Data General

ECLIPSE® S/280
Assembly Language Programming

ECLIPSE® S/280
Assembly Language Programming

DataGeneral

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.

Ordering No. 014-000689

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

Preface

This manual and its companion manual, *16-Bit Real-Time* ECLIPSE® *Assembly Language Programming* (DGC No. 014-000688), together provide extensive information for assembly language programmers working with the ECLIPSE® S/280 computer.

The *16-Bit Real-Time* ECLIPSE *Assembly Language Programming* manual explains the 16-bit processor-independent concepts, functions, and instruction set. The ECLIPSE *S/280 Assembly Language Programming* manual describes the S/280-dependent information such as physical memory size, address translation implementation, resident I/O devices, and instruction execution times.

Manual Organization

The ECLIPSE *S/280 Assembly Language Programming* manual contains eight chapters and five appendices.

Chapter 1, "Technical Summary," explains the system components and functions available on the ECLIPSE S/280 computer.

Chapter 2, "Memory Reference and Stack Management," summarizes memory referencing and the stack instructions.

Chapter 3, "Data Manipulation," summarizes fixed- and floating-point formats and instructions, and offers programming suggestions for accelerating data movement.

Chapter 4, "Program Flow Management," explains program flow, and interrupt and fault handling.

Chapter 5, "Device Management," explains the ECLIPSE S/280 I/O interrupt system, the resident I/O devices, and applicable instructions.

Chapter 6, "Memory and System Management," describes the S/280 address translation facilities, memory protection features, and special system functions for the operating system designer.

Chapter 7, "Virtual Console," describes the virtual console features and commands through which the operator interacts with the S/280 computer.

Chapter 8, "Powerup and Initialization," presents the S/280 powerup sequence, programming for powerfail/autorestart situations, and the initial state of the S/280 computer after power-up, system reset, and execution of an I/O Reset instruction.

Appendix A, "Instruction Summary," lists the complete S/280 instruction set alphabetically by assembler mnemonic.

Appendix B, "Instruction Execution Times," presents the typical execution time for each S/280 instruction.

Appendix C, "Register Fields," presents tabular data for the various program-accessible registers.

Appendix D, "Standard S/280 I/O Device Codes," lists Data General's standard device codes.

Appendix E, "Compatibility with Earlier ECLIPSE Computers," discusses the programming differences between the S/280 computer and other 16-bit Real-Time ECLIPSE computers.

Conventions and Abbreviations

In this manual, the following conventions and abbreviations represent assembler statements for instructions.

MNEMONIC — Uppercase characters indicate a literal argument (command mnemonic) in an assembler statement. When you include a literal argument with an assembler statement, use its exact form.

argument — Lowercase italic characters indicate a variable argument in an assembler statement. When you include the argument with an assembler statement, substitute a literal value for the variable argument.

[argument] — Square brackets enclosing an argument indicate that it is optional. Do not type the square brackets; they only indicate that you have a choice.

ac — The *ac* abbreviation indicates a fixed-point accumulator.

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

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

fpac — The *fpac* abbreviation indicates a floating-point accumulator.

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

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

In addition, diagrams in the following format show the arrangement of 16 bits in a word or fixed-point accumulator.

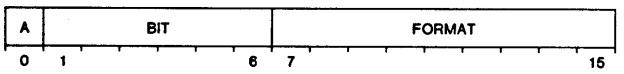


Table of Contents

Chapter 1

Technical Summary

The ECLIPSE S/280 enhances the high-end of the 16-bit real-time ECLIPSE computer family by offering superior performance in the small system environment. The ECLIPSE S/280 uses the full 16-bit real-time instruction set as presented in *16-Bit Real-Time ECLIPSE Assembly Language Programming* (DGC No. 014-000688). This set includes instructions to manipulate fixed-point data — including characters — and floating-point data; to define and manipulate stacks; to alter program flow; to control system input and output; and to manage memory.

The S/280 systems support all the new NOVA-ECLIPSE line peripherals as well as most older ones. Input/output to these peripherals occurs over the NOVA-ECLIPSE I/O bus and an optional burst multiplexor channel. The NOVA-ECLIPSE I/O bus includes data channel and programmed input/output facilities.

The data channel facility supports data transfers between memory and medium-speed devices such as disk drives, diskette drives, tape transports, and communications processors. The programmed I/O facility supports single-word transfers to and from low-speed devices such as display terminals and printers. The optional burst multiplexor channel transfers blocks of data between memory and high-speed disks and tape drives.

System Overview

The S/280 system incorporates five systems:

- A *memory system* providing logical-to-physical address translation, a cache for look-ahead/look-behind memory buffering, up to 2 Mbytes of main memory, and memory error checking and correction.
- A *processing system* providing ECLIPSE S/280 instruction decode and execution and a stack management facility.
- An *input/output system* providing facilities for control and data transfer between NOVA/ECLIPSE peripherals using the standard data channel, the programmed I/O facility, and the optional burst multiplexor channel.
- A *power system* providing power, diagnostic functions, and optional battery backup.
- A *system console* providing a virtual (soft) system console for user interaction with, and control of, the system.

Figure 1.1 shows the interconnection of these systems.

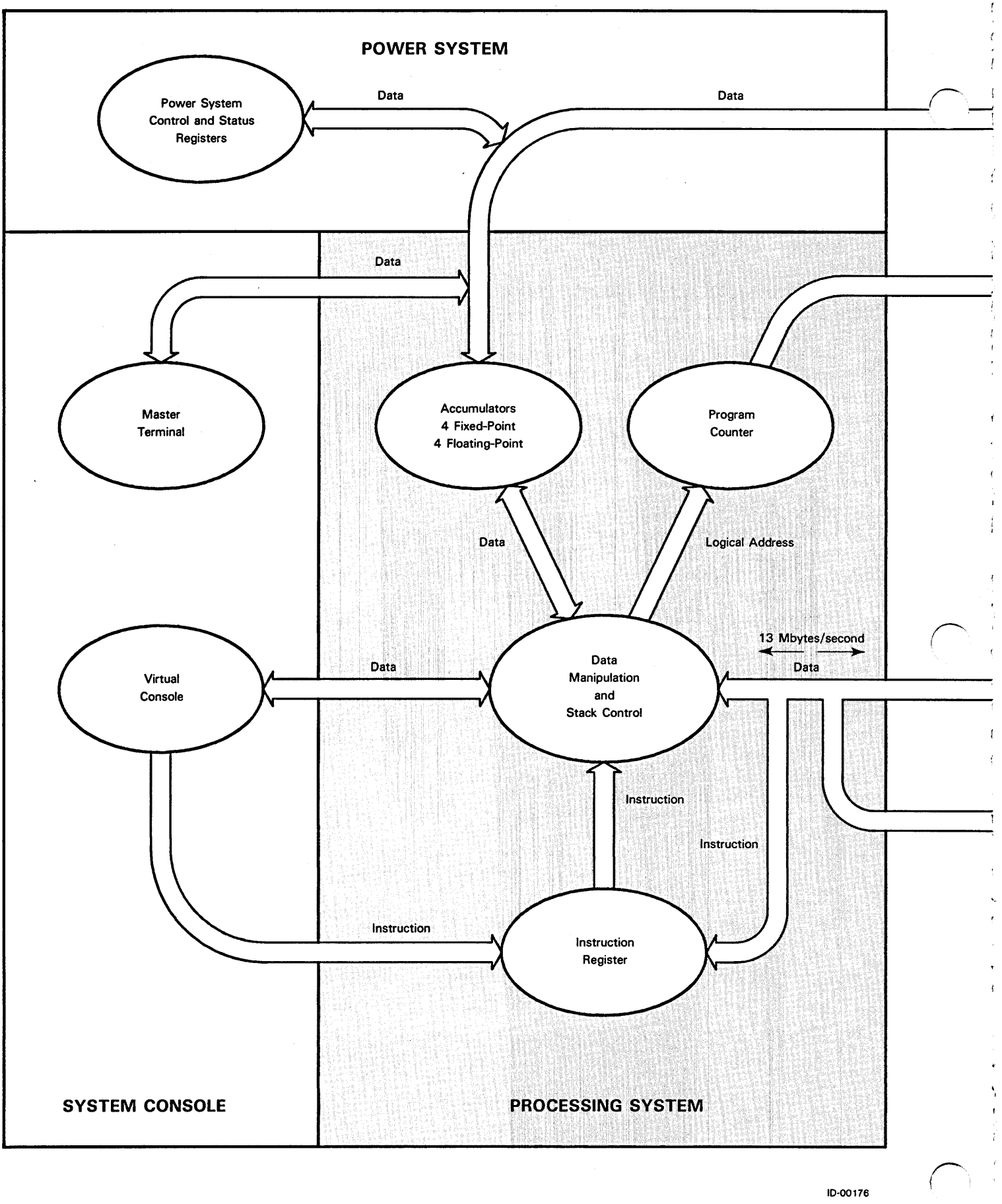
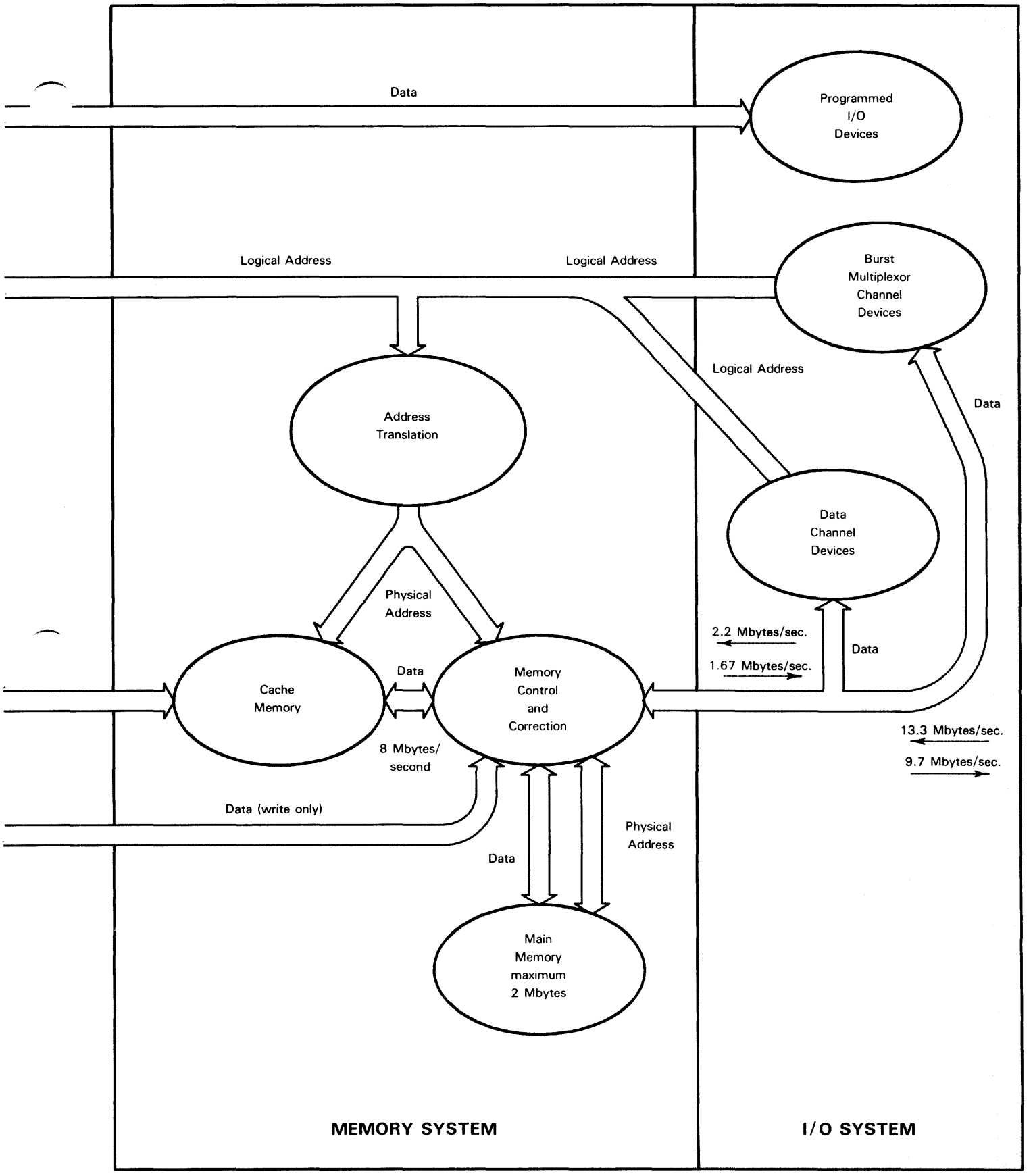


Figure 1.1 S/280 computer system


Data
Programmed I/O Devices
Logical Address
Logical Address
Burst Multiplexor Channel Devices
Logical Address
Data
Address Translation
Data Channel Devices
Physical Address
2.2 Mbytes/sec.
1.67 Mbytes/sec.
Data
Cache Memory
Data
Memory Control and Correction
13.3 Mbytes/sec.
9.7 Mbytes/sec.
8 Mbytes/ second
Data (write only)
Physical Address
Data
Main Memory maximum 2 Mbytes
MEMORY SYSTEM
I/O SYSTEM

Memory System

The S/280 memory system contains address translation facilities, a cache, a memory control unit, and memory modules.

Address Translation Facilities

The S/280 architecture has 64 Kbytes of logical address space available for the programmer and from 512 Kbytes to 2 Mbytes of physical address space for storage. Because the logical address space is smaller than the physical address space, the S/280 uses address translation to store the 2 Kbyte pages of each logical address space in physical memory.

The S/280 memory system has address translation facilities for memory access by user, data channel, and burst multiplexor channel processes. These facilities include program-accessible map tables which store the address translations for four user processes, four data channel processes, and a burst multiplexor process.

In addition to address translation, the facilities also perform all the hardware checks required by the programmable protection system. These checks include validation of process access, write access, I/O access, and indirection. The program can enable protection traps to occur when any of these checks detects a fault condition.

Cache

The cache functions as both a look-ahead and a look-back buffer for the processing system. This reduces the time the processing system needs to access main memory. The input/output system does not use the cache; it accesses main memory directly.

The cache stores 4 Kbytes of memory data. The data is organized into two sets of 256 four-word (8-byte) blocks; each block is associated with certain blocks of four contiguous locations in main memory. This means that cache blocks cannot contain arbitrary locations from memory. As Figure 1.2 shows, up to 1024 pages comprise memory, and each page contains 256 four-word blocks. Corresponding blocks in each memory page are mapped into the same block in each cache set. Thus, block 0 of either cache set can contain block 0 of any page in main memory. The cache keeps track of the memory page which supplied each cache block.

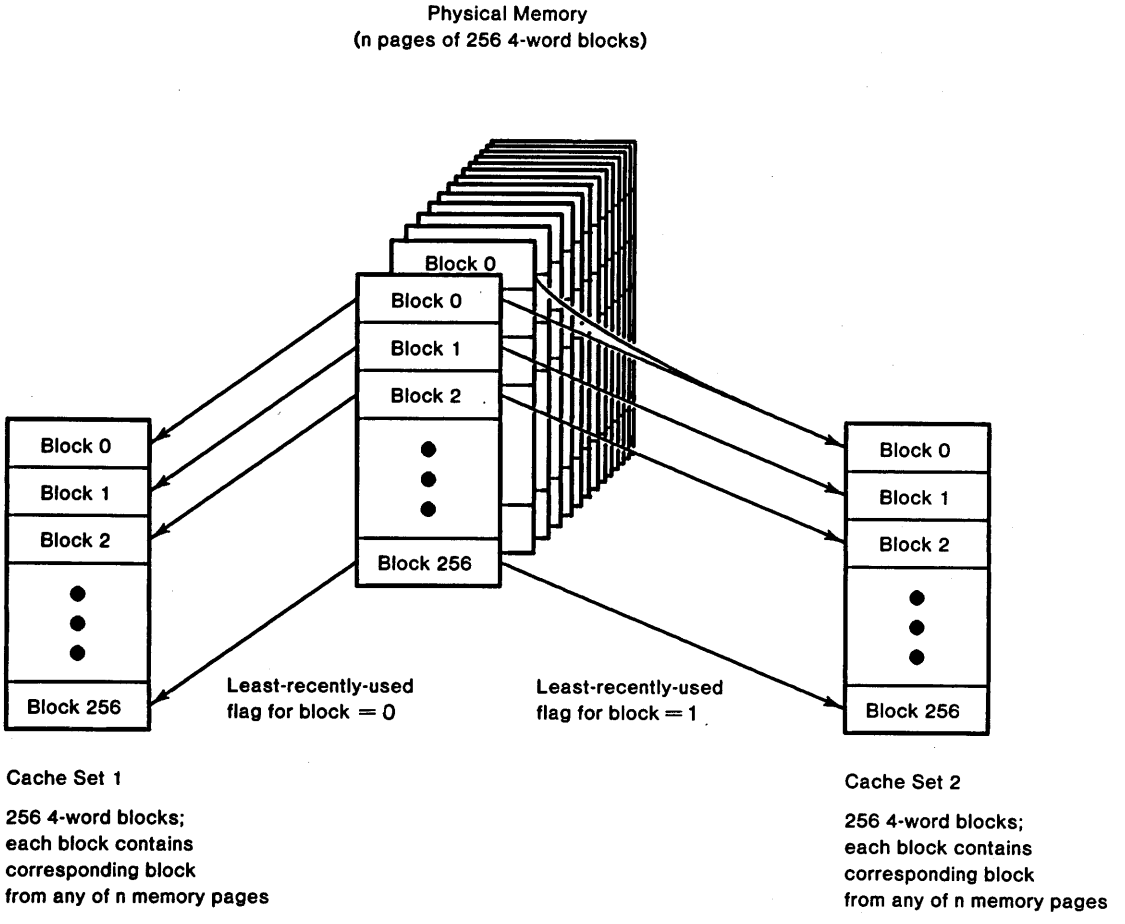


Figure 1.2 Memory-to-cache mapping

The cache also contains 256 least-recently-used (LRU) flags, one for each cache block. These flags indicate which set was NOT accessed the last time data was read from that cache block.

Whenever the central processor requests data from any block (*n*) of a memory page that is not currently in the cache (cache *miss*), the cache examines the LRU flag for cache block *n*. If the LRU flag is set to 1, the cache stores block *n* of the new page in block *n* of cache set 1; if the LRU flag is set to 0, the cache stores block *n* of the new page in block *n* of cache set 2. In either case, the cache overwrites the contents of cache block *n* with the new block (cache *fill*) and then updates the LRU flag for cache block *n*. Whenever a process requests data from a memory block *n* that is already in the cache (cache *hit*), the cache supplies the data from cache block *n* of the appropriate cache set and updates the LRU flag for cache block *n* appropriately.

Whenever a process requests that data be stored in block *n* of a memory page, the data is always written into main memory. If the block is currently in the cache, the cache also stores the data in block *n* of the least-recently-used cache set. This ensures that the cache and main memory contain the same data for the same block. If the block is not in the cache, the cache does nothing; in other words, the block is not loaded into the cache.

When a data channel device or a burst multiplexor device writes data to a block in main memory which is also in the cache, the cache invalidates the cache block. The next time a process requests data from this block, the cache retrieves the block from memory, thus ensuring that the correct data is used. The cache is also invalidated at powerup and when the processor enters virtual console mode.

Memory Control Unit

The memory control unit contains three ports: one for the processing system and two for direct transfers between memory and the I/O system. The I/O system's data channel uses one of these two ports and its burst mutiplexor channel uses the other. The control unit multiplexes the ports' access to the modules on a priority basis: the data channel has highest priority, the burst multiplexor channel has priority next, and the processing system has lowest priority.

Memory Modules

RAM modules are provided in three different sizes and organized into independent banks.

The 512 Kbyte module consists of four independent banks, each bank containing 16K pairs of double words. The 1 Mbyte and 2 Mbyte modules consists of two and four independent banks, respectively, each bank containing 64K pairs of double words. Each double-word location contains two 16-bit data words and seven check-code bits.

On modules with two banks (1 Mbyte), each bank contains every other pair of double words. This means that bank 0 contains words 0, 1, 2, 3, 10, 11, 12, 13, ...; bank 1 contains words 4, 5, 6, 7, 14, 15, 16, 17, ... On modules with four banks (512 Kbytes and 2 Mbytes), each bank contains every fourth pair of double words. This means that bank 0 contains words 0, 1, 2, 3, 20, 21, 22, 23, ...; bank 1 contains words 4, 5, 6, 7, 24, 25, 26, 27...; bank 2 contains words 10, 11, 12 ,13, 30, 31, 32, 33,...; bank 3 contains words 14, 15, 16, 17, 34, 35, 36, 37....

This arrangement allows memory operations to overlap so that one address can access one double word or two double words. The address provided to the memory modules specifies one pair of double words. Control signals select one or both of the double-words for the memory operation. In addition, this arranagement can be used to speed up memory access, since one bank can be addressed while data is being written into or read from another bank.

Memory Refresh and Error Correction

Because the S/280 memory modules consist of dynamic RAM, the memory control unit must refresh the modules. It performs a refresh cycle every 16 microseconds.

The memory control unit checks the memory modules for errors when it performs a refresh and when it reads a memory location. Its error correction logic corrects all single-bit errors, detects all double-bit and some triple-bit errors, and stores error information. The program can enable and disable the error correction logic and read error information.

When the memory control unit refreshes a module, it also reads one double word and runs that double word through its error correction logic. This process is called *sniffing*. If the error correction logic detects a single-bit error while sniffing, it corrects the error and writes the corrected double word back to memory.

When the memory control unit reads a double word from a module, it runs that double-word through its error correction logic. If this logic detects a single-bit error, it corrects the error before sending the double word to the requestor.

Chapter 6 presents information for programming the error correction logic.

Processing System

The processing system contains a central processor (CP) and an optional floating-point processor (FP).

Central Processor

The central processor (CP) executes the standard 16-bit ECLIPSE instruction set, the character instruction set, and the floating-point instruction set. It fetches the instruction, which is addressed by its 15-bit program counter, from main memory and decodes the instruction into microinstructions. These microinstructions then produce the control signals that run the central processor's arithmetic logic unit, access memory, perform I/O transfers, and send floating-point instructions to the floating-point processor, if this processor is present.

The central processor accelerates instruction execution with a three-stage pipeline. This pipeline allows the central processor to overlap instruction fetches with instruction execution when instructions are sequential in memory. The central processor clears (*flushes*) the pipeline when a non-skip instruction alters program flow.

On powerup, the central processor automatically executes a confidence test. The tests check the basic functions of the processing, memory, and I/O systems to ensure that software can be loaded.

The central processor contains the following program-accessible registers for manipulating data and managing the system:

four 16-bit fixed-point accumulators,
four 64-bit floating-point accumulators,
one 16-bit floating-point status register.

It uses the following program-accessible reserved memory locations for managing stacks:

one 16-bit stack limit register,
one 16-bit stack pointer register,
one 16-bit frame pointer.

NOTE: *The central processor's floating-point registers are used only when the floating-point processor is not present.*

Optional Floating-Point Processor

The optional floating-point processor operates in parallel with the central processor. It executes the ECLIPSE 16-bit floating-point instruction set approximately six and one-half times faster than the central processor. When the central processor fetches a floating-point instruction, it sends the instruction to the floating-point processor for execution. If the instruction references memory, the central processor also initiates the reference. While the floating-point processor executes the instruction, the central processor continues to fetch and execute instructions until it fetches another floating-point instruction.

In addition to executing instructions, the floating-point processor also performs hardware checks on floating-point arithmetic operations. These checks include detection of exponent overflow and underflow, mantissa overflow, and division by zero. When any of these fault conditions are detected, the program can enable a floating-point trap to occur.

The floating-point processor contains the following program-accesssible registers for manipulating and managing floating-point data:

four 64-bit floating-point accumulators,
one 16-bit floating-point status register.

Input/Output System

The input/output (I/O) system is electrically and program compatible with the NOVA-ECLIPSE I/O bus and the BMC (burst multiplexor channel) bus used by the 16-bit Real-Time ECLIPSE family. This means that the S/280 computer system supports the family of standard Data General NOVA-ECLIPSE line peripherals.

I/O Transfers

Both the data channel and the burst multiplexor channel transfer data directly to and from the memory system via an I/O control unit. Data on these channels never passses through the cache or the processing system.

The data channel transfers data to and from memory at a rate of up to 1.67 Mbytes per second on output and up to 2.2 Mbytes per second on input. The burst multiplexor channel transfers blocks of data to and from memory at the rate of up to 9.7 Mbytes per second on output and up to 13.3 Mbytes per second on input.

The programmed I/O facility transfers words between the central processor's accumulators and I/O devices. In addition to transferring data from low-speed devices, the programmed I/O facility is used to set up the transfers for higher-speed channels.

Resident I/O Devices

All S/280 computers have four basic I/O devices: a programmable interval timer, a real-time clock, and asynchronous input and output lines.

The *programmable interval timer* provides a time base independent of central processor timing. This time base can be programmed to intitiate program interrupts at a fixed programmable multiple of an interval in one of four jumper-selected ranges. The fixed intervals are multiples of 1, 10, or 100 microseconds, or 1 millisecond, depending on the selected range.

The *real-time clock* provides a programmable selection of precise time bases. Four frequencies are available: 10 Hz, 100 Hz, 1000 Hz, and line frequency.

The *asynchronous input/output lines* provide the communications link between the central processor and the master terminal. They support asynchronous transfers at jumper-selected rates ranging from 50 to 38,400 baud.

Power System

The power system converts ac line power into the dc voltages necessary to run the components in the computer chassis. In systems with the battery backup option, the power system uses battery power to supply the necessary voltages during an ac power failure.

A microprocessor-based controller governs the operation of the power system. At initial powerup, the controller runs a self-test to ensure proper operation. Then the controller begins to sequence up the output voltages, while monitoring for faults. If the controller detects a fault, it displays a fault code on the front console lights and cuts powers.

The controller contains several 8-bit program-accessible registers which are useful for monitoring system status, changing operating modes, and performing diagnostic functions.

System Console

The S/280 system consists of the resident asynchronous input/output line, the terminal connected to it, and virtual console firmware in the central processor. The system console terminal is the *master terminal.*

When the master terminal operates under the control of the operating system software (run mode), it functions as a system operator's terminal. In this mode, the operator controls the system with operating system commands and utilities.

When the master terminal operates under the control of the virtual console firmware (virtual console mode), it operates as a diagnostic tool and program debugger. In this mode, the operator uses virtual console commands to do such things as access system registers and main memory, to single-step through a program instruction by instruction and to run the confidence tests.

Chapter 2

Memory Reference and Stack Management

The S/280 processing system references memory and handles stacks using the same methods as the other members of the 16-bit Real-Time ECLIPSE computer family. This chapter summarizes methods for referencing memory, lists the S/280 memory locations reserved for special functions, and discusses instructions for manipulating stacks. The *16-Bit Real-Time* ECLIPSE *Assembly Language Programming* manual provides additional information on memory reference and stack manipulation.

Memory Reference

When the central processor executes a memory reference instruction, it calculates an effective logical address. First, the central processor calculates an intermediate address using the address mode specified by the instruction. The three possible address modes are

- *Absolute addressing* — the intermediate address comes from the the instruction's displacement field.
- *Program-counter-relative addressing* — the immediate address is the sum of the program counter and the instruction's displacement field.
- *Accumulator-relative addressing* — the immediate address is the sum of an accumulator and the instruction's displacement field.

Next, the central processor resolves any indirection. The S/280 central processor allows up to 14 levels of indirection when user address translation is enabled and an infinite number of levels when user address translation is disabled. The central processor sends the resolved logical address (the effective address) to the address translation facilities. For information on these facilites, refer to Chapter 6, "Memory and System Management."

Reading and writing to a non-existent memory location can produce undesirable results. The first time a process reads such a location, it receives meaningless data, and the cache writes the same meaningless data into the appropriate cache block. If a process later writes data to the same location, the cache stores the data since it contains the referenced location. As a result, the nonexistent memory location may seem real and memory sizing routines may size memory incorrectly.

Reserved Memory Locations

Within lower page 0, logical locations 0 through 7 and 40 through 47 are reserved for storing data which has special meaning. Most of these locations store addresses, some of which are indirectable and some of which are not. (See Table 2.1.) The operating system or user program can access any of the locations using absolute mode addressing with any memory reference instruction.

The operating system accesses locations 0 through 7 when it handles I/O interrupts and system calls. Locations 40 through 47 are accessed by either the operating system or by user programs when they use the stack facility, extended operation instructions, or floating-point instructions. Since the operating system and each user can have their own stack, extended operation tables, and floating-point handler, each user map table can translate these logical addresses to different physical locations.

The S/280 system does not have the autoincrementing and autodecrementing locations (20_8 through 37_8) of the earlier 16-bit Real-Time ECLIPSE computers.

Table 2.1 lists the S/280 reserved memory locations and their functions.

Stack Management

The S/280 processing system supports the standard 16-bit ECLIPSE stack facility and provides both stack overflow and stack underflow protection.

The processing system checks for stack overflow for a *Save* (SAVE) or a *Modify Stack Pointer* (MSP) instruction *before* executing the instruction. For any other instruction that pushes data onto the stack, it checks for overflow *after* executing the instruction. It also checks for stack underflow *after* executing any instruction that pops data off the stack.

Stack Instructions

Table 2.2 lists the instructions for manipulating stacks.

Location (octal)	Name	Function
0	I/O return	Return address from I/O interrupt. Also address of first instruction of autorestart routine.
1	I/O handler	Address of the I/O interrupt handler. Indirectable.
2	System call handler	Address of the system call instruction handler. Indirectable.
3	Protection fault handler	Address of the protection fault handler. Indirectable.
4	Vector stack pointer	Address of the top of the vector stack. Nonindirectable.
5	Current mask	Current interrupt priority mask.
6	Vector stack limit	Address of the last normally usable location in the vector stack. Nonindirectable.
7	Vector stack fault handler	Address of the vector stack fault handler. Indirectable.
40	Stack pointer	Address of top of stack. Nonindirectable.
41	Frame pointer	Address of frame reference within the stack. Nonindirectable.
42	Stack limit	Address of the last normally usable location in the stack. Nonindirectable.
43	Stack fault handler	Address of the stack fault handler. Indirectable.
44	XOP table	Address of the beginning of the Extended Operation table. Nonindirectable.
45	Floating-point fault handler	Address of the floating-point fault handler. Indirectable.
46-47	—	Reserved for future use.

Table 2.1 Reserved memory locations in logical address space $0\text{-}377_8$

Instruction Mnemonic	Action	Words Pushed or Popped	Words Required Beyond Stack Limit for Fault
FPOP	Floating-point pop	18	5
FPSH	Floating-point push	18	23
MSP	Modify stack pointer	—	—
POP	Pop multiple accumulators	1-4	5
POPB	Pop block	5	5
POPJ	Pop program counter and jump	1	5
PSH	Push multiple accumulators	1-4	6-9
PSHJ	Push jump to subroutine	1	6
PSHR	Push return address	1	6
RSTR	Restore	9	5
RTN	Return	5	5
SAVE	Save	5	10

Table 2.2 Stack instructions

NOTE: *The program counter pushed onto the stack when the address translators detect a validity or write-protection fault may not contain the address of the instruction that caused the fault.*

Chapter 3

Data Manipulation

The S/280 computer manipulates data with the same fixed- and floating-point operations as the other 16-bit Real-Time ECLIPSE computers. This chapter summarizes the data formats and instructions that perform these operations. For further information, refer to the the manual *16-Bit Real-Time* ECLIPSE *Assembly Language Programming* (DGC No. 014-000688).

Fixed-Point Manipulation

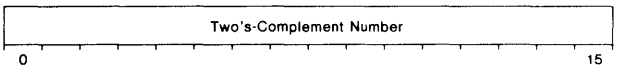
The fixed-point operations include arithmetic, logical, and decimal/byte operations.

Arithmetic Data Formats

Fixed-point arithmetic operations manipulate unsigned or signed two's-complement numbers. The accumulator formats for these data types are diagrammed below.

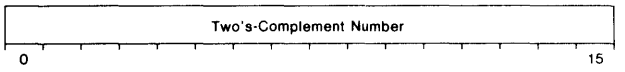
Unsigned Single-Precision Format

Word

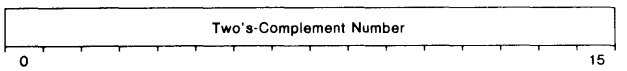


Unsigned Double Precision Format

Word 1

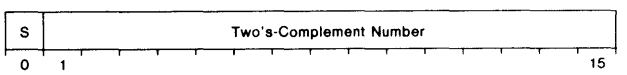


Word 2



Signed Single-Precision Format

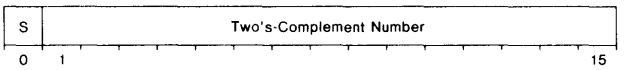
Word



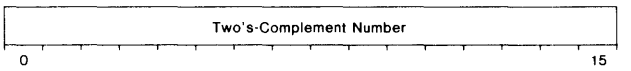
S = *sign*

Signed Double-Precision Format

Word 1



Word 2



S = *sign*

Arithmetic Instructions

Tables 3.1 through 3.5 list the instructions for moving and manipulating fixed-point data.

Instruction Mnemonic	Operation
BAM	Block move and add
BLM	Block move
ELDA	Extended load accumulator
ESTA	Extended store accumulator
LDA	Load accumulator
MOV	Move and skip
POP	Pop accumulators
STA	Store accumulator
XCH	Exchange accumulators

Table 3.1 Fixed-point data movement instructions

Instruction Mnemonic	Operation
ADC	Add complement
ADD	Add
ADDI	Extended add immediate
ADI	Add Immediate
BAM	Block move and add
INC	Increment
SUB	Subtract
SBI	Subtract immediate

Table 3.2 Fixed-point addition and subtraction instructions

Instruction Mnemonic	Operation
DIV	Unsigned divide
DIVS	Signed divide
DIVX	Sign extend and divide
HLV	Halve
MUL	Unsigned multiply
MULS	Signed multiply

Table 3.3 Fixed-point multiplication and division instructions

Instruction Mnemonic	Operation
ADC	Add complement with optional carry initialization
ADD	Add with optional carry initialization
AND	AND with optional carry initialization
COM	One's complement with optional carry initialization
INC	Increment with optional carry initialization
MOV	Move with optional carry initialization
NEG	Negate with optional carry initialization
SUB	Subtract with optional carry initialization

Table 3.4 Fixed-point initialize carry instructions

Instruction Mnemonic	Operation
ADC	Add complement with optional shift and optional skip
ADD	Add with optional shift and optional skip
DSZ	Decrement and skip if zero
EDSZ	Extended decrement and skip if zero
EISZ	Extended increment and skip if zero
INC	Increment with optional shift and optional skip
ISZ	Increment and skip if zero
MOV	Move with optional shift and optional skip
NEG	Negate with optional skip
SGE	Skip if ACS is greater than or equal to ACD
SGT	Skip if ACS is greater than ACD
SUB	Subtract with optional shift and optional skip

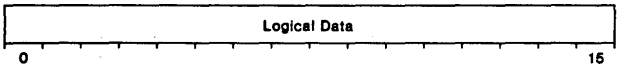
Table 3.5 Fixed-point shift and skip instructions

Logical Data Formats

Logical operations require binary data that begin on word boundaries. The accumulator formats for this data type are diagrammed below.

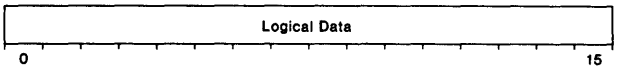
16-Bit Logical Format

Word

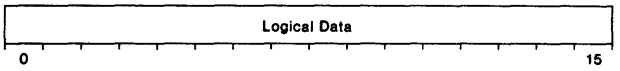


32-Bit Logical Data

Word 0



Word 1



Logical Instructions

Logical data is moved with the same instructions as fixed-point data. Refer to Table 3.1 for these instructions. Tables 3.6 and 3.7 list the instructions for manipulating logical.data.

Instruction Mnemonic	Operation
ANC	AND with complemented source
AND	AND
ANDI	AND immediate
COB	Count bits
COM	Complement
DLSH	Double logical shift
IOR	Inclusive OR
IORI	Inclusive OR immediate
LOB	Locate lead bit
LRB	Reset lead bit
LSH	Logical shift
NEG	Negate
SNB	Skip on zero bit
SZB	Skip on zero bit
SZBO	Skip on nonzero bit and set bit to one
XOR	Exclusive OR
XORI	Exclusive OR immediate

Table 3.6 Logical instructions

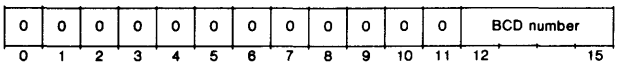
Instruction Mnemonic	Operation
AND	AND with optional shift and optional skip
COM	One's complement with optional shift and optional skip
DLSH	Double logical shift
LSH	Logical shift
NEG	Negate with optional shift and optional skip
SNB	Skip on nonzero bit
SZB	Skip on zero bit
SZBO	Skip on zero bit and set bit to one

Table 3.7 Logical shift and skip instructions

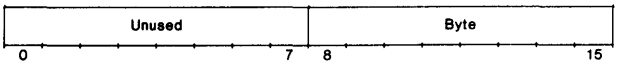
Decimal/Byte Data Formats

Decimal operations manipulate unsigned 4-bit binary-coded decimal (BCD) numbers. Byte operations move 8-bit bytes such as ASCII characters. The accumulator formats for these data types are diagrammed below.

Binary Coded Decimal (BCD) Format



Byte Format



Decimal/Byte Instructions

Decimal numbers can be moved using any of the instructions listed in Table 3.1. Table 3.8 lists the instructions for manipulating decimal numbers and performing hex shifts. Table 3.9 lists the instructions for moving characters.

Instruction Mnemonic	Operation
DAD	Decimal add
DSB	Decimal subtract
DHXL	Double hex shift left
DHXR	Double hex shift right
HXL	Hex shift left
HXR	Hex shift right

Table 3.8 Decimal and hex shift instructions

Instruction Mnemonic	Operation
ELDB	Extended load byte
ESTB	Extended store byte
LDB	Load byte
STB	Store byte
CMP	Character compare
CMT	Character move until true
CMV	Character move
CTR	Character translate and move or compare

Table 3.9 Byte movement instructions

Floating-Point Manipulation

Floating-point operations include arithmetic and numeric conversion operations.

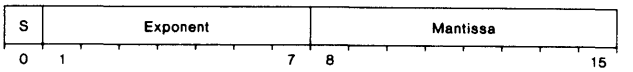
Floating-Point Data Formats

Floating-point operations manipulate normalized, signed numbers. These numbers are either single precision (32-bits) or double precision (64-bits). Single-precision numbers yield 6 to 7 significant decimal digits and double-precision numbers yield 15 to 17 decimal digits.

The accumulator formats for these numbers are diagrammed below.

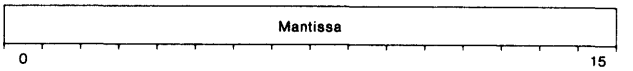
Single Precision Format

Word 1



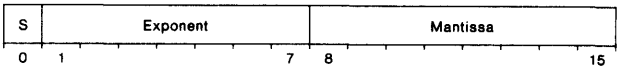
S = *sign*

Word 2



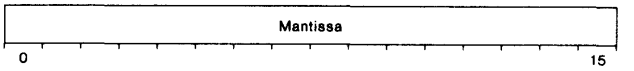
Double Precision Format

Word 1

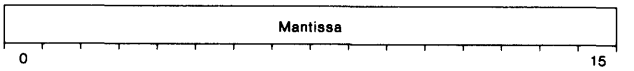


S = *sign*

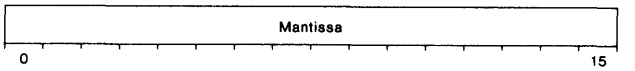
Word 2



Word 3



Word 4



Floating-Point Instructions

Tables 3.10 through 3.14 list the instructions for manipulating floating-point data.

Instruction Mnemonic	Operation
FAB	Compute absolute value (set sign of FPAC to 0)
FEXP	Load exponent (AC0<1-7> to FPAC<1-7>)
FFAS	Fix to AC (FPAC to AC)
FFMD	Fix to memory (FPAC to memory)
FINT	Integerize (FPAC)
FLAS	Float from AC (AC to FPAC)
FLMD	Float from memory (memory to FPAC)
FNEG	Negate (FPAC)
FNOM	Normalize (FPAC)
FRH	Read high word (FPAC<0-15> to AC0<0-15>)
FSCAL	Scale floating-point

Table 3.10 Floating-point-to-binary conversion operations

Instruction Mnemonic	Operation
FLDD	Load floating-point double
FLDS	Load floating-point single
FMOV	Move floating-point (FPAC to FPAC)
FSTD	Store floating-point double
FSTS	Store floating-point single

Table 3.11 Floating-point data movement instructions

Instruction Mnemonic	Operation
FAD	Add double (FPAC to FPAC)
FAS	Add single (FPAC to FPAC)
FAMD	Add double (memory to FPAC)
FAMS	Add single (memory to FPAC)
FSD	Subtract double (FPAC from FPAC)
FSMD	Subtract double (memory from FPAC)
FSMS	Subtract single (memory from FPAC)
FSS	Subtract single (FPAC from FPAC)

Table 3.12 Floating-point addition and subtraction instructions

Instruction Mnemonic	Operation
FDD	Divide double (FPAC by FPAC)
FDS	Divide single (FPAC by FPAC)
FDMD	Divide double (memory by FPAC)
FDMS	Divide single (memory by FPAC)
FHLV	Halve
FMD	Multiply double (FPAC by FPAC)
FMMD	Multiply double (FPAC by FPAC)
FMMS	Multiply double (memory by FPAC)
FMS	Multiply single (FPAC by FPAC)

Table 3.13 Floating-point multiplication and division instructions

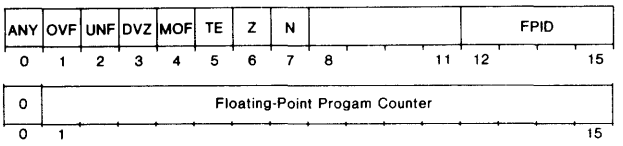
Instruction Mnemonic	Operation[1]
FCMP	Compare floating-point (set N and Z)
FNS	No skip
FSA	Always skip
FSEQ	Skip on 0 result (Z=1)
FSGE	Skip on greater than or equal to 0 (N=0)
FSGT	Skip on greater than 0 (Z=0)
FSLE	Skip on less than or equal to 0 (N=0 and Z=0)
FSLT	Skip on less than 0 (N=1)
FSND	Skip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

Table 3.14 Floating-point skip instructions

[1] *Parenthesis contain the value of relevant bit(s) in the floating-point status register. For more information, refer to the next section.*

Floating-point Status Register

Table 3.15 lists the instructions for manipulating the floating-point status register (FPSR). The accumulator format for the FPSR is diagrammed below.



Bits	Name	Contents or Function
0	ANY	If 1, one or more of bits 1-4 are set to 1.
1	OVF	If 1, exponent overflow occurred. The result is correct, except that the exponent is 128 too small.
2	UNF	If 1, exponent underflow occurred. The result is correct, except that the exponent is 128 too large.
3	DVZ	If 1, division by zero was attempted. The division operation was aborted, and the operands remain unchanged.
4	MOF	If 1, a mantissa overflow occurred.
5	TE	If 1, floating-point traps are enabled. Setting any of bits 1-4 to 1 causes a floating-point fault.
6	Z	If 1, the result is zero.
7	N	If 1, the result is negative.
8-11	—	Reserved for future use.
12-15	FPID	Floating-point model number. Should be 13_8 for firmware floating-point and 5_8 for hardware floating-point.
16	—	Reserved for future use.
17-31	Floating-point program counter	Floating-point program counter. In the event of a floating-point fault, this is the address of the floating-point instruction that caused the fault.

Instruction Mnemonic	Operation
FCLE	Clear errors (FPSR)
FLST	Load FPSR
FPOP	Pop floating-point state
FPSH	Push floating-point state
FSST	Store floating-point state
FTD	Floating-point trap disable (sets TE to 0)
FTE	Floating-point trap enable (sets TE to 1)

Table 3.15 Floating-point status register instructions

Accelerating Data Movement

Program performance can be enhanced by accelerating data movement operations as follows.

- Increase the number of cache hits.
- Avoid consecutive instructions that write to memory.

Increase Cache Hits

The cache stores 512 four-word blocks, each containing four contiguous words from main memory. Whenever the processor references memory, the cache checks to see if it has the cache block containing the referenced word. If the block is in the cache (cache hit), the cache either writes the word into the block or reads the word from the block, depending on the type of memory reference. If the block is not in the cache (cache miss), the cache's response depends on the type of reference. For a write operation, the cache does nothing, and the word is written directly to main memory; for a read operation, the cache retrieves the block containing the referenced word before reading it.

Keeping these conditions and operations in mind, the programmer can increase cache hits by

- Using sequential instruction flow whenever possible,
- Keeping instruction loops tight,
- Using program modules that require as little memory as possible.

Avoid Consecutive Writes

Since main memory's organization is double-word, memory reference instructions which write data to memory require one or more single-word write operations. A single-word write operation involves reading a double word from main memory, substituting the new single word for the old one, and rewriting the new double-word into main memory. Because of the cache's write-through feature, write operations that result in cache hits also involve these mutiple memory operations.

The increased memory reference time required by these multiple memory operations is transparent to the program except when an instruction that writes to memory is followed closely by another instruction which also accesses main memory, such as another write instruction or one that results in a cache miss.

Some instructions write double-words whenever possible: for instance, *Block Move* (BLM), *Store Floating-Point Double* or *Single* (FSTS, FSTD), *Vector on Interrupting Device* (VCT), and *Save* (SAVE). This diminishes both the effect of consecutive main memory accesses and instruction execution times. These instructions can be executed consecutively, without signficantly increasing program execution time, provided that the addresses specified by the instruction are even addresses.

Chapter 4

Program Flow Management

Program flow management involves the flow of instructions, interrupt handling, and fault handling. This chapter briefly discusses sequential instruction flow, the instructions that alter sequential flow, interrupt handling and fault handling. For further information on program flow management, refer to *16-Bit Real-Time Assembly Language Programming.*

Sequential Instruction Flow

Like all members of the 16-bit Real-Time ECLIPSE family, the S/280 processing system uses a 15-bit program counter (PC) to control the sequence of executing instructions. The program counter specifies the logical address of the currently-executing instruction which can be anywhere in the 64 Kbyte logical address space (locations 0 through 77777_8, except for the reserved memory locations.

To address the next instruction for normal (sequential) flow, the processor increments the program counter by

- One when executing a one-word instruction (such as ADD),
- Two when executing a two-word (extended) instruction (such as ADDI).

If the processor increments the program counter when it addresses the highest memory locations, 77777_8, address wraparound occurs. The program counter will contain 00000_8 or 000001_8, depending on the instruction's length.

Non-sequential Instruction Flow

To initiate a different program sequence, sequential flow is altered. In an S/280 system, any of the following events alter flow:

Execution of an Execute (XCT) instruction
Execution of a jump instruction
Execution of a skip instruction
Execution of a subroutine call or return instruction
Trapping on a fault
Detection of an I/O interrupt request

When any of these events occurs, except for the execution of a skip instruction, the processor clears (flushes) the instruction pipeline and forces an address into the program counter to initiate the new program sequence. If the program attempts to skip or jump into the middle of two-word instruction, the second word of the instruction will be executed as an instruction.

Program Flow Instructions

Table 4.1 lists the instructions for altering program flow. Table 4.2 gives the sequence of program flow instructions for entering and exiting from subroutines. For further information, refer to the *16-Bit Real-Time* ECLIPSE *Assembly Language Programmer's Reference.*

Instruction Mnemonic	Operation
DSPA	Dispatch
EJMP	Extended jump
EJSR	Extended jump to subroutine
JMP	Jump
JSR	Jump to subroutine
POPJ	Pop program counter and jump
PSHJ	Push jump
RTN	Return
XCT	Execute accumulator
XOP	Extended operation
XOP1	Extended operation alternate

Table 4.1 Program flow instructions

Instruction Type		
Call	**Save**	**Return**
JSR	SAVE	RTN
PSHJ	—	POPJ
XOP	—	POPB

Table 4.2 Sequence of subroutine instructions

Interrupt Handling

When the processor honors an interrupt, it disables further interrupts by setting the Interrupt On (ION) flag to 0, disables user address translation, stores the contents of the program counter in physical location 0, and stops user program execution to service the interrupt. The processor then fetches the contents of physical location 1, the address of the interrupt handler.

If this address is indirect, the processor resolves it into a final direct address that references the first instruction of the interrupt handler. Next the processor stores the contents of the program counter in physical location 0 and jumps to the first instruction of the interrupt handler.

How the processor stops program execution when it honors an interrupt depends on the instruction executing when the interrupt is honored. The executing instruction is either noninterruptible or restartable.

Table 4.3 lists the restartable instructions. All unlisted instructions are noninterruptible.

Restartable from Beginning			Restartable with Updated Values
FAD	FFAS[1]	FMS	BAM
FAMD	FFND[1]	FNOM[1]	BLM
FAMS	FHLV[1]	FPSH	CMP
FAS	FINT[1]	FPOP	CMT
FCMP	FLAS[1]	FSD	CMV
FDD	FLMD[1]	FSMD	CTR
FDMD	FMD	FSMS	LMP
FDMS	FMMD	FSS	BMC load/dump map table[2]
FDS	FMMS		

Table 4.3 Restartable Instructions

If an instruction is noninterruptible, the processor finishes executing that instruction before servicing the interrupt. Examples of noninterruptible instructions are *Add* (ADD), *Load Accumulator* (LDA), and *Complement* (COM).

If an instruction is restartable, the processor services the interrupt before the instruction finishes. When an interrupt occurs, the processor saves the address of the interrupted instruction (the contents of the program counter) before servicing the interrupt. When servicing is complete, the processor can restart the interrupted instruction in one of the following ways.

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

Fault Handling

While executing an instruction, the processing system and address translation facilities check the operation and data. If either detects an error, then one of the following faults occurs:

memory protection fault
I/O protection fault
indirection protection
stack fault
floating-point fault

Memory protection faults occur immediately after the fault condition occurs. Stack, I/O protection, and indirection protection faults occur immediately after the instruction causing the fault and before the execution of the next instruction. A floating-point fault occurs before the execution of the next floating-point instruction; other non-floating-point instructions may be executed in between. The *16-Bit Real-Time Assembly Language Programming* manual describes the handling of these faults.

The power system continually monitors the system for faults. Table 5.11 (Chapter 5) lists the error codes for these faults.

[1] *These instructions are only restartable in certain cases. For example, FHLV is restartable only when the number being halved is not normalized.*

[2] *An interrupted BMC load/dump instruction will not indicate that the BMC is busy when a SKBZ BMC or SKPBN BMC instruction is executed.*

Chapter 5

Device Management

The processing system accesses a device through the programmed I/O facility, data channel facility, or optional burst multiplexor channel facility. When a device requests service via the interrupt system, the operating system sets up device access through these facilities using programmed I/O instructions.

This chapter summarizes the I/O facilities, the general I/O instructions, the instructions for managing the interrupt system, and the instructions for managing the following basic devices which are integral parts of the S/280 I/O system:

- Programmble interval timer;
- Real-time clock;
- Asynchronous line input/output;
- Universal power supply controller.

Input/Output Facilities

Programmed I/O

The programmed I/O (PIO) facility transfers single words between a central processor accumulator and a device register. The operating system uses the PIO facility for transferring characters to and from low-speed devices, such as display terminals and printers, and for transferring commands and status to and from all devices.

The PIO facility is programmed using the general I/O instructions.

Data Channel

The data channel (DCH) facility transfers data between memory and a device buffer at the rate of up to 2.2 Mbytes per second input and 1.67 Mbytes per second output. The operating system uses the DCH facility for transferring blocks of data to and from medium-speed devices such as magnetic tape subsystems, disk, and network interfaces.

The size of the data blocks in data channel transfers is device dependent. The data block is transferred word by word while the processing system continues program execution between word transfers.

The DCH facility receives a 15-bit logical address from the device controller for each data word transfer and passes this address to the memory system's user/DCH address translator. When data channel address translation is enabled, the user/DCH address translator receives the DCH map table selector bits which select one of the four data channel map tables for address translation. The user/DCH address translator uses this map table to convert the 15-bit logical address into a 20-bit physical address. The physical address accesses the memory location for the DCH data transfer. When data channel address translation is disabled, no address conversion occurs and the physical address equals the logical address.

Programmed I/O instructions set up the user/DCH address translator and the parameters of the block transfer, but the transfer of the data itself proceeds without program intervention. Programming for the user/DCH address translator is discussed in Chapter 6, "Memory and System Management." Programming for the block transfer is device dependent and varies from controller to controller.

Burst Multiplexor Channel

The optional burst multiplexor channel (BMC) facility transfers data between memory and a device buffer at the rate of up to 13.3 Mbytes per second input and 9.7 Mbytes per second output. The operating system uses the BMC facility to transfer bursts of data between memory and high-speed disks.

Data bursts can range from one to 256 words. The processing system continues program execution during burst transfers as long as it does not need to access memory directly, that is, to write to memory or to read a memory location which is not in the cache. The BMC facility has priority over the processing system. Thus, when the processing system needs such memory access, burst transfers prevent program execution as long as devices continue to request BMC service. For this reason, burst size should be limited. Sizes of 8 or 16 words per burst are recommended.

The BMC facility receives a 20-bit logical address from the device controller and passes this destination address

to the memory system's BMC address translator. If the controller enables address translation, the BMC address translator converts the logical address into a 20-bit physical address using the appropriate map table entry. This address accesses the memory location for the BMC data transfer. After the BMC transfers each data word to or from memory, it increments the destination address. If the increment causes an overflow out of the 10 least-significant address bits, the next map table entry is used for the address conversion. Depending on the contents of the map table, this could mean that the BMC may not transfer successive words to or from consecutive pages in memory.

If the controller disables address translation, no address conversion occurs. As the BMC facility transfers each data word to or from memory, it increments the destination address, which, in this case, causes successive words to move to or from consecutive locations in memory.

The BMC facility checks the parity on each address and data word received from the device controller. When an address parity error occurs, the BMC facility informs the controller of the error, aborts the transfer, and sets the address parity error bit to 1 in the BMC status register. The BMC facility expects the controller to also abort the transfer. When a data parity error occurs, the BMC facility informs the controller of the error, sets the data parity error bit to 1 in the BMC status register, and continues the transfer.

Programmed I/O instructions set up the BMC address translator and the parameters of the burst transfer, but the transfer itself proceeds without program intervention. Programming for the BMC address translator is presented in Chapter 6, "Memory and System Management." Programming for the BMC data transfer is device dependent, varying from controller to controller.

The device handler can speed burst transfers by taking advantage of main memory's double-word organization. To do this, the device handler should specify bursts with an *even* number of words and start the burst transfer to or from an *even* physical memory address.

General I/O Instructions

A general set of I/O instructions provide device independent operations. When these instructions are issued to a specific device code, they communicate with the specified device to set up data transfers, perform special operations, read status, and initialize and test device Busy and Done flags. Table 5.1 lists the general I/O instructions.

Interrupt System

The operating system controls the interrupt system by manipulating an Interrupt On (ION) flag, interrupt mask, and device flags. The Interrupt On flag enables or disables all interrupt recognition. The interrupt mask enables or disables selective device interrupt recognition.

The device flags provide the interrupt communications link between the central processor and the device. By manipulating the flags and the interrupt mask, the operating system can ignore all interrupt requests, or selectively service certain interrupt requests.

If the Interrupt On flag and interrupt mask enable processor recognition of the interrupt request, the processor services the interrupt. To service the interrupt, the processor either halts the currently-executing instruction or finishes it, depending on the type of instruction. When execution halts, the processor immediately starts to execute the I/O interrupt handler by jumping indirectly through location 1. For more information on the processor actions to transfer program control to the interrupt handler and programming for the interrupt handler, refer to *16-Bit Real-Time* ECLIPSE *Assembly Language Programming.*

Instructions

The interrupt system responds to I/O instructions issued to the central processor. The assembler interprets these instructions using either the standard or special I/O instruction format. Device flags cannot be appended to the special format of an interrupt system instruction. Table 5.4 lists both formats of these instructions.

Device Code

77_8

Instruction Mnemonic

CPU

Priority Mask Bit

None

Device Flags

Devices flags determine whether the central processor can recognize an interrupt request and interrupt the current program to service it.

f=S	Sets the Interrupt On (ION) flag to 1, enabling the interrupt system.
f=C	Sets the Interrupt On (ION) flag to 0, disabling the interrupt system.
f=P	No effect unless used with the INTA instruction. The P flag causes the central processor to interpret the INTA instruction as the first word of the Vector (VCT) instruction.

Instruction Mnemonic	Operation
DIA[f]	Data in A (from A buffer of device)
DIB[f]	Data in B (from B buffer of device)
DIC[f]	Data in C (from C buffer of device)
DOA[f]	Data out A (to A buffer of device)
DOB[f]	Data out B (to B buffer of device)
DOC[f]	Data out C (to C buffer of device)
NIO[f]	No I/O transfer (initialize a Busy/Done flag)
SKP*t*	I/O skip (test a Busy/Done flag and skip on condition)

Table 5.1 General I/O instructions

Instruction Mnemonic for *f*	Bits 8	9	I/O Device Flag Busy	Done	CPU flag ION[2]
option omitted	0	0	No effect	No effect	No effect
S	0	1	Set to 1	Set to 0	Set to 1
C	1	0	Set to 0	Set to 0	Set to 0
P	1	1	Pulses a special I/O control line		No effect

Table 5.2 Device flags for general devices

Instruction Mnemonic for *t*	Bits 8	9	I/O Device Flag	CPU Flag[2]
BN	0	0	Test for Busy = 1	Test for ION = 1
BZ	0	1	Test for Busy = 0	Test for ION = 0
DN	1	0	Test for Done = 1	Test for Powerfail = 1
DZ	0	0	Test for Done = 0	Test for Powerfail = 0

Table 5.3 Device flags for skip instruction

Instruction Format		
Special	**Standard[3]**	**Action**
INTA	DIB[f] *ac*,CPU	Interrupt acknowledge
INTDS	NIOC CPU	Interrrupt system disable
INTEN	NIOS CPU	Interrupt system enable
MSKO	DOB[f] *ac*,CPU	Priority mask out
IORST	DIC[f] *ac*,CPU	I/O reset
—	SKP*t* CPU	CPU skip
RSTR	—	Restore
VCT	—	Vector on interrupting device

Table 5.4 Interrupt system instructions

[1]The [f] or t defines optional device flag handling as summarized in Tables 5.2 and 5.3, respectively.

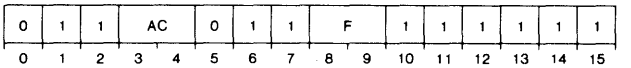
[2]ION is the Interrupt On flag.

[3]The [f] or t defines device handling as summarized in Tables 5.2 and 5.3, respectively.

INTA

Interrupt Acknowledge

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



Identifies interrupting device.

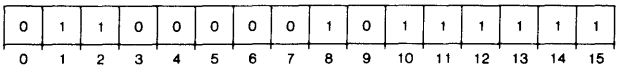
Places the 6-bit device code of the highest-priority device requesting service (the device that is physically closest to the central processor) into bits 10 through 15 of the specified accumulator. After the transfer, sets the ION flag as specified by *f*. The accumulator format after the operation is diagrammed below.

Bits	Name	Meaning or Function
0-9	—	Reserved for future use.
10-15	Device code	Device code of the highest priority device on the I/O bus that is requesting an I/O interrupt.

INTDS

Interrrupt Disable

NIOC CPU



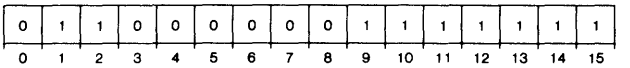
Disables the interrupt system.

Sets the ION flag to 0.

INTEN

Interrupt Enable

NIOS CPU

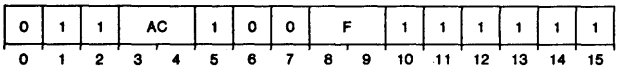


Enables the interrupt system.

Sets the ION flag to 1. If this instruction changes the state of the ION flag, the central processor executes one more instruction before recognizing an I/O interrupt. However, if the instruction is interruptible, then the central processor can recognize the first interrupt soon as the instruction begins executing.

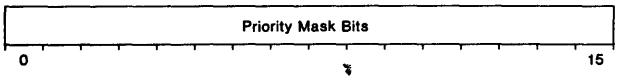
MSKO
Mask Out

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



Specifies the priority mask.

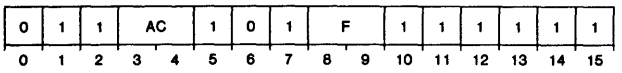
Places the contents of the specified accumulator in the priority mask (location 5). After the transfer, sets the ION flag as specified by *f*. The accumulator before and after the operation is diagrammed below.



Bits	Name	Meaning or Function
0-15	Priority mask bits	1 in any bit disables interrupt requests from the devices which use that mask bit.

IORST
I/O Reset

DIC*[f]* *ac*,CPU

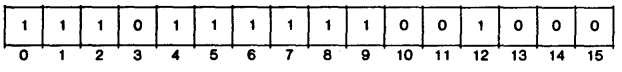


Resets the I/O system.

Sends a reset signal to all devices to set their Busy and Done flags to 0 and clear their states. Sets the priority mask to 0 and the ION flag according to the function specified by *f*. Also initializes the computer as described in Chapter 8, "Powerup and Initialization." The specified accumulator is ignored and remains unchanged.

> **NOTE:** *The* IORST *mnemnonic is equivalent to* DICC 0,CPU. *If the* DIC [f] ac,CPU *mnemonic is used, the accumulator field (*ac*) must be coded to avoid an assembly error even though the accumulator is not used.*

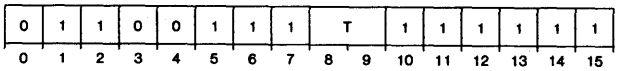
RSTR
Restore



Returns control from certain types of I/O service routines, including those called by a VCT instruction that changes the stack. Pops the five words of the stack normally associated with a standard return block plus four additional words that contain the four stack parameters. Refer to *16-Bit Real-Time* ECLIPSE *Assembly Language Programming* for further details.

SKP CPU
Central Processor Skip

SKP*t* CPU

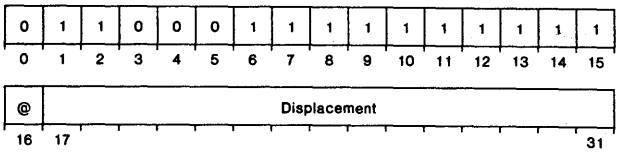


Tests the ION flag or powerfail flag.

Tests the flag specified by *t*. If the test condition is true, the central processor skips the next sequential word. (Table 5.3 lists the possible test conditions.)

VCT
Vector on Interrupting Device

VCT *[@]displacement[,index]*



Returns the device code of the interrupting device and uses that code as an index into a vector table. The absolute displacement (ignoring @) identifies the first entry in the vector table. Refer to *16-Bit Real-Time* ECLIPSE *Assembly Language Programming* for further details.

When a device requests an I/O interrupt, the central procesor fetches the first instruction specified by the I/O interrupt handler address in reserved location 1. The VCT instruction should be the first instruction fetched. It is executed before the central processor honors further interrupts.

To return from the I/O interrupt handler, use the instruction identified with the VCT mode as follows.

Mode	Return Instruction
A	JMP@0
B	JMP@0
C	POPB
D	(Restore saved stack parameters) JMP20
E	RSTR

Programmable Interval Timer

The programmable interval timer (PIT) is a CPU-independent time base which can be programmed to initiate program interrupts at fixed intervals in one of the four ranges listed in Table 5.5. The PIT can also be sampled with an I/O instruction at any point in its cycle to determine the time until the next interrupt. The operating system can use the PIT for precise time measurements or in a multiprogramming environment for allocating processor time to different programs on a time slice basis.

	Time Range		
Frequency (KHz)[4]	Minimum Interval	Maximum Interval	Increment Size
1000	1 μsec.	65.536 msec.	1 μsec.
100	10 μsec.	655.36 msec.	10 μsec.
10	100 μsec.	6.5536 sec.	100 μsec.
1	1 msec.	65.536 sec.	1 msec.

Table 5.5 Programmable interval timer rates

[4] *The frequency is selected by the system configuration jumpers as described in the "Tailoring" section of the S/280 installation data sheets (DGC No. 010-000338).*

PIT Registers

The PIT has two program-accessible registers: an interval select register and an interval counter.

The interval select register is a 16-bit register that specifies the number of increments desired between PIT interrupts.

The interval counter is a 16-bit counter which counts the number of increments since the last PIT interrupt. When the count equals the number specified by the interval select register, the PIT initiates a program interrupt and continues counting until stopped by the program.

PIT Instructions

The PIT responds to the I/O instructions listed in Table 5.6.

Instruction Mnemonic	Operation
DIA PIT	Reads the current value of the interval counter.
DOA PIT	Specifies the time between PIT interrupts.
IORST	Sets the Busy and Done flags, interval count register, and interval counter to 0, disables interrupt requests, and stops the counting cycle. (See "Interrupt System" earlier in this chapter for a detailed description of IORST.)

Table 5.6 Programmable interval timer instructions

Device Code

43_8

Instruction Mnemonic

PIT

Priority Mask Bit

6

Device Flags

Device flags start and stop the counting cycle.

f=**S** Sets Busy flag to 1 and Done flag to 0, and starts the counting cycle.

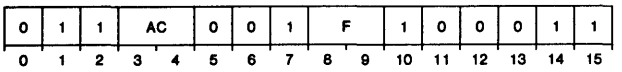
f=**C** Sets Busy and Done flags to 0, and stops the counting cycle.

f=**P** No effect.

DIA PIT

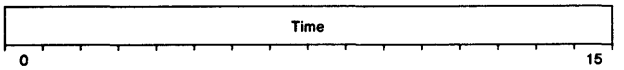
Read Interval Counter

DIA*[f]* *ac*,PIT



Returns the amount of time remaining until the next PIT interrupt.

Places the contents of the interval counter in the specified accumulator. After the transfer, performs the function specified by *f*. The accumulator format after the operation is diagrammed below.

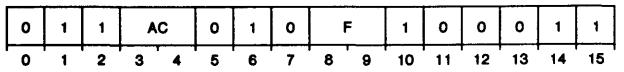


Bits	Name	Meaning or Function
0-15	Time	A signed number. If bit 0 is 1, the two's complement of the number of increments remaining until the next PIT interrupt. If bit 0 is 0, the number of increments since the PIT has been requesting interrupt service. (The jumper-selected frequency determines the size of the increments. See Table 5.5.)

DOA PIT

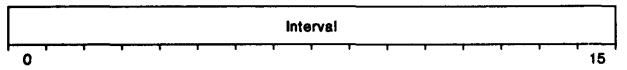
Specify Interval

DOA*[f]* *ac*,PIT



Specifies the time between PIT interrupts.

Places the contents of the specified accumulator in the interval select counter. After the transfer, performs the function specified by *f*. The accumulator format before and after the operation is diagrammed below.



Bits	Name	Meaning or Function
0-15	Interval	Two's complement of the number of increments in the desired interval between PIT interrupts. (The jumper-selected frequency determines the size of the increments. See Table 5.5.)

Programming

To obtain a particular time interval between interrupt requests, load the interval select counter with the two's complement of the number of clock increments desired between interrupts and start the counting cycle using a *Specify Interval* instruction with a *Start* command (DOAS PIT). The counter continues counting after the PIT generates an interrupt request until the program stops it.

To determine the time remaining until the next interrupt request or the time elapsed since the current interrupt was first requested, use the *Read Interval Counter* instruction (DIA PIT). To stop the counting cycle and thus clear PIT interrupt requests, use the *Clear* command with a *Read Interval Counter* instruction (DIAC PIT) or with an *No I/O Operation* instruction (NIOC PIT).

Real-Time Clock

The real-time clock (RTC) generates low-frequency I/O interrupts for performing CPU-independent time calculations. These interrupts, which occur at regular, predetermined intervals, may be used by the operating system to keep the time of day.

Four program-selectable frequencies are available: 10 Hz, 100 Hz, 1000 Hz, and line frequency.

RTC Instructions

The RTC responds to the I/O instructions listed in Table 5.7.

Instruction Mnemonic	Operation
DOA RTC	Selects frequency of RTC interrupts.
IORST	Selects line frequency and sets the Busy and Done flags to 0. (See "Interrupt System" earlier in this chapter for a detailed description of IORST.

Table 5.7 Real-time clock instructions

Device Code

14_8

Instruction Mnemonic

RTC

Priority Mask Bit

13

Device Flags

Device flags enable and disable the real-time clock (RTC).

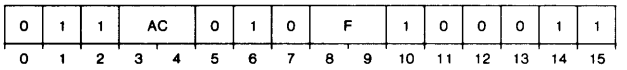
f=S Sets Busy flag to 1 and Done flag to 0 and enables the RTC.

f=C Sets Busy and Done flags to 0 and enables the RTC.

f=P No effect.

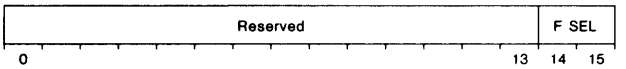
DOA RTC
Select Frequency

DOA *[f] ac*,RTC



Selects the frequency of RTC interrupts.

Places the contents of the specified accumulator in the frequency select register. After the transfer, performs the function specified by *f*. The accumulator format before and after the operation is diagrammed below.



Bits	Name	Contents or Function
0-13	Reserved	Reserved for future use.
14-15	F SEL	Selects the clock frequency.

Bits 14	Bits 15	Frequency
0	0	Line frequency (50 or 60 Hz)
0	1	10 Hz
1	0	100 Hz
1	1	1000 Hz

Programming

Programming the real-time clock consists of selecting the clock frequency, enabling clock interrupts, and servicing clock interrupts.

To select the clock frequency, use the *Select Frequency* instruction (DOA RTC). To enable clock interrupts, use the *Start* command either with the *Select Frequency* instruction (DOAS RTC) or with the *No I/O Transfer* instruction (NIOS RTC). Since the clock is free-running, the interrupt request may occur at any time up to one clock period after the Busy flag is set to 1 by the *Start* comand. When the clock period expires, the real-time clock sets the Busy flag to 0 and the Done flag to 0, thus initiating an interrupt request if the clock's interrupt disable flag is not set to 1.

When the interrupt handler services interrupt requests, it should issue a *No I/O Transfer* instruction with either a *Start* command (NIOS RTC) or a *Clear* command (NIOC RTC). The *Start* command enables an interrupt request at the expiration of the current clock period. The *Clear* command inhibits subsequent clock interrupts.

Powerup Response and Timing

After power-up, the line frequency is selected as the clock frequency and both the Busy and Done flags are set to 0.

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

Asynchronous Input/Output Line

The asynchronous input/output line of the system console provides the communications link between the central processor and the master terminal. It supports asynchronous communication at jumper-selected rates ranging from 50 to 38,400 baud in 7-bit codes with program-generated parity or 8-bit codes with no parity and one or two stop bits in either format.

Because asynchronous input (TTI) and output (TTO) can independently generate program interrupts, each has its own device code and is controlled by its own set of Busy and Done flags. When the Break key on the master terminal is pressed, the input line generates a non-maskable interrupt which gives the virtual console control, if the front console is not locked. The Break key has no effect if the program is in an infinite indirection loop.

Registers

The asynchronous input/output line has two program-accessible registers: an input buffer and an output buffer.

The 8-bit input buffer stores the assembled character that is received over the communications line in serial format. When the input buffer receives a character, the interface sets the input Busy flag to 1. The buffer holds the character until the next assembled character overwrites it.

The 8-bit output buffer stores the characters sent to the interface by the program. When the master terminal asserts a Clear To Send signal, the interface disassembles the character and sends it over the communications line in serial form.

Instructions

The asynchronous input/output line responds the the I/O instructions listed in Table 5.8.

Instruction Mnemonic	Operation
DIA TTI	Reads a character from the master terminal into an accumulator.
DOA TTO	Sends a character from an accumulator to the master terminal.
IORST	Sets the Busy and Done flags to 0; disables interrupts. (See "Interrupt System" earlier in this chapter for a detailed description of IORST.)

Table 5.8 Asynchronous Input/output line instructions

Device Codes

Input 10_8

Output 11_8

Instruction Mnemonics

Input TTI

Output TTO

Priority Mask Bit

Input 14

Output 15

Device Flags

Device flag commands determine the flag settings and transmission of an output character.

f=S Sets the Busy flag to 1 and the Done flag to 0.

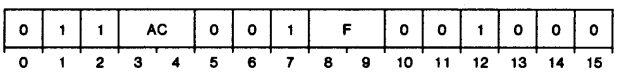
f=C Sets the Busy and Done flags to 0.

f=P No effect.

DIA TTI

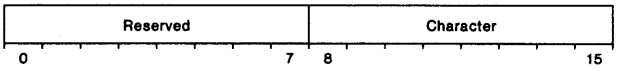
Read Character

DIA*[f]* *ac*,TTI



Reads a character from the input buffer.

Places the contents of the interface's input buffer into bits 8 through 15 of the specified accumulator. After the transfer, performs the function specified by *f*. The accumulator format after the operation is diagrammed below.

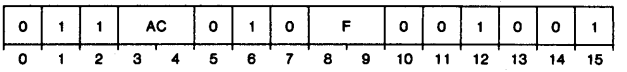


Bits	Name	Contents or Function
0-7	Reserved	Reserved for future use.
8-15	Character	Character read from the input buffer, right justified.

DOA TTO

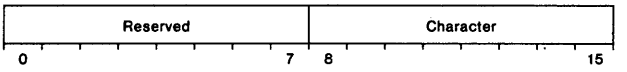
Write Character

DOA*[f]* *ac*,TTO



Writes a character into the output buffer

Places bits 8 through 15 of the specified accumulator in the interface's output buffer. After the transfer, performs the function specified by *f*. The accumulator format before and after the operation is diagrammed below.



Bits	Name	Contents or Function
0-7	Reserved	Reserved for future use.
8-15	Character	Character to be written into the output buffer, right-justified.

Programming

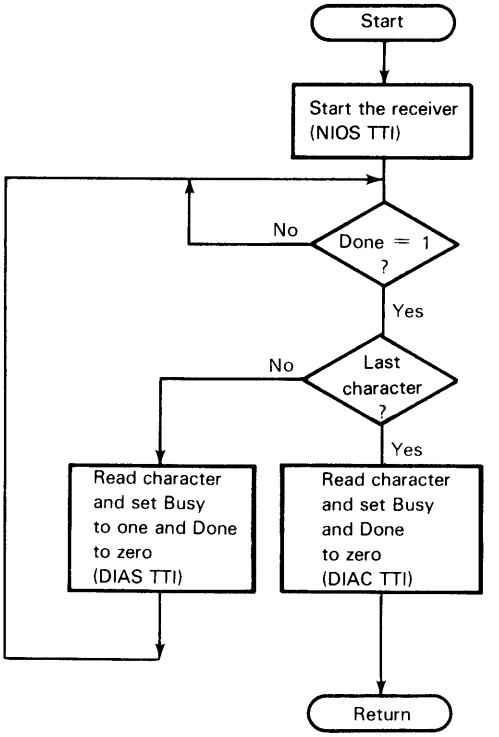
The asynchronous input/output line is set up to transmit and receive 8-bit characters without parity checking. A process may send or receive 7-bit characters with even, odd, or mark parity by using the most significant bit in the 8-bit character (bit 8 in the accumulator) as a parity bit. On transmission,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

If the master terminal operates with a character length of seven bits and does not generate parity, this device should be configured to operate with mark parity. When the program receives characters from a 7-bit device, it should mask out the parity bit after the character has been loaded into an accumulator. The parity bit, contained in bit 8 of the specified accumulator, is the most significant bit of the character.

> **NOTE:** *The Data Terminal Ready signal is always asserted.*

Reading Characters

Programming the line to read characters involves the steps shown in Figure 5.1. Initiate character reception using a *No I/O Transfer* instruction with a *Start* command (NIOS TTI). This command sets the input Busy flag to 1 and Done flag to 0. When the input buffer receives a character, it then sets the Done flag to 1 and initiates an interrupt request if interrupts are enabled. When the Done flag is 1, read the character into an accumulator using a *Read Character* instruction with either an *S* command (DIAS TTI) or a *C* command (DIAC TTI). The *S* command restarts character reception. The *C* command terminates character reception by setting both the Busy and Done flags to 0.

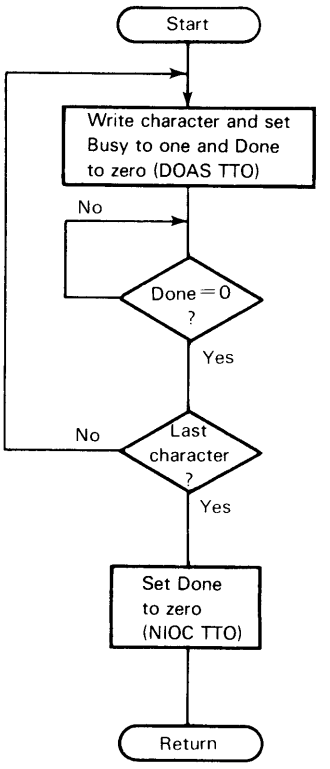


DG-08311

Figure 5.1 Reading characters

Writing Characters

Programming the interface to write characters involves the steps shown in Figure 5.2. Before sending a character to the output buffer, check the output Busy flag using an *I/O Skip* instruction (SKPBZ TTO). When Busy is 0, load a character from an accumulator into the output buffer using a *Write Character* instruction with an *S* command (DOAS TTO). The *S* command sets the output Busy flag to 1 and its Done flag to 0. After the line transmits the character over the communications line, it sets the Busy flag to 0 and the Done flag to 1, thus initiating an interrupt request if interrupts are enabled. After the last character is written, set the Done flag to 0 with an *No I/O Transfer* instruction (NIOC TTO).



DG-09006

Figure 5.2 Writing characters

Powerup Response and Timing

After powerup, the input and output Busy and Done flags are 0. After the input Done flag is set to 1, the character in the input buffer is available to the program for a time interval determined by the transmission rate (baud). To avoid possible data loss, the interrupt handler must respond to the interrupt request by reading the character within the time interval indicated in Table 5.9.

After the output Done flag is set to 1, the interrupt handler should supply another character within the time period indicated in Table 5.9 to maintain the maximum transmission rate.

Baud	Maximum Allowable Programmed I/O Latency (milliseconds)[5]
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

Table 5.9 Timing considerations, asynchronous input/output line

Universal Power Supply Controller

The universal power supply controller (UPSC) performs a powerup self-test; monitors system power; and, under program control, reports failures, problems, and status to the ECLIPSE S/280 processor.

The UPSC monitors the system for the following conditions:

- Power supply problems such as excessive temperature and overcurrent, undervoltages, and overvoltages;
- Ac overvoltage or undervoltage;
- Reed switches for sensing overload on +5V;
- Power switch On/Off status;
- Battery backup failure for backup systems with a failure signal;
- Excessive cabinet temperature input for cabinets with temperature sensors;
- Fan failure.

When failures occur, the UPSC causes the lights on S/280 chassis front panel to display a code identifying the type of failure. If UPSC interrupts are enabled, it also generates a program interrupt. The interrupt handler can retrieve the fault code that identifies the specific fault.

USPC Instructions

The UPSC responds to the I/O instructions listed in Table 5.10.

Instruction Mnemonic	Operation
DOAS UPSC	Enables UPSC interrupts on power system faults and/or masks out powerfail interrupts.
DOAP UPSC	Requests power system status.
DIA UPSC	Reads power system status.
IORST	Clears Busy and Done flags and disables interrupts. (See "Interrupt System" earlier in this chapter for a detailed description of IORST.)

Table 5.10 Universal power supply controller instructions

Device Code

4_8

Instruction Mnemonic

UPSC

Priority Mask Bit

13

Device Flags

Device flag commands to the UPSC set up data transfers and enable and disable UPSC interrupts.

f=S — Sets the Busy flag to 1 and the Done flag to 0. Functions as part of the *Enable* UPSC *Fault Interrupts* instruction (DOAS UPSC) which write data to the UPSC control register.

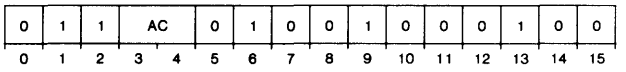
f=C — Sets the Busy and Done flags to 0.

f=P — Sets the Busy flag to 1 and the Done flag to 0. Functions as part of the *Enable* UPSC *Fault Interrupts* instruction (DOAP UPSC) which selects the information to be read during the next *Read Power System Status* instruction (DIA UPSC).

[5]Times assume that characters transmitted at 50 to 110 baud include 8 data bits and 2 stop bits, and that characters transmitted at 134.5 to 38,400 baud contain 8 data bits and 1 stop bit.

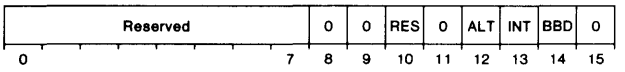
DOAS UPSC
Enable USPC Fault Interrupts

DOAS *ac*,UPSC



Enables and disables interrupts from the USCP.

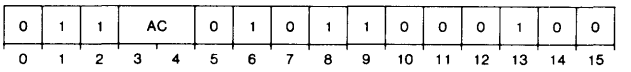
Sets the Busy flag to 1 and the Done flag to 0, then places the contents of the specified accumulator in the control register. When the transfer is finished, sets the Busy flag to 0 and the Done flag to 1. The accumulator format before and after the operation is diagrammed below.



Bits	Name	Contents or Meaning When 1
0-7,10	Reserved	Reserved for future use.
8,9	—	Both bits must be 0 to select the control register.
11	—	Used for diagnostic testing. Must be 0 for normal operation.
12	ALT	If 1, enables alternate powerfail mode to disable powerfail interrupts from device code 0. As a result, powerfail skip instructions (SKPDN and SKPDZ) always function as if there were no powerfail.
13	INT	If 1, enables UPSC to initiate a program interrupt when it detects a fault.
14	BBD	If 1, disables battery backup unit.
15	—	Used for diagnostic testing. Must be 0 for normal operation.

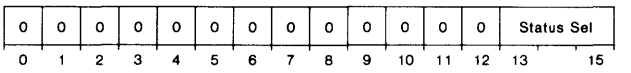
DOAP UPSC
Request Power System Status

DOAP *ac*,UPSC



Sets up the USPC to supply status information when the next *Read* USPC *Status* instruction (DIA UPSC) instruction is issued.

Sets the Busy flag to one and the Done flag to 0, then initializes the UPSC to transfer the information requested by bits 13 through 15 of the specified accumulator. After the initialization, the UPSC sets the Busy flag to 0 and the Done flag to 1. The accumulator format before and after the operation is diagrammed below.

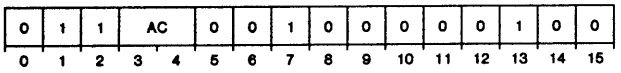


Bits	Name	Contents or Meaning When 1
0-12	—	Must be 0.
13-15	Status Sel	Selects the status information to be supplied by next *Read Power System Status* instruction (DIA UPSC). **Bits 13 14 15 — Information Selected** 0 0 0 — Control status 0 1 0 — Battery backup status 0 1 1 — Most recent fault code 1 0 0 — UPSC microcode revision Refer to the description of the *Read Power System Status* instruction (DIA UPSC) for details on each type of information.

DIA UPSC

Read Power System Status

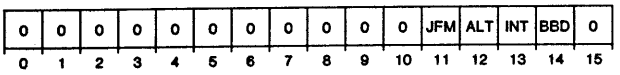
DIA *ac*,UPSC



Returns the status information requested by the previous *Request Power System Status* instruction (DOAP UPSC).

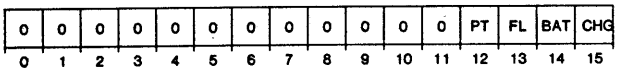
Loads the data requested by the previous *Request Power System Status* instruction (DOAP UPSC) into the specified accumulator. The accumulator format after the operation is diagrammed below.

Control Status



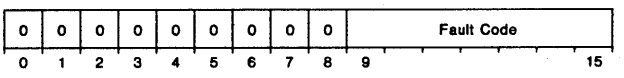
Bits	Name	Contents or Meaning When 1
0-10	—	Reserved for future use.
11	JFM	If 1, the UPSC is jumpered for voltage margining which may degrade the operation of the system.
12	ALT	If 1, alternate powerfail mode is enabled, disabling powerfail interrupts from device code 0. As a result, powerfail skip instructions (SKPDN and SKPDZ) always function as if there were no powerfail.
13	INT	If 1, fault interrupts enabled.
14	BBD	If 1, battery backup disabled.
15	—	Used for diagnostic testing. 0 for normal operation.

Battery Backup Status



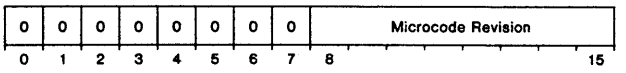
Bits	Name	Contents or Meaning When 1
0-11	—	Reserved for future use.
12	PT	If 1, partial battery backup is present.
13	FL	If 1, full battery backup is present.
14	BAT	If 1, the system is running on battery power.
15	CHG	If 1, the batteries are recharging.

Fault Code



Bits	Name	Contents or Meaning When 1
0-8	—	Reserved for future use.
9-15	Fault Code	Octal code identifying most recent fault. Bits 9-12 are the octal number of the fault in the octal fault type specified by bits 13-15. The UPSC flashes the octal fault type on the three front console lights. Table 5.11 lists the faults by type.

UPSC Microcode Revision



Bits	Name	Contents
0-7	—	Reserved for future use.
8-15	Microcode Revision	Current revision of the UPSC microcode. O indicates the first release of the microcode. Successive numbers indicate successive microcode modification.

Fault Type	Code (octal)	Meaning
0	000	System off or no fault
1		Environmental fault
	011	VNR[6] undervoltage
	021	VNR[6] overvoltage
	031	Power supply over temperature
	041	Chassis over temperature
2		Fan failure in computer chassis
	002	Blower or multiple fan failure
	012	Fan 1 failure
	022	Fan 2 failure
	032	Fan 3 failure
	042	Fan 4 failure
	052	Fan 5 failure
	062	Fan 6 failure
	072	UPSC cannot set fan signals
3		VNR[6] fault
	013	Battery backup fault
4		Power supply fault (includes undervoltages)
	004	Undervoltage on +5V
	014	Current not sharing on +5V
	044	Undervoltage on +5MEM, PS1
	054	Undervoltage on +5MEM, PS2
	064	Undervoltage on +5MEM, PS3
	074	Undervoltage on +12MEM or +12V, PS1
	104	Undervoltage on +12MEM or +12V, PS2
	114	Undervoltage on +12MEM or +12V, PS3
	124	Undervoltage on −5VMEM or −5V, PS1
	134	Undervoltage on −5VMEM or −5V, PS2
	144	Undervoltage on −5VMEM or −5V, PS3
	154	Undervoltage on unknown voltage, PS1
	161	Undervoltage on unknown voltage, PS2
	174	Undervoltage on unknown voltage, PS3
5		Overvoltage fault
	005	Overvoltage on +5V
	045	Overvoltage on +5MEM, PS1
	055	Overvoltage on +5MEM, PS2
	065	Overvoltage on +5MEM, PS3
	075	Overvoltage on +12MEM or +12V, PS1
	105	Overvoltage on +12MEM or +12V, PS2
	115	Overvoltage on +12MEM or +12V, PS3
	125	Overvoltage on −5MEM or −5V, PS1
	135	Overvoltage on −5MEM or −5V, PS2
	145	Overvoltage on −5MEM or −5V, PS3
	155	Overvoltage on unknown voltage, PS1
	165	Overvoltage on unknown voltage, PS2
	175	Overvoltage on unknown voltage, PS3
6		Over-current fault
	006	Reed switch sense low on +5V output
	016	Overcurrent on +5V, PS1
	026	Overcurrent on +5V, PS2
	036	Overcurrent on +5V, PS3
	046	Overcurrent on +5MEM, PS1
	156	Overcurrent on +5MEM, PS2
	166	Overcurrent on +5MEM, PS3
	167	Overcurrent on +12MEM or +12V PS1
	106	Overcurrent on +12MEM or +12V PS2
	116	Overcurrent on +12MEM or +12V PS3
	126	Overcurrent on −5MEM or −5V PS1
	136	Overcurrent on −5MEM or −5V PS2
	146	Overcurrent on −5MEM or −5V PS3
	156	Overcurrent on unknown voltage, PS1
	166	Overcurrent on unknown voltage, PS2
	167	Overcurrent on unknown voltage, PS3
7		UPSC fault
	007	Checksum error on UPSC ROM
	177	LED lamp test at power up

Table 5.11 Power system fault codes

[6]Voltage nonregulated unit

Table 5.11 Power system fault codes (continued)

Programming

When the front console Power switch is switched on, the UPSC starts a self-test to check its operation before supplying power to the rest of the S/280 system. When the self-test is complete, the UPSC powers up the rest of the S/280 system and starts monitoring the power system for faults.

The UPSC handles a fault as follows:

- Powers down the system, switches on battery backup, or takes no corrective action;
- Updates the fault code register;
- Flashes the fault type on the front panel lights for some faults;
- Notifies the operating system by generating a program interrupt if UPSC interrupts are enabled.

The UPSC usually powers down the system for all faults, except VNR undervoltage conditions, fan failures, and battery backup failures. If the fault is critical — an overvoltage condition, for instance — the UPSC powers down the system immediately. If the fault is less severe — an overcurrent condition, for instance — the UPSC brings the power down only when the fault persists for at least a 1 millisecond. The exceptions are overtemperature faults and fan failures which are allowed to persist for 15 seconds.

A VNR undervoltage fault occurs when the ac source voltage falls below specification. In this case, the UPSC switches on battery backup if it is present and enabled; otherwise, the UPSC powers down the system and causes a non-maskable powerfail interrupt, provided the operating system has not inhibited powerfail interrupts.

For battery backup failures, the UPSC takes no corrective action.

When UPSC interrupts are enabled and the UPSC detects a fault, it initiates a program interrupt. When the operating system issues an *Interrupt Acknowledge* instruction (INTA) in response to this interrupt, the UPSC returns device code 4. The interrupt handler can then interrogate the UPSC to determine the interrupt's cause.

A few milliseconds usually elapse between an interrupt handler's initial request for UPSC status and the UPSC's transfer of status to the CPU. To handle faults detected by the UPSC as quickly as possible, the interrupt handler can shorten this time by enabling UPSC interrupts with an *Enable* USPC *Fault Interrupts* instruction (DOAS UPSC) as soon as power comes up. When a fault occurs, the UPSC initiates a program interrupt and selects the fault code as the source of status information. In response, the interrupt handler can quickly determine the nature of the fault by simply issuing a *Read Power System Status* instruction (DIA UPSC). It does not need to initialize the UPSC to read the fault code since the UPSC does this itself in response to a fault if UPSC interrupts are enabled. The interrupt handler can access the fault code in this way until another UPSC interrupt occurs or until it issues a *Request Power System Status* instruction (DOAP UPSC) which selects status other than the fault code.

To determine the existence and/or state of the battery backup, the interrupt handler should issue a *Request Power System Status* instruction (DOAP UPSC), which selects battery backup information, and then a *Read Power System Status* instruction (DIA UPSC).

Chapter 6

Memory and System Management

The central processor and memory system supports memory management and system management facilities for an operating system. The memory management facilities provide

- Memory allocation and protection by translating logical addresses into physical addresses and controlling access to physical memory;
- Memory integrity by checking and correcting the contents of physical memory.

The system management facilities provide

- Information about system status and service faults;
- Special system functions.

This chapter summarizes these facilities and presents the instructions used by the operating system to access and control them.

Memory Allocation and Protection

The S/280 memory system has address translation facilities that allocate blocks of physical memory between processes and protect the system from unauthorized access to certain parts of memory or to I/O devices. All S/280 computers have a user and data channel address translator. Computers with the burst multiplexor facility also have a burst multiplexor address translator.

> **NOTE:** *In other Data General documentation an address translator is often referred to as a "MAP unit" or "MAP"; address translating as "mapping"; and map tables as "maps."*

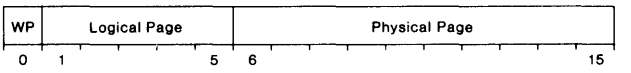
Both address translators convert the logical address of a piece of data into a physical address in memory. To perform the translation, the address translators use a map table which provides information about the pages in a process' address space. This map table contains one entry (*map*) for each 2 Kbyte logical page. The entry indicates whether or not the process can access the page and gives information for logical-to-physical address translation. For more information on logical-to-physical address translation, refer to *16-Bit Real-Time* ECLIPSE *Assembly Language Programming.*

User and Data Channel Address Translator

This address translator can store map tables for four user processes and four data channel (DCH) processes. These user and data channel processes are referred to as user A, B, C, D and data channel A, B, C, and D. The map tables can be read (loaded) and written (dumped) under program control.

The map table entries have the formats diagrammed below.

User Map Table Entry Format



Bits	Name	Contents or Function
0	WP	Write protection flag for the physical page addressed by bits 6-15. If 0, allows write access to the page. If 1, denies write access to the page.
1-5	Logical Page	Logical address of a page in a user's address space.
6-15	Physical Page	Physical address of the page with the logical address specified by bits 1-5.

DCH Map Table Entry Format

Bits	Name	Contents or Function
0	V	Access (validity) protection flag for the logical page addressed by bits 1-5. Access to page is invalid if flag is 1 and bits 6-15 are all ones; otherwise flag must be 0. See "Fault Handling" section of this chapter.
1-5	Logical Page	Logical address of a page in a data channel device's address space.
6-15	Physical Page	Physical address of the page with the logical address specified by bits 1-5.

Load Effective Address Flag

The *Load Effective Address* instruction (LEF) has the same format as an I/O instruction. The address translator has a LEF flag that determines whether an I/O format instruction is interpreted as an I/O or LEF instruction. When user address translation is enabled and the LEF flag is 1, all I/O format instructions are interpreted as *Load Effective Address* instructions (LEF). When the LEF flag is 0, all I/O format instructions are interpreted as I/O instructions.

Protection

The user/DCH address translator also performs all protection hardware checks. These checks validate validity access, write access, I/O access, and indirection. If any of these checks fails, the address translator initiates a protection fault to the operating system. For more information about the types of protection checks, refer to *16-Bit Real-Time* ECLIPSE *Assembly Language Programming.*

Registers

In addition to the eight map tables, the user/DCH address translator has three program-accessible registers.

- The 16-bit *status register* specifies the operation of the address translator.
- The 16-bit *page check register* selects the map table to be used when the program needs to check the contents of specific map table entry.
- The 16-bit *page 31 register* contains the physical page address of the memory locations that are referenced by logical page address 31 when user address translation is disabled.

Instructions

Though functionally part of the memory system, the user/DCH address translator is actually a device on the S/280's I/O bus and reponds to I/O instructions with device code 3_8 or assembler mnemonic MAP. This translator also responds to a special instruction (LMP). Table 6.1 lists these instructions.

Instruction Mnemonic	Operation
DOA MAP	Specifies the operation of the address translator.
LMP	Loads map table entries into the translator from memory.
DIA MAP	Returns translator status.
DOC MAP	Selects the map table entry for a specified logical page without changing the other status of the translator.
DIC MAP	Returns the physical address and some characteristics of the logical page specificied by the preceding DOC MAP instruction.
DOB MAP	When user address translation is disabled, selects the page 31 register as the map table entry for translating addresses in logical page 31.
NIOP MAP	Either translates one memory reference using the last selected map table or disables user address translation.
IORST	Disables user address translation and clears all bits in the translator's status register except for bits 1 and 15. (See "Interrupt System" in Chapter 5 for a detailed description of IORST.)

Table 6.1 User/DCH address translation instructions

Device Code

3_8

Instruction Mnemonic

MAP

Priority Mask Bit

None

Device Flags

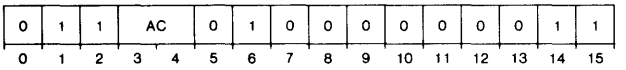
Device flags commands enable address translation for a single memory reference.

f=**S**	No effect.
f=**C**	No effect.
f=**P**	With *No I/O Transfer* instruction (NIO), either translates one memory reference using the last user page table or disables user address translation.

DOA MAP

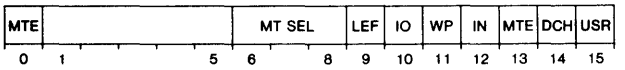
Load User/DCH Translator Status (Load Map Status)

DOA *ac*,MAP



Specifies new parameters for the user/DCH address translator.

Places the contents of specified accumulator in the translator's status register and the page check register. The accumulator format before and after the operation is diagrammed below.

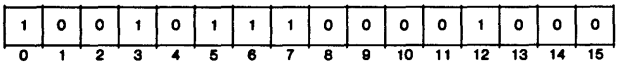


NOTE: *When this instruction sets the USR bit (bit 15) to 1, the interrupt system is disabled, and the address translator waits for an indirect reference before enabling the interrupt system and allowing the address translator to begin translating addresses with the user map table selected by bits 0 and 13 of its status register.*

Bits	Name	Contents or Function
0, 13	MTE	Enables the map table for the next user process. **Bits 0 13 — Map Table Enabled** 0 0 — User A 0 1 — User B 1 0 — User C 1 1 — User D
1-5	—	Reserved for future use.
6-8	MT SEL	Selects the page table to be loaded by the next *Load Map Table* instruction (LMP). **Bits 6 7 8 — Map Table Selected** 0 0 0 — User A 0 0 1 — User C 0 1 0 — User B 0 1 1 — User D 1 0 0 — Data channel A 1 0 1 — Data channel C 1 1 0 — Data channel B 1 1 1 — Data channel D
9	LEF	If 1 and user address translation is enabled for the next user (bit 15 = 1), then all I/O format instructions will be interpreted as Load Effective Address instructions (LEF) for the next user. If 0, all I/O format instructions, including LEF instructions, will be interpreted as I/O instructions for the next user.
10	IO	If 1, I/O protection will be enabled for the next user.
11	WP	If 1, write protection will be enabled for the next user.
12	IN	If 1, indirect protection will be enabled for the next user.
14	DCH	If 1, data channel address translation will be enabled immediately after execution of this instruction.
15	USR	If 1, user address translation will be enabled with the first memory reference after the next indirect reference.

LMP

Load User/DCH Map Table (Load Map)



Loads successive words from memory into the selected map table.

The successive words are loaded in consecutive, ascending order according to their addresses into the map table selected by bits 6-8 of the address translator's status register.

Two accumulators specify parameters for the instruction.

- AC1 must contain an unsigned integer equal to the number of words to be loaded into the map table.
- AC2 must contain the address of the first word to be loaded. If bit 0 is 1, the instruction follows the indirection chain and places the resulting effective address into AC2.

  NOTE: *Unlike some other real-time ECLIPSE computers, AC0 need not contain 0 for this instruction to execute properly.*

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

After completion of the instruction,

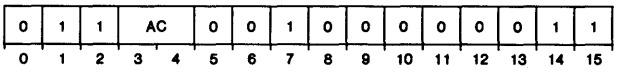
- AC0 and AC3 remain unchanged,
- AC1 contains 0,
- AC2 contains the address of the word following the last word loaded.

  NOTE: *If this instruction is issued while user address translation and I/O protection are enabled, the address translator and the accumulators are not altered and a protection fault occurs.*

DIA MAP

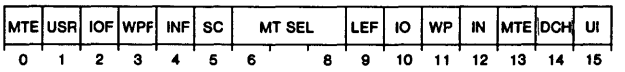
Read User/DCH Translator Status (Read MAP Status)

DIA *ac*,MAP



Supplies the current status of the user/DCH address translator.

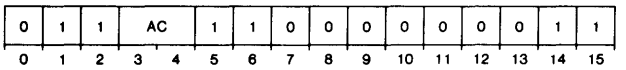
Places the contents of the user/DCH address translator status register in the specified accumulator. The accumulator format before and after the operation is diagrammed below.



Bits	Name	Contents or Function
0, 13	MTE	User map table currently enabled. **Bits 0 13 — Map Table Enabled** 0 0 — User A 0 1 — User B 1 0 — User C 1 1 — User D
1	USR	If 1, user address translation (mapped mode) is enabled.
2	IOF	If 1, the last protection fault was an I/O protection fault.
3	WPF	If 1, the last protection fault was a write protection fault.
4	IDF	If 1, the last protection fault was an indirect protection fault.
5	SC	If 1, the last protection fault occurred during a *Translate Single Cycle* instruction (NIOP MAP).
6-8	MT SEL	User map table loaded by the last *Load Map Table* instruction (LMP). **Bits 6 7 8 — Map Table Selected** 0 0 0 — User A 0 0 1 — User C 0 1 0 — User B 0 1 1 — User D 1 0 0 — Data channel A 1 0 1 — Data channel C 1 1 0 — Data channel B 1 1 1 — Data channel D
9	LEF	If 1, LEF instruction mode was enabled for the last user.
10	IO	If 1, I/O protection was enabled for the last user.
11	WP	If 1, write protection was enabled for the last user.
12	IN	If 1, indirect protection was enabled for the last user.
14	DCH	If 1, data channel address translation has been enabled.
15	UI	If 1, the last interrupt occurred while user address translation was enabled.

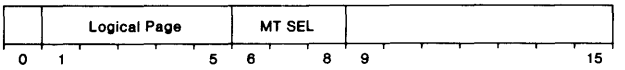
DOC MAP
Initiate Page Check

DOC *ac*,MAP



Selects a map table and specifies a table entry.

The contents of the specified accumulator are transferred to the address translator, with bits 6-8 also going to the page check register for use later by a *Page Check* instruction (DIC MAP). The accumulator format before and after the operation is diagrammed below.



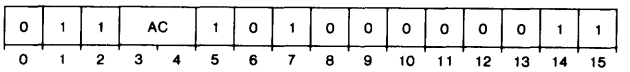
Bits	Name	Contents or Function
0	—	Reserved for future use.
1-5	Logical Page	Address of an entry in the map table selected by bits 6-8. This entry will be read by the next *Page Check* instruction (DIC MAP).
6-8	MT SEL	Selects the page table to be accessed by the next *Page Check* instruction (DIC MAP).

Bits 6 7 8	Map Table Selected
0 0 0	User A
0 0 1	User C
0 1 0	User B
0 1 1	User D
1 0 0	Data channel A
1 0 1	Data channel C
1 1 0	Data channel B
1 1 1	Data channel D

9-15	—	Reserved for future use.

DIC MAP
Page Check

DIC *ac*,MAP



Returns the physical address and the state of the Write Protect flag for the logical page specified by the preceding *Initiate Page Check* instruction (DOC MAP).

Loads bits 6-8 of the translator's page check register and bits 1 and 6-15 of the map table entry specified by the preceding *Initiate Page Check* instruction (DOC MAP) into the specified accumulator. The accumulator format after the operation is diagrammed below.

Bits	Name	Contents or Function
0	P	If 1, specifies the type of protection provided for the logical page correponding to the physical page specified by bits 6-15. If bits 1-3 select a user map table, write protection is provided for the specified page; if these bits select a data channel map table, access (validity) protection is provided for the page.
1-3	MT SEL	The map table selected to perform the address translation between the logical page and the physical page.

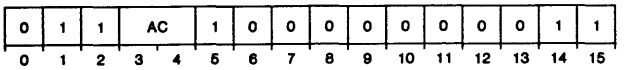
Bits 1 2 3	Map Table Selected
0 0 0	User A
0 0 1	User C
0 1 0	User B
0 1 1	User D
1 0 0	Data channel A
1 0 1	Data channel C
1 1 0	Data channel B
1 1 1	Data channel D

4-5	—	Reserved for future use.
6-15	Physical Page	The address of the physical page that corresponds to the logical page address given in the preceding *Initiate Page Check* instruction (DOC MAP).

DOB MAP

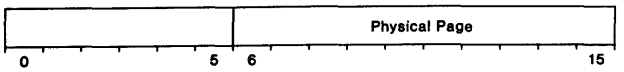
Translate Page 31
(Map Supervisor Page 31)

DOB *ac*,MAP



Supplies the physical page address for the translation of user memory references to logical page 31 when user mapping is disabled.

Loads bits 6-15 of the specified accumulator into the user page 31 register. The accumulator format after the operation is diagrammed below.

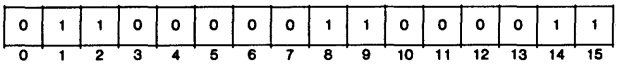


Bits	Name	Contents or Function
0-5	—	Reserved for future use.
6-15	Physical Page	The address of the physical page to which user memory references to logical page 31 are translated when user address translation is disabled.

NOTE: *After powerup, a system reset, or an* I/O Reset *instruction (IORST), the user page 31 register contains all ones (37_8), thereby translating logical page 31 addresses into physical page 31 addresses.*

NIOP MAP

Translate User Single Cycle (Map Single Cycle)
Disable User Translation (Disable User Mode)



Either translates one user memory reference or disables user address translation.

If issued while user address translation is disabled, this instruction translates one user memory reference. After the effective address, E, has been calculated for the next *Load Accumulator* instruction (LDA) or *Store Accumulator* instruction (STA), the instruction enables user address translation and translates E using the map table enabled by bits 1 and 13 of the status register. After the translation, the instruction disables user address translation. Thus, any additional memory references required by the instruction will not be translated.

NOTE: *The interrupt system is disabled from the beginning of the* Translate Single Cycle *instruction until the completion of the next LDA or STA instruction.*

If issued while user address translation is enabled and both LEF mode and I/O protection are disabled, this instruction disables user address translation. No user memory references are translated until user address translation is reenabled with a *Load User/Data Channel Address Translation* instruction (DOA MAP).

Examples

AC2 contains 405_8 and the following instructions are issued when user address translation is disabled:

```
NIOP MAP
LDA 3,2,2
```

As a result, the logical address 407_8 is mapped using the user map table specified by bits 0 and 13 of the translator's status register, and the word contained in the corresponding physical location is placed in AC3.

However, if the following instructions are issued

```
NIOP MAP
LDA 3,@2,2
```

the contents of physical location 407 are used as the first level of an indirection chain and the indirection chain is resolved. The resolved address is then translated and the word contained in the corresponding physical location is placed in AC3.

BMC Address Translator

This address translator stores one map table with 1024 entries. The map table entries have the following format.

BMC Map Table Entry

Bits	Name	Contents or Function
0	V	Access (validity) protection flag for the logical page associated with this map table entry. If 0, allows access to the page. If 1, denies access to the page.
1-5	—	Must be 0.
6-15	Physical Page	Address of a physical page in memory.

Protection

The BMC address translator also performs all protection hardware checks. These checks provide validation of access, address, and data. If any of these checks fails, the address translator initiates a protection fault to the operating system. For more information about access validation, refer to *16-Bit Real-Time* ECLIPSE *Assembly Language Programming*. For more information on address and data validation, refer to Chapter 5, "Device Management."

Registers

The BMC address translator has three program-accessible registers.

- A 16-bit *BMC status register* contains information about the state of the BMC address translator and the BMC facility.
- A 16-bit *map table transfer address* register holds the logical address of the next memory location to be accessed during a map table transfer operation (load or dump).
- A 16-bit *map table entry selector* holds the logical page address of the next map table entry to be accessed during a map table transfer operation (load or dump).

Instructions

Although the BMC address translator is functionally part of the memory system, it is actually a device on the S/280's I/O bus and reponds to I/O instructions with device code 5_8 or instruction mnemonic BMC. Table 6.2 lists these instructions.

Instruction Mnemonic	Operation
DIC BMC	Returns translator and BMC facility status.
DOA BMC	Specifies the least-significant bits of the physical address of the first memory location to be accessed during a map table transfer.
DOB BMC	Depending on the accumulator format, *either* enables a map table transfer and specifies the most-significant bits of the physical address of the first memory location to be accessed during the transfer *or* selects the first map table entry to be accessed during a map table transfer.
DOC BMC	Depending on the accumulator format, *either* specifies the number of map table entries to be transferred *or* defines the diagnostic operation of the translator.[1]
IORST	Clears all bits in the BMC status register except bit 15. (See "Interrupt System" in Chapter 5 for a detailed description of IORST.

Table 6.2 BMC address translation instructions

[1] *For information on this diagnostic instruction, refer to Appendix E.*

Device Code

5_8

Instruction Mnemonic

BMC

Priority Mask Bit

None

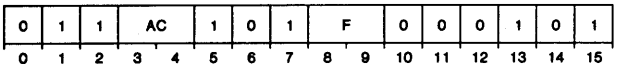
Device Flags

Device flags commands enable address translation for a single memory reference. The BMC address translator has no Busy or Done flags.

f=**S**	Initiates a BMC map table transfer operation.
f=**C**	Sets all bits of the BMC translator status register to 0 except bits 1 and 15.
f=**P**	No effect.

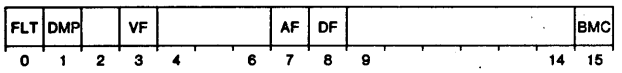
DIC BMC
Read BMC Status

DIC*[f]* *ac*, BMC



Returns the status of the BMC address translator and the BMC facility.

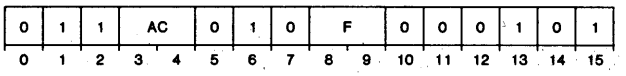
Places the contents of the user/DCH address translator status register in the specified accumulator. After the transfer, performs the function specified by *f*. The accumulator format before and after the operation is diagrammed below.



Bits	Name	Contents or Function
0	FLT	If 1, a validity protection fault, address parity fault, or data parity fault occurred.
1	DMP	If 1, the next map table transfer operation will be a load. If 0, the next map table transfer operation will be a dump.
2	—	Reserved for future use.
3	VF	If 1, a validity protection fault occurred.
4-6	—	Reserved for future use.
7	AF	If 1, an address parity faulty occurred.
8	DF	If 1, a data parity fault occurred.
9-14	—	Reserved for future use.
15	BMC	If 1, the BMC facility is present in the system.

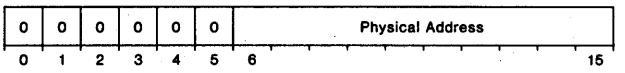
DOA BMC
Specify Initial Address
(Specify Low-Order Address)

DOA*[f]* *ac*, BMC



Specifies the 10 least significant address bits of the first memory location to supply or receive a word during a map transfer operation.

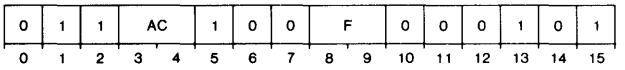
Places bits 6-15 of the specified accumulator into the translator's map transfer address register. After the transfer, performs the operation specified by *f*. The accumulator format before and after the operation is diagrammed below.



Bits	Name	Contents or Function
0-5	—	Must be 0.
6-15	Physical Address	Ten least significant physical address bits of the first memory location to supply or receive a map table entry during the next map table transfer operation.

DOB BMC
Specify BMC Map Table Transfer (Specify Operation and High-Order Address)

DOB*[f]* *ac*, BMC



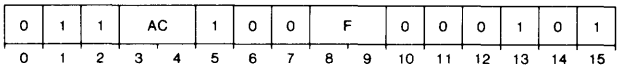
Specifies the map table transfer operation (load or dump) and the 10 most significant address bits of the first memory location to supply or receive a word during the operation.

Places bits 1 and 6-15 of the specified accumulator in the translator's status register and map transfer address register, respectively. After the transfer, performs the operation specified by *f*. The accumulator format before and after the operation is diagrammed below.

Bits	Name	Contents or Function
0	—	Must be 0. (Distinguishes this instruction from a *Select Initial* BMC *Map Table Entry* instruction which has the same assembler mnemonic.)
1	DMP	If 1, the next map table transfer operation is a dump. If 0, the next map table transfer operation is a load.
2-5	—	Must be 0.
6-15	Physical Address	Ten most significant physical address bits of the first memory location to supply or receive a map table entry during the next map table transfer operation.

DOB BMC
Select Initial BMC Map Entry (Specifiy Initial Map Register)

DOB*[f]* *ac*,BMC



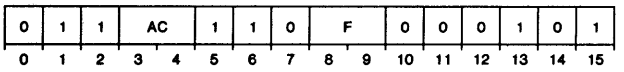
Selects the first map table entry to receive or supply a word during the next map table transfer (load or dump) operation.

Places bits 1 and 6-15 of the specified accumulator into the translator's status register and the map table transfer address register, respectively. After the transfer, performs the operation specified by *f*. The accumulator format before and after the operation is diagrammed below.

Bits	Name	Contents or Function
0	—	Must be 1. (Distinguishes this instruction from the *Specify* BMC *Map Table Transfer* instruction which has the same assembler mnemonic.)
1	—	Reserved for future use.
2-5	—	Must be 0.
6-15	Initial MTE	Logical page address of the first map table entry to receive or supply a word during the next map table transfer operation (load or dump).

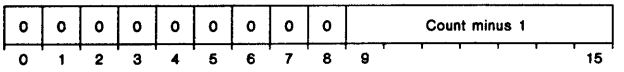
DOC BMC
Specify BMC Map Entry Count

DOC*[f]* *ac*,BMC



Specifies the number of map table entries to loaded or dumped during the next map table transfer operation.

Places bits 9-15 of the specified accumulator into the map table entry selector. After the transfer, performs the operation specified by *f*. The accumulator format before and after the operation is diagrammed below.



Bits	Name	Contents or Function
0	—	Must be 0. (Distinguishes this instruction from the diagnostic instruction, *Load* BMC *Status Register*, which has the same assembler mnemonic.)[2]
1-8	—	Must be 0.
9-15	Count minus 1	The number that is one less than the number of map table entries to be loaded or dumped during the next map table transfer operation. A total of 128 entries (four complete map tables) can be transferred in one operation.

[2]*For information on this diagnostic instruction, refer to Appendix E.*

Programming Address Translators

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

The supervisor must also maintain a map table in memory or on disk for itself and each process that requires address translation. Normally, the supervisor reserves the user A map table for itself. When setting up the map table for a process, the supervisor should invalidate all pages not needed by the process. This ensures that mistaken references to unneeded pages do not result in unwanted access to memory used by other processes. In other words, if a process only needs 12_{10} pages (24 Kbytes), then logical pages 13_{10} through 30_{10} should be invalidated. Invalidate a page by setting the write protect bit (user processes) or the validity protect bit (data channel or BMC processes) and the physical page bits to 1 in the map table entry for the page.

NOTE: *Physical page* 1777_8 *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 address translator. Load a map table into the translator from memory with the following instruction sequences.

Loading user/DCH map table

- *Load User*/DCH *Translator Status* instruction (DOA MAP) to select the map table to be loaded.
- *Load User*/DCH *Map Table* instruction (LMP) to start storing the map table.

NOTE: *The S/280 user/DCH address translator allows loading of data channel map tables from a DCH device, such as Data General's Intelligent Asynchronous Controller (IAC), which supports this feature. Instructions from the particular DCH device's instruction set are used to load data channel map tables in this way.*

Loading BMC map table

- *Select Initial* BMC *Map Entry* instruction (DOB BMC) and *Specify* BMC *Map Entry Count* instruction (DOC BMC) to specify the map table entries to be loaded.
- *Specify* BMC *Map Table Transfer* instruction with a *Start* command (DOBS BMC) to start storing the map table entries.

Both the *Load User/*DCH *Map Table* instruction (LMP) and any BMC instruction with a *Start* command (such as the DOBS BMC instruction above) are interruptible and resumable. Since neither address translator has a Busy flag, the supervisor can not tell when the instruction is finished. For this reason, interrupts should be disabled with an *Interrupt Disable* instruction (INTDS) before issuing this instruction and then reenabled with an *Interrupt Enable* instruction (INTEN).

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 User/*DCH *Translator Status* instruction (DOA MAP).

When a *Load User/*DCH *Translator Status* instruction (DOA 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 User/*DCH *Translator Status* instruction (DOA 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 those used to set up the parameters of a data channel transfer for the device.

Unlike user and data channel address translation, BMC address translation is always enabled unless disabled by a BMC device controller. Once BMC map table entries are loaded, a BMC process can use them without supervisor intervention.

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 or BMC map table is loaded that alters the translation of the logical page indicated by the DCH address register or the BMC address register for the next DCH or BMC 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.*

Fault Handling

Before logical-to-physical address translation occurs, the processor makes sure that the process is allowed to access the physical address, i.e., that the physical address is *valid* for the process. Whenever a process attempts to access an invalid physical address, a validity protection fault occurs.

When a user process causes a validity protection fault, the following events occur automatically.

- The current user map table is disabled.
- A five-word return block is pushed onto the system stack.
- Control transfers to the protection fault handler by an indirect jump through reserved memory location 3 which should contain the address of the protection fault handler.

When a data channel process causes a validity protection fault, none of the above events occur, and the operating system cannot determine that the fault occurred.

In addition to validity protection, the user/DCH address translator can also provide a *user* process with write protection, indirect protection, and I/O protection. When a process violates one of these types of protection, a fault occurs. The fault's effect is the same as a validity protection fault, except the appropriate fault bit is set to 1 in the translator's status register.

The protection fault handler can determine the type of fault that occurred using a *Read User/*DCH *Translator Status* instruction (DIA MAP).

When a BMC process causes a validity protection fault, none of the above events occur. Instead, only the validity protection fault bit is set to 1 in the BMC facility status register and the status register of the BMC device that caused the fault. The device handler servicing the BMC device should check for validity errors when it checks for data and address errors after a BMC transfer operation is complete. Check for errors by reading the BMC facility's status register or the status register for the BMC device causing the fault.

> **NOTE:** *After a validity or write protection fault, the program counter in the five-word block pushed onto the stack does* not *have any connection to the address of the instruction that caused the fault. Its contents are undefined.*

ERCC Facility

The integrity of the S/280 memory system is enhanced by the protection facilities of the address translators and by the error checking and correction (ERCC) facility. The protection facilities are discussed in the previous section "Memory Allocation and Protection." The ERCC facility is discussed here.

During all write operations, the ERCC facility generates and appends a 7-bit check code to each two 16-bit data words (double word) and sends all 39-bits to the addressed memory location.

When the ERCC facility is enabled by the program, it corrects all single-bit memory errors on read data from memory locations.

Whenever a memory location is read, the ERCC facility processes the double word with its check code to determine if an error has occurred. If a single-bit error has occurred in either word, the facility corrects the erroneous bit before sending the addressed word or double word to the requestor. The facility also detects, but does not correct, double bit and most triple bit errors. However, when any error is detected, it stores the fault address and an error syndrome code. If the program enables interrupts, the facility sets its Done flag to 1 and issues a program interrupt request.

The program can transfer the fault address and syndrome code to accumulators for an error handling routine. The fault address is the physical address of the first word in the faulty double word, except when the error is detected during a cache-fill operation (quad-word read). In this case, it is the address of the first word in the faulty quad-word block.

Modes

The ERCC facility operates in three program-selectable modes: check, sniff, idle. Normally, the facility operates in both check and sniff modes.

Check mode is automatically enabled at power-up but can be disabled by the program. In this mode, the facility checks and corrects all data read from memory. Since locations are organized as double words, all single-word write requests are implemented as read-modify-write operations in which the memory control unit first reads the double word containing the addressed word, then modifies the word in the double word, and finally rewrites the modified double word. Because of the processor's 16-bit organization, most write requests from the processor are single-word writes, and, even when they result in a cache hit, they still require a read-modify-write to memory because of the cache's write-back feature. Thus, except for reads from the cache, most memory operations can result in error checking.

Sniff mode is automatically enabled at powerup but can be disabled by the program to reduce the refresh time. Although sniff mode can be disabled without disabling check mode, it cannot be enabled unless check mode is also enabled. In sniff mode, the facility checks memory during refresh by reading a memory location, correcting any single-bit error, and rewriting the location. Operation in this mode is called *sniffing*. Sniffing usually takes 225 nanoseconds; it takes an additional 225 nanoseconds each time an error is corrected. Normally the program does not notice this time unless the cache hit rate is as low as 25%.

In *idle mode*, the facility does not check data; however, it does continue to generate check codes for each double word written to memory. The facility is in idle mode whenever both check and shift modes are disabled. Since checking and correcting require a small amount of additional time, programs run fastest in idle mode; however, the time is so insignificant that running in this mode is not worth the loss in data reliability that results.

ERCC Instructions

Although functionally part of the memory system, the ERCC facility is actually a device on the S/280's I/O bus and responds to I/O type instructions with device code 28 and assembler mnemonic ERCC.

The facility has a Done flag which is enabled only when ERCC interrupts are enabled. If the Done flag is enabled, the facility sets it to 1 after detecting an error and initiates an interrupt request. Interrupts can be enabled for errors detected during normal operation and/or during sniffing. The facility has no Busy flag and no mask bit in the priority mask. Its device code, 2, can be coded as the mnemonic ERCC when the ECLIPSE assembler is being used.

The facility responds to the I/O instructions listed in Table 6.3 and three diagnostic instructions. The diagnostic instructions listed in Appendix E are solely for use by Data General's diagnostic programs.

Instruction Mnemonic	Operation
DOA ERCC	Enable error checking and correction
DIA ERCC	Read memory fault address
DIB ERCC	Read memory fault code and address
IORST	Enables check and sniff modes and disables ERCC interrupts (See "Interrupt System" in Chapter 5 for a detailed description of IORST.)

Table 6.3 Error checking and correction instructions

Device Code

2_8

Instruction Mnemonic

ERCC

Priority Mask Bit

None

Device Flags

The ERCC facility responds to the following flag commands.

f=**S** Sets Done flag and Interrupt Request flag to 0.

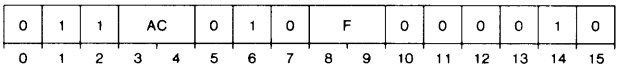
f=**C** No effect.

f=**P** No effect.

DOA ERCC

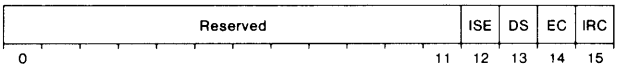
Enable ERCC

DOA *[f] ac*, ERCC



Selects functions of the error checking and correction (ERCC) facility.

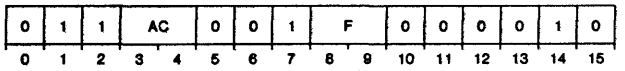
Sets the ERCC facility to function as selected by bits 13-15 of the specified accumulator and performs the function specified by *f*. The accumulator format before and after the operation is diagrammed below.



Bits	Name	Meaning When 1
0-11	Reserved	Reserved for future use
12	ISE	Enables interrupts when data errors are detected during sniffing if interrupts during checking are enabled. (Bit 15 is set to 1.)
13	DS	Disables sniffing.
14	EC	Enables checking.
15	IRC	Enables setting of the Done flag. Also enables interrupts when data errors are detected during checking. Note that errors detected during sniffing will not cause interrupts unless sniffing interrupts are also enabled. (Bit 12 is set to 1.)

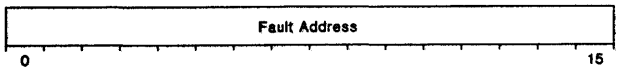
DIA ERCC
Read Memory Fault Address

DIA*[f]* *ac*,ERCC



Reads the sixteen least significant bits of the memory fault address.

Loads the sixteen least significant physical address bits of the memory location with the faulty data into the specified accumulator. After the transfer, performs the function specified by *f*. The accumulator format before and after the transfer is diagrammed below.

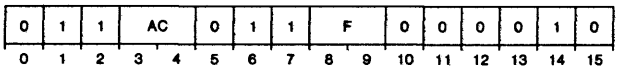


Bits	Name	Contents or Function
0-15	Fault address least significant bits	Sixteen least significant physical address bits of the double word or quad word with the faulty data.

NOTE: *The fault address and fault code (syndrome) are only valid while the Done flag is set to 1. The fault address is the address of the first word in faulty double word, except during cache fill operations, when it is the address of the first word in the faulty quad-word block.*

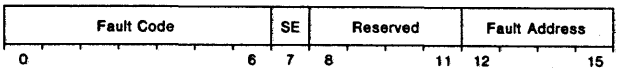
DIB ERCC
Read Memory Fault Code and Address

DIB*[f]* *ac*,ERCC



Reads the fault code and the four most significant bits of the memory fault address.

Loads the fault code (syndrome) into bits 0-6 and the four most significant physical address bits into bits 12-15 of the specified accumulator. Also sets accumulator bit 7 if the error is detected during sniffing. After the transfer, performs the function specified by *f*. The accumulator format before the transfer is diagrammed below.



Bits	Name	Contents or Function
0-6	Fault code	Fault code (syndrome) indicating no bit, 1 bit, 2 bit, or more than 2-bit errors as listed in Table 6.4.
7	SE	Error was detected during sniffing.
8-11	Reserved	Reserved for future use.
12-15	Fault address most significant bits	Four most significant physical address bits of the double word or quad word with the faulty data.

NOTE: *The fault address and fault code (syndrome) are only valid while the Done flag is set to 1. The fault address is the address of the first word in the faulty double word, except during cache fill operations, when it is the address of the first word in the faulty quad-word block.*

				6	0	1	0	1	0	1	0	1
Bit				5	0	0	1	1	0	0	1	1
0	1	2	3	4	0	0	0	0	1	1	1	1
0	0	0	0		X	38	37	T	36	T	T	30
0	0	0	1		35	T	T	27	T	5	M	T
0	0	1	0		34	T	T	25	T	3	15	T
0	0	1	1		T	M	13	T	23	T	T	M
0	1	0	0		33	T	T	24	T	2	M	T
0	1	0	1		T	1	12	T	22	T	T	M
0	1	1	0		T	M	10	T	20	T	T	M
0	1	1	1		16	T	T	M	T	M	M	T
1	0	0	0		32	T	T	M	T	M	14	T
1	0	0	1		T	M	11	T	21	T	T	M
1	0	1	0		T	M	9	T	19	T	T	31
1	0	1	1		M	T	T	29	T	7	M	T
1	1	0	0		T	M	8	T	18	T	T	M
1	1	0	1		17	T	T	28	T	6	M	T
1	1	1	0		M	T	T	26	T	4	M	T
1	1	1	1		T	0	M	T	M	T	T	M

Table 6.4 Memory fault code interpretation

ERROR KEY:

Number	Bit in error
T	Two bits in error
M	More than two bits in error
X	No errors

NOTE: *Bits 32-38 are the check bits.*

Programming

Programming the ERCC facility involves setting it up to operate as required by the application and handling memory faults.

Since the facility is automatically in check and sniff modes upon powerup, error checking and correction occurs during all memory reads and refresh operations, and maximum memory integrity is achieved. The facility continues to operate in this way unless the program disables check or sniff modes.

Error checking can be completely disabled or disabled only during refresh operations by issuing an *Enable* ERCC instruction (DOA ERCC) to disable check mode or sniff mode, respectively. To reenable error checking, issue the same instruction but enable either check mode or both check and sniff modes.

Before errors can be handled, the program must determine when errors occur either by testing the state of the Done flag, checking for ERCC interrupts, or testing the state of the Done flag. The usual method is to have the interrupt handler program check for ERCC interrupts.

Since ERCC interrupts are disabled on powerup, they can only occur after the interrupt handler issues an *Enable* ERCC instruction (DOA ERCC) which enables interrupts. With this instruction, the handler can enable interrupts only on errors detected during check mode operation or on errors detected during both check and sniff mode operation. It cannot enable interrupts only on errors detected during sniffing.

Since the ERCC facility does not have a mask bit, the handler cannot mask out ERCC interrupts.

When the interrupt handler detects an ERCC interrupt, it can determine the memory fault address by issuing a *Read Fault Address* instruction (DIA ERCC) and *Read Fault Code and Address* instruction (DIB ERCC). The information DIB returns gives the fault code (syndrome) and indicates if the error was detected during sniffing. Only when the error is detected during sniffing can the handler be sure that the fault address is the physical address of the first word in the faulty double word. When the error is detected during check mode operation, the fault address can be the first word in the faulty double word or faultly quad-word block. The handler's response to this information depends on the system application and/or the program running when the error was detected.

Timing and Powerup Response

In check mode, error checking requires no extra time. In sniff mode, error checking requires an additional 225 nanoseconds for each sniff operation (each double word checked). In either mode, an additional 225 nanoseconds is required to correct one double-word error.

Upon powerup, the ERCC facility is automatically placed in check and sniff modes, and all ERCC interrupts are disabled. Before any other memory operation can occur, the ERCC facility initializes all check code bits. Starting with the lowest memory location and continuing until the top memory location is reached, the facility reads the double word from the location, corrects the check code, and writes the double word along with the correct check code back into the location.

System Status and Special Functions

The S/280 processing system has system management facilities that provide information about system status and service faults and perform several special functions.

A central processor identification register stores information about the type of processor, the microcode revision level, and the size of memory.

System management facilities allow the operating system to perform the following special functions:

- Halt program execution,
- Transfer program control to a system call handler,
- Read the virtual console data switch register,
- Clear powerfail interrupts.

System Instructions

Table 6.5 lists the standard and special formats of instructions for retrieving system information and performing the special functions. Several of these instructions are I/O instructions issued to the central processor. The assembler interprets these instructions using either the standard or special I/O instruction format. Device flags cannot be appended to the special format of these instructions.

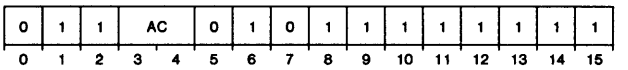
Instruction Mnemonic		
Special	**Standard**	**Action**
—	DOAP *ac*,CPU	Clear powerfail interrupts.
HALT	DOC*[f]* *ac*,CPU	Stop program execution.
NCLID	—	Read central processor identification.
READS *ac*	DIA*[f]* *ac*,CPU	Read virtual console data switch register.
SYC *acs,acd*	—	Transfer control to the system call handler.

Table 6.5 System Instructions

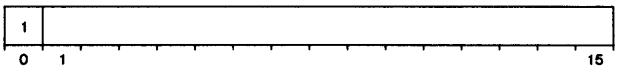
DOAP CPU

Clear Powerfail Interrupts

DOAP *ac*,CPU



Clears powerfail interrupts. The accumulator format before and after the operation is diagrammed below.

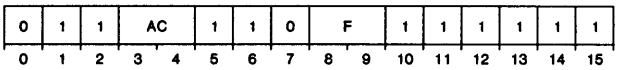


Bits	Name	Contents or Meaning When 1
0	—	Must be 1 to clear powerfail interrupts.
1-15	—	Reserved for future use.

HALT

Halt

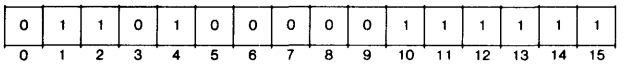
DOC*[f]* *ac*,CPU



Stops user program execution, enters the virtual console, and runs the confidence test.

The DOC CPU mnemonic sets the Interrupt On flag (ION) according to the function specified by *f*, then stops the processor. An accumulator must be specified with this mnemonic even though the accumulator contents remain unchanged.

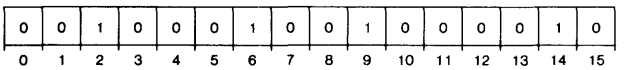
NCLID
Central Processor Identification



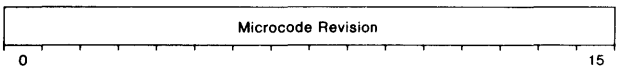
Identifies the processor type and sizes memory.

Loads the central processor identification information into accumulators AC0 through AC3. The accumulator formats after the transfer are diagrammed below.

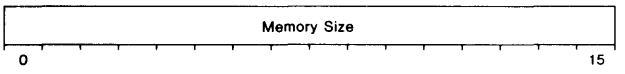
AC0 format



AC1 format



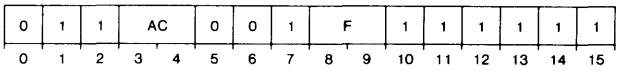
AC2 format



AC	Name	Contents or Function
0	—	Machine's model number, 21102_8.
1	Microcode Revision	Indicates current revision of the central processor microcode.
2	Memory Size	Number of 32 Kbyte blocks of physical memory available minus 1.

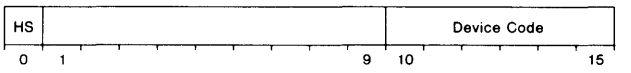
READS
Read Switches

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



Returns contents of virtual console switch register.

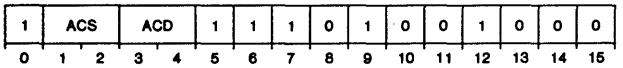
Places the contents of the virtual console switch register into the specified accumulator. After the operation, sets the Interrupt On (ION) flag according to the function specified by *f*. The accumulator format after the transfer is diagrammed below.



Bits	Name	Contents or Function
0	HS	If 1, a high-speed device performs the automatic program boot (load) operation.
1-9	—	Reserved for future use.
10-15	Device Code	Specifies the device to perform the automatic program boot (load) operation.

SYC
System Call

SYC *acs, acd*



Disables user address translation, pushes a standard return block onto the stack, and jumps indirect through location 2. The source and destination accumulators (*acs* and *acd*) remain unchanged. If both accumulators are specified as AC0, no return block is pushed onto the stack and AC0 remains unchanged.

The program counter in the return block contains the address of the instruction immediately following the SYC instruction.

I/O interrupts cannot occur between the execution of the SYC instruction and the execution of the next instruction.

> **NOTE:** *DGC assemblers recognize the mnemonic SCL as equivalent to SYC 1,1 and SVC as equivalent to SYC 0,0.*

Chapter 7

Virtual Console

The virtual console is a firmware program that allows a user to interact with the S/280 computer system through the master terminal. Simple commands entered on the terminal keyboard examine and/or modify computer processor registers and memory locations. A mnemonic mode accepts and displays assembler mnemonic statements as input or output of the processor registers and memory locations. A breakpoint feature stops program execution at selected places for debugging.

Entering the Virtual Console

Upon powerup, the virtual console performs a confidence test to determine if the system is functional enough to boot a program from an I/O device. The virtual console is also entered when

- The Break key on the master terminal is pressed when the LOCK switch on the front console is in the OFF position.
- The reset (RST) switch on the front console is pressed and the LOCK switch is in the OFF position.
- The program executes a *Halt* instruction.
- The program encounters a breakpoint.
- The program completes execution of an instruction in the single-cycle (one-step) mode.

When the virtual console is entered, it displays the contents of the accumulators, the program counter, and the carry bit, and finally the virtual console prompt !. The exception is at powerup when only

Self-Test-OK

and the prompt ! are displayed.

> **NOTE:** *The contents of the program counter displayed after the RST switch is pressed may be invalid.*

The virtual console prompt indicates the virtual console's readiness to accept a command. When the prompt is not preceded by a letter, it indicates that user address translation is disabled. When one of the letters A, B, C, or D precedes the prompt, the letter indicates that user address translation is enabled for user map table A, B, C, or D, respectively. The user map table indicated is used for all address translations. The status of the user/DCH address translator can be changed from within the virtual console as explained under "Address Translation Commands" later in this chapter.

When the virtual console is in control, input/output (I/O) interrupts and I/O protection are disabled and the RUN light on the chassis front console is not lit. The RUN light is only lit when a user program is executing.

Entering Commands

A virtual console command consists of a single character. Some commands require a leading argument that is an octal number or an expression. Valid numbers and expressions are

Digits	Ranging from 0 through 7. If the argument is an address, it must range from 0 through 77777.
Period (.)	Represents the value of the address last used.
+ or −	Entered between valid numbers, the + or − signs direct the virtual console to compute an arithmetic result and replace the original expression with it.

For clarity, all examples in this chapter show data entered by the user in this typeface:

USER ENTRY

and the system's response in this typeface:

SYSTEM RESPONSE

On the terminal, user input and program response are not differentiated.

Correcting Errors

Errors made when entering arguments can be corrected using the Rubout/Delete key or the K command.

Rubout/Delete

The Rubout/Delete key deletes the last character typed on the master terminal. The virtual console echoes the deletion with an underscore (_) on the master terminal. On hardcopy terminals, the underscore indicates that the character preceding it has been deleted. On display terminals, the deleted character is no longer displayed. Typing additional Rubouts deletes digits from right to left.

The Rubout/Delete key has no effect on the + or − symbols. Used after a period, the Rubout/Delete key deletes the rightmost digit of the last address. Since the virtual console only retains six digits at any time, the Rubout/Delete key will not resolve all errors.

Typing Rubouts immediately after opening a cell (defined below), causes the virtual console to delete the rightmost digits of the cell's contents just as though new contents had been entered. New values can be entered for these digits.

K Command

To cancel input in a line, type K. In response, the virtual console prints a question mark (*?*) followed by a New Line and a prompt. It also closes the current cell if it is open. The *?* followed by a New Line and a prompt is also printed in response to a typed character the virtual console does not recognize.

Cells

Several virtual console commands operate on *cells*. A cell is either a memory location (*memory cell*) or an internal register (*internal cell*) such as an accumulator (AC) or the program counter (PC). Each internal register accessible by the virtual console has an internal cell number. Table 7.1 lists these registers and their numbers.

Opening Cells

To examine or modify any cell, it must be opened using one of the commands listed in Table 7.2. The address of the cell is interpreted as a 15-bit logical address which is translated into a 20-bit physical address by the user/DCH address translator if user address translation is enabled. When a memory cell address of more than five octal digits (15 bits) is entered, only the last five digits are used. Leading zeroes are unnecessary. For example, to open the memory location with logical address 5, type

```
5/
```

Opening a cell causes its address and contents to be displayed on the master terminal. The contents are displayed either as an octal number or as both an octal number and its assembler mnemonic equivalent depending on the output mode. Once a cell is opened, its contents can be changed by entering octal digits or instruction mnemonic statements.

Closing Cells

To terminate an expression and close the open cell, use a Carriage Return, Line Feed, or New Line. Note that if Carriage Return is used, it also opens the next cell. Thus Carriage Return is convenient for entering data into consecutive locations.

Number	Cell
0-3	Accumulators AC0 thorugh AC3, respectively.
4	Either 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.
5	Carry bit. Bit 15 of this internal cell is equal to the value of the carry bit.
6	Processor status register.[1]
7	Reserved for future use.
10	Data switch register. Replaces the conventional console data switches. When the system is not in the virtual console, this register can be read with a *Read Switches* instruction (READS).
11	User/DCH address translator status register.[1]
12	Map table entry for user page 31.[2]
13	Page check register.[1]
14	Search mask.[3]
15	Reserved for future use.

Table 7.1 Internal cells

Command	Function
*expr*A	Opens the internal cell with the number specified by octal number *expr*.[4]
expr/	Opens the memory location specified by octal number *expr*.[4]
Carriage Return	Closes the current cell[5] and opens the next consecutive cell.
New Line[3]	Closes the current cell[5] and does not open another cell.
^	Closes the current cell and opens the previous cell.
/	Closes the current cell[5] and opens the memory cell whose address is equal to the contents of the current memory or internal cell.
=	Places the virtual console in octal output mode.
;	Places the virtual console in mnemonic output mode.

Table 7.2 Virtual console cell commands

[1] *Refer to Appendix C for the contents of these registers.*

[2] *Refer to the description of the* Initiate Page Check *(DOC MAP) instruction in the "User/DCH Address Translator" section of Chapter 2 for the contents of this register.*

[3] *Refer to "Search Command" later in this chapter for information on the search mask.*

[4] *The symbol* expr *represents any valid octal number or expression, as described at the beginning of the preceding "Commands" section.*

[5] *Current cell refers to the cell that was opened last.*

[6] *Line Feed on non-ANSI standard keyboards.*

Modes

The virtual console displays values in two modes: octal output and mnemonic output. Its interpretation of input depends on whether a location is opened or closed.

Output Modes

The equal sign (=) places the virtual console in octal ouput mode. In this mode, the octal equivalent of the contents is displayed. For example,

```
!=
!2000/002000 113410
```

Location 2000 contains 113410_8.

The semi-colon (;) places the virtual console in mnemonic output mode. In this mode, the octal equivalent of the cells contents are displayed first and then the assembler instruction mnemonic equivalent of the contents, unless the cell is an accumulator. Only the octal equivalent of an accumulator's contents are displayed. For example,

```
!;
!2000/002000 113410 LMP
```

Location 2000 contains 113410_8, the octal equivalent for the assembler LMP instruction (*Load User Map Table*).

```
!2A 000002 113410
```

AC2 contains 113410_8.

If the = or ; symbol is not entered directly after the prompt, then the virtual console displays either the octal or instruction mnemonic equivalent of the value that has been typed on the line. If neither symbol is used after the virtual console is entered, the octal equivalent is always displayed.

When the location contains the first word of a double-word instruction, the octal contents of the first word are displayed. In mnemonic mode, the assembler mnemonic for the double-word instruction is also displayed. If a Carriage Return was used to terminate the command line that opened the location, then the next location after the one containing the second instruction word is opened. If a location containing the second word of a double-word instruction is opened in mnemonic mode, then the virtual console attempts to display the mnemonic equivalent of the second word.

Input Modes

When a location is *opened*, all letters typed are interpreted as part of an instruction mnemonic. For example:

```
!2000/002000 145201 LDA 2 0
```

Changes the contents of location 2000 (145201_8) to 030000_8, the octal equivalent of the assembler statement LDA 2,0.

When no index mode is entered with the mnemonic, the virtual console uses the same rules as the ECLIPSE assembler to calculate the effective address. That is, for one-word instructions, the virtual console first tries to calculate the effective address using an index mode of 0. If it cannot calculate the address, it tries again using an index mode of 1. When the virtual console cannot calculate the effective address, it returns a question mark (*?*). For double-word instructions, the virtual console always calculates the effective address using an index mode of 1.

> **NOTE:** *The indirect and no-load mnemonic symbols (@ and #, respectively) can appear anywhere and in any quantity after the letters in the appropriate mnemonic and still be interpreted correctly.*

When a location is *not opened*, any letter typed is interpreted as a virtual console command. For example:

```
!10L
```

Loads a program from the programmed I/O device with device code 10 octal.

The virtual console resolves an expression typed without letters into an octal number. Thus, if an expression starting with a + or − is typed, the value of the expression is added to, or subtracted from, the current contents of the cell.

Examples of expression resolution when the last address was 100 follow.

100000_1	is replaced by 100001.
1000000	will be replaced by 000000, since the virtual console only remembers six digits.
.−3	is replaced by 75.
.7	is replaced by 1007.
0-7	is replaced by 177771.
6+.−3	is replaced by 103 (6+100−3=103)
75+_5	is replaced by 102. (75+5=102; the + is not deleted)
60+._	is replaced by 70 (60+10=70; the _ deleted the rightmost 0 of the last address, 100).

If an illegal digit is typed, the virtual console issues a prompt and does not change the cell contents. If more than 16 bits are typed when altering internal or memory cells, only the last 16 bits typed are used.

Examples showing the use of /, Carriage Return (CR), and New Line (NL) follow. In most of these examples, the system outputs an "A" before the prompt. This means that user map table A is enabled and used to translate addresses.

*A/*3A *000003A 000100*<**CR**>

AC3 contains 100. Internal cell 4 is opened as shown next.

000004A 000704 /000704 103000 <**NL**>

PC contained 704. Memory location 704 contains 103000. If the virtual console is in mnemonic mode ADD 0 0 would have been displayed after the 103000. The next memory location is not opened.

*A/*5A *000005A 000001* <**NL**>

Changes the carry bit to 1. The next cell is not opened.

A/ **100***/000100 025037.* <**NL**>

Changes the contents of memory location 100 (025037) to the current address (000100). The next memory location is not opened.

A/ **100***/000100 000100* <**CR**>

Confirms the preceding step. Memory location 101 is opened as shown on the next line.

000101 000602 +**1** <**NL**>

Increments the contents of 101. The next memory location is not opened.

A/./000101 000603

Confirms preceding step.

In all memory location examples above, the virtual console was not in mnemonic mode. Had it been in mnemonic mode, the mnemonic equivalent of the locations' contents would have been displayed after the octal contents.

Function Commands

The virtual console has special function commands in addition to the cells commands. Table 7.3 lists these function commands and the sections that follow explain them.

Program Control Commands

Several function commands control program flow by setting and deleting breakpoints, resuming program flow, and single-stepping through instructions.

Setting and Deleting Breakpoints

The B and D commands set and delete breakpoints.

Typing the *expr*B command sets a breakpoint at the address specified by the argument *expr* using the user map table which is currently enabled. The breakpoint is only valid for this map table. Do not place two breakpoints at the same physical location. If no address is specified, the B command lists all the current breakpoints and their assigned numbers.

The virtual console assigns numbers from 0 to 5 to breakpoints in reverse order — that is, 5 is assigned to the first breakpoint, 4 is assigned to the next breakpoint, and so on. A new breakpoint is always assigned the highest remaining number. For example, if numbers 5 and 3 are assigned, the next breakpoint will be 4, not 2.

Command	Function
*expr*B	Inserts breakpoint at the memory location specified by octal number *expr*. If no *expr* is specified, all breakpoints are listed.
*n*D	Deletes breakpoint number *n* where *n* is a number between 0 and 7. If no *n* is specified, all breakpoints are deleted.
CTRL-G[7]	Executes confidence test.
*n*H	Performs program load from the data channel device whose device code is *n*.
I	Executes an I/O reset instruction (IORST).
K	Cancels entire line just typed and displays a question mark (?).
*n*L	Performs program load from the programmed I/O device whose device code is *n*.
M	Displays contents of all four user map tables currently stored in the user/DCH address translator.
*n*M	Displays and/or alters the entry for logical page *n* in the currently enabled user map table. *n* is a number between 0 and 37_8.
*n*O	Steps through *n* instructions of the user's program.
P	Starts program execution at the memory location specified by the contents of internal cell number 4.
*n*P	Same as P, except the next *n* breakpoints are ignored.
*expr*R	Starts program execution at the memory location specified by octal number *expr*.
*n*S	Searches for the value *n* in physical memory which is accessible using the currently enabled user map table.
U	Enables a user map table and user address translation. Displays a colon (:) after which an A, B, C, or D must be typed to specify the user map table to be enabled. If any other character is entered, user address translation is disabled.

Table 7.3 Virtual console function commands

[7]*CRTL-G is typed by simultaneously pressing the CTRL key and the G on the master terminal.*

Typing the *n*D command deletes a breakpoint, regardless of whether or not user address translation is enabled. If no argument (*n*) is provided, D deletes all breakpoints.

Examples of setting and deleting breakpoints follow.

A/423B

Places a breakpoint at address 423 using user map table A.

A/623B?
A/

Request a breakpoint at a valid location when all six breakpoints are in use. You must delete a breakpoint before setting another. A prompt always follows a *?*.

A/B

Lists all current breakpoints as shown in the next examples.

5 075324

Breakpoint 5 is at address 75324.

3 000423

Breakpoint 3 is at address 423.

A/3D

Deletes breakpoint number 3.

A/12D*?*
A/

Only six breakpoints (numbers 0-5) are valid; no other number is allowed. A prompt always follows a *?*.

When a breakpoint is encountered during user program execution, the virtual console is entered. The address of the instruction at which the breakpoint was encountered is displayed and placed in internal cell 4, and that instruction is not executed. Then the virtual console displays a prompt indicating that the user can now examine and modify any internal cell or memory location.

Single Stepping

The O command executes a single instruction at a time. Typing this command causes a nonmaskable interrupt to occur immediately after 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, executes one instruction, and resumes control.

Typing O without an argument steps through one instruction. After the instruction is executed, the address of the instruction is displayed along with the contents of the PC, the ACs, and the state of the carry bit at the time of execution. After displaying this information, the virtual console displays a prompt.

NOTE: *Typing an O when the program counter points to an* Interrupt Enable *instruction (INTEN) causes both the INTEN and the following instruction to be executed before control returns to the virtual console.*

Typing *n*O steps through *n* instructions, where *n* is an octal number ranging between O and 177777. As each instruction is executed, the address of the instruction is displayed along with the contents of the PC, the ACs, and the state of the carry bit at the time of instruction execution. After *n* instructions have been executed, the virtual console displays a prompt. The *n*O command is a convenient way to locate skips or branches.

Although Breakpoints have no effect during single-stepping, the Break key does. The Break key will stop the virtual console before it finishes stepping through all *n* instructions. In response, the virtual console displays the address of the instruction after the one last executed and then displays a prompt. Interrupts will be honored if the interrupt system is enabled during single-stepping.

Resuming Program Execution

The P and R commands resume program execution after the virtual console is entered through a breakpoint, after single-stepping, or after the Break key is pressed.

Typing P restarts program execution at the location specified by the contents of internal cell 4, which specifies the return address. Typing *n*P restarts program execution in the same manner; however, the next *n* breakpoints are ignored.

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 to resume normal program flow. When the virtual console is entered in any other way, internal cell 4 contains the value of the PC plus one (PC+1); this should also be the next instruction in the normal program flow. In either case, the P instruction produces the required result.

Typing *expr*R restarts program excution at the location specified by *expr*. When the R command is issued, the virtual console inserts all previously specified breakpoints, clears nonmaskable interrupts, and resumes program execution at the logical address (*expr*) using the currently enabled user map table. If no address (*expr*) is specified or the specified address is invalid for the user, the virtual console simply displays a question mark (*?*) followed by a prompt.

Program Load Commands

The *n*L and *n*H commands perform a program load while in the virtual console. Entering *n*L loads a program from the low-speed (programmed I/O) device with device code

equal to *n*. Entering *n*H loads a program from the high-speed (data channel) device with the device code equal to the octal number *n*. If *n* contains more than two octal digits, only the two least significant digits specify the device code. Device code 0 is normally not a valid device code, device code 77 is reserved for the interrupt system and special programmable system functions, and device codes 2, 3, 4, 5, 10, 11, 14, and 43 are reserved for resident devices.

> **NOTE:** *The* n*L command will load a program from a high-speed device if* n *is in the range of* 100000_8 *to* 177776_8.

Once a program load begins, the virtual console is no longer in control. After any automatic program load has been performed — whether initiated by the *n*L or *n*H command or the SYSTEM switch on the front console — the device code of the loading device is placed in the switch register. The device code, shifted left one bit position, is also placed in AC0. If a high-speed program load is specified, the switch register's high-speed bit (bit 0) is set to 1.

I/O Reset Command

The I command causes the virtual console to immediately issue an *I/O Reset* instruction (IORST) to all I/O devices. This instruction disables the interrupt system and user/DCH address translation, and clears all I/O device controller Busy and Done flags.

For more information on the effects on the *I/O Reset* instruction (IORST), refer to the discussion of the interrupt system in Chapter 5.

Search Command

The *expr*S command searches the memory locations specified by the currently enabled user map table for the value of *expr*. The contents of each searched memory location are logically ANDed with the value of the search mask contained in internal cell 14 before the comparison is made. If the result of the AND operation is identical to the contents of the value of *expr*, then the address and contents of the location are displayed as follows.

AAAAAA DDDDDD

where

AAAAAA is the physical memory address
DDDDDD represents the contents of that address

If internal cell 14 contains 0, the search mask is not used.

An example of the use of the S command follows. The example searches the physical memory locations addressed by the currently enabled user map for any I/O instruction to device code 22.

*B!*14A *000014A DDDDDD* 160077 <CR>

Sets the search mask to 160077, thus limiting the comparison to all instructions with I/O format. DDDDDD equals the contents of internal cell 14 when it was opened.

*B!*060022S

Searches for all I/O format instructions to device code 22.

015643 060122
017423 061322
023654 062222

I/O instructions to device code 22 were found in three physical memory locations addressed by user map B.

Address Translation Commands

The status of the user/DCH address translator can be examined and altered by opening internal cell 6, the processor status register, with a 6A command. After typing the 6A command, type in the new status and then close the cell with a New Line. For information on the contents of status register, refer to the section "System Status and Special Functions" in Chapter 6.

The U and M commands enable, examine, and change user address translation. The U command enables a user map table for address translation. In response to this command, the virtual console displays a colon (:). To enable a different user map table and enable user address translation (if disabled), type the A, B, C, or D character associated with the desired map table. Typing *any other character*, disables user address translation. For example, typing A enables user map table A. In response to a valid character, the virtual console displays a prompt reflecting the newly (currently) enabled user map table.

The M command displays and/or alters the physical page to which the specified logical page of the enabled user map table is translated. Typing *n*M displays the physical page to which logical page *n* is translated in the currently enabled user map table together with the state of the write protection flag. *n* is the octal logical page address (0-1777). This information is displayed in the format:

WXPPPP

where

W is state of the write protection flag.
1 denies write access to the page.
0 allows write access to the page.

XPPPPP is an octal number. The 10 least significant bits of its binary equivalent specify the physical page address. This address is always in the range of 0 to 1777_8.

After this information is displayed, it can be altered by typing the new state of the write protection flag and physical page address in the same format followed by a New Line or a Carriage Return.

Entering M without an argument displays the above information for each logical page in each of the four user map tables currently resident in the address translator. In other words, it displays the write protection and physical page address information from all of the entries in the four currently resident map tables.

Examples showing the use of the U and M commands follow.

/U:B

User address translation is disabled. Command enables user address translation with user map table B.

B/U:C

Address translation is enabled with user map table B. Command enables address translation with user map table C.

B/11A 000011A 000051 77 <NL>

The user/DCH address translator status register contains 51_8. The command line changes the register's contents to 77_8.

B/11A 000011A 000077 <NL>

Confirms preceding step.

B/U:B

New translator status takes effect.

B/U: <NL>

Disables user address translation.

C/32M 00032M 000132 **100120** <NL>
C/

User map table C translates logical page 26 ($32_8 = 26_{10}$) into physical page 90 ($132_8 = 90_{10}$) and allows write access to this page. Command line changes user map table C to translate logical page 26 to physical page 80 ($120_8 = 80_{10}$) and denies write access to this page (write protects it).

C/5M 000005 100030 <CR>
000006M 000077 <NL>
C/

User map table C translates logical page 5 into physical page 24 ($30_8 = 24_{10}$) and denies write access to this page (write protects it). The same map table translates logical page 6 into physical page 63 ($77_8 = 63_{10}$) and allows write access to this page.

Confidence Test Command

The CTRL-G command runs a confidence test which detects faults that would prevent the loading of software. This is the same confidence test that runs automatically on powerup.

> **CAUTION:** *The memory-test section of the confidence test overwrites all physical memory; therefore, a program load must be performed after the confidence test.*

As each section of the confidence test passes, a single letter of the following message is displayed on the master terminal:

Self-Test-OK

The entire message is displayed if the confidence test completes successfully.

After issuing the complete message, the virtual console issues its prompt (!) and waits for a virtual console command to be entered. If a partial message or no message at all is displayed, then the confidence test failed to complete successfully because it detected an error. Pressing the front console system switch to the Reset position usually causes the virtual console to issue its prompt (!).

The partial *Self-Test-OK* message, which is displayed when the confidence test detects an error, indicates the sections that completed successfully. Table 7.4 lists the area tested for each successive letter displayed.

Message	Areas Tested
blank	Central processor (microcode sequencer and arithmetic logic unit, system clock circuitry), programmed I/O facility, interprocessor (intersystem) data bus, real-time clock, programmable interval timer.
S	Processor status register, central processor (instruction register, program counter, effective address calculation).
Se	User portion of user/DCH address translator, physical address bus.
Sel	I/O control unit, data channel portion of the user/DCH address translator, BMC address translator.
Self	Memory control unit, interprocessor (intersystem) data bus, cache, virtual console.
Self-	Central processor (instruction decoding).
Self-T	Central processor (instruction decoding, effective address calculation).
Self-Te	Error checking and correction, memory control unit, physical address bus, I/O control unit.
Self-Tes	Memory, memory control unit.
Self-Test	Memory, refresh logic.
Self-Test-	Hardware floating-point processor.
Self-Test-O	I/O control unit, backpanel, I/O data bus, power supply controller (UPSC).
Self-Test-OK	ECLIPSE instructions.

Table 7.4 Confidence test error messages

Chapter 8

Powerup and Initialization

This chapter describes the sequence of events that occur when an S/280 computer is powered up, the programming for powerfail/autorestart conditions, and the initial state of the computer immediately following powerup, system reset, and I/O reset.

Powerup Sequence

This section discusses the sequence of events when powerup occurs normally and the UPSC's handling of faults on powerup.

Normal Powerup

When the universal power supply controller (UPSC) receives power, the controller automatically runs a two-second self-test to ensure that its microprocessor is working. Next, the UPSC turns on the three front-panel lights and sequences power to the chassis in an orderly manner while monitoring for failures. If it detects a failure, it shuts down and may attempt to startup two more times. When it shuts down, it displays a fault code on the front panel lights.

Once the components in the chassis have power, the UPSC returns the lights to their normal state and starts monitoring power status. At this time, the central processor starts its powerup process and the UPSC continues to monitor power status.

While the UPSC is running its self-test, the rest of the systems are in an idle state. As soon as the central processor detects that voltages are within the required limits, it automatically performs a system reset to initialize the computer as described in the next section. Next, it runs a confidence test to check the parts of the computer that are needed to load software. The test runs for less than a minute.

If the confidence test finds an error, the central processor displays a one-letter code identifying the error on the master console. Error codes and their meanings are listed in Table 7.4. After displaying the error code, the central processor waits for the operator to enter the virtual console by pressing the Break key on the master terminal.

When the confidence test finishes without finding an error, then the central processor displays the virtual console prompt on the master terminal and waits for the operator to enter a virtual console command on the master terminal.

Powerup Faults

The UPSC handles faults as follows.

- Powers down the system, switches on battery backup, or takes no corrective action.
- Updates the fault code register.
- Flashes the fault type on the front panel lights for certain faults.
- Notifies the operating system by generating a program interrupt if UPSC interrupts are enabled.

The UPSC usually powers down the system for all faults except VNR undervoltage conditions, fan failures, and battery backup failures. If the fault is critical, such as an overvoltage condition, the UPSC powers down the system immediately. If the fault is less severe — an overcurrent condition for example — the UPSC only brings the power down when the fault persists for at least 1 millisecond. The exceptions are overtemperature faults and fan failures which are allowed to persist for about 15 seconds.

A VNR undervoltage fault occurs when the ac source voltage falls below specification. In this case, the UPSC switches on battery backup if is is present and enabled; otherwise, the UPSC powers down the system and causes a non-maskable powerfail interrupt, provided the operating system has not yet disabled powerfail interrupts.

The UPSC takes not corrective action for battery backup failures.

Powerfail/Autorestart Programming

When the ac source voltage falls below specification, the UPSC detects a VNR undervoltage fault. The UPSC normally responds by signaling the I/O control unit to set the Powerfail flag to 1 and generate a non-maskable powerfail interrupt from device code 0. The UPSC also

initiates its own interrupt request (device code 4); however, this request has lower priority than the powerfail interrupt. When the operating system issues an *Interrupt Acknowledge* instruction (INTA) in response to the interrupt request, the I/O control unit responds by returning device code 0. Since an INTA instruction issued when no device is interrupting also results in a device code 0 response, the interrupt handler must issue a *CPU Skip* instruction that tests the Powerfail flag (SKPDN CPU). If the Powerfail flag is 1, the interrupt handler should respond by entering a powerfail routine. The effect of the powerfail routine depends on whether battery backup is present.

The powerfail routine can clear the powerfail interrupt with a *Clear Powerfail Interrupt* instruction (DOAP CPU).

If battery backup is present, the powerfail routine should save the state of the processor in system memory, load a return instruction into physical memory location 0, and then execute a *Halt* instruction. If no battery backup is present, the routine should simply execute a *Halt* instruction. Whenever a *Halt* instruction is executed, the system stops user program execution and enters the virtual console.

In a system with full battery backup that saves the state of the processor on disk, the interrupt handler can turn off battery backup after completing this operation to shorten battery recharge time and extend battery life. To turn off battery backup, the interrupt handler should issue an *Enable UPSC Fault Interrupts* instruction (DOAS UPSC) that sets the disable battery backup bit to 1.

When a powerfail interrupt occurs, the interrupt handler does not have to enter a powerfail routine and execute a *Halt* instrution. Instead, the interrupt handler can do one of the following.

- Ignore the powerfail interrupt by disabling the interrupt system with an *Interrupt Disable* instruction (INTDS). This instruction disables all program interrupt requests from any device, including interrupt requests from the UPSC.
- Disable powerfail interrupts with an *Enable UPSC Interrupts* instruction (DOAS UPSC) which sets the alternate powerfail mode flag to 1. This instruction prevents the UPSC from notifying the I/O control unit of powerfail conditions, and thus only disables non-maskable powerfail interrupts initiated by the I/O control unit. The central processor will still receive the interrupt request initiated by the UPSC because of the powerfail condition, if UPSC interrupts are enabled, and from any other device with its interrupts enabled. This means that when power fails, the interrupt handler can still set up the UPSC to generate a fault interrupt as described in the next section.

In either case, the processor continues to execute instructions until dc voltages fall below specification. At this point, however, the processor automatically resets the system, stopping instruction execution. The results of the instruction executing when a system reset occurs are indeterminate. For this reason, neither method is recommended, unless the system is backed up by an uninterruptible power source which is supplied by the user. In this case, the interrupt handler should disable non-maskable powerfail interrupts as described above, and continue to log S/280 powerfails and handle other power system faults by enabling UPSC interrupts as described in the next section.

When power is restored to a system supported by battery backup, the powerfail interrupt is automatically cleared. If the powerfail routine saved the state of the processor and stored the address of the return instruction in location 0, then control can be returned to the the user's program. If the front panel is locked, this happens automatically when power is restored because the processor executes an indirect jump through location 0. If the front panel is unlocked, the processor enters the virtual console and issues a prompt to the master terminal. The user can continue the program by entering 0R on the master terminal.

Initialization

After powerup, a system reset, or the execution of the I/O Reset instruction (IORST), the S/280 system is in a given initial state.

When the system first powers up or after a system reset, the following conditions occur.

- The contents of the cache are declared invalid.
- The contents of main memory are indeterminate.
- All address translation is disabled, which means that logical addresses are equal to physical addresses.
- Page 31 in the user/DCH address translator contains 31_{10} (37_8), which means logical page 31 addresses equal physical page 31 addresses.
- The contents of all map tables in the user/data channel address translator and the BMC address translator are undefined.
- Check and sniff modes of the ERCC unit are enabled and ERCC interrupts are disabled.
- The processor status register is initialized and bits 0 through 9 of the floating-point status register are set to 0.
- The Interrupt On (ION) flag is set to 0, which disables the interrupt system, and the I/O interrupt priority mask is set to 0.
- All device Busy and Done flag are set to 0.
- The programmable interval timer (PIT) is stopped.
- Ac line frequency is selected as the clock frequency for the real-time clock (RTC).

- The central processor stops user program execution and enters virtual console mode.

After the execution of the *I/O Reset* instruction (IOSRT), the following conditions occur.

- All address translation is disabled, which means that logical addresses are equal to physical addresses.
- Page 31 in the user/DCH address translator contains 31_{10} (37_8), which means logical page 31 addresses equal physical page 31 addresses.
- Check and sniff modes of the ERCC unit are enabled and ERCC interrupts are disabled.
- The processor status register and bits 0 through 9 of the floating-point status register are set to 0.
- The I/O interrupt priority mask is set to 0.
- All device Busy and Done flags are set to 0.
- The programmable interval timer (PIT) is stopped.
- Ac line frequency is selected as the clock frequency for the real-time clock (RTC).

Appendix A

Instruction Summary

The Instruction Summary lists the S/280 computer instructions alphabetically by instruction mnemonic, giving the format, base octal value, data type used, action performed, and location contents before and after instruction execution.

The following abbreviations are used throughout the summary:

+	Addition
−	Subtraction or negation
()()	Multiplication
inter	Intermediate number

Instruction Format	Operation	Before	After
ADC *[c][sh][#] acs,acd[,skip]* 102000	Fixed-point ACS+ACD=ACD	ACS=unsigned integer ACD=unsigned integer	Unchanged Result
ADD *[c][sh][#] acs,acd[,skip]* 103000	Fixed-point ACS+ACD=ACD	ACS=unsigned integer ACD=unsigned integer	Unchanged Result
ADDI *i,ac* 163770	Fixed-point AC+I=AC	AC=unsigned integer I=unsigned integer	Result Unchanged
ADI *n,ac* 100010	Fixed-point AC+*n*=AC	AC=unsigned integer *n*=unsigned integer (1-4)	Result Unchanged
ANC *acs,acd* 100610	Fixed-point ACS AND ACD=ACD	ACS=unsigned integer ACD=unsigned integer	Unchanged Result
AND *[c][sh][#] acs,acd[,skip]* 103400	Fixed-point ACS AND ACD=ACD	ACS=unsigned integer ACD=unsigned integer	Unchanged Result
ANDI *i,ac* 143770	Fixed-point AC AND immediate field=AC	AC=unsigned integer immediate field=unsigned integer	Result Unchanged
BAM 113710	Fixed-point memory location+AC0=memory location	AC0=addend AC1=number of words AC2=source address AC3=destination address	Addend 0 Last+1 Last+1
BLM 133710	Fixed-point memory location = memory location	AC1=number of words AC2=source address AC3=destination address	0 Last+1 Last+1
BTO *acs,acd* 102010	Bit (ACS and ACD) (Bit=1)	ACS=word pointer ACD=word offset +bit pointer Memory location = address bit	Unchanged Unchanged 1
BTZ *acs,acd* 102110	Bit (ACS and ACD) (Bit=0)	ACS=word pointer ACD=word offset +bit pointer Memory location = Address bit	Unchanged Unchanged 0
CLM *acs,acd* 102370	Fixed-point ACS>L and ACS<H=skip If ACS=ACD	ACS=2's complement number ACD=L address ACS=2's complement number L=next word H=next word	Unchanged Unchanged Unchanged Unchanged Unchanged
CMP 157650	Character string1 compared to string2	AC0=String2 number of bytes AC1=String1 number of bytes AC2=String2 byte pointer AC3=String1 byte pointer	0 or unpredictable result Result Last+1 Last+1
CMT 167650	Character memory location = memory location Delimiter Check made	AC0=delimiter table address AC1=length and direction of string AC2=destination byte pointer AC3=source byte pointer	Delimiter table address remaining number of bytes in string Last+1 or to failing byte Last+1 or to failing byte
CMV 153650	Character memory location = memory location Carry = relative length	AC0=destination number of bytes AC1=source number of bytes AC2=destination byte pointer AC3=source byte pointer	0 0 or unpredictable result Last+1 Last+1

Instruction Format	Operation	Before	After
CTR 163650	Character memory location = memory location Translates	AC0=Translation table byte pointer AC1=1 and 2's complement bytes AC2=destination byte pointer AC3=source byte pointer	Unchanged 0 Last+1 Last+1
	or String1 is compared to string2 Translates	AC0=translation table byte pointer AC1=length of string and number of bytes AC2=string2 byte pointer AC3=string1 byte pointer	Unchanged Result Last+1 or to failing byte Last+1 or to failing byte
COB *acs,acd* 102610	Bit ACS (1's)+ACD =ACD	ACS=unsigned integer ACD=2's complement number	Unchanged Result
COM *[c][sh][#] acs,acd[,skip]* 100000	Fixed-point ACS=ACD	ACS=unsigned integer ACD=2's complement number	Unchanged Result
DAD *acs,acd* 100210	Decimal ACS+ACD=ACD	ACS=binary-coded decimal ACD=binary-coded decimal	Unchanged Result
DHXL *n,ac* 101610	Fixed-point AC and AC+1 hex shift left	AC=high order of number AC+1=low order of number *n*=unsigned integer (1-4)	Result Result Unchanged
DHXR *n,ac* 101710	Fixed-point AC and AC+1 Hex shift right	AC=high order of number AC+1=low order of number *n*=unsigned integer (1-4)	Result Result Unchanged
DIA *[f] ac,device*	I/O device (A buffer) = AC	A Buffer=unsigned integer	AC=Result
DIA*[f] ac,***ERCC**	Fault address register = AC	Fault address register = physical address of faulty data	AC = physical address of faulty data (low-order bits)
DIA *ac,***MAP**	User/DCH address translator status register = AC	User/DCH address translator status register = current translator status	AC = current translator status
DIA*[f] ac,***PIT**	PIT interval counter = AC	PIT interval counter = current interval count	AC = current PIT interval count
DIA*[f] ac,***TTI**	Asynchronous line input buffer = AC	Asynchronous line input buffer = character read	AC = character read
DIA*[f] ac,***UPSC**	Power system status register = AC	Power system status register = current power system status	AC = current power system status
DIB *[f] ac,device*	I/O device (B buffer) = AC	B Buffer=unsigned integer	AC=Result
DIB*[f] ac,***ERCC**	Fault code register and fault address = AC	Fault code register = last fault code. Fault address register = physical address of faulty data	AC = last fault code and physical address of faulty data (high-order bits)
DIC *[f] ac,device*	I/O device (C buffer) = AC	C Buffer=unsigned integer	AC=Result
DIC*[f] ac,***BMC**	BMC status register = AC	BMC status register = current BMC status	AC = current BMC status
DIC *ac,***MAP**	Page check register = AC	Page check register = map table entry	AC = map table entry
DIV 153710	Fixed-point (AC0 and AC1)/AC2	AC0=high-order number unsigned integer AC1=low-order number unsigned integer AC2=divisor unsigned integer	Remainder Quotient Unchanged

Instruction Format	Operation	Before	After
DIVS 157710	Fixed-point (AC0 and AC1)/AC2	AC0=high-order 2's complement AC1=low-order 2's complement AC2=divisor 2's complement	Remainder Quotient Unchanged
DIVX 137710	Fixed-point (AC0 and AC1)/AC2	AC0=sign of AC1 AC1=2's complement AC2=divisor 2's complement	Remainder Quotient Unchanged
DLSH *acs,acd* 101310	Fixed-point ACD and (ACD+1) shift left/right	ACS=2's complement for shift ACD=high order ACD+1=low order	Unchanged Result Result
DOA *[f] ac,device*	I/O AC=device (A buffer)	AC=unsigned integer	Unchanged A buffer = Result
DOA*[f] ac*,**ERCC**	AC = ERCC A buffer	AC = control information	Unchanged ERCC A buffer = control information
DOA *ac*,**MAP**	AC = user/DCH address translator status register	AC = translation control information	Unchanged User/DCH address translator status register = translation control information
DOA*[f] ac*,**PIT**	AC = interval select register	AC = number of increments in interval	Unchanged Interval select register = number of increments in interval
DOA*[f] ac*,**RTC**	AC = RTC frequency select register	AC = clock frequency	Unchanged RTC frequency select register = clock frequency
DOA*[f] ac*,**TTO**	AC = asynchronous line output buffer	AC = character	Unchanged Asynchronous line input buffer = character written
DOAP *ac*,**UPSC**	AC = UPSC A buffer	AC = status select information	Unchanged UPSC A buffer = status select information
DOAP CPU	Clear powerfail interrupts	AC = clear powerfail interrupt bit	Unchanged
DOAS *ac*,**UPSC**	AC = UPSC control register	AC = control information	Unchanged UPSC control register = control information
DOB*[f] ac*,**BMC**	AC = BMC address translator B buffer	AC = BMC map table transfer information	Unchanged B buffer = BMC map table transfer information
DOB*[f] ac*,**BMC**	AC = BMC address translator B buffer	AC = BMC map entry information	Unchanged B buffer = BMC map entry information
DOB *[f] ac,device*	I/O AC=device (B buffer)	AC=unsigned integer	Unchanged B buffer = Result Unchanged
DOB *ac*,**MAP**	AC = page 31 register	AC = physical page address	Page 31 register = physical page address
DOC*[f] ac*,**BMC**	AC = BMC map table entry selector	AC = map entry count	Unchanged BMC map table entry selector = map entry count

Instruction Format	Operation	Before	After
DOC *[f] ac,device*	I/O AC=device (C buffer)	AC=unsigned integer	Unchanged C buffer = Result Result
DOC *ac,MAP*	AC = User/DCH address translator C buffer	AC = page check control information	Unchanged C buffer = page check control information
DSB *acs,acd* 100310	Decimal ACD−ACS=ACD	ACS=binary coded decimal ACD=binary coded decimal	Unchanged Result
DSPA *ac,[@]displ.[,index]* 142710	Fixed-point AC<L or AC>H then (E−L)+unsigned integer address	AC=2's complement PC=PC	Unchanged Address of PC
DSZ *[@]displacement[,index]* 014000	Fixed-point memory location = memory location − 1. If the value is 0, then skip.	Memory location = Unsigned integer	Unsigned integer
EDSZ *[@]displacement[,index]* 116070	Fixed-point memory location = memory location − 1. If the value = 0, then skip.	Memory location = Unsigned integer	Unsigned integer
EISZ *[@]displacement[,index]* 112070	Fixed-point memory location = memory location + 1. If the value = 0, then skip.	Memory location = Unsigned integer	Unsigned integer
EJMP *[@]displacement[,index]* 102070	Fixed-point calculated effective address =PC	PC=PC	Calculated effective address
EJSR *[@]displacement[,index]* 106070	Fixed-point calculated effective address =PC	PC=PC AC3=unknown	Calculated effective address PC+1
ELDA *ac,[@]displ.[,index]* 122070	Fixed-point memory location = AC	AC=unknown Memory location = unsigned integer	Unsigned integer Unchanged
ELDB *ac,displacement[,index]* 102170	Byte memory location = AC (right)	AC=unknown Memory location = unsigned integer	Unsigned integer Unchanged
ELEF *ac,[@]displ.[,index]*	Fixed-point calculated effective address = AC	AC= unknown	Calculated effective address Bit 0 = 0
ESTA *ac,[@]displ.[,index]* 142070	Fixed-point AC=memory location	Memory location = unknown AC=unsigned integer	AC Unchanged
ESTB *ac,displacement[,index]* 122170	Byte AC right half = memory location	Memory location=unknown AC=unsigned integer	AC (right) Unchanged
FAB *fpac* 143050	Floating-point absolute value of (FPAC) =FPAC	FPAC=floating-point number FPSR(N,Z)	Absolute value of sign (floating-point number) Updated
FAD *facs,facd* 100150	Floating-point FACS+FACD =FACD	FACS=Floating-point number FACD=Floating-point number FPSR(N,Z)	Unchanged Floating-point (double precision) Updated
FAMD *fpac,[@]displ.[,index]* 101150	Floating-point memory location+FPAC =FPAC	Memory location = double floating-point number D FPAC=Floating-point number FPSR(N,Z)	Unchanged Floating-point (double precision) Updated

Instruction Format	Operation	Before	After
FAMS *fpac,[@]displ.[,index]* 101050	Floating-point memory location+FPAC =FPAC	Memory location = single floating-point number S FPAC=Floating-point number FPSR(N,Z)	Unchanged Floating-point (single precision) Updated
FAS *facs,facd* 100050	Floating-point FACS+FACD =FACD	FACS=Floating-point number FACD=Floating-point number FPSR(N,Z)	Unchanged Floating-point (single precision) Updated
FCLE 153350	Floating-point FPSR bits 0-4=0	FPSR ANY=unknown OVF=unknown UNF=unknown DVZ=unknown MOF=unknown	 0 0 0 0 0
FCMP *facs,facd* 103450	Floating-point FACS compared to FACD	FACS=Floating-point number FACD=Floating-point number FPSR(N,Z)	Unchanged Unchanged Updated
FDD *facs,facd* 100750	Floating-point FACD/FACS =FACD	FACS=Floating-point number FACD=Floating-point number FPSR(N,Z)	Unchanged Floating-point (double-precision) Updated
FDMD *fpac,[@]displ.[,index]* 101750	Floating-point FPAC/memory location =FPAC	Memory location = double floating-point number D FPAC=Floating-point number FPSR(N,Z)	Unchanged Floating-point number (double-precision) Updated
FDMS *fpac,[@]displ.[,index]* 101650	Floating-point FPAC/memory location =FPAC	Memory location = single floating-point number S FPAC=Floating-point number FPSR(N,Z)	Unchanged Floating-point number (single-precision) Updated
FDS *facs,facd* 100650	Floating-point FACD/FACS =FACD	FACS=Floating-point number FACD=Floating-point number FPSR(N,Z)	Unchanged Floating-point number (single-precision) Updated
FEXP *fpac* 123150	Floating-point AC0=FPAC	AC0=unknown FPAC=Floating-point number FPSR(N,Z)	Unchanged Floating-point number (new exponent) Updated
FFAS *ac,fpac* 102650	Floating-point integer (FPAC) =AC	FPAC=Floating-point number AC0=unknown	Unchanged Fixed-point number
FFMD *fpac,[@]displ.[,index]* 102750	Floating-point absolute value of sign (FPAC) =memory location	FPAC=Floating point number Memory location = unknown	Unchanged Fixed-point (double-precision)
FHLV *fpac* 163150	Floating-point FPAC=FPAC/2	FPAC=Floating-point number FPSR(N,Z)	Result Updated
FINT 143150	Floating-point integer (FPAC) =FPAC	FPAC=Floating-point number FPSR(N,Z)	Result Updated
FLAS *ac,fpac* 102450	Floating-point AC=FPAC	AC=2's complement number FPAC=unknown FPSR(N,Z)	Unchanged Floating-point (single-precision) Updated

Instruction Format	Operation	Before	After
FLDD *fpac,[@]displ.[,index]* 102150	Floating-point memory location = FPAC	Memory location = double floating-point number FPAC=unknown FPSR(N,Z)	Unchanged Floating-point (double-precision) Updated
FLDS *fpac,[@]displ.[,index]* 102050	Floating-point memory location=FPAC	Memory location = single floating-point number FPAC=unknown FPSR(N,Z)	Unchanged Floating-point number (single-precision) Updated
FLMD *fpac,[@]displ.[,index]* 102550	Floating-point memory location=FPAC	Memory location = 2's complement double floating-point number FPAC=unknown FPSR(N,Z)	Unchanged Floating-point number (double-precision) Updated
FLST *[@]displacement[,index]* 123350	Floating-point memory location=FPSR	Memory location = single floating-point number FPSR(all)	Unchanged Updated
FMD *facs,facd* 100550	Floating-point (FACD)(FACS) =FACD	FACS=Floating-point number FACD=Floating-number FPSR(N,Z)	Unchanged Floating-point number (double-precision) Updated
FMMD *fpac,[@]displ.[,index]* 101550	Floating-point (FPAC)(memory location) =FPAC	Memory location = double floating-point number FPAC=Floating-point number FPSR(N,Z)	Unchanged Floating-point number(double-precision) Updated
FMMS *fpac,[@]displ.[,index]* 101450	Floating-point (FPAC) (memory location) =FPAC	Memory location = single floating-point number FPAC=Floating-point number FPSR(N,Z)	Unchanged Floating-point number (single-precision) Updated
FMOV *facs,facd* 103550	Floating-point FACS=FACD	FACS=Floating-point number FACD=unknown FPSR(N,Z)	Unchanged FACS Updated
FMS *facs,facd* 100450	Floating-point (FACD)(FACS) =FACD	FACS=Floating-point number FACD=Floating-point number FPSR(N,Z)	Unchanged Floating-point number (single-precision) Updated
FNEG *fpac* 163050	Floating-point −FPAC=FPAC	FPAC=Floating-point number FPSR(N,Z)	Floating-point number Updated
FNOM *fpac* 103050	Floating-point norm (FPAC)=FPAC	FPAC=Floating-point number FPSR(N,Z)	Floating-point number Updated
FNS 103250	Floating-point never skip	PC=PC	PC
FPOP 167350	Floating-point stack=pop	stack=18 words	FPAC3 FPAC2 FPAC1 FPAC0 FPSR

Instruction Format	Operation	Before	After
FPSH 163350	Floating-point push=stack		Stack= FPSR FPAC0 FPAC1 FPAC2 FPAC3
FRH *fpac* 123050	Floating-point FPAC (high order) =AC0	FPAC=Floating-point number AC0=unknown	Unchanged Floating-point number High 16 bits
FSA 107250	Floating-point skip always	PC=PC	PC+1
FSCAL *fpac* 103150	Floating-point AC0− FPAC (exponent) =FPAC (mantissa is shifted) AC0=FPAC (exponent)	AC0=unsigned integer FPAC=Floating-point number FPSR(N,Z)	Unchanged Result Updated
FSD *facs,facd* 100350	Floating-point FACS−FACD =FACD	FACS=Floating-point number FACD=Floating-point number FPSR(N,Z)	Updated Floating-point number (double-precision) Updated
FSEQ 113250	Floating-point FPSR (if Z=0 then skip)		
FSGE 127250	Floating-point FPSR (if N=0 then skip)		
FSGT 137250	Floating-point FPSR (if Z and N=0 then skip)		
FSLE 133250	Floating-point FPSR (if Z or N=0 then skip)		
FSLT 123250	Floating-point FPSR (if N=1 then skip)		
FSMD *fpac,[@]displ.[,index]* 101350	Floating-point FPAC−memory location =FPAC	Memory location = double floating-point FPAC=Floating-point number FPSR(N,Z)	Unchanged Floating-point number (double-precision) Updated
FSMS *fpac,[@]displ.[,index]* 101250	Floating-point FPAC−memory location =FPAC	Memory location = single floating-point FPAC=Floating-point number FPSR(N,Z)	Unchanged Floating-point number (single-precision) Updated
FSND 147250	Floating-point FPSR (if DVZ=0 then skip)		
FSNE 117250	Floating-point FPSR (if Z=0 then skip)		
FSNER 177250	Floating-point FPSR (if 1-4=0 then skip)		
FSNM 143250	Floating-point FPSR (if MOF=0 then skip)		
FSNO 163250	Floating-point FPSR (if OVF=0 then skip)		

Instruction Format	Operation	Before	After
FSNOD 167250	Floating-point FPSR (if OVF and DVZ=0 then skip)		
FSNU 153250	Floating-point FPSR (if UNF=0 then skip)		
FSNUD 157250	Floating-point FPSR (if UNF and DVZ=0 then skip)		
FSNUO 173250	Floating-point FPSR (if UNF and OVF=0 then skip)		
FSS *facs,facd* 100250	Floating-point FACD−FACS=FACD	FACS=Floating-point number FACD=Floating-point number FPSR(N,Z)	Unchanged Floating-point number (single-precision) Updated
FSST *[@]displacement[,index]* 103350	Floating-point FPSR=memory location	Memory location = unknown FPSR=FPSR	FPSR Unchanged
FSTD *fpac,[@]displ.[,index]* 102350	Floating-point FPAC=memory location	FPAC=Floating-point number Memory location = unknown	Unchanged Floating-point number (double-precision)
FSTS *fpac,[@]displ.[,index]* 102250	Floating-point FPAC=memory location	FPAC=Floating-point number Memory location = unknown	Unchanged Floating-point number (single-precision)
FTD 147350	Floating-point FPSR (5=0)	FPSR (Trap enable bit)	0
FTE 143350	Floating-point FPSR (5=1)	FPSR (Trap enable bit)	1
HALT	Fixed-point stop		
HLV *ac* 143370	Fixed-point AC/2=AC	AC=2's complement	Result
HXL *n,ac* 101410	Fixed-point hex shift left =AC	*n*=unsigned integer (1-4) AC=unsigned integer	Unchanged Result
HXR *n,ac* 101510	Fixed-point hex shift right =AC	*n*=unsigned integer (1-4) AC=unsigned integer	Unchanged Result
INC *[c][sh][#] acs,acd[,skip]* 101400	Fixed-point ACS+1=ACD	ACS=unsigned integer ACD=unknown	Unchanged Result
IOR *acs,acd* 100410	Fixed-point ACS or ACD=ACD	ACS=unsigned integer ACD=unsigned integer	Unchanged Result
IORI *i,ac* 103770	Fixed-point i or AC=AC	I=unsigned integer AC=unsigned integer	Unchanged Result
ISZ *[@]displacement[,index]* 010000	Fixed-point if memory location = memory location +1 then skip	Memory location = unsigned integer	Unsigned integer+1
JMP *[@]displacement[,index]* 000000	Fixed-point calculated effective address	PC=PC AC3=unknown	Calculated effective address
JSR *[@]displacement[,index]* 004000	Fixed-point calculated effective address=PC	PC=PC	PC=calculated effective address

Instruction Format	Operation	Before	After
LDA *ac,[@]displ.[,index]* 020000	Fixed-point memory location = AC	AC=unknown	AC3=PC+1 Unchanged
LDB *acs,acd* 102710	Byte memory location = ACD	ACS=Byte pointer ACD=unknown Memory location=byte	Unchanged ACD=byte Memory location = unchanged
LEF *ac,[@]displ.[,index]* 060000	Fixed-point calculated effective address = AC	AC=unknown	Address
LMP 113410	Map memory location = map	AC1=unsigned integer loaded AC2=1st address	0 Last+1
LOB *acs,acd* 102410	Fixed-point ACS(0s)+ACD =ACD	ACS=unsigned integer ACD=2's complement number	Unchanged Result
LRB *acs,acd* 102510	Fixed-point ACS(0s)+ACD =ACD ACS (high order 0=1)	ACS=unsigned integer ACD=2's complement number	New unsigned integer Result
LSH *acs,acd* 101210	Fixed-point ACD(shifted) =ACD	ACS=2's complement (±) ACD=unsigned integer	Unchanged Result
MOV *[c][sh][#] acs,acd[,skip]* 101000	Fixed-point ACS=ACD	ACS=unsigned integer ACD=unsigned integer	Unchanged ACS
MSP *ac* 103370	Fixed-point stack pointer + AC = stack pointer	AC=2's complement number Stack pointer=unsigned integer	Unchanged Result
MUL 143710	Fixed-point (AC1)(AC2)=unsigned integer Unsigned integer + AC0= AC0 and AC1	AC0=intermediate unsigned integer AC1=unsigned integer AC2=unsigned integer	High result Low result Unchanged
MULS 147710	Fixed-point (AC1)(AC2)=unsigned integer Unsigned integer + AC0= AC0 and AC1	AC0=intermediate 2's complement number AC1=2's complement number AC2=2's complement number	High result Low result Unchanged
NCLID 064077	Processor identication register = AC0 and AC1 and AC2	AC0 = unknown AC1 = unknown AC2 = unknown	AC0 = machine model number 21102_8 AC1 = processor microcode revision AC2 = memory size
NEG *[c][sh][#] acs,acd[,skip]* 163050	Fixed-point −ACS=ACD	ACS=unsigned integer ACD=unknown	Unchanged Result
NIO *[f] device*	I/O device flags set		
NIOP MAP	Address translation	See instruction	
POP *acs,acd* 103210	Fixed-point stack=ACS> ACD	Stack=unknown	ACS−ACD
POPB 107710	Fixed-point stack=destination	Stack=5 words	−5 words. Return block
POPJ 117710	Fixed-point stack=PC	Stack=1 word PC=unknown	−1 word. Top word of stack

Instruction Format	Operation	Before	After
PSH *acs,acd* 103110	Fixed-point ACS>ACD=stack	Stack=unknown	ACS−ACD
PSHJ *[@]displacement[,index]* 102270	Fixed-point PC+1=stack Calculated effective address=PC	PC=PC Stack=unknown	Calculated effective address PC+1
PSHR 103710	Fixed-point PC+2=stack	Stack=unknown	PC+2
RSTR 167710	Fixed-point stack=destination	Stack=9 words	−9 words. Destination= Return block + Stack fault address + Stack limit + Frame pointer + Stack pointer.
RTN 127710	Fixed-point stack=destination	Stack pointer = stack pointer Stack=5 words Destination=unknown	Stack pointer −5 words Carry +PC = 1st word AC3 = 2nd word AC2=3rd word AC1 = 4th word AC0 = 5th word
SAVE *i* 163710	Fixed-point 5 words+I=stack	Stack=unknown	AC0 AC1 AC2 Frame pointer Carry+AC3
SBI *n,ac* 100110	Fixed-point AC−*n*=AC	AC=unsigned integer *n*=unsigned integer (1-4)	Result Unchanged
SGE *acs,acd* 101110	Fixed-point if ACS=ACD then skip	ACS=2's complement ACD=2's complement	Unchanged Unchanged
SGT *acs,acd* 101010	Fixed-point if ACS=ACD then skip	ACS=2's complement ACD=2's complement	Unchanged Unchanged
STA *ac,[@]displ.[,index]* 040000	Fixed-point AC=memory location	AC=unsigned integer memory location = unknown	Unchanged Unsigned integer
STB *acs,acd* 103010	Byte AC(right) =memory location	ACS=byte pointer ACD=byte	Unchanged Unchanged Memory location=byte
SKP *[t] device*	If t is true, then skip		
SNB *acs,acd* 102770	If addressed bit is set to one, then skip.	ACS=word pointer ACD=word offset and bit pointer Memory location=unknown	Unchanged Unchanged Unchanged
SUB *[c][sh][#] acs,acd[,skip]* 102400	Fixed-point acd−ACS=ACD	ACS=unsigned integer ACD=unsigned integer	Unchanged Result
SYC *acs,acd* 103510	Fixed-point 5 words =stack@location 2=pc	ACS=unknown ACD=unknown PC=PC Stack=unknown	Unchanged Unchanged @location 2 Return block
SZB *acs,acd* 102210	If addressed bit is set to zero, then skip.	ACS=word pointer ACD=word offset and bit pointer Memory location=unknown	Unchanged Unchanged Unchanged
SZBO *acs,acd* 102310	If addressed bit is zero, set bit to one and skip.	ACS=word pointer ACD=word offset and bit pointer Memory location=unknown	Unchanged Unchanged 1

Instruction Format	Operation	Before	After
VCT *[@]displacement[,index]*	Fixed-point. See Instruction		
XCH *acs,acd* 100710	Fixed point ACS=ACD ACD=ACS	ACS=unsigned integer ACD=unsigned integer	ACD ACS
XCT *ac* 123370	Fixed point AC=PC	PC=PC AC=Instruction	AC instruction Unchanged
XOP *acs,acd,operation #* 100030	Fixed-point unsigned integer + XOP table address =PC	44=table address Stack=unknown PC=PC AC2=unknown AC3=unknown AC1=unknown AC0=unknown	Unchanged Return block XOP unsigned integer Stack Address of ACS Stack Address of ACD AC1=Unchanged AC0=Unchanged
XOP1 *acs,acd,operation #* 100070	Fixed-point. See XOR		
XOR *acs,acd* 100510	Fixed-point ACS or ACD=ACD	ACS=unsigned integer ACD=unsigned integer	Unchanged Result
XORI *i,ac* 123770	Fixed-point i or AC=AC	AC=unsigned integer I=unsigned integer	Result Unchanged

Appendix B

Instruction Execution Times

The following table lists typical execution time for each S/280 instruction for revision 00 of the central processor microcode and revision 00 of the hardware floating point microcode. The times are typical and subject to change without notice. They may vary for other revisions of the microcode. All times are in microseconds. Notes follow the table.

Instruction		Time (μS)	# of Reads	Notes
ADD, AND, INC, SUB, MOV, COM, NEG, ADC		.15	1	1,2
ADDI, ADI, ANC		.3	1	
ANDI		.45	2	
BAM	N=# of words	1.8+.75N	1+N	3,4
BLM	N=# of words			
	Addresses both even or both odd	2.15+.45(N−2)	1	3,4,5
	Otherwise	2.15+.75N	1 + N	
BTO, BTZ		2.7	2	4,6
CLM	H = high limit			
	ACS=ACD ACS>H	.6	3	
	Otherwise	.75	3	
	ACS<>ACD ACS>H	.9	3	
	Otherwise	1.05	3	
CMP	N= # of bytes compared	4.05+3.3N	1+2N	7,8
CMT	N= # of comparisons	7.5+6.15(N−1)	1+3N	3,9
CMV	N= # of bytes moved	4.35+3.45(N−1)	1+2N	7,8
COB	B= # of bits moved	.45(N+1)		
CTR	N= # of bytes moved or compared			
	Move option	5.55+4.65(N−1)	1+3N	3
	Compare option	6.0+5.1(N−1)	1+5N	3
DAD		1.5	1	
DHXL	N=# of hex digits	2.4+2.4N	1	
DHXR	N=# of hex digits	2.55+2.4N	1	
DIA, DIB, DIC	Device code 10, 11,14, or 43		1	
	S, C, or P present	4.8		
	Otherwise	3.9		
	Other device codes			
	S, C, or P present	4.4		
	Otherwise	3.0		
DOA, DOB, DOC	Device code 10, 11, 14, or 43		1	
	S, C, or P present	3.9		
	Otherwise	1.2		
	Other device codes			
	S, C, or P present	3.9		
	Otherwise	1.2		

Instruction		Time (μS)	# of Reads	Notes
DIV		3.75	1	
DIVS		5.7	1	
DIVX		6.15	1	
DLSH	N=# of bits			
	Left	1.8+.6N	1	
	Right	1.95+.6N	1	
DSB		1.5	1	
DSPA	L = low limit; H = high limit			
	ACD<L	1.2	3	
	ACD>H	1.65	3	
	L<ACD<H	2.25	4	11,6
DSZ		.6	2	4,12,13,14
EDSZ,EISZ		.75	3	4,12,13,14
EJMP,EJSR		.6	4	4,13,14
ELDA		.45	3	4,13,14
ELDB		.9	4	
ELEF		.3	2	4,13,15
ESTA		.45	2	
ESTB		1.95	4	
HLV		.45	1	16
HXL	N=# of hex digits	1.2+.6N	1	
HXR	N=# of hex digits	1.2+1.2N	1	
INTA		5.2	1	
INTDS		1.5	1	
INTEN		2.55	1	
IOR		.3	1	
IORI		.45	2	
IORST		8.1		
ISZ		.6	2	4,12,18
JMP,JSR		.45	3	4,18
LDA		.3	2	4,18
LDB		1.05	2	
LEF		.6	1	
LOB	B=Lead bit position	.45+.3B	1	
LMP	N=# of map table entries to load			
	User map	2.1+1.2N	1+N	6,17
	Data channel map	2.25+1.2N	1+N	6,17
LRB		.9+.45B	1	
LSH	N=# of bits to shift			
	Left	1.05+.15N	1	20
	Right	1.5+.3N	1	
MSKO		5.7	1	
MSP		1.05	3	
MUL		3.0	1	
MULS		3.0	1	
NIOS		3.9	1	
POP	N=# of ACs to POP			
	N=1	.75	3	
	N=2	1.05	4	
	N>1	2.1+.45(N−1)	2+N	

Instruction		Time (μS)	# of Reads	Notes
POPB		3.0	7	
POPJ		1.95	6	
PSH	N=# of ACs to PUSH			
	N=1	1.65	3	
	N=2	2.65	3	
	N>2	2.65+.6(N−1)	3	
PSHJ		1.65	6	
PSHR		1.8	3	
RSTR		4.8	13	21
RTN		3.0	9	21
SAVE		4.0	5	
SBI		.3	1	
SGE		.3	1	12
SGT		.3	1	12
SKP	Device code 10, 11, 14, or 43	5.1	1	12
	Other device code	3.9		
SNB		2.7	2	4,6
STA		.3	1	4,18
STB		1.65	2	
SYC	ACS=ACD=0	1.8	4	
	Otherwise	5.25	5	
SZB		2.7	2	4,6
SZBO	No skip	2.25	2	4,6
	Skip	3.75	4	4,6
VCT	Mode A	4.5	4	
	Mode B	11.3	9	
	Mode C	12.15	10	
	Mode D	14.65	15	
	Mode E	15.15	16	
XCH		.6	1	
XCT		.75 + Instruction Execution Time	2	
XOP		7.5	7	4,6
XOP1		7.2	7	4,6
XOR		.3	1	
XORI		.45	2	

NOTES

	Explanation	Reads
1	Add .3 for skip but no swap.	1
2	Add .3 for swap and no skip or .6 for both swap and skip.	0 1
3	Add .45 for first indirection on an AC, 1.05 for the second, .6 for each additional.	1 1 1 each
4	Add 2.4 if instruction enables user address translations	0
5	Add .75 if starting address is odd and .6 if ending address is odd.	1 1
6	Add .3 for first defer, 1.05 for the second, .6 for each additional.	1 1 1 each
7	Add .3 if AC0<0, .3 if AC1<0.	0
8	Subtract .3 for each byte of an exhausted string.	−1 each
9	Add .3 if AC1<0.	0
10	Add .15 for digit overflow.	0
11	Add .6 additional if there is indirection on result, also add times on note 6 for indirection.	0 see note 6
12	Add .3 for skip.	1
13	Add .15 for PC-relative addressing.	0
14	Add .45 for the first defer (.3 if PC relative), 1.05 for the second defer, .6 for each additional.	1 1 1 each
15	Add .6 for the first defer (.3 if PC relative), 1.05 for the second defer, .6 for each additional.	1 1 1 each
16	Add .15 if initial value is >0.	0
17	Add .3 for each validity protected page.	0
18	Add .15 on first defer, 1.05 for second, .6 for each additional.	1 1 1 each
19	See note 15, except PC relative is also .6.	see note 15
20	Add .3 if N is odd.	0
21	Add 2.55 if instruction enables user address translations	0

Appendix C

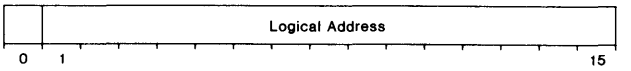
Register Fields

This appendix contains the formats for the following user or program-accessible registers available on the ECLIPSE S/280 computer.

Register	Contents	Accessible by
Program counter	Logical address of the currently executing instruction	Program
Processor status	Information about the current state of the system	Virtual console
Floating-point status	Information about floating-point computations	Program
User/DCH address translator status	Information about user and data channel address translation	Program and virtual console
Memory fault code and memory fault address	Information about memory data errors	Program
BMC status	Information about BMC address translation and the BMC facility	Program
Power system status	Information about the power system	Program

Program Counter

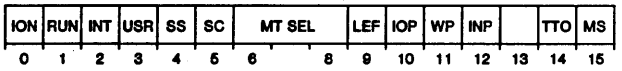
The format of the program counter follows.



Bits	Name	Contents or Function
0	—	Reserved for future use.
1-15	Logical Address	15-bit logical address of the currently executing instruction.

Processor Status Register

The processor status register occupies reserved memory location 6. Its format follows.

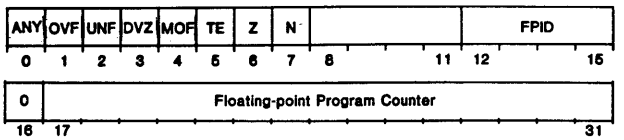


Bits	Name	Contents or Function
0	ION	If 1, the Interrupt On (ION) flag is set to 1.
1	RUN	The central processor is in run mode, not virtual console mode.
2	INT	If 1, the interrupt system is on. This bit is set to 1 immediately following the execution of the first instruction after the Interrupt On (ION) flag is set to one.
3	USR	If 1, user address translation is enabled.
4	SS	If 1, the virtual console is in single step mode.
5	SC	If 1, a translate user single cycle operation is pending.
6-8	MT SEL	The map table currently selected for use.

Bits 6	7	8	Currently Selected Map Table
0	0	0	User A
0	0	1	User C
0	1	0	User B
0	1	1	User D
1	0	0	Unmapped map table (used when address translation is disabled)

9	LEF	If 1, LEF instruction mode is enabled for the current map table (bits 6-8).
10	IOP	If 1, I/O protection is enabled for the current map table (bits 6-8).
11	WP	If 1, write protection is enabled for the current map table (bits 6-8).
12	INP	If 1, indirection protection is enabled for the current map table (bits 6-8).
13	—	Reserved for future use.
14	TTO	If 1, the asynchronous output line's Busy or Done flag is set to 1.
15	MS	If 1, a load user/DCH address translator status register operation is pending. (That is, a *Load User/DCH Address Translator Status* instruction (DOA MAP) that enables user address translation (with bit 15 = 1) was issued but an indirect reference or return-type instruction has not been issued yet.

Floating-point Status Register

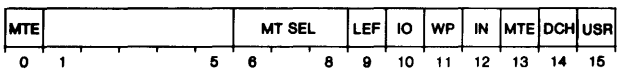


Bits	Name	Contents or Function
0	ANY	If 1, one or more of bits 1-4 are set to 1.
1	OVF	If 1, exponent overflow occurred. The result is correct except that the exponent is 128 too small.
2	UNF	If 1, exponent underflow occurred. The result is correct except that the exponent is 128 too large.
3	DVZ	If 1, division by zero was attempted. The division operation was aborted and the operands remain unchanged.
4	MOF	If 1, a mantissa overflow occurred.
5	TE	If 1, floating-point traps are enabled. Setting any of bits 1-4 to 1 will cause a floating-point fault.
6	Z	If 1, the result is zero.
7	N	If 1, the result is negative.
8-11	—	Reserved for future use.
12-15	FPID	Floating-point model number. Should be 13_8 for firmware floating-point and 5_8 for hardware floating-point.
16	—	Reserved for future use.
17-31	Floating-point program counter	Floating-point program counter. In the event of a floating-point fault, this is the address of the floating-point instruction that caused the fault.

User/DCH Address Translator Status Register

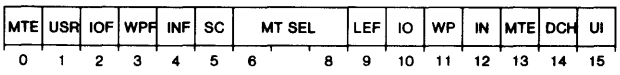
The meaning of the contents of this register varies depending on how the program accesses it. If the program accesses it with a *Load User*/DCH *Translator Status* instruction (DOA MAP), the contents define the state of the address translator after the next memory reference instruction. On the other hand, if the program accesses it with a *Read User*/DCH *Translator Status* instruction (DIA MAP), the contents describe the current state of the address translator. The format of the register for each case follow.

Format using DOA MAP Instruction



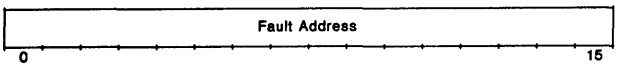
Bits	Name	Contents or Function
0, 13	MTE	Enables the map table for the next user process. **Bits 0 13 — Map Table Enabled** 0 0 — User A 0 1 — User B 1 0 — User C 1 1 — User D
1-5	—	Reserved for future use.
6-8	MT SEL	Selects the page table to be loaded by the next *Load Map Table* instruction (LMP). **Bits 6 7 8 — Map Table Selected** 0 0 0 — User A 0 0 1 — User C 0 1 0 — User B 0 1 1 — User D 1 0 0 — Data channel A 1 0 1 — Data channel C 1 1 0 — Data channel B 1 1 1 — Data channel D
9	LEF	If 1 and user address translation is enabled for the next user (bit 15 = 1), then all I/O format instructions will be interpreted as *Load Effective Address* instructions (LEF) for the next user. If 0, all I/O format instructions, including LEF instructions, will be interpreted as I/O instructions for the next user.
10	IO	If 1, I/O protection will be enabled for the next user.
11	WP	If 1, write protection will be enabled for the next user.
12	IN	If 1, indirect protection will be enabled for the next user.
14	DCH	If 1, data channel address translation will be enabled immediately after execution of this instruction.
15	USR	If 1, user address translation will be enabled with the first memory reference after the next indirect reference.

Format using DIA MAP Instruction

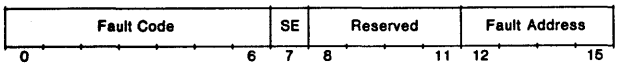


Bits	Name	Contents or Function
0, 13	MTE	User map table currently enabled. **Bits 0 13 — Map Table Enabled** 0 0 — User A 0 1 — User B 1 0 — User C 1 1 — User D
1	USR	If 1, user address translation (mapped mode) is enabled.
2	IOF	If 1, the last protection fault was an I/O protection fault.
3	WPF	If 1, the last protection fault was a write protection fault.
4	IDF	If 1, the last protection fault was an indirect protection fault.
5	SC	If 1, the last protection fault occurred during a *Translate Single Cycle* instruction (NIOP MAP).
6-8	MT SEL	User map table loaded by the last *Load Map Table* instruction (LMP). **Bits 6 7 8 — Map Table Selected** 0 0 0 — User A 0 0 1 — User C 0 1 0 — User B 0 1 1 — User D 1 0 0 — Data channel A 1 0 1 — Data channel C 1 1 0 — Data channel B 1 1 1 — Data channel D
9	LEF	If 1, LEF instruction mode was enabled for the last user.
10	IO	If 1, I/O protection was enabled for the last user.
11	WP	If 1, write protection was enabled for the last user.
12	IN	If 1, indirect protection was enabled for the last user.
14	DCH	If 1, data channel address translation has been enabled.
15	UI	If 1, the last interrupt occurred while user address translation was enabled.

Memory Fault Address and Fault Code Registers



Bits	Name	Contents or Function
0-15	Fault Address least significant bits	Sixteen least significant physical address bits of the double-word or quad-word with the faulty data.



NOTE: *The fault address and fault code (syndrome) is only valid while the Done flag is set to one. The fault address is the address of the first word in faulty double word, except during cache fill operations, when it is the address of the first word in the faulty quad-word block.*

Bits	Name	Contents or Function
0-6	Fault Code	Fault code (syndrome) indicating no bit, 1 bit, 2 bit, or more than 2 bit errors as listed in Table C.1.
7	SE	Error was detected during sniffing.
8-13	Reserved	Reserved for future use.
12-15	Fault Address most significant bits	Four most significant physical address bits of the double word or quad word with the faulty data.

				6	0	1	0	1	0	1	0	1
Bit				5	0	0	1	1	0	0	1	1
0	1	2	3	4	0	0	0	0	1	1	1	1
0	0	0	0		X	38	37	T	36	T	T	30
0	0	0	1		35	T	T	27	T	5	M	T
0	0	1	0		34	T	T	25	T	3	15	T
0	0	1	1		T	M	13	T	23	T	T	M
0	1	0	0		33	T	T	24	T	2	M	T
0	1	0	1		T	1	12	T	22	T	T	M
0	1	1	0		T	M	10	T	20	T	T	M
0	1	1	1		16	T	T	M	T	M	M	T
1	0	0	0		32	T	T	M	T	M	14	T
1	0	0	1		T	M	11	T	21	T	T	M
1	0	1	0		T	M	9	T	19	T	T	31
1	0	1	1		M	T	T	29	T	7	M	T
1	1	0	0		T	M	8	T	18	T	T	M
1	1	0	1		17	T	T	28	T	6	M	T
1	1	1	0		M	T	T	26	T	4	M	T
1	1	1	1		T	0	M	T	M	T	T	M

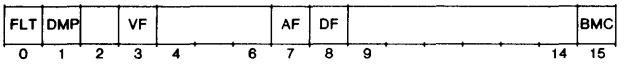
Table C.1 Memory fault code interpretation

ERROR KEY:

Number	Bit in error
T	Two bits in error
M	More than two bits in error
X	No errors

NOTE: *Bits 32-38 are the check bits.*

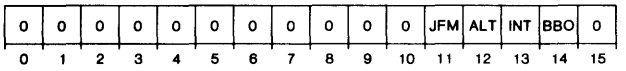
BMC Status Register



Bits	Name	Contents or Function
0	FLT	If 1, a validity protection fault, address parity fault, or data parity fault occurred.
1	DMP	If 1, the next map table transfer operation will be a load. If 0, the next map table transfer operation will be a dump.
2	—	Reserved for future use.
3	VF	If 1, a validity protection fault occurred.
4-6	—	Reserved for future use.
7	AF	If 1, an address parity fault occurred.
8	DF	If 1, a data parity fault occurred.
9-14	—	Reserved for future use.
15	BMC	If 1, the BMC facility is present in the system.

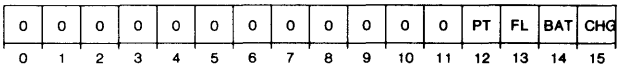
Power System Status Registers

Control Status



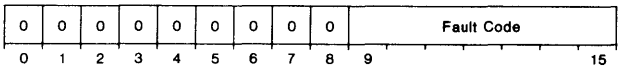
Bits	Name	Contents or Function
0-10	—	Reserved for future use.
11	JFM	If 1, the UPSC is jumpered for voltage margining which may degrade the operation of the system.
12	ALT	If 1, alternate powerfail mode is enabled, disabling out powerfail interrupts from device code 0. As a result, powerfail skip instructions (SKPDN and SKPDZ) always function as if there were no powerfail.
13	INT	If 1, fault interrupts enabled.
14	BBD	If 1, battery backup disabled.
15	—	Used for diagnostic testing. 0 for normal operation.

Battery Backup Status



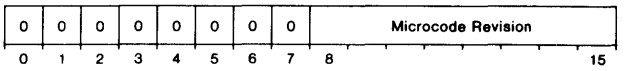
Bits	Name	Contents or Function
0-11	—	Reserved for future use.
12	PT	If 1, partial battery backup is present.
13	FL	If 1, full battery backup is present.
14	BAT	If 1, the system is running on battery power.
15	CHG	If 1, the batteries are recharging.

Fault Code



Bits	Name	Contents or Function
0-8	—	Reserved for future use.
9-15	Fault Code	Octal code identifying most recent fault. Bits 9-12 are the octal number of the fault in the octal fault type specified by bits 13-15. The UPSC flashes the octal fault type on the three front console lights. Table C.2 lists the faults by type.

UPSC Microcode Revision



Bits	Name	Contents
0-7	—	Reserved for future use.
8-15	Microcode Revision	Current revision of the UPSC microcode. O indicates the first release of of the microcode. Successive numbers indicate successive modifications to the microcode.

Fault Type	Code (octal)	Meaning
0	000	System off or no fault
1		Environmental fault
	011	VNR[6] undervoltage
	021	VNR[6] overvoltage
	031	Power supply over temperature
	041	Chassis over temperature
2		Fan failure in computer chassis
	002	Blower or multiple fan failure
	012	Fan 1 failure
	022	Fan 2 failure
	032	Fan 3 failure
	042	Fan 4 failure
	052	Fan 5 failure
	062	Fan 6 failure
	072	UPSC cannot set fan signals
3		VNR[6] fault
	013	Battery backup fault
4		Power supply fault (includes undervoltages)
	004	Undervoltage on +5V
	014	Current not sharing on +5V
	044	Undervoltage on +5MEM, PS1
	054	Undervoltage on +5MEM, PS2
	064	Undervoltage on +5MEM, PS3
	074	Undervoltage on +12MEM or +12V, PS1
	104	Undervoltage on +12MEM or +12V, PS2
	114	Undervoltage on +12MEM or +12V, PS3
	124	Undervoltage on −5VMEM or −5V, PS1
	134	Undervoltage on −5VMEM or −5V, PS2
	144	Undervoltage on −5VMEM or −5V, PS3
	154	Undervoltage on unknown voltage, PS1
	161	Undervoltage on unknown voltage, PS2
	174	Undervoltage on unknown voltage, PS3

Table C.2 Power system fault codes

[6]*Voltage nonregulated unit*

Fault Type	Code (octal)	Meaning
5		Overvoltage fault
	005	Overvoltage on +5V
	045	Overvoltage on +5MEM, PS1
	055	Overvoltage on +5MEM, PS2
	065	Overvoltage on +5MEM, PS3
	075	Overvoltage on +12MEM or +12V, PS1
	105	Overvoltage on +12MEM or +12V, PS2
	115	Overvoltage on +12MEM or +12V, PS3
	125	Overvoltage on −5MEM or −5V, PS1
	135	Overvoltage on −5MEM or −5V, PS2
	145	Overvoltage on −5MEM or −5V, PS3
	155	Overvoltage on unknown voltage, PS1
	165	Overvoltage on unknown voltage, PS2
	175	Overvoltage on unknown voltage, PS3
6		Over-current fault
	006	Reed switch sense low on +5V output
	016	Overcurrent on +5V, PS1
	026	Overcurrent on +5V, PS2
	036	Overcurrent on +5V, PS3
	046	Overcurrent on +5MEM, PS1
	156	Overcurrent on +5MEM, PS2
	166	Overcurrent on +5MEM, PS3
	167	Overcurrent on +12MEM or +12V PS1
	106	Overcurrent on +12MEM or +12V PS2
	116	Overcurrent on +12MEM or +12V PS3
	126	Overcurrent on −5MEM or −5V PS1
	136	Overcurrent on −5MEM or −5V PS2
	146	Overcurrent on −5MEM or −5V PS3
	156	Overcurrent on unknown voltage, PS1
	166	Overcurrent on unknown voltage, PS2
	167	Overcurrent on unknown voltage, PS3
7		UPSC fault
	007	Checksum error on UPSC ROM
	177	LED lamp test at power up

Table C.2 Power system fault codes (continued)

Appendix D

Standard ECLIPSE S/280 I/O Device Codes

Octal Device Codes	Mnem	Priority Mask Bit	Device Name
00	—	—	Unused
01	ABL	—	Automatic Boot Load (ABL) register
02	ERCC	—	Error checking and correction
03	MAP	—	User/DCH address translator
04	UPSC	13	Universal power supply controller
05	BMC	—	Burst multiplexor channel
06	MCAT	12	Multiprocessor adapter transmitter
07	MCAR	12	Multiprocessor adapter receiver
10	TTI	14	TTY input
11	TTO	15	TTY output
12			
13			
14	RTC	13	Real-time clock
15	PLT	12	Incremental plotter
16	CDR	10	Card reader
17	LPT	12	Line printer
20	DSK	9	Fixed-head disk
21			
22	MTA	10	Magnetic tape
23			
24			
25			
26	DKB	9	Fixed-head DG/Disk
27	DPF	7	DG/Disc storage subsystem
30			
31			
32			
33	DKP	7	Moving head disk
34[1]	DCU[2]	4	Data control unit
	MX1	11	Multiline asynchronous controller
35	MX2	11	Multiline asynchronous controller
36	IPB	6	Interprocessor bus-half duplex
37	IVT	6	IPB watch dog timer
40	DPI	8	IPB full-duplex input
41	DPO	8	IPB full-duplex output
	—	8	Digital I/O
42	DIO	7	Digital I/O timer
43	DIOT	6	Programmable Interval Timer
	PIT	6	
44			
45			
46	MCAT1	12	Second multiprocessor transmitter
47	MCAR1	12	Second multiprocessor receiver
50	TTI1	14	Second TTY input
51	TTO1	15	Second TTY output
52			
53			
54	RTC1	13	Second real-time clock
55	PLT1	12	Second incremental plotter
56	CDR1	10	Second card reader
57	LPT1	12	Second line printer
60	DSK	9	Second fixed-head disk
61			
62	MTA1	10	Second magnetic tape
63			
64			
65			
66	DKB1	9	Second fixed-head DG/Disc
67	DPF1	7	Second DG/Disc storage subystem
70			
71			
72			
73	DKP1	7	Second moving head disc
74			
75			
76	DPU	4	DCU To Host interface
77	CPU	—	CPU and console functions

[1] *Code returned by INTA and used by VCT.*

[2] *Can be set to any unused device code between 1 and 76.*

Appendix E

Compatibility with Earlier ECLIPSE Computers

The ECLIPSE S/280 series computers are program compatible with the earlier 16-Bit Real-Time ECLIPSE line of computers. Except for the few differences listed below, a program that runs on one of these computer also runs on an ECLIPSE S/280 computer with the same size memory and same peripheral devices.

> **NOTE:** *In microECLIPSE computers, such as the S/120, implementation of a few functions differs slightly from the rest of the 16-bit Real-Time ECLIPSE line. For differences that might affect running a microECLIPSE program on an S/280 computer, refer to the assembly language programming manual for the microECLIPSE computer.*

Unique Features

The cache memory in S/280 computers may confuse memory sizing routines. A memory sizing routine can only determine if a memory location exists by writing and then reading to a previously *unaccessed* memory location. Any future access to this location may indicate that the location *does exist* even when it does not, since the first read placed the contents of the location in the cache. To protect against this, the appropriate cache block should be flushed before any sizing attempt. If the location to be sized for has address X, then reading any two locations with addresses having the same ten least significant bits as X but different physical page numbers (2000+X or X-10000, for instance) ensures that X is not in the cache. The routine can then size for X by writing and then reading X. Chapter 1 explains cache operation.

The S/280 data channel facility supports down-line loading of data channel map tables (map slots) from I/O devices and indivisible read-modify-write operations. Refer to the *Interface Designer's Manual for* NOVA *and* ECLIPSE *Line Computers* (DG No. 014-000629) for information on these features.

S/280 computers use a power supply with a universal power supply controller (UPSC) that allows software to control several power supply functions. The UPSC interfaces to the ECLIPSE I/O bus as device code 4. Chapter 5 contains information on UPSC programming.

Execution Timing

Routines that depend on processor timing to determine real-time delays produce different results on S/280 computers than on previous ECLIPSE line computers. All instruction times and I/O times differ. Refer to Appendix B for instruction times.

Program Flow

Like previous ECLIPSE processors, the contents of the program counter that the S/280 processor pushes on the stack after a validity or write protection fault may bear no resemblance to the address of the instruction that caused the fault. However, the value pushed by the S/280 processor will differ from that pushed by other processors.

In certain obscure cases, the S/280 processor will not detect a write operation to a location already stored in its instruction pipeline, causing unpredictable results. These cases only occurs if address translation (mapping) is enabled when the processor writes to an address

- whose least significant 10 bits are one or two greater than the least significant 10 bits of the address currently in the program counter,
- on a different logical page than the address in the program counter,
- on a logical page that translates (maps) to the same physical page as the address in the program counter.

Memory

The automatic increment and automatic decrement locations — locations 20_8 to 27_8 and 30_8 to 37_8 — are not available for this purpose on S/280 computers.

Address Translation

In an S/280 computer, once the user address translation (mapping) is enabled by setting bit 15 to 1 in the user/DCH address translator (MAP) status register, *any* indirect reference enables user address translation (mapping). This is similar to earlier ECLIPSE computers except that most of them do not enable address translation when the indirect reference occurs during certain instructions, such as BAM, BLM, DSPA, floating-point, or character.

An attempt to read a page with validity protection does not change the accumulator and carry values except in the cases listed below.

- During a *Return* (RTN) instruction, the carry is set to 0 if a protection fault occurs when popping (reading) the word containing the carry off the stack.
- During a *Return* (RTN) instruction or a *Pop Block* (POPB) instruction, if the stack location storing AC0 has validity protection but the location storing the program counter does not, then AC0 contains 0 after the processor handles the protection fault.

In an S/280 computer, bit 5 of the user/DCH address translator (MAP) status register when read with a *Read User/DCH Translator Status* instruction (DIA MAP) indicates whether or not the last memory protection fault occurred during a single-cycle operation. In most previous ECLIPSE computers, this bit indicates whether or not the last memory reference was a single-cycle operation.

Single cycle references in an S/280 computer are only triggered by a *Load Accumulator* instruction (LDA) or a *Store Accumulator* instruction (STA). In previous ECLIPSE computers, single-cycle references are also triggered by an *Extended Load Accumulator* instruction (ELDA) or an *Extended Store Accumulator* instruction (ESTA).

The S/280 user/DCH address translator (MAP) does not provide write protection for memory accessed by data channel processes. It only provides validity protection. When the *Load User/DCH Map Table* instruction (LMP) loads a data channel map table, the write protect bit (bit 0) of each map table entry (map slot) is ignored, unless the page is also being declared invalid (validity protected). Chapter 6 provides more information on validity protection.

The burst multiplexor channel (BMC) address translator has no Busy or Done flags so an interrupt handler program cannot test the Busy flag to determine when a BMC map table transfer operation (load or dump) is completed. A BMC instruction with a Start command to initiate a map table transfer (load or dump) is interruptible and resumable with updated values.

Error Checking and Correction

The instructions for programming the S/280 error checking and correction (ERCC) facility differ because the S/280 uses memory modules organized around double words (32 data bits) instead of single words (16 data bits). The S/280 ERCC facility uses seven check bits for each double word instead of the five check bits for each single word used by the ERCC facilities in the earlier ECLIPSE computers. The ERCC Done flag must be enabled by a *Enable ERCC* instruction (DOA ERCC) before it can be set to 1. The memory fault address and fault code returned by the *Read Memory Fault Address* (DIA ERCC) and the *Read Memory Fault Code and Address* (DIB ERCC) instructions are only valid when the Done flag is set to 1. Chapter 6 provides information on the S/280 ERCC instructions.

Diagnostic and Special Instructions

The following opcodes listed in Table E.1 are used as diagnostic or other special instructions. These instructions are for use only by Data General diagnostic or special systems software since they alter normal system parameters and may degrade system performance.

Opcode	Function
062002	Write check bits
062402	Read 39 bits
063002	Write double word
103410	Test memory
117410	Size memory
123410	Test memory
153410	Load special user/DCH map table

Table E.1 Opcodes for diagnostic and other special instructions

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.*

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

DG OFFICES

NORTH AMERICAN OFFICES

Alabama: Birmingham
Arizona: Phoenix, Tucson
Arkansas: Little Rock
California: Anaheim, El Segundo, Fresno, Los Angeles, Oakland, Palo Alto, Riverside, Sacramento, San Diego, San Francisco, Santa Barbara, Sunnyvale, Van Nuys
Colorado: Colorado Springs, Denver
Connecticut: North Branford, Norwalk
Florida: Ft. Lauderdale, Orlando, Tampa
Georgia: Norcross
Idaho: Boise
Iowa: Bettendorf, Des Moines
Illinois: Arlington Heights, Champaign, Chicago, Peoria, Rockford
Indiana: Indianapolis
Kentucky: Louisville
Louisiana: Baton Rouge, Metairie
Maine: Portland, Westbrook
Maryland: Baltimore
Massachusetts: Cambridge, Framingham, Southboro, Waltham, Wellesley, Westboro, West Springfield, Worcester
Michigan: Grand Rapids, Southfield
Minnesota: Richfield
Missouri: Creve Coeur, Kansas City
Mississippi: Jackson
Montana: Billings
Nebraska: Omaha
Nevada: Reno
New Hampshire: Bedford, Portsmouth
New Jersey: Cherry Hill, Somerset, Wayne
New Mexico: Albuquerque
New York: Buffalo, Lake Success, Latham, Liverpool, Melville, New York City, Rochester, White Plains
North Carolina: Charlotte, Greensboro, Greenville, Raleigh, Research Triangle Park
Ohio: Brooklyn Heights, Cincinnati, Columbus, Dayton
Oklahoma: Oklahoma City, Tulsa
Oregon: Lake Oswego
Pennsylvania: Blue Bell, Lancaster, Philadelphia, Pittsburgh
Rhode Island: Providence
South Carolina: Columbia
Tennessee: Knoxville, Memphis, Nashville
Texas: Austin, Dallas, El Paso, Ft. Worth, Houston, San Antonio
Utah: Salt Lake City
Virginia: McLean, Norfolk, Richmond, Salem
Washington: Bellevue, Richland, Spokane
West Virginia: Charleston
Wisconsin: Brookfield, Grand Chute, Madison

DG-04976

INTERNATIONAL OFFICES

Argentina: Buenos Aires
Australia: Adelaide, Brisbane, Hobart, Melbourne, Newcastle, Perth, Sydney
Austria: Vienna
Belgium: Brussels
Bolivia: La Paz
Brazil: Sao Paulo
Canada: Calgary, Edmonton, Montreal, Ottawa, Quebec, Toronto, Vancouver, Winnipeg
Chile: Santiago
Columbia: Bogata
Costa Rica: San Jose
Denmark: Copenhagen
Ecuador: Quito
Egypt: Cairo
Finland: Helsinki
France: Le Plessis-Robinson, Lille, Lyon, Nantes, Paris, Saint Denis, Strasbourg
Guatemala: Guatemala City
Hong Kong
India: Bombay
Indonesia: Jakarta, Pusat
Ireland: Dublin
Israel: Tel Aviv
Italy: Bologna, Florence, Milan, Padua, Rome, Tourin
Japan: Fukuoka, Hiroshima, Nagoya, Osaka, Tokyo, Tsukuba
Jordan: Amman
Korea: Seoul
Kuwait: Kuwait
Lebanon: Beirut
Malaysia: Kuala Lumpur
Mexico: Mexico City, Monterrey
Morocco: Casablanca
The Netherlands: Amsterdam, Rijswijk
New Zealand: Auckland, Wellington
Nicaragua: Managua
Nigeria: Ibadan, Lagos
Norway: Oslo
Paraguay: Asuncion
Peru: Lima
Philippine Islands: Manila
Portugal: Lisbon
Puerto Rico: Hato Rey
Saudi Arabia: Jeddah, Riyadh
Singapore
South Africa: Cape Town, Durban, Johannesburg, Pretoria
Spain: Barcelona, Bibao, Madrid
Sweden: Gothenburg, Malmo, Stockholm
Switzerland: Lausanne, Zurich
Taiwan: Taipei
Thailand: Bangkok
Turkey: Ankara
United Kingdom: Birmingham, Bristol, Glasgow, Hounslow, London, Manchester
Uruguay: Montevideo
USSR: Espoo
Venezuela: Maracaibo
West Germany: Dusseldorf, Frankfurt, Hamburg, Hannover, Munich, Nuremburg, Stuttgart

DataGeneral

Ordering Technical Publications

TIPS is the Technical Information and Publications Service—a new support system for DGC customers that makes ordering technical manuals simple and fast. Simple, because TIPS is a central supplier of literature about DGC products. And fast, because TIPS specializes in handling publications.

TIPS was designed by DG's Educational Services people to follow through on your order as soon as it's received. To offer discounts on bulk orders. To let you choose the method of shipment you prefer. And to deliver within a schedule you can live with.

How to Get in Touch with TIPS

Contact your local DGC education center for brochures, prices, and order forms. Or get in touch with a TIPS administrator directly by calling (617) 366-8911, extension 4086, or writing to

Data General Corporation
Attn: Educational Services, TIPS Administrator
MS F019
4400 Computer Drive
Westborough, MA 01580

TIPS. For the technical manuals you need, when you need them.

DGC Education Centers

Boston Education Center
Route 9
Southboro, Massachusetts 01772
(617) 485-7270

Washington, D.C. Education Center
7927 Jones Branch Drive, Suite 200
McLean, Virginia 22102
(703) 827-9666

Atlanta Education Center
6855 Jimmy Carter Boulevard, Suite 1790
Norcross, Georgia 30071
(404) 448-9224

Los Angeles Education Center
5250 West Century Boulevard
Los Angeles, California 90045
(213) 670-4011

Chicago Education Center
703 West Algonquin Road
Arlington Heights, Illinois 60005
(312) 364-3045

Technical Products Publications
Comment Form

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

Title: _____

Document No. 014-000689-00

Yes	No			
☐	☐	Is this manual easy to read?	○ You (can, cannot) find things easily. ○ Language (is, is not) appropriate. ○ Technical terms (are, are not) defined as needed.	○ Other:
		In what ways do you find this manual useful?	○ Learning to use the equipment ○ As a reference ○ As an introduction to the product	○ To instruct a class. ○ Other:
☐	☐	Do the illustrations help you?	○ Visuals (are,are not) well designed. ○ Labels and captions (are,are not) clear. ○ Other:	
☐	☐	Does the manual tell you all you need to know? What additional information would you like?		
☐	☐	Is the information accurate? (If not please specify with page number and paragraph.)		

Name: _____ Title: _____

Company: _____ Division: _____

Address: _____ City: _____

State: _____ Zip: _____ Telephone: _____ Date: _____

DG-06895

DataGeneral
Data General Corporation, Westboro, Massachusetts 01580

CUT ALONG DOTTED LINE

FOLD FOLD

TAPE TAPE

FOLD FOLD

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: Technical Products Publications (F-131)
4400 Computer Drive
Westboro, MA 01581

DataGeneral

users group

Installation Membership Form

Name _____ Position _____ Date _____

Company, Organization or School _____

Address _____ City _____ State _____ Zip _____

Telephone: Area Code _____ No. _____ Ext. _____

1. Account Category

- ☐ OEM
- ☐ End User
- ☐ System House
- ☐ Government

2. Hardware

	Qty. Installed	Qty. On Order
M/600		
MV/Series ECLIPSE®		
Commercial ECLIPSE		
Scientific ECLIPSE		
Array Processors		
CS Series		
NOVA®4 Family		
Other NOVAs		
microNOVA® Family		
MPT Family		

Other _____
(Specify) _____

3. Software

- ☐ AOS
- ☐ AOS/VS
- ☐ AOS/RT32
- ☐ MP/OS
- ☐ MP/AOS
- ☐ RDOS
- ☐ DOS
- ☐ RTOS
- ☐ Other

Specify _____

4. Languages

- ☐ ALGOL
- ☐ DG/L
- ☐ COBOL
- ☐ Interactive COBOL
- ☐ PASCAL
- ☐ Business BASIC
- ☐ BASIC
- ☐ Assembler
- ☐ FORTRAN 77
- ☐ FORTRAN 5
- ☐ RPG II
- ☐ PL/1
- ☐ APL
- ☐ Other

Specify _____

5. Mode of Operation

- ☐ Batch (Central)
- ☐ Batch (Via RJE)
- ☐ On-Line Interactive

6. Communication

- ☐ HASP
- ☐ HASP II
- ☐ RJE80
- ☐ RCX 70
- ☐ RSTCP
- ☐ 4025
- ☐ X.25
- ☐ SAM
- ☐ CAM
- ☐ XODIAC™
- ☐ DG/SNA
- ☐ 3270
- ☐ Other

Specify _____

7. Application Description

○ _____

8. Purchase

From whom was your machine(s) purchased?

- ☐ **Data General Corp.**
- ☐ Other

Specify _____

9. Users Group

Are you interested in joining a special interest or regional Data General Users Group?

○ _____

CUT ALONG DOTTED LINE

DataGeneral

Data General Corporation, Westboro, Massachusetts 01580, (617) 366-8911

FOLD FOLD

TAPE TAPE

FOLD FOLD

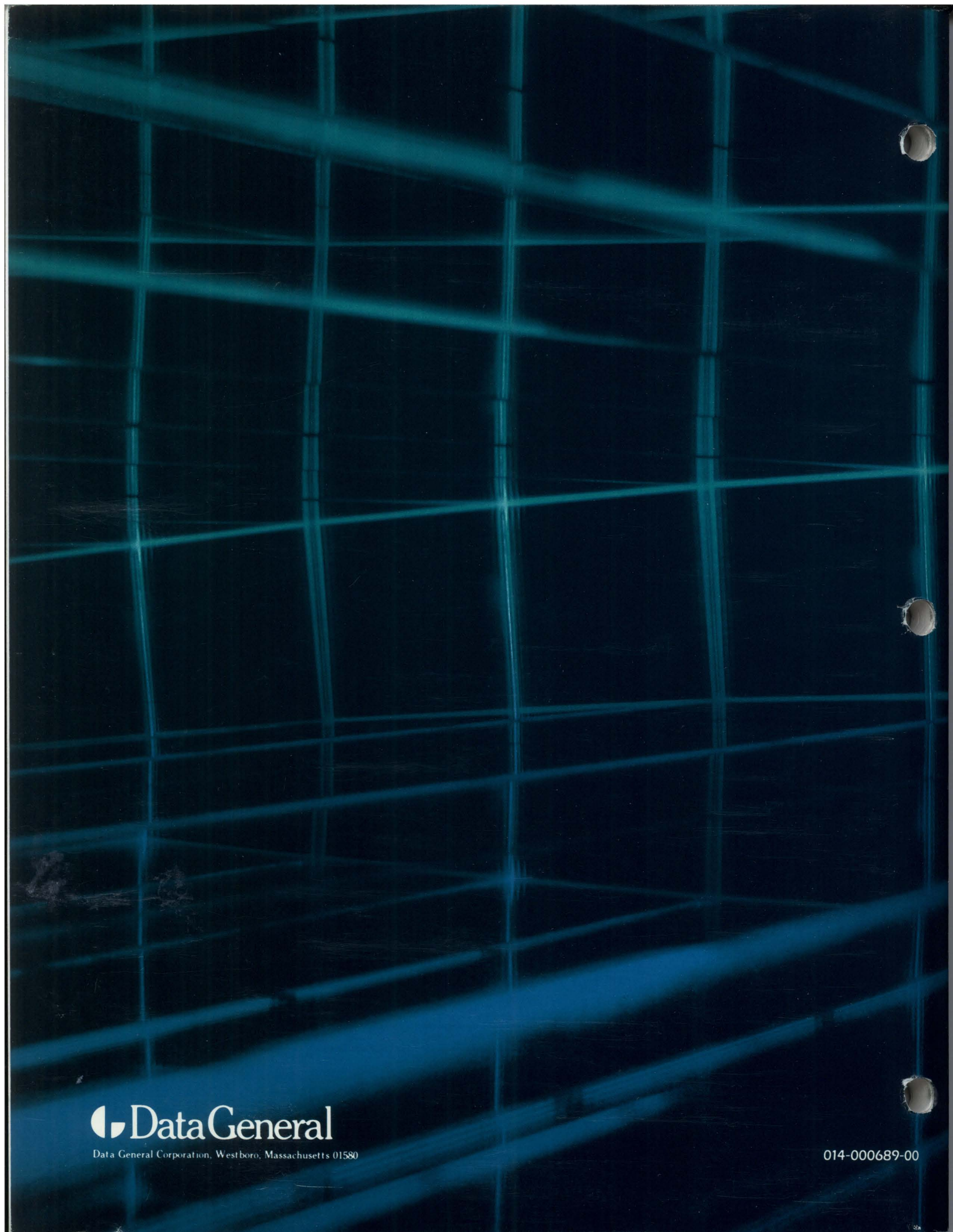
NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 26 SOUTHBORO, MA. 01772

Postage will be paid by addressee:

Data General

ATTN: Users Group Coordinator (C-228)
4400 Computer Drive
Westboro, MA 01581



Data General

Data General Corporation, Westboro, Massachusetts 01580

014-000689-00