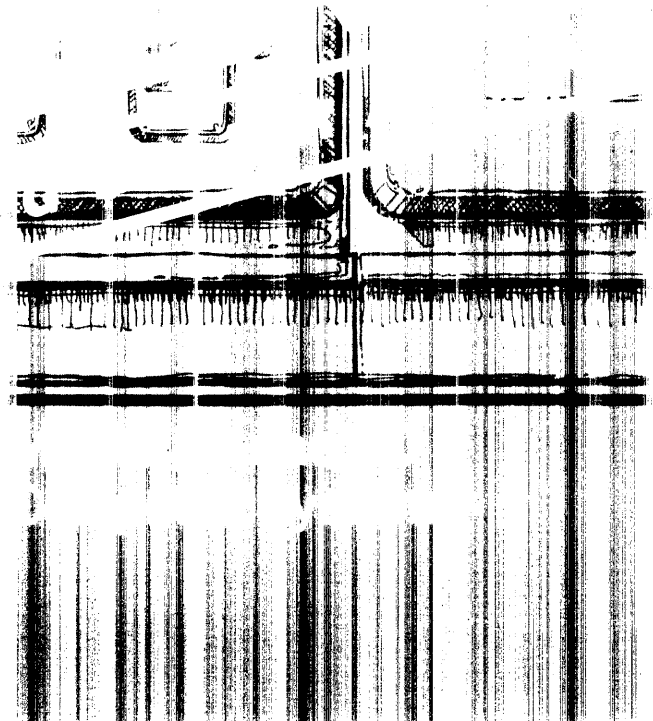
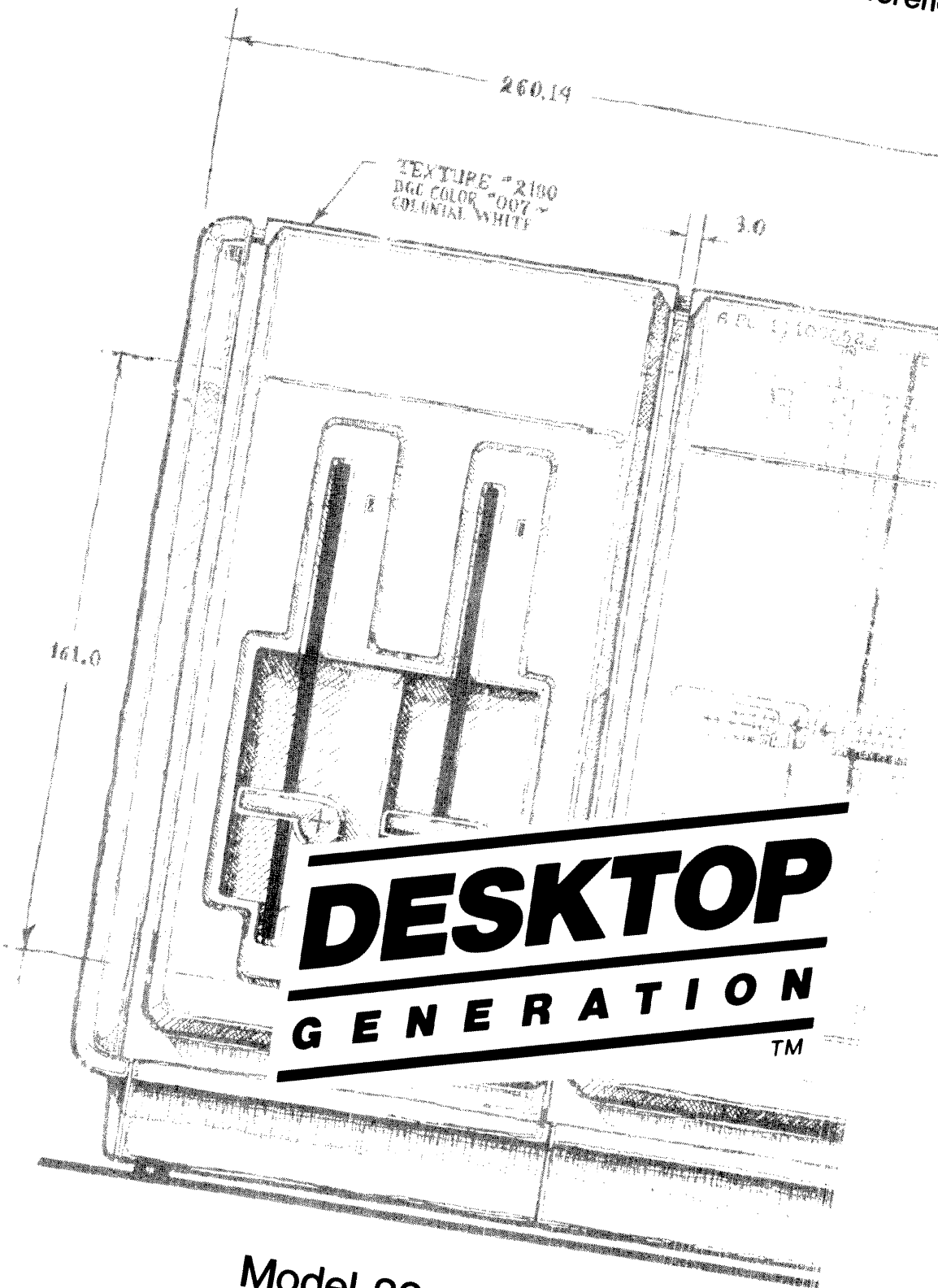


Data General

Technical Referen



Model 20 and 30
Computer Systems

Notice

Data General Corporation (DGC) has prepared this document for use by DGC personnel, customers, and prospective customers. The information contained herein shall not be reproduced in whole or in part without DGC's prior written approval.

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

The terms and conditions governing the sale of DGC hardware products and the licensing of DGC software consist solely of those set forth in the written contracts between DGC and its customers. No representation or other affirmation of fact contained in this document including but not limited to statements regarding capacity, response-time performance, suitability for use or performance of products described herein shall be deemed to be a warranty by DGC for any purpose, or give rise to any liability of DGC whatsoever.

In no event shall DGC be liable for any incidental, indirect, special or consequential damages whatsoever (including but not limited to lost profits) arising out of or related to this document or the information contained in it, even if DGC has been advised, knew or should have known of the possibility of such damages.

CEO, DASHER, DATAPREP, ECLIPSE, ENTERPRISE, INFOS, microNOVA, NOVA, PROXI, SUPERNOVA, ECLIPSE MV/4000, ECLIPSE MV/6000, ECLIPSE MV/8000, TRENDVIEW, MANAP, SWAT, GENAP, and **PRESENT** are U.S. registered trademarks of Data General Corporation, and **AZ-TEXT, DG/L, DG/XAP, GW/4000, ECLIPSE MV/10000, GDC/1000, REV-UP, UNX/VS, XODIAC, DEFINE, SLATE, DESKTOP GENERATION, microECLIPSE, BusiPEN, BusiGEN,** and **BusiTEXT** are U.S. trademarks of Data General Corporation.

Ordering No. 014-000767

Rev. 00, October 1983

Notice

Data General Corporation (DGC) has prepared this document for use by DGC personnel, customers, and prospective customers. The information contained herein shall not be reproduced in whole or in part without DGC's prior written approval.

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

The terms and conditions governing the sale of DGC hardware products and the licensing of DGC software consist solely of those set forth in the written contracts between DGC and its customers. no representation or other affirmation of fact contained in this document including but not limited to statements regarding capacity, response-time performance, suitability for use or performance of products described herein shall be deemed to be a warranty by DGC for any purpose, or give rise to any liability of DGC whatsoever.

In no event shall DGC be liable for any incidental, indirect, special or consequential damages whatsoever (including but not limited to lost profits) arising out of or related to this document or the information contained in it, even if DGC has been advised, knew or should have known of the possibility of such damages.

CEO, DASHER, DATAPREP, ECLIPSE, ENTERPRISE, INFOS, microNOVA, NOVA, PROXI, SUPERNOVA, ECLIPSE MV/4000, ECLIPSE MV/6000, ECLIPSE MV/8000, TRENDVIEW, MANAP, SWAT, GENAP, and **PRESENT** are U.S. registered trademarks of Data General Corporation, and **AZ-TEXT, DG/L, DG/XAP, ECLIPSE MV/10000, GW/4000, GDC/1000, REV-UP, UNX/VS, XODIAC, DEFINE, SLATE, DESKTOP GENERATION, microECLIPSE, BusiPEN, BusiGEN,** and **BusiTEXT** are U.S. trademarks of Data General Corporation.

Ordering No. 014-000767

Rev. 01, October 1983

ADDENDUM to Model 20 and 30 Computer Systems

042-000074-00

This addendum updates manual 014-000767-00 to:

014-000767-01

See the updating instructions on the following pages. The pages have been formatted so that you can insert the information at the beginning or end of a chapter.

Replace the old "Notice" page with the new one provided.

1 Programming the CPU

Page 1-9

Under page heading, "Address Translation and Protection", Replace the first paragraph with the following paragraph:

A program can load an address translation map consisting of 32 11-bit words for a maximum of four users, plus one for a data channel. Each user's logical address space consists of 32 1024-word (2 Kbyte) pages. Of the 11 bits specified by the program, ten of them specify the physical page to which a logical page is mapped; and one bit specifies whether that page is write-protected. A twelfth bit, derived and appended by hardware, specifies whether the page is validity-protected. Hardware derives this bit from the 11 bits supplied by the program; a page becomes validity-protected when these bits are all set to ones.

Page 1-15

Table 1-8 Status Instructions
Action entry for DIS CPU Read Processor Status should be replaced with the following:

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, and single-step NMI.

Page 1-20

Table 1-17 I/O Instructions
Action entry for Mnem NIO*[f]* should read:

Used to issue an I/O flag command; Start, Clear, or Pulse.

Page 1-20

Table 1-18 I/O command flags
The table should be replaced with the following:

Mnem	Flag Value	Action
[f] omitted	00	No effect
[f] =S	01	Issues a Start command: effect is device dependent
[f] =C	10	Issues a Clear command: effect is device dependent
[f] =P	11	Issues a Pulse command: effect is device dependent

Page 1-22

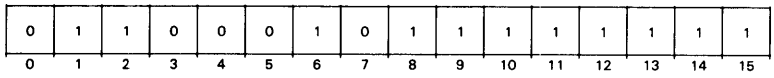
Table 1-23 MAP instructions
Delete the entire table entry for mnemonic LMPA.

Page 1-28

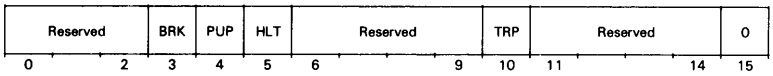
CPU Acknowledge instruction description should be replaced with the following:

CPU Acknowledge

DOAP *ac* ,CPU



Clears selective nonmaskable interrupt bits 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)
—	0-2	—	Reserved. Set to 0 or 1.
BRK	3	Break Key	Clears the Break key NMI bit
PUP	4	Power Up	Clears the Power Up NMI bit.
HLT	5	Halt	Clears the Halt NMI bit.
—	6-9	—	Reserved. Set to 0 or 1.
TRP	10	Trap	Clears the virtual console single-instruction mode NMI bit
—	11-14	—	Reserved. Set to 0 or 1.
—	15	—	Reserved. Must be set to 0.

Page 1-31

Under page heading, "Unmapped Mode", third sentence,
first 32 kilobytes of physical memory
should read:

first 64 kilobytes of physical memory

Page 1-31

Under page heading, "MAP Address Translation",
Replace text with the following:

Figure 1-7 illustrates the address translation performed by the MAP unit. Each 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, 11-bit words for each of up to four user and one map for a data channel. A 12th bit, derived by the MAP unit hardware from the 11 bits supplied by the program, is appended to the 11 bits and loaded into user translation map words. Each 12-bit word in a user's map includes 10 bits that specify the physical page to which a logical page is mapped; and two bits that indicate a MAP protection code. One map code bit marks the page as write-protected or write enable; the other code bit, derived and appended by hardware, specifies that the page is validity-protected or unprotected. A page becomes validity-protected when the 11 bits supplied by the program are all set to ones.

Page 1-37

Read MAP Status instruction — Accumulator format table
Entries for bit(s) 0,13 and 1 should be replaced with the following:

Bits	Name	Contents or Function
0,13	NME	Next MAP enabled. Depending on the bit settings, the last DOA MAP instruction enabled: Bit 1 Bit 13 User Enabled 0 0 A 0 1 B 1 0 C 1 1 D
1	MPN	MAP state — 1 indicates mapping

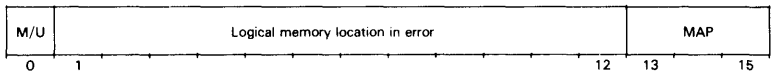
Page 1-40

Map Single Cycle instruction
Second paragraph of this instruction should read:

From user (mapped) mode: if the LEF mode and I/O protection are disabled, the NIOP instruction turns off the MAP after the next memory reference.

Page 1-43

Figure 1-8 Programming summary: accumulator formats
Accumulator format for *Read Parity Fault Address* instruction (DIA) should be replaced with the following format:

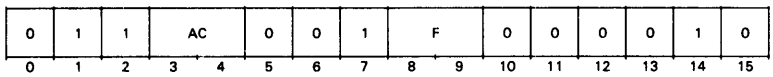


Page 1-44

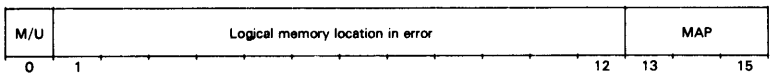
Read Parity Fault Address instruction description should be replaced with the following:

Read Parity Fault Address

DIA*[f]* *ac* ,2



Places the 12 most significant bits of the logical address of the last memory location in error in bits 1 to 12 of the specified accumulator. Bits 13 to 15 indicate which map, if any, was enabled at the time of the memory error. Bit 0 is set to one if the error occurred in mapped mode or to zero if the error occurred in unmapped mode. The format of the specified accumulator is



Bits	Name	Contents or Function
0	M/U	When 1, mapping was enabled when the parity fault occurred. When 0, mapping was disabled. When 0, bits 13-15 are set to zeros.
1-12	—	Most significant 12 bits of the logical memory address when the parity error occurred.
13-15	MAP	Specifies which map was enabled, if the fault occurred while in mapped mode. Set to zeros if the fault occurred while in unmapped mode.

Page 2-3

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

Entry *f* = C Receiver section should be followed by:

Sets the receiver Busy flag to 1 and the Done flag to 0.

Page 2-6

Read Character instruction
Device code mnemonic for this instruction should read TTI.

Page 2-8

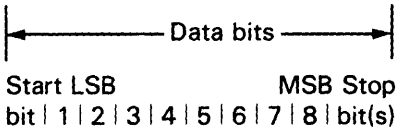
Figure 2-2 Writing characters

Change Done = 0 ? decision block to Done = 1 ?

Page 2-9

Figure 2-3 Asynchronous transmission of 8-bit character

Correct this figure with the following:

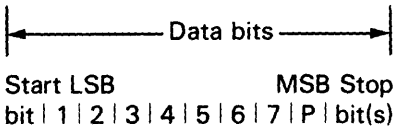


The earliest point where next character can start is after the stop bit(s).

Page 2-10

Figure 2-4 Asynchronous reception of 7-bit character

Correct this figure with the following:



P = Parity bit

The earliest point where next character can start is after the Stop bit(s).

Figure caption should read:
Asynchronous reception of 7-bit character

Page 2-15

Under page heading, "Clock Counter",
Insert the following sentence after "...enabled.", line 4:

After the Busy flag sets to 0 and the Done flag sets to 1, the counter is again loaded with the count in the Initial Count register and increments as before.

Page 2-16

Under page heading, "Done Flag",
Add the following sentence to the paragraph:

After the Done flag sets to 1, the Clock Counter is again loaded with the count in the Initial Count register and the count continues.

Page 2-21

Table 2-10 Programming summary: diskette formats
Byte packing format for IBM PC diskette should read:

Low byte/High byte

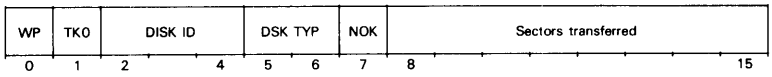
Page 2-21

Table 2-10 Programming summary: diskette formats
Byte packing format for DG MPT/100 diskette should read:

High byte/Low byte

Page 2-22

Figure 2-7 Programming summary: accumulator formats
Accumulator format for *Read Diskette Status* instruction (DIA), when a Get Number of Sectors Transferred was previously issued, should be replaced with the following format:



Page 2-27

Table 2-12 Diskette commands
Command column:

Get Number of Sectors should read:

Get Number of Sectors Transferred

Description column for the Get Number of Sectors Transferred command:

Delete the word "Transferred" from the end of the second line.

Description column for the Recalibrate command:

Change reference from Table 1 to Table 2-13.

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 describes the functional and physical organization of the Desktop Generation Model 20 and Model 30 computer systems. 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 4 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 CPU instruction set and describes in detail those instructions that are unique to the Model 20 and Model 30 CPU.
- Chapter 2 defines the instruction sets and tells you how to program the asynchronous communications interface, 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 Model 20 and Model 30 computer system.

- Chapter 4 describes the one-card Model 20 and 30 SPU, including the microECLIPSE processor, the MAP unit, the asynchronous communications interface, real-time clock, programmable interval timer and the microI/O bus interface. 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 a Model 20 and 30 dynamic random-access memory cards.
- Chapter 6 discusses the operation of the Model 30 hardware floating point card.
- Chapter 7 discusses the operation of the Model 20 and 30 diskette interface card.
- Chapter 8 provides a description of the Desktop Generation power supply.

The chapter in part 3 illustrates the mechanical components of the Model 20 and Model 30 computer system.

- Chapter 9 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 provides information to calculate the execution times for any of the commercial instructions.
- Appendix D discusses instruction differences between the Desktop Generation Models 20 and 30 and other ECLIPSE computers.
- The index alphabetically lists the concepts and terms in this book and references the pages on which they appear.
- Last, a publications comment form invites you to help Data General improve future publications by evaluating this manual.

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 Computer Systems
Technical Reference

In addition to the functional and physical organization of Model 10 and 10/SP 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 10 and 10/SP. DGC ordering no. 014-000766.

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.

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.

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.

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.

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 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.

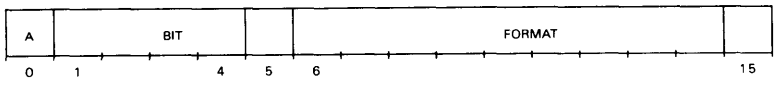
In dialogs between system and user, we use this typeface to show your input:

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

System Overview

Data General Desktop Generation Model 20 and Model 30 systems are micro-ECLIPSE-based 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 20 and Model 30 computer systems. It discusses their configurations and functional organization and briefly describes their major components, including: the Model 20 and Model 30 system processors and their memories, the Model 30 hardware floating point unit, the diskette and Winchester disk subsystems, the power subsystem, and the optional cartridge tape subsystem, communications multiplexors, and input/output (I/O) interfaces. System technical specifications conclude the chapter.

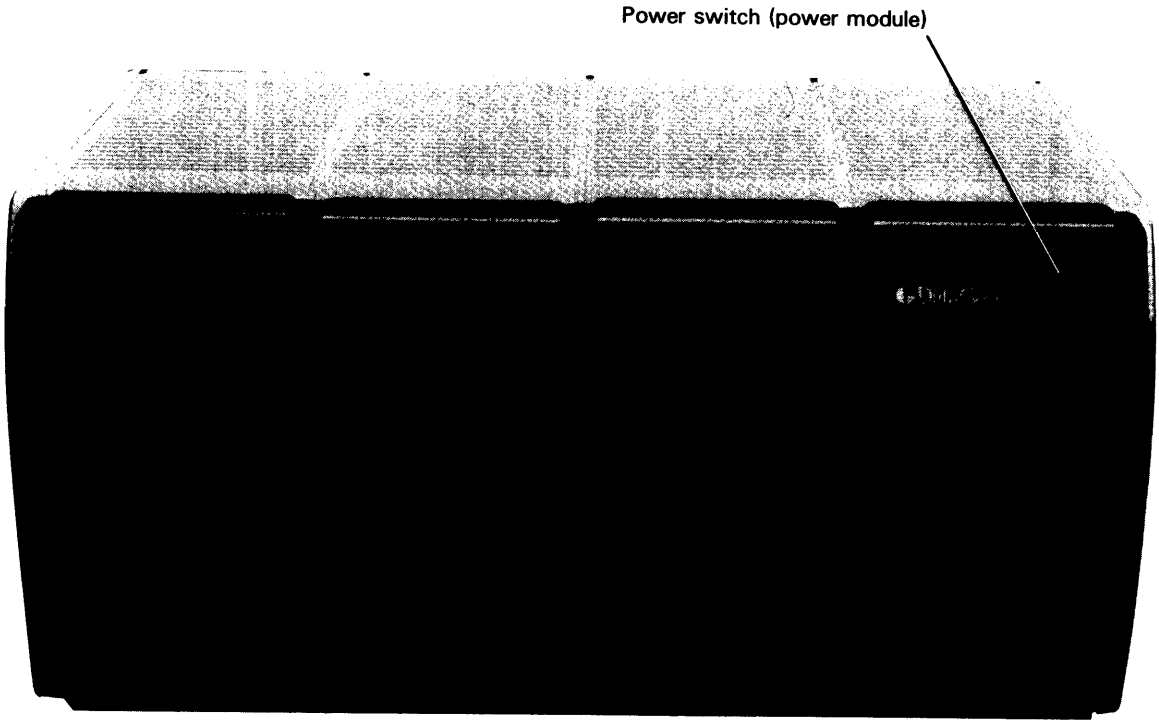


Figure 0-1 Model 20 and Model 30 computer systems

Configurations

Model 20 and Model 30 computers 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. Preconfigured systems consist of a system console and four modules: the three basic modules and a disk module. These modules house the following components:

1. Power module, containing:
   One 123-watt power supply (3-module unit); or
   Two 123-watt power supplies (4-module unit or greater)
   Cooling blower
   Optional line frequency clock generator card
2. CPU logic module, containing:
   Model 20 or Model 30 system processor unit
   256 Kbyte or 512 Kbyte semiconductor memory with byte parity
   Hardware floating point unit (Model 30 only)
   Three additional memory or I/O card slots, Model 20; or
   Two additional memory or I/O card slots, Model30
3. Diskette module, containing:
   Diskette controller
   One or, optionally, two 5.25-inch diskette drives
4. Disk module, containing:
   Disk controller
   One 5.25-inch Winchester disk drive

A disk module can be added to the basic, 3-module, computer unit and both minimum and preconfigured systems can be expanded by the addition of the following components:

- 256 Kbyte or 512 Kbyte memory cards up to a maximum memory capacity of 2 megabytes for Model 20 systems and 1.5 megabytes for Model 30 systems.
- Asynchronous/synchronous communications multiplexors, including an IEEE-488 bus interface.
- 5-slot logic expansion module that accommodates up to five I/O interface cards.
- Sensor I/O subsystems, including analog-to-digital, digital-to-analog, and digital I/O interfaces.

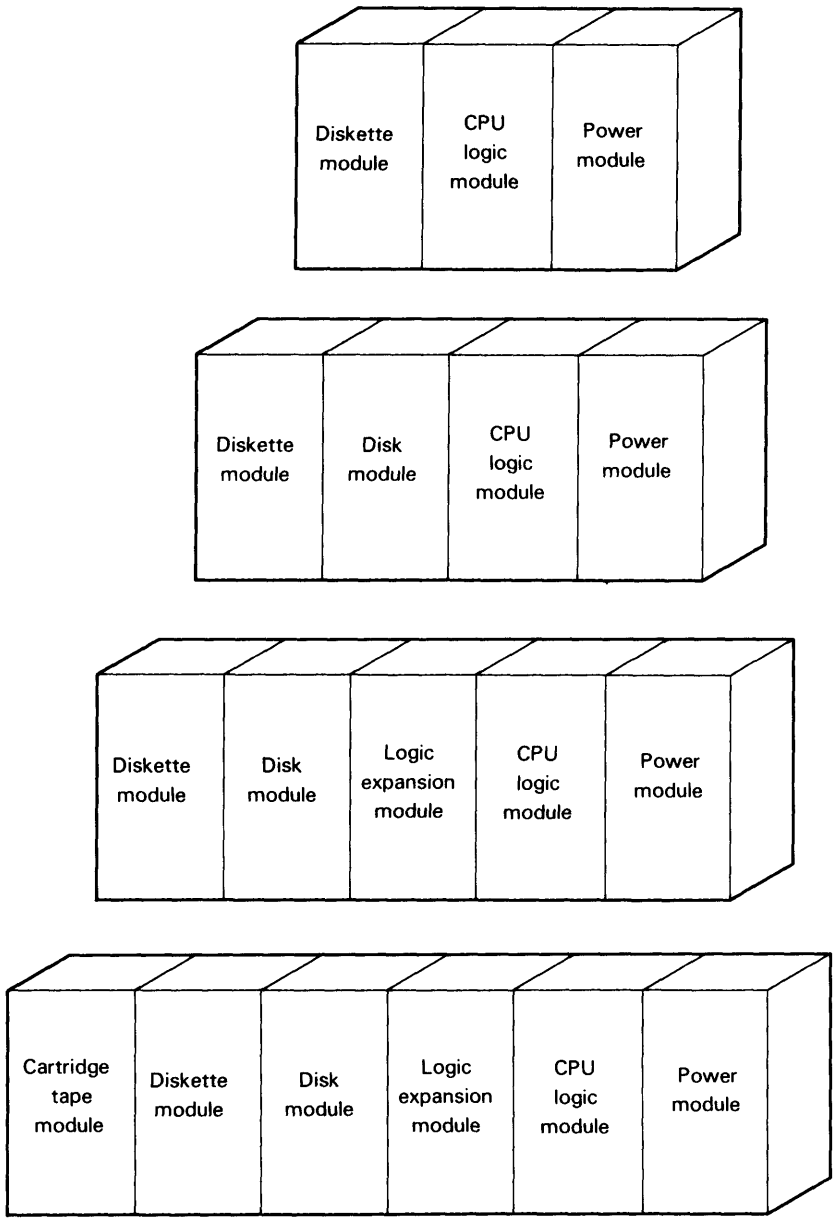
- Disk expansion unit, consisting of a disk module (less the controller) and a power module with one 123-watt power supply.
- Cartridge tape module, containing: a controller, a 5.25-inch cartridge tape drive, and a power supply.

System Overview

Page 4

Last component entry should read:

- Cartridge tape module, containing a controller, a 1/4 inch cartridge tape drive, a fan, and a power supply.



ID-00648

Figure 0-2 Model 20 and Model 30 configurations

Organization

As shown in Figure 0-3, the components of Model 20 and Model 30 systems 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 processor and its memories, and the Model 30 hardware 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 system processor and its I/O subsystems.

Components

The components that comprise Model 20 and Model 30 systems are described below.

System Processors

The system processor unit (SPU) of the Model 20 and Model 30 systems contains the microprogrammed, microECLIPSE central processing unit (CPU). The Model 20 CPU supports the character instruction set and firmware floating point. The Model 30 CPU, in conjunction with its hardware floating point card, supports the commercial instruction set and hardware floating point.

Both SPUs also contain:

- Memory allocation and protection unit (MAP)
- Asynchronous communications interface for the system console
- Virtual console
- Real-time clock
- Programmable interval timer
- Parity checking logic for memory data
- Power status monitor
- Power-up diagnostics

The MAP performs logical-to-physical address translation for up to 2 megabytes of memory. It also provides the following protection mechanisms: validity, write, I/O and indirection.

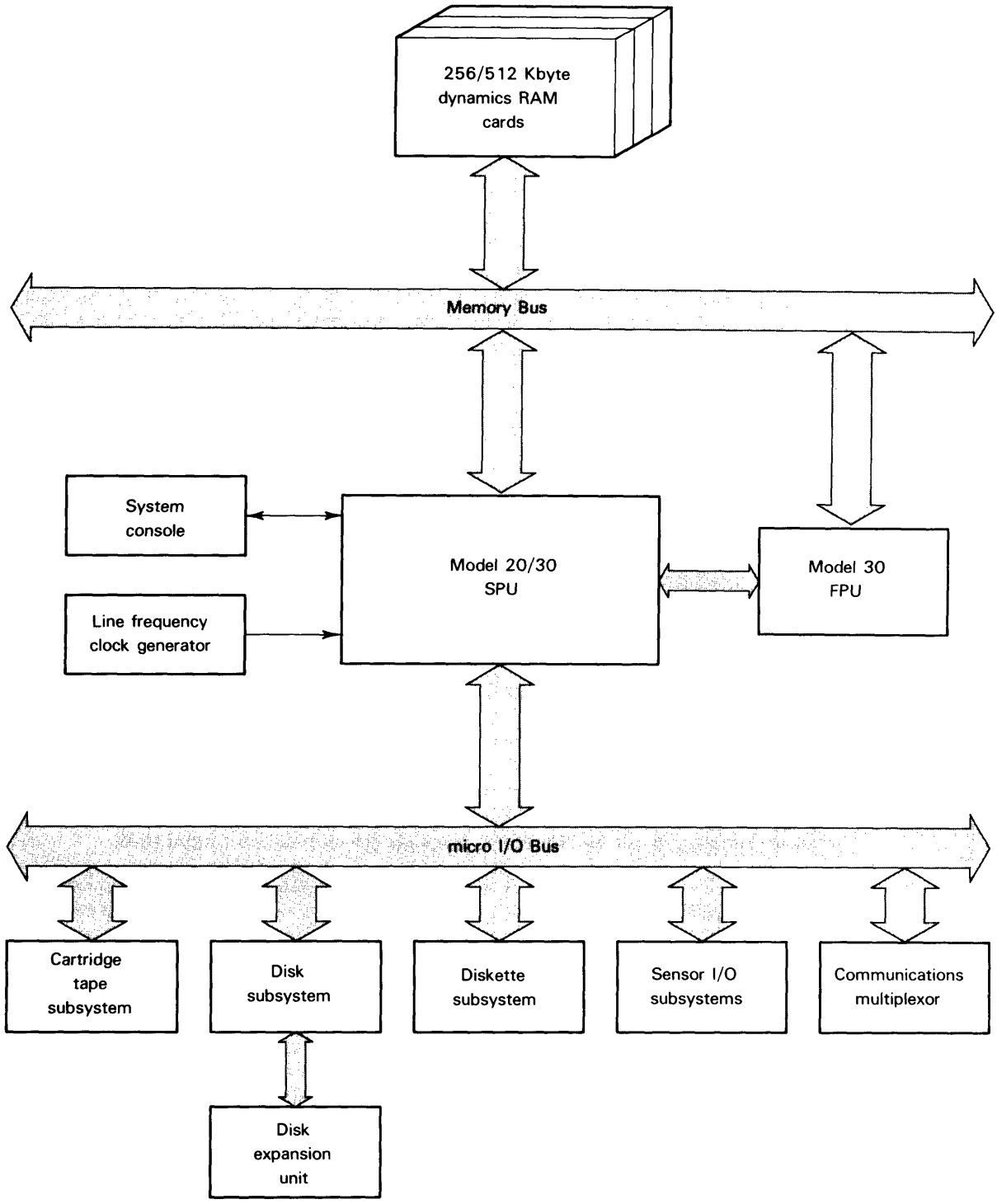


Figure 0-3 Model 20 and Model 30 system organization

The asynchronous communications port provides an EIA-RS232C or 20 MA current loop line interface, jumper-selectable, for a selection of DASHER™ display terminals that are offered for use as the Model 20 and Model 30 system console.

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, or 1,000 Hz. And, when the optional line frequency clock generator is present, the real-time clock can generate it's interrupts at the ac line frequency rate. The PIT can be programmed to generate I/O interrupts at fixed intervals ranging from 1 microsecond to 65.536 seconds, in clock rate increments of 1 microsecond to 1 milliseconds. The clock rate of the PIT is switch-selectable.

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 enabled by the program, a parity error generates a program interrupt.

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 less than one second test routine — verifies the basic integrity of the system each time power is applied. It checks the first 64 Kbytes of memory, the virtual console, the CPU, and the system console interface.

Memory

Model 20 and 30 CPUs support up to four or three 256 Kbyte or 512 Kbyte memory boards, respectively. Thus, Model 20 systems support up to 2 megabytes of memory while Model 30 systems support up to 1.5 megabytes.

Both 256 Kbyte and 512 Kbyte memory cards contain 64K by 1-bit elements of dynamic MOS random-access memory (RAM). Each card includes byte parity bits for data and supports its memory with integral refresh logic.

Diskette Subsystem

The diskette subsystem consists of a diskette controller and 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.

The diskette controller can be programmed to read and write Data General formatted diskettes (9 sectors per track) or CP/M-86®[1] 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).

Each sector stores 512 bytes of data. The diskette controller provides a one-word (2-bytes) buffer for data transfers and uses the microI/O bus and the data

[1] *CP/M-86 is a registered trademark of Digital Research Corporation.*

channel facility of the CPU to transfer 16-bit data words between memory and the subsystem.

Disk Subsystem

The 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 a disk 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 the data channel facility of the CPU to transfer 16-bit data words between memory and the subsystem.

Cartridge Tape Subsystem

The optional cartridge tape subsystem consists of a controller, a 1/4 inch magnetic tape cartridge drive, 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

Model 20 and 30 systems support a selection of asynchronous/synchronous communications multiplexors and sensor I/O subsystems. Each communicates with the CPU via the microI/O bus and each card occupies an I/O slot in either the CPU logic module or the 5-slot expansion module.

Power Subsystem

The power subsystem consists of two, 123 watt, power supply assemblies and a cooling blower that reside in the power module. This module also houses a line frequency clock generator card.

One power supply provides dc power to the CPU logic module and the diskette subsystem while the second supply supports the main disk module and the 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 Model 20 and Model 30 computer systems.

Table 0-1 General specifications, models

Unit	Description
Model 20 System	Basic 3-module system includes: Model 20 system processing unit, one 256 Kbyte or 512 Kbyte memory card, diskette subsystem with controller and one 5.25-inch diskette drive, and a power module containing one 123-watt power supply and a cooling blower. Preconfigured system also includes a disk module containing a 15-Mbyte Winchester disk subsystem and second 123-watt power supply.
Model 30 System	Basic 3-module system includes: Model 30 system processing unit, hardware floating point card, one 512 Kbyte memory card, diskette subsystem with controller and one 5.25-inch diskette drive, and a power module with one 123-watt power supply and a cooling blower. Preconfigured system also includes a disk module containing a 15-Mbyte Winchester disk subsystem and second 123-watt power supply.
Model 20/30 System console	One DASHER display terminal.

Table 0-2 General specifications, basic components

Unit	Description	
System Processing Units (SPUs)		
CPUs	microECLIPSE central processor	
	Instruction set	
	Model 20	Character instruction set with firmware floating point
	Model 30	Commercial instruction set with hardware floating point (Model 30 hardware floating point must be present in slot 2 of CPU logic module.)
	Instruction length	16 and 32 bits
	Accumulator length	Fixed point, 16 bits; floating point, 64 bits
	Number of accumulators	4 fixed point (2 usable as index registers) 4 floating point
	Priority interrupt levels	16 programmable
	Maximum data channel rates	
	input	267 Kbytes/second
	output	337 Kbytes/second
	Memory allocation and protection unit	Supports 2 Mbytes of physical memory with: 4 user maps 1 data channel map write protection validity protection I/O protection indirect protection
System Console Interface		Full-duplex, asynchronous interface with jumper-selectable EIA RS-232C or 20 mA current loop line interface. Baud rates, ranging from 50 to 19,200 bits per second, are switch selectable.

Table 0-2 General specifications, basic components (Continued)

Unit	Description
Real-time Clock	Initiates I/O interrupts at one of the three frequences: 10 Hz, 100 Hz or 1,000 Hz. Clock frequency is program-selectable. Can also initiate interrupts at a fourth frequency: a.c. line frequency, when the optional line frequency clock generator card is present.
Programmable Interval Timer (PIT)	Initiates I/O interrupts at fixed intervals ranging from 1 microsecond to 65.536 seconds, in 1 microsecond to 1 millisecond increments. Clock rate is switch-selectable.
Memory	System location: CPU logic module. Model 20, slots 2-5; model 30, slots 3-5 Type semiconductor MOS Increment size/board 256 Kbytes and 512 Kbytes with byte parity Memory cycle time 500 nanoseconds Memory configurations Model 20 minimum 256 Kbytes maximum 2 Mbytes Model 30 minimum 512 Kbytes maximum 1.5 Mbytes System location: diskette module
Diskette Subsystem	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) *512 bytes per sector System location: Power module
Power Subsystem	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

Table 0-3 Mechanical specifications

Unit	Width	Depth	Height	Weight
Systems				
Basic 3-module system with covers	15.6 in 39.6 cm	12.5 in 31.8 cm	10.5 in 26.7 cm	39.7 lbs 18.0 kg
Four-module system with covers and disk module	20.4 in 51.8 cm	12.5 in 31.8 cm	10.5 in 26.7 cm	54.2 lbs 24.6 kg
Four-module system with covers and logic expansion unit	20.4 in 51.8 cm	12.5 in 31.8 cm	10.5 in 26.7 cm	50.0 lbs 22.7 kg
Five-module system with covers, disk module, logic expansion unit	25.2 in 64.0 cm	12.5 in 31.8 cm	10.5 in 26.7 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	10.5 in 26.7 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	12.5 in 31.8 cm	10.5 in 26.7 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	12.5 in 31.8 cm	10.5 in 26.7 cm	18.7 lbs 8.5 kg
CPU logic module with model 20 SPU and 1 memory card	4.8 in 12.2 cm	12.5 in 31.8 cm	10.5 in 26.7 cm	8.0 lbs 3.6 kg
CPU logic module with model 30 SPU, FPU and 1 memory card	4.8 in 12.2 cm	12.5 in 31.8 cm	10.5 in 26.7 cm	8.0 lbs 3.6 kg
Diskette module with controller and 1 drive	4.8 in 12.2 cm	12.5 in 31.8 cm	10.5 in 26.7 cm	11.5 lbs 5.2 kg
Diskette module with controller and 2 drives	4.8 in 12.2 cm	12.5 in 31.8 cm	10.5 in 26.7 cm	14.8 lbs 6.7 kg
Disk module with controller and 1 drive	4.8 in 12.2 cm	12.5 in 31.8 cm	10.5 in 26.7 cm	14.5 lbs 6.6 kg
Logic expansion unit (empty)**	4.8 in 12.2 cm	12.5 in 31.8 cm	10.5 in 26.7 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	10.5 in 26.7 cm	15.5 lbs 7.0 kg
Disk expansion unit with 1 drive and 1 power supply	10.8 in 27.4 cm	12.5 in 31.8 cm	10.5 in 26.7 cm	28.5 lbs 12.9 kg

* *Each side cover adds 0.6 inch (1.5 cm) to width.*

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

Table 0-4 Electrical specifications

Power requirements	
Voltage:	100 volts a.c., ± 10% 120 volts a.c., + 10%, − 15% 240 volts a.c., + 10%, − 15%
Line frequency:	47-63 Hz
Line current (maximum)	
Unit with 1 power supply:	3 amperes @ 120 volts a.c.
Unit with 2 power supplies:	6 amperes @ 120 volts a.c.
Power consumption (maximum)	
Unit with 1 power supply (includes blower unit):	180 watts
Unit with 2 power supplies (includes blower unit):	360 watts

Table 0-5 Environmental specifications

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

Part

ONE

Programming

Programming the CPU

1

The heart of the Model 20 and 30 computer systems is the microECLIPSE central processing unit (CPU) which accesses memory, manages data, and controls program flow. The CPU performs fixed-point and floating-point computations in the Model 20. It performs fixed-point computations and initiates and controls data transfers for hardware floating-point computations in the Model 30 systems.

This chapter summarizes the programming capabilities of the both systems. It includes brief descriptions of important programming features; presents all Model 20 and 30 CPU instructions in table form; and describes in detail those instructions and facilities that are unique to these CPUs. It also includes lists of program-accessible registers, reserved memory locations, and instruction execution times. The chapter concludes with descriptions of system initialization at power up and powerfail/autorestart.

Summary of CPU Capabilities

This section summarizes the programming capabilities of the Models 20 and 30. It includes brief descriptions of important programming features such as addressing modes and data formats, and explains SPU operations. This section is not intended as a programming reference. Rather, it serves as an introduction to the capabilities of the SPUs. For complete information on programming the Model 20 and 30 SPUs refer to the *16-bit Real-Time* ECLIPSE *Assembly Language Programming.*

Addressing

The size of the CPU's logical address space is 64 Kbytes. The physical address space is a maximum of 2 Mbytes in the Model 20 systems and 1.5 Mbytes in the Model 30. Refer to the section entitled "MAP Operations" for a summary of logical-to-physical address translation.

The Model 20 and 30 computers have two classes of instructions: short and extended. Short class instructions contain an 8-bit address displacement; extended class instructions contain a 15-bit address displacement. Both classes 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 CPUs permit 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.)

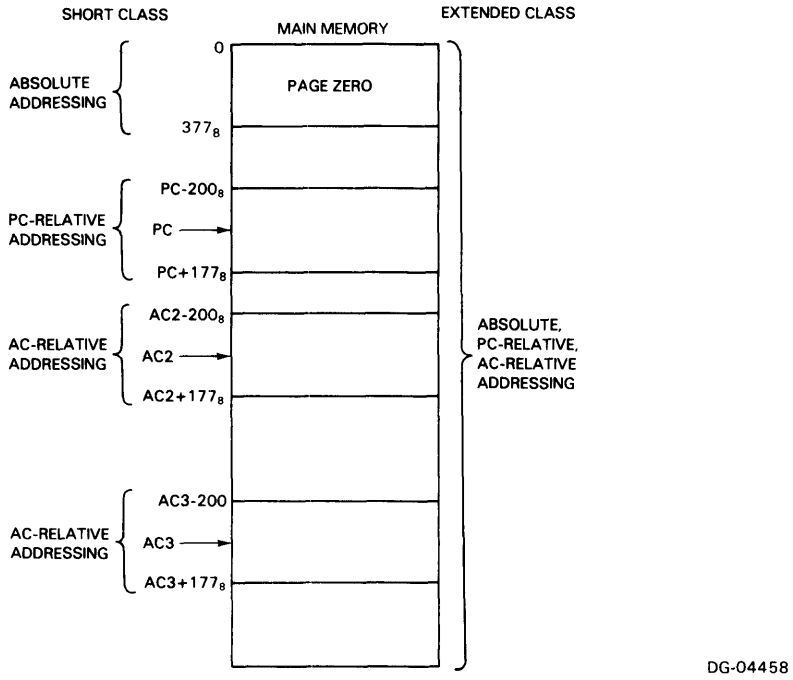


Figure 1-1 Addressing modes

Addressing Modes

Direct or indirect word addressing 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 1-1.
- *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).

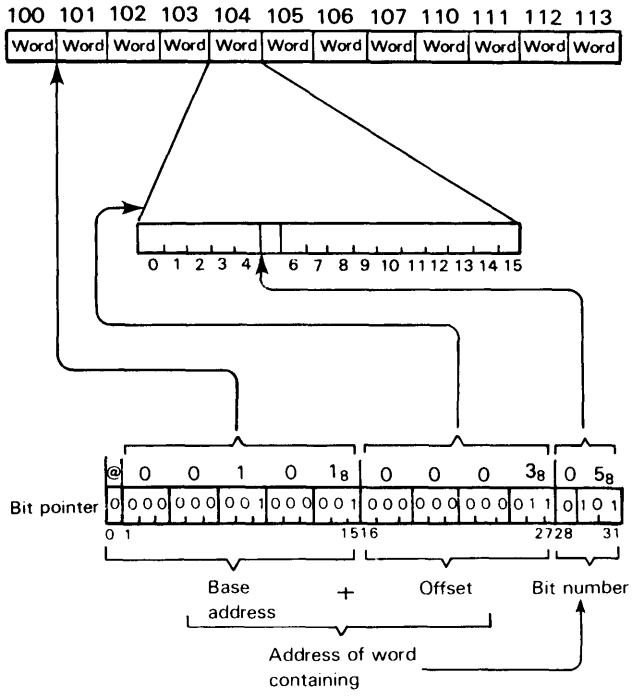
Figure 1-1 illustrates the accessible memory ranges for the two instruction classes and three addressing modes (direct addressing). Note that absolute addressing mode can be used to access lower page zero, locations $0\text{-}377_8$, regardless of the current contents of the program counter.

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; extended class instructions use their displacement field to hold the pointer.

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 1-2 illustrates the bit addressing process.



DG-08290

Figure 1-2 Bit pointer

Data Formats

This subsection summarizes integer formats in Figure 1-3, commerical formats in Figure 1-4, and floating point formats in Figure 1-5. Floating-point numbers are normalized at the end of all floating-point mathematic operations.

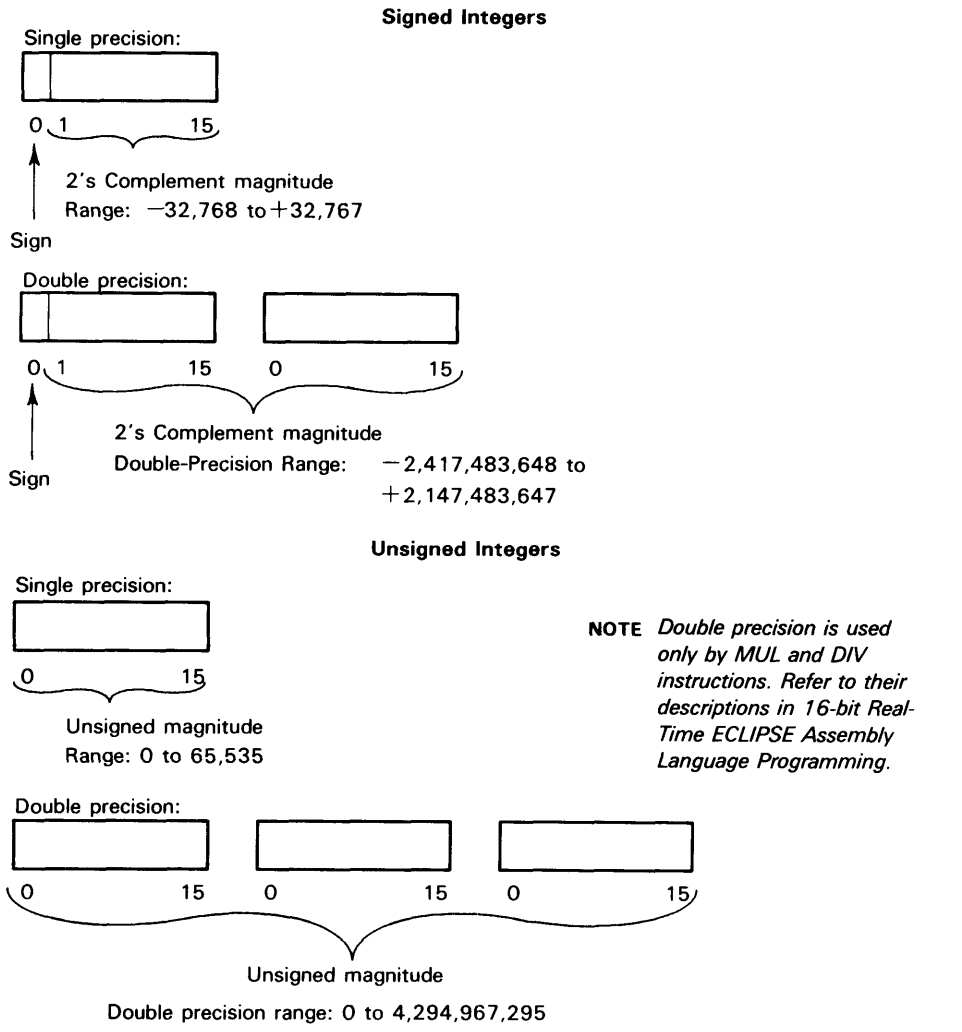


Figure 1-3 Integer formats

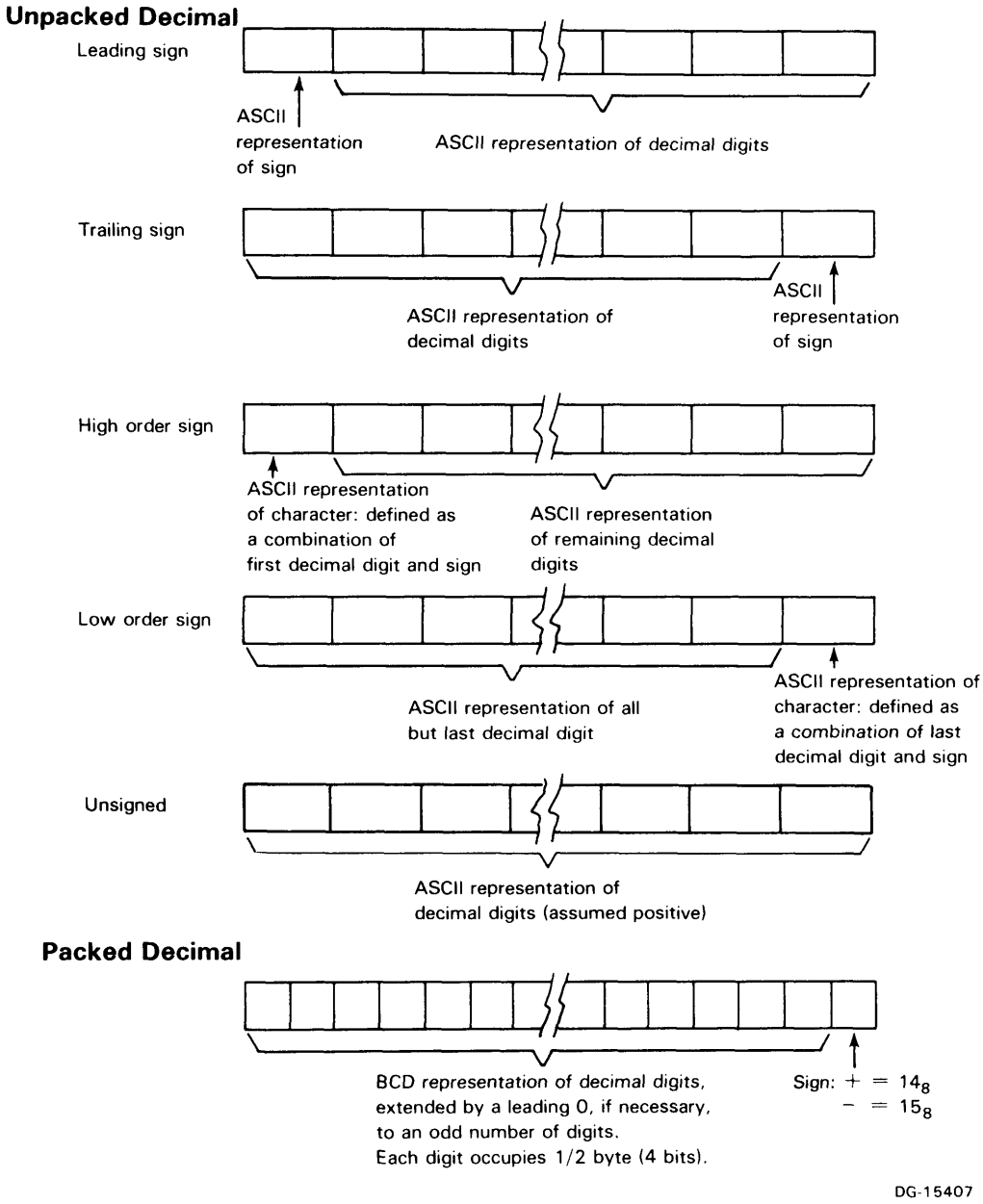


Figure 1-4 Commercial formats

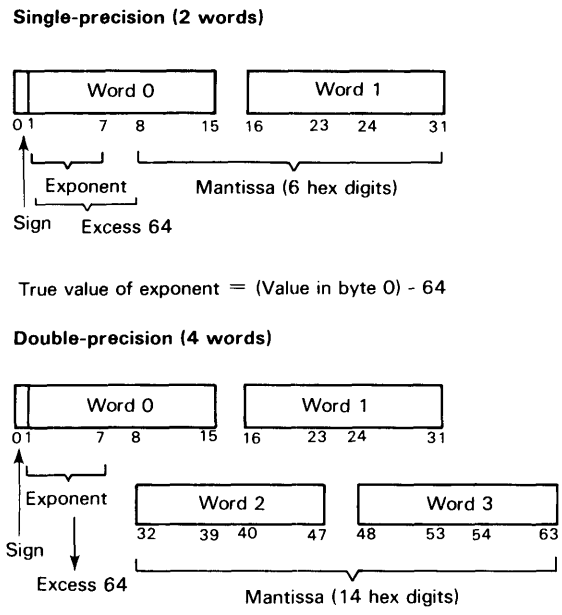


Figure 1-5 Floating-point formats

Arithmetic/Logic Class Operations

The arithmetic/logic class (ALC) instructions include ADC, ADD, AND, 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 1-6 illustrates the sequence of operations performed by a general ALC instruction.

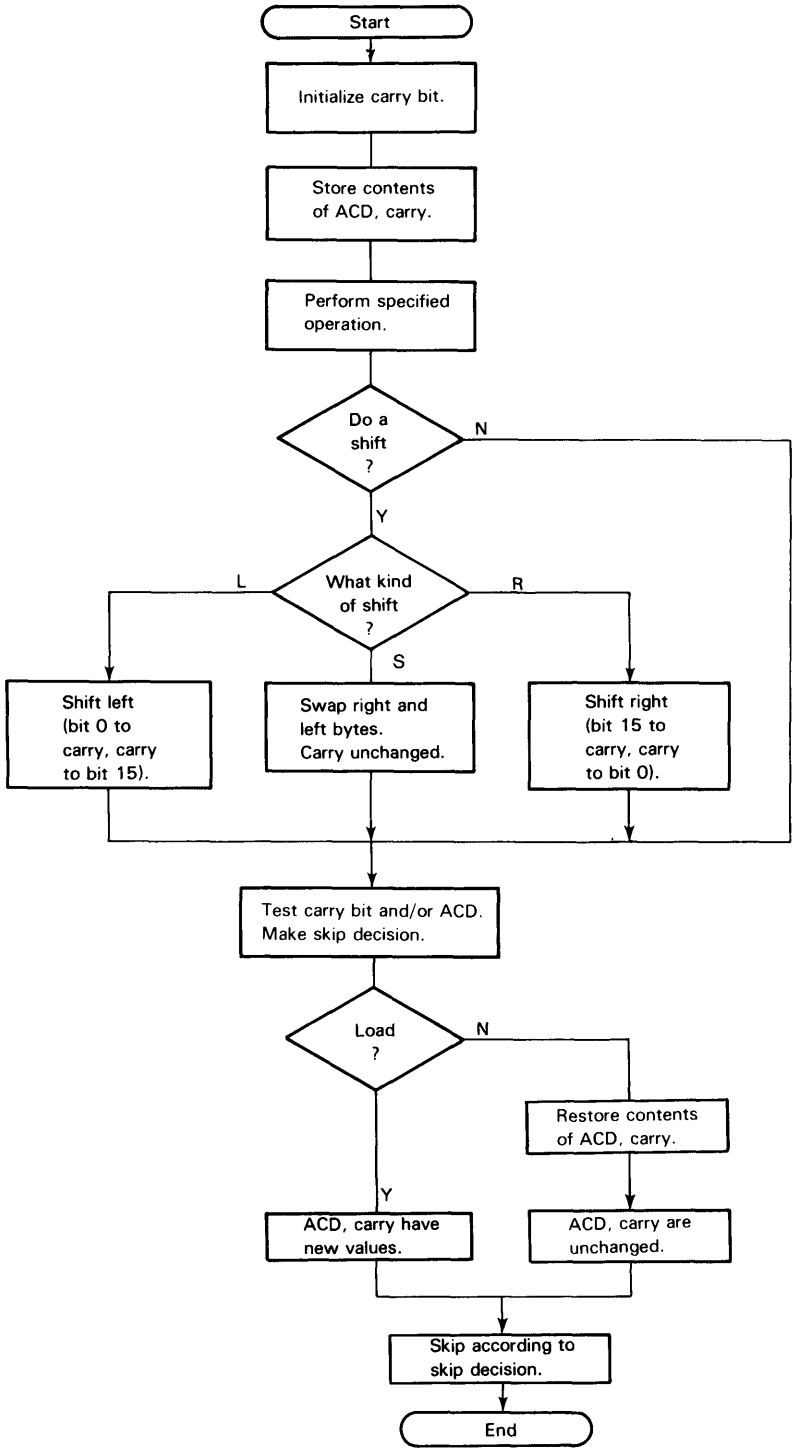


Figure 1-6 ALC instruction operation sequence

Stack Operations

The processor contained in the Models 20 and 30 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 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 floating point accumulators (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 the other 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 1-5.

String Operations

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

Commercial Operations

The Model 30 commercial instructions allow you to load decimal numbers (in BCD format) into a floating point accumulator (with change to floating point format), store decimal numbers from a floating point accumulator into memory, load the sign of a number, and convert decimal integers to byte strings, and then process the strings.

MAP Operations

This subsection explains the functions of address translation and protection performed by the Model 20 or Model 30 MAP unit, and describes the page 31 register. The address translation procedure is 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 a maximum of four users and a data channel. (The software actually loads 11 bits; the twelfth bit is hardware-derived from the logical AND of the physical address.) Each 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 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.

In addition, 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. Also, 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 later.

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 is used.

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" in 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. After the address has been retrieved, control transfers to the procedure.

Emulator Trap

The CPU in the Model 20 and Model 30 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 — providing 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

The parity checking facility, when enabled, detects a parity error in any byte of memory. Each memory word consists of 18 bits. These include 16 data bits followed by 2 parity bits — one for each byte. The parity logic checks the parity bits read from memory. If an error occurs, the parity logic requests an interrupt. The CPU obtains the address of the error with a Read Parity Fault Address (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 of the fault.

I/O Operations

The CPU addresses an I/O controller by means of the device code occupying bits 10-15 of an I/O instruction. The basic I/O instruction set is used to control I/O devices, to set up data channel operations, and to pass data to and from these devices. Programming details for a specific device or interface are available in the manual for that device.

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 its 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.

  Testing the device's Busy/Done flags with an *I/O Skip* instruction

  Placing the interrupter's device code in an accumulator with an INTA instruction

Identifying the interrupting device, saving return information, and jumping through a table to an individual device's interrupt handling routine with a VCT instruction

The CPU Instruction Set

This section presents Model 20 and Model 30 CPU instructions in table form. The instructions are grouped into the following categories.

Computing instructions

Program flow management

Stack and data management

Commerical instructions (Model 30 only)

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.
FNEG	Negate	Inverts the sign bit of the FPAC.

Table 1-7 Logic instructions (Continued)

Mnem	Instruction	Action
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 instructions that fall under this category 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.

Table 1-15 Stack instructions (Continued)

Mnem	Instruction	Action
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.
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.

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.

Commerical Instructions

The instructions in this category are available on Model 30 computer systems only. The commerical instructions are summarized in Table 1-16.

Table 1-16 Commercial instructions

Mnem	Instruction	Action
EDIT	Edit	Converts a decimal integer to a string of bytes controlled by an edit subprogram; or manipulates a string of bytes.
LDI	Load integer	Converts a decimal integer to normalized floating-point form and places it in a specified floating-point accumulator.
LDIX	Extended load integer	Distributes a decimal integer into four floating-point accumulators.
LSN	Load-sign	Evaluates a number in memory and returns a code indicating the sign of the number.
STI	Store integer	Converts the contents of a floating-point accumulator to a specified format and stores it in memory.
STIX	Extended store integer	Converts the contents of four floating-point accumulators to integer form and uses the eight low-order digits of each to form a 32-digit integer.

Device Management

The instructions for device management include basic I/O (Table 1-17), I/O command flag (Table 1-18), I/O interrupt (Table 1-19), I/O skip flag (Table 1-20), CPU device (Table 1-21), and CPU skip flag (Table 1-22) instructions.

Table 1-17 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 1-18 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 1-19 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.
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.

Table 1-19 I/O interrupt instructions (Continued)

Mnem	Instruction	Action
XCT	Execute	Executes contents of an accumulator as an instruction.

Table 1-20 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 1-21 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 Chapter 3 for a description of this register.*

Table 1-22 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 in this category are summarized in Table 1-23.

Table 1-23 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.

Reserved Memory Locations

Table 1-24 lists the program-accessible locations in page 0 of memory These locations have been reserved for the storage of data that have special meanings for the CPU.

Table 1-24 Reserved memory locations

Memory Address (octal)	Contents or Use
00000	Return address for I/O interrupts. Also, first instruction of auto-restart routine.
00001	Address of I/O interrupt handling routine. Indirectable.
00002	Address of system call instruction handler. Indirectable.
00003	Address of MAP fault handling routine. Indirectable.
00004	Address of the top of the vector stack. Nonindirectable.
00005	Current interrupt priority mask.
00006	Address of the last normally usable location in the vector stack.

Table 1-24 Reserved memory locations (Continued)

Memory Address (octal)	Contents or Use
00007	Address of the vector stack fault handler. Indirectable.
00011	Address of emulator trap handler. (If contents equal zero, an NOP is performed.)
00040	Address of the top of the stack. Nonindirectable.
00041	Address of the start of the current stack frame minus one. Nonindirectable.
00042	Address of stack upper limit.
00043	Address of stack fault routine. Indirectable.
00044	Address of the beginning of the XOP table. Nonindirectable.
00045	Address of floating-point fault handler. Indirectable.
00046	Reserved for future use.
00047	Reserved for future use.

Instruction Execution Times

Table 1-25 lists typical execution times for all instructions. The numbers in parentheses are execution times for the floating point instructions. Refer to Appendix C for more information on execution times for commerical instructions.

Table 1-25 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	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	2.00	4	7
DIS	2.00	4	7
DIV	12.00	24	
DIVS	20.50	41	
DIVX	19.50	39	
DLSH	6.50	13	5
DOA,B,C	2.00	4	7

Table 1-25 Instruction execution times (Continued)

Instruction	Execution time (microsec.)	Cpu Cycles	Notes
DSB	8.00	16	
DSPA	6.00	12	2
DSZ	2.00	4	2,1
EDIT	150.00	300	5
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
ESTA	1.00	2	2
ESTB	3.50	7	
FAB	11.00 (7.00)	22	8,9
FAD	87.00 (12.50)	174	8,9
FAMD	92.00 (18.00)	184	8,9,10
FAMS	67.00 (15.50)	134	8,9,10
FAS	65.00 (11.50)	130	8,9
FCLE	3.00 (1.50)	6	8
FCMP	29.00 (4.50)	58	8
FDD	900.00 (42.75)	1800	8,9,11
FDMD	900.00 (48.25)	1800	8,9,10,11
FDMS	190.00 (20.50)	380	8,9,10,11
FDS	190.00 (16.75)	380	8,9,11
FEXP	13.00 (9.25)	26	8,9
FFAS	46.00 (11.00)	92	8,9
FFMD	45.50 (12.50)	90	8,9,10
FHLV	30.00 (9.00)	60	8,9,11
FINT	20.50 (19.50)	41	8,9
FLAS	31.00 (10.00)	62	8
FLDD	16.50 (10.00)	32	8,10
FLDS	15.00 (7.50)	30	8,10
FLMD	31.00 (13.50)	62	8,10
FLST	11.50 (11.00)	23	8,9,10
FMD	266.00 (46.00)	532	8,9
FMMD	266.00 (51.50)	532	8,9,10
FMMS	80.50 (22.00)	161	8,9,10
FMOV	17.00 (4.50)	34	8
FMS	80.50 (18.50)	161	8,9
FNEG	12.50 (7.50)	25	8,9
FNOM	43.00 (11.50)	86	8,9
FNS	1.00 (4.00)	2	8
FPOP	32.00 (43.00)	64	8
FPSH	31.50 (38.00)	63	8
FRH	6.00 (4.00)	12	8
FSA	1.50 (5.00)	3	8
FSCAL	48.00 (15.50)	96	8,9
FSD	87.00 (12.75)	174	8,9
FSEQ	5.00 (4.50)	10	8
FSGE	4.50 (4.50)	9	8
FSGT	5.00 (4.50)	10	8
FSLE	5.00 (4.50)	10	8
FSLT	4.50 (4.50)	9	8
FSMD	92.00 (18.25)	184	8,9,10
FSMS	67.00 (15.50)	134	8,9,10
FSND	5.00 (4.50)	10	8
FSNE	5.00 (4.50)	10	8

Table 1-25 Instruction execution times (Continued)

Instruction	Execution time (microsec.)	Cpu Cycles	Notes
FSNER	5.00 (4.50)	10	8
FSNM	5.00 (4.50)	10	8
FSNO	5.00 (4.50)	10	8
FSNOD	5.00 (4.50)	10	8
FSNU	5.00 (4.50)	10	8
FSNUD	5.00 (4.50)	10	8
FSNUO	5.00 (4.50)	10	8
FSS	65.00 (12.00)	130	8,9
FSST	7.50 (7.00)	15	8,10
FSTD	12.50 (5.50)	25	8,10
FSTS	9.50 (4.00)	19	8,10
FTD	3.00 (1.50)	6	8
FTE	5.00 (3.75)	10	8,9
HALT	6.50	13	
HLV	1.50	3	
HXL	2.00	4	5
HXR	2.00	4	5
INC	0.50	1	1
INTA	2.00	4	7
IOR	1.50	3	
IORI	1.50	3	
IORST	3.00	6	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	
LDI	130/85	260/170	5, 17
LDIX	500	1000	5
LEF	1.00	2	2
LMP	4.50	9	2,12
LOB	4.00	8	5
LRB	5.00	10	5
LSH	8.50	17	5
LSN	155/67	310/134	5, 17
MOV	0.50	1	1
MSKO	2.00	4	7
MSP	3.00	6	13
MUL	9.50	19	
MULS	9.50	19	
NEG	0.50	1	1
NIO	2.00	4	7
POP	1.50	3	14
POPB	6.50	13	
POPJ	3.00	6	
PSH	3.00	6	13,14
PSHJ	4.50	9	13
PSHR	4.50	9	
RSTR	10.50	21	
RTN	6.00	12	
SAVE	8.00	16	13
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

Table 1-25 Instruction execution times (Continued)

Instruction	Execution time (microsec.)	Cpu Cycles	Notes
STB	1.50	3	
STI			
STIX			
SUB	0.50	1	1
SYC	10.50	21	2
SZB	4.00	8	1,4
SZBO	6.00	12	1,4
VCT	8.00 to 34.00	16 to 68	15
XCH	1.50	3	
XCT	2.00	4	16
XOP	20.50	41	
XOP1	21.50	43	
XOR	2.50	5	
XORI	2.50	5	

NOTES

1. If skip occurs, add 0.50 microseconds.
2. If indirect chain followed, add 0.50 microseconds + (number of indirects − 1*1.00.
3. For each item moved, add the amount shown.
4. If ACS < > ACD, add 1.00 microseconds + 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 devices PAR or MAP. For other internal (SIO) devices, time is 3.5 microseconds (7 T-periods) for external devices, times are 6 microseconds (12 T-periods) for output transfers and 7.5 microseconds (15 T-periods) for input transfers.
8. Floating-point execution times in parentheses are achieved with the optional floating-point board.
9. This instruction can take a floating point trap, which would add 19 microseconds to the execution time.
10. This instruction does an effective address calculation, which can add 15 microseconds to the execution time.
11. 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).
12. For each word moved, add 1.50 microseconds.
13. For stack overflow, add 9.00 microseconds + (Note 2).
14. For each accumulator pushed/popped, add 0.50 microseconds.
15. Vector execution times depend on the mode employed.
16. Add instruction execution time.
17. Time shown are for data types 0-5/data types 6,7.

System Management

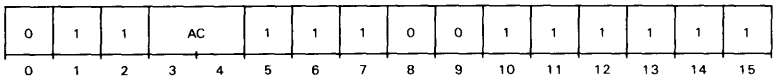
This section describes system management programmable elements that are unique to the Model 20 and Model 30 SPU. It describes program-accessible registers and defines instructions used to access those registers; defines unique instructions; and describes how the HALT instruction effects the CPUs.

CPU Status Register

The CPU status register reports status pertaining to the system processing unit. 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

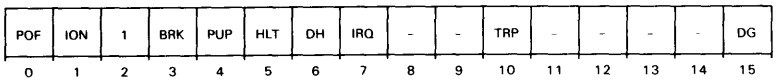
DIS*[f]* *ac*,CPU



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:



Mnem	Bit	Instruction	Action (If Set to 1)
POF	0	Power Fail	Power Fail flag set.
ION	1	Interrupt On	Interrupt On flag set.
1	2	--	Set to one.
BRK	3	Break Key Interrupt	Reset resulting from depression of console Break key. Used only by the virtual console.
PUP	4	Power Up Reset	Power came up since last **DOAP CPU** used only by the virtual console.
HLT	5	Halt	**HLT** instruction was executed. Used only by the virtual console.
DH	6	Halt Dispatch	If one, indicates that a **Halt** instruction will cause the processor to enter the virtual console. If zero, indicates that a **Halt** instruction will cause the processor to halt.
IRQ	7	Interrupt	An interrupt is being requested.
--	8-9	--	Undefined.
TRP	10	Trap	Indicates that the virtual console single-instruction mode is in use.
--	11-14	--	Undefined.
DG	15	Diagnostic Test	Indicates that the virtual console self-test is looping; for diagnostic purposes only.

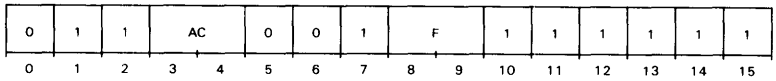
Program Load Register

The load register indicates which device last booted the system. After a program load has been performed, the device code of the device loaded from is placed in the virtual console switch register. A READS instruction can then be used to return the device code. The program load register definitions are listed below. For more information on program load, refer to Chapter 3, "Virtual Console."

Read Virtual Console Registers

READS ac

DIA*[f]* *ac*,CPU



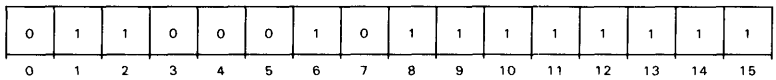
Places the contents of the virtual console register into an accumulator.

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	--	Reserved.
DLY	2	Delay	A one minute delay in the execution of the load.
--	3-9	--	Reserved.
--	10-15	Device Code	Specifies last device to perform a program load.

CPU Acknowledge

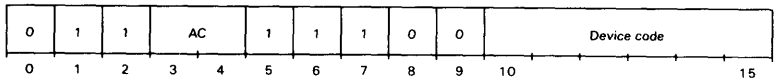
DOAP CPU



Clears nonmasked interrupts or the power-fail interrupt.

Data In Status

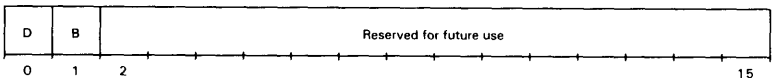
DIS*[f]* *ac*,device



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.

Halt Instruction

This instruction stops user program execution and returns to the virtual console program if the Halt Dispatch function is enabled by jumpering. If the Halt Dispatch function is not enabled, the processor honors data channel requests, but not program interrupt requests. Refer to Chapter 3 "Virtual Console" for more information.

Memory Management

The memory management facilities allocate and protect user memory by translating logical memory addresses into physical memory addresses and controlling access to physical memory. The facility also provides memory data integrity by appending and checking parity on the contents of physical memory.

This section describes the memory management facilities, provides mapping definitions, details functions, summarizes instructions in table form, defines the relevent instructions, discusses programming considerations, and defines power-up response.

Memory Allocation and Protection

The memory allocation and protection (MAP) unit provides the necessary hardware to control and use more than 64 kilobytes of physical memory. In addition, the MAP provides protection functions to maintain the integrity of a large system.

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

A MAP unit gives several users access to the resources of the computer by dividing the memory space available into blocks assigned to each 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 MAP and the CPU, 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 logical address space is 32,768 words long and is addressed by a 15-bit address.
- *Physical Address.* The address used by the MAP to address the physical memory. The maximum size of the physical address space in a Model 20 system is 2 Mbytes and it is addressed by a 20-bit address. The maximum size of physical memory in a Model 30 system is 1.5 Mbytes.
- *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 process. They translate logical addresses to physical addresses for every memory reference.
- *Data Channel Map.* The set of address translation functions defined by the user-specified map. They translate logical addresses to physical addresses when data channel devices address the memory.
- *Supervisor.* The part of the operating system which controls system functions such as the operation of the MAP unit.

Translation Function

The primary function of the MAP unit is address translation. A user map assigns each logical page of a 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 MAP allows 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 unit provides two types of maps: *User* maps and the *Data channel* map.

User Maps

Each user requires a separate user map. The 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 or another as needed. If there are more than four users, new user maps must be loaded as needed. This is simplified by the use of the LMP instruction which loads a complete map with one instruction and uses relatively little time.

Data Channel MAP

The data channel MAP can access memory without direct control from the user's program. Thus, the 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. The data channel map can be enabled or disabled at any time.

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 MAP can also operate in unmapped mode. In this mode, no address translation occurs, so that addresses issued by the 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 1-7 illustrates the address translation performed by the MAP unit. Each 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 each of up to four users and one map for a data channel. Each 12-bit word in a user's map includes 10 bits that specify the physical page and two bits that indicate a MAP protection code. One map code bit marks the page as write protected or write enable and the other bit specifies that the page is validity protected or unprotected.

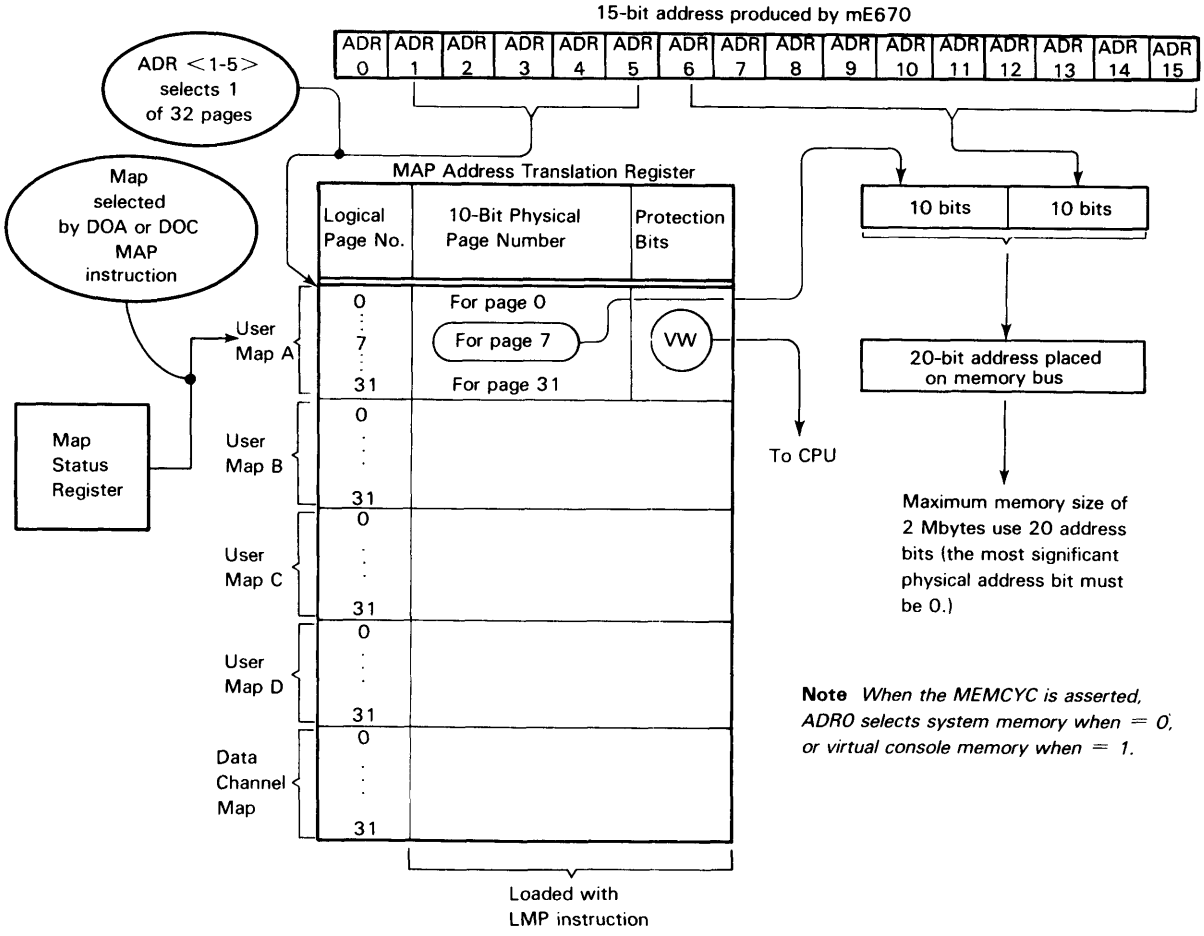


Figure 1-7 MAP address translation

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 MAP provides four types of protection. The MAP flags the nature of each protection violation and causes a MAP protection fault. The four types of MAP protection are:

- Validity protection
- Write protection
- Indirect protection
- I/O protection

Validity Protection

Validity protection protects one user's memory space 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 will cause 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 the MAP feature 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.

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

Write Protection

Write protection allows users to read the protected memory locations, but not to write into them. In this way, the integrity of common areas of memory can be 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 the 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 users. This section should be write protected so that users can read the functions but cannot change them.

Indirect Protection

Indirect protection allows the supervisor to ensure that the CPU will not be placed in an indirection loop. When in an indirection loop without indirect protection, the CPU 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.

Map Protection Faults

When a 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.

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* beyond the maximum physical address space will result in undefined data being returned.

Any attempt to *write* beyond the maximum physical address space will have no effect and will not produce an error.

Load Effective Address Mode

The Load Effective Address (LEF) instruction has the same format as I/O instructions. The 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.

MAP Instructions

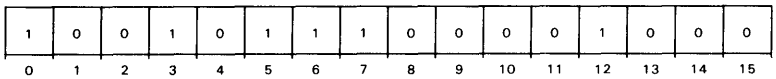
The MAP instructions, shown in Table 1-26, control the actions of the MAP. They are used by the supervisor program to change the mapping functions or to check the status of the various maps. All except Load Map (LMP) are in I/O format using the device mnemonic MAP.

Table 1-26 MAP instructions

Mnem	Instruction	Action
DIA	Read Map Status	Reads the status of the current map.
DIC	Page Check	Provides the identity and some characteristics of the physical page that corresponds to the logical page identified by the immediately preceding **DOC** *ac* **MAP** instruction.
DOA	Load Map Status	Defines the parameters of a new map.
DOB	Map Supervisor Page 31	Specifies the physical page corresponding to logical page 31 of unmapped address space.
DOC	Initiate Page Check	Identifies a logical page; selects the map without changing status.
LMP	Load Map	Loads successive words from memory into the MAP where they are used to define a user or data channel map.
NIOP	Map Single Cycle	Maps one memory reference using the last user map.

Load Map

LMP



Loads successive words from memory into the MAP.

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

Three$_1$ accumulators affect the LMP instruction:

AC0 must contain zero.

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.

[1] *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:

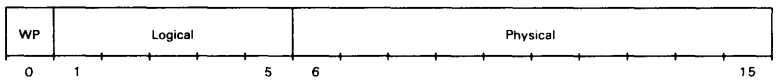
AC0 remains unchanged.

AC1 contains zero.

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

The words loaded into the MAP define the address translation functions for the various 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 the words loaded into the MAP is:



Bits	Name	Contents or Function
0	Write Protect[1]	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.

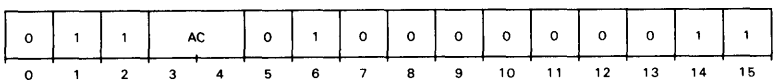
[1] *To declare a logical page invalid, set the Write Protect bit to one and all of bits 6-15 to one.*

NOTE *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.

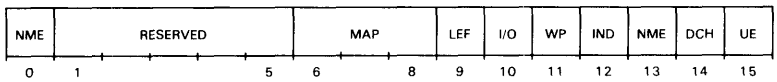
Load MAP Status

DOA *ac*,MAP



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

The format of the specified accumulator is:

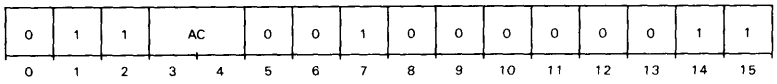


Bits	Name	Contents or Function
0,13	NME	Depending on the bit settings, the next user map enabled will be that for: Bit 0 Bit 13 User enabled 0 0 A 0 1 B 1 0 C 1 1 D
1-5	--	Reserved for future use.
6-8	MAP	Specifies which map will be loaded by the next **LMP** instruction as follows: Bit 6 Bit 7 Bit 8 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**).

If the Load Map Status 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 MAP Status

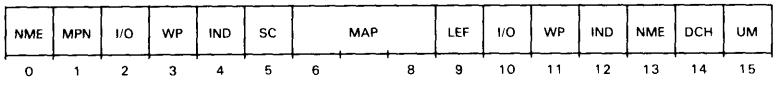
DIA *ac*,MAP



Reads the status of the current map.

Places the contents of the 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:

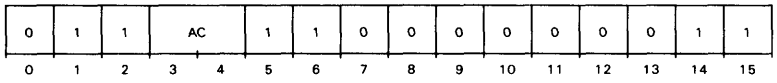
NOTE *The IORST instruction will clear bits 0 to 5 and 13 to 15 of the MAP status register. IORST also turns off the MAP.*



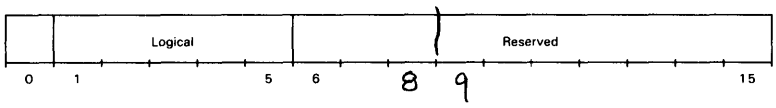
Bits	Name	Contents or Function
1	MPN	MAP state -- 1 indicates mapping
13	NME	Next MAP enabled. Depending on the bit settings, the last **DOA MAP** instruction enabled: Bit 1 Bit 13 User Enabled 0 0 A 0 1 B 1 0 C 1 1 D
2	I/O	If one, the last protection fault was an I/O protection fault.
3	WP	If one, the 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 **LMPA** 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 *ac*,MAP



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



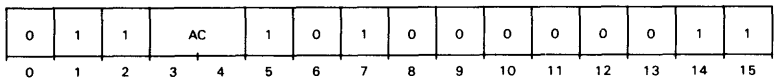
Bits	Name	Contents or Function
0	--	Reserved for future use.
1-5	Logical	Number of the logical page for which the check Page is requested.
6-8	Map	Specify which map should be used for check as follows:

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

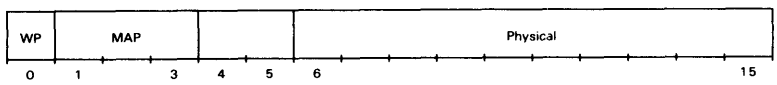
9-15	--	Reserved for future use.

Page Check

DIC *ac*,MAP



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

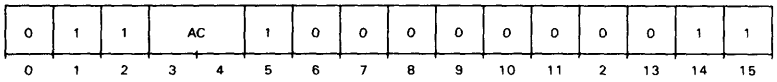


Bits	Name	Contents or Function
0	WP	The write protect bit for the logical page which corresponds to the physical page specified by bits 6-15.
1-3	MAP	The map which was used to perform the translation between logical page number and physical page number as follows: Bit 1 Bit 2 Bit 3 User Enabled 0 0 0 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 the page which corresponds to the logical page given in the preceding **DOC MAP** instruction.

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

Map Supervisor Page 31

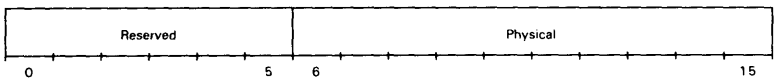
DOB *ac*,MAP



Specifies that mapping take place for a single page of an unmapped address space. 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:



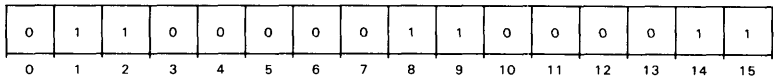
Bits	Name	Contents or Function
0-5	--	Reserved for future use.
6-15	Physical Page	The number of the physical page to which logical page 31 should be mapped when in supervisor mode.

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



The effect of this instruction 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.~~ ~~until~~ after the next memory reference

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
```

then the logical address 407_8 will be mapped using the last enabled user 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
```

then 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.

Programming the MAP

To manage address translation, a memory supervisor routine must maintain information about memory usage, such as the allocation of physical pages between processes, the pages available to processes, the pages that must be shared, and 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. Load a map table into the 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 MAP instruction LMP is 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 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 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 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

The Model 20 and Model 30 SPU generates and appends a parity bit to each byte of data written to memory. In addition, the SPU 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:

Control Register

Fault Code Register

Fault Address Register

Done Flag

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

Enable Parity Checking DOA

Read Parity Fault Address DIA

Read Parity Fault Code DIA

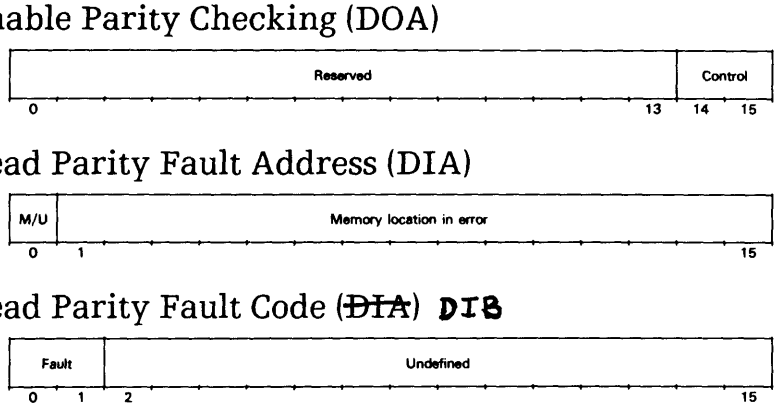
The program also uses I/O Reset IORST instruction to manipulate the state of some elements.

Programming Summary

The following programming summary provides general facility specifications; shows the accumulator formats for the three facility instructions; and explains the facilities response to the START, CLEAR, and PULSE flag commands and the I/O Reset instruction IORST.

Table 1-27 Programming summary: parity checking facility

Mnemonics	PAR
Device code	02 octal
Priority mask bit	None
Parity field	Odd or even



DG-25857

Figure 1-8 Programming summary: accumulator formats

Table 1-28 Programming summary: START, CLEAR, PULSE, 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 address, 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 0s using the I/O Reset instruction IORST. Refer to the Enable Parity Checking instruction for control details.
- *Fault Address Register.* The fault address register stores the 15-bit logical address of the last memory location accessed that contained a parity error. 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 which byte of the accessed memory location contained a parity error. 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:

1. Load the two control bits from a specified CPU accumulator into the control register (DOA).
2. Read a fault address from the fault address register into a specified CPU accumulator (DIA).
3. Read a fault code from the fault code register into a specified CPU accumulator (DIB).

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:

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

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

Identify an interrupting source (INTA).

Initialize the interface (IORST).

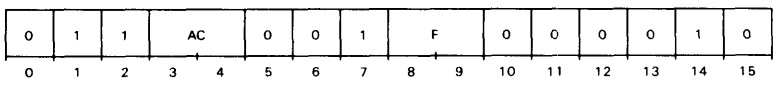
These instructions are fully described in *16-bit Real-Time* ECLIPSE *Assembly Language Programming.*

Assembly language coding and instruction bit patterns are shown for each of the three instructions described. 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.

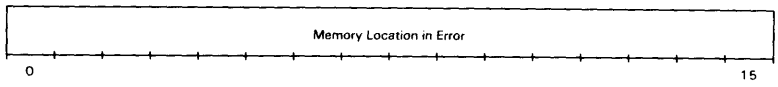
Read Parity Fault Address

DIA*[f]* *ac,2*



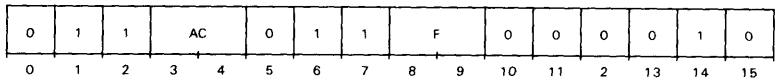
Places the logical address of the last memory location in error in bits 1 to 15 of

the specified AC. Bit 0 is set to one if the error occurred in mapped mode; to zero if the error occurred in unmapped mode. The format of the specified AC is



Read Parity Fault Code

DIB*[f]* *ac*,2



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

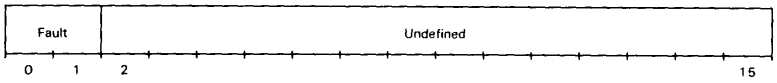
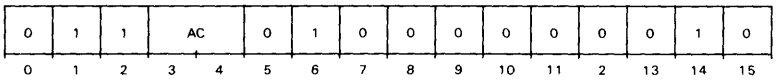


Table DIB__2

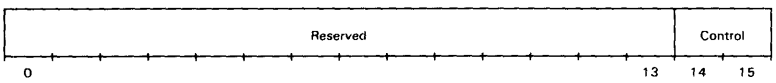
Bits	Name	Contents or Function
0,1	Fault	A 2-bit error code specifying which byte contained a parity error: 10 Upper byte 01 Lower byte 00 No error 11 Both bytes
2-15	--	Undefined.

Enable Parity Checking

DOA*[f]* *ac*,2



The parity checking facility is enabled according to the setting of bits 14 and 15 of the specified accumulator. The contents of bits 0 to 13 of the specified accumulator are not changed. The format of accumulator 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 (odd) parity check field.
		01 Disable checking; write even parity field.
		10 Disable checking; write valid (odd) parity check field.
		11 Enable parity checking; interrupt on parity error.

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.

Program-Accessible Registers

Figure 1-9 shows the program-accessible registers of the system processing unit, their accumulator formats, and the instructions used to access them.

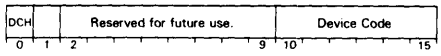
Name **Format**

CPU Status

Read with DIS ac*, CPU

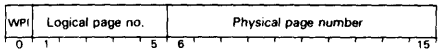
Auto Program Load

Read with DIA ac, APL



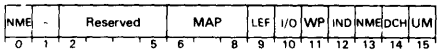
MAP Address

Write with LMP ac



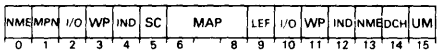
MAP Status Out

Write with DOA ac, MAP



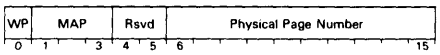
MAP Status In

Read with DIA ac, MAP



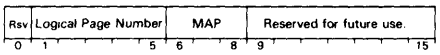
MAP Page Check

Read with DIC ac, MAP



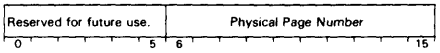
MAP Initiate Page Check

Write with DOC ac, MAP



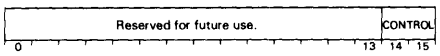
MAP Page 31

Write with DOB ac, MAP



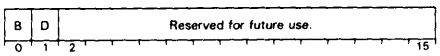
Parity Enable

Write with DOA ac, PAR



Parity Check Status

Read with DIS ac, PAR

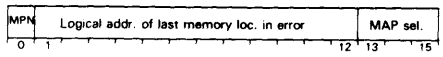


**This accumulator must not be AC0.*

Name **Format**

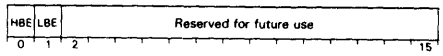
Memory Fault Address

Read with DIA ac, PAR



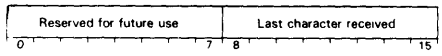
Memory Fault Code

Read with DIB ac, PAR



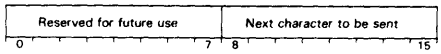
TTI Character Buffer

Read with DIA ac, TTI



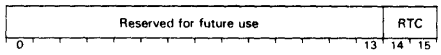
TTO Character Buffer

Write with DOA ac, TTO



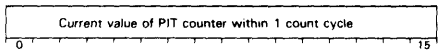
Real Time Clock

Write with DOA ac, RTC



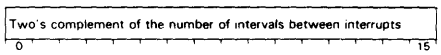
PIT Count

Read with DIA ac, PIT



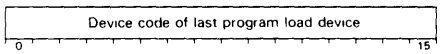
PIT Initial Count

Write with DOA ac, PIT



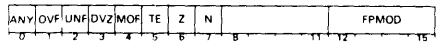
Virtual Console

Read with READS ac



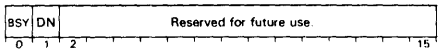
Floating Point Status (first word)

Read/Write with Status Register Instructions



Device Status

Read with DIS ac*, device



**This accumulator must not be AC0.*

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

DG-08947

Figure 1-9 Program accessible registers

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, I/O interface registers are cleared, the state machines in the microI/O bus interface are initialized, bits 0-5, 9-12, 14, and 15 of the MAP status register are cleared, and the CPU status register is cleared except for the Power-up bit (bit 4), which is set to one. The contents of the Page 31 register are set to 37 (octal). The parity fault code and parity enable registers are cleared.

From the Halt state, the CPU enters the virtual console, which clears the Power-up bit and performs a short diagnostic routine. The diagnostic routine checks physical memory locations 0 through 31K, the virtual console scratchpad memory, the CPU, and the console interface. If an error is detected, an error message is displayed and the virtual console program waits for the user to press the Break Key[1]. (The error messages are described in Chapter 3, "The Virtual Console.")

After successfully completing the diagnostic routine, the CPU remains in the virtual console program and issues a prompt to notify the user that the system is ready to perform a program load function.

[1] *Pressing the Break Key after an error code display will allow the virtual console program to run, but the indicated error condition will continue and may cause unpredictable results.*

Powerfail/Autorestart

When an abnormal power condition occurs, the power supply asserts a signal that causes the system I/O integrated circuit (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, and Halts.

If the power failure condition persists longer than 2 ms, the program and data in the CPU and RAM memory will be 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 discussion applies.

If the Halt Dispatch bit in the SIO register is set to 1, the Halt instruction transfers program control to the virtual console when power is restored. The virtual console automatically clears the powerfail interrupt, the instruction in location 0 is executed and control returns to the user's program. (Refer to Chapter 3, "Virtual Console."

NOTE *If the virtual console is not entered within 150 ms after the powerfail interrupt, it is possible that the powerfail condition is cleared before it is tested. If this occurs, the virtual console will retain control but will not automatically execute the location 0 instruction.*

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 still persists the CPU Done bit remains set.*

Programming Basic Model 20 and Model 30 I/O Interfaces

2

This chapter describes the programmable elements of the basic Model 20 and 30 I/O interfaces; summarizes their specifications, instruction accumulator formats, and 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 basic I/O interfaces include the Asynchronous Communications Interface, Real-Time Clock, Programmable Interval Timer, and Diskette Subsystem

Asynchronous Communications Interface

The asynchronous communications interface is a programmed I/O interface which allows full-duplex communications between the CPU and a terminal connected to the A connector of the Model 20 and Model 30 SPU printed circuit card. The interface contains both a double-buffered transmitter (TTO) section and a receiver (TTI) section that function as separate devices. The transmitter section is assigned device code 11_8 and the receiver section is assigned device code 10_8.

NOTE *The Model 20 and Model 30 asynchronous communications interface receives and transmits eight bit data characters without parity. If the terminal device being used with the Model 20 and Model 30 SPU asynchronous communications port operates with a data character length of seven bits, you should configure the device to operate with "mark parity". When receiving data characters from a 7-bit terminal 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 asynchronous 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 interface 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:

Write Character (DOA)

Read Character (DIA)

I/O Skip (SKP)

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 asynchronous communications interface 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: interface specifications

Mnemonics	
Receiver	TTI
Transmitter	TTO
Device codes	
Receiver	10 octal
Transmitter	11 octal
Priority mask bits	
Receiver	14
Transmitter	15
Line speed	50, 75, 110, 134.5, 150, 200, 300, 600, 1200, 1800, 2000, 2400, 4800, 9600, 19200, or 38400 baud
Character structure	8 data bits no parity 2 stop bit (50, 75, 110, and 134.5 baud) 1 stop bits (150 baud and above)
Programmed I/O latency	219 mS at 50 baud to 0.26 ms at 38,400 baud
Modem control signal used	Clear To Send

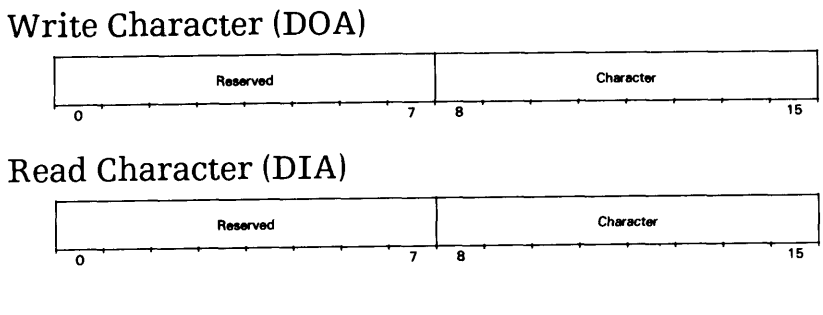


Figure 2-1 Programming summary: accumulator formats

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

f = S	Receiver section Sets the receiver Busy flag to 1, the Done flag to 0. 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.

Registers and Flags

The program-accessible registers of the asynchronous communications interface include the receive and transmit registers. Its program-accessible flags include the Busy, Done, Interrupt Disable, and Break flags.

Receive Register The receive register stores the 8-bit assembled character that is 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. The transmit register is double-buffered. Characters to be transmitted are loaded into one buffer by the program and transferred to the second buffer, when it it empty, to be transmitted over the communications line. When *Clear To Send* is asserted by the terminal, the transmitter section disassembles the character contained in the second buffer of 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, indicates a character is being received over the asynchronous communications line. This flag remains set while a character is being 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, however program manipulation of this flag has no effect in the Model 20 nor Model 30 system. 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 receiver Busy flag using the *I/O Skip* instruction (SKP) addressed to the receiver section of the asynchronous communications interface (TTI).

The Busy flag of the transmitter section, when set to 1, indicates an output character is contained in the first buffer of the transmit register. This flag remains set until the character has been transferred into the second buffer of the transmit register where it will be 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 (SKP) addressed to the transmitter section of the asynchronous communications interface (TTO).

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 or Clear flag commands. A program can test the Done flag using the *I/O Skip* instruction (SKP) addressed to the receiver section of the asynchronous communications interface.

The transmitter Done flag, when set to 1, initiates an interrupt request to the CPU (unless interrupts are disabled) and specifies that the interface has completed transfer of the character in the first buffer of the transmit register to the second buffer of the register (the program can load another character into the transmit

buffer). (There is no indication to the program when transfer of the character in the second buffer has been completed to the communications line.) 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 (SKP) addressed to the receiver section of the asynchronous communications interface.

Interrupt Disable Flags The receiver and transmitter Interrupt Disable flags enable and disable their respective asynchronous communications 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 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.*

Break Flag The receiver Break flag, when set to 1 and enabled, initiates an nonmaskable interrupt request to the Model 20 and Model 30 SPU and specifies that the receiver section of the interface has detected a "break" character on the communications line. The program sets the break flag to 0 using the *I/O Reset* instruction (IORST) or the Pulse flag command. The virtual console program can interrogate the Break flag by issuing a *Read CPU Status* instruction (DIS) and examining bit 3 of the word returned to a specified accumulator. The Break flag is not ordinarily manipulated by the user.

I/O Instruction Set

Two programmed I/O instructions enable you to program the asynchronous communications interface 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 asynchronous communications interface 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 interface — busy, done, or idle (SKP).

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

Enable/disable the transmit and/or receive section of the asynchronous communications interface interrupt facility (MSKO).

Identify an interrupting source (INTA).

Initialize the asynchronous communications interface (IORST).

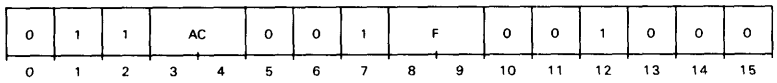
The latter instructions are fully described in detail in *16-bit Real-Time ECLIPSE Assembly Language Programming*.

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 20 and Model 30 asynchronous communications interface device codes; 10_8 for the receiver section; 11_8 for the transmitter section. Mnemonics TTI or 10_8 used in the assembly language coding, address the receiver section of the asynchronous communications interface, while mnemonics TTO or 11_8 address the transmitter 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 asynchronous communications interface.

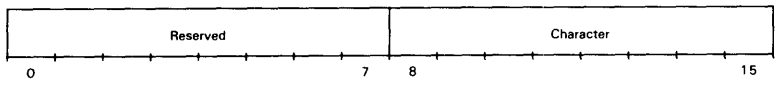
Read Character

DIA*[f]* *ac*,ITI



Reads the character most recently received by the asynchronous communications interface.

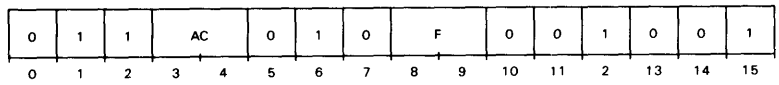
Loads the contents of the receive register into bits 8-15 of the specified accumulator. Bits 0-7 of the specified accumulator are set to zero. After the data transfer, the command sets the Busy and Done flags of the asynchronous communications interface receiver section according to the function specified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



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

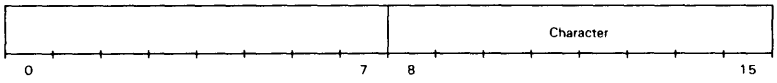
Write Character

DOA*[f]* *ac*,TTO



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 asynchronous communications 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 asynchronous communications interface involves the following steps:

Writing characters

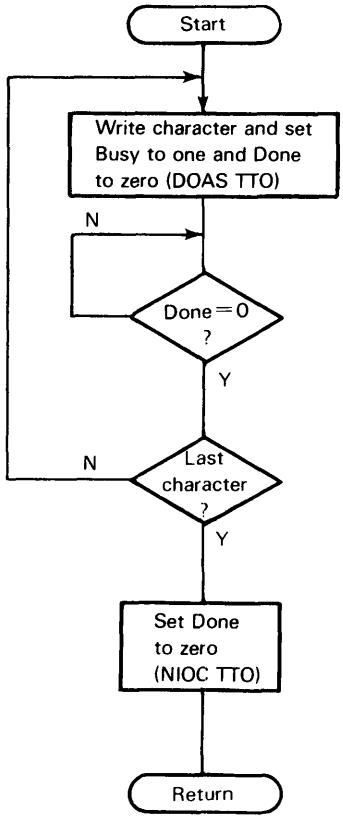
Reading characters

Servicing interrupts

Writing Characters

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

1. Issue an *I/O Skip* instruction (SKP) 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 2-2 Writing characters

Reading Characters

Use the following programming sequence to obtain a character received by the interface from the communications line.

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 or Clear flag command clears the receiver interrupt by setting the Busy flag to 1 and the Done flag to 0.

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 the time interval indicated in Table 2-3. The time intervals tabulated in the table are based on the assumption that characters transmitted at 50 to 134.5 baud contain 11 bits (including one start and two stop bits), and characters transmitted at 150 to 38,400 baud contain 10 bits (including one start and one stop bit).

Table 2-3 Asynchronous communications interface: timing considerations

Baud	Maximum Allowable Programmed I/O Latency (ms)
50	219.00
75	146.00
110	100.00
134.5	74.35
150	66.66
200	54.75
300	33.33
600	16.66
1200	8.33
1800	5.55
2000	5.00
2400	4.16
4800	2.08
9600	1.04
19,200	0.52
38,400	0.26

After the transmitter Done flag sets to one, the program should provide another character to the interface within the time period of the designated baud rate indicated in Table 2-3 to maintain the maximum transfer rate.

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 stop bits. See Figure 2-3.

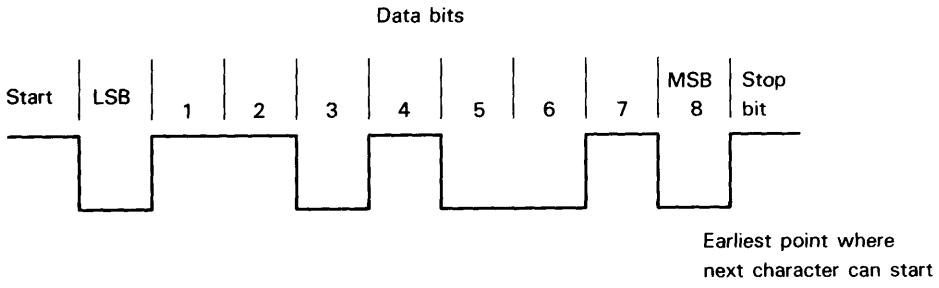


Figure 2-3 Asynchronous transmission of 8-bit character

A 7-bit character is transmitted as a start bit, then 7 data bits (LSB first), a parity bit, and stop bits. See Figure 2-4.

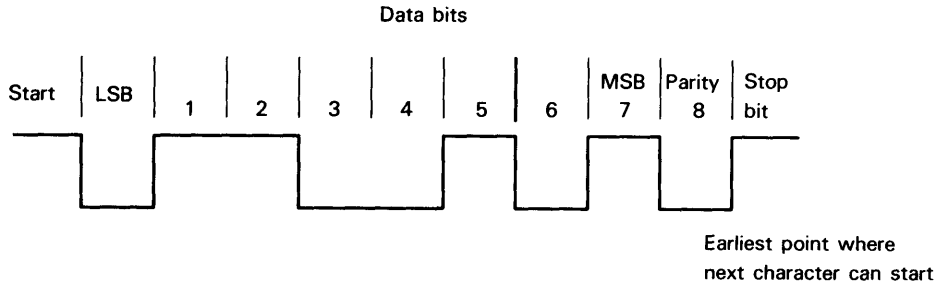


Figure 2-4 Asynchronous transmission of 7-bit character

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 20 and 30 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 (SKP)

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

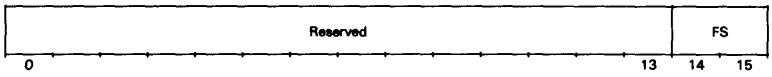
Programming Summary

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 2-4 Programming summary: RTC

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

Select Frequency (DOA)



DG-25847

Figure 2-5 Programming summary: accumulator formats

Table 2-5 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 (SKP) 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 (SKP) 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).

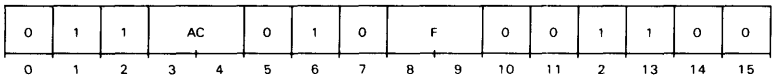
The latter instructions are fully described in detail in *16-bit Real-Time* ECLIPSE *Assembly Language Programming*.

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) 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 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	1000 Hz

NOTE *Do not select line frequency interrupts unless the model 20 or 30 system contains the optional line frequency clock generator card.*

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 switch-selectable increments ranging from 1 microsecond to 1 millisecond as listed in Table 2-6. 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.

Table 2-6 PIT rates

Selected Frequency (kHz)	Time interval	
	Minimum	**Maximum**
1	1 millisecond	65.536 seconds
10	100 microseconds	6.5536 seconds
100	10 microseconds	655.36 milliseconds
1000	1 microsecond	65.536 milliseconds

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 (SKP)

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: PIT

Mnemonic	PIT
Device code	43 octal
Priority mask bit	6
Time intervals	1 uS to 65.536 seconds
Count rate	1 uS, 10 uS, 100 uS, or 1 mS

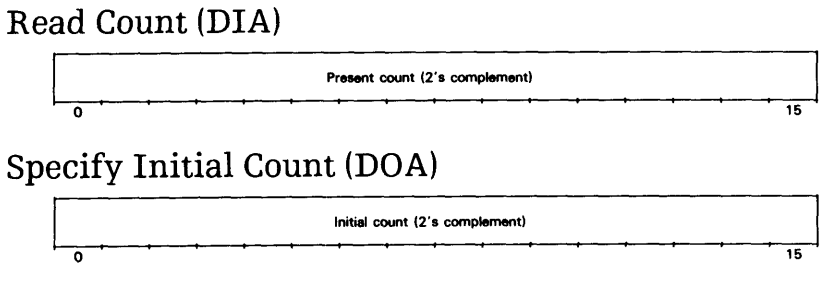


Figure 2-6 Programming summary: accumulator formats

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 zero's.

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. 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. A program can test the Busy flag using the *I/O Skip* instruction (SKP) 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 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 (SKP) 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 instructions 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:

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

Read 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 (SKP).

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*.

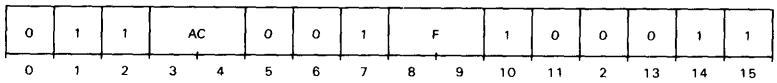
Assembly language coding and instruction bit patterns are shown for each of the two instructions described. The device code field of the instruction is shown with the standard programmable interval timer device code, 43_8. Mnemonics

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.

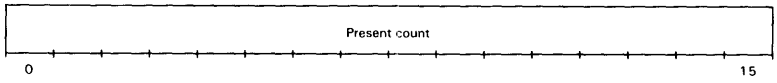
Read Count

DIA*[f]* *ac*,PIT



Reads the count remaining before an interrupt is requested.

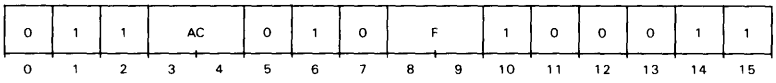
Loads the contents of the counter into bits 0-15 of the specified accumulator. After the data transfer, the command sets the Busy and Done flags according to the function specified by *f*. The format of the specified CPU accumulator after the transfer is:



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

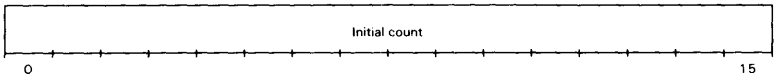
Specify Initial Count

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. Refer to Table 2-6 for a list of PIT rates.

NOTE *The selected frequency of the timer is determined by switches on the SPU card, and is not affected nor can it be read by programming.*

Starting the Counting Cycle To start the timing cycle, 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 time interval expires, the timer sets the Done flag to one, thus initiating a 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 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 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 20 and Model 30 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 (SKP)

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 subsystem specifications in Table 2-9; describes diskette formats in Table 2-10; shows the accumulator formats for the five interface instructions in Figure 2-7; 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.

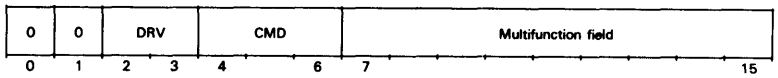
Table 2-9 Programming summary: general subsystem specifications

Mnemonic	DEO
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

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	High-byte/Low byte 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	Low-byte/High-byte High byte / Low byte

Specify Command and Diskette Address (DOA)

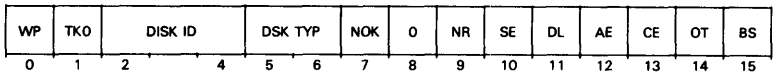


Commands

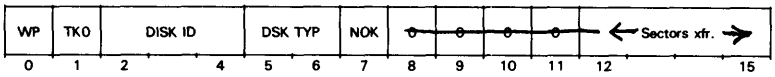
000	Read
001	Read header
010	Write
011	Format track
100	Diagnostic operation
101	Get number of sectors transferred
110	Seek
111	Recalibrate

Read Diskette Status (DIA)

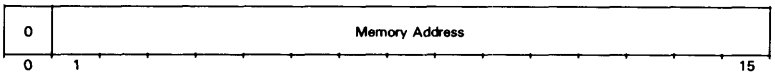
Context: A Get Number of Sectors Transferred was not previously issued.



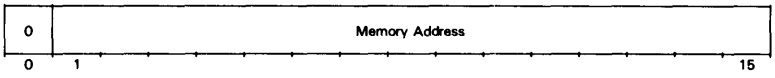
Context: A Get Number of Sectors Transferred was previously issued.



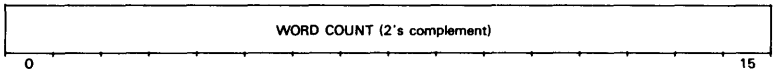
Load Memory Address Register (DOB)



Read Memory Address Register (DIB)



Load Word Count (DOC)



DG-25849

Figure 2-7 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.

Notes
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 (SKP) addressed to the diskette subsystem.

Done Flag The Done flag, when set to 1, initiates an interrupt request to the Model 20 and Model 30 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 (SKP) 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 (SKP).

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.

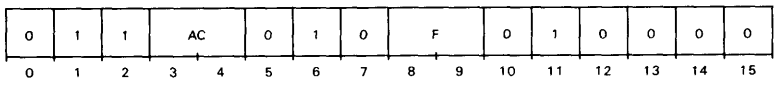
The latter instructions are fully described in detail in *16-bit Real-Time* ECLIPSE *Assembly Language Programming.*

Assembly language coding and instruction bit patterns are shown for each of the five instructions described. The device code field of the instruction is shown with the standard Model 20 and Model 30 diskette interface device code, 20_8. Mnemonics DE0 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.

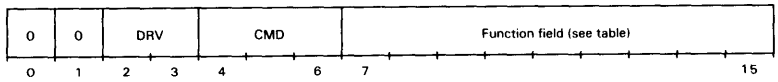
Specify Command and Diskette Address

DOA*[f]* *ac*,20



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	Function 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

Page 2-28

Figure 2-8 Read header command:format of words returned

The **NOTE** in this figure should read:

Sector length code = 2 (base 16) = 512 bytes/sector

Page 2-33

Instruction title, "Memory Address Register", should read:

Read Memory Address Register

Page 2-37

Figure 2-9 Head positioning

Busy = 0 decision block in right column should read:

Busy = 1

Page 2-39

Figure 2-10 Read or write data

Busy = 0 ? decision block in left column should read:

Busy = 1 ?

Page 2-47

Figure 2-14 Format flowchart

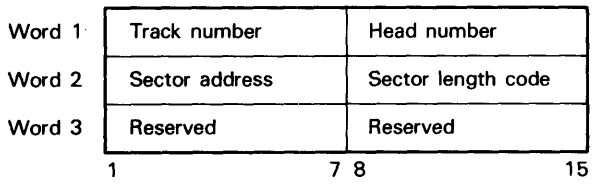
The No path from the Busy = 1 ? decision block in the left column should lead to the Position heads block only. Delete horizontal line pointing to the right.

Add Yes to the vertical line below the Errors ? decision block, and No to the horizontal line extending right from the Errors ? decision block (right column).

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 Figure 2-8.
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 transferred	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 Table 1 can be different for each drive connected to the diskette interface.

SM	FM	DH			SD		HS	BS
7	8	9	10	11	12	13	14	15



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

2

ID-00620

Figure 2-8 Read header command: format of words returned

Table 2-13 Operating mode bit descriptions

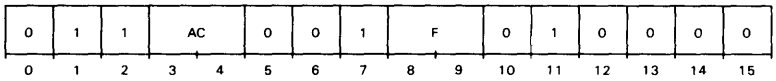
Bit(s)	Name	Function
7	Set Mode (SM)	Specifies Recalibrate or Recalibrate and Set Mode, as follows: 0 = Recalibrate (bits 8-15 are ignored) 1 = Recalibrate and Set Mode
8	48/96 (FN)	Specifies number of positioning step pulses per track as follows: 0 = one step pulse per track (default value)[1] 1 = two 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 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.*

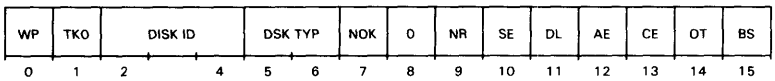
Read Diskette Status

DIA*[f]* *ac*,20



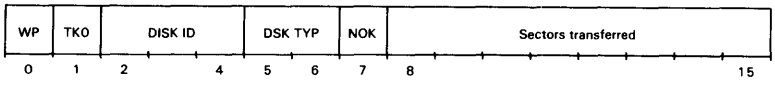
Transfers the status, or selective status along with a number of sectors transferred, of the currently-selected diskette drive to the processor.

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	Subsytem Identity	Identifies the diskette subsystem; always set to a value of 2_8 (010_2).
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	Specifies that the interface failed its self-test after either a power-up or a self-test diagnostic commmand. This bit resets when the interface passes its self-test diagnostic.
8		Reserved (always set to 0).
9	Not Ready	Specifies that the selected diskette drive is not ready to accept commands. This may be because the drive is not up 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 Timeout	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.

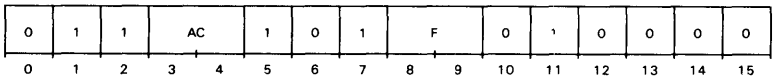
If the previous command issued to the diskette interface was a *Get Number of Sectors Transferred,* this command loads selective status information of the currently selected diskette drive and a sectors transferred count into specific bit positions of the specified CPU accumulator. After the data transfer, 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 a value of 2_8 (010_2).
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 commmand. 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.

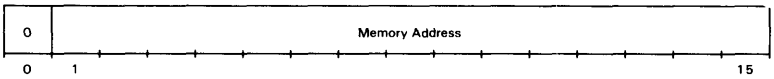
Load Memory Address Register

DOB*[f]* *ac,*20



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

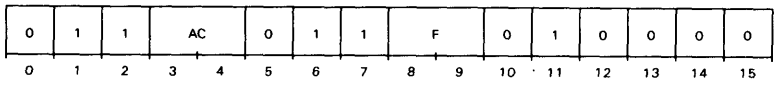
Bits 1-15 of the specified accumulator are loaded into the memory address register. After the data transfer, the command sets the interface Busy and Done flags according to the function specified by *f*. The format of the specified accumulator before and after the transfer is:



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.

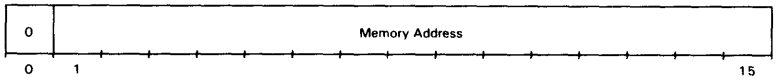
Read Memory Address Register

DIB*[f]* *ac*,20



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

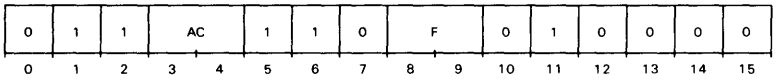
Loads the contents of the memory address register into bits 1-15 of the specified accumulator. Bit 0 of the specified accumulator is set to zero. After the data transfer, the command sets the interface Busy and Done flags according to the function specified by *f*. The format of the specified CPU accumulator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



Bit(s)	Name	Function
0	—	Reserves (always set to 0).
1-5	Memory Address	Location in memory to be accessed for the next data channel transfer.

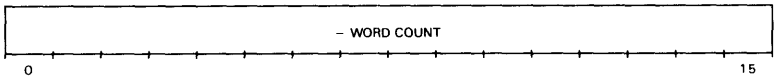
Load Word Count

DOC*[f]* *ac*,20



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
1. *Word count is calculated as the number of sectors times the number of words per sector.*
2. *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 discussion to come 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 (SKP) 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.)
3. Issue a *Read Diskette Status* instruction (DIA) and verify that the selected drive is ready to perform an operation.

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, as 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 and 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, as 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 2-9. Make sure the diskette subsystem is ready to perform an operation, as 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.

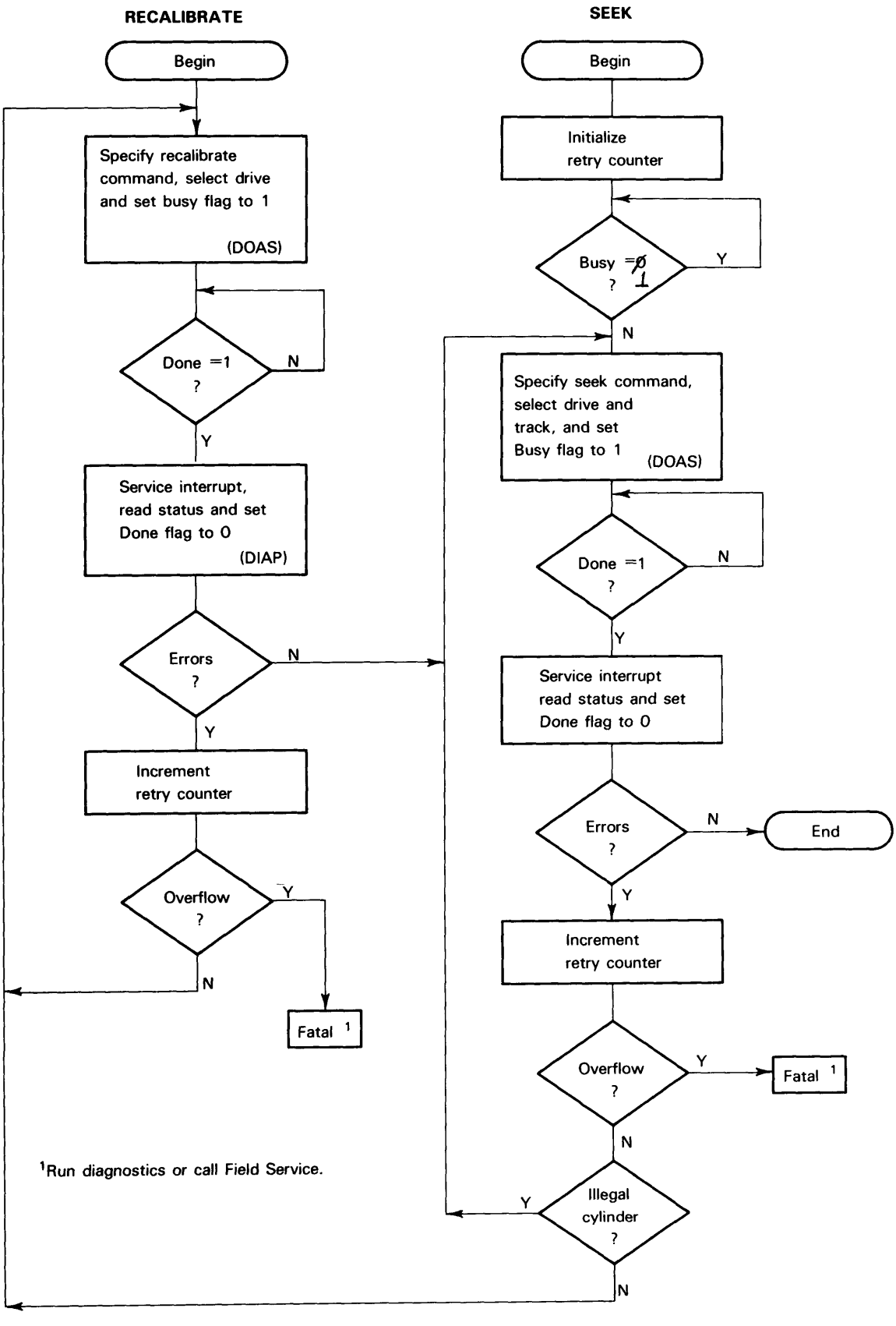


Figure 2-9 Head positioning

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 coditions.

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

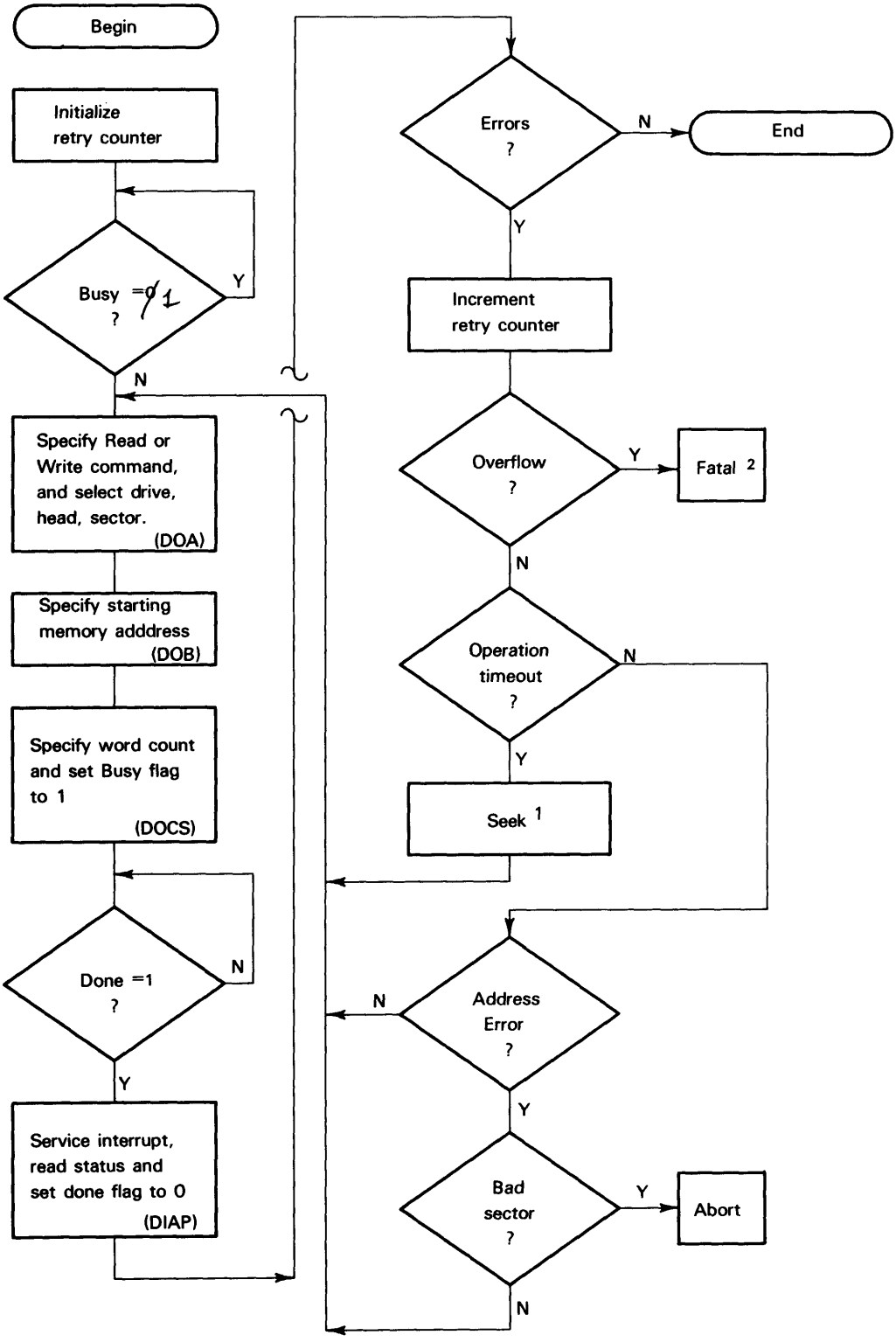
Be sure that the heads are positioned over the desired cylinder. (Refer to "Positioning the Read/Write Heads" in this section.)

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 2-10. Ensure that the diskette subsystem is ready to perform an operation, as 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 2-10 Read or write data

Continue with the following programming sequence to write data onto the diskette. Refer to Figure 2-10. Ensure that the diskette subsystem is ready to perform an operation, as 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

About Diskette Formatting

Data General diskettes are completely formatted before they are shipped. You do not need to reformat a diskette unless it develops a problem. Data General supplies a stand-alone reformatting program on the Customer Mode Diagnostic diskette. Its operation is described in *Testing a Model 20 and Model 30 System* (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 20 and Model 30 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 20 or Model 30 diskette subsystem, as indicated in Figure 2-11 through Figure 2-13.

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

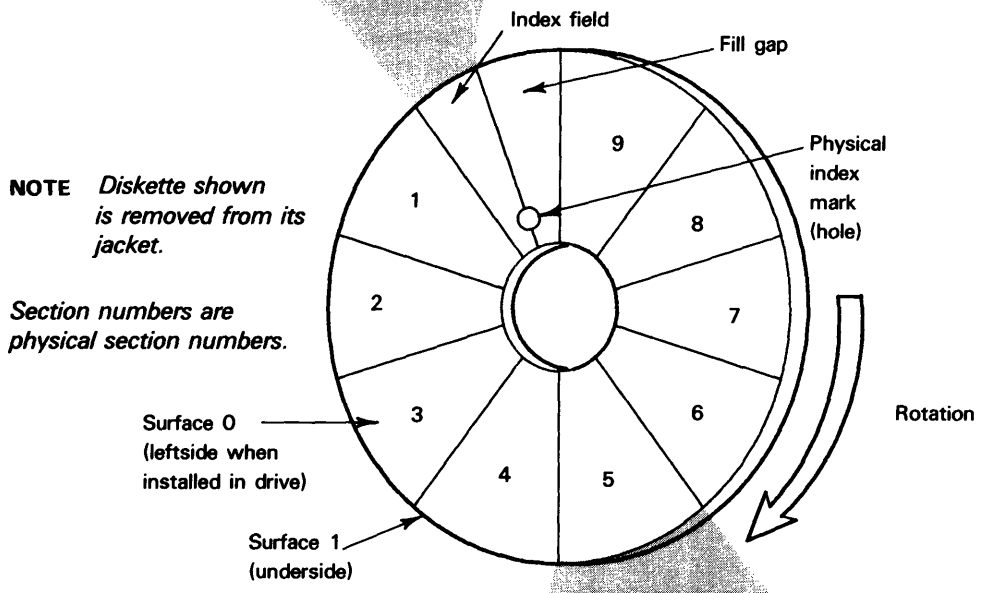
Each diskette has two surfaces, and each surface contains 40 recording tracks. The top surface is labeled 0. One particular track position of each surface constitutes a diskette cylinder. Data General standard formatted diskettes contain nine sectors per track, physically addressed 1 through 9. IBM PC 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 can not 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 2-11 Data General standard 9 sector, 512 bytes/sector format

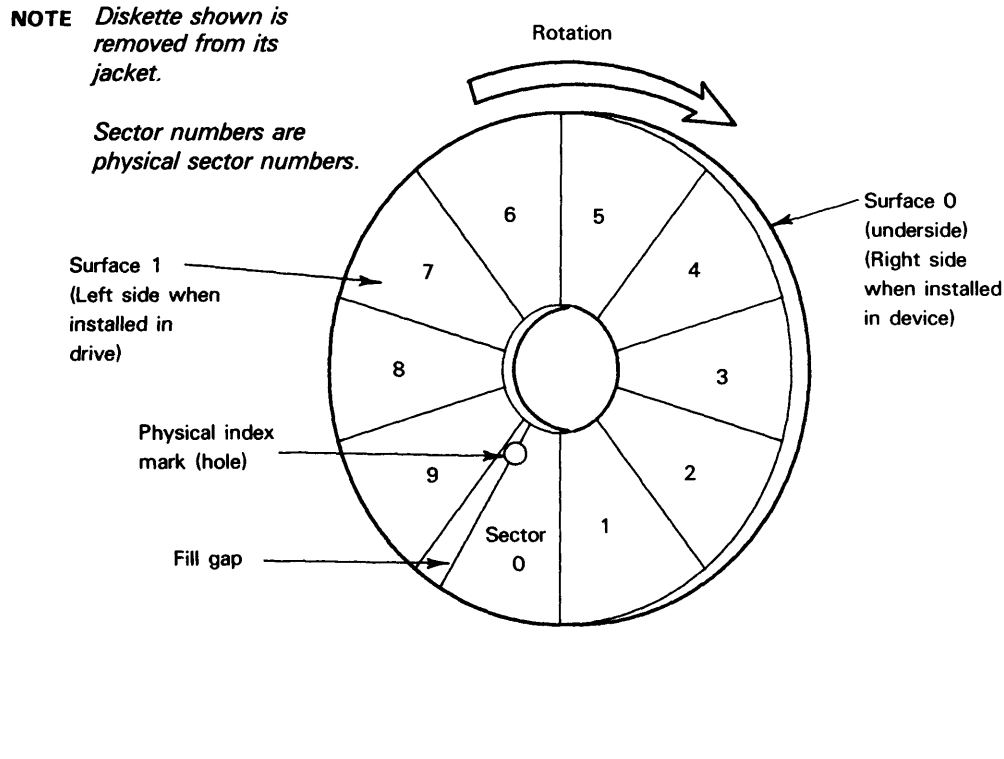
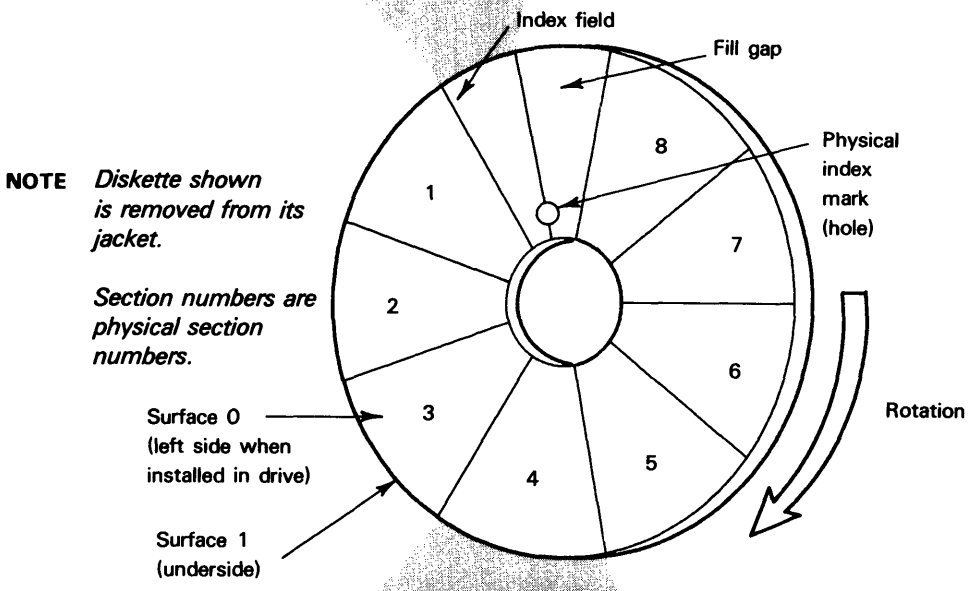


Figure 2-12 Data General MPT/100 10 sector, 512 bytes/sector format

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	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-00683

Figure 2-13 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, a buffer must be set up in memory that contains the binary image of the entire track. This buffer must contain the correct number of words required for the format that is to be output to the diskette . In addition, the binary image in this buffer must be set up to implement the specified format. The image must contain specially coded bytes to inform the interface when to write index marks, address marks, and CRC bytes. These bytes are:

F5 Write address mark

F6 Write index mark

F7 Write calculated CRC (two bytes)

Table 2-14 and Table 2-15 show examples of a memory buffer set up to format diskettes in standard Data General and IBM PC formats. (Note that only the first sector is shown in detail and it must be repeated for remaining sectors of the track.)

Use the following programming sequence to reformat a diskette track. Refer to Figure 2-14. Ensure that the diskette subsystem is ready to perform an operation as explained earlier under "Initiating an Operation."

1. Set up the required format buffer in memory. Refer to Table 2-14 and Table 2-15.

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.

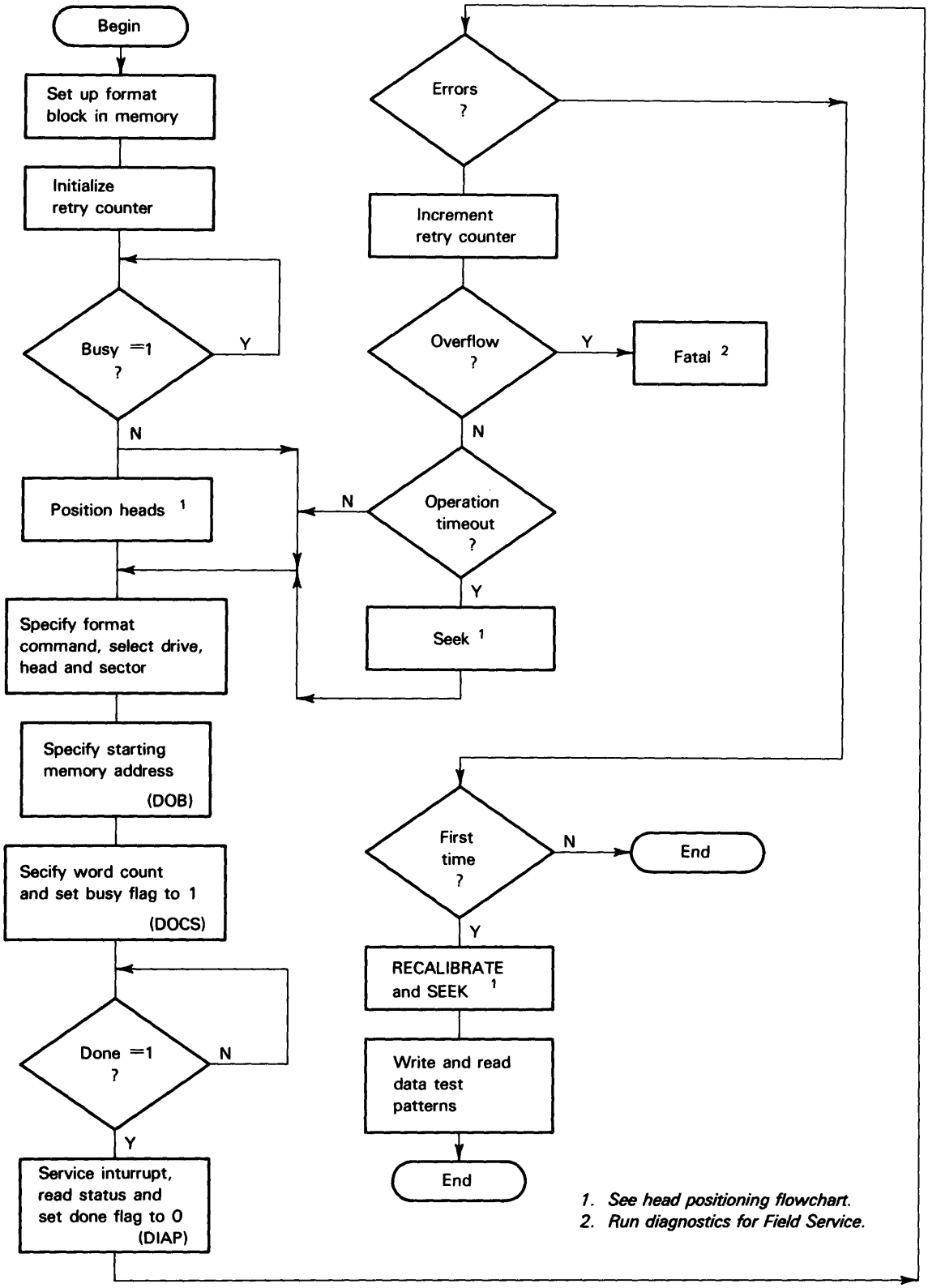
6. Issue a *Load Word Count Register* instruction (DOC) with a Start flag command, using the appropriate accumulator bits to specify the number (in two's complement) of 16-bit words to be transferred. The Start flag command sets the interface's Busy flag to 1, sets the Done flag and Interrupt request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



ID-00654

Figure 2-14 Format flowchart

Table 2-14 Format buffer for Data General 9-sector, 512 bytes/sector format

Number of words (decimal)[2]	High byte value (hex)	High byte value (hex)	Function
40	4E	4E	Gap 4
6	00	00	Sync zone
1	F6	F6	Write index mark code
1	F6	FC	Write index mark
25	4E	4E	Gap 1
6	00	00	Sync zone
1	F5	F5	Write address mark code
1	F5	FE	Address mark
1	(Track #)	(Surface #)	Address field
1	(Sector #)	02	Address and bytes/sector field
1	F7	4E	Write CRC code and first gap 2 byte
10	4E	4E	Gap 2
1	4E	00	Gap 2 and first sync zone byte
5	00	00	Sync zone
1	00	F5	Last sync zone byte and write data mark code
1	F5	F5	Write data mark code
1	FB	xx [1]	Data mark and first data byte
255	xx [1]	xx [1]	Data
1	xx [1]	F7	Last data byte and write CRC code
40	4E	4E	Gap 3
2608	Repeat the above block 8 times (once for each sector), starting after Gap 1 and changing the data in the sector number byte to correspond to each sector.		
200	4E	4E	Fill gap (accounts for speed variations of the diskette drive)

[1] *xx = any hex value other than F5 through F7.*
[2] *Total word count = 3,207*

Table 2-15 Format buffer for IBM PC 8-sector, 512 bytes/sector format

Number of words (decimal)[2]	High byte value (hex)	High byte value (hex)	Function
40	4E	4E	Gap 4
6	00	00	Sync zone
1	F6	F6	Write index mark code
1	F6	FC	Write index mark
25	4E	4E	Gap 1
6	00	00	Sync zone
1	F5	F5	Write address mark code
1	F5	FE	Address mark
1	(Track #)	(Surface #)	Address field
1	(Sector #)	02	Address and bytes/sector field
1	F7	4E	Write CRC code and first gap 2 byte
10	4E	4E	Gap 2
1	4E	00	Gap 2 and first sync zone byte
5	00	00	Sync zone
1	00	F5	Last sync zone byte and write data mark code
1	F5	F5	Write data mark code
1	FB	xx [1]	Data mark and first data byte
255	xx [1]	xx [1]	Data
1	xx [1]	F7	Last data byte and write CRC code
40	4E	4E	Gap 3
2282	Repeat the above block 7 times (once for each sector), starting after gap 1, and changing the data in the sector number byte to correspond to each sector.		
500	4E	4E	Fill gap (accounts for speed variations of the diskette drive)

[1] *xx = any hex value other than F5 through F7.*
[2] *Total word count = 3,181*

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 Model 20 and Model 30 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 before the drive can execute any command. The spindle motors deenergize if no diskette commands are received within approximately 30 seconds. If the spindle motors are not energized when a diskette command is received, an interface generated one-second delay allows the spindles to come up to speed before the command is executed.

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 conditons, 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 is not ready to accept commands. 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 Not Ready flag is 1 when a Read, Read Header, Write, Format Track, Seek or Recalibrate command is initiated, the operation is not performed. 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

Not Ready

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

In addition, following power-up, a self-test diagnostic is run on the diskette interface card. If any errors are detected during this self-test, a light-emitting diode (LED) lights up on the interface card. The LED remains lit until another self-test sequence runs and completes successfully. While the LED is enabled, the Not OK status bit is set in the diskette interface status information. Diskette status information can be interrogated by the CPU. (See the description of *Read Diskette Status* under "I/O Instruction Set" in this chapter.)

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 Model 20 and Model 30 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 data channel facility of the microI/O bus. 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 vitrual console is entered; defines virtual console cells; discusses the console commands and their formats, and how typographical errors are corrected. A description of errors caused by incorrectly entered commands concludes the chapter.

The virtual control is a program that can aid you in working with the Desktop Generation Model 20 and Model 30 computer system. It allows you to interact with the computer through the terminal that is connected to the Model 20 and Model 30 system processor unit (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 *Input/Output (I/O) interrupts are disabled when the virtual console is executing. 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 Model 20 and Model 30 (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 (0.75 second) self-test routine; then, if the test completes successfully, the virtual console program retains control and issues a prompt to the system console. If the test fails to complete successfully, the virtual console program issues a error code to the system console. Error codes and their description are presented in "The Control-G Command" later in this chapter.

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 Break Key of the system console and the break function on the SPU card is enabled by setting SIO switch 2 on (to 1).

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."

Page 3-3

Table 3-1 Internal cells

Footnote 2 should read:

Refer to *16-bit Real-time ECLIPSE Assembly Language Programming*, "Program Accessible Registers" in Chapter 1, or the CPU register contents table in Chapter 4 for the contents of this register.

Footnote 3 should read:

Refer to the *Reads* instruction in Chapter 1 for the contents of this register.

Add footnote 4 which reads:

Refer to the *Read MAP Status* instruction in Chapter 4 for the contents of this register.

Change the footnote number for the MAP status register to 4 instead of 3.

Page 3-6

Under page heading, "Setting Breakpoints",

The fourth line under "Examples:" should read:

3 423 Breakpoint 3 is at address 423

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), or to Tables 00 and 00 in this manual, for the contents of this register.*

[3] *Refer to the programmer's reference listed above, or to p. 0 in this manual, 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. Data entered by the user appears in bold face.

A! 3A 000003A 000100<CR> AC3 contains 100.

000004A 000704/000704
024132<NL>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.)
^G	Executes the power-up self-test sequence.*
*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 3-1.)
*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.

** ^G represents the simultaneous depression of the console CTRL and G keys.*

Setting Breakpoints

NOTE *Breakpoints will not work and should not be used if the Halt Dispatch bit in the SIO switch register is not set to 1.*

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.
423	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*su04. 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, the L or H command for causing a program load, and the !^G! command for initiating the power-up self-test sequence.

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 any program load of whatever type, the state of the accumulators will be indeterminate.*

The Control-G Command

The !^G! command causes the virtual console to execute the power-up self-test sequence. This test is not meant to be a definitive hardware test. It is designed primarily to detect faults that would prevent the loading of diagnostic programs.

NOTE *Because the memory-test portion of the power-up self-test is destructive to user and virtual console memory, a program load must be performed after the self-test.*

The virtual console flags any errors that occur by outputting an error code to the system terminal as follows:

An I is output when an I/O fault occurs.

An H is output when a virtual console memory fault occurs.

An M is output when a user memory fault occurs.

Once the virtual console has output an error flag, nothing further will happen until the user depresses the Break Key. If there are no errors, or the user presses the BREAK key after an error flag has been output, the virtual console will issue a prompt to the system console.

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 20 and Model 30 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

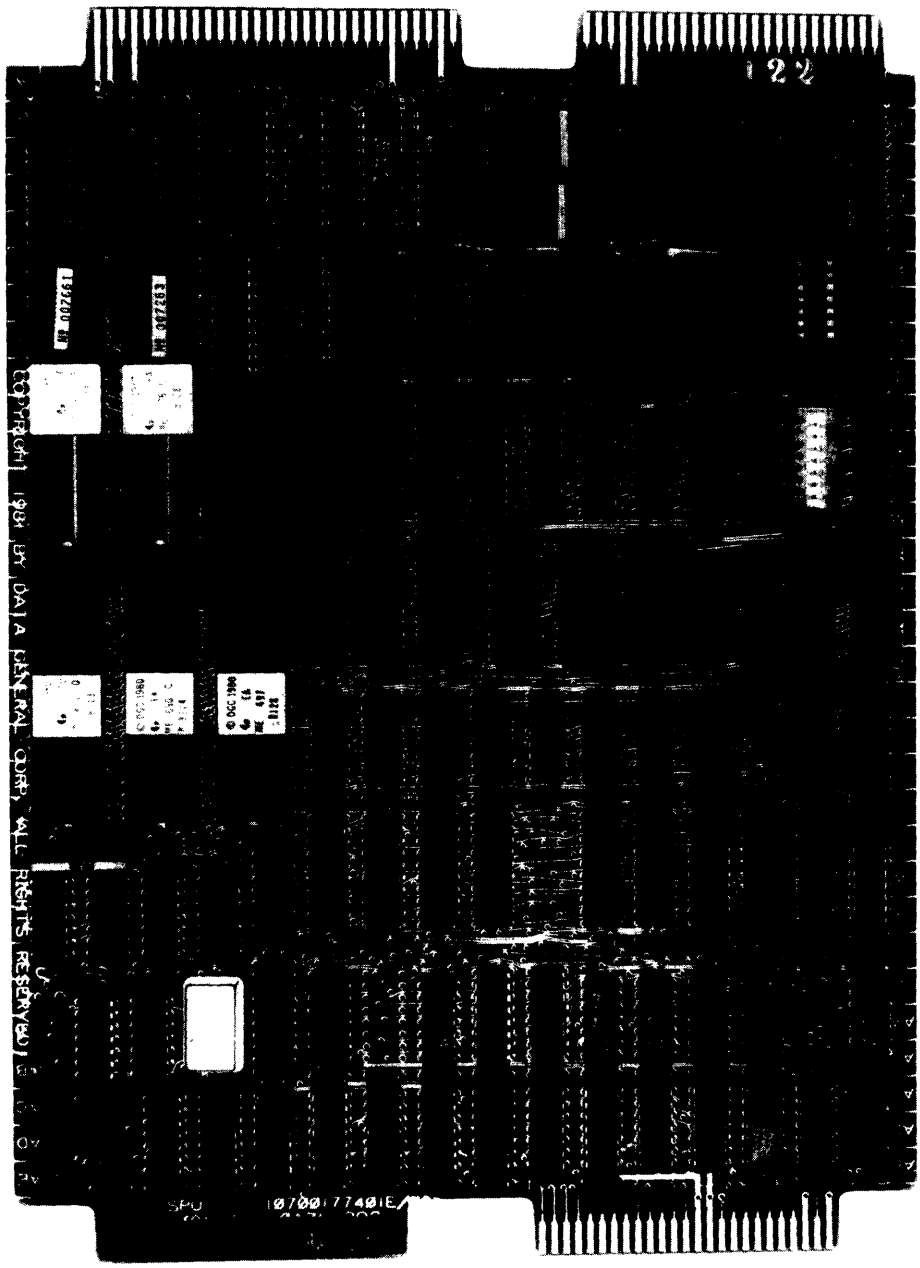
TWO

Theory of Operation

System Processing Unit

4

This chapter describes the major elements of the Model 20 and Model 30 unit; describes power monitoring and initialization; provides power requirements; details interconnections with the system; and explains its functional theory of operation. Tables of internal and external signal descriptions conclude the chapter.



PH-0551

Figure 4-1 The Model 20 and Model 30 SPU

Designed as the basic module in Data General's Desktop Generation computer systems, the system processing unit (SPU) cards, shown in Figure 4-1, consists of the following elements:

A central processing unit (CPU) based on a microECLIPSE mE670 16-bit CPU integrated circuit (IC) and two or three external microcontroller chips (XMCs);

A multidevice section based on a mE676 system I/O (SIO) IC, containing a powerfail monitor, a programmable interval timer, a real-time clock, a

Page 4-3

Under page heading, "CPU and XMCs", The second paragraph should read:

A kernal of these microinstructions for the ECLIPSE instruction set reside in the CPU IC, while microinstructions for the remaining instructions reside in two or three supporting XMCs.

The first sentence of the fourth paragraph should begin:

The two XMCs of the Model 30 SPU....

Page 4-5

Under page heading, "Multidevice Section",
The first sentence of the bulleted paragraph should begin:

The asynchronous line controller is the interface to the primary terminal....

Add the following note after the description of the Real-time clock in the fifth paragraph:

NOTE *Operation of the Real-time clock at ac line frequency requires the presence of the line frequency clock generator card in the power module.*

Page 4-20

Under page heading, "The Microcode Controller Chips",
The first sentence of the third paragraph should read:

As mentioned earlier in this chapter, the Model 20 and 30 SPU card contains two or three XMCs.

Page 4-25

Under page heading, "Extended Memory Cycles",
The reference in the last sentence (in parentheses) should be to Chapter 5.

Page 4-42

Table 4-4

Change Table ?? in the footnotes to read:

external signals table

full-duplex asynchronous communications interface, and an SPU status register;

A memory allocation and protection (MAP) unit;

Parity checking logic;

A virtual console residing in 1 Kbyte of read-only memory (ROM), with 2 Kbytes of static read/write memory (RAM).

The SPU receives its power and communicates with other printed circuit cards (memory or I/O controllers) in the CPU logic module (CLM) card cage using its B connector, which plugs into the CLM backpanel. It uses its A connector for full-duplex communication with a serial, asynchronous communacitions device. In the Model 30, the SPU also communicates with the hardware floating point card using its C connector.

Major Elements

This section describes the functional characteristics of each major element on the SPU card.

CPU and XMCs

The CPU in the Model 20 and Model 30 computers is a microprogrammmed processor that incorporates the full ECLIPSE 16-bit architecture, including four fixed point and four floating point accumulators, a floating point status register, and a MAP status register. The CPU implements the ECLIPSE instruction set by executing a series of microinstructions. These microinstructions control the flow of data through the CPU internal registers, in the arithmetic logic units, and along the data paths of the SPU.

A kernel of these microinstructions for the ECLIPSE instruction set reside in the mE670 CPU IC, while the microinstructions for the remaining instructions reside in three supporting XMCs. The XMCs communicate with the mE670 via a dedicated, 8-bit time-multiplexed bus.

The three XMCs on the Model 20 SPU card contain microcode for all floating point instructions plus any ECLIPSE instructions not in the mE670 kernal.

The three XMCs on the Model 30 SPU card contain microcode for non-kernal ECLIPSE instructions and the commerical instructions, in addition to control microcode for the hardware floating point card.

The mE670 CPU uses the 16-bit wide memory address/data bus to communicate with memory elements. During operations in the memory mapped mode, the memory address width expands from 15 to 20 address bits.

The CPU communicates with its support elements, such as the multidevice SIO chip and MAP unit, via the 16-bit wide CPU bus. A microI/O bus interface is connected to this local bus and enables the CPU to communicate with standard I/O device controllers.

Model 20 and 30 memory access time is 0.500 microseconds. Instruction execution times range from 0.50 to 34.00 microseconds for fixed-point operations. Firmware floating-point operations in the Model 20 are controlled by floating point XMCs and require from 1.00 to about 900 microseconds. Hardware floating point operations in the Model 30 require from 1.50 to about

50 microseconds. Maximum I/O interrupt latency of Model 20 computers is 110 microseconds; maximum I/O interrupt latency of Model 30 computers is 350 microseconds.[1] Maximum data channel latency is 6 microseconds. A complete listing of instruction execution times for the CPUs appears in "Programming the CPU" in Chapter 1.

[1] *These figures assume the absence of any data channel operations.*

The Model 20 and Model 30 systems can contain up to 2 Mbyte of memory with validity, I/O and write protection, and parity checking. In addition, the CPU

Supports direct memory access (DMA) via data channel,

Has two distinct program interrupt facilities: maskable and non-maskable,

Supports 16 levels of programmed interrupt priorities.

The CPU's data channel facility allows devices to transfer data to and from fast memory over the microI/O bus at speeds of 337 kilobytes per second for output and 267 kilobytes per second for input.

The CPU has two interrupt facilities: standard and nonmaskable. The *standard interrupt facility* services four types of interrupt requests based on the following order of priorities: (1) powerfail, (2) stack overflow, (3) programmed interval timer, (4) real-time clock, (5) TTI (SPU asynchronous interface receiver), (6) TTO (transmitter), and (7) external I/O. The SPU reserves the *nonmaskable interrupt facility* for user entry into the virtual console.

The 16 levels of programmed interrupt priority are associated with the standard interrupt facility. They are established by a 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.

Memory Allocation and Protection

The memory allocation and protection (MAP) feature performs logical-to-physical address translation, accessing a maximum of 2 Mbytes of physical memory. Maximum memory size of 2 Mbytes is obtained by using four 512-Kbyte cards in an Model 20 system. Maximium memory size in a Model 30 system is 1.5 Mbytes and is obtained using three 512 Kbyte memory cards. The MAP unit stores five address maps that can be used by the system. There are four user address maps and one data channel map.

In addition to translating addresses, the MAP feature performs the following functions:

Validity protection

Write protection

I/O protection

Indirection protection

The MAP feature also allows the implementation of the *Load Effective Address* instruction and the emulator trap feature. These and the above protection features are discussed more fully under "SPU Instruction Set" later in this chapter.

Parity Checking

The Model 20 and 30 CPUs append a parity bit to each byte of data written to memory and checks this bit for each byte read from memory. Detection of an incorrect parity bit causes an interrupt request if such requests have been enabled.

Multidevice Section

The heart of the multidevice section is the mE676 system I/O (SIO) integrated circuit chip. This chip contains three internal I/O devices, along with a powerfail monitor and SPU status register. The internal I/O devices are (1) an asynchronous line controller, (2) a real-time clock, and (3) a programmable interval timer. These I/O devices are programmed using I/O format instructions as though they were external I/O devices (refer to "Programming Standard I/O Devices" in Chapter 2). The powerfail monitor and SPU status register are also programmed using I/O format instructions with the device code specifying the CPU (refer to "System Management Instructions" in Chapter 1.

- The Asynchronous line controller. Interface to the primary terminal of the Model 20 and 30 system. It can transmit and receive serial asynchronous information at switch-selectable rates of 50 to 38,400 baud. The controllers line characteristic is switch selectable to be either EIA RS-232 or 20-milliamp current-loop.

NOTE *The Model 20 and Model 30 system processor unit asynchronous communications interface receives and transmits eight-bit data characters without parity. If the terminal device connected to this interface port operates with a data character length of seven bits, it should be configured to operate with* mark parity. *If the terminal device connected to this interface 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.

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.

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 1 microseconds to 65.536 seconds in increments of 1 microseconds. The clock rate of the PIT is switch-selectable (see "Programmable Interval Timer" in Chapter 2) 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 *powerfail monitor* tracks the state of a power status signal supplied by the power supply, and initiates an interrupt request of the CPU whenever there is a change in this signal. The SIO controller returns device code 0 to the CPU when the CPU acknowledges the interrupt.

The *CPU status register* reports on the following conditions:

Occurrence of a powerfail interrupt

Enabling of interrupts

Occurrence of a Break Key interrupt

Power-up condition

Decoding of a HALT instruction

State of the Halt Dispatch bit in the SIO switch register

Occurrence of an interrupt request

Occurrence of a virtual console trap (in virtual console single-step mode)

Virtual Console

This resident firmware, with its 2 Kbytes of read/write memory, allows the user whose terminal is connected to the SPU asynchronous interface to inspect and modify the system's state. It also aids program debugging. The virtual console allows users to (1) stop, start, and continue program execution; (2) examine and alter CPU registers and memory locations; and (3) initiate program load sequences. Chapter 3 discusses the virtual console in full detail.

Power Monitoring and Initialization

The power supply generates two status signals that are monitored by the SPU. One indicates that all voltages are within specified limits; the other indicates that a power loss is imminent.

After the SPU is notified that all voltages are within specified limits on power up, it enters the Halt state with interrupts enabled. When this occurs, the SPU can be started by issuing a virtual console program load command (refer to the "Power-up Response" section in Chapter 1 and "Virtual Console", Chapter 3.)

An imminent power loss generates a program interrupt. When the power loss is first detected, the SPU has a full two milliseconds of operating time before any voltages fall below specifications.

Installation and Tailoring

The Model 20 or 30 SPU must reside in Slot 1 of the CPU logic module (CLM) Its power requirements are listed in Table 4-1.

Table 4-1 SPU power requirements

Supply Voltage (Volts DC)	Current draw (Amps)	Power dissipation (Watts)	Pin numbers B Connector
+5 (± 0.5)	5.50	30.25	57, 59, 60
−5 (± 0.5)	0.03	0.15	58
+12 (± 0.5)	0.10	1.2	55, 56
−12 (± 0.5)	0.10	1.2	39
Ground			3, 11, 14, 36, 53, 54

Tailoring

The SPU card contains eight jumpers and two dual-in-line-package (DIP) switches that select SPU operating characteristics. Refer to *Installation Guide for your system for their functions, location, and tailoring.*

Interfacing

The Model 20 and Model 30 SPU communicates with other parts of the system via its edge connectors. The SPU receives power and communicates with memory and I/O cards in the card cages through the B connector. It also communicates with a system console through the A connector. In addition, the Model 30 CPU communicates with the hardware floating-point card through the C connector. The SPU does not use the D connector. Figure 4-2 shows the connector positions of the SPU card.

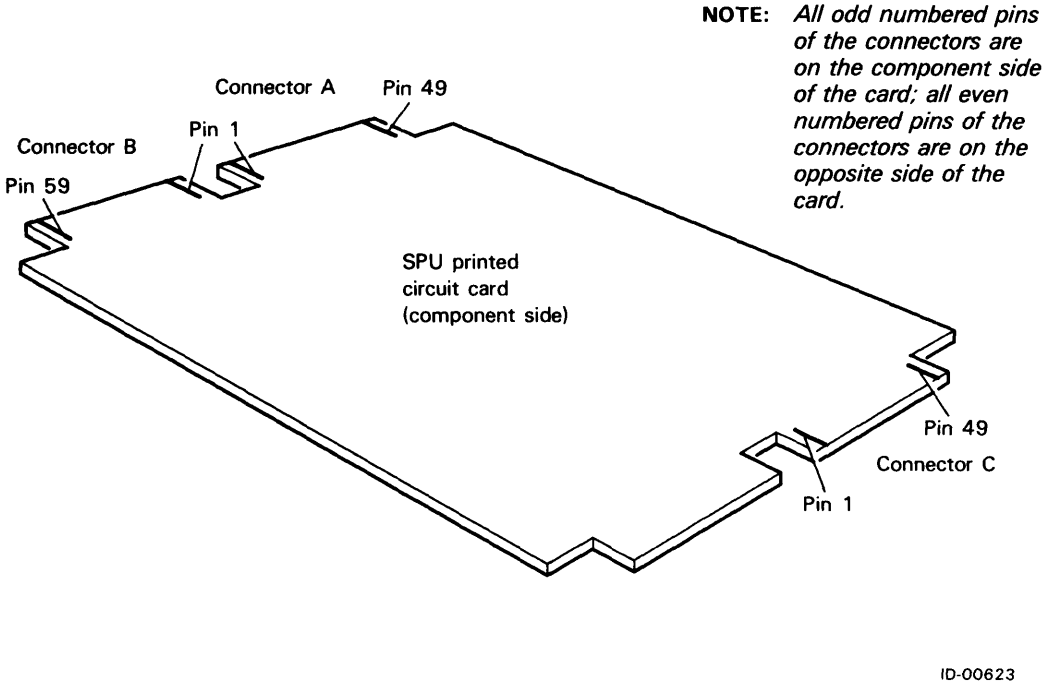


Figure 4-2 SPU card connector positions

The 50-pin A connector carries control and data signals between the SPU and the system console. Figure 4-3 shows the pin assignments.

Even	Signal Names		Odd
2	CTS	TTIN	1
4		-5V	3
6			5
8		GND	7
10	DTR	+5V	9
12			11
14			13
16			15
18			17
20			19
22		TTOUT	21
24		+12V	23
26			25
28			27
30			29
32			31
34			33
36			35
38			37
40			39
42			41
44			43
46			45
48			47
50			49

Note *Blank pins are not used.*

DG-08909

Figure 4-3 Pin assignments, A connector

The 60 pins of the B connector are TTL-compatible unless otherwise marked. Figure 4-4 shows the pin assignments.

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	BBOOT	43
42	DATA2	DATA10	41
40	XMA3	−12V	39
38	DATA3	DATA11	37
36	GND	RTC	35
34	DATA4	DATA12	33
32	DATA5	DATA13	31
30	WH/PARH	WL/PARL	29
28	DATA6	DATA14	27
26	BUSMEMCYC	XMA4	25
24	DATA7	DATA15	23
22	POWEROK	DCHPOUT	21
20	PF	INTPOUT	19
18	BUS READY	HALT	17
16	BI/OCLOCK*	BI/OCLOCK*	15
14	GND	BI/ODATA2*	13
12	BI/ODATA2	GND	11
10	CONSOLELOCK	EXT DCHR	9
8	EXTINT	PWRFAIL	7
6	CLEAR	BI/ODATA1*	5
4	BI/ODATA1*	GND	3
2	BM CLOCK*	BMCLOCK*	1

**Not TTL compatible*

ID-00624

Figure 4-4 Pin assignments, B connector

The 50-pin C connector carries control signals between the Model 1 SPU and its hardware floating point card. Figure 4-5 shows the pin assignments.

Even	Signal Names		Odd
2	GND	GND	1
4	$\overline{\text{TEST}}$		3
6	GND	GND	5
8			7
10			9
12			11
14			13
16			15
18			17
20	GND	QPIPE	19
22	+5V	+5V	21
24		GND	23
26		QUACK	25
28		$\overline{\text{QSKIP}}$	27
30			29
32		QREQ	31
34		$\overline{\text{QFETCH}}$	33
36			35
38			37
40			39
42			41
44		$\overline{\text{BMCR}}$	43
46		$\overline{\text{BLOCK}}$	45
48		$\overline{\text{BMCGRANT}}$	47
50		GND	49

Note *Blank pins are not used.*

DG-08297

Figure 4-5 Pin assignments, C connector

Device Cables

A number of cables are available to connect the Model 20 or 30 system to a serial, asynchronous terminal with either an EIA RS-232C interface or a 20-mA current-loop interface. (Refer to the *Installation Guide* for your system.)

Theory of Operation

This section launches its discussion of operating theory with overviews of system architecture, system timing, and the system processing unit. A brief list of references follows. Then the three major areas of SPU operation — the CPU, CPU support, and microI/O bus areas — are examined in depth. A summary of internal signals concludes the section.

System Architecture

As shown in Figure 4-6, the Model 20 and 30 systems are organized around two system busses, the memory bus and the microI/O bus. In addition, the SIO TTI/TTO interface communicates with the system console over the console bus, and a fourth bus carries control signals between the SPU and the hardware floating point card.

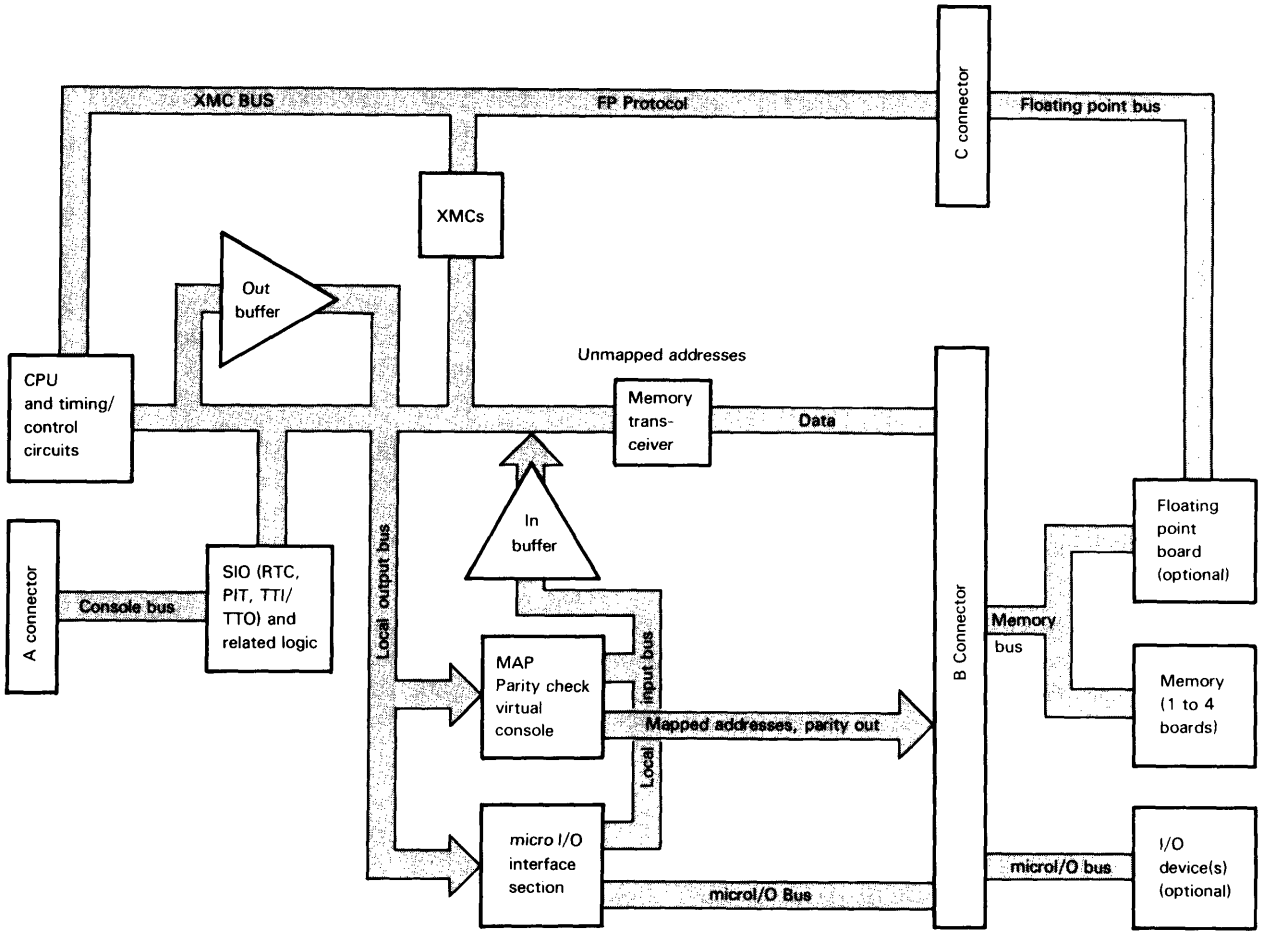


Figure 4-6 Model 20 and Model 30 system architecture

The memory bus transfers data and addresses between the SPU and system memory. The data and addresses are multiplexed on the memory bus, which consists of 16 bidirectional data/address lines ($\overline{DATA<0\text{-}15>}$), 5 unidirectional extended address lines ($\overline{XMA<0\text{-}4>}$), and 8 timing and control lines.

The 16-line microI/O bus transfers data and I/O instructions or status between the SPU and I/O device controllers. These transfers occur on two bidirectional serial lines (BI/ODATA1 and BI/ODATA2), and are synchronized by a differentially-driven clock signal (BI/OCLOCK). Two interrupt lines — programmed I/O ($\overline{EXTINT}$) and data channel ($\overline{EXT\ DCHR}$) — allow I/O interfaces to request processor time. A system reset line ($\overline{CLEAR}$), two device priority lines (INTPOUT) and (DCHPOUT), a differentially driven master clock signal (BMCLOCK), and three ground lines comprise the remainder of the I/O bus.

The hardware floating point unit communicates with the SPU and system memory over the memory bus. This unit exchanges protocol and status signals with the SPU over the floating point bus, which uses the C connector.

System Timing

The Model 20 and Model 30 participate in two kinds of data exchange: memory and I/O. Addresses and data are multiplexed on the CPU and memory busses. Thus, data exchanges on these busses take place in two phases: an address phase and a data phase. An address phase/data phase pair corresponds to one CPU microcycle with a duration of 500 nanoseconds. A CPU microcycle is also called a T period.

Located in the CPU section, the I/O decode circuitry detects instructions coded in I/O format (instruction formats starting 011 . . .). Once detected, these instructions are referred to the internal devices — the SIO chip, the MAP unit, the parity check unit — or to the I/O interface section. The I/O interface produces I/O timing signals for instructions and data intended for I/O devices connected to the microI/O bus. The timing signals are based on a 125-nanosecond I/O clock cycle.

The "address" for an instruction directed to an element of the SPU itself (that is, a device contained in the multifunction controller or in the CPU support section) is an encoded version of the actual instruction. This address is issued by the CPU and decoded by the I/O decode circuitry during the CPU address phase. Then the data are transferred during the data phase.

Devices that require more than 500 nanoseconds to send or receive data can extend the data phase of the cycle for as long as necessary. To do this, an external device holds the memory bus control signal BUS READY low. The CPU will not initiate another system cycle until BUS READY is asserted high.

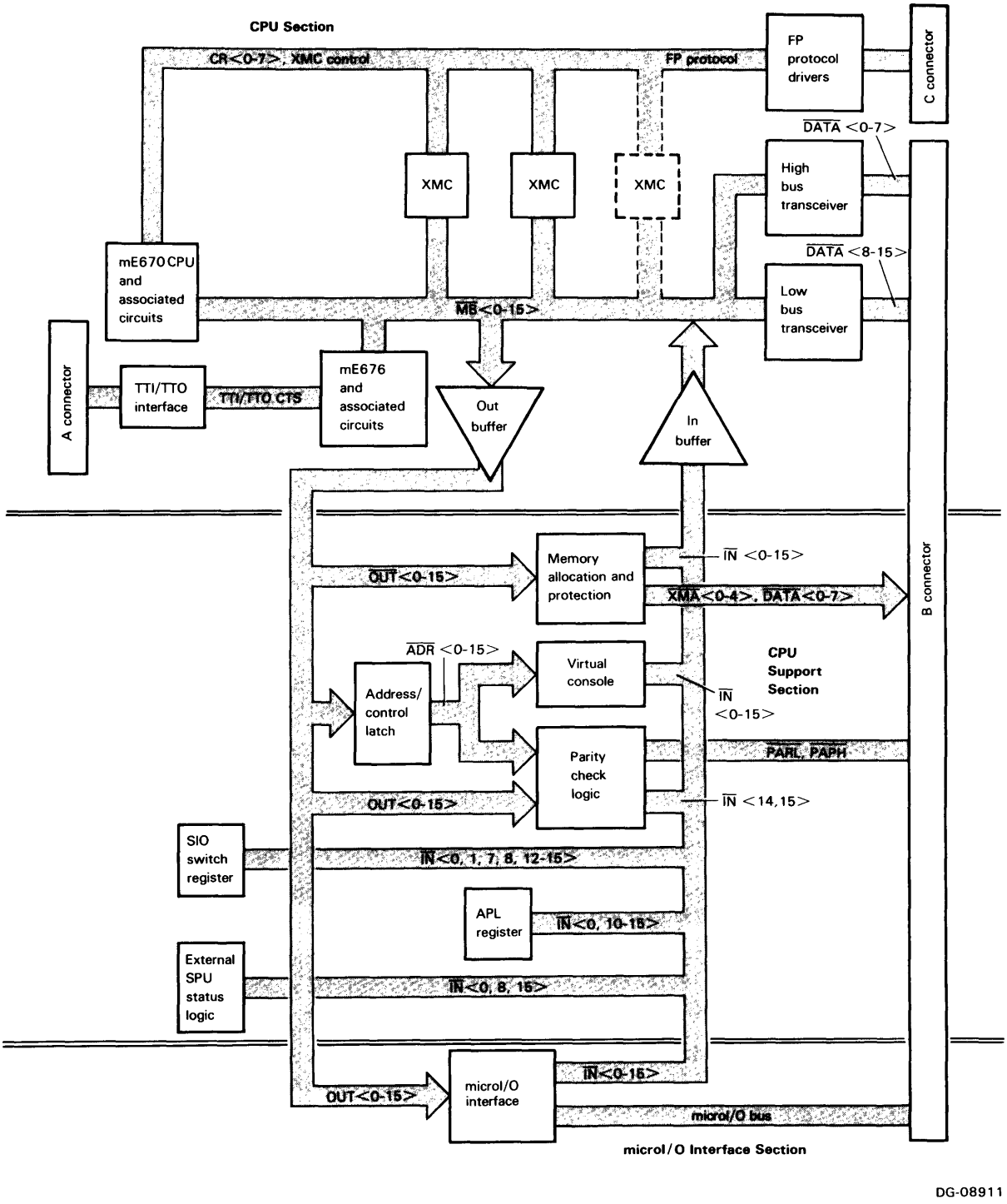


Figure 4-7 SPU block diagram

The System Processing Unit

From a functional point of view, the SPU consists of three major sections:

- The CPU section

- The CPU support section
- The microI/O bus interface section

As Figure 4-8 shows, these sections are organized around four internal busses:

16-bit *bidirectional* CPU bus ($\overline{MB<0\text{-}15>}$)

16-bit local *output* bus ($\overline{OUT<0\text{-}15>}$)

16-bit local *input* bus ($\overline{IN<0\text{-}15>}$)

16-bit *address* bus ($\overline{ADR<0\text{-}15>}$)

The CPU section sends and receives addresses and data over the CPU bus to and from the local busses and memory transceivers. In response to control and timing signals from the CPU control section, the memory transceivers convey addresses and data to and from the system memory.

- *CPU bus*. Is a multimaster bus; that is, any system component capable of asserting the signals necessary to control the bus cycle specification can be a bus master. In particular, the system state machine in the microI/O bus interface section becomes the bus master during data channel operations, as will be described.
- *Local bus transceivers*. Convey addresses and data to and from the CPU support section (see Figure 4-7). The elements contained in this section perform functions that support CPU operations. The MAP unit supports the memory allocation and protection function of the CPU. The parity check supports the CPU error reporting function. And the virtual console program contains object code in read-only memory that runs system self-tests and allows the user to access SPU and memory locations directly. For more information about these elements and the functions they perform, see the subsection entitled "CPU Support Elements."
- *I/O interface section*. Contains logic and signal processing circuitry that convert data from the SPU format (16-bit parallel) to microI/O bus format (2-line serial). This section also transfers the data to and from the I/O bus with the appropriate timing and handles data channel operations, taking over the system bus and asserting bus cycle signals as needed.
- *Address bus*. Carries internal addresses from the address latch to the virtual console and parity checking circuitry.

The SPU incorporates integrated circuits from the microNOVA microcomputer chip set. Refer to the *microNOVA Integrated Circuits Data Manual* (DGC No. 014-000074) for more information about these.

The sections to come present material relative to Logic schematic DGC 001-003078. All signals presented in the figures or text of these sections are listed in Table 4-3 and Table 4-4.

The CPU Section

The CPU section, shown in Figure 4-8, contains the following features.

An mE670 16-bit central processing unit

Two or three external microcode controller chips (XMCs)

Floating point protocol drivers for the Model 20 and Model 30 hardware floating point card

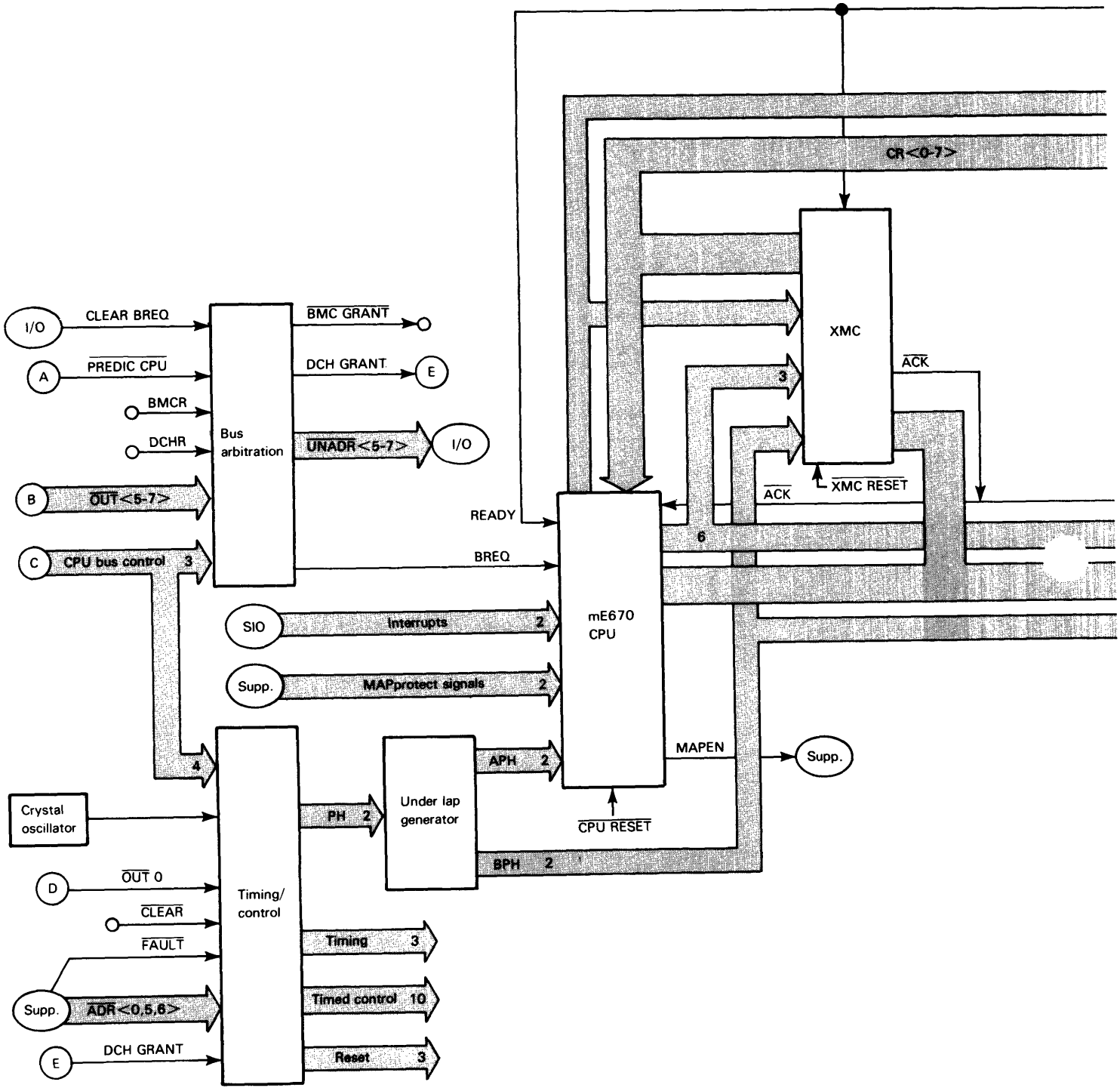
Bus arbitration logic

Timing/control circuitry

An I/O decoder

Two octal memory bus transceivers

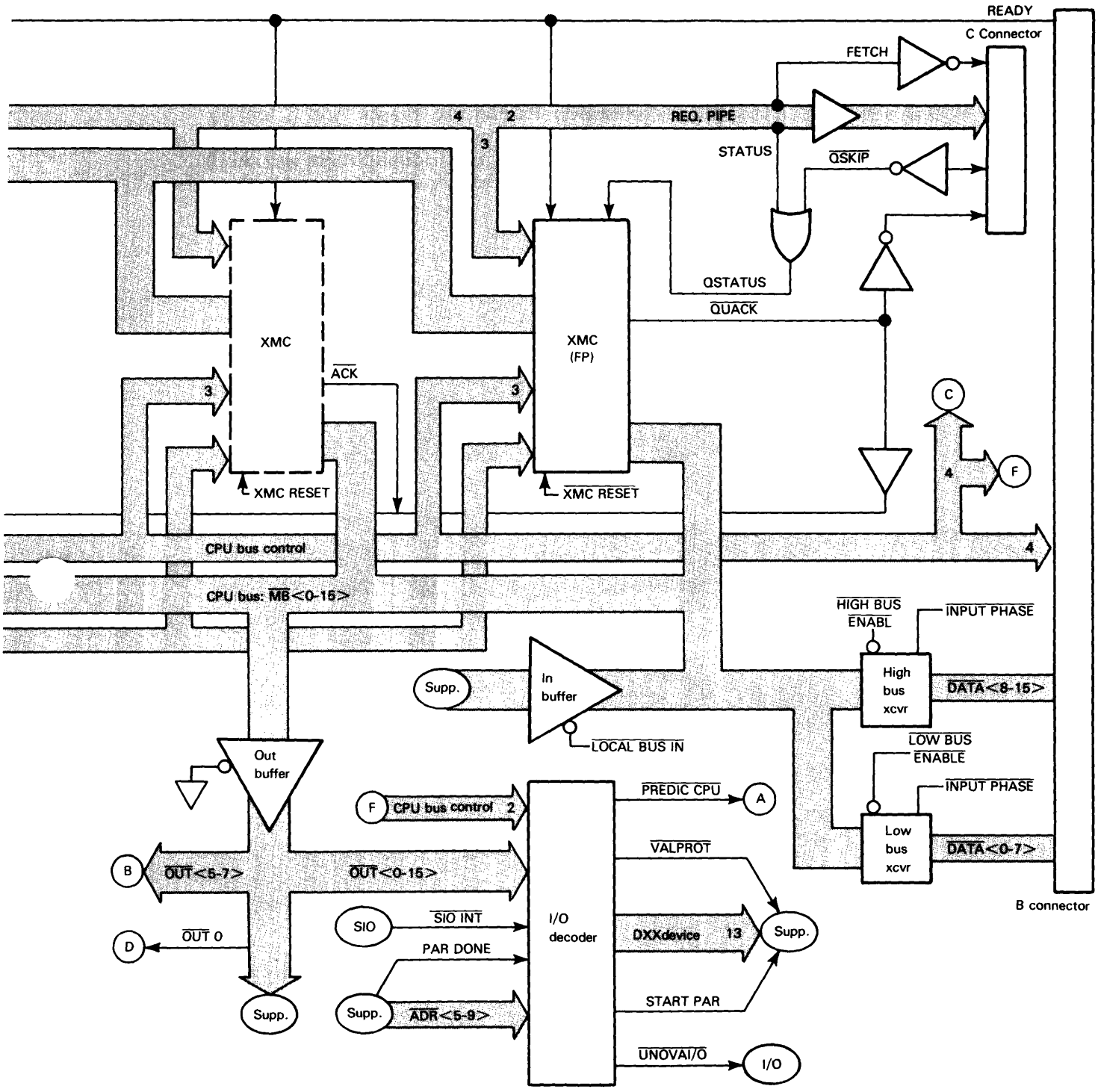
Two local bus buffers



DG-08912

DG-08912

Figure 4-8 The CPU section


READY
C Connector
FETCH
REQ. PIPE
STATUS
QSKIP
QSTATUS
QUACK
XMC
XMC (FP)
ACK
XMC RESET
XMC RESET
CPU bus control
CPU bus: MB<0-15>
C
F
HIGH BUS ENABL
INPUT PHASE
High bus xcvr
DATA<8-15>
Supp.
In buffer
LOCAL BUS IN
LOW BUS ENABLE
INPUT PHASE
Low bus xcvr
DATA<0-7>
B connector
Out buffer
F CPU bus control 2
PREDIC CPU
A
B
OUT<5-7>
OUT<0-15>
VALPROT
SIO
SIO INT
I/O decoder
DXXdevice 13
Supp.
D
OUT 0
PAR DONE
Supp.
Supp.
ADR<5-9>
START PAR
UNOVAI/O
I/O

The mE670 CPU

The mE670 CPU is a single, large-scale integrated circuit (IC) device that executes the kernel of the instruction set described earlier. It supports up to 2 megabytes of memory while implementing indirect protection, I/O protection, the LEF instruction, and the emulator trap capability. The CPU executes 16-bit register-to-register operations in a single 500-nanosecond microcycle and external bus-to-register moves in two microcycles.

The simplified block diagram in Figure 4-9 shows the major, functional components of the CPU: the CPU bus transceiver, four internal busses (A, B, C, and M), an autonomous prefetch control unit, a program counter (PC) pipeline, an instruction register (IR) pipeline, the register file, the arithmetic/logic unit (ALU) and shifter, the MAP status register, and the microprogrammed control logic.

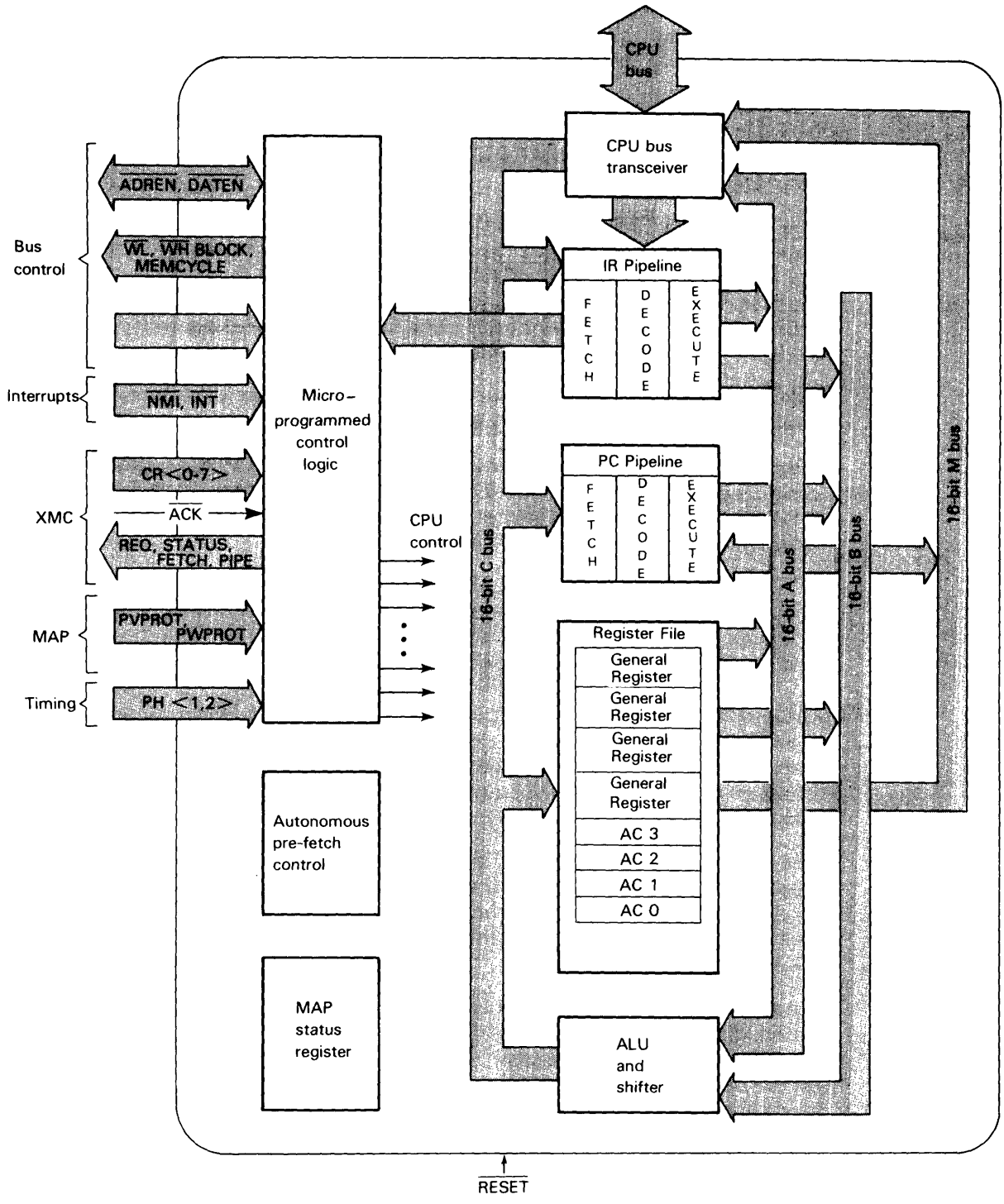


Figure 4-9 Internal CPU block diagram

The *CPU bus transceiver* forms the interface between the CPU's internal busses and the time-multiplexed external CPU bus. The transceiver drives addresses onto the external CPU bus during the first half of a microcycle, that is, during the CPU address phase. During the second half of the microcycle (the data phase), data are read from or placed on the external CPU bus.

The *four busses* internal to the CPU enable rapid and flexible transfer of data between the components that they connect. The A and B busses, for example, allow the simultaneous transfer of two source operands from the register file to the ALU and shifter.

The *autonomous prefetch control unit* causes two words from the instruction stream, beyond the currently executing instruction, to be fetched and placed in the IR pipeline; the corresponding instruction sequence is preserved in the PC pipeline. In cases where instructions alter the program flow or make memory references to the locations of prefetched instructions, the contents of the pipelines are invalidated, or *flushed*.

The *register file* includes the four program-accessible accumulators and four general registers accessible only to the microcode.

The *ALU and shifter* perform all arithmetic and logic operations under the control of the microprogrammed control logic. The microprogram for the control logic is contained partly in microcode that is resident on the CPU and partly on external microcontroller chips (XMCs).

In addition to supervising the manipulation of data, the CPU-resident, *microprogrammed control logic* produces CPU and bus control signals; responds to interrupt requests; handles the microcode transfer protocol; and sends, receives, and processes MAP control signals.

The Microcode Controller Chips

Each external microcontroller chip (XMC) contains a decode programmed logic array (PLA) that can decode up to 64 macroinstructions. When the PLA decodes an instruction contained on the XMC, it transfers control to the XMC's microcode controlling logic. The XMC then sends microcode for the macroinstruction to the CPU.

The CPU accepts 16-bit external microcode instructions from the microcontroller chips (XMCs) via a dedicated, 8-bit time-multiplexed bus (CR<0-7>). In addition, the CPU and XMCs use five control signals for microcode transfer protocol. The protocol allows conditional branching that depends on the result of CPU microcode execution.

As mentioned earlier in this chapter, the Model 20 and 30 card contains three XMCs. The Model 20 XMCs contain microcode for the floating point instruction set, along with an instruction set that supplements the instruction kernel residing on the mE670. The Model 30 XMCs contain the supplementary instructions, the commerical instruction set, and the microcode that decodes floating point instructions and provides command and status signals for the hardware floating point card.

Each XMC monitors the system bus to maintain a duplicate of the CPU's macroinstruction pipeline. This allows the CPU and multiple XMCs to decode macroinstructions simultaneously. As a result, only one microcycle is needed to obtain the microcode for an instruction not contained on the mE670.

Bus Arbitration

The bus arbitration logic allocates control of the memory bus between the CPU and any data channel controller. It consists of a 4-bit

registered PAL device, clocked by $\overline{\text{PHASE}}$. It arbitrates among the presence or absence of the following conditions:

The current phase is an address phase.

The CPU is not issuing BLOCK to prevent interruption of its operation.

A device is not asserting $\overline{\text{READY}}$ to extend the length of a memory cycle.

A data channel request has been received.

An *I/O Reset* instruction has been decoded.

The microI/O bus interface state machine has signalled that it has finished a data channel operation.

Based on the current combination of the conditions just described, the bus arbitration logic issues one or more of the following signals.

DCH GRANT	to the MAP section and the microI/O bus interface state machine
BREQ	to the CPU

The bus arbitration PAL is also used to translate bits $\overline{\text{OUT}<5\text{-}7>}$ into bits $\overline{\text{UNADR}<5\text{-}7>}$ for an I/O Reset instruction.

Timing/Control

The timing/control circuitry consists of a state machine (32 x 8 PROM and octal flip-flop) and a programmed array logic (PAL) timing decoder. Associated with these are a 16-MHz crystal-controlled oscillator and an underlap generator.

Figure 4-10 shows the signals produced by the timing state machine. Also shown in Figure 4-9 are the 16-MHz reference signal and the timing input signals for the CPU and XMCs.

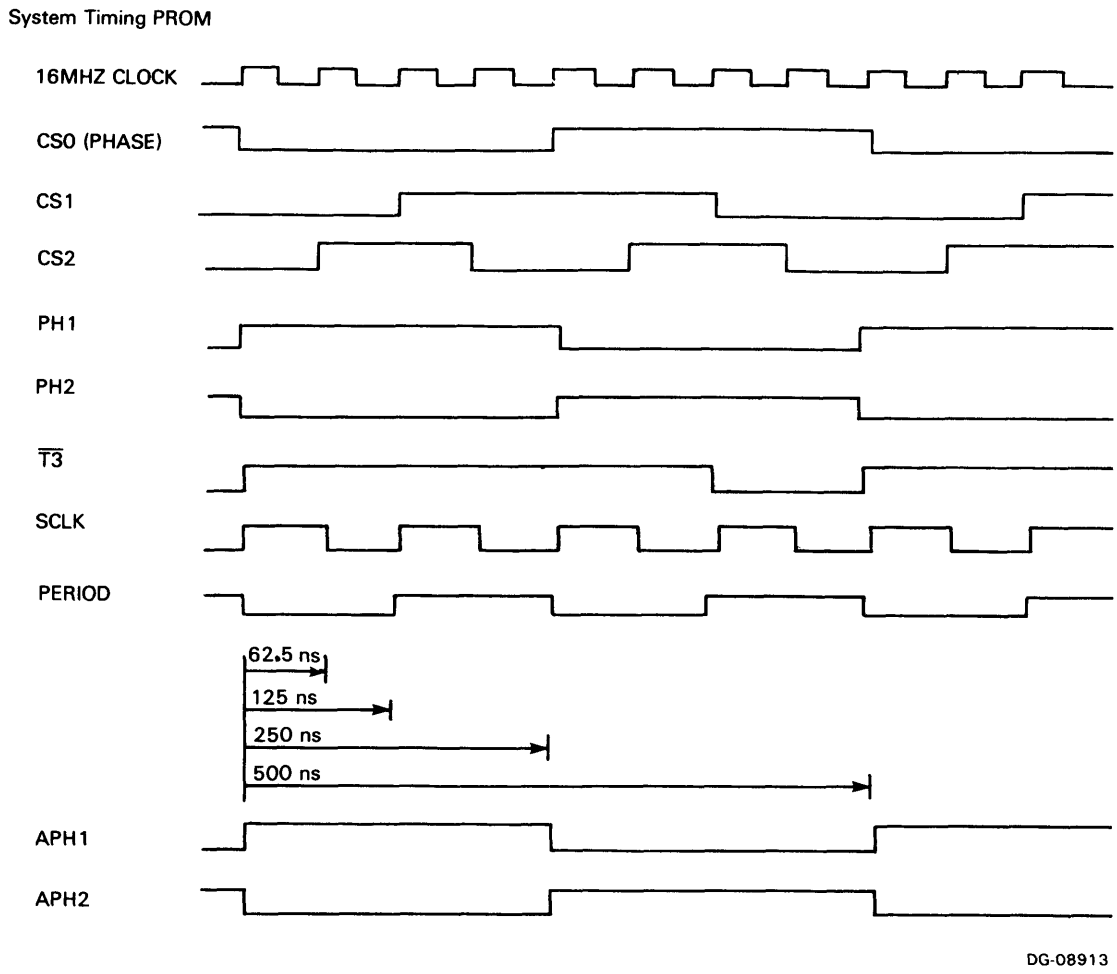


Figure 4-10 Timing finite state machine signals

The timing signals and their meanings are as follows.

CS<0-2>: state machine outputs from which system timing signals are derived. CS0 is also named PHASE.

PH<1,2>: inputs to the underlap generator; they generate APH<1,2>, the two-phase, nonoverlapping clock signals for the CPU and XMCs.

$\overline{T3}$: a timing signal for reading from and writing to the scratchpad memory in the virtual console.

SCLK: the source of ($\overline{\text{SYSTEM CLOCK}}$), which clocks the system state machine in the microI/O bus interface and is the microI/O bus master clock.

PERIOD and PHASE: sources of state information for the system state machine in the microI/O bus interface.

The state machine outputs are always present after powerup.

Figure 4-11 shows the inputs to and outputs from the PAL timing decoder; also represented are the bus control signals $\overline{\text{ADREN}}$ and $\overline{\text{DATEN}}$, which are issue from the CPU. The timing signals from both sources, along with their meanings, are as follows.

$\overline{\text{HIGH PROM OUT}}$: the enable signal for reading the virtual console PROMs.

$\overline{\text{LOCAL BUS IN}}$: the enable signal for the local input buffer.

$\overline{\text{INPUT PHASE}}$: the direction control signal for the CPU bus transceivers.

BUS TERM: source of the enabling signals for the CPU bus transceivers, $\overline{\text{HIGH BUS ENABL}}$ and $\overline{\text{LOW BUS ENABLE}}$. This signal is asserted for all read operations unless a fault occurs.

$\overline{\text{DATEN}}$: the enable signal for data drivers; asserted for all read operations unless a fault occurs. This signal remains asserted during extended memory operations and prevents the assertion of $\overline{\text{ADREN}}$. It is produced by the circuit element that is currently the source of the data.

RAM ENABLE: source of the chip select signal $\overline{\text{RAM CS}}$ for the virtual console scratchpad memory.

$\overline{\text{ADREN}}$: the enable signal for address drivers. It is asserted during the address phase by the current bus master, which will be the CPU or the microI/O bus interface state machine.

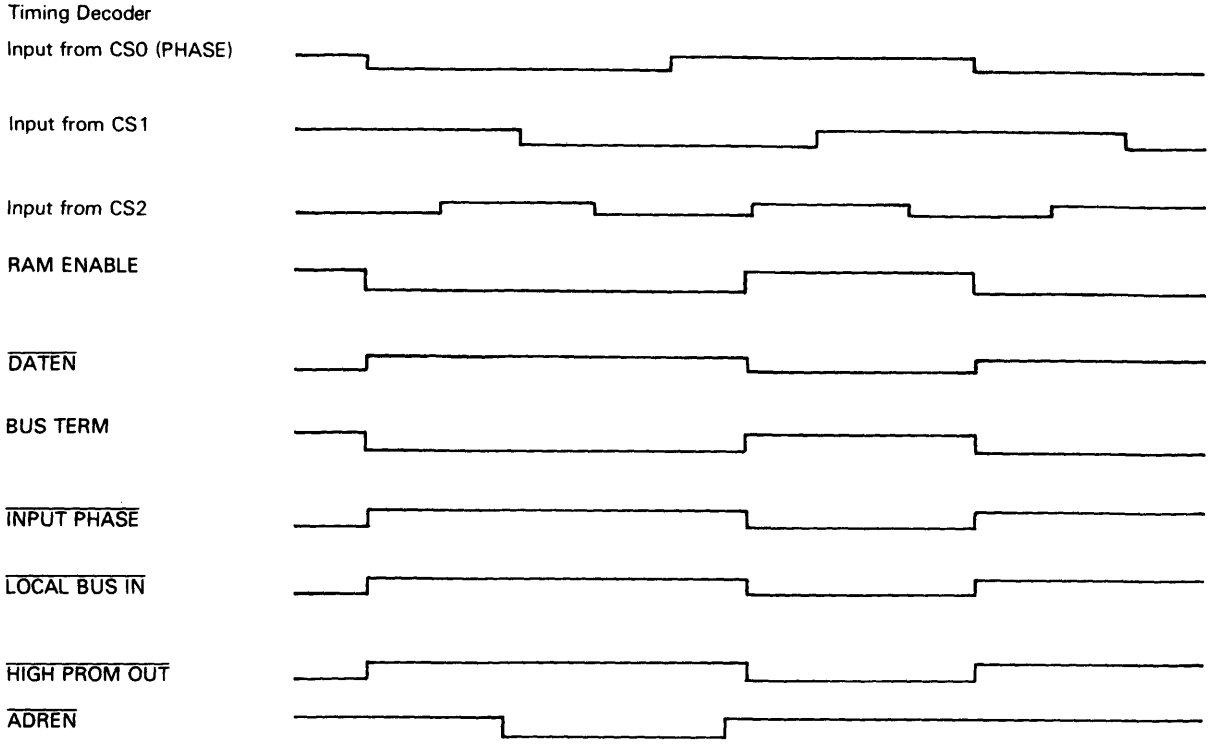


Figure 4-11 Timing decoder signals

The signals already described, in logical combination with system control signals such as MAPEN and $\overline{\text{WL}}$, produce other timing control signals. These derived signals and their meanings are as follows:

$\overline{\text{RAM WRITE}}$: a write-enable signal for the virtual console scratchpad memory. It is asserted at the same time as
$\overline{\text{T3}}$, when the operation is a write.

MAP ADDRESS OUT: an enabling signal for high-order memory address bits produced by the MAP section. It is asserted during the address phase of a memory operation when the MAP is enabled.

ADDRESS LATCH: a signal whose falling edge latches the data on the local output bus, along with several system control signals, into the address latch in the CPU support section. It is held low during extended memory operations.

The following signals convey SPU state information to the decoder: from the CPU — ADREN, MEMCYCLE, WRITE, MAPEN; from the local output bus — OUT0; from the microI/O bus — CLEAR; from the MAP unit — FAULT; from the address latch — ADR<0,5,6>; and from the bus arbitration logic — DCH GRANT.

The timing/control logic also produces the three system reset signals: CPU RESET, XMC RESET, and SIO RESET.

I/O Decoder

The I/O decoder consists of two PAL devices, two 1-of-8 decoders, and associated latches and gates. Its function is to decode I/O format instructions during the address phase and route them to the internal devices they were intended for. In the case of an I/O format instruction not intended for an internal device, the decoder asserts UNOVA I/O, which activates the microI/O bus interface state machine.

The I/O decoder also monitors the output of the MAP unit during the address portion of memory cycles, asserting VALPROT whenever it detects a write-protected page assigned to the last physical page (physical address bits all 1).

Memory Bus Transceivers

The two octal memory bus transceivers, diagrammed in Figure 4-12 and Figure 4-13 are bidirectional tri-state output devices. As described earlier, the outputs of these transceivers are enabled by HIGH BUS ENABL and LOW BUS ENABLE. The direction of transfer is determined by the timing control signal INPUT PHASE: low into the CPU, high out to the bus. LOW BUS ENABLE occurs during the address and data phases of a memory reference operation; HIGH BUS ENABL always occurs during the data phase but during the address phase only when no mapping is involved.

Memory Read/Write Operations

Figure 4-12 illustrates the timing of memory read/write operations. The bus master asserts ADREN during the address phase. (The bus master is either the CPU, for normal memory operations, or the microI/O bus interface state machine, for data channel operations.) It then places the address on the memory bus. During the same phase the bus master also asserts WH, WL, or both — corresponding to the memory operations write high byte, write low byte, or write word. If no write signal is asserted, the operation is a memory read.

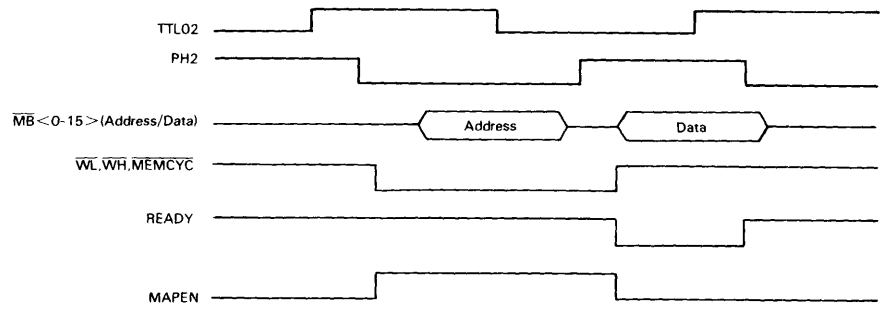


Figure 4-12 Memory bus control signals

Bus control signals derived from BUS TERM enable the memory bus transceivers during the address phase. The address is clocked into its destination when the low-to-high transition of PHASE occurs.

If a MAP fault occurs during a mapped memory operation, the MAP logic asserts FAULT. This signal prevents the timing decoder PAL from asserting BUS TERM and inhibits the WH and WL signals as well. FAULT is latched by the low-to-high transition of PHASE and remains latched until PHASE goes low again, provided the fault condition is removed.

During the data phase the bus transceivers are again enabled; when the rising edge of PHASE occurs, the data is clocked into its destination. However, no data is written into memory when the latched FAULT signal, FAULT CYCLE, is present.

Extended Memory Cycles

A device can request additional time to complete a memory cycle by pulling the READY line low. In response, the CPU extends the data phase as long as necessary. The CPU samples the state of READY at the end of each data phase. If READY is low, the CPU extends the memory operation for another complete 500-nanosecond cycle. If READY is high at the end of this second cycle, the CPU begins a new memory operation; otherwise, it extends the current operation for yet another cycle. If the CPU is the data source, it makes the data available for the entire period through which the memory cycle is extended; the memory does not clock the data in, however, until the end of the memory cycle in which READY is not low. If the memory is the data source, it places the data on the bus during the first data phase after the refresh operation that necessitated the extended cycle.

Figure 4-13 illustrates the extension of a memory cycle. The diagram shows that memory has pulled the READY line low because it received the address for a CPU memory operation during a refresh operation — one that was pended during a data channel operation. (Refer to Chapter 3 for a full description of these operations.)

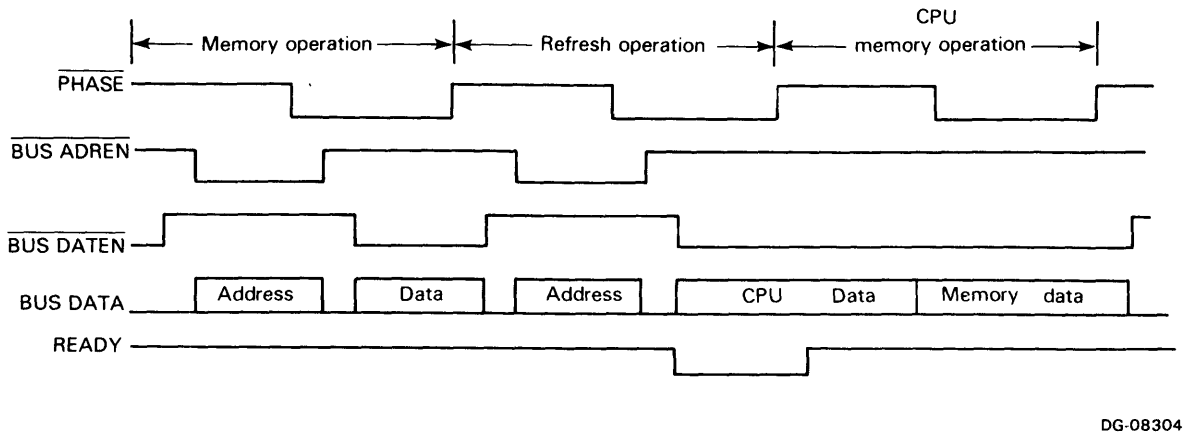


Figure 4-13 Extended memory cycle

Data Channel Operations Data channel operations are managed by the microI/O bus interface, described in a later section.

Local Bus Buffers The 16-bit local bus buffers are bidirectional tri-state output devices. The out buffer is always enabled. The in buffer is controlled by $\overline{\text{LOCAL BUS IN}}$.

CPU Support Elements

The CPU support elements, shown earlier in Figure 4-7, include the following.

The SIO chip and TTI/TTO interface

The address latch

The memory allocation and protection unit (MAP)

The parity checking unit

The virtual console

The three registers: SIO, APL, and CPU status

SIO Chip The SIO chip contains five devices

a powerfail monitor

a programmable interval timer (device 43)

a real-time clock (device 14)

an asynchronous interface (TTI — device 10, TTO — device 11)

the CPU status register

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:

when PF goes from negative to positive, indicating that AC power has failed, or

when 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 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, non-maskable 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 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 user program can read interrupt latency by interrogating the present counter value with a DIA PIT instruction. A Clear command (NIOC PIT) stops the counter, and a Start command (NIOS PIT) starts it. The PIT counter/register is double-buffered, so no counts are lost when the register is read. The PIT counter rate is determined by setting the SIO switches as described earlier in this chapter.

The real-time clock generates interrupts at any one of four program selectable frequencies. 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 asynchronous interface is a universal asynchronous receiver/transmitter (UART). It consists of a terminal input (TTI) with device code 10 (interrupt priority mask bit 14) and a terminal output with device code 11 (mask bit 15). The buffers, registers, and state machine logic for the interface are contained on the SIO chip. The driver and receiver for the EIA RS-232 or 20-mA current loop lines are separate IC devices.

The interface operates at one of 16 baud rates, depending on the setting of the SIO switches as described in "SPU Card Tailoring" in *Installion Guide for Model 20 and Model 30 Systems*. The following baud rates have two stop bits: 50, 75, 110, and 134.5. The remaining rates — 150, 200, 300, 600, 1200, 1800, 2000, 2400, 4800, 9600, 19200, and 38400 — have one stop bit.

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 CTS is changed while it is in progress.

The terminal output section is also equipped with a Data Terminal Ready (DTR) signal, which it continually asserts as long as the system is powered up.

The terminal input section contains logic that detects the depression of the console Break Key. The logic contained on the SIO chip interprets two consecutive framing errors as a depression of this key. Then, provided the Break Key enabling switch has been set (see "SPU Card Tailoring" in *Installion Guide for Model 20 and Model 30 Systems*, 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 non-maskable interrupt causes the processor to enter the virtual console mode. For information on the Virtual Console, refer to Chapter 3.

NOTE *The Model 20 and Model 30 system processor unit asynchronous communications interface receives and transmits eight-bit data characters without parity. If the terminal device connected to this interface port operates with a data character length of seven bits, it should be configured to operate with "mark parity".* If the terminal device connected to this interface 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.

The 12-bit CPU status register, is contained on the SIO chip. A *Read CPU Status* instruction DIS *ac*,CPU places the contents of this register into the specified accumulator. Six status bits report the occurrences of nonmaskable interrupts: 0, 3, 4, 5, and 10. 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 %tablenum(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 (3, 4, 5, and 10). They are not ordinarily manipulated by the user.*

CAUTION *Never set bit 15 to 1. The least significant bit in the accumulator specified by* a CPU Acknowledge *(DOAP CPU) instruction must be 0.*

Table 4-2 CPU status register contents

Bit	Name	Function
0	Powerfail	Indicates the state of the power supply powerfail line.
1	Interrupt On	A 1 signifies that CPU interrupts are on (enabled), and a 0 that CPU interrupts are off (disabled).
2	SIO	Always set to 1, this bit is used for diagnostics.
3	Break Key	A 1 indicates that the system console Break Key has been depressed.
4	Power Up	A 1 indicates that the system has just been powered up.
5	Halt	A 1 signifies that a Halt instruction has been decoded and the Halt Dispatch bit in the SPU register is 1.

Table 4-2 CPU status register contents (Continued)

Bit	Name	Function
6	Halt Dispatch	Indicates the state of the Halt Dispatch switch in the SIO switch register. A 1 indicates that the Halt Dispatch switch is set to 1 (on). In this case, control passes to the virtual console when a Halt instruction has been decoded. A 0 means that the Halt Dispatch switch is set to 0 (off). Accordingly, when a Halt instruction is decoded, the CPU will simply halt; bit 5 in the SPU status register will not be set.
7	Interrupt Request	Indicates the state of the SIO interrupt request pin (SIO INT). A 1 means that a device is requesting an interrupt. (Can be used to test for interrupt requests when interrupts are disabled.)
8-9	---	Reserved for future use. Set to 0.
10	Run	A 1 signifies that a user program instruction has just been executed in the virtual console one-step mode.
11-14	---	Reserved for future use. All are set to 0.
15	Test	Always set to 0, this bit is used for diagnostics.

Address Latch The address latch preserves the data on the local output bus, $\overline{\text{OUT<0-15>}}$, from the time ADDRESS LATCH is asserted by the timing/control logic (during the address phase) until the next address phase occurs. The latched data word is the address $\overline{\text{ADR<0-15>}}$. Part or all of this address is supplied to the virtual console, to the timing/control and I/O decode circuits of the CPU section, and to the parity section.

The signal ADDRESS LATCH also latches the following system status and control signals for the same time period: MAPEN, MEMCYCLE, $\overline{\text{ADREN}}$, WRITE, $\overline{\text{FAULT}}$, $\overline{\text{EXT INTR}}$, $\overline{\text{SIO INT}}$, PAR DONE, and $\overline{\text{UNADR7}}$. In addition, the CPU I/O decoder uses ADDRESS LATCH to latch its decoded I/O instructions, in this way preserving them throughout the data phase.

Parity Checking The parity checking section contains Busy/Done logic and three registers:

a parity control register,

a parity status register, and

a parity address register.

The parity control register is loaded with bits $\overline{\text{OUT<14,15>}}$ when it receives the control strobe $\overline{\text{DOA PAR}}$ from the I/O decoder. $\overline{\text{DIC CPU}}$ from the I/O decoder clears this register. Its contents determine the mode of operation of the parity calculation and comparison logic. (See the *16-bit Real-Time* ECLIPSE *Assembly Language Programming*.)

The parity calculation logic checks the parity of the two bytes, $\overline{\text{OUT<0-7>}}$ and $\overline{\text{<OUT8-15>}}$, that it receives from the local output buffer. During a memory cycle when $\overline{\text{INPUT PHASE}}$ and READY are high, the results of the calculation are sent out on the memory bus lines $\overline{\text{WH/PARH}}$ and $\overline{\text{WL/PARL}}$. During a memory cycle when $\overline{\text{INPUT PHASE}}$ is low and READY high, the results of the parity calculation are compared with the parity bits $\overline{\text{PARH}}$ and $\overline{\text{PARL}}$ received from memory.

A parity error determined by the above comparison will store the results of the check (high error or low error) and set the parity Done flag to 1, assuming that parity checking has been enabled. The logical address currently on the address bus ($\overline{ADR<1\text{-}15>}$) will be latched into the parity address register. The state of the Done flag will cause a CPU interrupt.

When the parity address latch receives a $\overline{DIA\ PAR}$ from the I/O decoder, it places the address it contains on the local input bus. When the parity status register receives a $\overline{DIB\ PAR}$ from the same source, it places its contents on the local bus (bits $\overline{IN<0,1>}$). When the Busy/Done logic receives a $\overline{DIS\ PAR}$, it places the state of the Done flag on the local input bus (bit $\overline{IN0}$).

Virtual Console

The virtual console consists of a 2-kilobyte array of random access (read/write) memory, along with a programmed read-only memory containing a 512-word (16 bits per word) program. This program consists of a self-check routine and a set of commands that give users access to CPU registers and memory for debugging machine language programs and allow users to perform program loads. The routines are described in Chapter 3 of this manual.

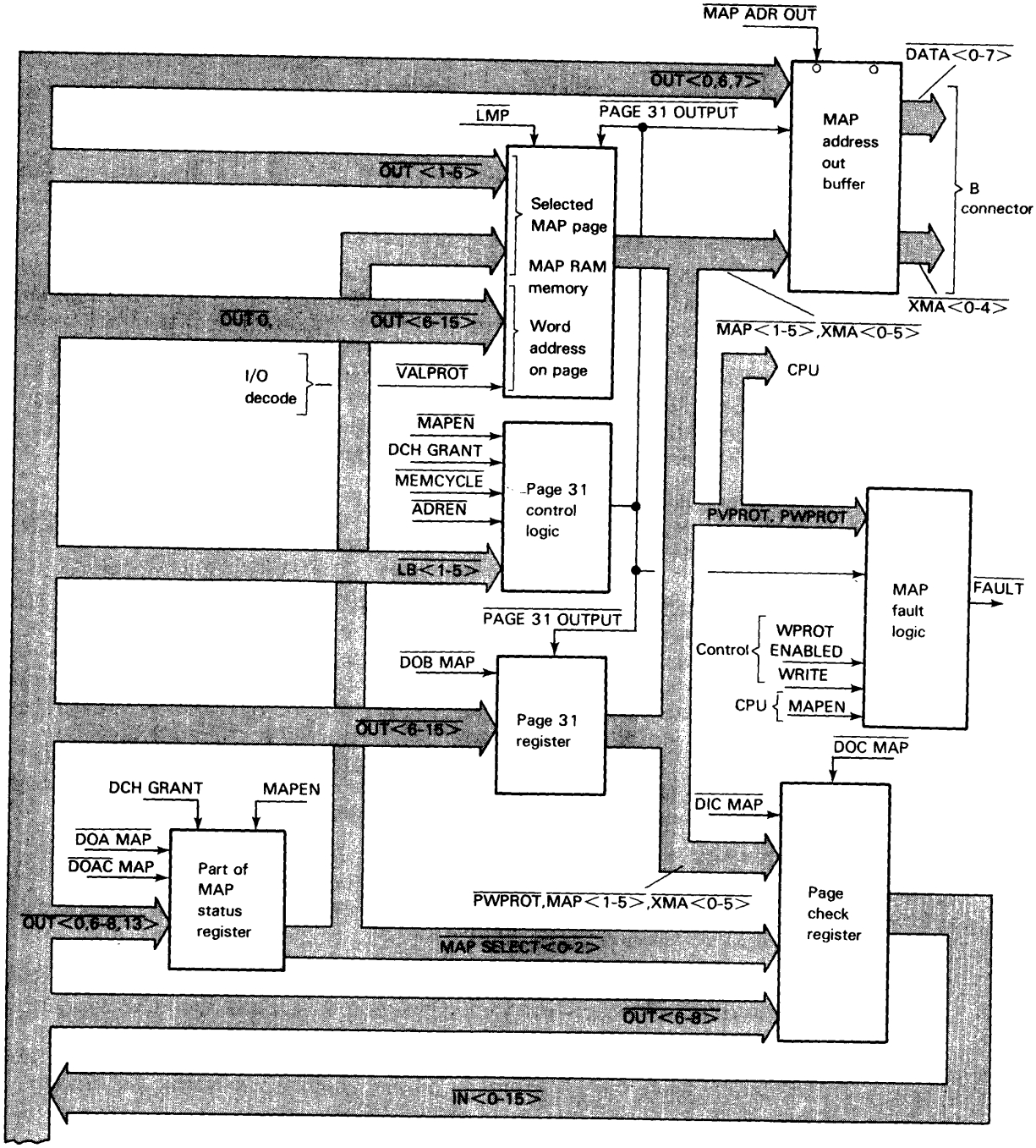
The contents of the read-only memory location, addressed by $\overline{ADR<1\text{-}15>}$ on the address bus, are placed on the local input bus when the timing/control PAL chip asserts $\overline{PROM\ OUT}$.

The virtual console read/write memory is enabled when it receives $\overline{RAM\ CS}$ from the timing/control PAL chip. The contents of the location addressed by $\overline{ADR<1\text{-}15>}$ are either read or written, with the data transfer occurring via the local input and output busses. The control signal $\overline{RAM\ WRITE}$ controls the direction of transfer.

Memory Allocation and Protection

The memory allocation and protection (MAP) unit, shown in Figure 4-14, consists of the following:

- the MAP memory, consisting of a 256 x 12 RAM array,
- the 1-word by 10-bit page 31 register,
- the 1-word by 16-bit MAP status register,
- the 1-word by 13-bit page check register,
- page 31 control logic,
- fault logic,
- a device code decoder.



DG-08305

Figure 4-14 The MAP unit

The MAP unit receives addresses and data (to be placed in the MAP memory and registers) through the local bus. It produces the physical page numbers required to map the current user's logical address space into physical address

space. These bits are $\overline{XMA<0\text{-}4>}$ and $\overline{DATA<1\text{-}5>}$. Local output bus bits $\overline{OUT<0,6,7>}$ are passed through unchanged to become $\overline{DATA<0,6,7>}$.

The Load Map Operation

Four user maps and one data channel map are stored in the MAP memory in response to a *Load Map* instruction preceded by a *Load MAP Status* (DOA MAP) or an *Initiate Page Check* (DOC MAP) instruction.

DOC MAP or DOA MAP causes bits $\overline{OUT<6\text{-}8>}$ to be loaded into the MAP status register. (The DOC MAP loads these bits into the page check register as well.) The MAP status register produces three map select signals, $\overline{MAP\ SELECT<0\text{-}2>}$. These bits address the selected user area (A, B, C, D, or data channel) within the MAP memory to be loaded by the next Load Map instruction. Also, the write protection enable bit is loaded by DOA MAP (bit 11).

When the Load Map instruction is executed, the MAP page assignments are made as follows: bits $\overline{OUT<1\text{-}5>}$ on the local bus represent the logical page number; they address the user page number in the selected user area of the MAP memory. Bits $\overline{OUT<6\text{-}15>}$ represent the physical page number to be assigned to the given logical page number; these bits are stored in the addressed location in MAP memory when $\overline{LMP}$ is asserted. Additionally, two more bits are stored in that addressed location: the write protect bit (received via $\overline{OUT0}$) and the validity protect bit (received via the device code decoder).

The device code decoder detects that bit 0 and bits 6-15 are all set to one, indicating a validity-protected page, and also supplies a recoded device code to the I/O decode and control logic.

Mapped Mode Operation

The mapped mode is entered with a *Load Map Status* instruction (DOA MAP) that sets a user or data channel enable bit (14 or 15) in the MAP status register to one. If a user map is to be enabled, it is selected by the same instruction (setting bits 0 and 13 in the MAP status register). Setting the enable user map bit inhibits the interrupt system, and the MAP waits for either an indirect reference or a return type instruction. Either event releases the interrupt system and allows the MAP to begin translating addresses. Address translation begins (1) after the first level of the next indirect reference; or (2) after the first Pop Block, Pop Jump, Return, or Restore with no stack fault.

NOTE *The MAP status register physically located in the MAP unit is only a copy of part of the MAP status register that is contained in the mE670 CPU. In particular, bits 1-5, 9-12, 14, and 15 of the MAP status word are stored in the mE670 MAP status register.*

The MAP status register asserts signals $\overline{MAP\ SELECT}$ and WPROT ENABLED as just described. When an address appears on the local bus, the MAP memory sends the physical memory page number corresponding to the logical page number (bits 1-5 of the address) for the selected user to the MAP buffer. The MAP buffer sends this page number on the memory bus during the address phase, if $\overline{PAGE\ 31\ OUTPUT}$ is not asserted (low).

Page 31 Register

The page 31 register is loaded by a *Map Page 31* instruction (DOB MAP). Its 10-bit contents contain the physical page number to which logical page 31 is to be mapped in the unmapped mode. The page 31 control logic will assert $\overline{PAGE\ 31\ OUTPUT}$ to enable the output of the page 31 register when bits $\overline{OUT<1\text{-}5>}$ are all one, MAPEN is low, and the operation is a nondata channel memory cycle during an address phase.

Page Check

An *Initiate Page Check* instruction (DOC MAP) loads the page check register with the physical translation of the logical page number of the

page to be checked. The instruction also loads the user map code of the user map to be checked. The page number is received via the MAP memory as MAP<1-5> and $\overline{\text{XMA<0-4>}}$. A *Page Check* instruction (DIC MAP) causes the page check register to send this physical page number and the user map code to the CPU over the local bus.

Map Faults

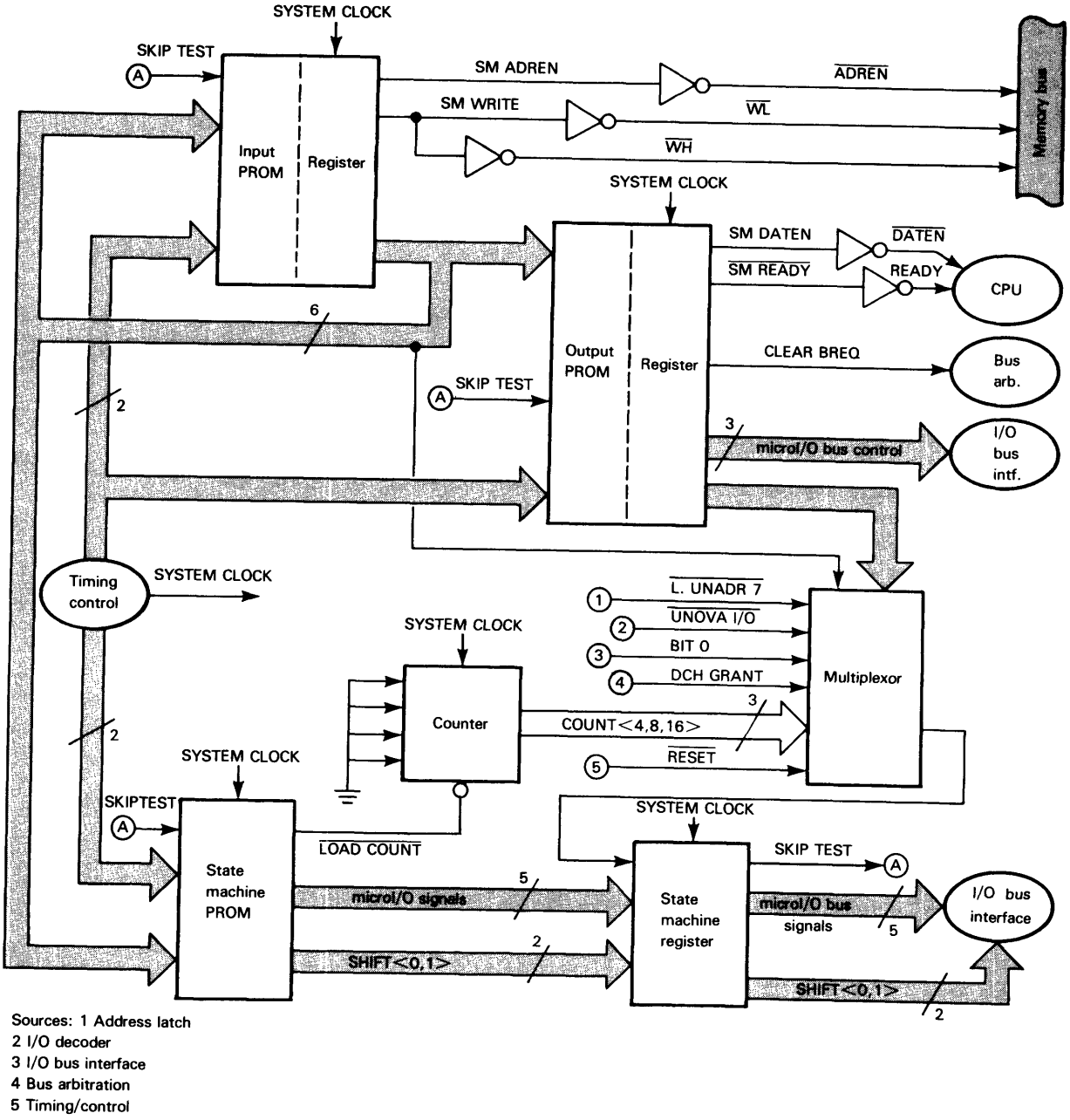
The MAP fault logic asserts the signal $\overline{\text{FAULT}}$ if any of the following conditions occur during a processor-initiated memory transition: (1) a memory reference to a validity-protected page, or (2) a write attempt to a write-protected page when write protection is enabled.

microI/O Bus Interface

The microI/O bus interface performs the serial-parallel and parallel-serial conversions, timing, and buffering necessary to pass data and instructions from the SPU local bus to I/O devices connected to the Model 20 and Model 30 system with the microI/O bus. The interface section consists of two subsections: the microI/O bus interface state machines and the microI/O bus interface.

State Machine Section

The state machines coordinate the timing of system operations, which take 4 cycles of the 8-MHz SYSTEM CLOCK signal, with I/O bus operations, which take 3 to 60 cycles of the SYSTEM CLOCK signal. The state machines also produce memory bus control signals during data channel operations. Figure 4-15 diagrams the state machines of the microI/O bus interface.



Sources: 1 Address latch
2 I/O decoder
3 I/O bus interface
4 Bus arbitration
5 Timing/control

DG-08916

Figure 4-15 microI/O bus interface state machine

The state machines consist of two registered 512 x 8 PROMs and an unregistered 512 x 8 PROM with an external register. A synchronous, 4-bit counter and a 1-of-8 multiplexor are associated with these PROMS. The state machines are driven by the SYSTEM CLOCK signal, shown earlier in Figure 4-10. Their states are determined by the conditions of seven signals:

- Two system timing signals, PHASE and PERIOD (see Figure 4-11)
- Two I/O control signals — $\overline{\text{UNOVA I/O}}$ from the I/O decode logic, and

L.UNADR7 derived from OUT7 by the address latch — that signal the performance of an I/O operation and indicate the direction of I/O transfer

- BIT0, from the microI/O bus interface serial to parallel converter, which specifies a memory operation if 0 or a virtual console RAM memory operation when 1
- DCH GRANT, from the bus arbitration PAL circuit
- One system reset signal

The outputs of the state machines consist of four data channel control signals; one control signal CLEAR BREQ for the bus arbitration PAL circuit; three microI/O bus interface control signals; and seven microI/O bus interface signals.

The data channel control signals are SM ADREN, SM DATEN, SM READY, and SM WRITE. They perform the functions of the corresponding, CPU-issued system control signals when the system state machine takes control of the system bus for data channel operations.

The three microI/O bus interface control signals are as follows.

LATCH UNOVA: when high, causes data to be latched into the holding latch of the bus interface.

UNOVA INPUT: enables the output of the holding latch.

CLEAR INPUT: when low, clears the serial-to-parallel converter in the bus interface.

The seven microI/O bus interface signals include the following.

I/O DATA1 and I/O DATA2: the serial data transmitted on the I/O bus.

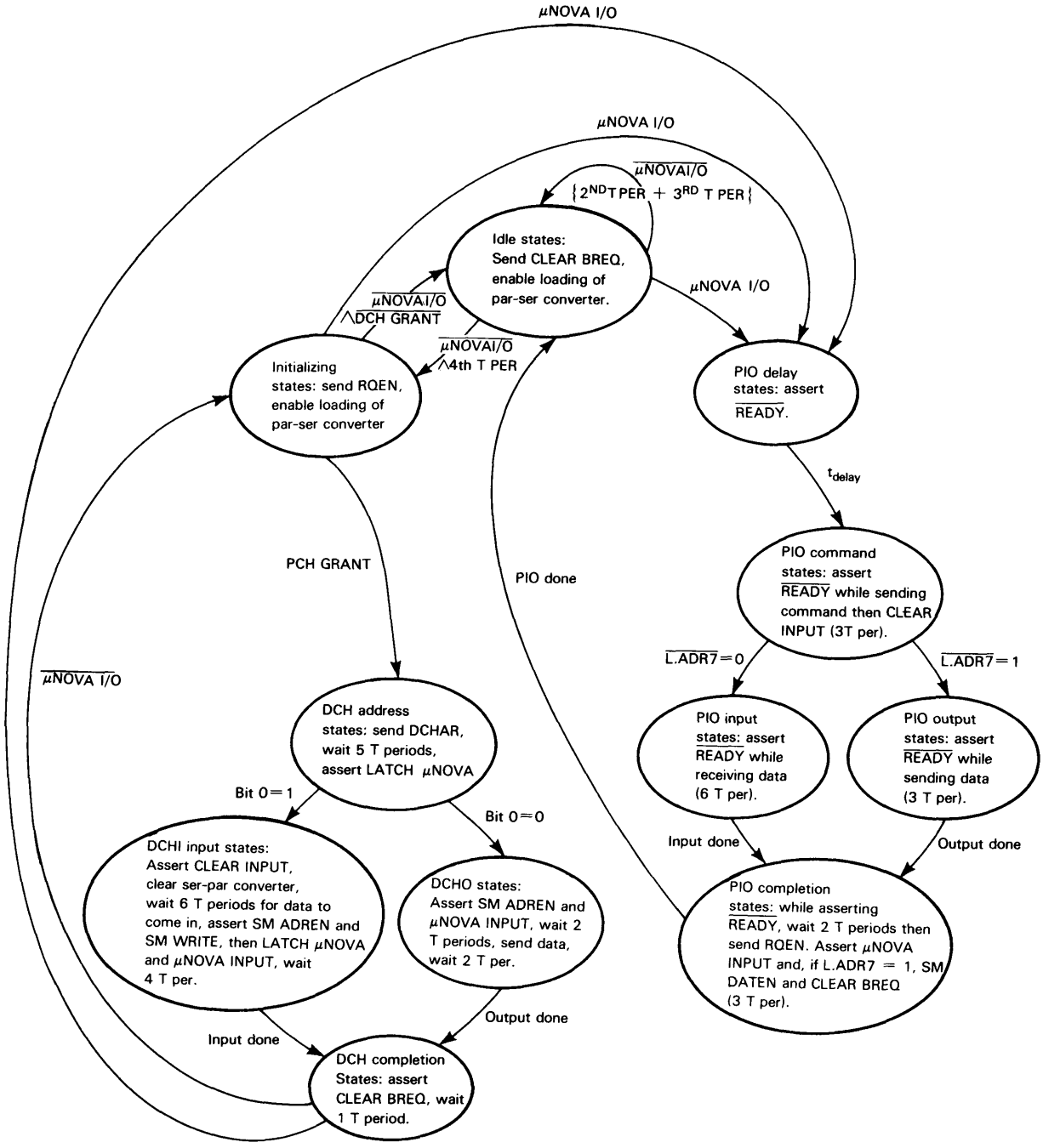
INPUT: indicates the direction of data transfer (high indicates in to the CPU, low indicates out).

I/O CLOCK: which synchronizes data transmission on the I/O DATA lines.

SHIFT<1,2>: these signals control the direction of the shifting done by the parallel-to-serial converter in the bus interface.

PREAMBLE: when low, enables the sending of short data transfers (Request Enable, Data Channel Address Request, Data, and I/O Command).

Figure 4-16 shows the loops of the microI/O bus state machines in summary form.

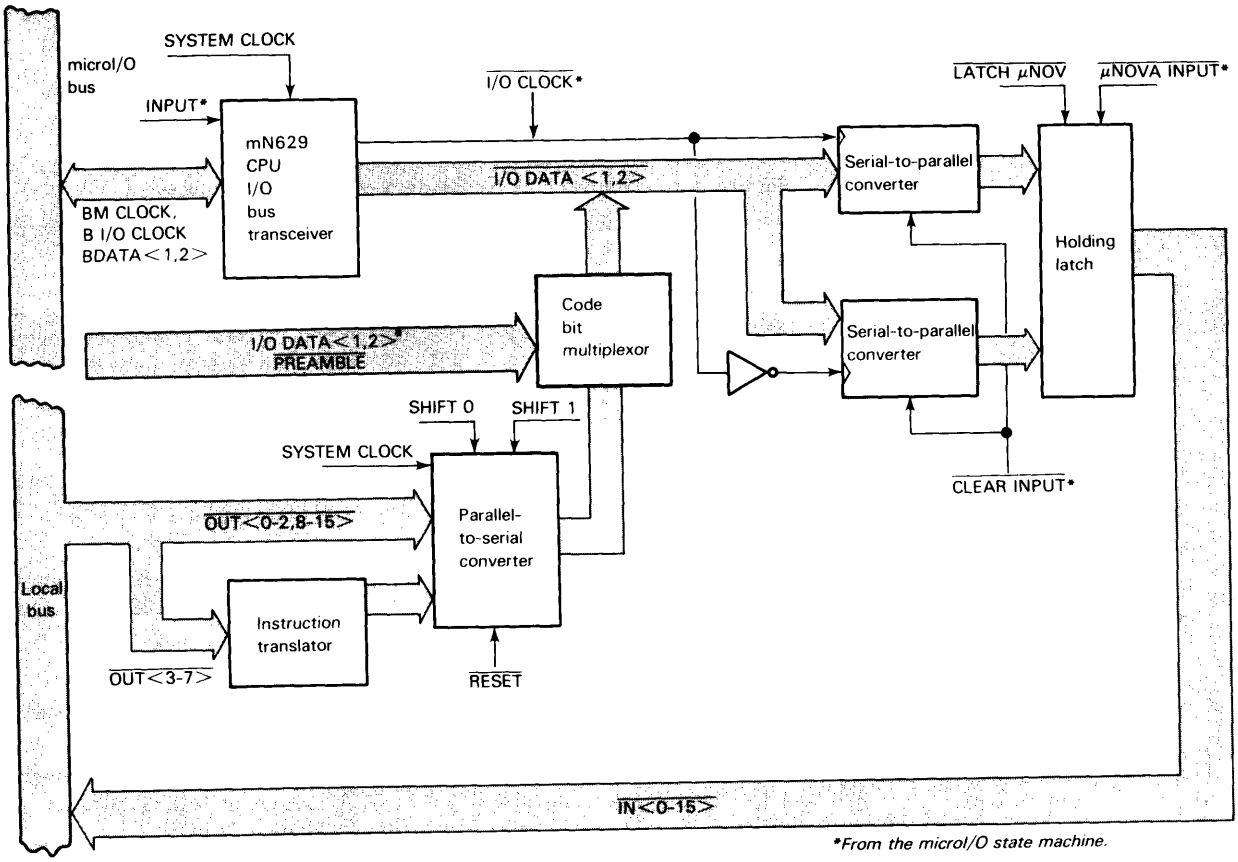


DG-08917

Figure 4-16 Summarized operation of the microI/O state machine

Bus Interface

The bus interface, shown in Figure 4-17, consists of an mN629 CPU I/O transceiver, a parallel-to-serial converter, a serial-to-parallel converter, and a holding latch.



DG-08309

Figure 4-17 Bus interface

Data to be transmitted from the local output bus are input to the bus interface parallel-to-serial converter via the microI/O bus instruction translater. SYSTEM CLOCK clocks this parallel data out in serial form under the control of signals SHIFT<0,1> from the microI/O bus state machines. The instruction translator performs the instruction conversion needed to make the microI/O bus operate compatibly with the ECLIPSE instruction set. (DIC CPU is translated to DOA CPU.)

Short-form transmissions to be sent on the I/O bus are received from the microI/O bus state machine. These transmissions are multiplexed with the output of the parallel-to-serial converter to form the input to the mN629 transceiver.

The mN629 transceiver receives data from the I/O bus and sends it, in serial form, to the serial-to-parallel converter. There the data is clocked in by the I/O synchronizing signal from the microI/O bus state machine, I/O CLOCK. The output of the converter is latched into the holding latch when LATCH UNOVA is

asserted (low). The timing signal UNOVA INPUT places the output of the holding latch on the local bus.

Signals

Table 4-3 lists all the external signals sourced or sunk by the Model 20 and Model 30. Table 4-4 concludes the chapter by listing all the internal signals of the Model 20 and Model 30 card that are mentioned in the text or figures of this book.

Table 4-3 External signals

Name	Pin(s)	Source	Destination	Description
Memory Bus Signals				
DATA0	B50	SPU Memory FPU	FPU, Memory SPU, FPU SPU	Address phase: user mem. Data phase: data word msb
DATA<1-15>	B23, 24, 27, 28, 31-34, 37, 38 41, 42 45, 46 49	SPU Memory FPU	FPU, Memory SPU, FPU SPU	Address phase: low-order address bits. Data phase: low-order data bits
XMA<0-4>	B25, 40, 44, 47 51	SPU	Memory	Address phase: high-order address bits (XMA0 is msb)
SYSCLOCK	B52	SPU	FPU, memory	System clock
BUS READY	B18	Memory	SPU	Data phase: indicates memory is ready to finish cycle
BUS MEMCYC	B26	SPU	FPU, memory	Address phase: indicates memory ref. (otherwise cycle is I/O)
BUS ADREN	B48	SPU SPU	FPU, memory	Indicates valid address
WH/PARH,	B30	SPU Memory	Memory SPU	Address phase: high byte write Data phase: high byte data odd parity
WL/PARL	B29	---	---	Address phase: low byte write Data phase: low byte data odd parity
microI/O bus Signals				
BMCLOCK, BMCLOCK	B1, 2	I/O bus interface	I/O bus	Differential system clock for I/O devices
BI/ODATA1, BI/ODATA1	B4, 5	I/O bus interface I/O bus	I/O bus I/O bus interface	Differential bidirectional I/O data line used for serial transfer of high byte of data word
BI/ODATA2, BI/ODATA2	B12, 13	I/O bus interface I/O bus	I/O bus I/O bus interface	Differential bidirectional I/O data line used for serial transfer of low byte of data word

Table 4-3 External signals (Continued)

Name	Pin(s)	Source	Destination	Description
BI/OCLOCK, $\overline{\text{BI/OCLOCK}}$	B15, 16	I/O bus interface I/O bus	I/O bus I/O bus interface	Differential bidirectional clock; synchronizes data transfers
$\overline{\text{CLEAR}}$	B6	SPU RESET	I/O bus, FPU	System reset caused by power up
$\overline{\text{EXTINT}}$	B8	I/O device	SPU	Interrupt request
INTPOUT	B19	SPU	I/O device	Interrupt priority line
DCHPOUT	B21	SPU	I/O device	Data channel priority line
$\overline{\text{EXTDCHR}}$	B9	I/O device	SPU	Data channel request
Power Supply Signals				
POWEROK	B22	Power supply	microI/O interface	Indicates that DC power is within specifications
$\overline{\text{PWRFAIL, PF}}$	B7, B20	Power supply	SIO	Indicates that AC power is not within specs
$\overline{\text{RTC}}$	B35	Power supply	SIO	Line frequency
DC Power				
Ground	B3, 11, 14, 36, 53, 54, C1, A7	---	SPU, FPU, Memory, System console	
+5V	B57, 59, 60, A9	Power supply	SPU, FPU, Memory, System console	
−5V	B58, A3	Power Supply	SPU, Memory, System console	
+12V	B55, 56, A23	Power supply	SPU, Memory, System console	
Floating Point Signals				
QUACK	C25	SPU	FPU	Acknowledges microcode
$\overline{\text{QSKIP}}$	C27	FPU	SPU	Used to control microcode flow for FPU
QREQ	C31	SPU	FPU	Requests microcode
QPIPE	C19	SPU	FPU	Controls instruction flow
$\overline{\text{QFETCH}}$	C33	SPU	FPU	Controls instruction flow
Other C-connector Signals				
$\overline{\text{TEST}}$	C4	---	SPU	Causes the SCP self-test to loop
$\overline{\text{BLOCK}}$	C45	SPU	BMC	Indicates that the CPU is executing an indivisible process

Table 4-3 External signals (Continued)

Name	Pin(s)	Source	Destination	Description
System Console Signals				
TTIN	A1	System console	SPU	Serial data input
TTOUT	A21	SPU	System console	Serial data output
DTR	A10	SPU	System console	Opens and holds open the communication line
CTS	A2	System console	SPU	Enables data transmission

Table 4-4 Internal signals

Name	Pin(s)	Description
CPU Bus		
$\overline{\text{MB<0-15>}}$	CPU17-CPU32, SIO17-SIO32, XMC12-XMC29	CPU address/data, bidirectional
$\overline{\text{ADREN}}$*	CPU16, SIO16, XMC30	Address enable; carried on the system memory bus
$\overline{\text{DATEN}}$	CPU13, SIO12	Data enable
$\overline{\text{WH, WL}}$	CPU11, 12, SIO10, 11, XMC31 (WL), B29, 30	Write operation to high or low order bytes requested; carried on the system memory bus
READY**	CPU4, SIO13, XMC9	Bus transaction in progress will terminate this cycle; also affects bus arbitration; carried on system memory bus
BLOCK	CPU10, SIO47, C45	Indivisible operation in progress; also affects bus arbitration
MEMCYCLE***	CPU15, SIO15, XMC32	Memory cycle is asserted (otherwise I/O cyc.); carried on system memory bus
Bus Arbitration Signals**		
BREQ	CPU47	Requests CPU relinquish bus
DCH GRANT	---	Grants a data channel to an I/O device
CLEAR BREQ	---	Indicates microI/O bus data channel operation completed
Timing Signals (see Figure 00)		
CS<0-2>	---	Timing state machine output (CS0 is also called PHASE)
PH<1,2>	---	2-phase system clock
APH<1,2>	CPU43, 44, SIO43, 44, XMC33, 34	2-phase non-overlapping clock for CPU, SIO, and XMCs
$\overline{\text{T3}}$	---	Virtual console scratchpad memory write enable and I/O decoder enable

Table 4-4 Internal signals (Continued)

Name	Pin(s)	Description
SCLK, SYSTEM CLOCK	---	microI/O bus master clock and clock for I/O interface state machines
PERIOD	---	System state information for I/O interface
Timing/Control Signals (See Figure 00)		
BUS TERM	---	CPU bus transceiver enable timing
$\overline{\text{INPUT PHASE}}$	---	CPU bus transceiver direction
$\overline{\text{LOCAL BUS IN}}$	---	Local input bus buffer enable
$\overline{\text{HIGH PROM OUT}}$	---	Virtual console PROM output enable
RAM ENABLE	---	Virtual console scratchpad memory chip select
$\overline{\text{HIGH BUS ENABL}}$	---	Enables high bus transceiver during data phase of all transfers and address phase of unmapped memory references, except if a MAP fault occurs
$\overline{\text{LOW BUS ENABLE}}$	---	Enables low bus transceiver during all data transfers, except during MAP faults
$\overline{\text{RAM WRITE}}$	---	Virtual console scratchpad memory write enable
$\overline{\text{MAP ADDRESS OUT}}$	---	Enables MAP address out buffer during address phase of mapped memory operation
$\overline{\text{ADDRESS LATCH}}$	---	Latches logical address and I/O encoding information
Reset Signals		
$\overline{\text{CPU RESET}}$	CPU3	Causes CPU to enter and remain in a halted state until it receives an NMI
$\overline{\text{SIO RESET}}$	SIO3	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
$\overline{\text{XMC RESET}}$	XMC7	Puts XMCs in a known idle state
I/O Decode Signals		
$\overline{\text{DOAP CPU}}$	---	Write to run light register
$\overline{\text{DOA MAP}}$	---	Write to MAP status register
$\overline{\text{LMP}}$	---	Write to MAP RAMs
$\overline{\text{DOB MAP}}$	---	Write to MAP page 31 register
$\overline{\text{DOC MAP}}$	---	Write to initial page check register
$\overline{\text{DOA PAR}}$	---	Write to parity enable register
$\overline{\text{DIC CPU}}$	---	Reset CPU
$\overline{\text{DIA APL}}$	---	Read Auto Program Load register
$\overline{\text{DIB PAR}}$	---	Read Memory Fault Code
$\overline{\text{DIB CPU}}$	---	Read device code of interrupting device (Interrupt Acknowledge)
$\overline{\text{DIS PAR}}$	---	Read parity status
$\overline{\text{DIS CPU}}$	---	Read CPU status register
$\overline{\text{DIA PAR}}$	---	Read memory fault address
$\overline{\text{DIC MAP}}$	---	Read page check register
$\overline{\text{DIS}}$	---	Read microI/O bus device status

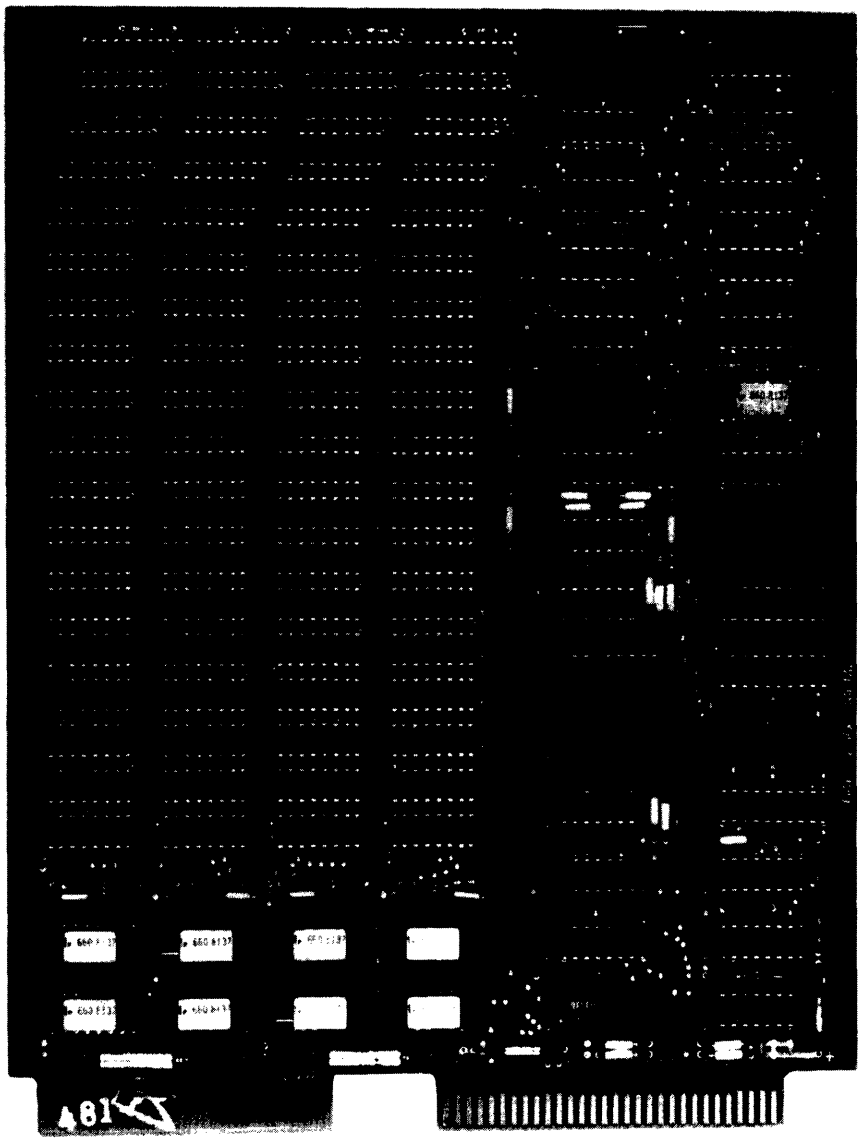
Table 4-4 Internal signals (Continued)

Name	Pin(s)	Description
PRE DIC CPU	---	A reset has been decoded
UNOVA I/O	---	An instruction for a microI/O bus device has been decoded
MAP Signals		
MAPEN	CPU8	Enable MAP logic
PVROT	CPU48	Validity-protection violation
PWROT	CPU2	Write-protection violation
FAULT	---	MAP fault
XMA<0-4>	---	Physical page number
MAP<1-5>		Physical page number
MAP SELECT<0-2>	---	Selected user or data channel map
PAGE 31 OUTPUT	---	Indicates memory reference to highest page with the MAP disabled
Parity Signals		
START PAR	---	From the I/O decoder, clears the parity done flag
PAR DONE	---	Parity done flag, asserted only if parity checking enabled; causes a CPU interrupt
microI/O bus Interface Signals		
LATCH UNOVA	---	Data latch for microI/O bus interface holding latch
UNOVA INPUT	---	Output enable for interface holding latch
CLEAR INPUT	---	Clears interface serial-to-parallel converter
INPUT	---	Controls direction of microI/O bus interface transfer: high in, low out
SHIFT<1,2>	---	Parallel-to-serial converter shift direction control
PREAMBLE	---	Short data transfer enable; enables sending of Request Enable, Data Channel Address Request, (the next word is) Data, and (the next word is a) Command
SKIP TEST	---	State condition signal from interface multiplexor, sent via the state machine register to all the interface PROMs
LOAD COUNT	---	Resets the interface counter
Miscellaneous Signals		
UNADR<5-7>	---	microI/O I/O address translated by the bus arbitration PAL from OUT<5-7>
L.UNADR7	---	Latched UNADR7; indicates direction of I/O transfer
NMI	CPU 45	Nonmaskable interrupt
INT	CPU 46	I/O control program interrupt
CR <0-7>	XMC1-XMC3, XMC39-XMC45, CPU42-CPU34	Microinstruction from XMC to CPU

* *See also BUS ADREN in Table ??.*
** *See also BUS READY in Table ??.*
*** *See also BUS MEMCYC in Table ??.*
**** *Additional bus arbitration signals appear under "microI/O bus signals" in this table, and in Table ?? under "Other C-connector Signals."*

Memory Cards 5

This chapter describes the Model 20 and Model 30 memory card, which provides 256 or 512 Kbytes of dynamic MOS random-access memory (RAM) for the Model 20 and Model 30 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 (36 ICs); the 512 Kbyte memory card contains four banks of RAM ICs (72 ICs).



PH-0518

Figure 5-1 Random-access memory card

A maximum of four 512-Kbyte memory cards provides the system with a 2-Mbyte physical address space. All memory cards present in the Model 20 and Model 30 system must be installed in the cage of the CPU logic module (CLM). The RAM cards are positioned in the system's memory space with jumpers. Each card has an independent controller for generating its timing and refresh operations.

Figure 5-1 shows a random-access card and Table 5-1 summarizes its characteristics.

Table 5-1 Memory card characteristics

Memory Type:	Dynamic n-channel MOD RAM
Card Capacity:	262,144 or 524,288 bytes 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

Memory Card Size	Supply Voltage (Volts DC)	Current draw (Amps)	Power dissipation (Watts)
256-kilobyte	+5	2.1	10.5
512-kilobyte	+5	2.4	12.0

The memory card contains four card-select jumpers that allow you to position the memories in various blocks of memory space. Normally, RAM is added to a system from the bottom (address 0) of memory up. Refer to *Installing a Model 20 and Model 30 System* for the jumper locations and configurations for the memory.

Interfacing

Memory cards communicate with the CPU and data channel controllers through their B edge connector. This connector also provides all power to the memory. (Model 20 and Model 30 memories do not use the A connector.) Figure 5-2 shows the RAM card connector positions. Figure 5-3 shows the B connector pin assignments.

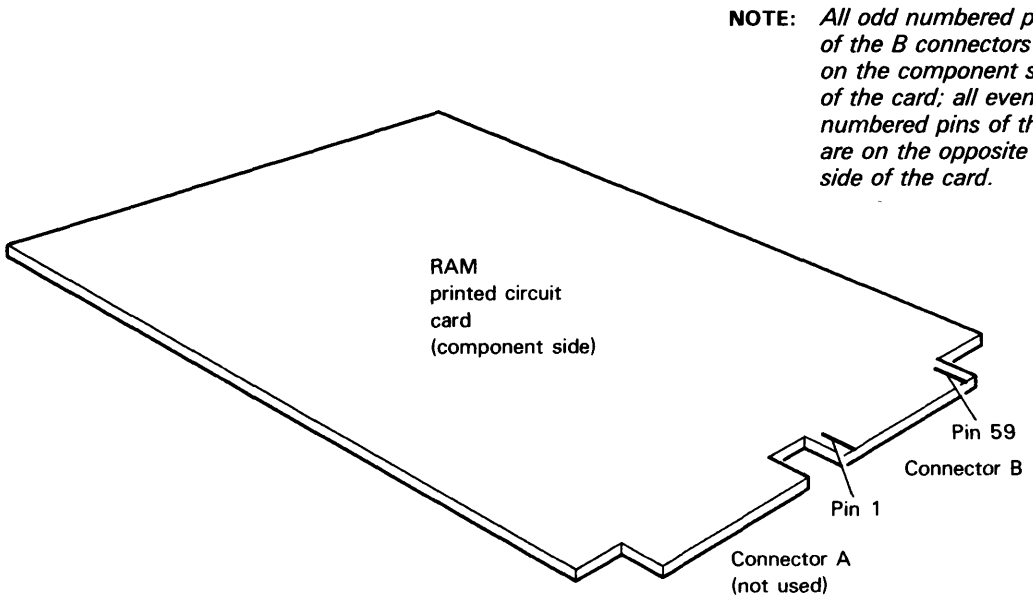


Figure 5-2 Memory card connection positions

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

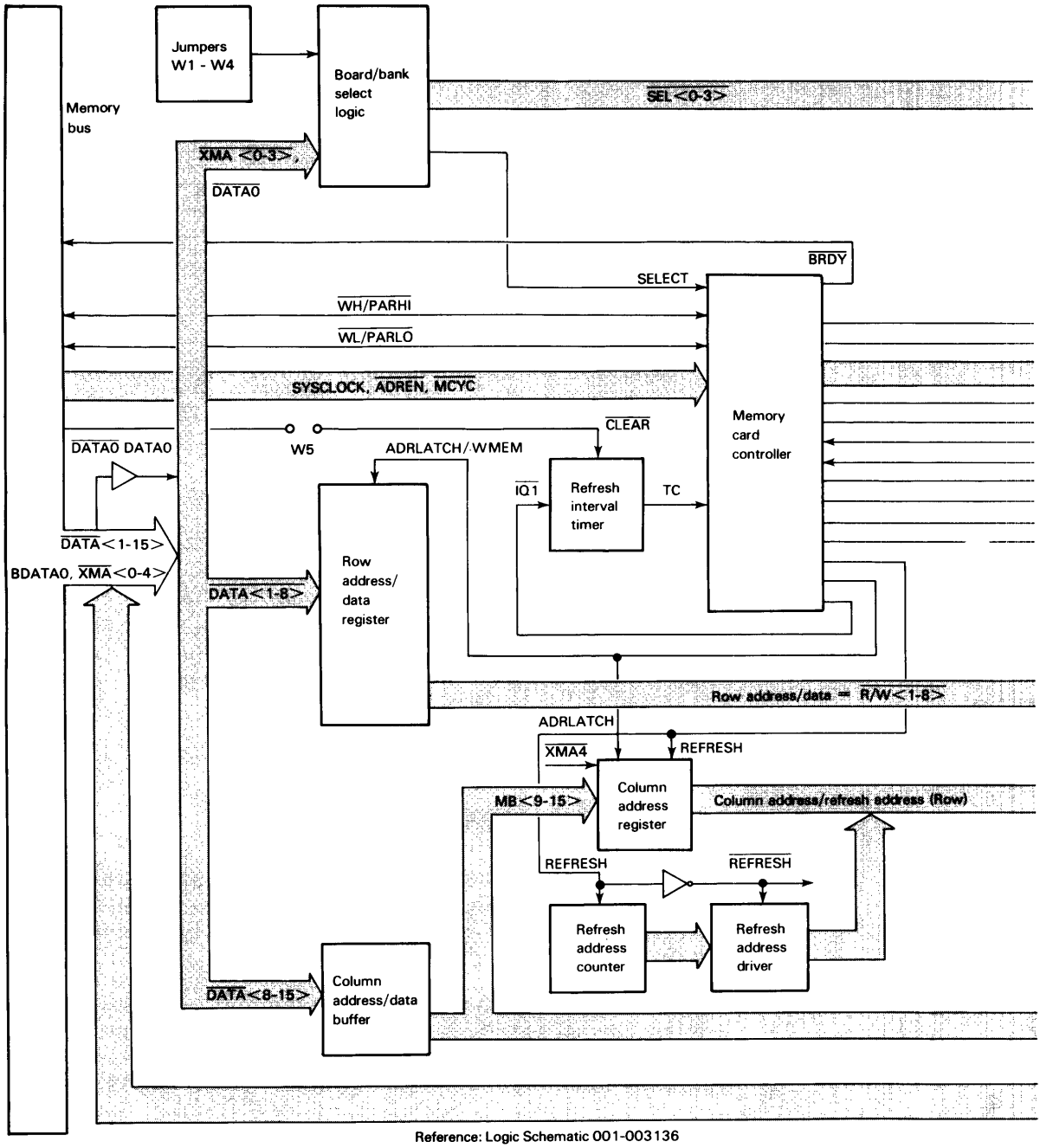
ID-00625

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. These locations are assigned to system memory locations by the card-select jumpers.

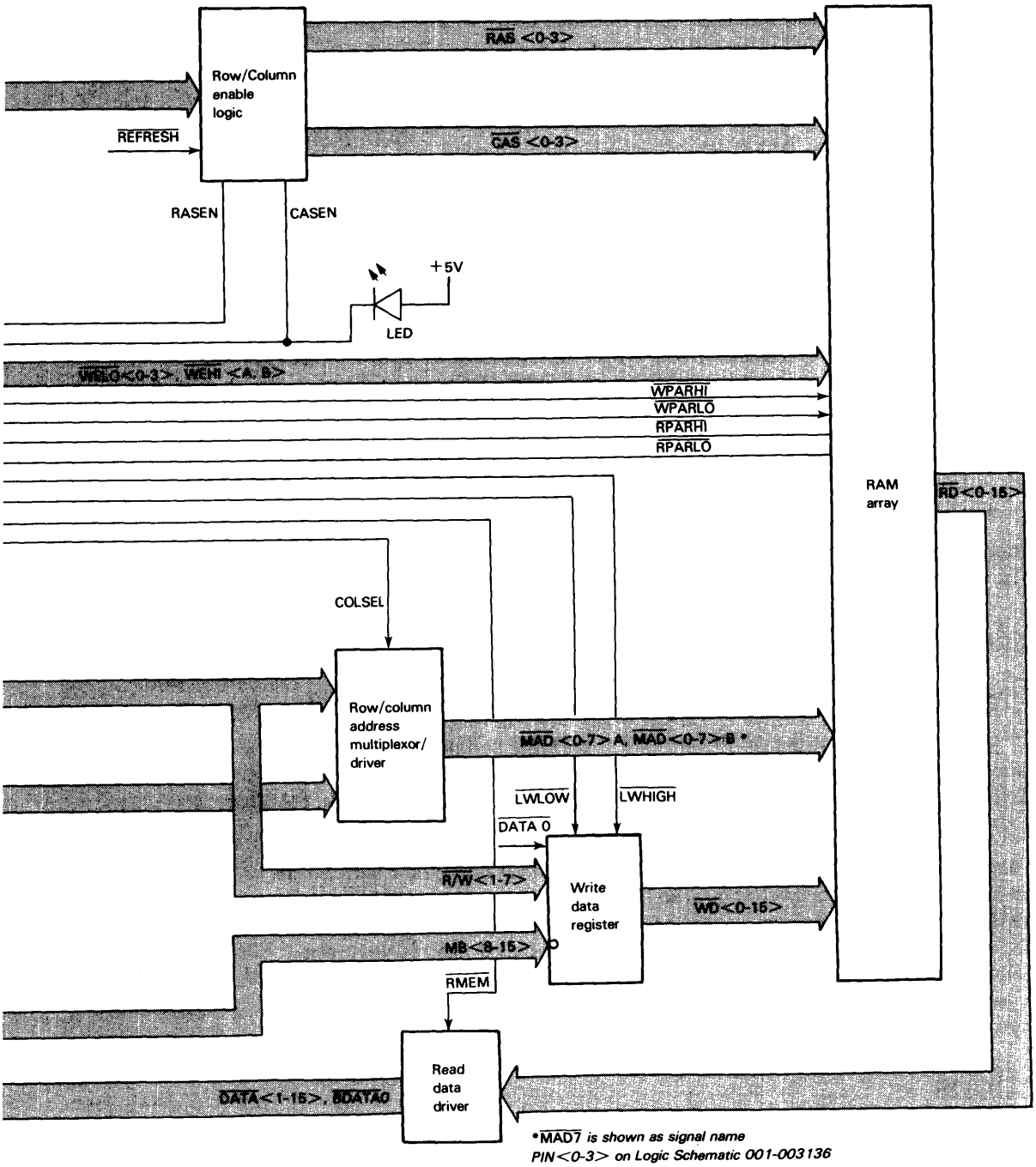
Figure 5-4 shows the interconnection of the RAM array and the other components that comprise the RAM cards, as well as the various signals that control the flow of data between the memory bus and the RAM array. The descriptions that follow relate to Data General logic schematic no. 001-003136.



Reference: Logic Schematic 001-003136

DG-08318

Figure 5-4 Memory card block diagram



*MAD7 *is shown as signal name*
PIN<0-3> on Logic Schematic 001-003136

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 MCYC and ADREN low and DATA0 high (DATA0 = 0), and sends a 20-bit physical address over the memory bus. The address consists of DATA<1-15> and XMA<0-4>. When DATA0 is high (referencing user memory space), the card/bank select logic of the memory card decodes address bits DATA1 and XMA<0-4>. If the card is jumpered 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 XMA<0-2> select the card, while signal XMA3 selects one of the two banks of RAM ICs. On 512-Kbyte memory cards, signals XMA<0-1> select the card, while signals 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 ADREN, provided that the addressed location is within the selected memory range and MCYC is low. When the controller drives ADRLATCH and R/WLATCH high in response to ADREN, the bank select logic latches the memory bank select line (SEL0, SEL1, SEL2, or SEL3); then the row and column address registers latch the row address DATA<1-8>) and the column address (XMA4 DATA<9-15>), respectively, and send them to the row/column address multiplexor. At this point, the controller begins a read or write operation.

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 SPU is read or written with each byte.

Each memory card features a light emitting diode (LED), which lights when a memory reference to an address on the card is in progress.

Row and Column Address Selection

As soon as the address multiplexor receives the address, it sends the row address bits (described above) to the RAM array via the MAD<0-7> lines. (MAD 7 is shown as signal name PIN9<0-3> on logic schematic 001-003136.) The controller then asserts RASEN. At the same time, the row/column enable logic drives the row address strobe of the selected memory bank (RAS0, RAS1, RAS2, or RAS3) low, latching the row address into the appropriate RAMs.

When the controller drives COLSEL high, the address multiplexor sends the column address bits (described above) to the RAM array via the MAD<0-7> lines. (MAD 7 is shown as signal name PIN9<0-3> on logic schematic 001-003136.) Then the controller asserts CASEN, causing the row/column enable logic to drive the column strobes of all the memory banks (CAS0, CAS1, CAS2, and CAS3) low, latching the column address into the RAM array. The assertion of CASEN also causes the LED on the memory card to light up.

Read Operations

If WH and WL are both high (unasserted) while ADREN is driven low during the address phase of the memory operation, the memory card 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 (RD<0-15>) and the two parity bits associated with the data are sent to the controller.

The controller drives 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 WH and WL lines, respectively. (Note that WH and 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.

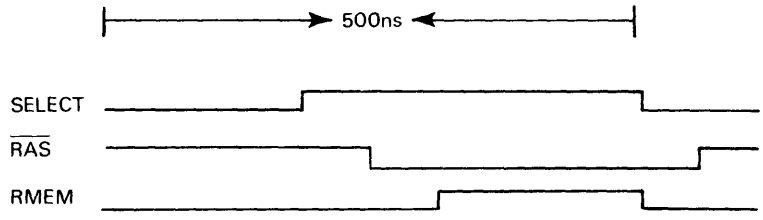


Figure 5-5 Read timing

Write Operations

If WH and/or WL are low during the address phase while ADREN is asserted, the controller initiates a byte or word write operation; signals LWHIGH and/or LWLOW are driven low, depending on the states of WH and 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 (DATA<8-15>) as MB<8-15>; the row address/data register transmits seven bits of the high-order byte DATA<1-7>) as R/W<1-7>; and the card circuitry transmits bit 0 as DATA0.

The write data register drives the appropriate byte(s) of data onto the WD<0-15> lines, inverting bits MB<8-15>. The controller transmits the parity bit(s) to the RAM array by driving signals WPARLHI and/or WPARLO high or low, depending on the states of WH and WL. The controller then issues a write enable pulse — WELO for low-order bytes and 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.

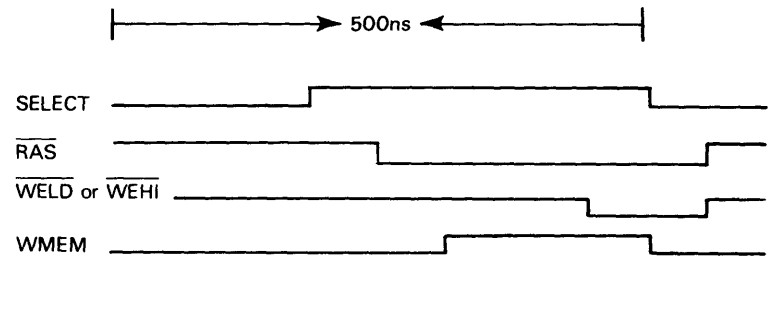


Figure 5-6 Write timing

Refresh Operations

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 $\overline{\text{REFRESH}}$, enabling all rows of the RAM array ($\overline{\text{RAS<0-3>}}$). The controller then strobes the refresh address onto $\overline{\text{MAD<0-7>}}$ ($\overline{\text{MAD 7}}$ is shown as signal name PIN<0-3> on logic schematic No. 001-0003136) 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 $\overline{\text{BRDY}}$ 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.

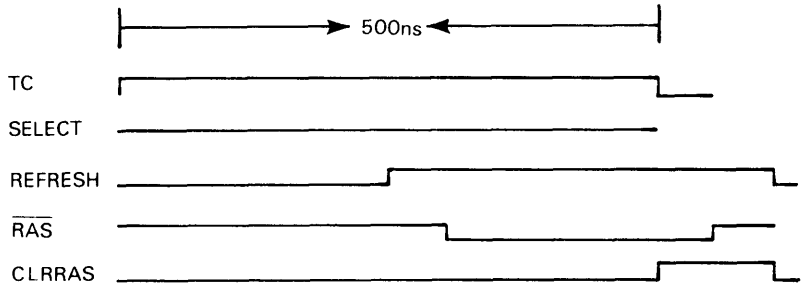


Figure 5-7 Refresh timing

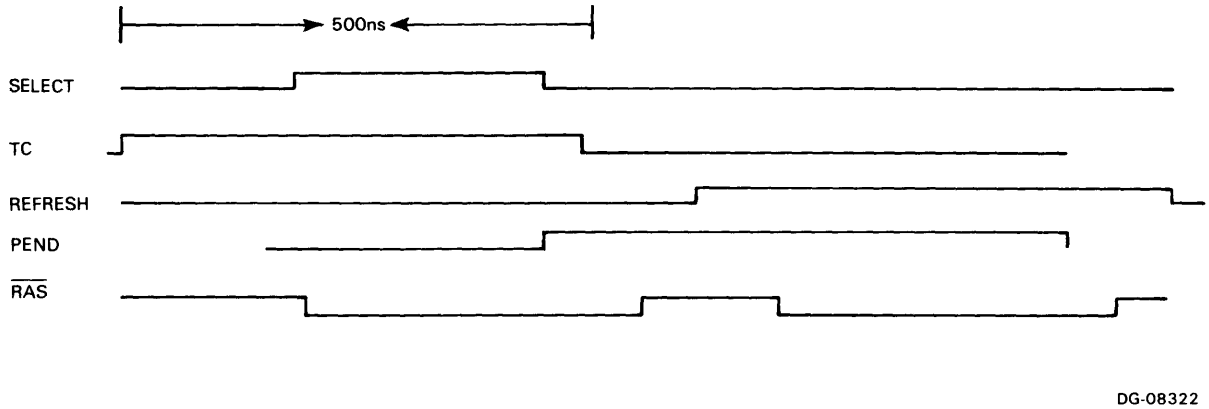


Figure 5-8 Pended refresh timing

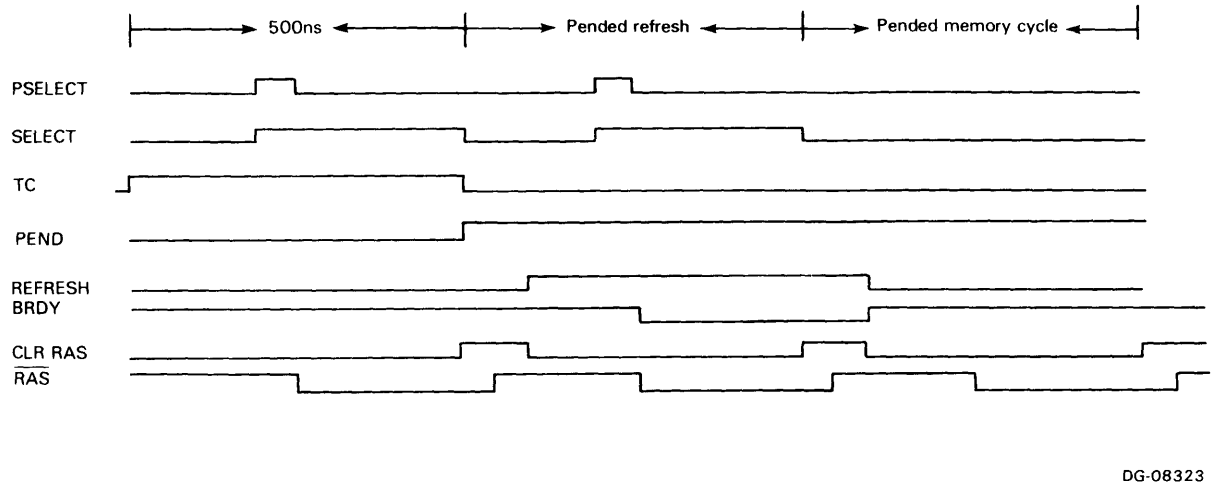


Figure 5-9 Pended memory cycle

Model 30 Hardware Floating Point

6

This chapter describes the Model 30 hardware floating point card; summarizes the instructions it executes and their execution times; describes installation and power requirements; details interconnections with the system; and describes power-up and reset response. A functional theory of operation concludes the chapter.

PH-0513

Figure 6-1 Model 30 hardware floating point card

The Model 30 hardware floating point card, shown in Figure 6-1, executes the ECLIPSE hardware floating point instruction set. This instruction set performs operations on numbers supplied by main memory or by the four, floating point accumulator (FPAC) registers. These numbers are either 4 bytes wide (single-precision) or 8 bytes wide (double-precision). Each number consists of a sign, a fractional part called the *mantissa,* and an exponent. The first byte contains the sign and exponent; the remaining bytes contain the mantissa.

A floating point external microcode controller (FPXMC) is contained on the SPU card; it initiates the operation of the hardware floating point option and sends

microcode to the CPU. The microcode commands the CPU to control all data transfers between itself, the Model 30 hardware floating point card, and main memory. Under the FPXMC microcode control, the CPU synchronizes the operations of the Model 30 hardware floating point card and memory to transfer a number whenever a floating point instruction stores, or uses a number stored in memory. In like manner, when a floating point instruction stores or uses data in a CPU accumulator, the CPU, under the control of microcode from the FPXMC, synchronizes its operation with that of the Model 30 hardware floating point card to transfer the data.

Model 30 Hardware Floating Point Instructions

The Model 30 hardware floating point card executes the entire ECLIPSE hardware floating point floating point instruction set. These instructions are listed in Table 6-1 and their execution times by the Model 30 hardware floating point card are listed in Table 6-2. Refer to the *16-bit Real-Time* ECLIPSE *Assembly Language Programming* for complete descriptions of these instructions and their use.

Note that several instructions have two forms; one ending in *S* and another in *D*. The first form uses single-precision floating point format, while the second uses double-precision floating point format. The function of the two forms is otherwise identical.

Table 6-1 Instructions executed by the Model 20 and Model 30 hardware floating point card

Mnem	Instruction	Action
FAB	Absolute Value	Sets the sign bit of a floating point accumulator (FPAC) to 0.
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.
FCLE	Clear Errors	Sets bits 0-4 of the floating point status register (FPSR) to 0.
FCMP	Compare Floating Point	Compares two floating point numbers and sets the zero (Z) and negative result (N) flags in the FPSR accordingly.
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.
FEXP	Load Exponent	Places bits 1-7 of AC0 in bits 1-7 of the specified FPAC.
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.

Table 6-1 Instructions executed by the Model 20 and Model 30 hardware floating point card (Continued)

Mnem	Instruction	Action
FHLV	Halve	Divides the floating point number in FPAC by 2.
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.
FLDS, FLDD	Load Floating Point	Copies a floating point number from memory to a specified FPAC.
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.
FLST	Load Floating Point Status	Copies the contents of two specified memory locations to the FPSR.
FMMS, FMMD	Multiply Memory by FPAC	Multiplies the floating point number in memory by the floating point number in an FPAC.
FMOV	Move Floating Point	Moves the contents of one FPAC to another FPAC.
FMS, FMD	Multiply (FPAC by FPAC)	Multiplies the floating point number in one FPAC by the floating point number in another FPAC.
FNEG	Negate	Inverts the sign bit of the FPAC.
FNOM	Normalize	Normalizes the floating point number in FPAC.
FNS	No Skip	No operation.
FPOP	Pop Floating Point State	Pops an 18-word floating point 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 block onto the user stack.
FRH	Read High Word	Places the high-order 16 bits of an FPAC in AC0.
FSA	Skip Always	Skips the next sequential word.
FSCAL	Scale	Shifts the mantissa of the floating point number in FPAC either right or left, depending on the contents of bits 1-7 of AC0.
FSEQ	Skip On Zero	Skips the next sequential word if the Z flag of the FPSR is 1.
FSGE	Skip On Greater Than Or Equal To Zero	Skips the next sequential word if the N flag of the FPSR is 0.
FSGT	Skip On Greater than Zero	Skips the next sequential word if both the Z and N flags of the FPSR are 0.
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 1.
FSLT	Skip On Less Than Zero	Skips the next sequential word if the N flag of the FPSR is 1.
FSMS, FSMD	Subtract (memory from FPAC)	Subtracts the floating point number in memory from the floating point number in an FPAC.
FSND	Skip On No Zero Divide	Skips the next sequential word if the divide by zero (DVZ) flag of the FPSR is 0.
FSNE	Skip On Non-Zero	Skips the next sequential word if the Z flag of the FPSR is 0.

Table 6-1 Instructions executed by the Model 20 and Model 30 hardware floating point card (Continued)

Mnem	Instruction	Action
FSNER	Skip On No Error	Skips the next sequential word if bits 1-4 of the FPSR are all 0.
FSNM	Skip On No Mantissa Overflow	Skips the next sequential word if the mantissa overflow (MOF) flag of the FPSR is 0.
FSNO	Skip On No Overflow	Skips the next sequential word if the overflow (OVF) flag of the FPSR is 0.
FSNOD	Skip On No Overflow And No Zero Divide	Skips the next sequential word if the OVF and the divide by DVZ flags of the FPSR are 0.
FSNU	Skip On No Overflow	Skips the next sequential word if the underflow (UNF) flag of the FPSR is 0.
FSNUD	Skip On No Underflow And No Zero Divide	Skips the next sequential word if the UNF and the divide by DVZ flags of the FPSR are 0.
FSNUO	Skip On No Underflow And No Overflow	Skips the next sequential word if the UNF and OVF flags of the FPSR are 0.
FSS, FSD	Subtract (FPAC from FPAC)	Subtracts the floating point number in one FPAC from the floating point number in another FPAC.
FSST	Store Floating Point Status	Copies the contents of the FPSR to two memory locations.
FSTS, FSTD	Store Floating Point	Copies the contents of a specified FPAC into memory.
FTD	Trap Disable	Sets the trap enable flag of the FPSR to 0.
FTE	Trap Enable	Sets the trap enable flag of the FPSR to 1.

Table 6-2 Instruction execution times

Instruction	Execution time (us)	Notes
FAB	7.00	1
FAD	12.50	1
FAMD	18.00	1,2
FAMS	15.50	1,2
FAS	11.50	1
FCLE	11.50	
FCMP	4.50	
FDD	42.75	1,3
FDMD	48.25	1,2,3
FDMS	20.50	1,2,3
FDS	16.75	1,3
FEXP	9.25	1
FFAS	11.00	1
FFMD	12.50	1,2
FHLV	9.00	1,3
FINT	19.50	1
FLAS	10.00	
FLDD	10.00	2
FLDS	7.50	2
FLMD	13.50	2
FLST	11.00	1,2

Table 6-2 Instruction execution times (Continued)

Instruction	Execution time (us)	Notes
FMD	46.00	1
FMMD	51.50	1,2
FMMS	22.00	1,2
FMOV	4.50	
FMS	18.50	1
FNEG	7.50	1
FNOM	11.50	1
FNS	4.00	
FPOP	43.00	
FPSH	38.00	
FRH	4.00	
FSA	5.00	
FSCAL	15.50	1
FSD	12.75	1
FSEQ	4.50	
FSGE	4.50	
FSGT	4.50	
FSLE	4.50	
FSLT	4.50	
FSMD	18.25	1,2
FSMS	15.50	1,2
FSND	4.50	
FSNE	4.50	
FSNER	4.50	
FSNM	4.50	
FSNO	4.50	
FSNOD	4.50	
FSNU	4.50	
FSNUD	4.50	
FSNUO	4.50	
FSS	12.00	1
FSST	7.00	2
FSTD	5.50	2
FSTS	4.00	2
FTD	1.50	
FTE	3.75	1

[1] *This instruction can take a floating point trap, which will add 17 microseconds to its execution time.*

[2] *This instruction does an effective address calculation, which may add 15 microseconds to its execution time.*

[3] *Floating point divide execution times depend on the number of zero and one bits in the quotient (the more one's, the longer the time.*

Installation and Power Requirements

The Model 30 hardware floating point unit card must be inserted into slot 2 of the Model 30 CPU logic module (CLM) cage. An interconnection cable to the system processor unit card is also required (see "Interfacing" below). No jumpering or other tailoring is required.

Power Requirements

The Model 30 hardware floating point card requires only one voltage level: + 5 V. The maximum current requirement at this voltage level is 4.6 A, for a maximum power dissipation of 23 W. The voltage is supplied to the card via B connector pins 57, 59, and 60.

Interfacing

The FPU card communicates with the system processor unit (SPU) card and memory card(s) through its B and C edge connectors. The B connector connects to printed circuit wiring contained on the CLM backpanel through a backpanel connector. The C connector connects to the SPU card through an interconnection cable that plugs onto the C connector of both the SPU and the FPU card. Figure 6-2 shows the connector positions of the FPU card.

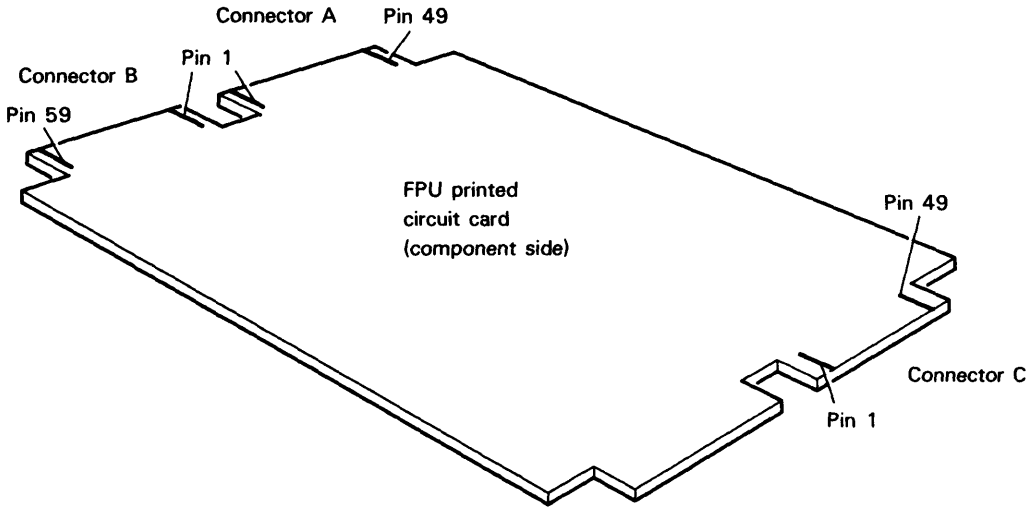


Figure 6-2 FPU card connector positions

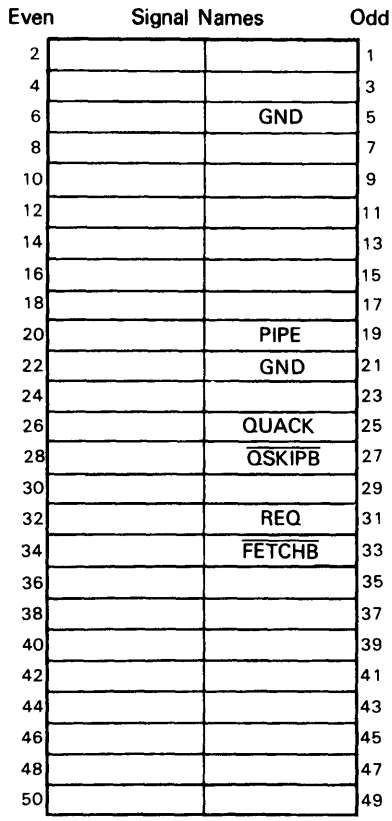
The 60-pin B connector supplies data, power, and memory bus control signals to and from the Model 30 hardware floating point card. Figure 6-3 shows the B-connector pin assignments.

Even	Signal Names		Odd
60	+5V	+5V	59
58		+5V	57
56			55
54	GND	GND	53
52	SYSCLKB		51
50	DATA00B	DATA08B	49
48	ADRENB	DATEN8	47
46	DATA01B	DATA09B	45
44			43
42	DATA02B	DATA10B	41
40			39
38	DATA03B	DATA11B	37
36	GND		35
34	DATA04B	DATA12B	33
32	DATA05B	DATA13B	31
30	WHIGHB		29
28	DATA06B	DATA14B	27
26	MEMCYC		25
24	DATA07B	DATA15B	23
22			21
20			19
18			17
16			15
14	GND		13
12		GND	11
10			9
8			7
6	CLEAR		5
4		GND	3
2			1

ID-00627

Figure 6-3 Pin assignments, B connector

The 50-pin C connector carries protocol signals for synchronizing floating point operations between the SPU and the Model 30 hardware floating point card. Figure 6-4 shows the C-connector pin assignments.



DG-08919

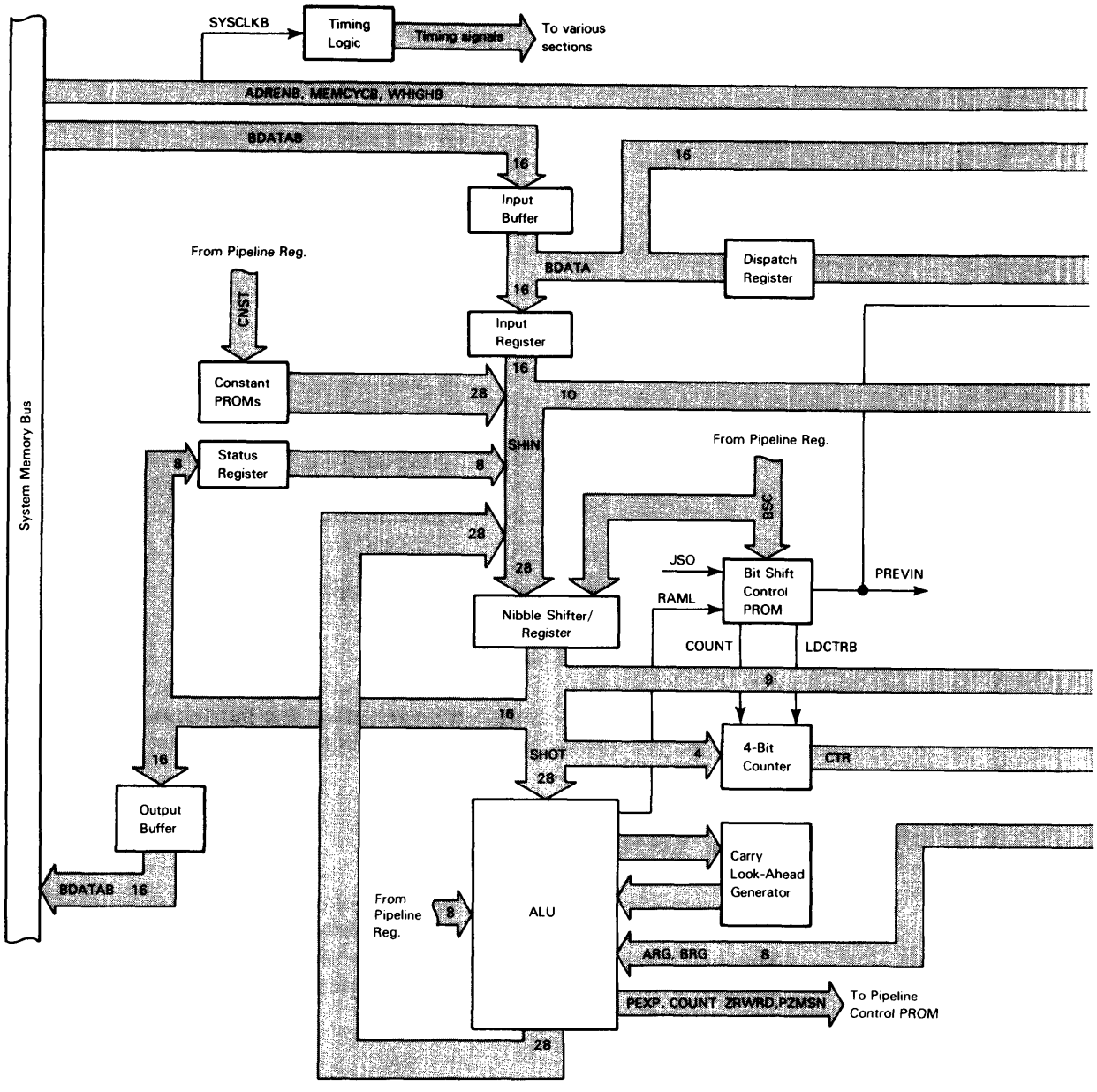
Figure 6-4 Pin assignments, C connector

Power-up/Reset Response

When the Model 30 system is powered up, the hardware floating point option enters an idle loop. The card also enters the idle loop after the execution of any jump instruction.

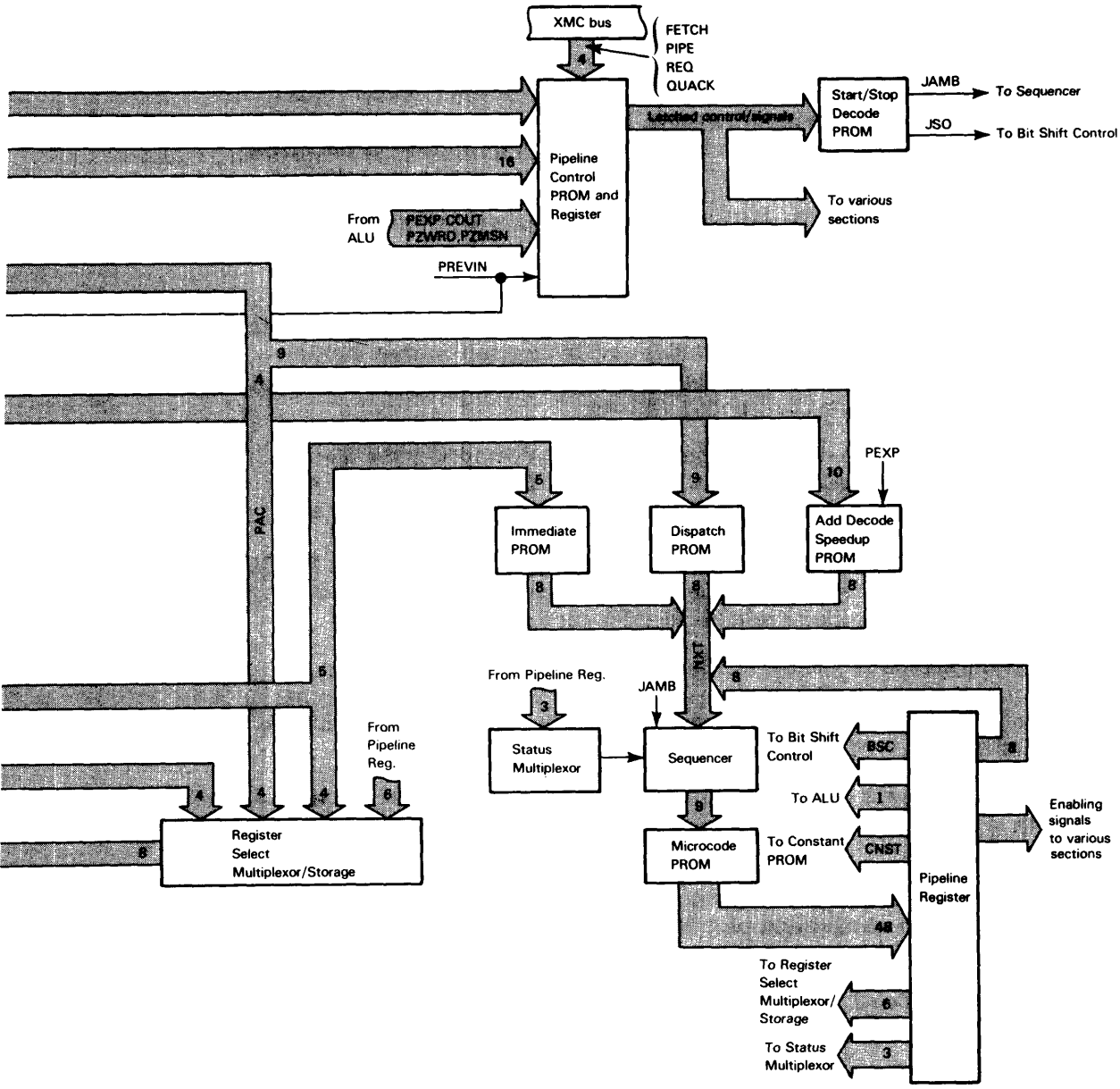
Theory of Operation

The organization of the hardware floating point card is shown in Figure 6-5. For the purpose of discussion, the card may be thought of as comprising three sections: the timing/control section, the data manipulation section, and the address/instruction section. The description that follows refers to Data General logic schematic No. 001-002863.



DG-08920

Figure 6-5 Model 30 hardware floating point unit block diagram


XMC bus
4
FETCH
PIPE
REQ
QUACK
Pipeline Control PROM and Register
16
From ALU
PEXP COUT PZWRO PZMSN
PREVIN
Latched control signals
Start/Stop Decode PROM
JAMB
To Sequencer
JSO
To Bit Shift Control
To various sections
9
4
PAC
5
6
9
10
PEXP
Immediate PROM
Dispatch PROM
Add Decode Speedup PROM
8
8
8
NXT
8
From Pipeline Reg.
3
JAMB
Status Multiplexor
Sequencer
To Bit Shift Control
BSC
8
To ALU
1
9
Microcode PROM
To Constant PROM
CNST
Enabling signals to various sections
Pipeline Register
48
To Register Select Multiplexor/ Storage
6
To Status Multiplexor
3
From Pipeline Reg.
6
4
4
4
8
Register Select Multiplexor/Storage

Timing/Control

The timing/control circuitry consists of the timing logic, the pipeline control PROM and latch, and the start/stop decode PROM. Together these elements produce the timing, synchronization, and control signals required to coordinate operation of the floating point card with the system memory address bus and the combination of the mE670/FPXMC.

Data Paths and Data Manipulation

The input buffer passes data and instructions from the system memory bus to the dispatch register and the input register. The input register is loaded during each CPU data phase. When enabled, it passes the data it contains to the nibble shifter/register. (The dispatch register is discussed in the section to come.)

The add decode speedup PROM dispatches an 8-bit, microprogram address to the microsequencer in order to speed up add/subtract prealignment and compare operations.

The nibble shifter/register can perform nibble (4-bit) shifts one place to the left, or one or two places to the right. It includes an 8-bit register so that multiple-word shifts can be performed for all operations. The nibble shifter is controlled by signals BSC<0,2> from the pipeline register.

The (possibly) shifted data output by the nibble shifter comprises the 28-bit input to the arithmetic/logic unit (ALU). The ALU consists of seven 2901 4-bit bipolar microprocessor slices. These as a unit split the 64-bit contents of a floating point accumulator (FPAC) into two 28-bit mantissas and one 8-bit sign/exponent. The addresses (0, 1, 2, or 3) of the source and destination FPACs are stored in the register select multiplexor/storage circuits, whose output determines which ALU register is to be used for each operation.

The ALU performs the sequences of operations needed to implement the floating point instruction set under the control of instructions received from the microcode PROM via the pipeline register. It does this in coordination with the operation of the nibble shifter.

The coordination of shift outputs and inputs, necessary for multiply and divide operations, is performed by the bit shift control PROM. The bit shift control PROM is addressed by BSC<0-2> from the pipeline register.

The shift control PROM works in conjunction with a 4-bit counter that keeps track of multiply and divide iterations. The counter also allows direct specification of one of the registers in the ALU.

The data output of the ALU passes through the nibble shifter and output buffer to the system memory bus.

In addition to data from the input register, the nibble shifter sometimes receives 28-bit constants from the constant PROMs. Two 32 × 8 PROMs provide the constants, which are addressed by signals CNST<0-6> from the pipeline register. The nibble shifter also receives and passes to the output buffer the 8-bit floating point status word from the status register, when that register's output is enabled.

Address and Instruction Paths

Just as the ALU is central to the data manipulation section, the microcode PROM

is the heart of the address/instruction section. The 512x48-bit microcode PROM contains the following items.

instructions for the ALU

ALU register select controls

ALU data source select controls

microcode branch addresses

addresses of the constants in the — constant PROMs

control signals for the sequencer

bit shift control signals

branch address source control signals

miscellaneous control signals

These items are clocked into the 48-bit-wide pipeline register at the beginning of each phase of floating point operation. The items latched are those addressed by the 8-bit word produced by the sequencer. The sequencer's output depends on the input it receives from the following sources.

the pipeline register

the dispatch PROM

the add decode speedup PROM

the immediate PROM

The dispatch PROM provides a dispatch for all floating point instructions based on bits 1-9 of an instruction word. It receives this word from the dispatch register.

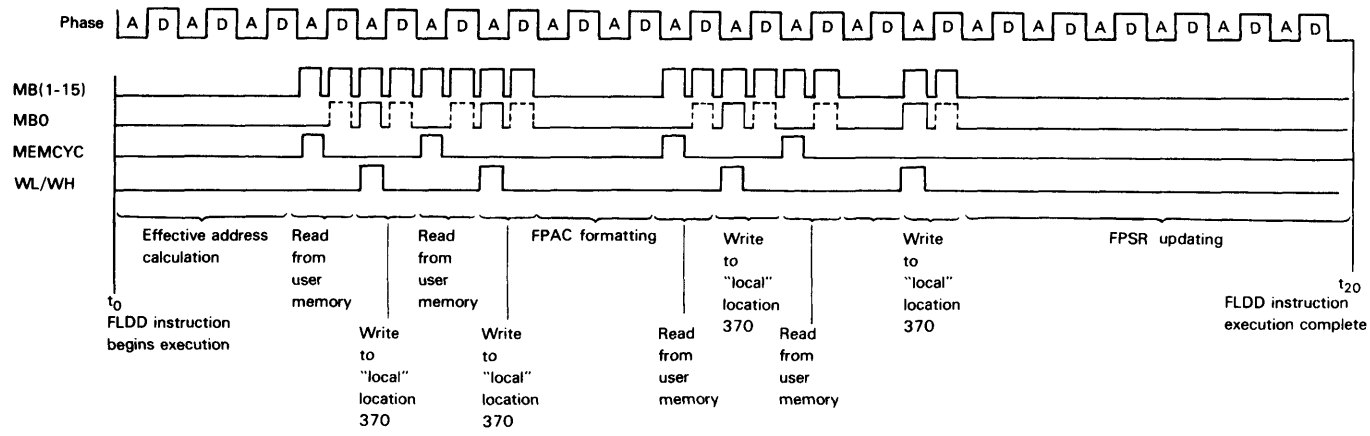
The immediate PROM affords 32 entry points for immediate instruction executions. It is addressed by bits SHOT<00-04> from the nibble shifter.

The status multiplexor selects the state of one of several conditions as an input to the next address selection of the sequencer. These conditions pertain to the results of ALU operations (for example, zero, carry) or to the state of the FPACs (recently read or written).

Floating Point Operations and the System Memory Bus

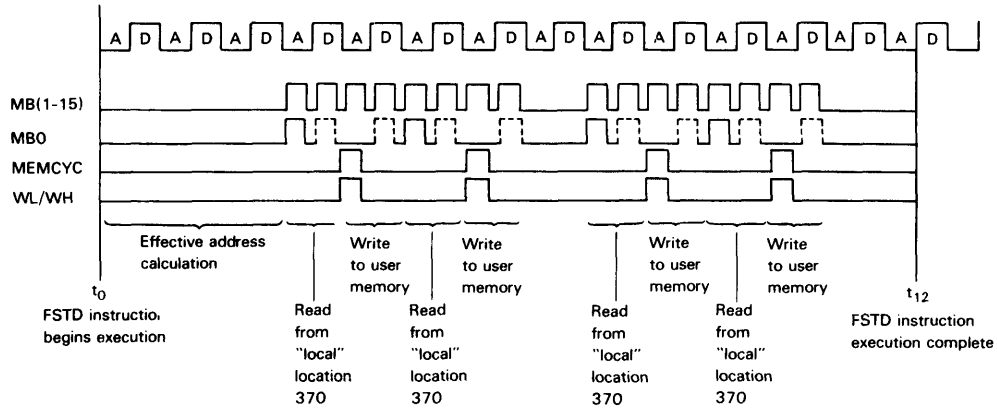
Figure 6-6 illustrates the timing of a floating point load double operation. A sequence of events such as that shown in this figure occur whenever a floating point operation must use data from main memory.

Figure 6-7 illustrates the timing of a floating point store double operation. The load operation requires more cycles per (16-bit) word than does the store operation because the load operation, as shown in the figure, must wait for the floating point status register to be updated. Extra microcycles are also required to shift each half of the 56-bit mantissa into the 28-bit ALU. (Local location 370, referred to in the figures, is used for transparent interdevice communication. When this location is accessed, address bit 0 — ADR0 — is 1 and MEMCYC is low, while ADR<1-15> are 000 000 011 111 000.)



DG-08946

Figure 6-6 Floating point load double timing diagram



DG-08921

Figure 6-7 Floating point store double timing diagram

Diskette Subsystem

7

The Model 20 and 30 diskette subsystem provides up to 737 Kbytes of formatted storage for the computer system. The subsystem consists of a diskette interface card and one or two diskette drives housed in the diskette module (FM). Each diskette drive accepts one removable, double-density, double-sided diskette. The interface connects to the microI/O bus and transfers data between the host's memory and subsystem via the data channel facility at an average rate of 23 Kbytes per second.

This chapter describes the diskette subsystem and media, its interconnections with the system, and its functional theory of operation.

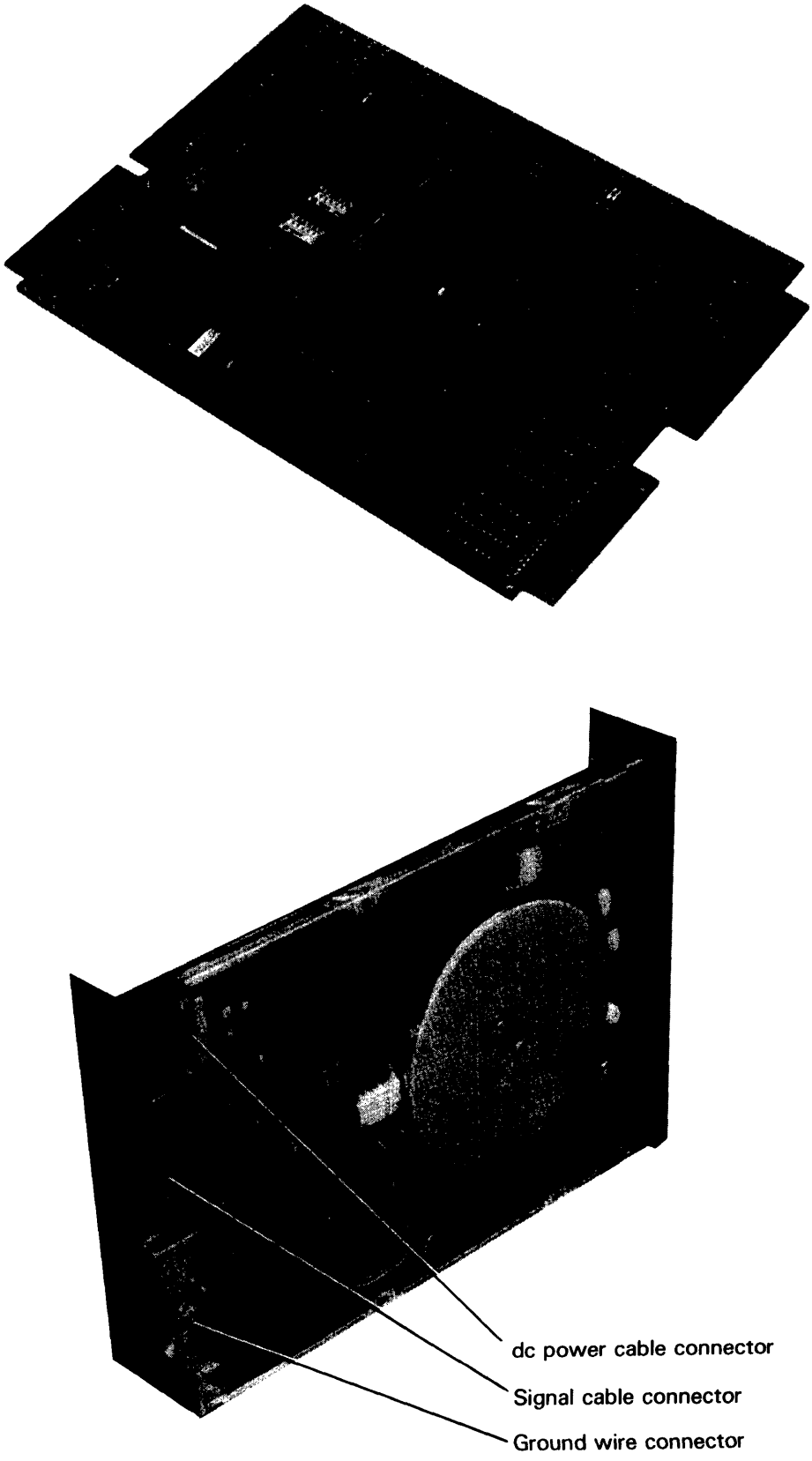


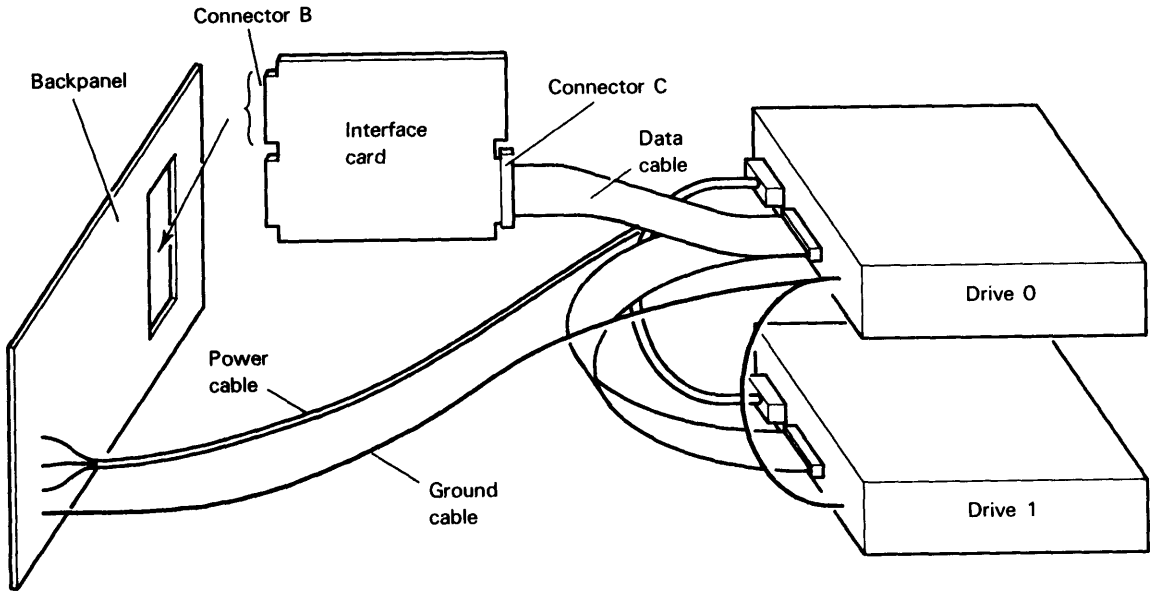
Figure 7-1 Model 20 and Model 30 diskette interface card and drive

Subsystem Overview

This section provides an overview of the diskette subsystem: the interface card (shown in Figure 7-1), diskette media, the initial program load feature (IPL), and, in general terms, power-up self-test.

Diskette Interface

The diskette interface card connects to the microI/O bus of the SPU through one of its printed circuit card connectors that plugs into a mating connector located on the FM backpanel. A daisy chain ribbon cable connects the drive(s) to the interface. Each drive is assigned a unique address, which is selectable on the drive unit, allowing the interface to select individual drives for seek, data transfers, and status operations. A second daisy chain, the power harness, connects the drive(s) to power carried on the FM backpanel. Figure 7-2 shows the diskette subsystem connections.



ID-00628

Figure 7-2 Diskette subsystem connections

The diskette interface contains four major integrated circuits (ICs):

1. 8-bit microprocessor CPU,
2. Byte-oriented, diskette drive controller (FDC),
3. Data channel controller (DCH),

4. MicroI/O bus controller (IOC).

The *microprocessor* supervises the major portion of the diskette interface and controls the positioning of the diskette drive read/write heads. The *drive controller* handles the read/write operations of the interface. The FDC also performs extensive error checks, including address checking and cyclic redundancy checks (CRC) on the recorded data. The *data channel controller* uses the data channel facility of the microI/O bus to handle data transfer activities between memory and the diskette controller, on a demand mode basis: a data channel transfer occurs each time the diskette controller requires a 16-bit data word transfer. The *microI/O bus controller* acts as an interface between the diskette interface and the microI/O bus.

Transfers to and from the diskette are double-word buffered to improve data channel latency. Up to 18 contiguous sectors (16 in IBM PC format), available at a particular head position, can be transferred by one command. (When a Data General MPT/100 formatted diskette is inserted in the drive, up to 20 contiguous sectors can be read by one command.) A multiple-sector transfer operation beginning on head zero can continue on head one.

Diskette Media

A diskette has two surfaces, each providing 40 recording tracks. The top surface is labeled 0. The same track position of the upper and lower surface constitutes a diskette cylinder. Diskettes can be formatted in either Data General standard or IBM PC format. Each track, when properly formatted, contains either nine standard Data General or eight IBM PC recording sectors.

In addition, the diskette subsystem can be programmed to read Data General MPT/100 formatted diskettes. These diskettes provide 35 recording tracks per surface and contain 10 sectors per track.

Surface zero of standard Data General and IBM PC formatted diskettes is located on the left side of the diskette when installed in the 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.

Diskette tracks are *soft-sectored*, meaning that there are no physical reference points to identify the sector boundaries; thus sector boundaries must be recorded. Recording of sector boundaries is performed by formatting each track into sectors. Each formatted sector contains an address field that allows the subsystem to locate it. Diskette formats and the formatting operation are detailed in Chapter 2.

Initial Program Load (IPL)

The diskette subsystem provides an Initial Program Load (IPL) feature that can transfer 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 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. The IPL feature is fully detailed in Chapter 2.

Page 7-5

Under page heading, "Installation and Tailoring",
Add the following note after the first paragraph:

NOTE *These switches are set when installed in the system and should not be changed.*

Page 7-10

Figure 7-5 Pin assignments, B-connector

Change pin 57 from −5 VDC to +5 VDC

Page 7-13

Table 7-2 Drive interface signal descriptions

Delete the up-arrow (^) symbol before each signal name in the table and place a bar over each signal name.

In the **Description** column, change each occurrence of "when 1" to "when low".

In the **Description** column, change each occurance of "when 0" to "when high".

Page 7-17

Under page heading, "Byte Swap Network",
Change the entire description of this network to read:

This network can be enabled to swap high- and low-order bytes of 16-bit words between memory and the DCH during data channel activity. This network must be enabled when reading from or writing to diskettes in IBM PC format. This format writes or reads the low-order byte of a 16-bit word first.

Power-Up Self-Test

When the Model 20 or 30 system is powered-up, the diskette interface logic is automatically initialized, the diskette drive operating modes are set to their default values, and the diskette interface card performs a short self-test. If the test is completed successfully, the interface idles and waits for a command. If the test is not successful, a diagnostic light-emitting-diode (LED) on the interface card turns on and the NOT OK flag bit is set to 1 in the interface's status register. The LED will turn off and the NOT OK flag will set to 0 only after the interface is reset and passes the self-test. (Refer to "Power-Up Response" in the diskette subsystem section of Chapter 2 for interface initialization and diskette drive operating mode default values.)

Installation and Tailoring

The diskette interface card inserts into the I/O slot of the diskette module. An interconnection cable to the diskette drive(s) is also required, as explained later in this section. The card contains two sets of dual-in-line-package (DIP) switches that select diskette drive characteristics and the interface's device code. Refer to the installation manual for your system for details on function, location, and tailoring of these switches.

Table 7-1 gives the power requirements of the diskette interface card.

Interfacing

The diskette interface card receives power and communicates with the SPU card through its B edge connector. It communicates with the diskette drive(s) through its C edge connector. The B connector joins with printed circuit wiring on the FM backpanel through a backpanel connector. The backpanel connects, through adjacent module backpanel interconnectors, to the SPU card in the CPU logic module and to power supply 1 in the power module (PM). The C connector joins the diskette drive(s) through an interconnection cable that plugs onto a connector located on the rear of the drive. This cable connects in daisy chain fashion to the second drive, when present.

Figure 7-3 shows the connector positions of the diskette interface card. Diskette subsystem interconnections are illustrated in Figure 7-4. Figure 7-5 and Figure 7-6 show the pin assignments of the B and C connectors, respectively. Table 7-2 describes the drive interface signals.

Power to the diskette drive(s) is provided through a dc power cable soldered to the FM backpanel. This power cable connects, in daisy chain fashion, to the second drive, when present. Table 7-3 lists the power requirements for each diskette drive.

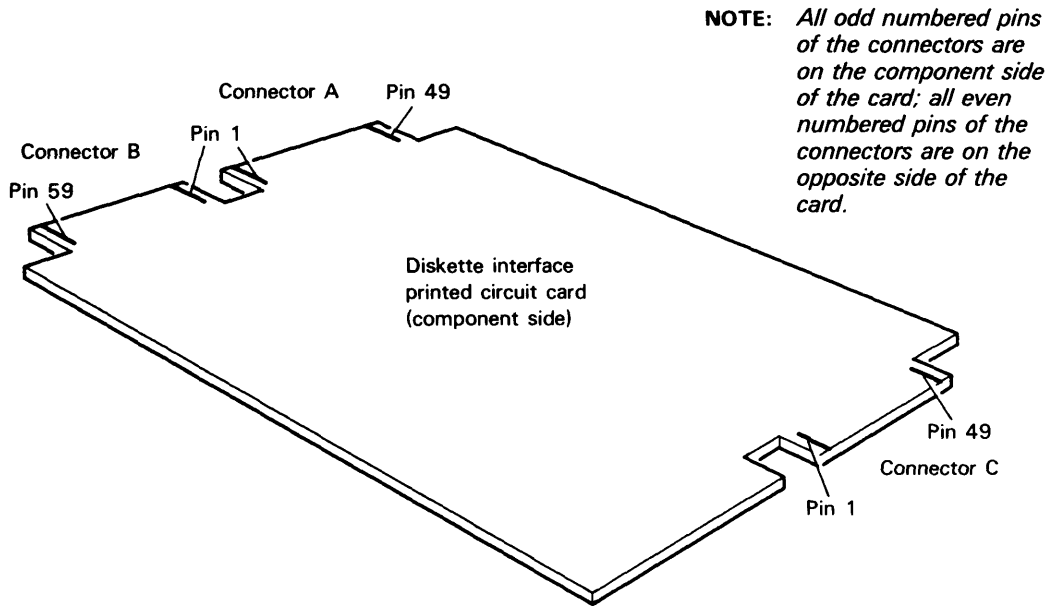


Figure 7-3 Diskette interface card connector positions

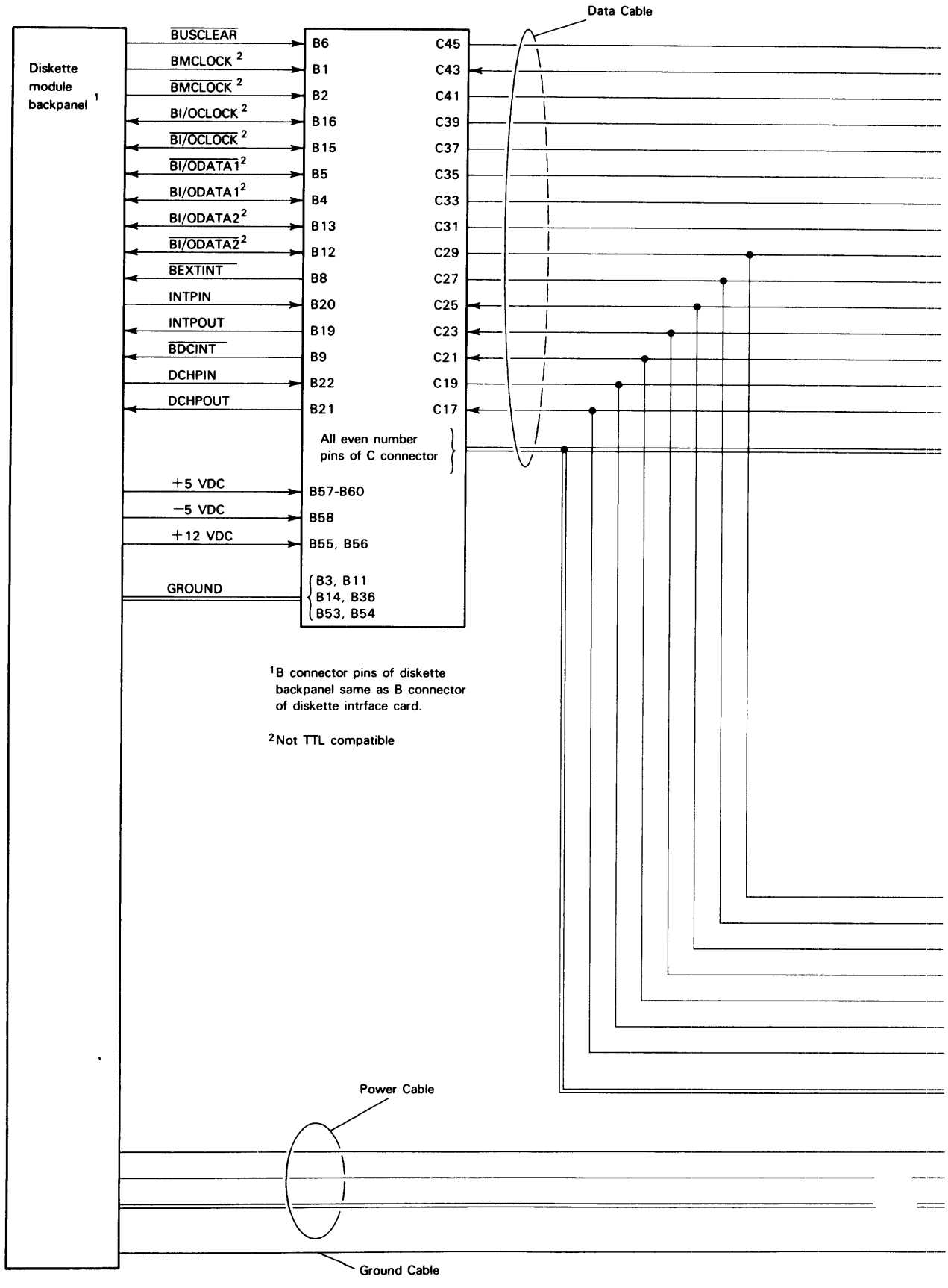
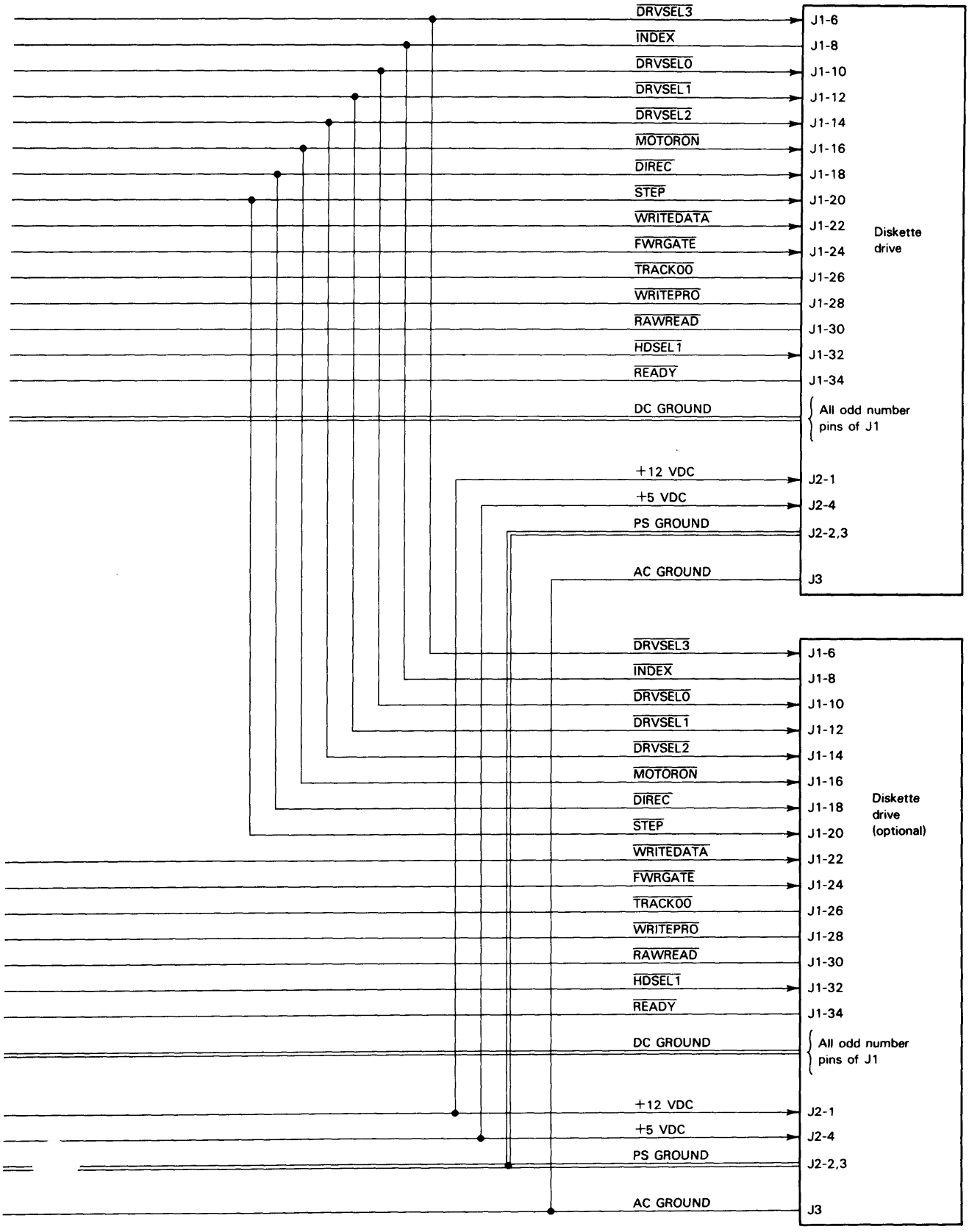


Figure 7-4 Diskette subsystem interconnections


DRVSEL3
INDEX
DRVSEL0
DRVSEL1
DRVSEL2
MOTORON
DIREC
STEP
WRITEDATA
FWRGATE
TRACK00
WRITEPRO
RAWREAD
HDSEL1
READY
DC GROUND
J1-6
J1-8
J1-10
J1-12
J1-14
J1-16
J1-18
J1-20
J1-22
J1-24
J1-26
J1-28
J1-30
J1-32
J1-34
All odd number pins of J1
Diskette drive
+12 VDC
+5 VDC
PS GROUND
AC GROUND
J2-1
J2-4
J2-2,3
J3
DRVSEL3
INDEX
DRVSEL0
DRVSEL1
DRVSEL2
MOTORON
DIREC
STEP
WRITEDATA
FWRGATE
TRACK00
WRITEPRO
RAWREAD
HDSEL1
READY
DC GROUND
J1-6
J1-8
J1-10
J1-12
J1-14
J1-16
J1-18
J1-20
J1-22
J1-24
J1-26
J1-28
J1-30
J1-32
J1-34
All odd number pins of J1
Diskette drive (optional)
+12 VDC
+5 VDC
PS GROUND
AC GROUND
J2-1
J2-4
J2-2,3
J3

Even	Signal Names		Odd
60	+5 VDC	+5 VDC	59
58	−5 VDC	−5 VDC	57
56	+12 VDC	+12 VDC	55
54	GROUND	GROUND	53
52			51
50			49
48			47
46			45
44			43
42			41
40			39
38			37
36	GROUND		35
34			33
32			31
30			29
28			27
26			25
24			23
22	DCHPIN	DCHPOUT	21
20	INTPIN	INTPOUT	19
18			17
16	BI/OCLOCK*	$\overline{\text{BI/OCLOCK*}}$	15
14	GROUND	BI/ODATA2*	13
12	$\overline{\text{BI/ODATA2*}}$	GROUND	11
10		$\overline{\text{BCDINT}}$	9
8	$\overline{\text{BEXTINT}}$		7
6	$\overline{\text{BUSCLEAR}}$	$\overline{\text{BI/ODATA1*}}$	5
4	BI/ODATA1*	GROUND	3
2	$\overline{\text{BMCLOCK*}}$	BMCLOCK*	1

NOTE **Not TTL compatible.*
Blank pins are not used.

ID-00631

Figure 7-5 Pin assignments, B connector

Even	Signal Names		Odd
2	GROUND		1
4	GROUND		3
6	GROUND		5
8	GROUND		7
10	GROUND		9
12	GROUND		11
14	GROUND		13
16	GROUND		15
18	GROUND	READY	17
20	GROUND	HDSEL1	19
22	GROUND	RAWREAD	21
24	GROUND	WRITEPRU	23
26	GROUND	TRACK00	25
28	GROUND	FWRGATE	27
30	GROUND	WRITEDATA	29
32	GROUND	STEP	31
34	GROUND	DIREC	33
36	GROUND	MOTORON	35
38	GROUND	DRVSEL2	37
40	GROUND	DRVSEL1	39
42	GROUND	DRVSEL0	41
44	GROUND	INDEX	43
46	GROUND	DRVSEL3	45
48	GROUND		47
50	GROUND		49

NOTES *Blank pins are not used*

ID-00632

Figure 7-6 Pin assignments, C connector

Table 7-1 Power requirements for interface card

Supply Voltage	Current Draw (max)	Power Dissipation	Pin Numbers B Connector
+5 Vdc	1.50 amps	7.5 W	57, 59, 60
−5 Vdc	0.025 amps	0.125 W	58
+12 Vdc	0.025 amps	0.3 W	55, 56
Ground	—		3, 11, 14, 36, 53, 54

Table 7-2 Drive interface signal descriptions

Signal	Description
^MOTORON	When 1, activates the spindle motor of all drives connected to the subsystem.
^WRITEPRO	When 1, indicates the diskette contained in the selected drive is write-protected.
^DRVSEL<0-1>	When 1, selects the appropriate drive to respond to the interface.
^READY	When 1, indicates the selected drive is ready to accept commands.
^TRACK00	When 1, indicates the selected drive's read/write heads are positioned at track 00 (the outermost track).
^INDEX	When 1, indicates the index hole of the diskette in the selected drive is being sensed.
^DIREC	Defines the direction of read/write head motion when the ^STEP signal line is pulsed; when 1, specifies the head is to move towards the center of the diskette (in), when 0, specifies the head is to move away from the center of the diskette (out).
^STEP	On the signal transition from 1 to 0, causes the read/write head to move one track position in the direction defined by the ^DIREC signal line.
^HDSEL1	When 1, selects read/write head of surface 1 to be used for reading or writing.
^RAWREAD	The serial read data stream to the interface.
^FWRGATE	When 1, enables data is to be recorded on the diskette.
^WRITEDATA	The serial write data stream to the diskette.

Table 7-3 Power requirements for diskette drives

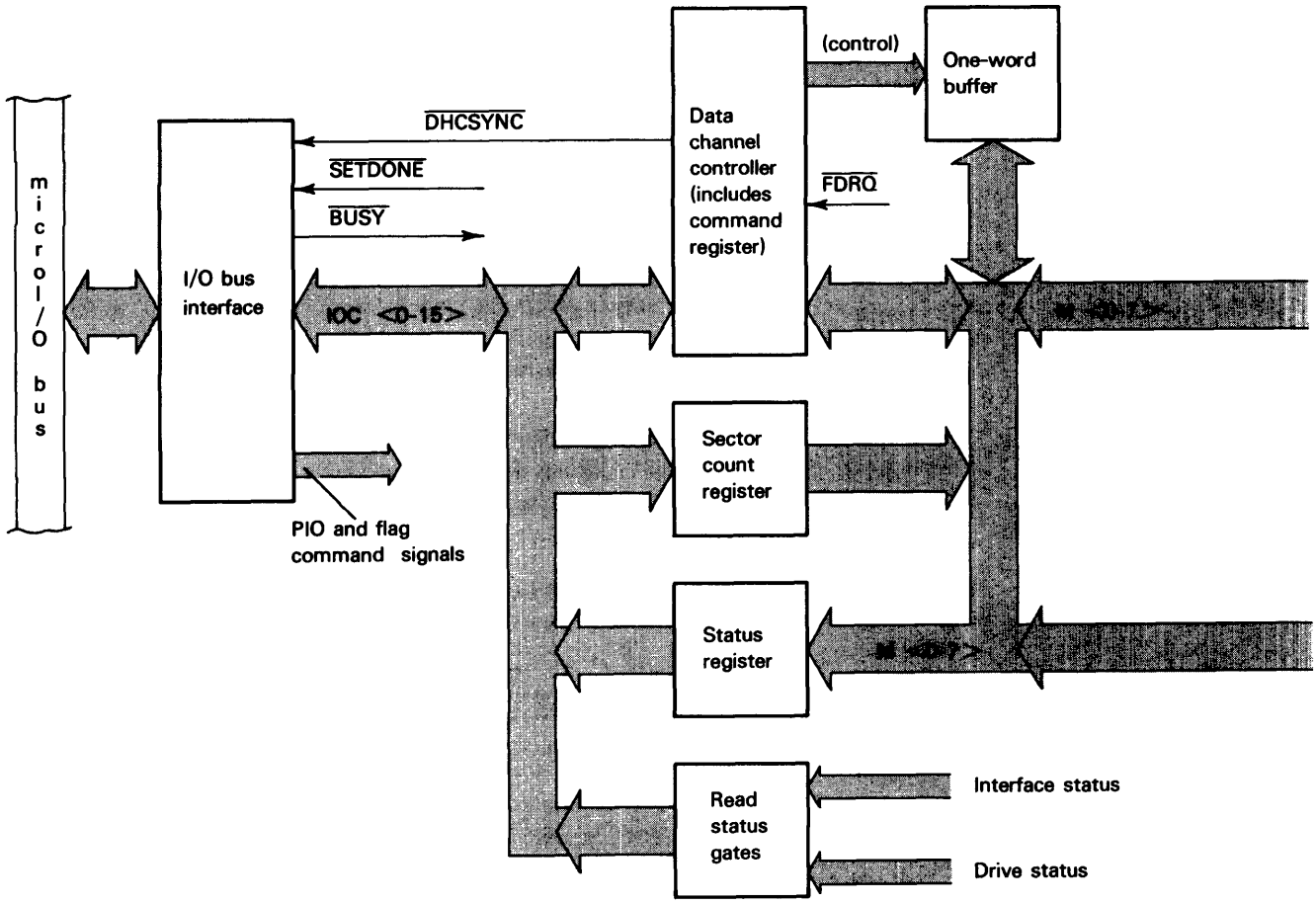
Supply Voltage	Current Draw (max)	Power Dissipation	Pin Numbers (Drive)
+5 Vdc	1 amp	5 W	4
+12 Vdc	1 amp	12 W	1
Ground	—		2, 3

Theory of Operation

This section describes the operation of the diskette interface card. It begins with a discussion of the major elements of the interface, followed by a general explanation of interface operations. The discussions in this section reference logic schematic DGC No. 001-003343.

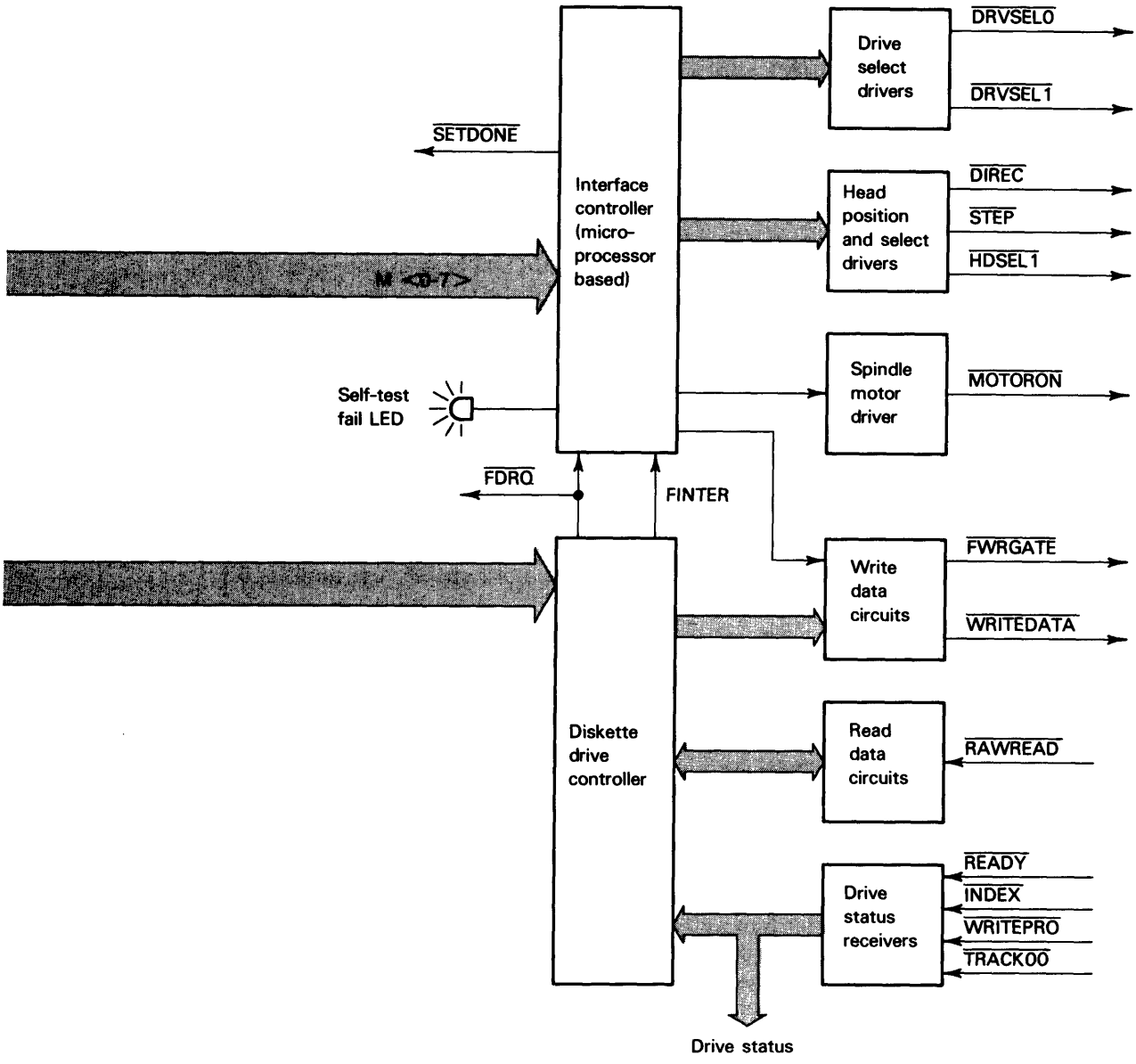
Interface Elements and Functions

The diskette subsystem interface directs all activities of the Model 20 and 30 diskette drive(s). It executes programmed input/output instructions from the host processor, supports data channel transfers from one sector to a full cylinder (two tracks) and between host memory and the diskette drive(s). Table 7-3 shows the major components of the interface and their interconnection.



ID-0063.

Figure 7-7 Diskette interface block diagram


Drive select drivers
DRVSEL0
DRVSEL1
SETDONE
Interface controller (micro-processor based)
Head position and select drivers
DIREC
STEP
HDSEL1
M <0-7>
Spindle motor driver
MOTORON
Self-test fail LED
FDRQ
FINTER
Write data circuits
FWRGATE
WRITEDATA
Diskette drive controller
Read data circuits
RAWREAD
READY
INDEX
WRITEPRO
TRACK00
Drive status receivers
Drive status

I/O Bus Interface The I/O bus interface connects the diskette interface to the microI/O bus and contains the memory address and word count registers. For more information on this interface, see *Input/Output and Interfacing.*

Interface Controller The interface controller is based on an 8-bit microprocessor that supervises *all* interface functions and controls the positioning of the drive read/write heads. The microprocessor monitors the Busy flag ($\overline{\text{BUSY}}$) to determine when to start an operation. When the Busy flag sets to 1, the microprocessor decodes and executes the command by:

Controlling the interface

Selecting a drive

Turning the drive spindle motor on

Positioning, loading, and selecting the read/write heads

Controlling the write circuits during write operations.

When the operation completes, the interface controller flags the CPU by signaling the I/O bus interface to clear the Busy flag and set the Done flag ($\overline{\text{SETDONE}}$). In addition, the microprocessor executes a self-test on power-up that checks itself and the diskette controller. If this check fails to complete successfully, the microprocessor turns on the self-test fail LED and sets the Not OK status flag.

Status Register and Status Gates This logic returns diskette interface and drive status to the CPU during a Read Status instruction. The description of the Read Status instruction in the diskette subsystem section of Chapter 2 discusses the status information returned.

Sector Count Register This register duplicates selective bits of the word count register contained in the I/O bus interface. These bits specify the number of sectors to be transferred during an operation.

Data Channel Controller and One-Word Buffer The data channel controller (DCH) connects the I/O bus interface to the diskette controller for data channel activities and contains the diskette command register. During read operations, it assembles 8-bit bytes from the diskette controller into 16-bit words and transfers them to memory. During write operations, it receives 16-bit words from memory and transfers them in 8-bit bytes to the diskette controller.

During read operations, requests to transfer data read from the diskette controller ($\overline{\text{FDRQ}}$) cause the DCH to transfer an 8-bit data byte from the diskette controller to the one-word buffer. Then the DCH transfers this 8-bit byte from the one-word buffer into a high- or low-order byte position of a 16-bit internal DCH register (high-order loads first). Each time the DCH transfers a byte into the low-order byte position, it requests a data channel cycle to transfer a word into memory. This process continues until the requested number of words are transferred *to* memory.

During write operations, a data channel cycle is initially requested ($\overline{\text{DCHSYNC}}$) to load a word from memory into a 16-bit internal DCH register. The DCH then loads the high-order 8-bit byte into the one-word buffer. As the diskette controller issues requests ($\overline{\text{FDRQ}}$) for data bytes to write, the DCH transfers the byte stored in the one-word buffer to the diskette controller. Following the transfer, the DCH loads the next byte to be written into the one-word buffer. Each time it loads the low-order byte into the buffer, the DCH requests another data channel cycle to fill

its 16-bit internal register. This process continues until the requested number of words are transferred *from* memory.

Byte Swap Network This network can be enabled to

(1) Swap high- and low-order bytes of 16-bit words between memory and the DCH during data channel activity, or

(2) Swap bytes when reading from or writing to diskettes in Data General MPT/100 format. This format writes or reads the low-order byte of a 16-bit word first.

Data Channel Direction Logic This logic inserts the data channel direction bit returned to the CPU data channel facility, along with the 15-bit memory address, during the address portion of each data channel cycle.

Diskette Drive Controller The diskette drive controller (DDC) handles read/write operations. It provides the modified frequency modulation (MFM) method of encoding and decoding recorded data, along with the serialization (write) and deserialization (read), needed to read and write data on the diskette. The DDC contains a data shift register that assembles the serial read data stream received from the diskette drive into 8-bit bytes during read operations. The shift register also disassembles 8-bit bytes into a serial write data stream to be transferred to the diskette drive during write operations.

The DDC also contains five microprocessor-selectable registers whose functions are defined below.

- *Command register.* Specifies the next operation (read, read header, or write) to be performed by the selected drive.
- *Track address register.* Holds the track address specified during the last seek command issued to the diskette interface.
- *Sector address register.* Holds the sector address that is to supply or receive data during the next sector transfer between the DDC and the selected drive. This register receives the sector address specified during a Read, Read Header, or Write command issued to the diskette interface, and it updates as each sector is transferred.
- *Data register.* Used as a holding register during diskette read and write operations. During a read operation, it holds the byte assembled by the DDC's data shift register from the serial read data stream and latches the byte until the data channel controller transfers it to the one-word buffer. During a write operation, it holds the byte transferred from the one-word buffer by the data channel controller until the DDC's data shift register is ready to convert it into the serial write data stream.
- *Status register.* Holds status information about the DDC, the selected diskette drive, and data transfers.

The DDC requests data transfers ($\overline{\text{FDRQ}}$) as required from the data channel controller (DCH). When the DCH services the request, it transfers a data byte to or receives a data byte from the one-word buffer and removes the data transfer request. In addition, the DDC notifies the microprocessor when it completes any command (FINTR).

Read Circuits These circuits convert the serial read data stream from the drive into serial data bits and clock signals that are sent to the diskette controller.

Using the clock signals, the DDC assembles eight data bits and loads them into its data register.

Write Circuits These circuits provide the write precompensation required for high-density recording.

Diskette Subsystem Interface Operations

The discussion that follows explains in general terms how the diskette interface operates. At times it refers to the previous discussion of functional elements for background and supplementary information.

Program Control All diskette drive operations must be set up by data transfers between the CPU and the diskette interface. Each of these transfers involves reading or writing an interface register under direct program control. All programmed I/O instructions received by the diskette interface are decoded by the I/O bus interface. Then the instruction is either carried out by the I/O bus interface *or* the I/O bus interface asserts control signals to cause the diskette interface to carry out the instruction.

Positioning the Read/Write Heads The microprocessor controls four different head positioning operations:

Recalibrate	Steps the heads to track 00.
Seek	Steps the heads to the track specified by the track field of the command register.
Step in	Steps the heads from their current position one track further from track 00.
Step out	Steps the heads from their current position one track towards track 00.

Before data can be transferred between memory and the diskette, the heads must be positioned over the track to be accessed. The program must define the head positioning operation by writing the appropriate control information to the diskette interface command register located in the data channel controller (DCH), and initiate the operation. Once this information is loaded into the command register, the microprocessor begins the operation by sending control signals to the drives that select the drive unit and position the drive read/write heads. The microprocessor also loads the target track number, stored in the command register, into the apprropriate register of the diskette drive controller (DDC). When the head positioning operation is complete, the microprocessor flags the CPU by signaling the I/O bus interface to set the Busy flag to 0 and the Done flag to 1. Setting the Done flag to 1 initiates an interrupt if interrupts are enabled.

Setting Up Data Transfers Once the heads are positioned over the desired track, the program must set up the standard data channel transfer by loading the memory address and the word count registers, located in the I/O bus interface, with the starting memory address and number of words to be transferred into or out of memory. Next, the program must specify the type of data transfer, the starting head number, and the starting sector number by loading the appropriate command information into the diskette interface command register, located in the DCH, and initiate the operation.

Once the program specifies and initiates the operation in the command register,

the microprocessor begins the operation, sending the control signals to the drives that select the drive and read/write head. The microprocessor also loads the command, starting sector and head numbers from the command register into appropriate registers of the diskette drive controller (DDC) and initiates the data channel controller (DCH). (The track number was loaded into the diskette drive controller during the last head positioning operation.) In addition, the microprocessor transfers the content of the sector count register into one of its internal registers. For a write operation, the DCH requests a data channel cycle to obtain the first word to be written from memory.

Writing Sectors

Once the command is loaded into the DDC, the DDC begins the operation. For a write operation, it first checks the diskette media in the selected drive to see if it is write-protected; if so, the DDC terminates the operation and asserts its operation done line to the microprocessor. The microprocessor flags the CPU by signaling the I/O bus interface to set the Busy flag to 0 and the Done flag to 1. Setting the Done flag to 1 initiates an interrupt when interrupts are enabled. If the diskette is not write-protected or if the operation is a read, the DDC continues the operation.

The DDC locates the starting sector by reading consecutively, starting with the first address fields it encounters and comparing the track, sector, and head address fields to the contents of its track, sector, and head address registers.

When the starting sector is located during a write operation, the DDC requests the data channel controller (DCH) to supply a data byte. (Refer to "Data Channel Controller" in this section for details on the transfer of data between the DCH and DDC.) When the DDC receives the data byte from the DCH, it requests the next data byte and encodes the current one into a modified frequency modulated serial write data stream that is sent to the selected diskette drive through the write circuits. The DDC continues to write the rest of the diskette data field in the same way, requesting DCH data transfers for each byte that it writes. The DCH requests data channel cycles for each two bytes that it transfers to the DDC.

Reading Sectors

The DDC locates the starting sector by reading consecutively. It starts with the first address fields it encounters, and compares the track, sector, and head address fields to the contents of its track, sector, and head address registers.

When the starting sector is located during a read operation, the DDC reads the sector's data field. As each byte is assembled, the DDC requests DCH data transfers to transfer the data byte to the DCH. (Refer to "Data Channel Controller" in this section for details on the transfer of data between the DDC and the DCH.) The DDC continues to read the rest of the sector in the same way, requesting DCH data transfers for each byte that it assembles. The DCH requests data channel cycles for each two bytes that it transfers from the DDC.

Ending the Data Transfer

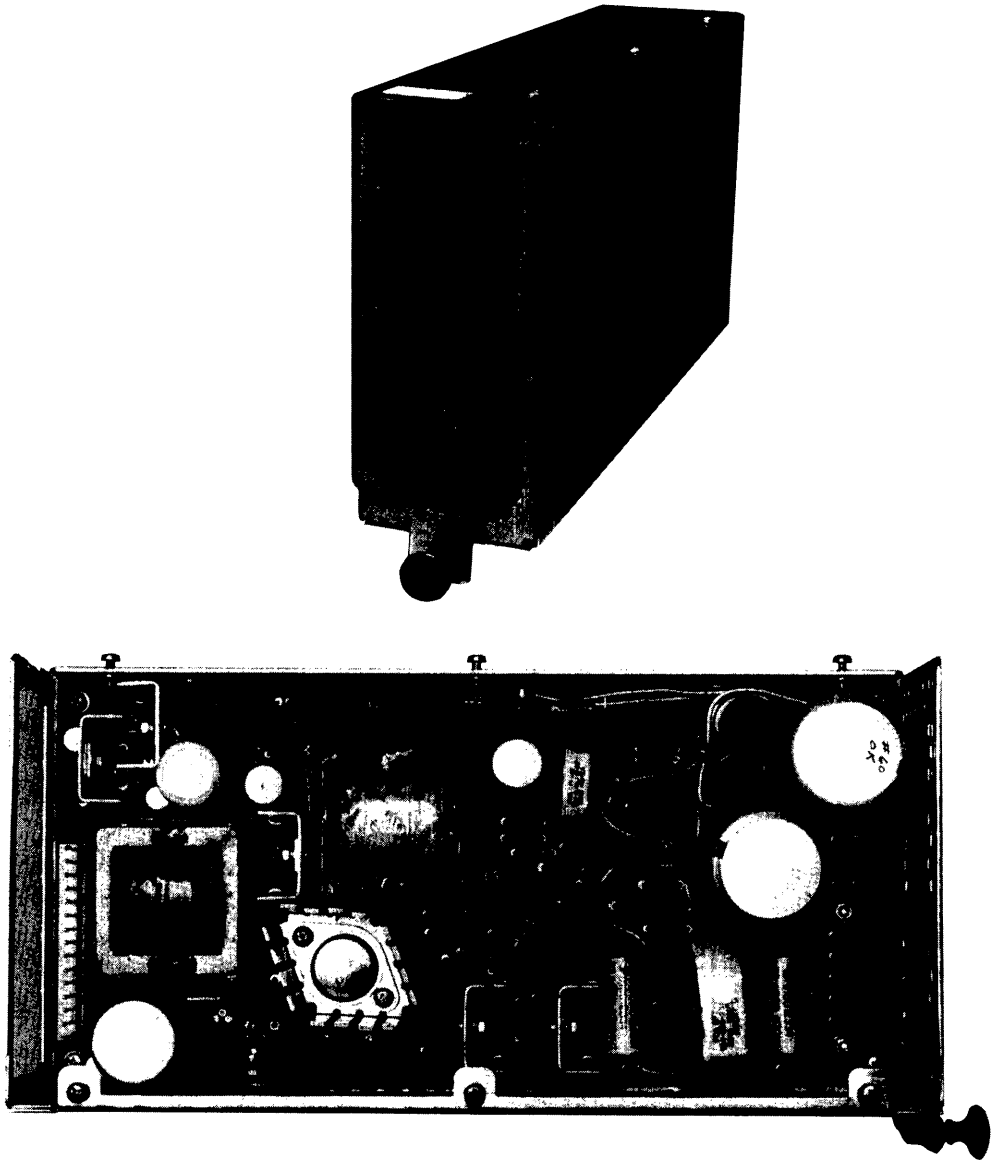
The DDC terminates a read or write operation after the last byte of the data field is read or written, and notifies the microprocessor. The microprocessor interrogates the sector count, stored in one of its internal registers. If the sector count specified a single-sector transfer, the microprocessor flags the CPU by signaling the I/O bus interface to set the Busy flag to 0 and the Done flag to 1. Setting the Done flag to 1 initiates an interrupt when interrupts are enabled. If the sector count specified a multiple-sector transfer, the microprocessor decrements the sector count and initiates another read or write operation as previously described, switching to head 1 if required, until the specified number of sectors are read or written or the end of the cylinder (end of track while head 1 is selected) is reached.

Power Supply Assembly

8

A Model 20 and Model 30 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 8-1. 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 the power supply's ac input and dc output specifications, explains its theory of operation, and demonstrates its interconnection with the system.



PH-0717, 718

Figure 8-1 Model 20 and Model 30 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 PCB. 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 8-1 and Table 8-2 provide the ac input and dc output specifications for each power supply assembly.

Figure 8-2 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 + 12V, − 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 Model 20 and Model 30 power supply, as shown in Figure 8-2. 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 8-1

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 8-2

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

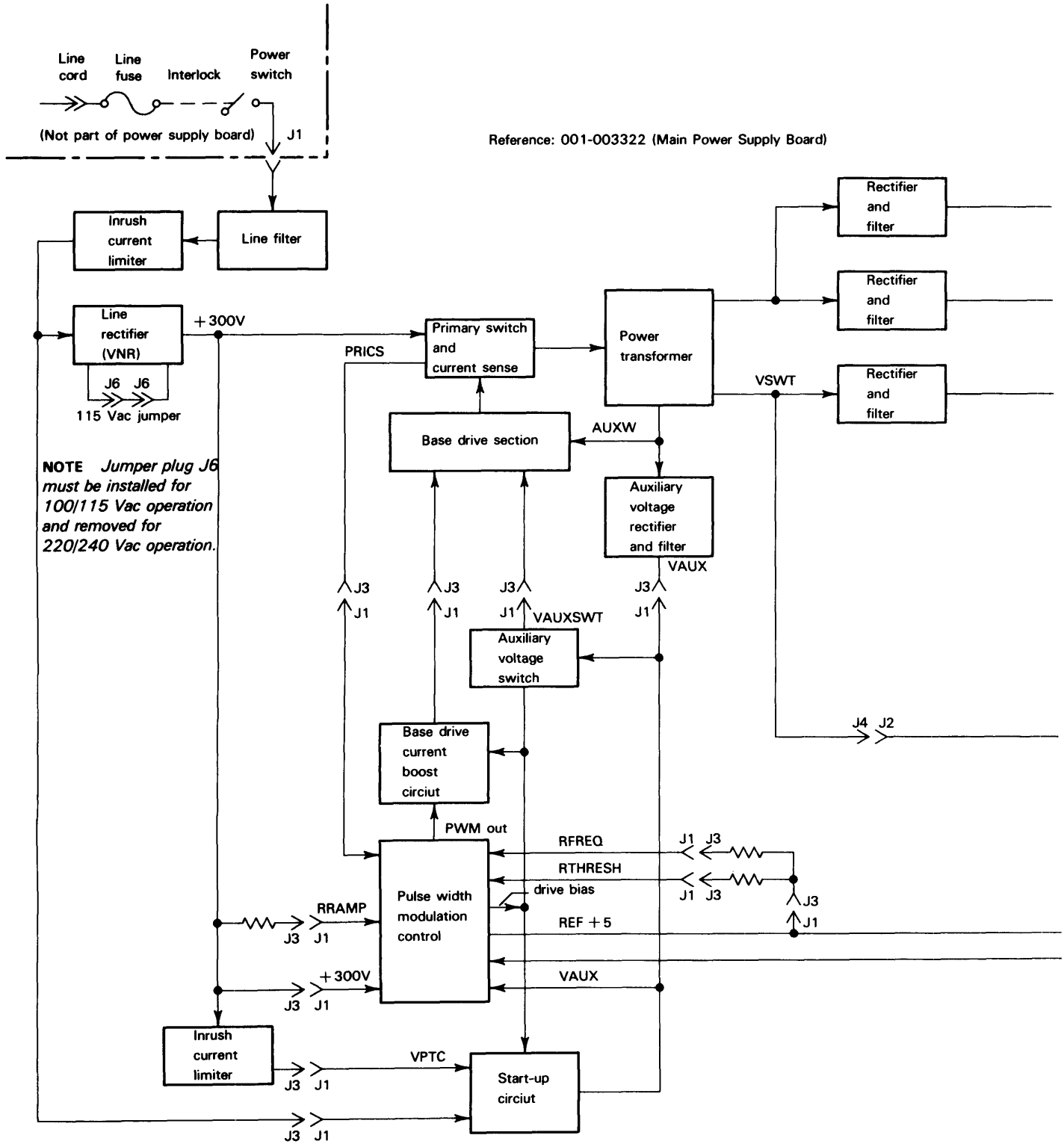
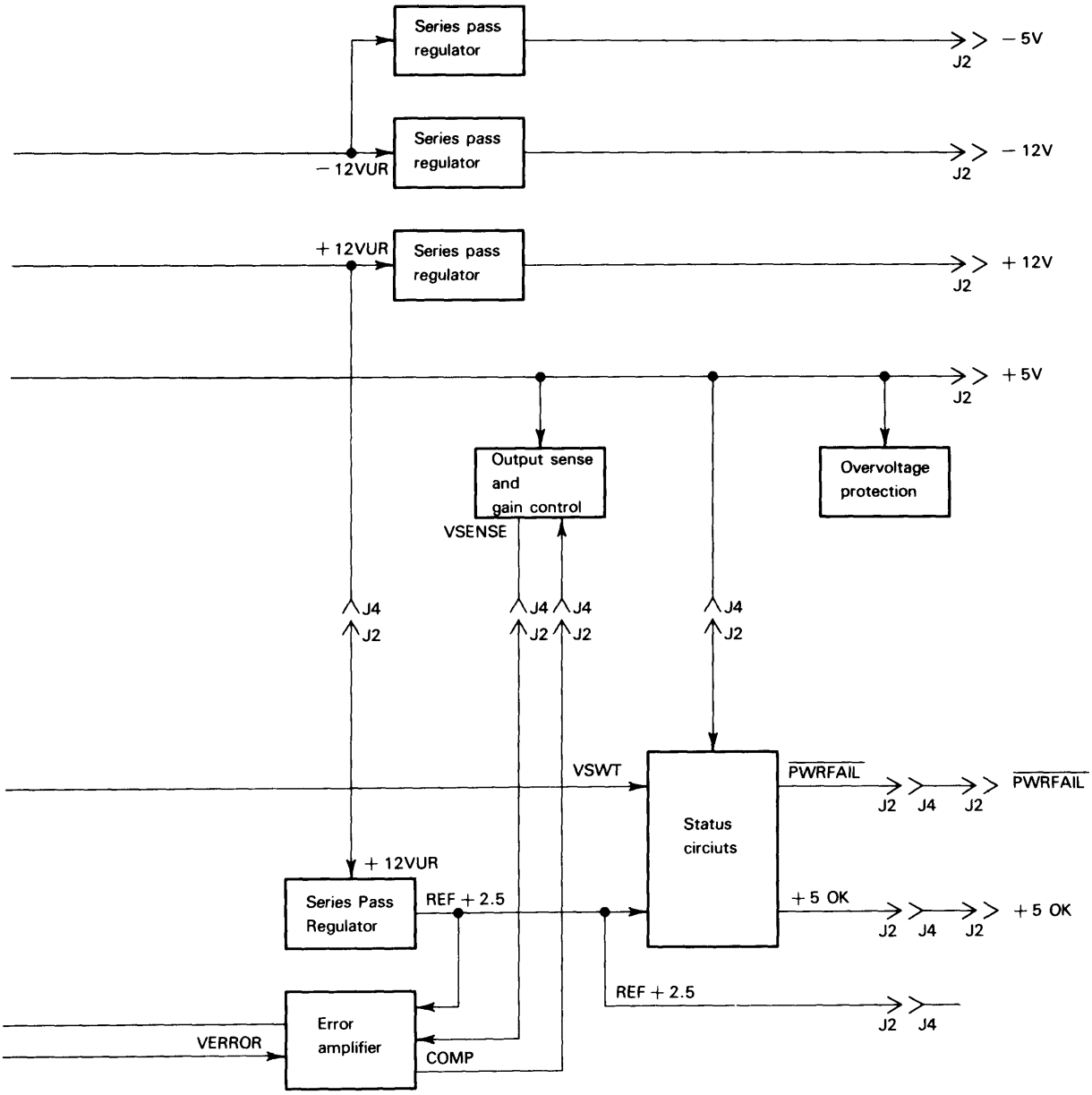


Figure 8-2 Model 20 and Model 30 power supply functional block diagram



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).

Page 8-10

Figure 8-3 Status signal timing diagram (power up and power down)

Change the 5 ms minimum time to 300 ms minimum

Change the 10 ms minimum time to 2 ms minimum

Page 8-13

Table 8-5 Power status signals

Delete the up-arrow (^) symbol before the signal name PWRFAIL in the table and place a bar over this signal name.

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

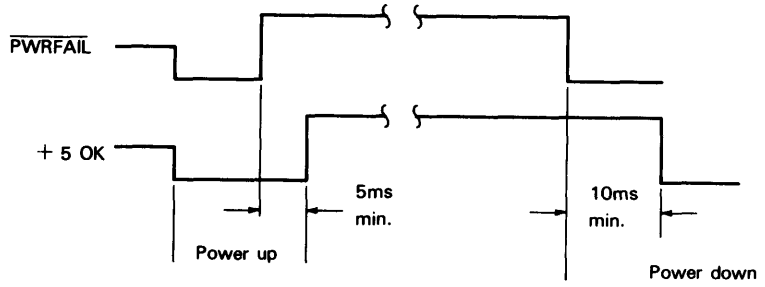
This section consists of a secondary winding of the power transformer, and a rectification network that provides auxiliary voltage (VAUX) to power the control circuitry of the power supply once the supply begins operation. The section also contains an auxiliary voltage switch that provides magnetizing 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 8-3.



ID-0037

Figure 8-3 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.

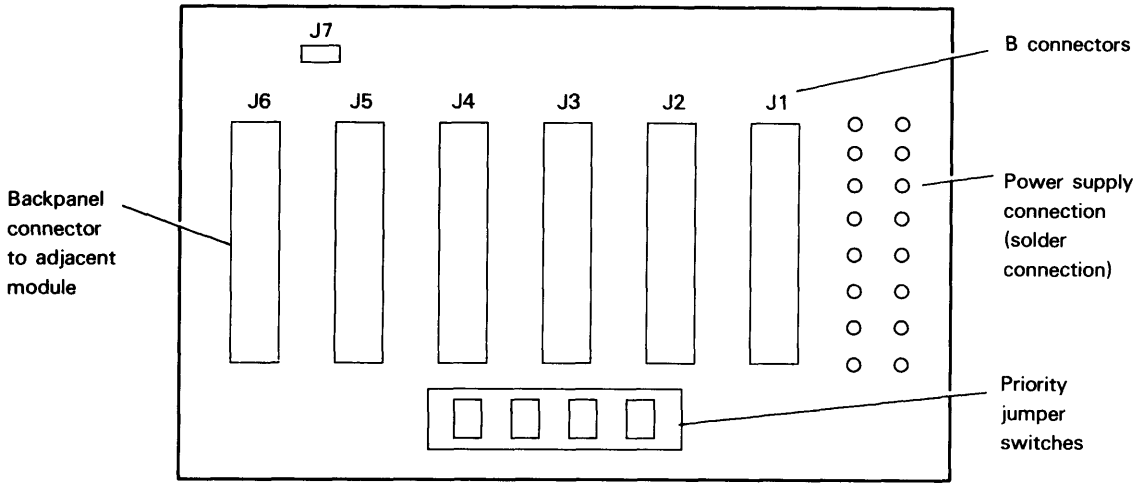
The power ok monitor compares the + 5V output, applied to a voltage divider, with a threshold level established from the reference voltage (REF + 2.5). During power up, an R/C circuit (R12 and C7) delays the assertion of the + 5 OK signal so that the supply voltages can stabilize. Should the + 5V output 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 Model 20 and Model 30 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 8-3 through Table 8-5 list each signal received or generated by the power supply card together with the jack location(s) of the signal. Refer to Figure 8-4 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 8-4 CLM backpanel

Table 8-3 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 8-4 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 8-5 Power status signals

Signal	Jack Pin
+5 OK	J2-14
^PWRFAIL	J2-15

Part

THREE

Mechanical Assemblies

Model 20 and Model 30 Modules and Configurations

9

Model 20 and Model 30 systems consist of three to six modules horizontally connected to form a single desk- or shelf-top unit. Figure 9-1 demonstrates this modular design. There are six module types, each type accommodating specific components of the system.

This chapter describes the Model 20 and Model 30 system modules and their architecture, configurations, and interconnections. The chapter also details system cabling and system expansion potential.

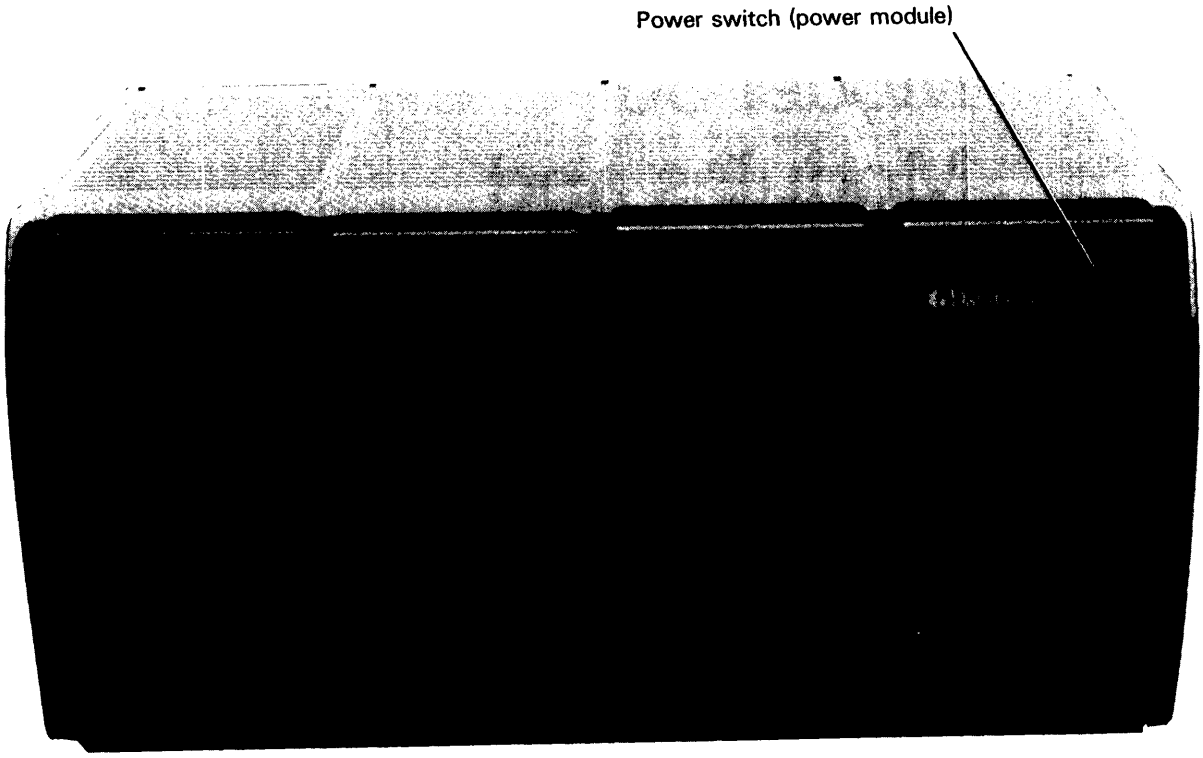


Figure 9-1 Model 20 and Model 30 modules

Page 9-9

Under page heading, "Module Architecture",
The first two sentences of the paragraph should read:

Each module consists of a metal cage with a base plate and removable front radio-frequency interference (RFI) shield panel. The metal cage is enclosed in removable plastic front, rear, and top panels.

Page 9-11

Add the following phrase to the second sentence of the second paragraph, after "removing its plastic front cover":

" and RFI shield panel."

Page 9-12

Under paragraph heading, "Input/Output Bus",
The fourth sentence of the paragraph, beginning in line 5, should read:

The input/output bus can be extended to incorporate compatible Data General peripherals, as explained under "Input/Output Bus Extension".

Page 9-14

Under page heading, "Input/Output Bus Extension",
First paragraph, line 2; third paragraph, lines 3 and 4:

Change the word "standard" to "compatible"; delete the word "Microproducts".

Add the following note after the first paragraph:

NOTE *Data General does not provide support for controller/peripheral combinations connected to the Models 20 or 30 systems by means of the external, extended microI/O bus cable.*

Change the fourth paragraph, beginning on the bottom of the page, to read:

Remember that when compatible peripheral(s) are connected to the system by means of the extended microI/O bus, the last peripheral connected to the extended microI/O bus must terminate the bus.

Page 9-29

Under page heading, "Line Cords",
Add the following warning:

WARNING *In some configurations it is possible that your computer system may exceed standard (15 amps) duplex wall outlet power specifications. This will cause an outlet overload, resulting in a tripped circuit breaker (or blown fuse) at your local ac power distribution panel. In this case, the system will require higher capacity wiring and wall outlet.*

Unit Architecture

The six module types that can comprise Model 20 and Model 30 systems are pictured in Figure 9-2 through Figure 9-4 and are described below.

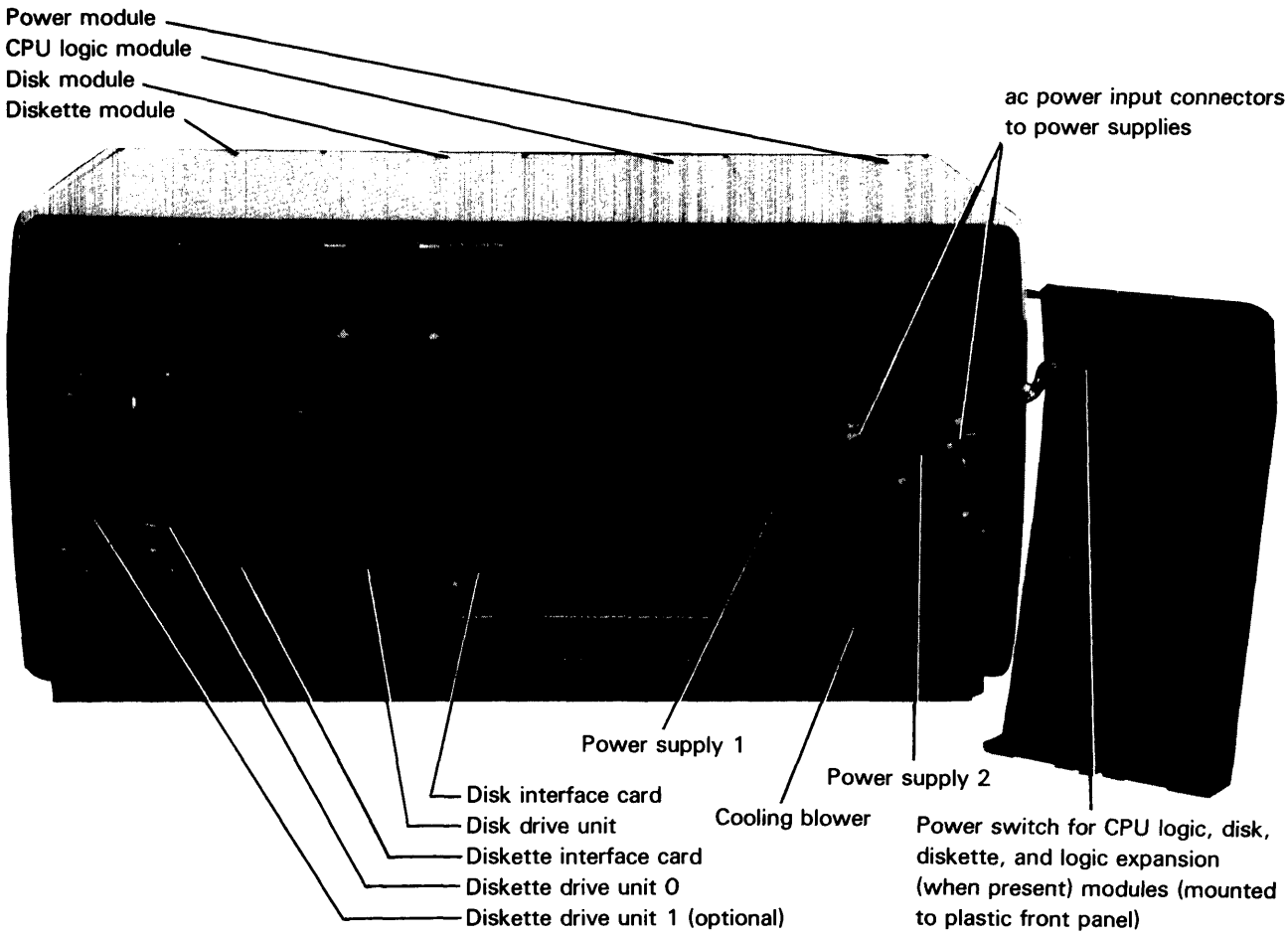


Figure 9-2 Model 20 and Model 30 system modules (front view)

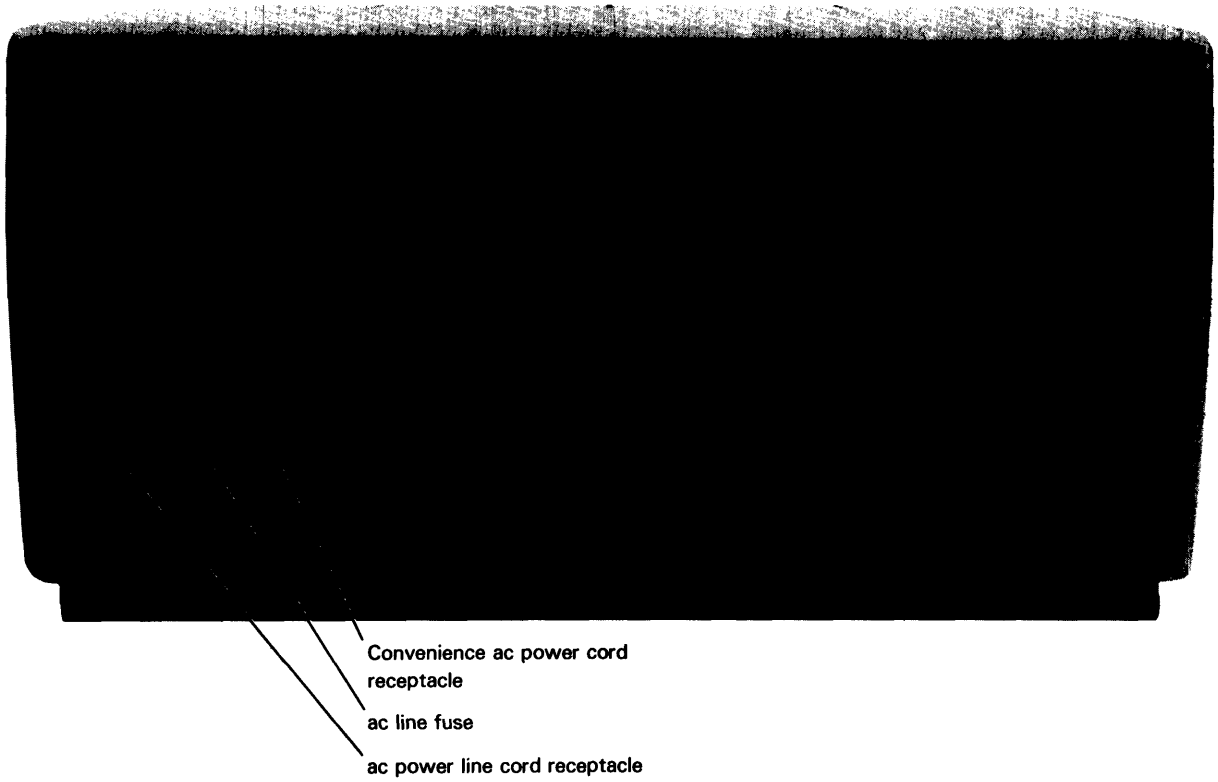


Figure 9-3 Model 20 and Model 30 system modules (rear view)

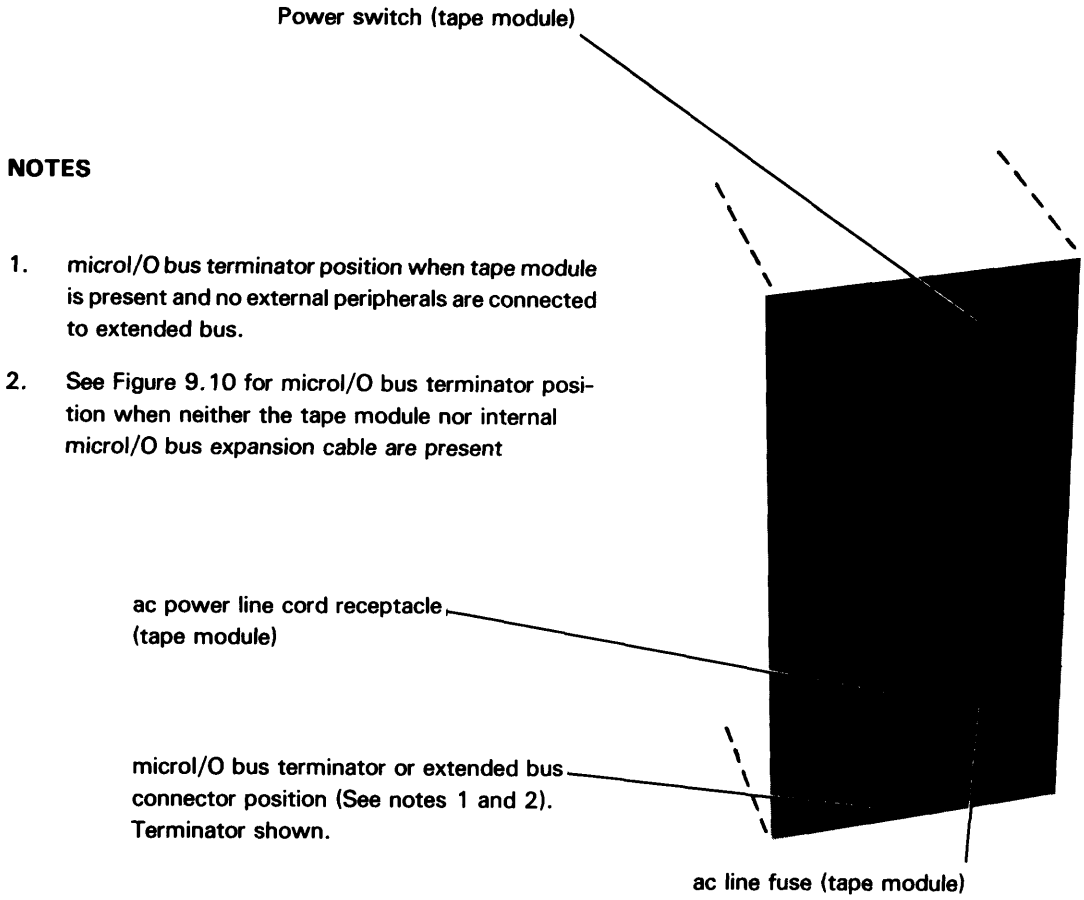


Figure 9-4 Tape module (rear view)

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 a line frequency clock card; and space for two power supply assemblies. Power supply (supply 1) occupies the left-most position; 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 9-5 shows the air flow pattern.

CPU Logic Module (CLM)

This module accepts up to five 7-inch by 9-inch printed circuit cards including the system processor card, memory cards, and I/O interface cards. The hardware floating point card of the Model 30 system is also contained in this module. 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. A 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 interface contained in the system unit.

Diskette Module (FM)

This module contains the diskette controller card and space for two diskette drive units, one standard, and one optional. The diskette controller card interfaces to both drives. 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.

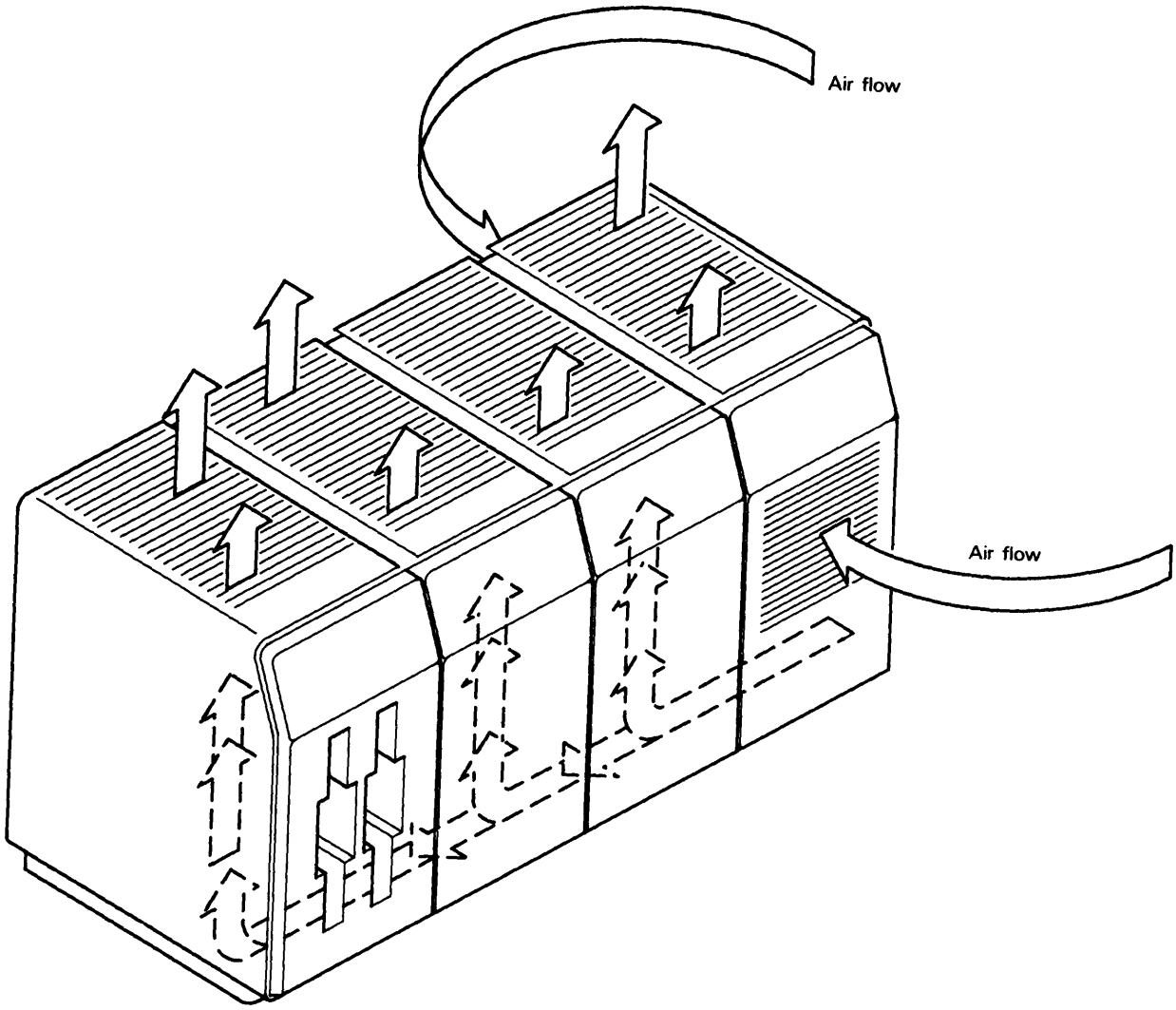


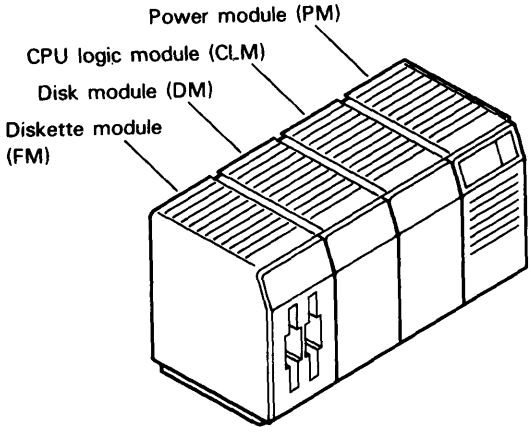
Figure 9-5 Air flow

ID-00655

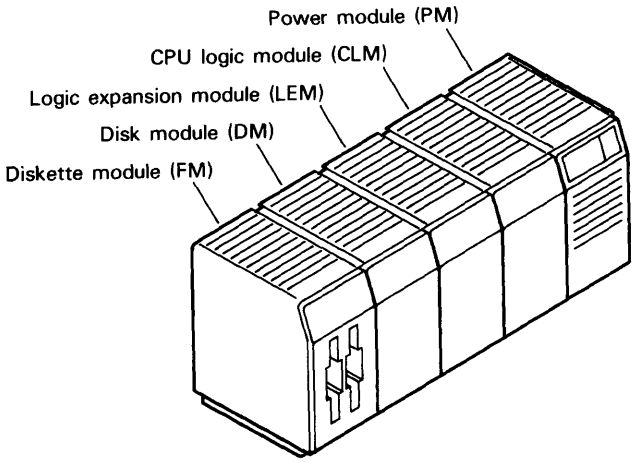
Configurations

Preconfigured Model 20 and Model 30 system units always consist of at least four modules; a power module with two power supplies, CPU logic module, disk module, and diskette module. They can be expanded to six modules by the addition of a logic expansion module and tape 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 tape module is added it must be placed to the left of and adjacent to the diskette module. Three module system units consist of a power module with one power supply assembly, CPU logic module, and diskette module.

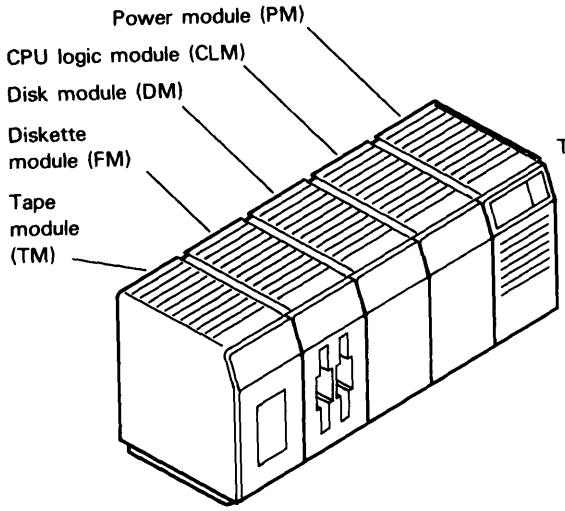
The disk expansion unit, when present, always consists of two modules; a power module with one power assembly, and a disk module.



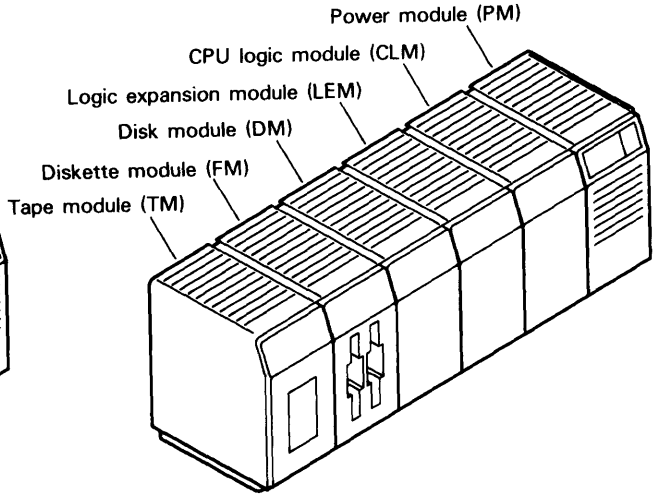
Basic configuration



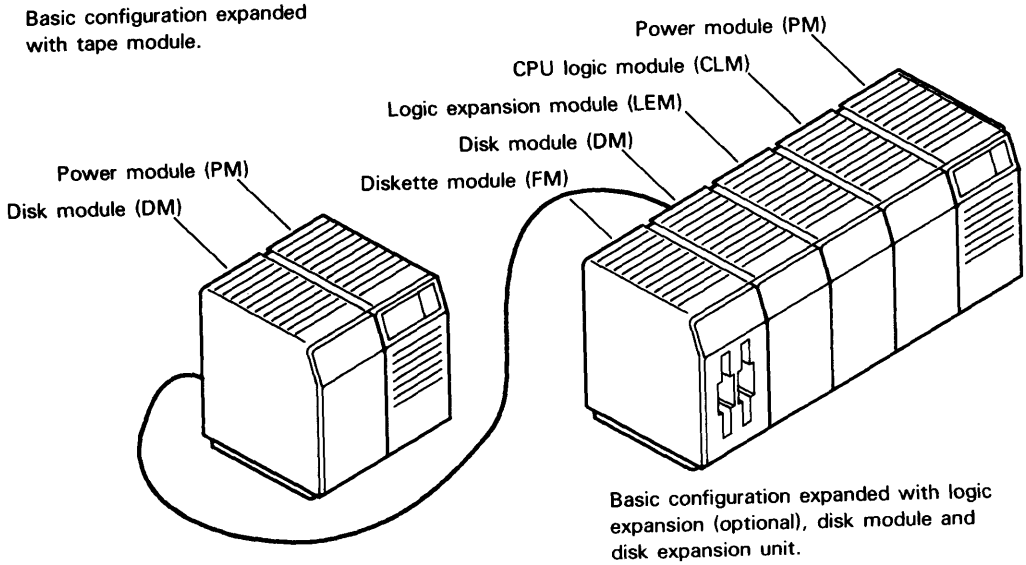
Basic configuration expanded with logic expansion



Basic configuration expanded with tape module.



Basic configuration expanded with logic expansion and tape modules



Basic configuration expanded with logic expansion (optional), disk module and disk expansion unit.

ID-00656

Figure 9-6 Model 20 and Model 30 system unit configuration

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, and expansion disk modules, 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 backpanel 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 9-7 shows a typical Model 20 and Model 30 module.

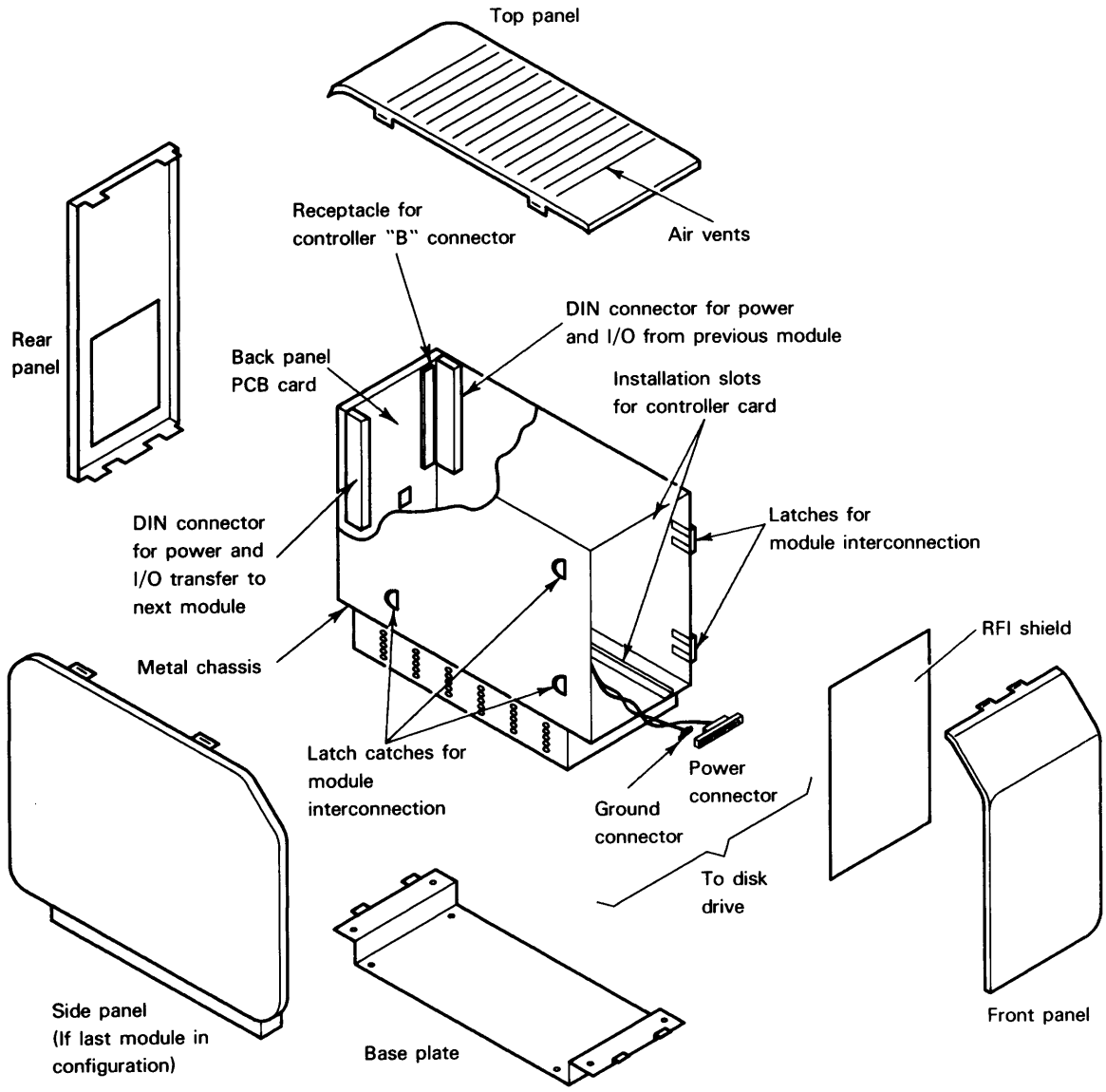


Figure 9-7 Typical Model 20 and Model 30 module

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 power supply assemblies as shown in Figure 9-8. 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" 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 respective module after removing its plastic front cover. The tape controller card inserts from the rear of its module after removing the rear plastic cover.

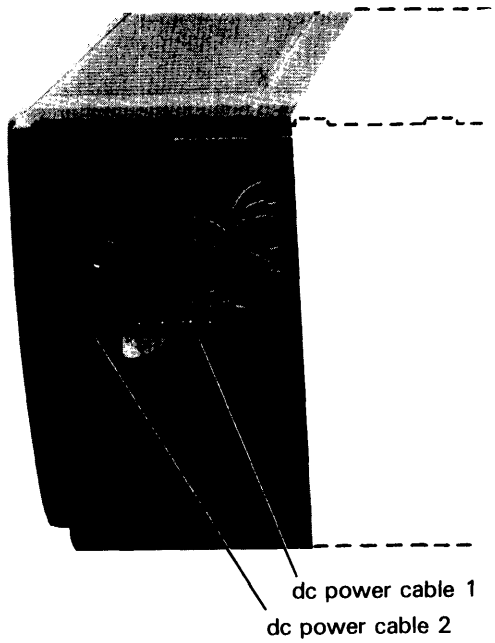


Figure 9-8 Dc power cable connections

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 module 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 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

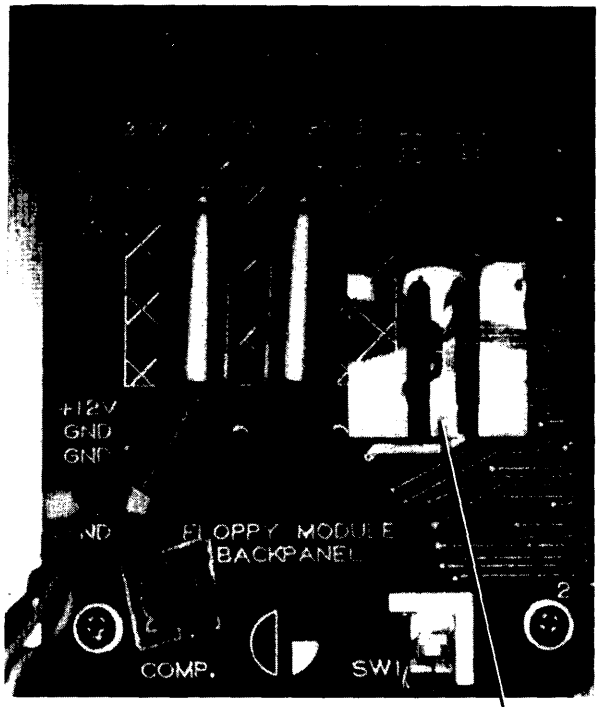
A printed circuit memory bus extends horizontally across the CLM backpanel. It connects the Model 20 and Model 30 system processor unit and optional hardware floating point cards to the system memory card(s). The memory bus contains 26 signal lines for address and data transfers, timing, and control.

Input/Output Bus

A printed circuit input/output bus extends horizontally across the CLM, LEM, FM, and DM backpanels. It connects the Model 20 and Model 30 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. The input/output bus can be extended to incorporate standard Data General Microproducts 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

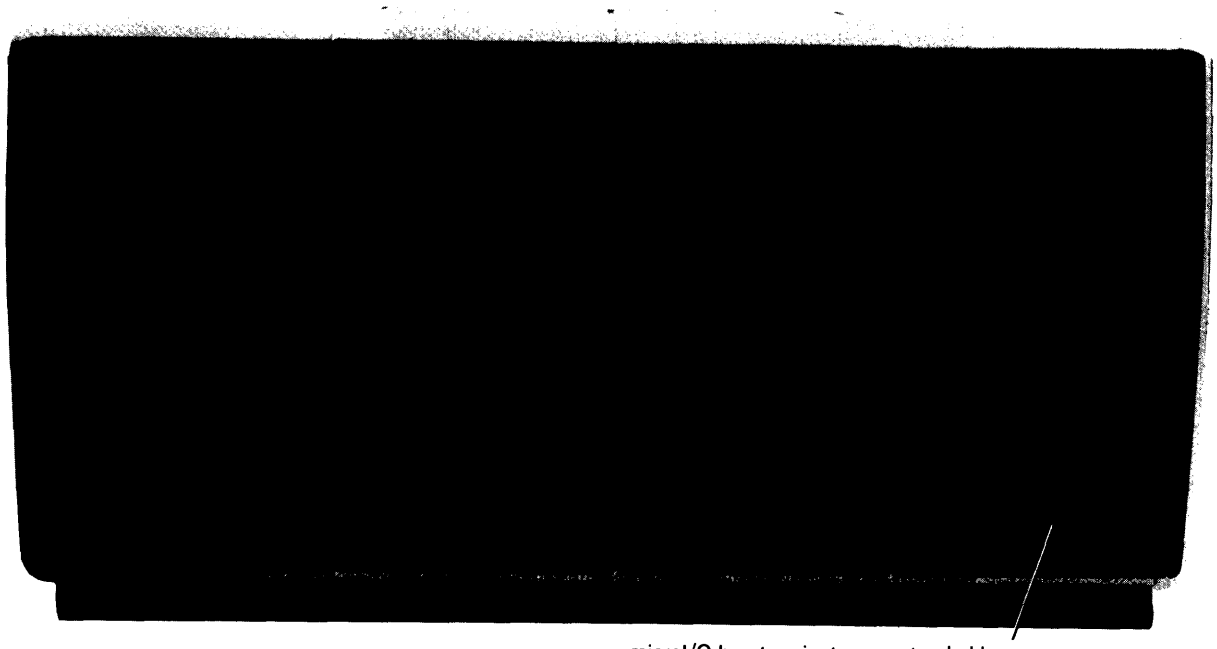
Termination for the input/output bus must be provided at the tape module (in the diskette module when the tape module is not present) when the extended microI/O bus is not present. Termination is accomplished with a DIP termination block that plugs onto pins located on the FM backpanel (see Figure 9-9 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 9-10). 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 Microproducts peripheral connected to the expanded bus.



microl/O bus terminator position (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.*

Figure 9-9 microl/O bus terminator (in diskette module)



NOTES

1. 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.
2. See Figure 9.9 for microI/O bus terminator position when neither the tape module nor internal microI/O bus expansion cable are present.

Figure 9-10 External microI/O bus connector or terminator positions

Input/Output Bus Extension

The Model 20 and Model 30 system unit can be electrically connected to standard Data General Microproducts 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 9-10.

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 Microproducts peripheral connected to the Model 20 and Model 30 system. The extended microI/O bus cable is available in various lengths to accommodate peripheral placement.

Remember that when Microproducts 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 20 and Model 30 CPU logic and logic expansion modules are listed in Figure 9-11 and Figure 9-12 respectively.

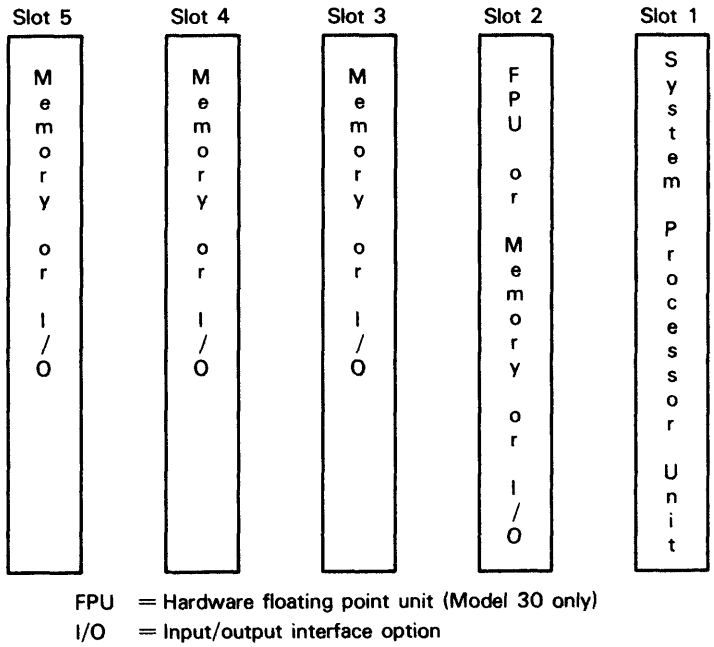
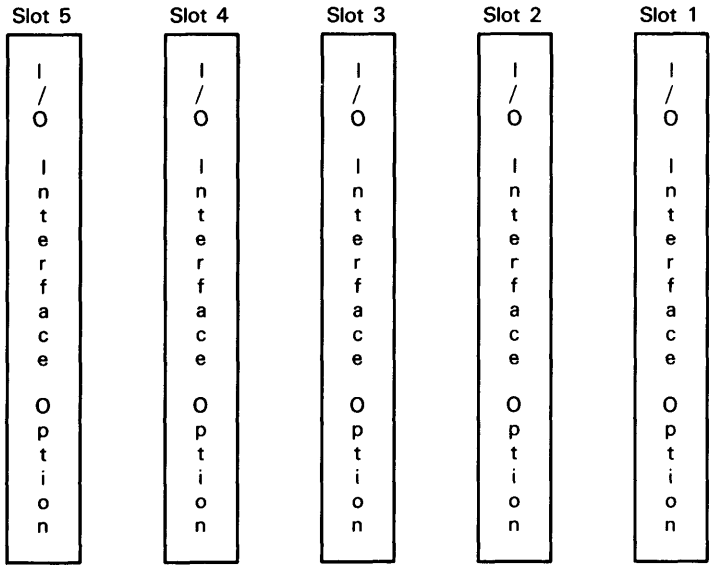


Figure 9-11 Slot assignments: CPU logic module



NOTE *Slot 1 must contain an I/O interface printed circuit card to pass interrupt and data channel priority signals.*

ID-00659

Figure 9-12 Slot assignments: logic expansion module

Backpanel Pin Assignments

Actual backpanel layouts and their pin assignments for all modules are shown in Figure 9-13 through Figure 9-16.

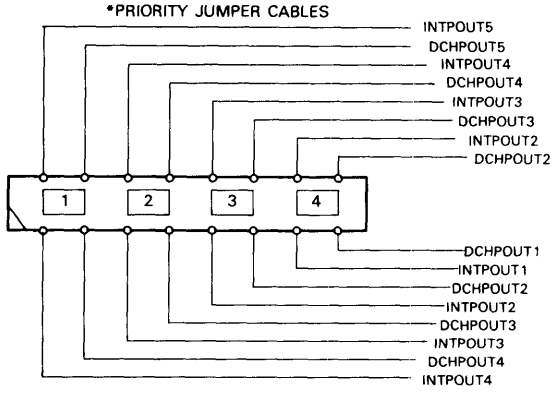
J6

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 9-13 Backpanel pin assignments: CPU logic

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

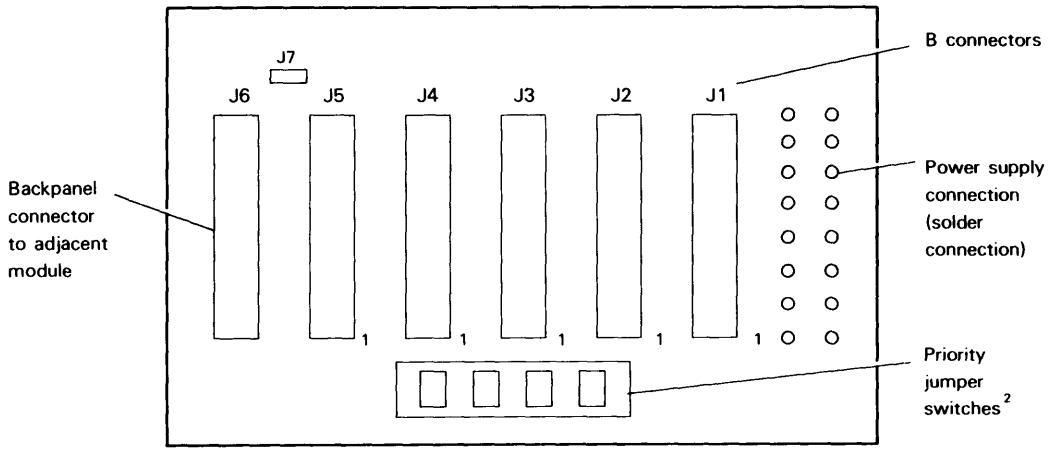
ID-00685

Figure 9-13 Backpanel pin assignments: CPU logic

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

Actual Layout of backpanel (viewed from front of module)



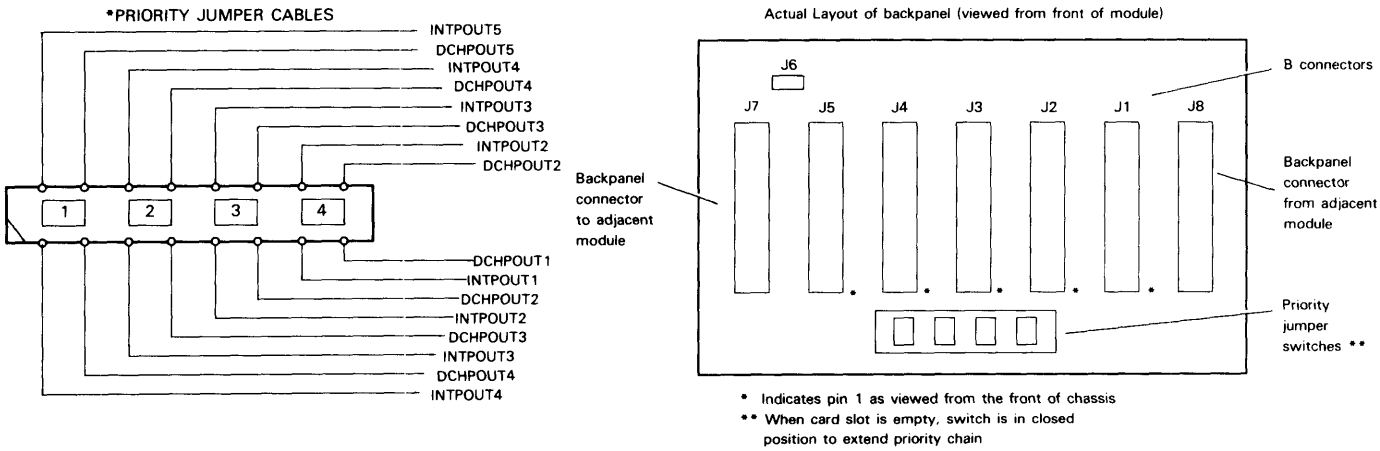
1. Indicates pin 1 as viewed from front of chassis
2. When card slot empty, switch is in closed position to extend priority chain.

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 9-14 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	
40		39	−12 PS #2
38		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

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

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

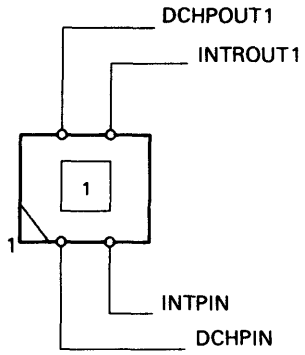
J1

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 9-15 Backpanel pin assignments: diskette module

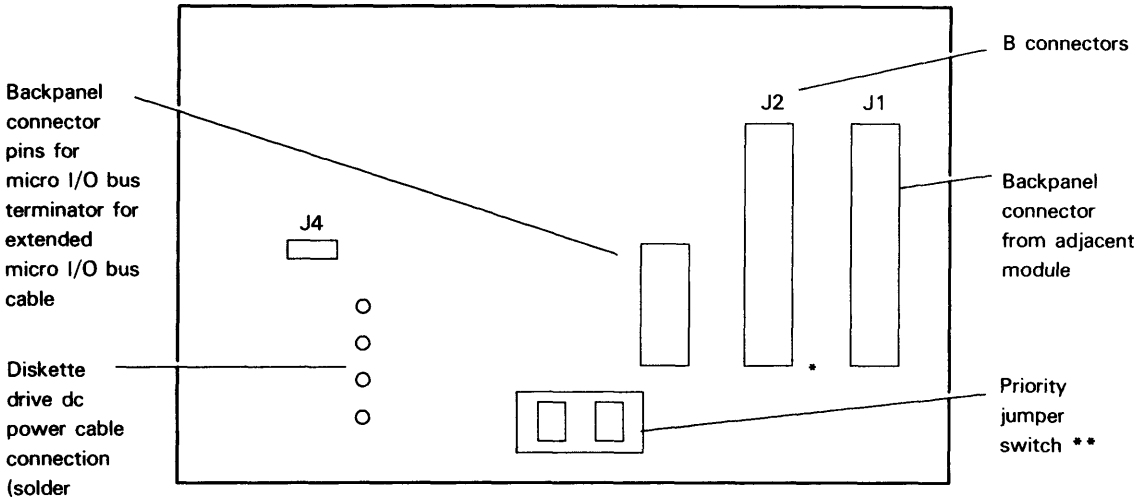
Priority jumper switch



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

Disk drive
dc power cable connection

E1	-	+12 PS #1
E2	-	GROUND
E3	-	GROUND
E4	-	+5 PS #1
E5	-	NO CONNECTION
E6	-	GROUND

ID-00688

Figure 9-16 Backpanel pin assignments: disk module

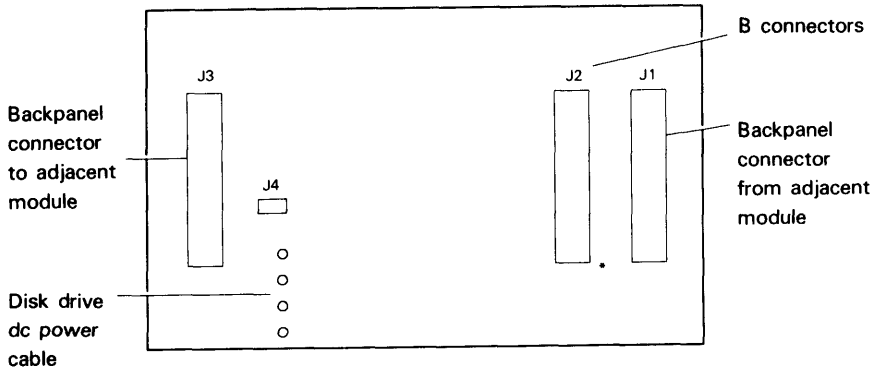
SLOT 1 - J2

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

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

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 20 and Model 30 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 9-13 through Figure 9-16. 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. (Refer to [Installation Data Sheets], DGC No. 010-######.)

Power Switch

The power switch, located on the front of each power module (Figure 9-17), 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 9-18).

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.*

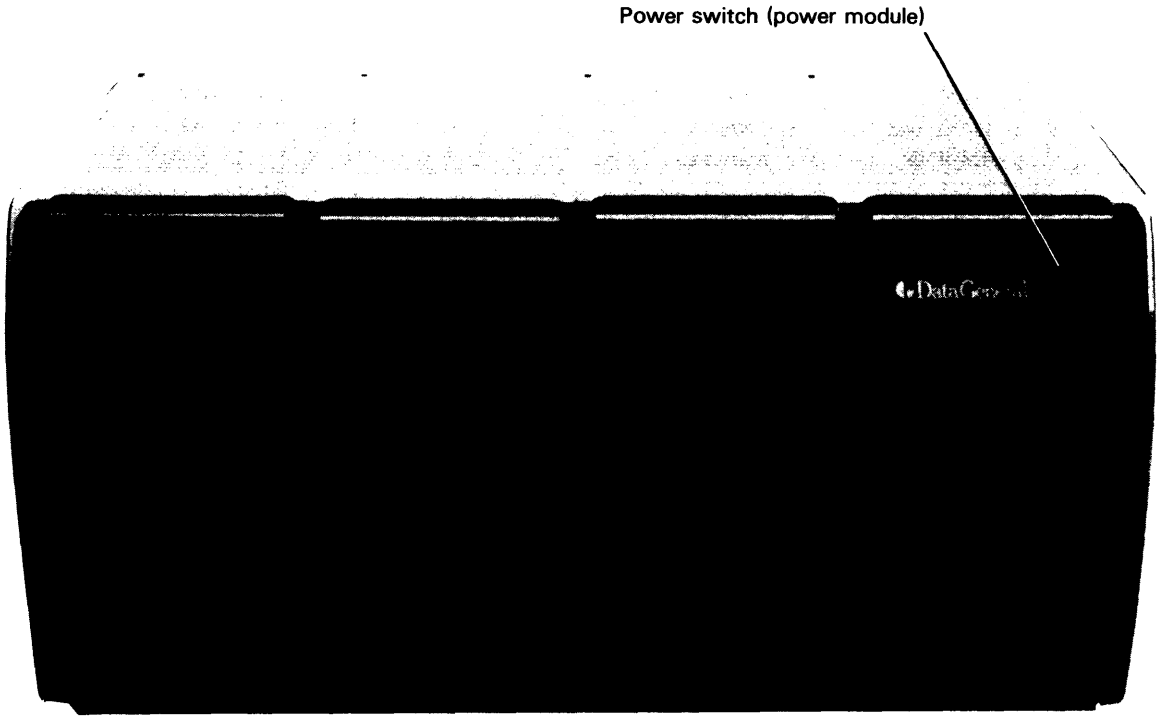


Figure 9-17 Power switch, power module

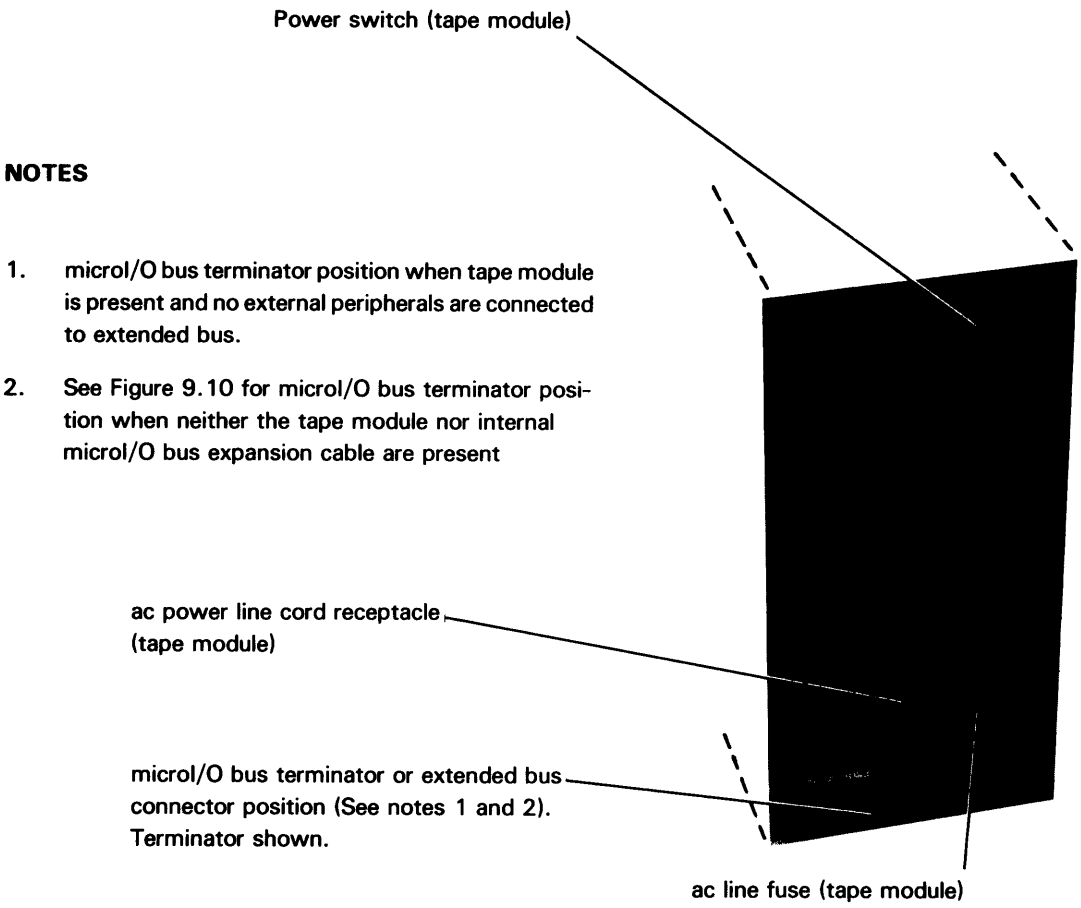


Figure 9-18 Tape module (rear view)

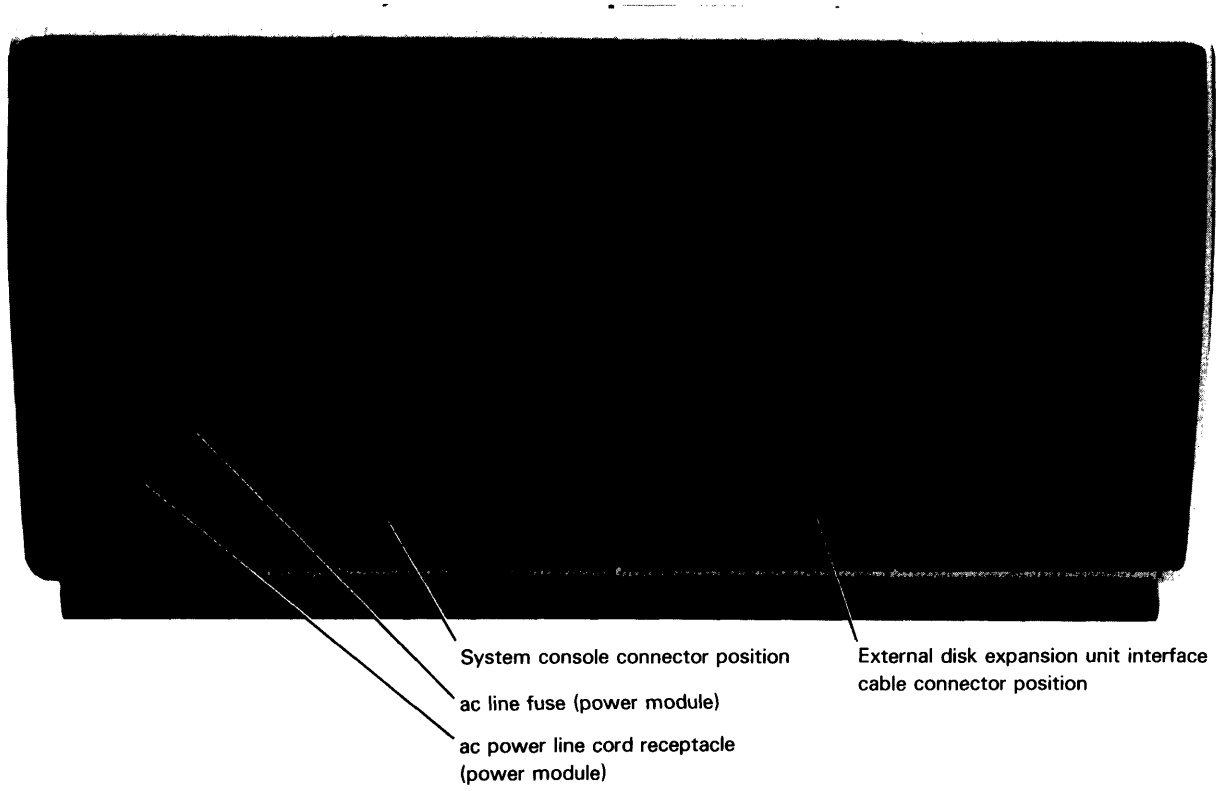


Figure 9-19 Model 20 and Model 30 system modules (rear view)

Line Fuses

Power fuses, located on the rear of each power module and the tape module as shown in (Figures 9-17 and 9-18), protects the ac line input. Line power to the Model 20 and Model 30 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, system console cable, and I/O cables connect the system unit and their components to ac line voltage and input/output devices. In addition, in the Model 30, the hardware floating point card connects to the system processor unit card through an internal cable. Internal interface and power cables connect 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 9-19. 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 20 and Model 30 SPU directly to an optional terminal or system console device. One end of this cable contains a 50-pin connector that plugs onto the asynchronous interface port (A connector) of the SPU card as shown in Figure 9-19. The other end contains a connector that plugs into a mating connector on the system console device.

Hardware Floating Point Interconnection Cable

This cable carries protocol signals for synchronizing floating point operations between the SPU card and the hardware floating point card in the Model 30. Each end of this cable contains a connector that plugs onto the C connectors of both cards, as shown in Figure 9-20.

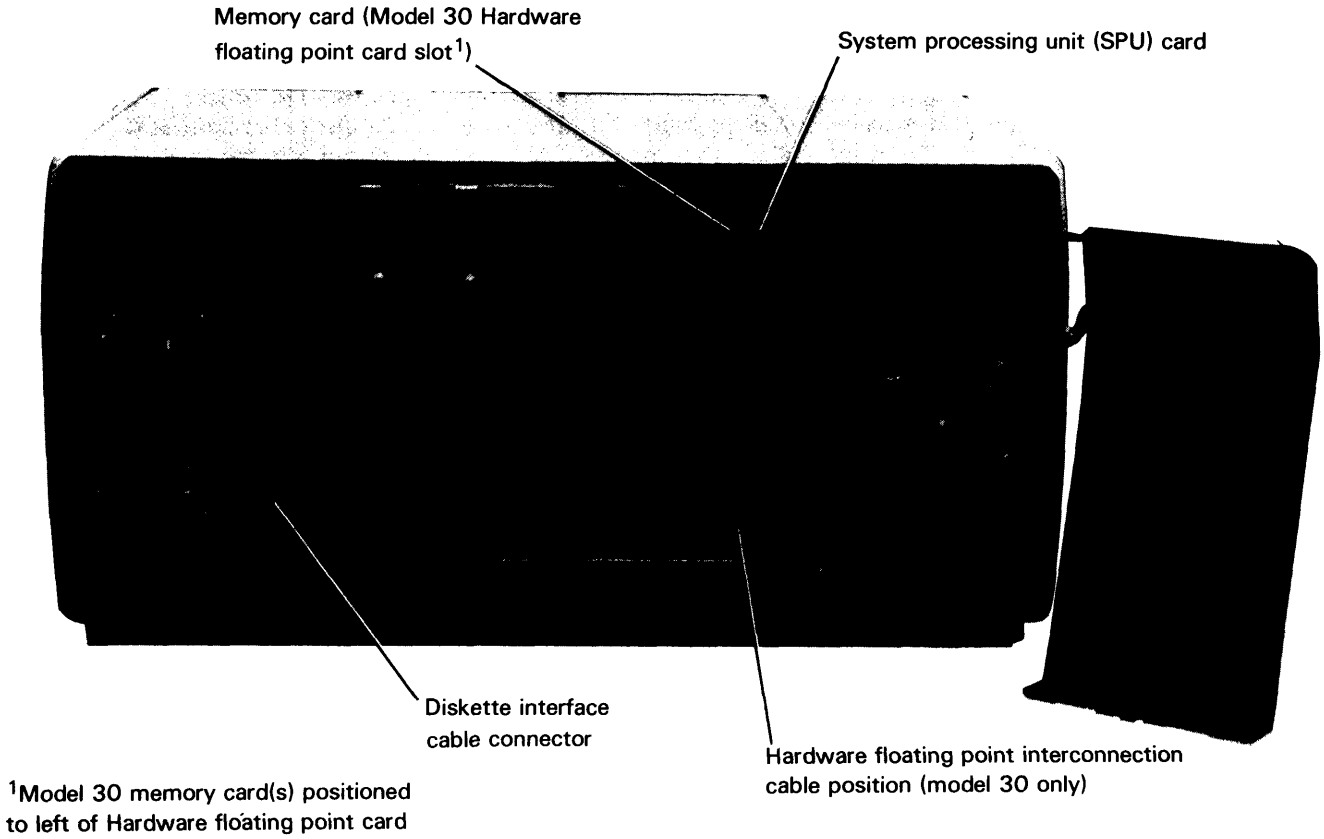
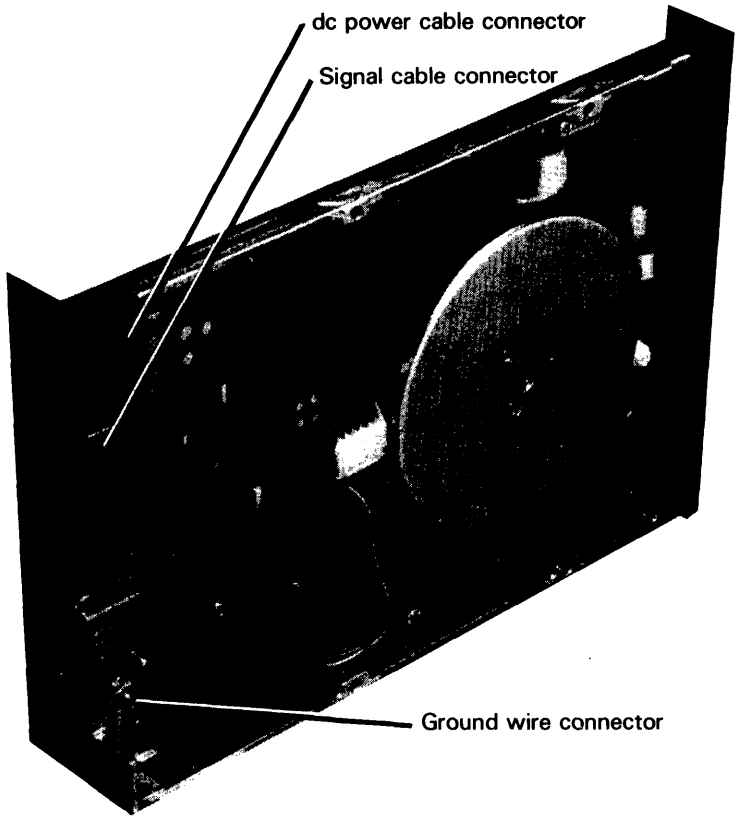


Figure 9-20 Model 20 and Model 30 system module (front view, covers removed)

Diskette Interface Cable The diskette interface cable connects the diskette interface to one or two diskette drives. One end of this cable contains a 50 pin connector that plugs onto the C connector of the diskette interface card, as shown in Figure 9-20. The other end contains two connectors that plug into mating connectors located on the rear of the diskette drives, as shown in Figure 9-21.

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 9-15. The other end contains two 4-pin connectors that plug into mating connectors located on the rear of the diskette drives, as shown in Figure 9-21. In addition, a safety ground wire connects J4 of the diskette module's backpanel (shown in Figure 9-15) to J3, located on the rear of the diskette drives (shown in Figure 9-21).

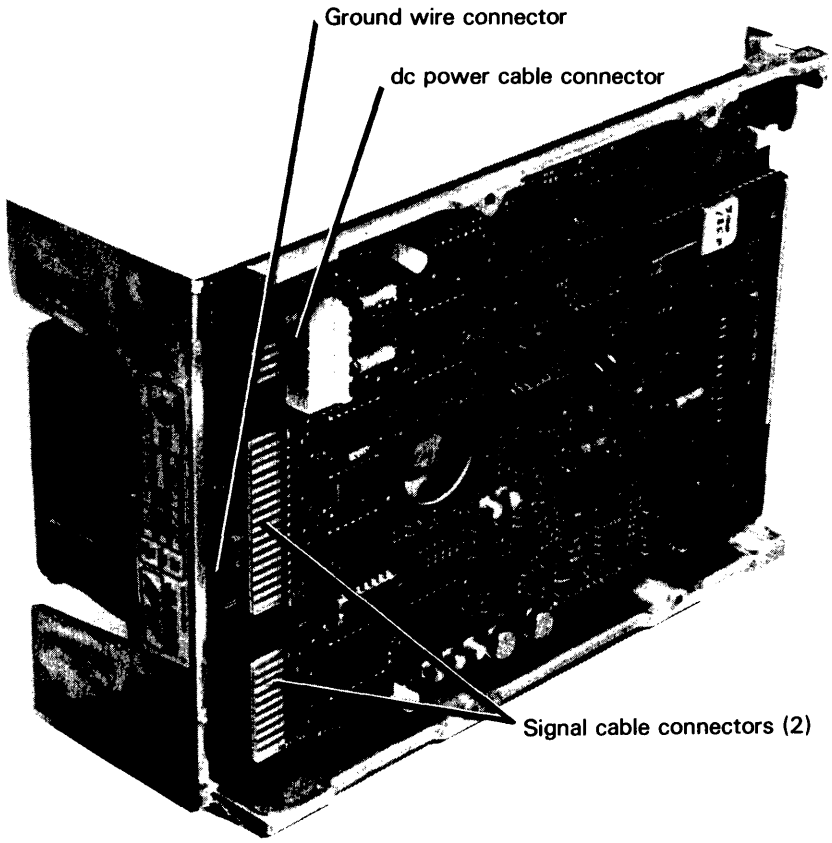


DG-25833

Figure 9-21 Diskette drive unit (rear view)

Disk Interface Cable The disk interface cable connects the 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. 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 9-22. 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 9-16. The other end contains a connector that plugs into a mating connector located on the rear of the disk drive, as shown in Figure 9-22. In addition, a ground wire connects J4 of the disk module's backpanel (shown in Figure 9-16) to a terminal located on the rear of the disk drive (shown in Figure 9-22). 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 9-22. 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.



DG-25834

Figure 9-22 Disk drive unit (rear view)

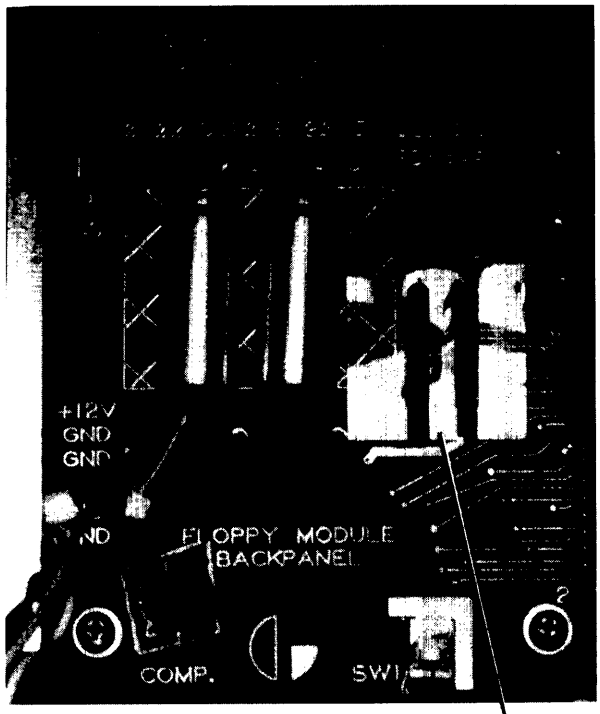
Disk Expansion Unit Cable The system's main disk module electrically connects to an expansion disk module (when present) with three interface cables. One of these cables is the disk interface cable described above that is located in the main disk module.

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 9-19 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 9-20.

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 9-23 and Figure 9-15. (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.

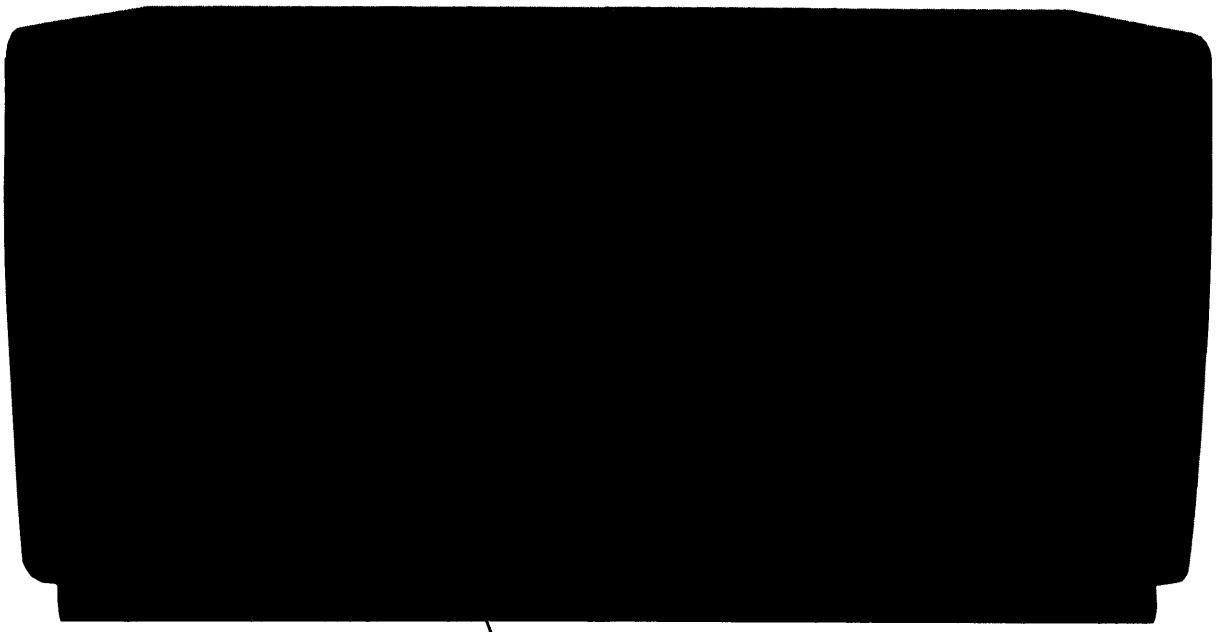


Extended microI/O bus cable connector position (tape module I/O bus connector plugs onto top 16 pins of the 20 pin provided)

Figure 9-23 Diskette module, internal view of backplane

Input/Output Cables

Input/output cables connect 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 9-24. The other end contains a connector that plugs into a mating connector on the device the interface is controlling.



NOTE *Unused connector positions are covered with a metal plate fastened to the modules cage and a plastic insert that snaps into place on the plastic rear panel.*

Figure 9-24 Input/output cable connector positions

Page A-2

Delete reference to part number, the second-to-last line:

Related Drawings

A

The table below lists Model 20 and Model 30 engineering drawings

Part Number	Part	Schematic	Illustrated Parts List
005-017609	System Processor Unit (SPU) card with firmware floating point	001-003078	016-001321
005-017611	System Processor Unit card with hardware floating unit interface	001-003078	016-001321
005-020324	System Processor Unit card with hardware floating unit interface and commercial instruction set	001-003078	016-001321
005-016544	Hardware Floating Point Unit card	001-002863	016-001133
005-020323	Hardware Floating Point Unit card with commercial instruction set	001-002863	016-001133
005-009663	SPU to FPU interconnection cable	—	018-000113
005-019538	256 Kbytes Memory card	001-003136	016-001448
005-019537	512 Kbytes Memory card	001-003136	016-001448
005-019593	Diskette module backpanel	001-003345	016-001457
005-019698	Diskette interface card	001-003343	016-001455
005-021149	Diskette data cable	001-003421	018-001510
005-019331	Diskette power cable	—	018-001488
005-019591	Diskette drive assembly	—	—
005-008152	Microl/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 20 and Model 30 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 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.

Table B-1 Diagnostic commands (Continued)

Bits 11-15	Command	Description
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.*

Execution Times for Commercial Instructions

C

C/30 commercial instruction execution times

Instruction	Overhead	0	1	2	3	4	5	6	7
LDI	67	85+A	100+A	85+A	100+A	85+A	90+B	56+C	26+D
LDIX	165+I +30F	85F+A	100+A +85F'	85F+A	100+A +85F'	85F+A	90F+B	—	—
STI	491+I	356	356	356	356	356	355	—	—
STI*	—	—	—	—	—	—	—	155+E	230
STIX	541F+ 103+I	356F	356F	356F	356F	356F	355F	—	—
LSN	J	38 +40N	36 +40N	29 +40N	27 +41N	17 +37N	32 +37N	46 +11T	46 +11T

**If the FPAC = 0, the overhead will be 110 CPU cycles, and there will be no interrupts.*

NOTES

A = 15*(Number of digits − 1)

B = 35*((Number of digits + 1)/2)

C = 6*(Number of bytes in the string)

D = 16*(Number of words in the destination)

E = 12*(Number of bytes in the string)

F = Number of FPACs to convert

F' = (Number of FPACs to convert) − 1

I = 40*(Number of program interrupts taken)

J = 15*(Number of program interrupts taken)

N = Number of digits — 1

T = Number of bytes

The overhead — in CPU cycles — for the Edit instruction is 23 + 13*(number of subinstructions executed). The following table lists the number of CPU cycles required for each Edit subinstruction.

DEND 4	DNDF 20	DSTK 24	DDTK 20
DSSZ 1	DSSO 1	DSTZ 1	DMVO 35 + 25/digit
DMVN 35 + 25/digit	DSTO 1	DINT 13	DAPT 3
DMVC 39 + 16/digit	DMVA 49 + 28/digit	DINS 14	DICI 15
DADI 15	DASI 24	DMVF 22 + 17/digit	DIMC + 7/digit
DMVS 17 + 38/digit	DAPU 14		

Maximum number of interrupts:

LDI — 0

LDIX — 4

STI — 3

STIX — 12

LSN — 1 per digit (byte)

EDIT — 1 per subinstruction

Compatibility With ECLIPSE Line Computers

D

The microECLIPSE™ based computers are compatible with the ECLIPSE line of computers, up to and including the ECLIPSE S/140 series computers. Any program currently running on an ECLIPSE S/140 computer will run on a microECLIPSE™ based computer with the following changes:

Unique features — Data In Status returns the status of the addressed device and places this data into the specified accumulator.

Emulator trap — The CPU in the Desktop Generation Model 20 and Model 30 systems has a hardware provision for instruction emulation. If the CPU encounters an undefined instruction while operating in the mapped mode, it automatically makes a jump through location 11_8 — provided that the contents are not zero. This location can contain the indirect location of an emulator routine.

Execution timing — The program may not be dependent on instruction execution times or I/O transfer times. Times for the ECLIPSE S/140 series computers may be faster than a microECLIPSE™ based computer depending upon the application.

Reserved memory locations — Memory location 11 is now the location that the processor will jump to for an emulator trap and should contain the address fault handler routine.

Virtual console — Commands are entered on a terminal keyboard.

Automatic increment/Automatic decrement locations — Memory locations 20_8 to 27_8 and 30_8 to 37_8 are not available for this purpose on microECLIPSE™ based computers.

No load always skip — This is an undefined operation code for the Desktop Generation Model 20 and Model 30; therefore, it results in an emulator trap.

Floating-point manipulation — The return address pushed during a floating-point trap is the address of the instruction following the instruction that caused the trap.

The floating-point program counter is valid only when ANY is set.

Bit 9 of the FPSR is the resume bit. It should be ignored and not modified when saving and restoring the floating-point status register. When initializing the floating-point status, this should be set to zero.

Stack — No underflow protection is automatically provided. This may be accomplished with a user subroutine if desired.

Bit 0 of the stack pointer will be set on a stack overflow if the return block pushed by the stack overflow routine wraps around from the top location (77777_8) to the bottom location 0 of memory.

MAP — If an instruction changes the current map state, the next instruction will be fetched from and executed in the new map state. See **DOA MAP.**

During a MAP fault, the program counter produces unpredictable results.

There are four user maps available.

Any attempt to read beyond the maximum physical address space will return undefined data. Any attempt to write beyond the maximum physical address space will have no effect.

Error checking — Desktop Generation Model 20 and Model 30 computers check for parity errors only.

Instructions

DIA MAP — Bit 0 will contain the extra map select bit; bit 1 will be set to 0 if the map is off, and will be set to 1 if the map is on.

No validity traps occur on map single-cycle references.

DIVS and DIVX — Carry will be set to one only if an overflow condition occurs.

IORST — Will not affect any bits in the floating-point status register.

SKP — The AC field (bits 3 and 4) must be zero.

If the device specified by the device code is nonexistent, busy and done equal 0.

LMP — When I/O protection is on, a map fault occurs and no accumulators are altered.

AC0 must be equal to zero or three.

NIO — The AC field (bits 3 and 4) must be zero.

XOP1 — Operates like the *Extended Operation* instruction except that it adds 32 to the entry number before it adds the entry number to the XOP origin address.

Index

Within the index, the letter "f" following a page entry indicates "and the following page"; the letters "ff" following a page entry indicate "and the following pages". The letter "t" following a page entry indicates that a table resides on the page. ?, virtual console outputs a 3-9

A

D

E

J

K

L

M

S

T

U

V

W

X

moisten & seal

Documentation Comment Form

Manual Title _____

Manual No. _____

Your Name _____

Your Title _____

Company _____

Street _____

City _____ State _____ Zip _____

Please help us improve our future publications by answering the questions below. Use the space provided for your comments. Thank you.

	Yes	No
Is this manual easy to read?	☐	☐
Is it easy to understand?	☐	☐
Are the topics logically organized?	☐	☐
Is the technical information accurate?	☐	☐
Can you easily find what you want?	☐	☐
Does it tell you everything you need to know?	☐	☐
Do the illustrations help you?	☐	☐

If you wish to order manuals, contact your sales representative or dealer.

Comments:

Date

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 26 SOUTHBORO, MA. 01772

Postage will be paid by addressee:

Data General

ATTN: Design Services (E219)
4400 Computer Drive
Westboro, MA 01581