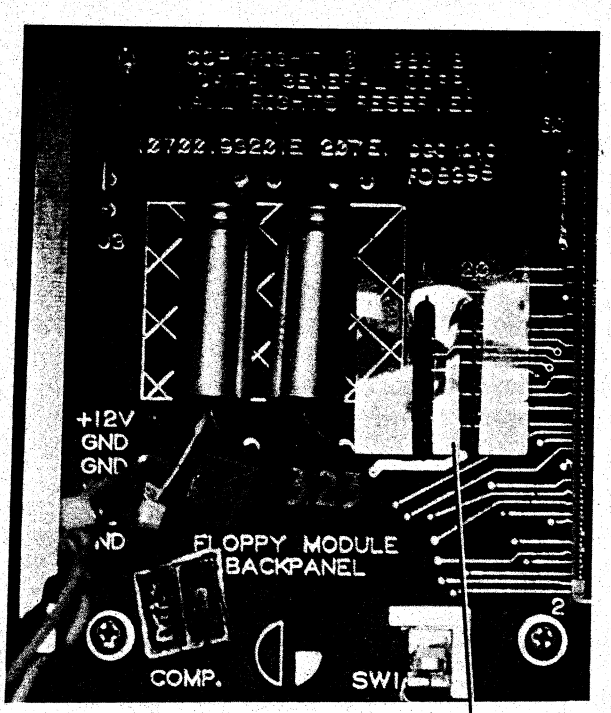
A printed circuit memory bus extends horizontally across the CLM backpanel. It connects the Model 10 and Model 10/SP system processor unit and optional video interface card to the optional system memory card. The memory bus contains 26 signal lines for address and data transfers, timing, and control.

Input/Output Bus

A printed circuit input/output bus extends horizontally across the CLM, LEM, FM, and DM backpanels. It connects the Model 10 and Model 10/SP two-card system processor unit to all system interfaces except the system console device. This bus extends across backpanels using the connectors by which they are electrically interconnected. When a cartridge tape unit is present, the input/output bus is extended by means of the cable from the diskette module to the tape unit. The input/output bus can be extended to incorporate compatible Data General peripherals, as explained under "Input/Output Bus Extension." The input/output bus contains 13 signal lines for data transfers, control, timing, and priority enforcement.

Input/Output Bus Termination

Termination for the input/output bus must be provided at the tape module (in the diskette module when the tape module is not present) when the extended microI/O bus is not present. Termination is accomplished with a DIP termination block that plugs onto pins located on the FM backpanel (see Figure 8-9) when the tape module or the optional internal extended microI/O bus cable within the diskette module is not present. When the tape module or the internal extended microI/O bus cable is present, termination is accomplished by a D connector type terminator that plugs onto the extended microI/O bus connector at the rear of the respective module (see Figure 8-10). When the extended microI/O bus is present, termination of the input/output bus must be accomplished by an I/O bus terminator at the last peripheral connected to the expanded bus.

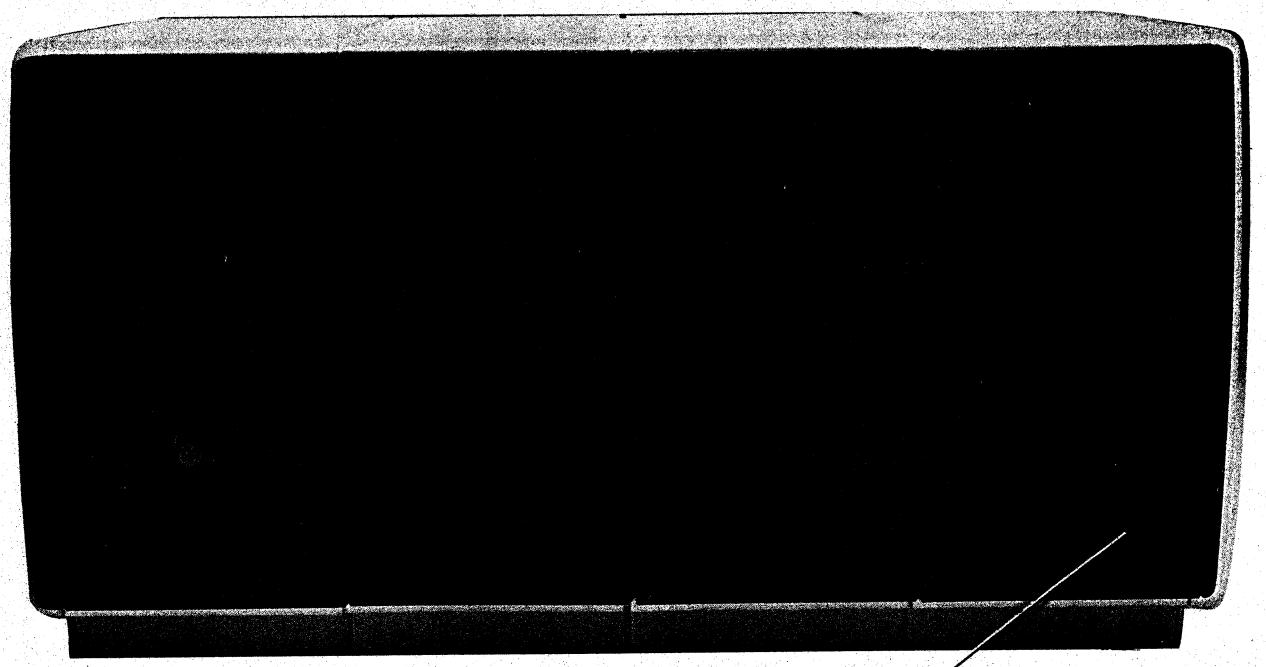


microl/O bus terminator (see note)

NOTE *Connector position for microl/O bus cable to tape module, when present, or optional internal extended microl/O bus cable, when present.*

DG-25944

Figure 8-9 microl/O bus terminator (in diskette module)



NOTES

1. microI/O bus terminator position when tape module is present and no external peripherals are connected to extended bus.
2. microI/O bus terminator position when tape module is not present or optional internal extended microI/O bus cable is present but no external peripherals are connected to extended bus.

DG-25981

Figure 8-10 External microI/O bus connector or terminator positions

Input/Output Bus Extension

The Model 10 and Model 10/SP system unit can be electrically connected to standard Data General peripherals with one internal and one external extended microI/O bus cable.

The internal microI/O bus cable is located in the tape module when present, otherwise an optional cable is located in the diskette module. One end contains a 20-pin DIP connector that plugs onto the diskette module backpanel pins provided for the microI/O bus terminator described above. The other end contains a 25-pin D connector that mounts to the rear of the module's cage, as shown in Figure 8-10.

The external extended microI/O bus cable contains 25-pin D connectors at both ends. One end plugs into a connector located at the rear of the tape or diskette module as described above; while the other end plugs into the first standard

peripheral connected to the Model 10 and Model 10/SP system. The extended microI/O bus cable is available in various lengths to accommodate peripheral placement.

Remember that when peripheral(s) is connected to the system, the last peripheral connected to the extended microI/O bus must terminate the bus.

Slot Assignments

Slot assignments for the Model 10 and Model 10/SP CPU logic and logic expansion modules are listed in Figure 8-11 and Figure 8-12 respectively.

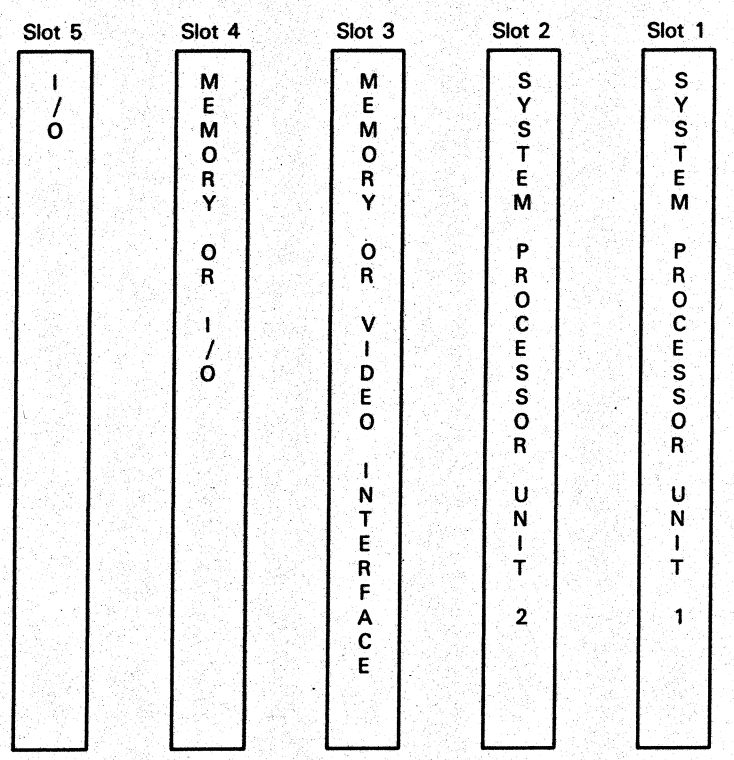
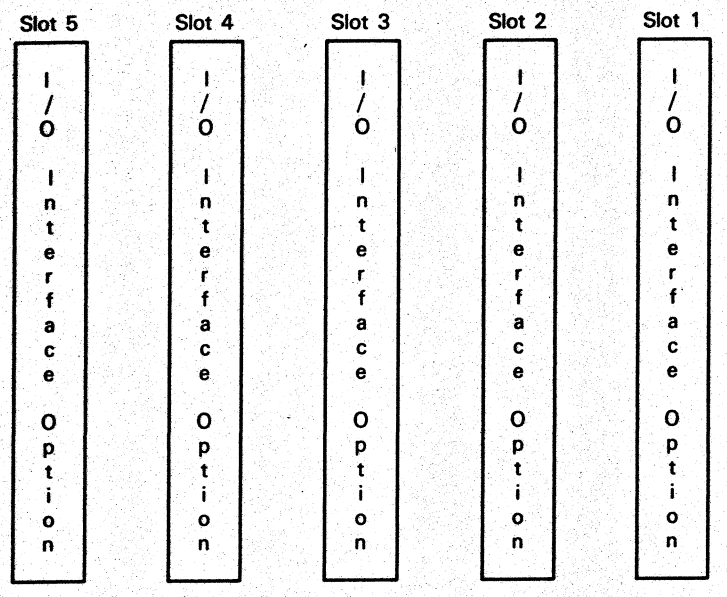


Figure 8-11 Slot assignments: SPU logic module



NOTE *Slot 1 must contain an I/O interface printed circuit card to pass interrupt and data channel priority signals.*

ID-00659

Figure 8-12 Slot assignments: logic expansion module

Backpanel Pin Assignments

Actual backpanel layouts and their pin assignments for all modules are shown in Figure 8-13 through Figure 8-16.

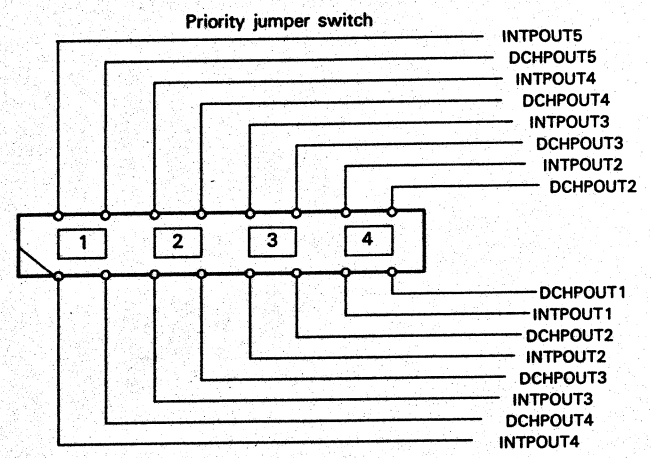
J6

PIN	COLUMN A	COLUMN B	COLUMN C
32	+5 PS #1	+5 PS #1	+5 PS #1
31	+5 PS #1	+5 PS #1	+5 PS #1
30	+5 PS #1	+5 PS #1	+5 PS #1
29	+5 PS #1	+51 PS #1	+5 PS #1
28	+5 PS #1	+5 PS #1	+5 PS #1
27	+5 PS #1	+5 PS #1	+5 PS #1
26	GROUND	GROUND	GROUND
25	GROUND	GROUND	GROUND
24	GROUND	GROUND	GROUND
23	GROUND	GROUND	GROUND
22	GROUND	GROUND	GROUND
21	GROUND	GROUND	GROUND
20	GROUND	BMCLOCK	$\overline{\text{BMCLOCK}}$
19	DCHPOUT5	$\overline{\text{BI/ODATA1}}$	BI/ODATA1
18	$\overline{\text{EXTDCHR}}$	INTPOUT5	$\overline{\text{CLEAR}}$
17	$\overline{\text{EXTINT}}$	BI/ODATA2	$\overline{\text{BI/ODATA2}}$
16	GROUND	$\overline{\text{BI/OCLOCK}}$	BI/OCLOCK
15	GROUND	GROUND	GROUND
14	GROUND	GROUND	GROUND
13	GROUND	GROUND	GROUND
12	−12 PS #1	+12 PS #1	+12 PS #1
11	−5 PS #1	−5 PS #2	+5 PS #1
10	−12 PS #2	+12 PS #2	+12 PS #2
9	GROUND	GROUND	GROUND
8	GROUND	GROUND	GROUND
7	GROUND	GROUND	GROUND
6	+5 PS #2	+5 PS #2	+5 PS #2
5	+5 PS #2	+5 PS #2	+5 PS #2
4	+5 PS #2	+5 PS #2	+5 PS #2
3	+5 PS #2	+5 PS #2	+5 PS #2
2	+5 PS #2	+5 PS #2	+5 PS #2
1	+5 PS #2	+5 PS #2	+5 PS #2

Power supply connection

Optional real time clock connections

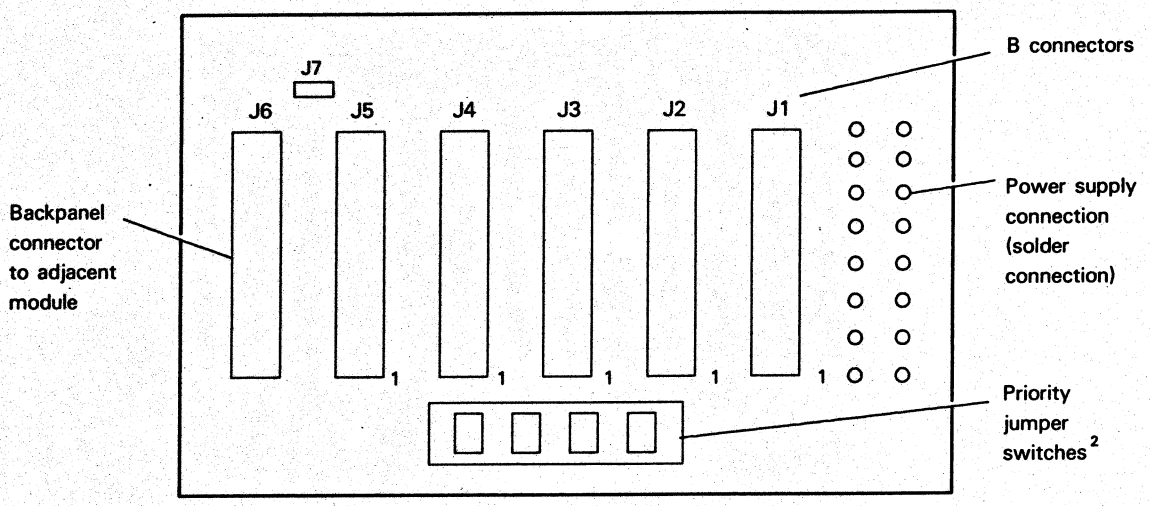
GROUND	-	E36			
−5 PS #1	-	E34	RTC	-	E33
+5 PS #1	-	E32	+5 PS #1	-	E31
+5 PS #1	-	E30	+5 PS #1	-	E29
−5 PS #1	-	E28	NO CON.	-	E27
GROUND	-	E26	GROUND	-	E25
GROUND	-	E24	GROUND	-	E23
−12 PS #1	-	E22	NO CON.	-	E21
+12 PS #1	-	E20	+12 PS #1	-	E19
+12 PS #2	-	E18	+12 PS #2	-	E17
−12 PS #2	-	E16	NO CON.	-	E15
GROUND	-	E14	GROUND	-	E13
GROUND	-	E12	GROUND	-	E11
−5 PS #2	-	E10	NO CON.	-	E9
+5 PS #2	-	E8	+5 PS #2	-	E7
+5 PS #2	-	E6	+5 PS #2	-	E5
$\overline{\text{PF1}}$	-	E4	POWER OK1	-	E3
$\overline{\text{PF2}}$	-	E2	POWER OK2	-	E1



ID-00685

Figure 8-13 Backpanel pin assignments: SPU logic module

Actual Layout of backpanel (viewed from front of module)



1. Indicates pin 1 as viewed from front of chassis
2. When card slot empty, switch is in closed position to extend priority chain.

SLOT 5 - J5

EVEN PINS	ODD PINS
60 +5 PS #1	59 +5 PS #1
58 −5 PS #1	57 +5 PS #1
56 +12 PS #1	55 +12 PS #1
54 GROUND	53 GROUND
52 SYSCLOCK	51 XMA1
50 DATA0	49 DATA0
48 BUSADREN	47 XMA0
46 DATA1	45 DATA9
44 XMA2	43 SPARE 1
42 DATA2	41 DATA10
40 XMA3	39 −12 PS #1
38 DATA3	37 DATA11
36 GROUND	35 SPARE 2
34 DATA4	33 DATA12
32 DATA5	31 DATA13
30 WH/PARH	29 WL/PARL
28 DATA6	27 DATA14
26 BUSMEMCYC	25 XMA4
24 DATA7	23 DATA15
22 DCHPOUT4	21 DCHPOUT5
20 INTPOUT4	19 INTPOUT5
18 BUSREADY	17 SPARE 3
16 BI/OCLOCK	15 BI/OCLOCK
14 GROUND	13 BI/ODATA2
12 BI/ODATA2	11 GROUND
10 SPARE 4	9 EXTDCHR
8 EXTINT	7 SPARE 5
6 CLEAR	5 BI/ODATA1
4 BI/ODATA1	3 GROUND
2 BMCLOCK	1 BMCLOCK

SLOT 4 - J4

EVEN PINS	ODD PINS
60 +5 PS #1	59 +5 PS #1
58 −5 PS #1	57 +5 PS #1
56 +12 PS #1	55 +12 PS #1
54 GROUND	53 GROUND
52 SYSCLOCK	51 XMA1
50 DATA0	49 DATA0
48 BUSADREN	47 XMA0
46 DATA1	45 DATA9
44 XMA2	43 SPARE 1
42 DATA2	41 DATA10
40 XMA3	39 −12 PS #1
38 DATA3	37 DATA11
36 GROUND	35 SPARE 2
34 DATA4	33 DATA12
32 DATA5	31 DATA13
30 WH/PARH	29 WL/PARL
28 DATA6	27 DATA14
26 BUSMEMCYC	25 XMA4
24 DATA7	23 DATA15
22 DCHPOUT3	21 DCHPOUT4
20 INTPOUT3	19 INTPOUT4
18 BUSREADY	17 SPARE 3
16 BI/OCLOCK	15 BI/OCLOCK
14 GROUND	13 BI/ODATA2
12 BI/ODATA2	11 GROUND
10 SPARE 4	9 EXTDCHR
8 EXTINT	7 SPARE 5
6 CLEAR	5 BI/ODATA1
4 BI/ODATA1	3 GROUND
2 BMCLOCK	1 BMCLOCK

SLOT 3 - J3

EVEN PINS	ODD PINS
60 +5 PS #1	59 +5 PS #1
58 −5 PS #1	57 +5 PS #1
56 +12 PS #1	55 +12 PS #1
54 GROUND	53 GROUND
52 SYSCLOCK	51 XMA1
50 DATA0	49 DATA0
48 BUSADREN	47 XMA0
46 DATA1	45 DATA9
44 XMA2	43 SPARE 1
42 DATA2	41 DATA10
40 XMA3	39 −12 PS #1
38 DATA3	37 DATA11
36 GROUND	35 SPARE 2
34 DATA4	33 DATA12
32 DATA5	31 DATA13
30 WH/PARH	29 WL/PARL
28 DATA6	27 DATA14
26 BUSMEMCYC	25 XMA4
24 DATA7	23 DATA15
22 DCHPOUT2	21 DCHPOUT3
20 INTPOUT2	19 INTPOUT3
18 BUSREADY	17 SPARE 3
16 BI/OCLOCK	15 BI/OCLOCK
14 GROUND	13 BI/ODATA2
12 BI/ODATA2	11 GROUND
10 SPARE 4	9 EXTDCHR
8 EXTINT	7 SPARE 5
6 CLEAR	5 BI/ODATA1
4 BI/ODATA1	3 GROUND
2 BMCLOCK	1 BMCLOCK

SLOT 2 - J2

EVEN PINS	ODD PINS
60 +5 PS #1	59 +5 PS #1
58 −5 PS #1	57 +5 PS #1
56 +12 PS #1	55 +12 PS #1
54 GROUND	53 GROUND
52 SYSCLOCK	51 XMA1
50 DATA0	49 DATA0
48 BUSADREN	47 XMA0
46 DATA1	45 DATA9
44 XMA2	43 SPARE 1
42 DATA2	41 DATA10
40 XMA3	39 −12 PS #1
38 DATA3	37 DATA11
36 GROUND	35 SPARE 2
34 DATA4	33 DATA12
32 DATA5	31 DATA13
30 WH/PARH	29 WL/PARL
28 DATA6	27 DATA14
26 BUSMEMCYC	25 XMA4
24 DATA7	23 DATA15
22 DCHPOUT1	21 DCHPOUT2
20 INTPOUT1	19 INTPOUT2
18 BUSREADY	17 SPARE 3
16 BI/OCLOCK	15 BI/OCLOCK
14 GROUND	13 BI/ODATA2
12 BI/ODATA2	11 GROUND
10 SPARE 4	9 EXTDCHR
8 EXTINT	7 SPARE 5
6 CLEAR	5 BI/ODATA1
4 BI/ODATA1	3 GROUND
2 BMCLOCK	1 BMCLOCK

SLOT 1 - J1

EVEN PINS	ODD PINS
60 +5 PS #1	59 +5 PS #1
58 −5 PS #1	57 +5 PS #1
56 +12 PS #1	55 +12 PS #1
54 GROUND	53 GROUND
52 SYSCLOCK	51 XMA1
50 DATA0	49 DATA0
48 BUSADREN	47 XMA0
46 DATA1	45 DATA9
44 XMA2	43 SPARE 1
42 DATA2	41 DATA10
40 XMA3	39 −12 PS #1
38 DATA3	37 DATA11
36 GROUND	35 RTC #
34 DATA4	33 DATA12
32 DATA5	31 DATA13
30 WH/PARH	29 WL/PARL
28 DATA6	27 DATA14
26 BUSMEMCYC	25 XMA4
24 DATA7	23 DATA15
22 POWER OK	21 DCHPOUT1
20 PF	19 INTPOUT1
18 BUSREADY	17 SPARE 3
16 BI/OCLOCK	15 BI/OCLOCK
14 GROUND	13 BI/ODATA2
12 BI/ODATA2	11 GROUND
10 SPARE 4	9 EXTDCHR
8 EXTINT	7 SPARE 5
6 CLEAR	5 BI/ODATA1
4 BI/ODATA1	3 GROUND
2 BMCLOCK	1 BMCLOCK

ID-00685

Figure 8-13 Backpanel pin assignments: SPU logic module

J7

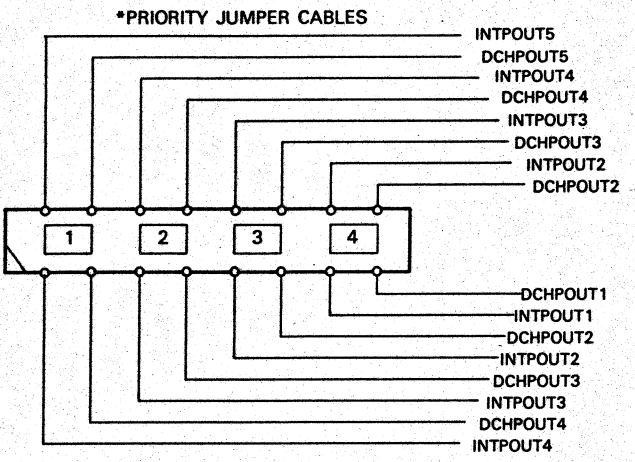
PIN	COLUMN A	COLUMN B	COLUMN C
32	+5 PS #1	+5 PS #1	+5 PS #1
31	+5 PS #1	+5 PS #1	+5 PS #1
30	+5 PS #1	+5 PS #1	+5 PS #1
29	+5 PS #1	+51 PS #1	+5 PS #1
28	+5 PS #1	+5 PS #1	+5 PS #1
27	+5 PS #1	+5 PS #1	+5 PS #1
26	GROUND	GROUND	GROUND
25	GROUND	GROUND	GROUND
24	GROUND	GROUND	GROUND
23	GROUND	GROUND	GROUND
22	GROUND	GROUND	GROUND
21	GROUND	GROUND	GROUND
20	GROUND	BMCLOCK	$\overline{\text{BMCLOCK}}$
19	DCHPOUT5	$\overline{\text{BI/ODATA1}}$	BI/ODATA1
18	$\overline{\text{EXTDCHR}}$	INTPOUT5	$\overline{\text{CLEAR}}$
17	$\overline{\text{EXTINT}}$	BI/ODATA2	$\overline{\text{BI/ODATA2}}$
16	GROUND	$\overline{\text{BI/OCLOCK}}$	BI/OCLOCK
15	GROUND	GROUND	GROUND
14	GROUND	GROUND	GROUND
13	GROUND	GROUND	GROUND
12	−12 PS #1	+ 12 PS #1	+ 12 PS #1
11	−5 PS #1	− 5 PS #2	+ 5 PS #1
10	−12 PS #2	+ 12 PS #2	+ 12 PS #2
9	GROUND	GROUND	GROUND
8	GROUND	GROUND	GROUND
7	GROUND	GROUND	GROUND
6	+5 PS #2	+5 PS #2	+5 PS #2
5	+5 PS #2	+5 PS #2	+5 PS #2
4	+5 PS #2	+5 PS #2	+5 PS #2
3	+5 PS #2	+5 PS #2	+5 PS #2
2	+5 PS #2	+5 PS #2	+5 PS #2
1	+5 PS #2	+5 PS #2	+5 PS #2

J8

PIN	COLUMN C	COLUMN B	COLUMN A
32	+5 PS #1	+5 PS #1	+5 PS #1
31	+5 PS #1	+5 PS #1	+5 PS #1
30	+5 PS #1	+5 PS #1	+5 PS #1
29	+5 PS #1	+51 PS #1	+5 PS #1
28	+5 PS #1	+5 PS #1	+5 PS #1
27	+5 PS #1	+5 PS #1	+5 PS #1
26	GROUND	GROUND	GROUND
25	GROUND	GROUND	GROUND
24	GROUND	GROUND	GROUND
23	GROUND	GROUND	GROUND
22	GROUND	GROUND	GROUND
21	GROUND	GROUND	GROUND
20	$\overline{\text{BMCLOCK}}$	BMCLOCK	GROUND
19	BI/ODATA1	$\overline{\text{BI/ODATA1}}$	DCHPIN
18	$\overline{\text{CLEAR}}$	INTPIN	$\overline{\text{EXTDCHR}}$
17	$\overline{\text{BI/ODATA2}}$	BI/ODATA2	$\overline{\text{EXTINT}}$
16	BI/OCLOCK	$\overline{\text{BI/OCLOCK}}$	GROUND
15	GROUND	GROUND	GROUND
14	GROUND	GROUND	GROUND
13	GROUND	GROUND	GROUND
12	+12 PS #1	+ 12 PS #1	− 12 PS #1
11	+12 PS #2	− 5 PS #2	− 5 PS #1
10	+12 PS #2	+ 12 PS #2	− 12 PS #2
9	GROUND	GROUND	GROUND
8	GROUND	GROUND	GROUND
7	GROUND	GROUND	GROUND
6	+5 PS #2	+5 PS #2	+5 PS #2
5	+5 PS #2	+5 PS #2	+5 PS #2
4	+5 PS #2	+5 PS #2	+5 PS #2
3	+5 PS #2	+5 PS #2	+5 PS #2
2	+5 PS #2	+5 PS #2	+5 PS #2
1	+5 PS #2	+5 PS #2	+5 PS #2

ID-00686

Figure 8-14 Backpanel pin assignments: logic expansion module



SLOT 5 - J3

EVEN PINS		ODD PINS	
60	+5 PS #2	59	+5 PS #2
58	−5 PS #2	57	+5 PS #2
56	+12 PS #2	55	+12 PS #2
54	GROUND	53	GROUND
52		51	
50		49	
48		47	
46		45	
44		43	
42		41	
		39	−12 PS #2
		37	
36	GROUND	35	
34		33	
32		31	
30		29	
28		27	
26		25	
24		23	
22	DCHPOUT4	21	DCHPOUT5
20	INTPOUT4	19	INTPOUT5
18		17	
16	BI/OCLOCK	15	$\overline{\text{BI/OCLOCK}}$
14	GROUND	13	BI/ODATA2
12	$\overline{\text{BI/ODATA2}}$	11	GROUND
10		9	$\overline{\text{EXTDCHR}}$
8	$\overline{\text{EXTINT}}$	7	
6	$\overline{\text{CLEAR}}$	5	$\overline{\text{BI/ODATA1}}$
4	BI/ODATA1	3	GROUND
2	$\overline{\text{BMCLOCK}}$	1	BMCLOCK

SLOT 4 - J4

EVEN PINS		ODD PINS	
60	+5 PS #2	59	+5 PS #2
58	−5 PS #2	57	+5 PS #2
56	+12 PS #2	55	+12 PS #2
54	GROUND	53	GROUND
52		51	
50		49	
48		47	
46		45	
44		43	
42		41	
40		39	−12 PS #2
38		37	
36	GROUND	35	
34		33	
32		31	
30		29	
28		27	
26		25	
24		23	
22	DCHPOUT3	21	DCHPOUT4
20	INTPOUT3	19	INTPOUT4
18		17	
16	BI/OCLOCK	15	$\overline{\text{BI/OCLOCK}}$
14	GROUND	13	BI/ODATA2
12	$\overline{\text{BI/ODATA2}}$	11	GROUND
10		9	$\overline{\text{EXTDCHR}}$
8	$\overline{\text{EXTINT}}$	7	
6	$\overline{\text{CLEAR}}$	5	$\overline{\text{BI/ODATA1}}$
4	BI/ODATA1	3	GROUND
2	$\overline{\text{BMCLOCK}}$	1	BMCLOCK

SLOT 3 - J3

EVEN PINS		ODD PINS	
60	+5 PS #2	59	+5 PS #2
58	−5 PS #2	57	+5 PS #2
56	+12 PS #2	55	+12 PS #2
54	GROUND	53	GROUND
52		51	
50		49	
48		47	
46		45	
44		43	
42		41	
40		39	−12 PS #2
38		37	
36	GROUND	35	
34		33	
32		31	
30		29	
28		27	
26		25	
24		23	
22	DCHPOUT2	21	DCHPOUT3
20	INTPOUT2	19	INTPOUT3
18		17	
16	BI/OCLOCK	15	$\overline{\text{BI/OCLOCK}}$
14	GROUND	13	BI/ODATA2
12	$\overline{\text{BI/ODATA2}}$	11	GROUND
10		9	$\overline{\text{EXTDCHR}}$
8	$\overline{\text{EXTINT}}$	7	
6	$\overline{\text{CLEAR}}$	5	$\overline{\text{BI/ODATA1}}$
4	BI/ODATA1	3	GROUND
2	$\overline{\text{BMCLOCK}}$	1	BMCLOCK

ID-00686

Figure 8-14 Backpanel pin assignments: logic expansion module

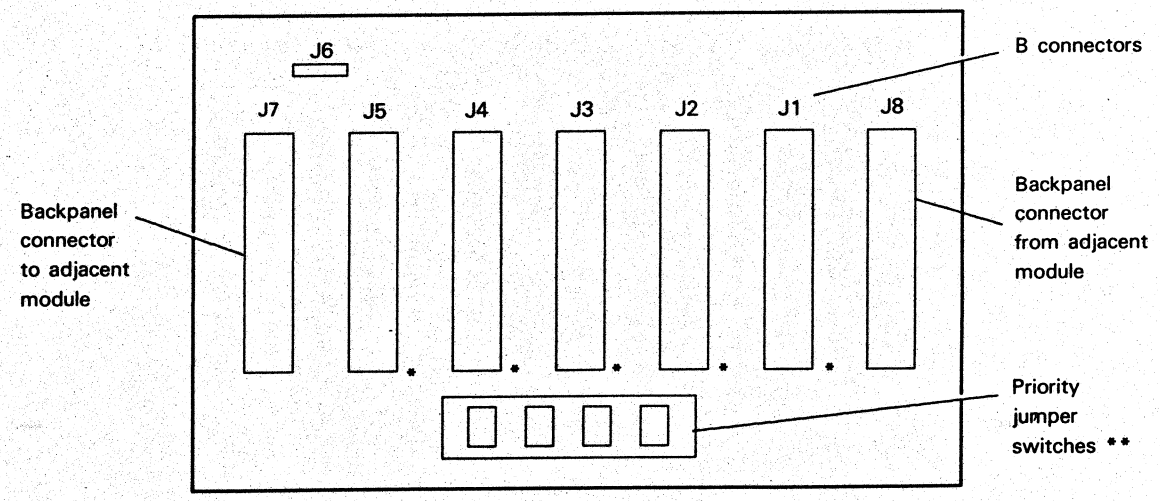
SLOT 2 - J2

EVEN PINS		ODD PINS	
60	+5 PS #2	59	+5 PS #2
58	−5 PS #2	57	+5 PS #2
56	+12 PS #2	55	+12 PS #2
54	GROUND	53	GROUND
52		51	
50		49	
48		47	
46		45	
44		43	
42		41	
40		39	−12 PS #2
38		37	
36	GROUND	35	
34		33	
32		31	
30		29	
28		27	
26		25	
24		23	
22	DCHPOUT1	21	DCHPOUT2
20	INTPOUT1	19	INTPOUT2
18		17	
16	BI/OCLOCK	15	$\overline{\text{BI/OCLOCK}}$
14	GROUND	13	BI/ODATA2
12	$\overline{\text{BI/ODATA2}}$	11	GROUND
10		9	$\overline{\text{EXTDCHR}}$
8	$\overline{\text{EXTINT}}$	7	
6	$\overline{\text{CLEAR}}$	5	$\overline{\text{BI/ODATA1}}$
4	BI/ODATA1	3	GROUND
2	$\overline{\text{BMCLOCK}}$	1	BMCLOCK

SLOT 1 - J1

EVEN PINS		ODD PINS	
60	+5 PS #2	59	+5 PS #2
58	−5 PS #2	57	+5 PS #2
56	+12 PS #2	55	+12 PS #2
54	GROUND	53	GROUND
52		51	
50		49	
48		47	
46		45	
44		43	
42		41	
40		39	−12 PS #2
38		37	
36	GROUND	35	
34		33	
32		31	
30		29	
28		27	
26		25	
24		23	
22	DCHPIN	21	DCHPOUT1
20	INTPIN	19	INTPOUT1
18		17	
16	BI/OCLOCK	15	$\overline{\text{BI/OCLOCK}}$
14	GROUND	13	BI/ODATA2
12	$\overline{\text{BI/ODATA2}}$	11	GROUND
10		9	$\overline{\text{EXTDCHR}}$
8	$\overline{\text{EXTINT}}$	7	
6	$\overline{\text{CLEAR}}$	5	$\overline{\text{BI/ODATA1}}$
4	BI/ODATA1	3	GROUND
2	$\overline{\text{BMCLOCK}}$	1	BMCLOCK

Actual Layout of backpanel (viewed from front of module)



* Indicates pin 1 as viewed from the front of chassis

** When card slot is empty, switch is in closed position to extend priority chain

SLOT 1 - J2

EVEN PINS	ODD PINS
60 + 5 PS #1	59 + 5 PS #1
58 − 5 PS #1	57 + 5 PS #1
56 + 12 PS #1	55 + 12 PS #1
54 GROUND	53 GROUND
52	51
50	49
48	47
46	45
44	43
42	41
40	39 −12 PS #1
38	37
36 GROUND	35
34	33
32	31
30	29
28	27
26	25
24	23
22 DCHPIN	21 DCHPOUT1
20 INTPIN	19 INTPOUT1
18	17
16 BI/OCLOCK	15 $\overline{\text{BI/OCLOCK}}$
14 GROUND	13 BI/ODATA2
12 $\overline{\text{BI/ODATA2}}$	11 GROUND
10	9 $\overline{\text{EXTDCHR}}$
8 $\overline{\text{EXTINT}}$	7
6 $\overline{\text{CLEAR}}$	5 $\overline{\text{BI/ODATA1}}$
4 BI/ODATA2	3 GROUND
2 $\overline{\text{BMCLOCK}}$	1 BMCLOCK

Extended microI/O bus connection or bus terminator

1	BMCLOCK	20 $\overline{\text{BMCLOCK}}$
2	GROUND	19 BI/ODATA1
3	$\overline{\text{BI/ODATA1}}$	18 $\overline{\text{CLEAR}}$
4	INTPOUT1	17 EXTINT
5	$\overline{\text{EXTDCHR}}$	16 DCHPOUT1
6	GROUND	15 $\overline{\text{BI/ODATA2}}$
7	BI/ODATA2	14 GROUND
8	$\overline{\text{BI/OCLOCK}}$	13 BI/OCLOCK
9	NO CONN.	12 NO CONN.
10	GROUND	11 REM/PWR/ON

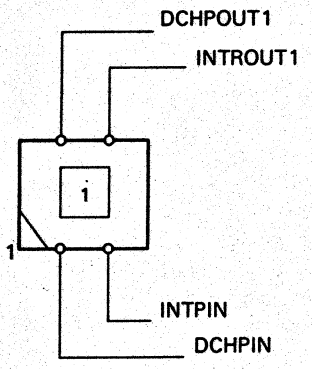
J1

PIN	COLUMN C	COLUMN B	COLUMN A
32	+5 PS #1	+ 5 PS #1	+ 5 PS #1
31	+5 PS #1	+ 5 PS #1	+ 5 PS #1
30	+5 PS #1	+ 5 PS #1	+ 5 PS #1
29	+5 PS #1	+ 5 PS #1	+ 5 PS #1
28	+5 PS #1	+ 5 PS #1	+ 5 PS #1
27	+5 PS #1	+ 5 PS #1	+ 5 PS #1
26	GROUND	GROUND	GROUND
25	GROUND	GROUND	GROUND
24	GROUND	GROUND	GROUND
23	GROUND	GROUND	GROUND
22	GROUND	GROUND	GROUND
21	GROUND	GROUND	GROUND
20	$\overline{\text{BMCLOCK}}$	BMCLOCK	GROUND
19	BI/ODATA1	$\overline{\text{BI/ODATA1}}$	DCHPIN
18	$\overline{\text{CLEAR}}$	INTPIN	$\overline{\text{EXTDCHR}}$
17	$\overline{\text{BI/ODATA2}}$	BI/ODATA2	$\overline{\text{EXTINT}}$
16	BI/OCLOCK	$\overline{\text{BI/OCLOCK}}$	GROUND
15	GROUND	GROUND	GROUND
14	GROUND	GROUND	GROUND
13	GROUND	GROUND	GROUND
12	+12 PS #1	+ 12 PS #1	− 12 PS #1
11	+12 PS #2	− 5 PS #2	− 5 PS #1
10	+12 PS #2	+ 12 PS #2	− 12 PS #2
9	GROUND	GROUND	GROUND
8	GROUND	GROUND	GROUND
7	GROUND	GROUND	GROUND
6	+5 PS #2	+ 5 PS #2	+ 5 PS #2
5	+5 PS #2	+ 5 PS #2	+ 5 PS #2
4	+5 PS #2	+ 5 PS #2	+ 5 PS #2
3	+5 PS #2	+ 5 PS #2	+ 5 PS #2
2	+5 PS #2	+ 5 PS #2	+ 5 PS #2
1	+5 PS #2	+ 5 PS #2	+ 5 PS #2

ID-00687

Figure 8-15 Backpanel pin assignments: diskette module

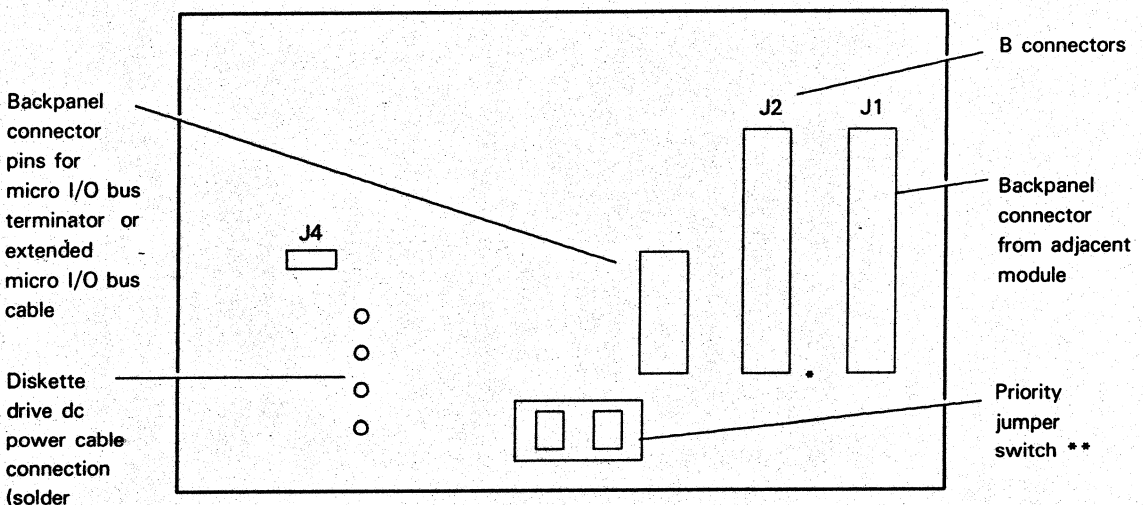
Priority jumper switch



Diskette drive
dc power cable connection

E1	-	+12 PS #1
E2	-	GROUND
E3	-	GROUND
E4	-	+5 PS #1
E5	-	NO CONNECTION
E6	-	GROUND

Actual layout of backpanel (viewed from front of module)



* Indicates pin 1 as viewed from front of chassis

** When card slot empty, switch is in closed position to extend priority chain

J3

PIN	COLUMN C	COLUMN B	COLUMN A
32	+5 PS #1	+5 PS #1	+5 PS #1
31	+5 PS #1	+5 PS #1	+5 PS #1
30	+5 PS #1	+5 PS #1	+5 PS #1
29	+5 PS #1	+5 PS #1	+5 PS #1
28	+5 PS #1	+5 PS #1	+5 PS #1
27	+5 PS #1	+5 PS #1	+5 PS #1
26	GROUND	GROUND	GROUND
25	GROUND	GROUND	GROUND
24	GROUND	GROUND	GROUND
23	GROUND	GROUND	GROUND
22	GROUND	GROUND	GROUND
21	GROUND	GROUND	GROUND
20	GROUND	BMCLOCK	$\overline{\text{BMCLOCK}}$
19	DCHPOUT1	$\overline{\text{BI/ODATA1}}$	BI/ODATA1
18	$\overline{\text{EXTDCHR}}$	INTPOUT1	$\overline{\text{CLEAR}}$
17	$\overline{\text{EXTINT}}$	BI/ODATA2	$\overline{\text{BI/ODATA2}}$
16	GROUND	$\overline{\text{BI/OCLOCK}}$	BI/OCLOCK
15	GROUND	GROUND	GROUND
14	GROUND	GROUND	GROUND
13	GROUND	GROUND	GROUND
12	−12 PS #1	+12 PS #1	+12 PS #1
11	−5 PS #1	−5 PS #2	+12 PS #2
10	−12 PS #2	+12 PS #2	+12 PS #2
9	GROUND	GROUND	GROUND
8	GROUND	GROUND	GROUND
7	GROUND	GROUND	GROUND
6	+5 PS #2	+5 PS #2	+5 PS #2
5	+5 PS #2	+5 PS #2	+5 PS #2
4	+5 PS #2	+5 PS #2	+5 PS #2
3	+5 PS #2	+5 PS #2	+5 PS #2
2	+5 PS #2	+5 PS #2	+5 PS #2
1	+5 PS #2	+5 PS #2	+5 PS #2

J1

PIN	COLUMN C	COLUMN B	COLUMN A
32	+5 PS #1	+5 PS #1	+5 PS #1
31	+5 PS #1	+5 PS #1	+5 PS #1
30	+5 PS #1	+5 PS #1	+5 PS #1
29	+5 PS #1	+5 PS #1	+5 PS #1
28	+5 PS #1	+5 PS #1	+5 PS #1
27	+5 PS #1	+5 PS #1	+5 PS #1
26	GROUND	GROUND	GROUND
25	GROUND	GROUND	GROUND
24	GROUND	GROUND	GROUND
23	GROUND	GROUND	GROUND
22	GROUND	GROUND	GROUND
21	GROUND	GROUND	GROUND
20	$\overline{\text{BMCLOCK}}$	BMCLOCK	GROUND
19	BI/ODATA1	$\overline{\text{BI/ODATA1}}$	DCHPIN
18	$\overline{\text{CLEAR}}$	INTPIN	$\overline{\text{EXTDCHR}}$
17	$\overline{\text{BI/ODATA2}}$	BI/ODATA2	$\overline{\text{EXTINT}}$
16	BI/OCLOCK	$\overline{\text{BI/OCLOCK}}$	GROUND
15	GROUND	GROUND	GROUND
14	GROUND	GROUND	GROUND
13	GROUND	GROUND	GROUND
12	+12 PS #1	+12 PS #1	−12 PS #1
11	+12 PS #2	−5 PS #2	−5 PS #1
10	+12 PS #2	+12 PS #2	−12 PS #2
9	GROUND	GROUND	GROUND
8	GROUND	GROUND	GROUND
7	GROUND	GROUND	GROUND
6	+5 PS #2	+5 PS #2	+5 PS #2
5	+5 PS #2	+5 PS #2	+5 PS #2
4	+5 PS #2	+5 PS #2	+5 PS #2
3	+5 PS #2	+5 PS #2	+5 PS #2
2	+5 PS #2	+5 PS #2	+5 PS #2
1	+5 PS #2	+5 PS #2	+5 PS #2

ID

Figure 8-16 Backpanel pin assignments: disk module

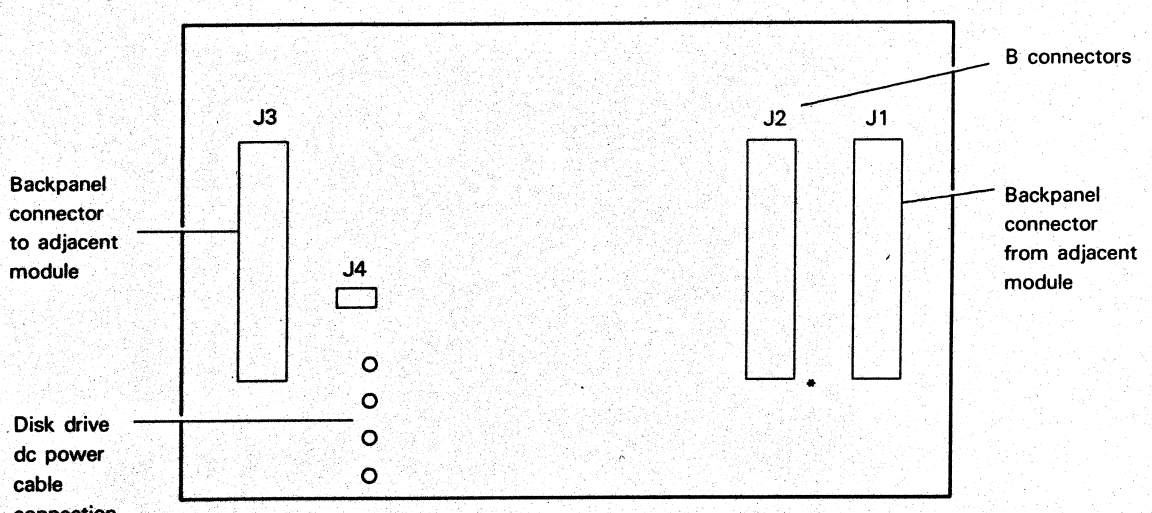
SLOT 1 - J2

EVEN PINS	ODD PINS
60 +5 PS #1	59 +5 PS #1
58 −5 PS #1	57 +5 PS #1
56 +12 PS #1	55 +12 PS #1
54 GROUND	53 GROUND
52	51
50	49
48	47
46	45
44	43
42	41
40	39 −12 PS #1
38	37
36 GROUND	35
34	33
32	31
30	29
28	27
26	25
24	23
22 DCHPIN	21 DCHPOUT1
20 INTPIN	19 INTPOUT1
18	17
16 BI/OCLOCK	15 $\overline{\text{BI/OCLOCK}}$
14 GROUND	13 BI/ODATA2
12 $\overline{\text{BI/ODATA2}}$	11 GROUND
10	9 $\overline{\text{EXTDCHR}}$
8 $\overline{\text{EXTINT}}$	7
6 $\overline{\text{CLEAR}}$	5 BI/ODATA1
4 BI/ODATA2	3 GROUND
2 $\overline{\text{BMCLOCK}}$	1 BMCLOCK

Disk drive
dc power cable connection

E1	-	+12 PS #1
E2	-	GROUND
E3	-	GROUND
E4	-	+5 PS #1
E5	-	NO CONNECTION
E6	-	GROUND

Actual layout of backpanel (viewed from front of module)



* Indicates pin 1 as viewed from front of chassis

Backpanel Priority Switches

When any backpanel slot between the Model 10 and Model 10/SP SPU card and the farthest I/O interface card is unused or used by a card other than an I/O interface card, priority switching is required to pass two priority signals of the I/O bus along the backpanel. These priority signals, interrupt priority (INTP), and data channel priority (DCHP) connect across a slot by closing a slot priority switch (up position) located on the front of the backpanels, as shown in Figure 8-13 through Figure 8-16. Four slot priority switches reside on the CLM and LEM backpanels. One slot priority switch is included on the FM backpanel. Closing a slot priority switch connects both the interrupt and data channel priority chains across the slot.

Power Switch

The power switch, located on the front of each power module (Figure 8-17), turns power on and off for the system unit. This switch does not remove power from a device plugged into the convenience outlet of the power module. When an expansion disk unit is present, its power switch works jointly with that of the system unit: both switches must be turned on to apply power to both units, and both switches must be turned off to remove power. The power switch for the tape module, when present, is located on the rear of that module (Figure 8-18).

NOTE *When a tape module is present and is to be powered up, it must be powered up before the system unit. Likewise, the tape module must not be powered down while the system unit is powered up.*

Figure 8-17 Power switch, power module

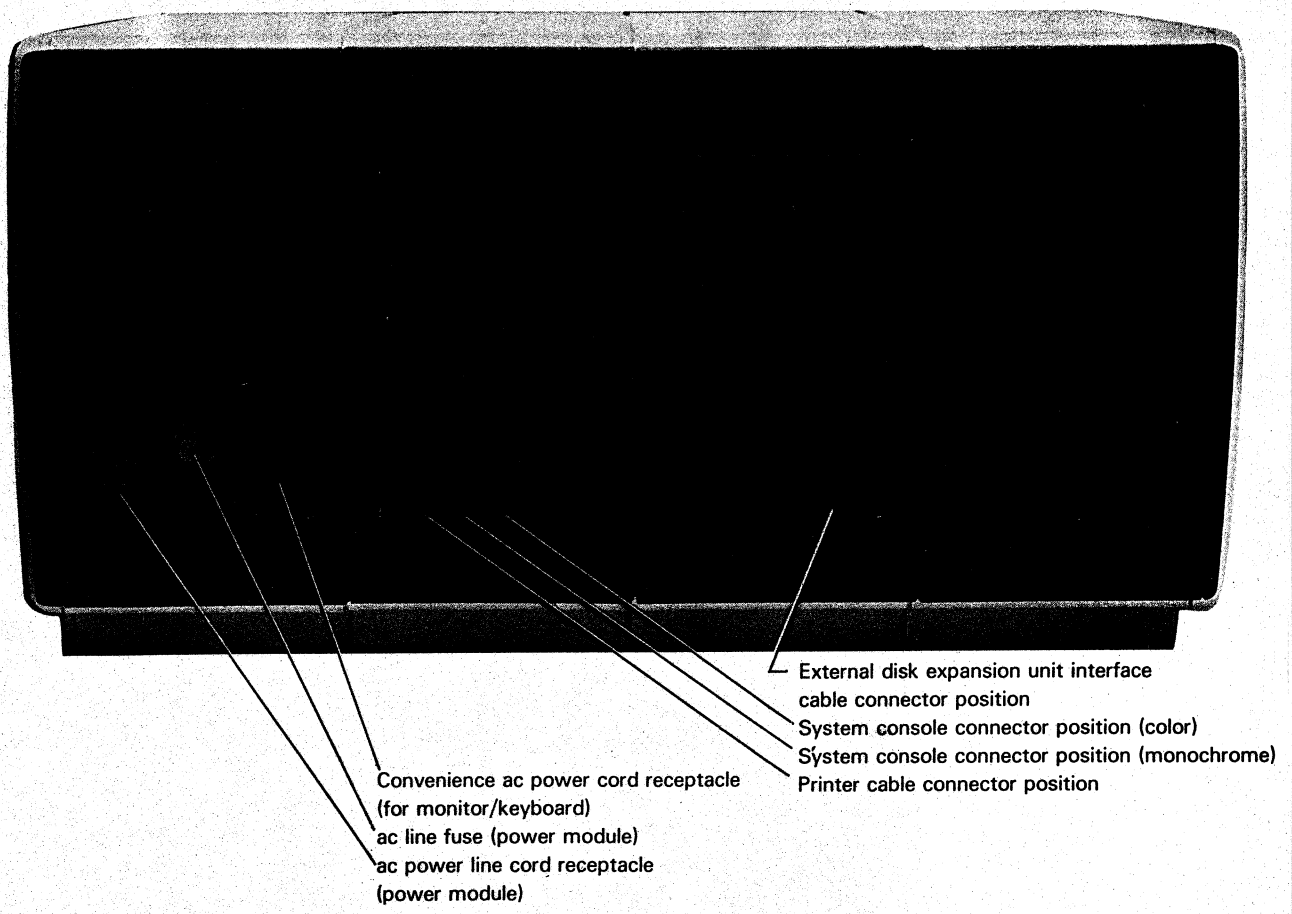
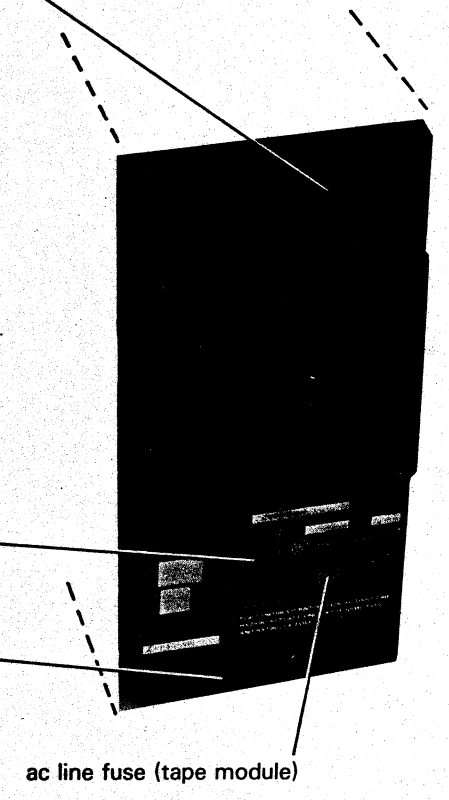


Figure 8-18 Models 10 or 10/SP system modules (rear view)



NOTES

1. microl/O bus terminator position when tape module is present and no external peripherals are connected to extended bus.
2. See Figure 8.10 for microl/O bus terminator position when neither the tape module nor internal microl/O bus expansion cable are present.

Figure 8-19 Tape module (rear view)

Line Fuses

Power fuses, located on the rear of each power module and the tape module as shown in (Figures 8-17 and 8-18), protects the ac line input. Line power to the Model 10 and Model 10/SP unit and to any device plugged into the power module's convenience outlet is applied through this fuse.

Power Interlock

The power interlock, located behind the front cover of each power module, removes ac power from the blower and power supply assemblies when the module's front cover is removed.

WARNING *Line power is present to the ac power connector and the convenience ac power connector (both located at the rear of the PM), as well as to the power interlock connector, when the ac power cord is plugged into its connector.*

System Cables

Line cords, internal SPU cable, system console cable, printer cable, diskette drive cable, and other I/O cables connect the system unit and their components to ac line voltage and input/output devices. In addition, when the Model 10 video interface is present, a cable connects it to the video monitor. Internal interface and power cables connect the cartridge tape, the diskette and disk drives to their respective interface cards and to dc power sources. An external interface cable connects the main disk module to the expansion disk module, when present. An internal microI/O bus cable connects the tape subsystem interface (when present) to the diskette module backpanel.

Line Cords

Line cords plug into connectors located at the rear of each power supply module and the tape module, when present, as shown in Figure 8-19. The line cord also supplies ac power, through the line fuse, to the device plugged into the power module convenience outlet.

System Console Cable

A system console cable connects the Model 10 and Model 10/SP SPU directly to the system console monitor and keyboard. One end of this cable contains a 50-pin connector that plugs onto the asynchronous interface port (A connector) of the SPU two card as shown in Figure 8-19. The other end contains a connector that plugs into a mating connector on the system console device. When the color monitor is used as the system console device the video interface interconnection cable described below is used to connect keyboard signals from the SPU-2 card to the video interface card.

Video Interface Interconnection Cable

This cable carries signals between the optional video interface and SPU-2. One end of this cable contains a 6-pin connector that plugs into a connector located on the front edge of the video interface card, as shown in Figure 8-20, and the other end contains a 50-pin connector that is attached to the A connector of the SPU-2 card.

Video Interface Monitor Cable

This cable carries signals between the optional video interface and the system console monitor and keyboard. One end of this cable contains a 50-pin connector that plugs onto the A connector of the video interface card, as shown in Figure 8-18, and the other end contains a connector that plugs into the video monitor.

Two-Card SPU Interconnection Cable This cable carries signals between the two SPU cards. Each end of this cable contains a 60 pin connector that plugs onto the D connectors of both cards, as shown in Figure 8-20.

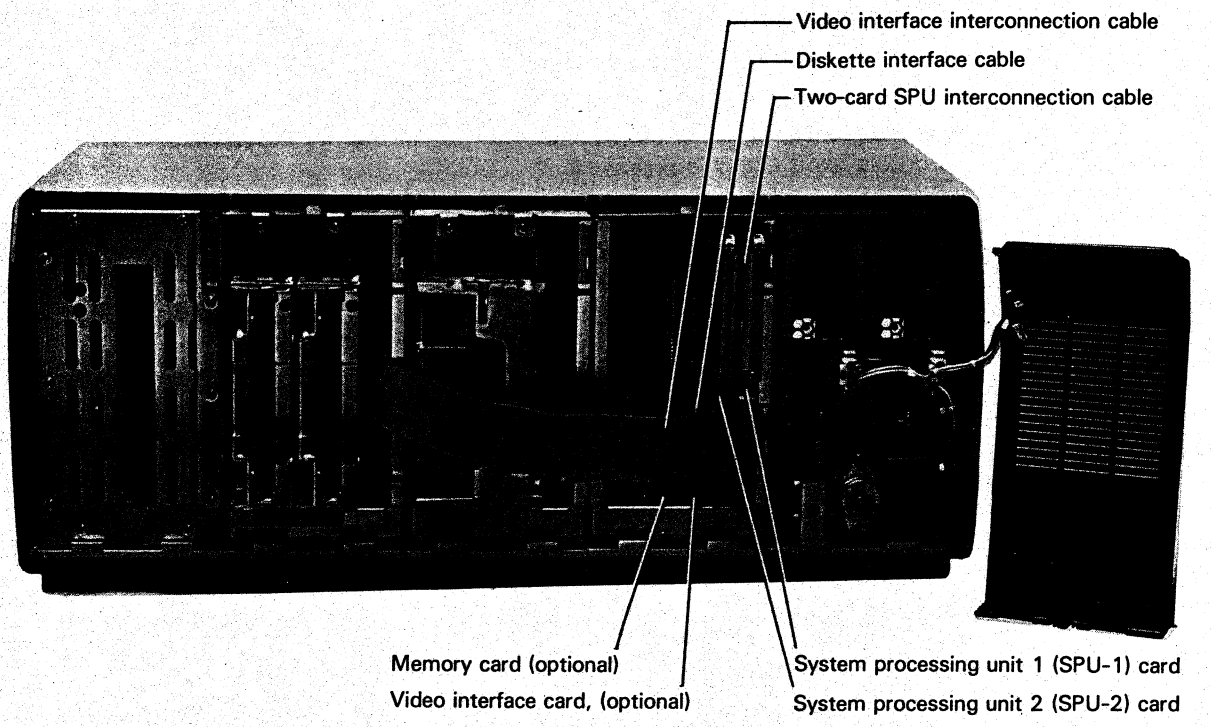


Figure 8-20 Models 10 or 10/SP system modules (front view) covers removed

Diskette Interface Cable The diskette interface cable connects the diskette interface contained on the SPU-1 card to one or two diskette drives. One end of this cable contains a 50 pin connector that plugs onto the C connector of the SPU-1 card, as shown in Figure 8-20. The other end contains two connectors that plug into mating connectors located on the rear of each diskette drive, as shown in Figure 8-21.

Diskette Power Cable The diskette power cable provides dc power to the diskette drive(s). One end of this cable is soldered directly to the diskette module backpanel, as shown in Figure 8-15. The other end contains two 4-pin connectors that plug into mating connectors located on the rear of each diskette drive, as shown in Figure 8-21. In addition, a safety ground wire connects J4 of the diskette module's backpanel (shown in Figure 8-15) to J3, located on the rear of each diskette drive (shown in Figure 8-21).

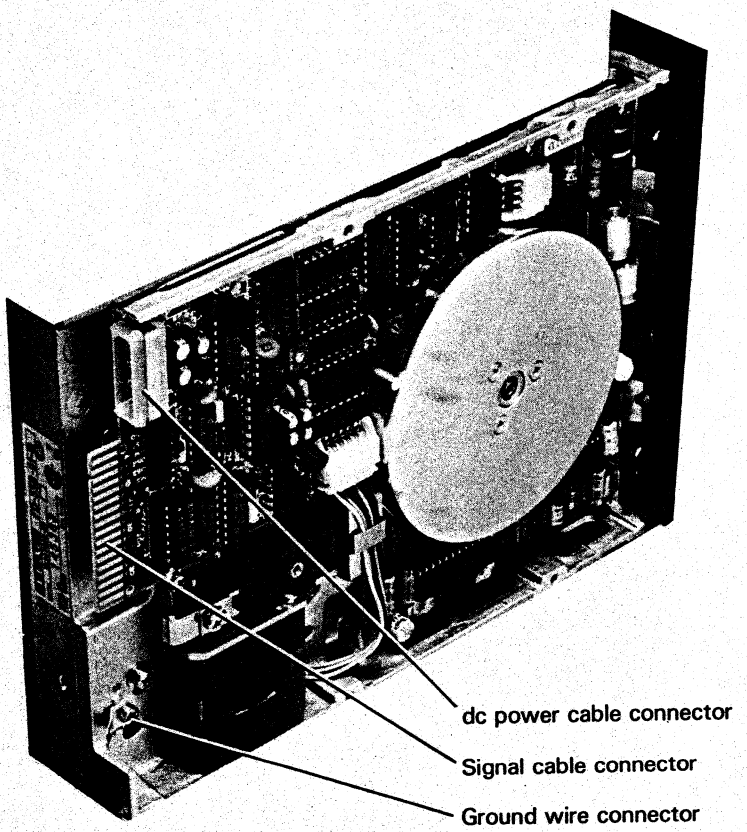


Figure 8-21 Diskette drive unit (rear view)

Disk Interface Cable The disk interface cable connects the optional fixed disk interface to the fixed disk drive. One end of this cable contains a 50-pin connector, with two flat-ribbon cables, that plugs onto the A connector of the disk interface card as shown in Figure 8-22. One flat-ribbon cable contains a connector that plugs into a mating connector located on the rear of the disk drive, as shown in Figure 8-22. The second cable contains a connector that mounts to the rear of the module's cage to facilitate connection to the expansion disk unit, when present.

Disk Power Cable The disk power cable provides dc power to the fixed disk drive. One end of this cable is soldered directly to the DM backpanel, as shown in Figure 8-16. The other end contains a connector that plugs into a mating connector located on the rear of the disk drive, as shown in Figure 8-22. In addition, a ground wire connects J4 of the disk module's backpanel (shown in Figure 8-16) to a terminal lug located on the rear of the disk drive (shown in Figure 8-22). The power cable for the expansion disk drive contains a connector at one end that plugs into a load card contained in the expansion disk drive module. The other end contains a connector that plugs into a mating connector located on the rear of the disk drive, as shown in Figure 8-22. The load card connects to the expansion power supply assembly by a dc power cable. A ground wire also connects between the load card and the expansion disk drive.

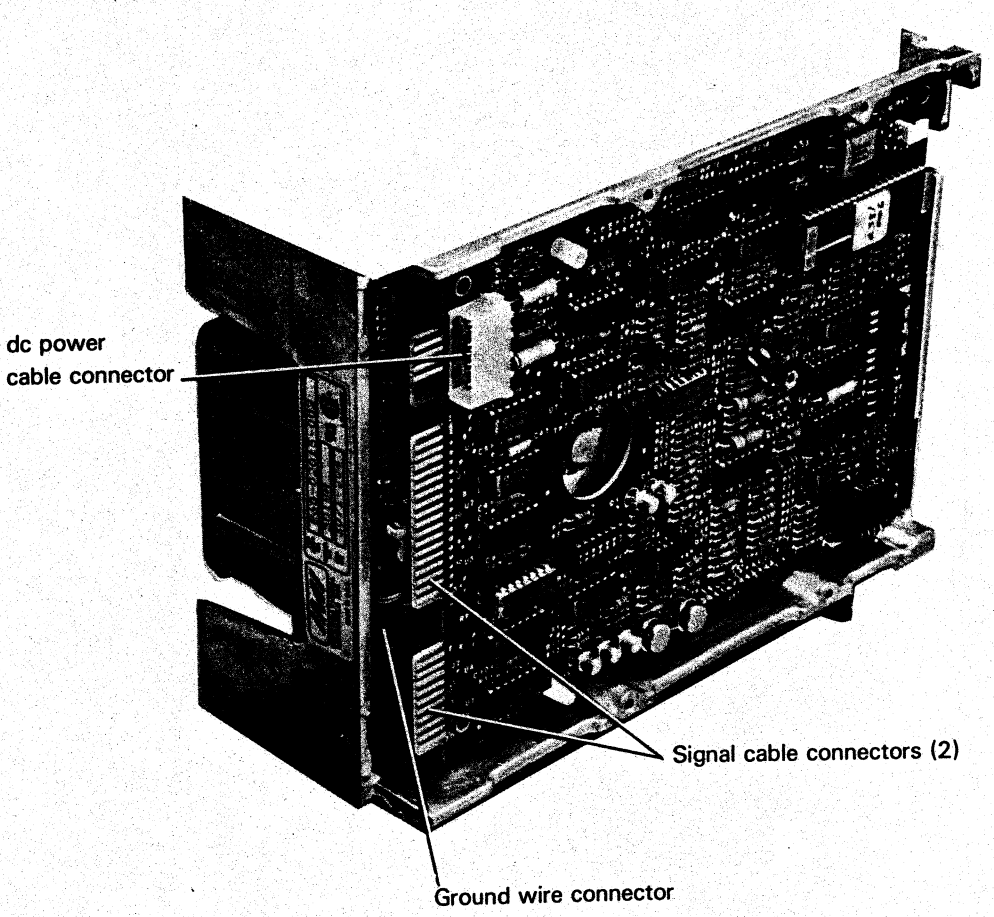


Figure 8-22 Disk drive unit (rear view)

Disk Expansion Unit Cable

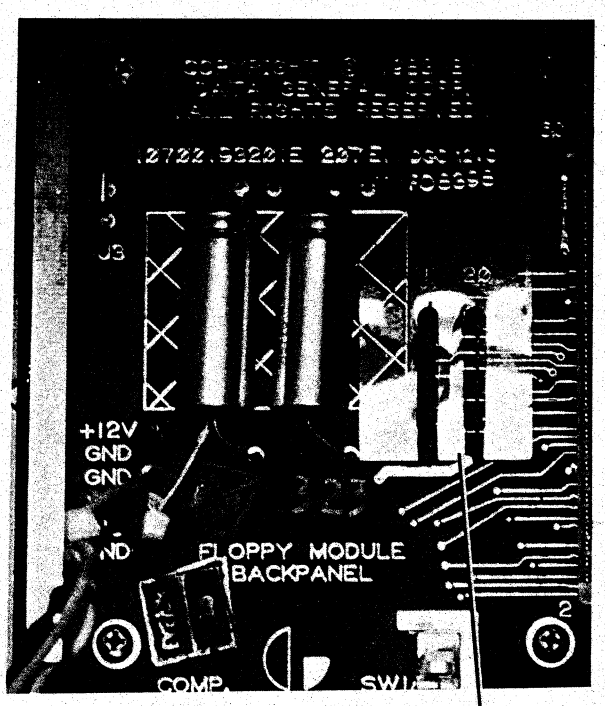
The system's main disk module electrically connects to an expansion disk module (when present) with three interface cables. One of these cables is the disk interface cable described above that is located in the main disk module.

A second interface cable contains 50-pin connectors at both ends. Each end of this cable plugs into connectors located on the rear of the main disk and expansion disk modules. Figure 8-19 shows this cable connector position on the disk module.

A third interface cable located in the expansion disk module, contains two connectors: one connector is mounted on the rear of the expansion module's cage; the other connector plugs into the expansion disk drive, as shown in Figure 8-20.

Tape Module I/O Bus Cable

This internal cable, when present, extends the microI/O bus from the diskette module to the tape controller located in the tape module. One end contains a 16-pin DIP connector that plugs onto pins located on the diskette module backpanel as shown in Figure 8-23 and Figure 8-15. (This connector plugs onto the upper 16 pins of the 20 pins provided on the backpanel.) The other end contains a connector that plugs onto the tape controller card.



Extended microI/O bus cable connector position (tape module I/O bus connector plugs onto top 16 pins of the 20 pin provided)

DG-25945

Figure 8-23 Diskette module, internal view of backplane

Input/Output Cables

Input/output cables connect system I/O interfaces to the device they control. One end of these cables contains a 50-pin connector that plugs onto the A connector of the respective I/O interface card, as shown in Figure 8-24. The other end contains a connector that plugs into a mating connector on the device the interface is controlling.

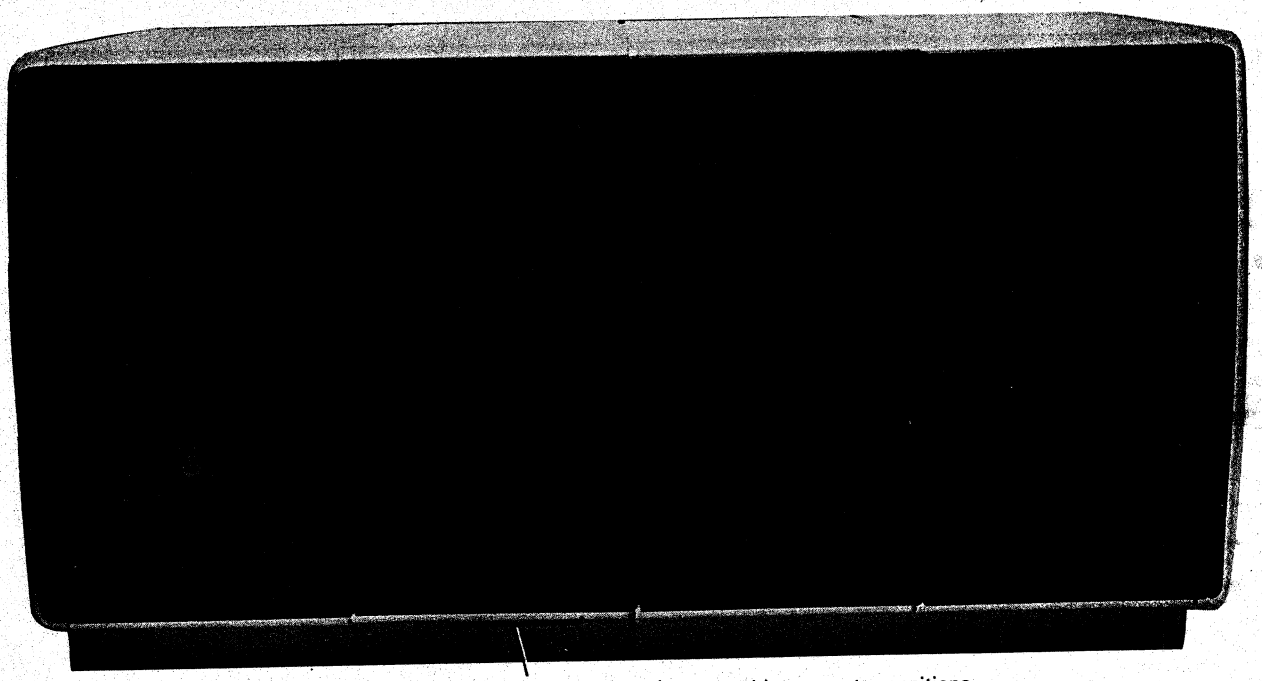


Figure 8-24 Input/output cable connector positions

Related Drawings

A

The table below lists Model 10 and Model 10/SP engineering drawings

Part Number	Part	Schematic	Illustrated Parts List (Wiring List)
005-020621	System Processor Unit #1 (SPU1) card without firmware floating point	001-003677	016-001596
005-020626	System Processor Unit #1 card with firmware floating point	001-003677	016-001596
005-020623	System Processor Unit #2 (SPU2) card	001-003678	
005-020375	Optional 256-Kbyte Expansion Memory card	001-003673	
005-020349	Optional 512-Kbyte Expansion Memory card	001-003673	
005-019594	Optional Color Video Interface card	001-003342	016-001454
005-019588	SPU1 to SPU2 interconnection cable	—	(018-001522)
005-019373	SPU to monochrome monitor (system console) cable	001-003436	(018-001492)
005-019922	SPU to video interface cable (used when color monitor is system console)	—	(018-001900)
005-019913	Video interface to color monitor cable	001-003437	(018-001493)
005-019593	Diskette module backpanel	001-003345	016-001457
005-019332	CPU to diskette cable	001-003421	(018-001475)
005-019331	Diskette power cable	—	(018-001488)
005-019591	Diskette drive assembly	—	—
005-008152	microI/O bus terminator	001-001027	016-000345
005-020331	Power supply PCB, 100V	001-003322	016-001465
005-020385	Power supply control PCB, 100V	001-003357	016-001466
005-020620	Power supply PCB, 120V and 220/240V	001-003322	016-001465

005-019683	Power supply control PCB, 120V and 220/240V	001-003357	016-001466
005-019299	CPU logic module backpanel	001-003344	016-001456
005-019337	dc power cable	001-003433	(018-001487)
005-021050	Logic expansion module backpanel	001-003418	016-001474
005-019347	Disk module backpanel	001-003401	016-001472
005-019426	Disk interface card	001-003306	016-001441
005-019915	Disk data cable	001-003417	(018-001473)
005-019914	Disk power cable	—	(018-001489)
005-019344	Disk drive assembly, 5 Mbytes	—	—
005-019346	Disk drive assembly, 15 Mbytes	—	—

Diskette Diagnostic Commands

B

This appendix explains the special diagnostic commands that can be carried out under program control. It may help you to understand the diagnostic programs if you are troubleshooting the Model 10 or 10/SP diskette subsystem. It is not the intent of this section to explain how the commands are used or how the hardware works.

A diagnostic operation is performed when a diagnostic command is specified by the command field (bits 4-6) of the accumulator transferred to the diskette interface during a *Specify Command and Diskette Address* instruction (DOA). The diagnostic operation to be performed is encoded in bits 11-15 of the same accumulator.

Diagnostic commands override other commands and may redefine the accumulator formats of the programmed instructions. Diagnostic commands and a brief description of their operations are listed in Table B-1.

Table B-1 Diagnostic commands

Bits 11-15	Command	Description
00000	Read Controller Type	Sets bits 8-15 of the interface status register to zeros. Diskette diagnostics interrogate these bits to determine the interface hardware implementations to test.
00001	NOP	No operation is performed. The interface Done flag is set.
00010	NOP	No operation is performed. The interface Done flag is set.
00011	Read Drive Type Bits	Loads the diskette drive configuration switches of the interface card into bits 8-15 of the diskette interface status register[1]. Bits 10 and 11 define them. When both bits are 0, drive is double-sided, 48 TPI; when bit 10 is 0 and bit 11 is 1, drive is double-sided, 96 TPI.
00100	Write DIA Bits 8-15	Loads bits 0-3 and 7-10 of the command register into bits 8-15 of the interface status register, and into bits 0-7 and 8-15 of the diskette interface word count register.[1]
00101	Step In	Positions the head(s) of the selected drive to the next higher-numbered track and increments the current cylinder register.
00110	Step Out	Positions the heads of the selected drive to the next lower-numbered track and decrements the current cylinder register.
00111	Start Data Channel	Simulates a data channel facility data transfer. When bit 8 of the *Specify Command and Diskette Address* instruction that issued this diagnostic operation is 1, a word will be transferred from memory to the interface buffer; when bit 8 is 0, a word will be transferred from the interface buffer to memory. The interface contains only a single-byte buffer for use by this operation. Thus only the least significant byte transferred on a data channel out operation will be stored. A subsequent data channel in operation will receive this least significant byte in both bytes of the word received. The interface byte swapping feature can be enabled to test the most significant byte. Prior to issuing this command, the memory address register should be loaded with a memory address and the word count register with -1.
01000	NOP	No operation is performed. The interface Done flag is set.
01001	Read DOC	Loads bits 2-9 of the interface word count Bits 2-9 register into bits 8-15 of the interface status register[1]. These bits contain the portion of the word count register that specify the number of sectors to transfer.
01010	NOP	No operation is performed. The interface Done flag is set.
01011	Read Control Program Revision Number	Loads the revision number of firmware in the interface microprocessor into bits 8-15 of the interface status register.[1]
01100	Read Current Track Address	Loads the current track address for the selected drive into bits 8-15 of the interface status register.[1]
01101	Read Drive Mode Bits	Loads the interface mode byte into bits 8-15 of the interface status register.[1] (Refer to Table 2-5 for description of the interface mode bits.)
01110	NOP	No operation is performed. The interface Done flag is set.
01111	NOP	No operation is performed. The interface Done flag is set.
10000	NOP	No operation is performed. The interface Done flag is set.

10001	NOP	No operation is performed. The interface Done flag is set.
10010	NOP	No operation is performed. The interface Done flag is set.
10011	NOP	No operation is performed. The interface Done flag is set.
10100	NOP	No operation is performed. The interface Done flag is set.
10101	Write Diskette Controller Track Register	Loads bits 0-3 and 7-10 of the interface command register into bits 8-15 of the controller track register.[1]
10110	Write Diskette Controller Sector Register	Loads bits 0-3 and 7-10 of the interface command register into bits 8-15 of the controller sector register.[1]
10111	Write Diskette Controller Data Register	Loads bits 0-3 and 7-10 of the interface command register into bits 8-15 of the controller data register.[1]
11000	Read Diskette Controller Status Register	Loads bits 7-0 of the controller status register into bits 8-15 of the interface status register.[1]
11001	Read Diskette Controller Track Register	Loads bits 7-0 of the controller track register into bits 8-15 of the interface status register.[1]
11010	Read Diskette Controller Sector Register	Loads bits 7-0 of the controller sector register into bits 8-15 of the interface status register.[1]
11011	Read Diskette Controller Data Register	Loads bits 7-0 of the controller data register into bits 8-15 of the interface status register.[1]
11100	Read DOA Bits 8-15	Loads bits 8-15 of the interface command register into bits 8-15 of the interface status register.[1]
11101	Read DOA Bits 0-7	Loads bits 0-7of the interface command register into bits 8-15 of the interface status register.[1]
11110	Read Self-Test Results	Loads a two-hexadecimal digit code that specifies the results of the interface self-test into bits 8-15 of the controller status register.[1] The following hexadecimal codes specify the self-test results: 00 = no error 01 = failure in microprocessor RAM test 02 = failure in microprocessor ROM test 03 = failure in diskette controller chip test
11111	Run Self-Test	Initiates a subset of the power-up self-test. The results of this test can be determined by reading the diskette status register and interrogating the Not OK status bit. (See Read Self-Test Results above to determine the cause of a failure specified by a Not OK status condition.)

[1] *Diagnostic commands that load information into the interface status register do not cause the information loaded to be returned to the SPU. The diagnostic command must be followed by a* Read Diskette Status *instruction (DIA) to return the information to the SPU.*

ATP Interface Programming Example

C

This example is only meant to be informative: it is not intended to be a recommendation.

```
          NIOC ATP                  ;Clear the attached processor functions
                                    ;   perform a hard reset on the 8086.
          LDA     1,WRDCNT          ;Get the word count of logical/physical pairs
          ELEF    2,LMPATB          ;Get the address of the LMPA table
          LMPA                      ;Perform a LMPA. (Load MAP translation
                                    ;   unit.)
STRATP:   SUB     3,3               ;ac3 = 0
          DOAS    3,ATP             ;Start the attached processor, no execution of
                                    ;   microECLIPSE code
                                    ;   occurs until 8086 stops. ac3
                                    ;   specifies no interrupt on Done = 1.
ATPRTN:   SKPBZ   ATP               ;8086 stopped at this point and the
                                    ;   microECLIPSE has been restarted.
                                    ;   If a microI/O interrupt had occurred, the
                                    ;   user interrupt handler return would be
                                    ;   to this point.
                                    ;Otherwise, the ATP is done now.
                                    ;   Check if ATP is busy (interrupted).
          JMP     STRATP            ;Busy = 1. The ATP must have been
                                    ;   interrupted. Jump to restart ATP at its last
                                    ;   (held) address.
          SKPDN   ATP               ;Check to see if ATP is done.
          JMP     ATPERROR          ;ATP is not busy or done: error.
          DIA     0,ATP             ;Read the attached processor status.
                                    ;   (Note that at this point, the attached
                                    ;   processor must be done, that is, Done flag
                                    ;   = 1.)
          MOVZR   0,0               ;Did a validity fault occur (bit 14 = 1)?
```

```
        MOVR #    0,0,SZC     ;Did an OUT instruction occur?
                              ;No: return from microI/O interrupt
        JMP       VALFT       ;Yes: handle the validity fault.
        JMP       STRATP      ;and restart the 8086 at its last (held)
                              ;  address.
        LDA       0,ATPIV     ;Yes: handle the OUT instruction call (can be
                              ;  a system call), perhaps meant to start the
                              ;  8086 at a new address, so get 8086
                              ;  interrupt vector.
        DOBS      0,ATP       ;Send interrupt vector to 8086 and pend an
                              ;  8086 interrupt. Start command then starts
                              ;  the 8086. No execution of microECLIPSE
                              ;  code occurs until the 8086 stops.
        JMP       ATPRTN      ;8086 stopped at this point and the
                              ;  microECLIPSE has been restarted. Handle
                              ;  by going to ATP RTN code above.
WRDCNT: 7                     ;Number of translation pairs for ATP MAP
                              ;  (via LMPA instruction).
LMPATB: 000000                ;Logical page 0000 mapped to word pair 1
        000222                ;  physical page 0222
        000042                ;Logical page 42 mapped to word pair 2
        001646                ;  physical page 1646
        000043                ;Logical page 43 mapped to word pair 3
        001732                ;  physical page 1732
        000044                ;Logical page 44 mapped to word pair 4
        000032                ;  physical page 0032
        000045                ;Logical page 45 mapped to word pair 5
        000732                ;  physical page 0732
        000046                ;Logical page 46 mapped to word pair 6
        001777                ;  physical page 1777
        000012                ;Logical page 1012 is to be word pair 7
        101777                ;  validity protected
```

NOTE *All the numbers in the example are octal.*

Memory Formats for Bit-Mapped Screen Buffer

D

The information presented here is primarily intended to clarify the relationship between the data stored in memory and the picture elements (pixels) painted on the monochrome monitor screen and to indicate how the CRT chip and character attributes can be controlled. This appendix is not intended to be a guide to programming the monochrome monitor screen display. The reader should refer to *Model 10 and 10/SP System Console Programmer's Reference* for programming information. The programmer's reference tells the user how to use the downloaded D200/210 keyboard emulator and the downloaded graphics command interpreter and how to write directly into the graphic screen buffer.

Memory Mapping the Screen Buffer

Figure D-1 illustrates how a user might go about setting up a memory map for a screen buffer area in 64-kilobyte (32-kiloword) user memory. Chapter 1 gave the explicit series of steps required to do the setup. In Figure D-1, the user has chosen user location 30000 (map page 14) to be the start of the screen buffer. User pages 14-25 (10 decimal pages in all) are mapped to physical pages 1760-1771, the actual screen buffer pages. (If one looked at the addresses on the memory bus, one would see that the electrical addresses on that bus correspond to addresses for pages 0-11.) Location 30000 has been chosen in the example because that happens to be the location that control memory code uses for its screen buffer start — any other value could be chosen, and the pages used do not necessarily have to be contiguous. (The algorithm to follow would need to be modified if they were not contiguous.)

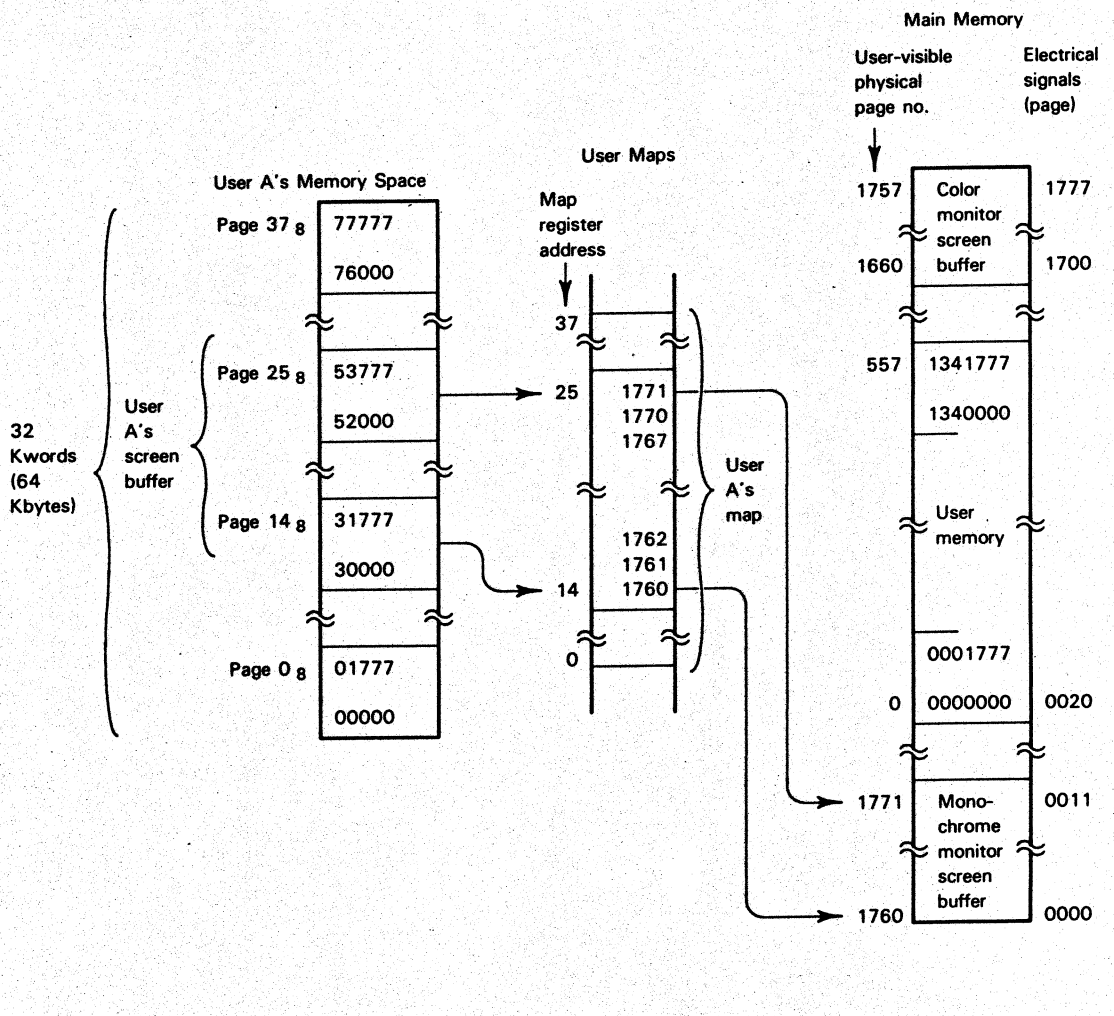
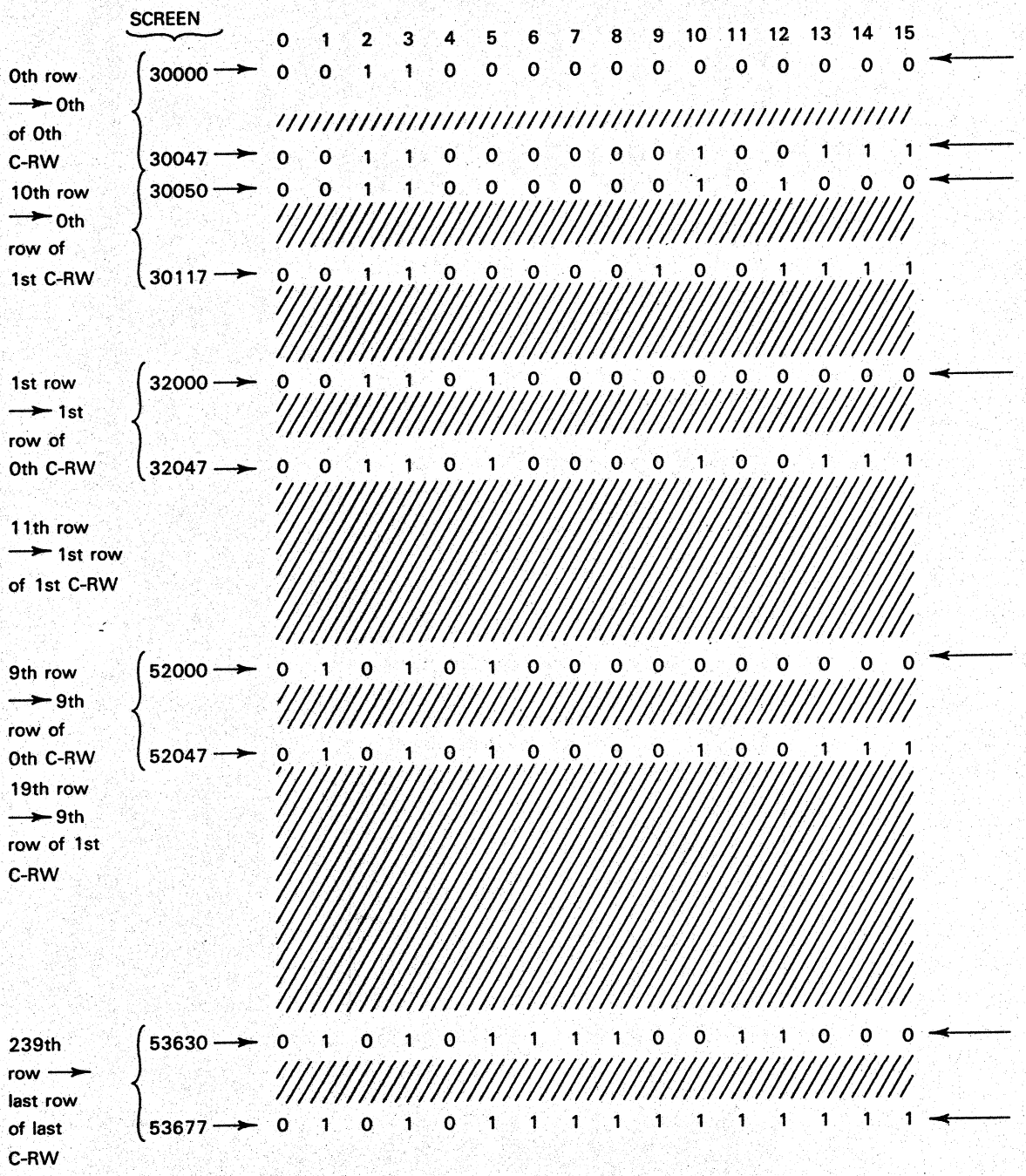


Figure D-1 Typical screen buffer map

Mapping Screen Pixels to the Buffer

Within the screen buffer, each bit of each memory word that is used corresponds to one picture element (pixel) on the screen. Because of the way the CRT controller chip generates the addresses of the words it finds the pixel data in, a six-step algorithm must be used to pack the screen buffer. Figure D-2 illustrates the addressing scheme used for this buffer.



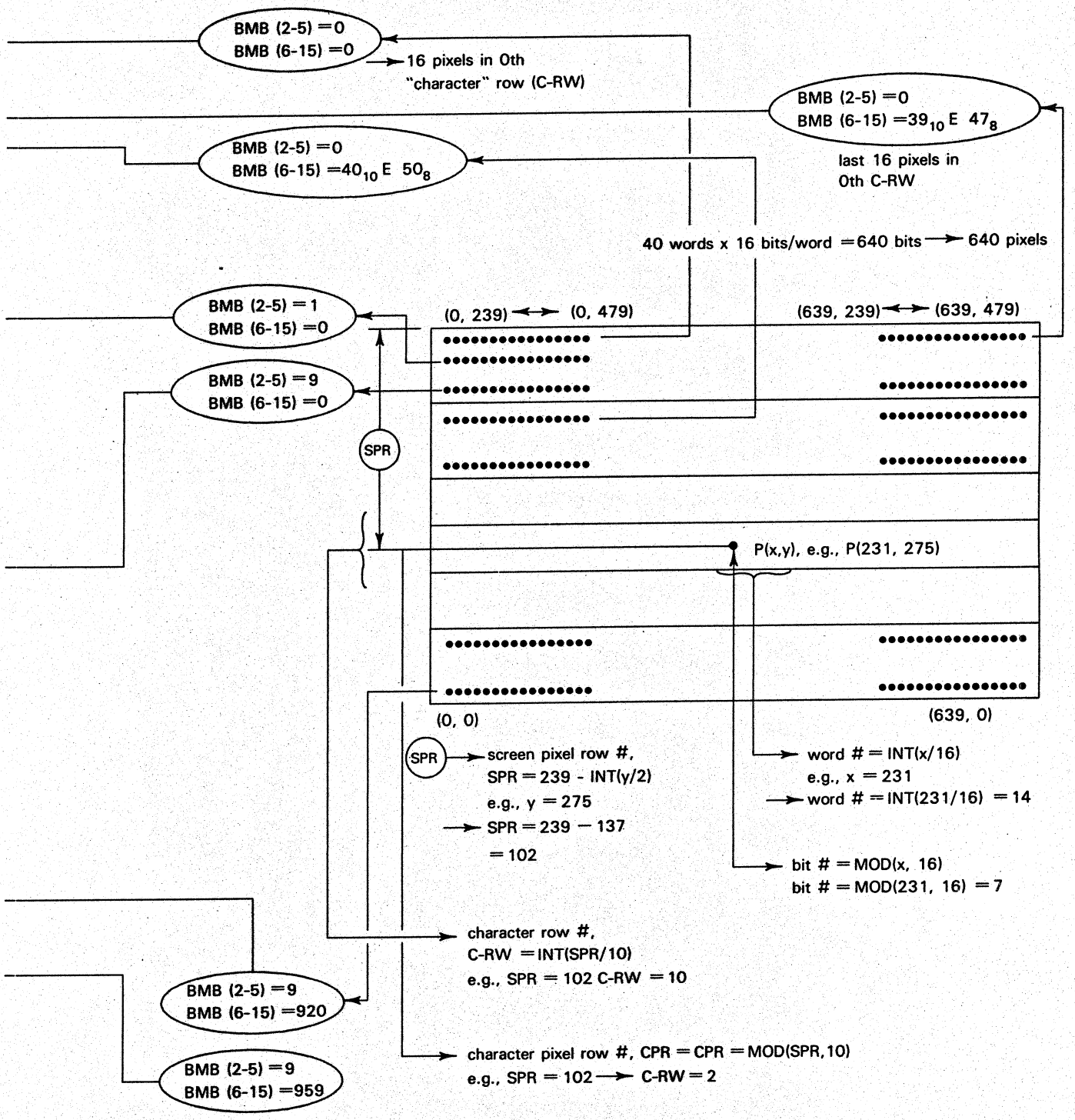
character row # word # character pixel row #

WA = SCREEN + WORDADDR + 40*[(239 - Y/2)/10] + X/16 + 1024* MOD[(239 - Y/2), 101

Bit address in word of pixel = bit #

ID-00746

Figure D-2 Screen organization


BMB (2-5) =0
BMB (6-15) =0
16 pixels in 0th "character" row (C-RW)
BMB (2-5) =0
BMB (6-15) =39₁₀ E 47₈
last 16 pixels in 0th C-RW
BMB (2-5) =0
BMB (6-15) =40₁₀ E 50₈
40 words x 16 bits/word =640 bits → 640 pixels
BMB (2-5) =1
BMB (6-15) =0
(0, 239) ↔ (0, 479)
(639, 239) ↔ (639, 479)
BMB (2-5) =9
BMB (6-15) =0
SPR
P(x,y), e.g., P(231, 275)
(0, 0)
(639, 0)
SPR → screen pixel row #,
SPR = 239 - INT(y/2)
e.g., y = 275
→ SPR = 239 − 137
= 102
→ word # = INT(x/16)
e.g., x = 231
→ word # = INT(231/16) = 14
→ bit # = MOD(x, 16)
bit # = MOD(231, 16) = 7
character row #,
C-RW = INT(SPR/10)
e.g., SPR = 102 C-RW = 10
BMB (2-5) =9
BMB (6-15) =920
BMB (2-5) =9
BMB (6-15) =959
character pixel row #, CPR = CPR = MOD(SPR,10)
e.g., SPR = 102 → C-RW = 2

As the figure shows, the location of a pixel on the video screen can be determined by an *xy*-coordinate system. The usual practice is to use the lower lefthand corner as the origin, point (0,0). The screen is 640 pixels wide, so the maximum value for *x* is 639. The screen is actually 240 pixels high, but because of the size of the space between adjacent horizontal pixels as compared with the size of the space between adjacent vertical pixels, a vertical scale of 0 to 480 is used. Using this vertical scale, the ordinary algebraic manipulation of the pixel coordinates produce circles that are round; not doubling the vertical scale causes elliptical circles. The maximum value for *y* is 479.

The logic in the CRT controller's address generator was designed to address locations in a ROM to produce bit patterns for characters. For this reason, it produce addresses arranged in rows as illustrated in Figure D-2. The word in the top left corner of the screen is the first to be displayed, and therefore the first to be addressed by the CRT controller.

The value of the address of the beginning of the screen buffer is called SCREEN. The character start address that is programmed into the CRT controller is called WORDADDR. It is taken in the example of Figure D-2 to be 0. The word address of a pixel with coordinates (*x*,*y*) is shown in the figure as WA, and the bit address within the word is shown as bit #. The general procedure for manipulating a pixel in the screen buffer is, then:

1. Calculate the number of the memory word within a screen pixel row (word # = *x*/16) and the bit number within the word (bit # = MOD(*x*, 16)) by shifts and masks.
2. Calculate the row number of the pixel on the screen (taking into account the vertical scaling and the reversed pixel numbering with SPR = 239 - INT(*y*/2)) by shifting and subtracting.
3. Calculate the row number of a pixel within a character row (CPR = MOD(SPR,10) = MOD[(239 - INT(*y*/2),10]) by a shift and mask operation.
4. Calculate the value of address bits 6-15 by multiplying the character row number by 40 (40*C-RW = 40*INT(SPR/10) = 40*INT[239 − INT(*y*/2)/10].)
5. Calculate the word address (WA) by adding all the pieces together.
6. Load an accumulator with the appropriate pixel mask (a 1 out of 16 code) by table lookup on the bit address (bit # = MOD(x/16).)

You can then perform the desired operation with the pixel mask applied to the word whose address was calculated. For example, to set a bit, use

(WA[X,Y]) = (WA[X,Y]) OR (BITTABLE + bit #)

To reset a bit the operation is an AND, and to complement a bit, the operation is XOR.

BITTABLE is the starting address of a table of 16 constants, namely, 2**15, 2**14, 2**13, and so on. This will have the result that the most significant bit of a word appears leftmost on the screen.

In the example, the starting word address WORDADDR, is 0. To set up the CRT controller so that this is the case, a user program must write a 0 into CRT registers 12 and 13. The following sequence of instructions will accomplish this:

```
R12:     12
R13:     13
CRTREG:  055677
CRTADD:  055676
         LDA 1,R12          ;AC1 SELECTS CRT REGISTER 12
         ELEF 0,CRTADD,0    ;AC0 = 55676 = DESTINATION OF WHYP
         WHYP               ;WRITE REG.ADDR.(12) INTO CRT ADDR.REG.
         SUB 1,1            ;AC1 = 0
         ELEF 0,CRTREG,0    ;AC0 = 55677 = DESTINATION OF WHYP
         WHYP               ;WRITE 0 INTO CRT REG.12
         LDA 1,R13          ;AC1 SELECTS CRT REGISTER 13
         ELEF 0,CRTADD,0    ;AC0 = 55676 = DESTINATION OF WHYP
         WHYP               ;WRITE REG.ADDR.(13) INTO CRT ADDR.REG.
         SUB 1,1            ;AC1 = 0
         ELEF 0,CRTREG,0    ;AC0 = 55677 = DESTINATION OF WHYP
         WHYP               ;WRITE 0 INTO CRT REG.13
```

The WHYP instruction is coded as 64003.

Character Cell Attributes

The data that controls the character attributes of a Model 10 or 10/SP computer's monochrome monitor resides in a dedicated 4-bit memory. Each location stores the completely independent attribute functions of blink, dim, reverse video, and underscore for one character cell. The CRT controller logic retrieves the attributes in synchronization with the bit-mapped character data in the videc screen buffer and appropriately modifies the character video signals. The CRT controller chip produces a hardware cursor that is enabled and gated to blink with the same clock phasing as characters. The cursor, when on, unconditionally inverts all pixels in the character cell at the cursor's location.

Attribute Addressing

Attribute nibbles (4-bit data) are ordered in an unpacked linear array of words that are accessed by two special instructions, whose formats are presented further on. The absolute address of the attribute modifies the character at a given screen position is:

$$A = H + C + (R * W) \quad \text{mod } 2^{**}L \quad + B$$

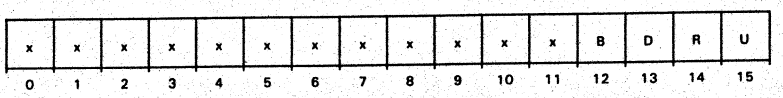
where: H = the current buffer address of the home position
C = the current column position [0,...,W-1]
R = the current row position [0,...]
W = the width of the screen in character columns
L = the number of bits allocated to the buffer address
B = the absolute address of the character buffer

The standard Model 10 or 10/SP display is configured as follows:

W = 80
L = 11
B = 10000

Attribute Data

Attribute data is stored in the following format (positive true):



B = blink
D = dim
R = reverse video
U = underscore

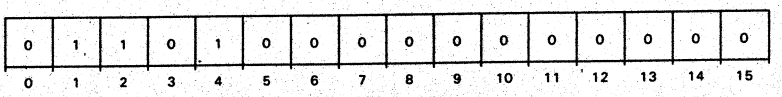
An x indicates that the bit is a "do not care" for write operations and a zero (0) when it is read.

Attribute Control Instructions

Two instructions control character attribute data and do so by using AC0 and AC1.

Read Character Attribute

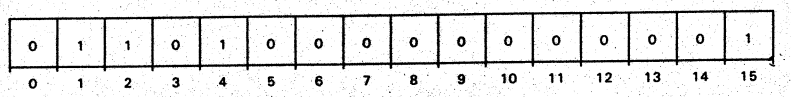
NIO *AC1,0*



AC0 contains the address of the character attribute to be read and AC1 will contain the attribute data after the read operation.

Write Character Attribute

NIO *AC1,1*



AC0 contains the address of the character attribute to be written and AC1 will contain the attribute data to be written.

Attribute Control

Use of the *Write Character Attribute* instruction accompanied by any arbitrary data in AC1, will control the independent blink enables of attribute and cursor video when used with the following address in AC0:

20000	Attribute Blink Disable
20001	Attribute Blink Enable
20002	Cursor Blink Disable
20003	Cursor Blink Enable

Model 10 and 10/SP systems use character attribute addresses 10000 through 37777 for data and control.

Index

Within the index, the letter "f" following a page entry indicates "and the following page"; the letters "ff" following a page entry indicate "and the following pages".

A

D

E

J

K

L

M

N

O

P

R

S

T