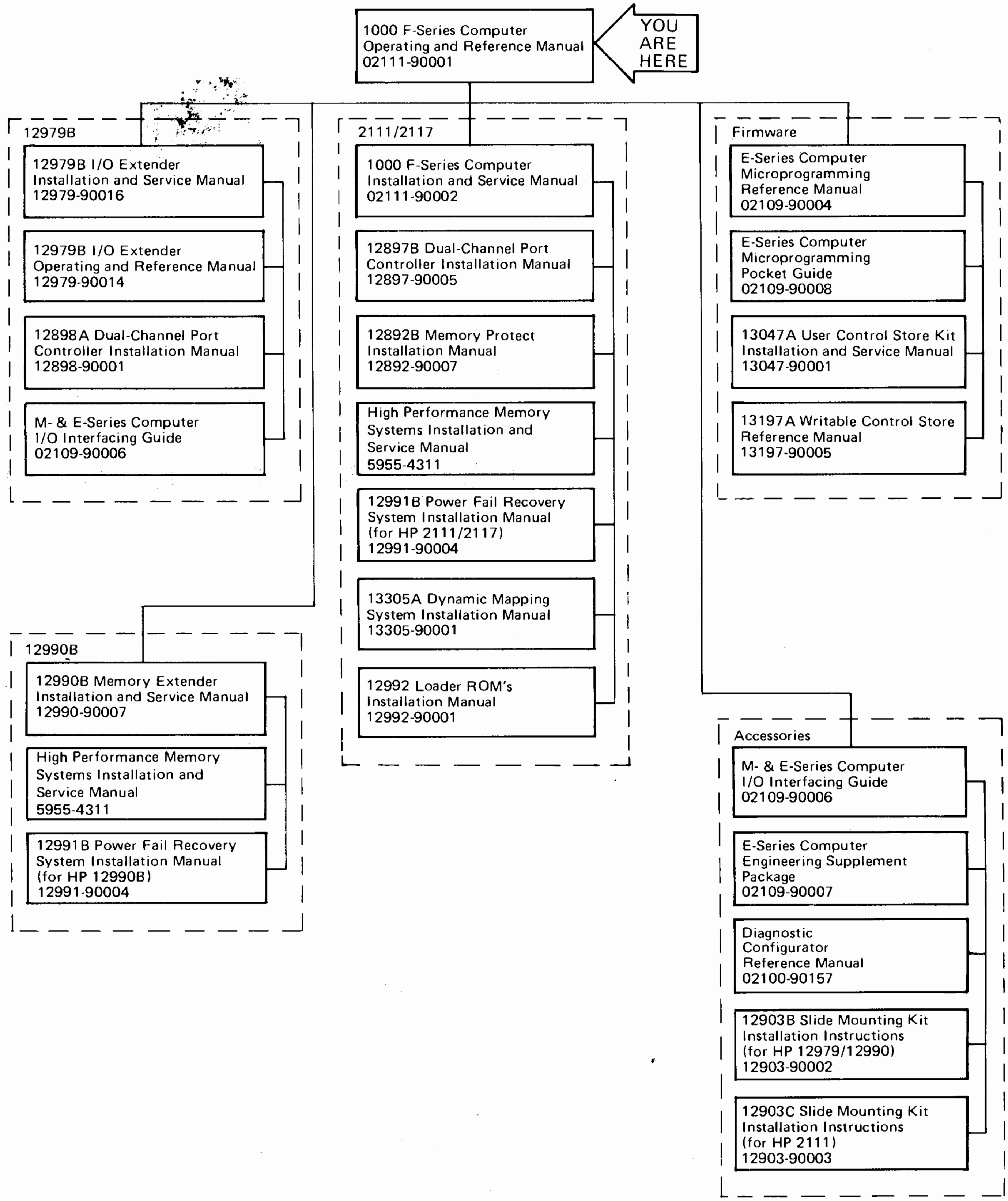
HEWLETT hp PACKARD

HP 1000 F-Series Computer
Operating and Reference Manual

DOCUMENTATION MAP



1000 F-Series Computer
Operating and Reference Manual
02111-90001

YOU
ARE
HERE

12979B

- 12979B I/O Extender Installation and Service Manual 12979-90016
- 12979B I/O Extender Operating and Reference Manual 12979-90014
- 12898A Dual-Channel Port Controller Installation Manual 12898-90001
- M- & E-Series Computer I/O Interfacing Guide 02109-90006

2111/2117

- 1000 F-Series Computer Installation and Service Manual 02111-90002
- 12897B Dual-Channel Port Controller Installation Manual 12897-90005
- 12892B Memory Protect Installation Manual 12892-90007
- High Performance Memory Systems Installation and Service Manual 5955-4311
- 12991B Power Fail Recovery System Installation Manual (for HP 2111/2117) 12991-90004
- 13305A Dynamic Mapping System Installation Manual 13305-90001
- 12992 Loader ROM's Installation Manual 12992-90001

Firmware

- E-Series Computer Microprogramming Reference Manual 02109-90004
- E-Series Computer Microprogramming Pocket Guide 02109-90008
- 13047A User Control Store Kit Installation and Service Manual 13047-90001
- 13197A Writable Control Store Reference Manual 13197-90005

12990B

- 12990B Memory Extender Installation and Service Manual 12990-90007
- High Performance Memory Systems Installation and Service Manual 5955-4311
- 12991B Power Fail Recovery System Installation Manual (for HP 12990B) 12991-90004

Accessories

- M- & E-Series Computer I/O Interfacing Guide 02109-90006
- E-Series Computer Engineering Supplement Package 02109-90007
- Diagnostic Configurator Reference Manual 02100-90157
- 12903B Slide Mounting Kit Installation Instructions (for HP 12979/12990) 12903-90002
- 12903C Slide Mounting Kit Installation Instructions (for HP 2111) 12903-90003

HP 1000 F-Series Computer

operating and reference manual

HEWLETT hp PACKARD

HEWLETT-PACKARD COMPANY
11000 WOLFE ROAD, CUPERTINO, CALIFORNIA, 95014

Library Index Number
21M.320.02111-90001

MANUAL PART NO. 02111-90001

Printed: JUN 1978
Printed in U.S.A.

CONTENTS

Section I Page

SYSTEM FEATURES

Section II Page

OPERATING PROCEDURES

Section III Page

PROGRAMMING INFORMATION

Section IV Page

DYNAMIC MAPPING SYSTEM

CONTENTS (continued)

Section IV Page

DYNAMIC MAPPING SYSTEM (Cont.)

Section V Page

MICROPROGRAMMING

Section VI Page

INTERRUPT SYSTEM

Section VII Page

INPUT/OUTPUT SYSTEM

Appendix Page

ILLUSTRATIONS

Title **Page**

TABLES

ALPHABETICAL INDEX OF STANDARD INSTRUCTIONS

ALPHABETICAL INDEX OF STANDARD INSTRUCTIONS (Continued)

ALPHABETICAL INDEX OF DYNAMIC MAPPING SYSTEM INSTRUCTIONS*

*Standard on HP 2117F Computer, optional on HP 2111F.

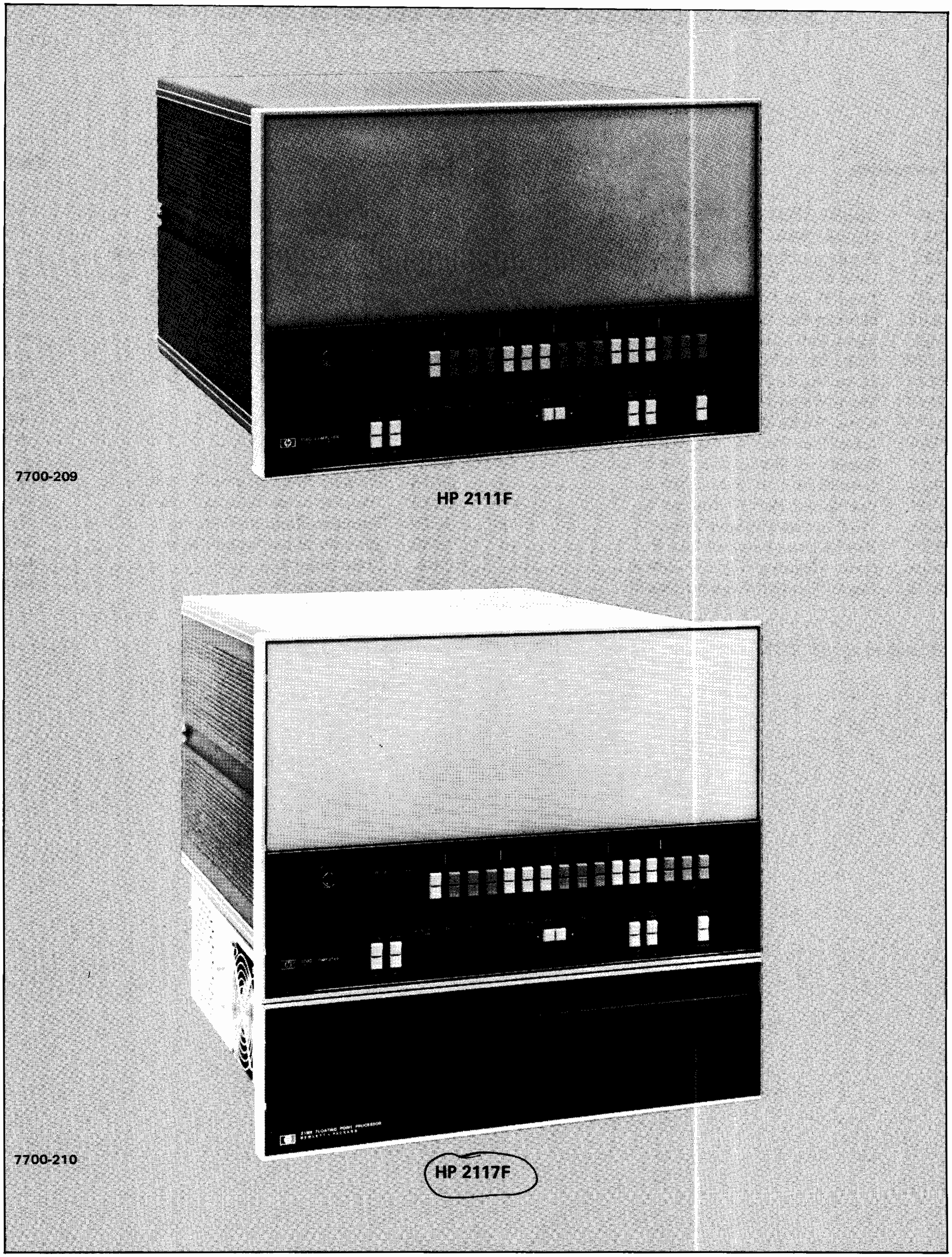


Figure 1-1. HP 1000 F-Series Microprogrammable Computers

SYSTEM FEATURES

SECTION I

The HP 1000 F-Series Computers HP 2111F and HP 21117F shown in figure 1-1 are high-performance machines designed to satisfy a wide range of computing needs. Because of a unique design philosophy, many features have been incorporated as standard in the F-Series; this same philosophy allows optional features to be added at low cost. F-Series computers have traditional HP quality and reliability built in from the ground up and compatibility with previous Hewlett-Packard computers is maintained. HP 1000 F-Series Computers (hereafter referred to as F-Series computers), provide very cost-effective solutions to a variety of systems applications.

1-1. ARCHITECTURE

F-Series computers have a proven architecture that features a fully microprogrammed control processor, including all arithmetic functions, input/output, and operator panel control. Four general purpose registers are available, two of which may be used as index registers. There are 128 standard instructions including index instructions, integer and floating point arithmetic instructions, input/output instructions, and a full complement of instructions for logical operations as well as bit and byte manipulation.

The F-Series computers have been optimized for performance. At the heart of the computer is a microprogrammed control section that directs the operations of the other functional units of the computer. In the F-Series, the control section has been speeded up for certain operations due to a sophisticated technique of varying the microinstruction cycle time. In addition, the efficiency of the microprogrammed routines that determine the machine language operation has been increased. The result is a high-performance computer that retains compatibility with earlier Hewlett-Packard computers and the flexibility of user microprogramming. The central processor unit (CPU) to memory interface is totally asynchronous in F-Series computers, thus assuring that faster memory speeds can be used to good advantage.

A hardware-implemented Floating Point Processor in the F-Series computers provides high-speed dedicated logic for performing floating point arithmetic operations. The Floating Point Processor (FPP) provides exceptionally fast execution of both single precision (32-bit) and extended precision (48-bit) floating point operations. Single precision operations are accurate to at least 6 decimal places and extended precision operations are accurate to at least 11 decimal places. (For the HP 2111F, the FPP is housed within the computer mainframe; for the HP 2117F, the FPP is housed in a separate assembly cabled to the computer mainframe.)

The F-Series computers include a powerful Scientific Instruction Set (SIS) that performs nine trigonometric and logarithmic functions. These instructions utilize the Floating Point Processor as a powerful computing resource to achieve extremely high performance in conjunction with high accuracy.

In addition, a set of instructions that accelerate execution of frequently-used FORTRAN operations are included in the F-Series Computers. These Fast FORTRAN Processor instructions perform parameter passing, array address calculations, and floating point conversion, packing, rounding and normalization operations.

The F-Series instruction set may be further expanded through the use of HP-supplied or user-written instruction sets. Optional enhancements include the DS/1000 firmware for DS/1000 network communications and RTE-IV Extended Memory Area firmware.

1-2. USER MICROPROGRAMMING

The power and flexibility of microprogramming is made readily available to the F-Series computer user through the microinstruction set of 211 micro-orders. Microprogrammers have access to 12 special scratch pad registers in addition to the other internal registers of the F-Series, and may address up to 16K, 24-bit words of control store. F-Series computers also support up to three levels of nested subroutines in microprograms. Closely resembling assembly language programming in simplicity, microprogramming offers the advantages of speed and security as well as the ability to expand the instruction set to meet any computing need. Microprogramming is supported by Hewlett-Packard through software assembly and debug packages and customer training courses. User-developed microprograms may be permanently fused in programmable Read-Only Memory (pROM) chips for mounting on the Firmware Accessory Board, or may be loaded into Writable Control Store (WCS) modules where they can be dynamically altered.

1-3. SELF-TEST ROUTINES

A comprehensive set of self-test routines permanently stored in Read-Only Memory (ROM) are standard in F-Series computers. Two of these routines, executed each time the IBL/TEST function is performed, tests the CPU and the enabled memory (up to 64k bytes) for quick verification of operating condition. A third test, executed every time the machine is powered up, tests the CPU and all installed memory. Two more tests, executed via the operator panel, test the Floating Point Processor and the Scientific Instruction Set firmware.

1-4. LOCAL AND REMOTE BOOTSTRAP LOADERS

The initial binary loading (IBL) function is easily performed on F-Series computers. For local bootstrap loading, a 64-word ROM-resident IBL program is called by pushbutton switch on the operator panel. A paper tape loader ROM and a disc loader ROM are standard. One or two additional loader ROM's may be included; these programs may be purchased as accessories or may be user-generated.

Computers at remote sites can be force-loaded from a central location through the use of the Remote Program Load (RPL) capability. All of the information normally keyed into the operator panel is set in a special set of switches on the CPU board so that the bootstrapping sequence may be initiated from a remote site.

1-5. POWER SYSTEM

F-Series computers are equipped with power systems designed to continue normal operations in environments where power may fluctuate widely. Input line voltages and frequencies may vary widely without affecting the operation of the computer. The optional Power Fail Recovery System provides automatic restart capability and, depending on the memory size, also provides between 1.75 and 4.25 hours of memory sustaining power in the event of complete power failure. (See Power Fail Recovery System specifications in table 1-1.)

1-6. MEMORY SYSTEMS

F-Series computers are available with either of two semiconductor memory systems based on 4k-bit and 16k-bit MOS/RAM semiconductor chips that offer field-proven reliability and economy. The standard memory system in the F-Series computers consists of an HP 2102E High Performance Memory Controller and either 64k bytes (HP 2111F) or 128k bytes (HP 2117F) of memory. For data integrity, memory parity checking is provided as a standard feature.

The optional fault control memory system provides fault-secure memory operation for the F-Series computers. The system consists of an HP 2102H High Performance Memory Fault Controller and one or more check bit array boards, along with the appropriate number of memory modules. The system is capable of correcting all single-bit errors and of detecting all double-bit errors and most multiple-bit errors. The fault control system is particularly valuable in computer systems with large amounts of memory, or where fault secure operation is essential.

Addressing physical memory configurations larger than 64k bytes is made possible by the use of the HP 13305A Dynamic Mapping System which is standard in the HP 2117F and optional in the HP 2111F. The Dynamic Mapping System (DMS), which is a combination of hardware and firmware, is a powerful memory management scheme that allows F-Series computer users to address up to two million bytes of memory and provides read and/or write protection of each individual 2048 byte page. Four independent memory maps are provided, one for the system, one for the user, and two for Port Controller maps for direct memory access operations. Control of the DMS is implemented through the use of 38 instructions.

1-7. INPUT/OUTPUT

The input/output system for F-Series computers features a multilevel vectored priority interrupt structure. There are 60 distinct interrupt levels, each of which has a unique priority assignment. Any I/O device can be selectively enabled or disabled, or the entire interrupt system (except power fail and parity error interrupts) can be enabled or disabled under program control.

Data transfer between the computer and I/O devices may take place under program control, Dual Channel Port Controller (DCPC) control, or under microprogram control. The DCPC provides two direct links between memory and I/O devices and is program assignable to any two devices. DCPC transfers occur on an I/O cycle-stealing basis not subject to the I/O priority interrupt structure. The total bandwidth through both DCPC channels is 2.0 million bytes per second (input); see Direct Memory Access specifications in table 1-1 for the output transfer rates and the DCPC latency times.

The HP 2111F Computer has nine I/O channels in the mainframe; the HP 2117F Computer has fourteen. The number of available channels may be increased by adding one or two HP 12979B I/O Extenders, providing sixteen channels each. All I/O channels are fully powered, buffered, and bidirectional. Because of the modular design of the F-Series computers, mainframe memory capacity is completely independent of I/O capacity so that either memory or I/O modules may be added without taking valuable mainframe space from the other. A full line of I/O interface controllers is available with F-Series computers for interfacing to any of the broad line of HP manufactured peripherals or to specialized devices.

1-8. SOFTWARE

The F-Series computers maintains extensive program compatibility with earlier HP 1000 and HP 2100 series computers so that users can take advantage of many man-years of software development. This includes compatibility of the Floating Point Processor (FPP) with the standard firmware floating point instructions of previous machines. (Note, however, that the FPP might not provide the same answers since it is far more accurate.)

A wide range of operating system software is available. Real-time executive (RTE) systems, available in disc and main memory-resident versions, are multiprogramming systems that permit priority scheduling of several real-time programs while concurrent background processing takes place.

The memory-based RTE-M and disc-based RTE-IV operating systems can support up to 2.048 megabytes of memory, managed by DMS. Comprehensive software is also available for computer networking.

Languages supported by Hewlett-Packard operating systems include two high-level compilers: HP FORTRAN IV; and HP BASIC; plus an extended, efficient assembler that is callable by FORTRAN. Utility software includes a debugging routine, editor, and an extensive library of commonly used computational routines.

F-Series computer users may also take advantage of a wide variety of thoroughly tested and documented programs that have been contributed to the Hewlett-Packard User Library.

1-9. SPECIFICATIONS

Table 1-1 lists the specifications for the HP 2111F and the HP 2117F Computers and the High Performance and Fault Control Memory Systems. Both computers have been product accepted by the Underwriters' Laboratories (UL) and the Canadian Standards Association (CSA).

1-10. SYSTEM EXPANSION AND ENHANCEMENT

Table 1-2 lists the options and accessories available to expand or enhance the computer system.

Table 1-1. Specifications

HARDWARE SUPPLIED	
Computer Mainframe:	HP 2111F Computer mainframe HP 2117F Computer mainframe and Floating Point Processor assembly
Memory Controller:	HP 2102E
Memory Module:	HP 2111F: Two HP 12741A modules HP 2117F: HP 12747H module
Dynamic Mapping System:	HP 13305A (HP 2117F)
CENTRAL PROCESSOR	
Address Space:	4,096 bytes (direct addressing) 65,536 bytes (indirect addressing) 2,097,152 bytes with Dynamic Mapping System
Word Size:	16 bits
Instruction Set:	163 standard instructions (HP 2111F); 201 standard instructions (HP 2117F)
Memory Reference:	14
Register Reference:	43
Input/Output:	13
Extended Arithmetic:	10
Index:	32
Bit Byte, Word Manipulation:	10
Floating Point:	16
Scientific:	9
Fast FORTRAN:	16
Dynamic Mapping:	38 (HP 2117F)
Registers:	10
Accumulators:	Two (A and B), 16 bits each. Explicitly addressable; also addressable as memory locations.
Index:	Two (X and Y), 16 bits each
Memory Control:	Two (T and P), 16 bits each; one (M), 15 bits.
Supplementary:	Two (Overflow and Extend), one bit each
Display:	One, 16 bits

Table 1-1. Specifications (Continued)

CONTROL PROCESSOR	
Address Space:	16,384 words (64 modules of 256 words each)
Word Size:	24 bits
Word Formats:	Four
Word Fields:	Five
Instruction Execution Time:	Variable; 175 nS or 280 nS
Micro-Orders	211
Operations:	15
Special:	32
ALU and Conditional:	68
Store:	32
S-Bus:	32
Reverse Skip Sense:	32
SCIENTIFIC INSTRUCTION SET	
Data Format:	Single precision (32-bit) floating point
Accuracy:	RMS relative error for the various Scientific Instruction Set functions is as follows:

Function	RMS Rel. Error	Function	RMS Rel. Error
SIN	8.80E-8	SQRT	6.74E-8
COS	8.82E-8	EXP	1.38E-7
TAN	1.99E-7	ALOG	1.28E-7
ATAN	1.34E-7	ALOGT	1.39E-7
TANH	1.33E-7		

INITIAL BINARY LOADERS	ROM resident; capacity of four 64-word programs callable from operator panel. Computer can be configured for forced cold loading from a remote site.
SELF-TEST	Automatic tests of CPU and memory operating condition. Executed on cold power-up and whenever operator panel IBL/TEST switch is pressed.
INPUT/OUTPUT	
Interrupt Structure:	Multilevel vectored priority interrupt; priority determined by interrupt location.

I/O SYSTEM SIZE	HP 2111F	HP 2117F
Standard I/O Channels	9	14
With One Extender	25	30
With Two Extenders	41	46

Compatibility:	Instruction set: The HP 1000 F-Series instruction set is backwards compatible with all previous 2100 Series computers. Program: All programs written for 2100 Series computers are compatible with the F-Series, except those with timing loop dependence.
Current Available for:	I/O, Memory and Accessories

MODEL	+5V	+12V	−12V	−2V
2111F	21.9A	2.5A	2.0A	6.0A
2117F	28.8A	2.5A	2.0A	6.0A

Table 1-1 Specifications (Continued)

INPUT/OUTPUT (Continued)

DC Required:

MODEL	+5V	−2V
12892B Memory protect	1.25A	.05A
12897B DCPC	2.4A	.05A
12731A MEM	3.9A	—
13307A DMI	.78A	—
13197A 1k WCS	2.2A	.01A
12990B Memory extender	0	—
12992 Loader ROMS (each)	.13A	—
12979B I/O Ext. buffer card	2.0A	1.35A
2102E Memory Controller	2.56A	—
12741A 32k Byte Memory Module	0.85A	—
12747H 128k Byte Memory Module	.48A	—
2102H Fault Memory Controller	3.3A	—
12779H Check Bit Array	.25A	—
12780H Check Bit Array	.3A	—
13047A 2k UCS	7.39A[1]	—
13304A FAB	1.8A[2]	—

[1] *1.15A + 0.78A for each 256 instructions; 7.39A when fully loaded.*

[2] *Due to power saver circuit, this is the maximum current regardless of amount of microcode loaded on the board.*

DIRECT MEMORY ACCESS Available only with DCPC accessory.

Number of Channels: Two

Word Size: 16 bits

Maximum Transfer Block Size: 32,768 words

I/O Assignable: Assignable to any two I/O channels; all logic necessary to facilitate bidirectional direct memory transfer to and from I/O is contained on DCPC (controller).

Transfer Rate:
(All rates in Mbytes/s)

(High Performance Memory)

(High Performance Fault Control Memory)

HP 2102E	Minimum	Typical	Maximum
Input w/DMS	2.282	2.284	2.284
w/o DMS			
Output w/DMS	2.038	2.114	2.196
w/o DMS	2.196	2.284	2.284
HP 2102H			
Input w/DMS	2.28	2.28	2.28
w/o DMS			
Output w/DMS	1.902	1.968	2.038
w/o DMS	2.038	2.114	2.196

DCPC Latency (Channel 1): Latency is defined as the time interval between the generation of a Service Request (SRQ) signal by an I/O device through the initiation of a DCPC channel 1 cycle to the completion of the I/O data transfer to or from the I/O interface PCA. Subsequent consecutive cycles execute at a specified DCPC rate.

Table 1-1. Specifications (Continued)

DIRECT MEMORY ACCESS (Continued)

Input and Output Latency Times:
(Times in us)

(High Performance Memory)

(High Performance Fault Control Memory)

HP 2102E	Minimum	Typical	Maximum
Input w/o DMS	2.21	2.21	2.28
w/DMS	2.21	2.28	2.35
Output w/o DMS	2.75	2.75	2.84
w/DMS	2.80	2.87	2.98
HP 2102H			
Input w/o DMS	2.31	2.42	2.52
w/DMS	2.38	2.56	2.74
Output w/o DMS	2.84	2.94	3.05
w/DMS	2.98	3.16	3.33

A USER MICROCODE sequence of seven consecutive reads may provide conditions to produce the absolute worst case latency time. The absolute worst case latency times are as follows:

Memory System	Input	Output
HP 2102E	3.050 us	3.681 us
HP 2102H	3.510 us	4.100 us

PHYSICAL CHARACTERISTICS

Width: 42.6cm (16 3/4 in) behind rack mount; 48.3cm (19 in) front panel width on sides

Depth: 62.2cm (24 1/2 in); 58.4cm (23 in) behind rack mounting ears

MODEL	HP 2111F	HP 2117F
Height	31.1cm (12-1/4 in)	44.5cm (17-1/2 in)
Weight	30kg (66 lbs)	50kg (110 lbs)

ELECTRICAL CHARACTERISTICS

Line Voltage: 2111F: 88-132V; 176-264V with option 015.
2117F: Computer mainframe same as 2111F; Floating Point Processor voltage selector offers choice of 90-110V, 108-132V, 198-242V, and 216-264V ranges.

Line Frequency: 47.5 to 66 Hz

Power Dissipation: 2111F: 770 watts (maximum). 2117F: 970 watts (maximum).

Table 1-1. Specifications (Continued)

ELECTRICAL CHARACTERISTICS (Continued)

Power Supply: Sustains computer over a line loss of no less than 8 ms at the nominal line (mains) voltage.

Input Line Transients: Sustains ±500V, 50μs pulse on power lines; sustains ±1kV, 100ns pulses on power lines.

Output Protection: All regulated voltages protected from overvoltage and overcurrent conditions.

Output Voltage Regulation: ±5% (except −2V is ±10%, and +30V is unregulated).

Thermal Sensing: Monitors internal temperature and automatically shuts down when computer temperature exceeds specified maximum operating temperature. Resets automatically when temperature returns to below specified maximum operating temperature.

ENVIRONMENTAL LIMITATIONS

Operating Temperature: 0° to 55°C (+32° to 131°F)

Storage Temperature: −40° to 75°C (−40° to 167°F)

Relative Humidity: 20% to 95% at 40°C (104°F), non-condensating

Ventilation and Heat Dissipation: Intake: left-hand side; Exhaust: right-hand side.

	MODEL	2111F	2117*
Heat dissipation	KCal/hr. max.	580	752
	BTU/hr. max.	2303	2986
Air flow	cubic meters /min.	7.9	11
	cubic feet /min.	280	390

*Includes both computer and floating point processor mainframes.

Altitude: Transportable to 15 240m (50 000 ft) in non-operating condition and 4570m (15 000 ft) for operation

Vibration and Shock: Vibration: 0.30mm (0.012 in) p-p, 10-55 Hz, 3 axis

Shock: 30g, 11 Ms, 1/2 sine, 3 axis

Contact factory for review of any application requiring operation under continuous vibration.

MEMORY SYSTEMS

Type: 4k and 16k N-channel MOS semiconductor RAM.

Word Size: 16 bits plus parity bit

Configuration: Controller plus multiple plug-in memory modules. Available in 32k and 128k byte modules.

Page Size: 2,048 bytes

Address Space: 65 536 bytes without DMS; 2 097 152 bytes with DMS (2111F and 2117F)

Table 1-1. Specifications (Continued)

MEMORY SYSTEMS (Continued)

System Cycle Times:
(All cycle times in ns)

(High Performance Memory)

(High Performance Fault Control Memory)

Cycle	Minimum	Typical	Maximum
READ w/o DMS	350	350	385
READ w/DMS	385	420	455
WRITE	350	350	385
REFRESH	350	350	385
READ w/o DMS	386	421	456
READ w/DMS	456	491	526
WRITE	386	421	456
REFRESH	386	421	456

Volatility Protection: Sustaining power for line loss of no less than 8ms at the minimum line (mains) voltage. Power fail recovery system is optional.

Parity Error Detection: Monitors all words read from memory. Switch selectable for either halt or ignore interrupt error when detected. With memory protect or DMS, interrupt on parity error occurs.

POWER FAIL

Interrupt Priority: Highest priority interrupt.

Power Failure: Detects power failure and generates an interrupt to user-written power-failure routine. A minimum of 500 μs is available for the routine.

Power Fail Recovery System: Available as an accessory. (HP 12991B). ✓

Power Restart: Detects resumption of power and generates an interrupt to user-written automatic restart program which has been protected in memory by the sustaining battery.

Power Control and Charge Unit: Monitors battery charge status and provides trickle charge.

Sustaining Battery: Type: 14 volt, 5 ampere-hours, sealed lead acid
Charging rate: 2A, maximum
Capacity:HP 12991B will sustain memory for the period of time shown in the graph below.

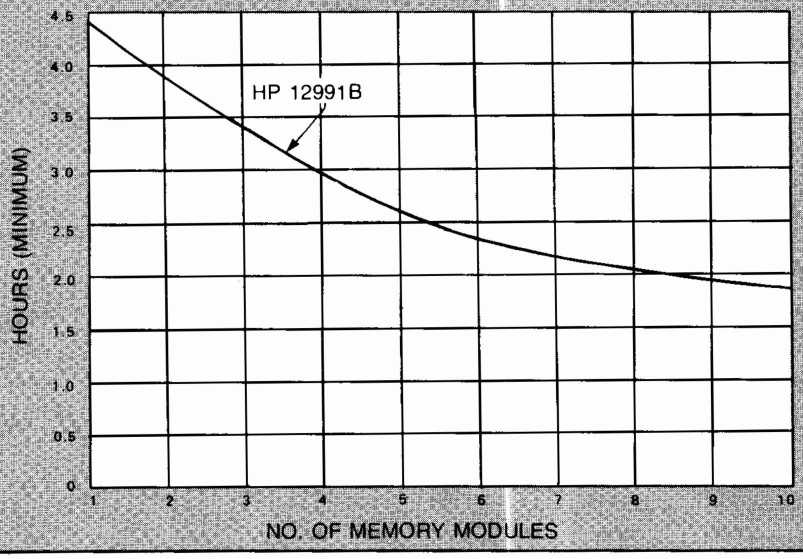


Table 1-2. Options and Accessories

DESCRIPTION	OPTION NO.	ACCESSORY NO.
HP 2111F and HP 2117F Computers		
Delete standard memory when ordering alternate memory	−014	—
230V Operation	−015	—
Power Fail Recovery System (HP 2111F, HP 2117F HP 12990B)	—	12991B
Memory Protect	—	12892B
Dual-Channel Port Controller (DCPC)	—	12897B
Dynamic Mapping System[1]	—	13305A
1K Writable Control Store	—	13197A
Slide Mounting Kit (HP 12979B, HP 12990B)	—	12903B
Slide Mounting Kit (HP 2111F)	—	12903C
Dynamic Mapping Instructions	—	13307A
Memory Expansion Module	—	12731A
Loader ROM for 2644A and 2645[2]	—	12992C
Loader ROM for 7970B/E[2]	—	12992D
2k User Control Store	—	13047A
HP 2102 MOS Memory Systems		
High Performance Memory System	—	—
Memory Controller	—	2102E
32k Byte Memory Module	—	12741A
128k Byte Memory Module	—	12747H
High Performance Fault Control Memory System	—	—
Memory Controller	—	2102H
Check Bit Array (lower half)	—	12779H
Check Bit Array (full)	—	12780H
32k Byte Memory Module	—	12741A
128 Byte Memory Module	—	12747H
Input/Output Extender	—	12979B
230V Operation	−015	—
DCPC for I/O Extender	—	12898A
Memory Extender	—	12990B
230V Operation	−015	—

[1] Includes Memory Protect.
[2] See latest Loader ROM Installation Manual for a complete listing of Loader ROMs.

OPERATING PROCEDURES

SECTION II

The section describes the hardware registers accessible to the programmer and the functions of the various operating controls and indicators. Also included are basic operating examples such as a cold start procedure to load a program via an input device, manually loading a short program via the operator panel, and running a program after it has been loaded into memory.

2-1. HARDWARE REGISTERS

The computer has eight 16-bit working registers which can be selected for display and modification by operator panel controls; two 1-bit registers; and one 16-bit display register. The functions of these registers are described in following paragraphs.

2-2. A-REGISTER

The A-register is a 16-bit accumulator that holds the results of arithmetic and logical operations performed by programmed instructions. This register can be addressed directly by any memory reference instruction as location 000000 (octal), thus permitting interrelated operations with the B-register (e.g., "add B to A," "compare B with A," etc.) using a single-word instruction.

2-3. B-REGISTER

The B-register is a second 16-bit accumulator, which can hold the results of arithmetic and logic operations completely independent of the A-register. The B-register can be addressed directly by any memory reference instruction as location 000001 (octal) for interrelated operations with the A-register.

2-4. M-REGISTER

The M-register holds the address of the memory cell currently being read from or written into by the CPU.

2-5. T-REGISTER

All data transferred into or out of memory is routed through the T-register. When displayed, the T-register indicates the contents of the memory location currently pointed to by the M-register. The A- or B-register contents are displayed if the M-register contents are 000000 or 000001, respectively.

2-6. P-REGISTER

The P-register holds the address of the next instruction to be fetched from memory.

2-7. S-REGISTER

The S-register is a 16-bit utility register. The S-register can be addressed as an input/output device (select code 01) and, in the run mode, it is displayed in the operator panel display register. Thus, the S-register may serve as a communication link between the computer and operator.

2-8. EXTEND REGISTER

The one-bit extend register is used by rotate instructions to link the A- and B-registers or to indicate a carry from the most-significant bit (bit 15) of the A- or B-register by an add instruction or an increment instruction. This is of significance primarily for multiple-precision arithmetic operations. If already set (logic 1), the extend bit cannot be cleared by a carry. However, the extend bit can be selectively set, cleared, complemented, or tested by programmed instructions. When the operator panel EXTEND indicator is lighted, the extend bit is set. This register can also be accessed from the operator panel by entering the special register display mode described under paragraph 2-23.

2-9. OVERFLOW REGISTER

The one-bit overflow register is used to indicate that an add instruction, divide instruction, or an increment instruction referencing the A- or B-register has caused (or will cause) the accumulators to exceed the maximum positive or negative number that can be contained in these registers. The overflow bit can be selectively set, cleared, or tested by programmed instructions. The operator panel OVERFLOW indicator will remain lighted until the overflow is cleared. This register can also be accessed from the operator panel by entering the special register display mode described under paragraph 2-23.

2-10. DISPLAY REGISTER

The display register, which is included on the operator panel, provides a means of displaying and/or modifying the contents of the eight 16-bit working registers and the special registers when the computer is in the halt mode.

An illuminating indicator is located directly above each of the 16-bit switches; a lighted indicator denotes a logic 1 and an unlighted indicator denotes a logic 0. When the computer is in the run mode, the contents of the S-register are displayed automatically.

2-11. X- AND Y-REGISTERS

These two 16-bit registers, designated X and Y, are accessed through the use of 30 index register instructions and 2 jump instructions described under paragraphs 3-24 and 3-25, respectively. These registers can also be accessed from the operator panel by entering the special register display mode described under paragraph 2-23.

2-12. OPERATOR PANEL AND POWER SUPPLY OPERATING CONTROLS

The location and function of the various controls and indicators mounted on the operator panel and the power supply are illustrated in figure 2-1 and described in table 2-1. All operator panel controls are two-position, momentary-contact rocker switches; the status of the computer is displayed by light-emitting diodes.

2-13. REAR PANEL

The rear panel and the I/O PCA cage for the HP 2111F and HP 2117F Computers are shown in figures 2-2 and 2-3 and described in tables 2-2 and 2-3, respectively.

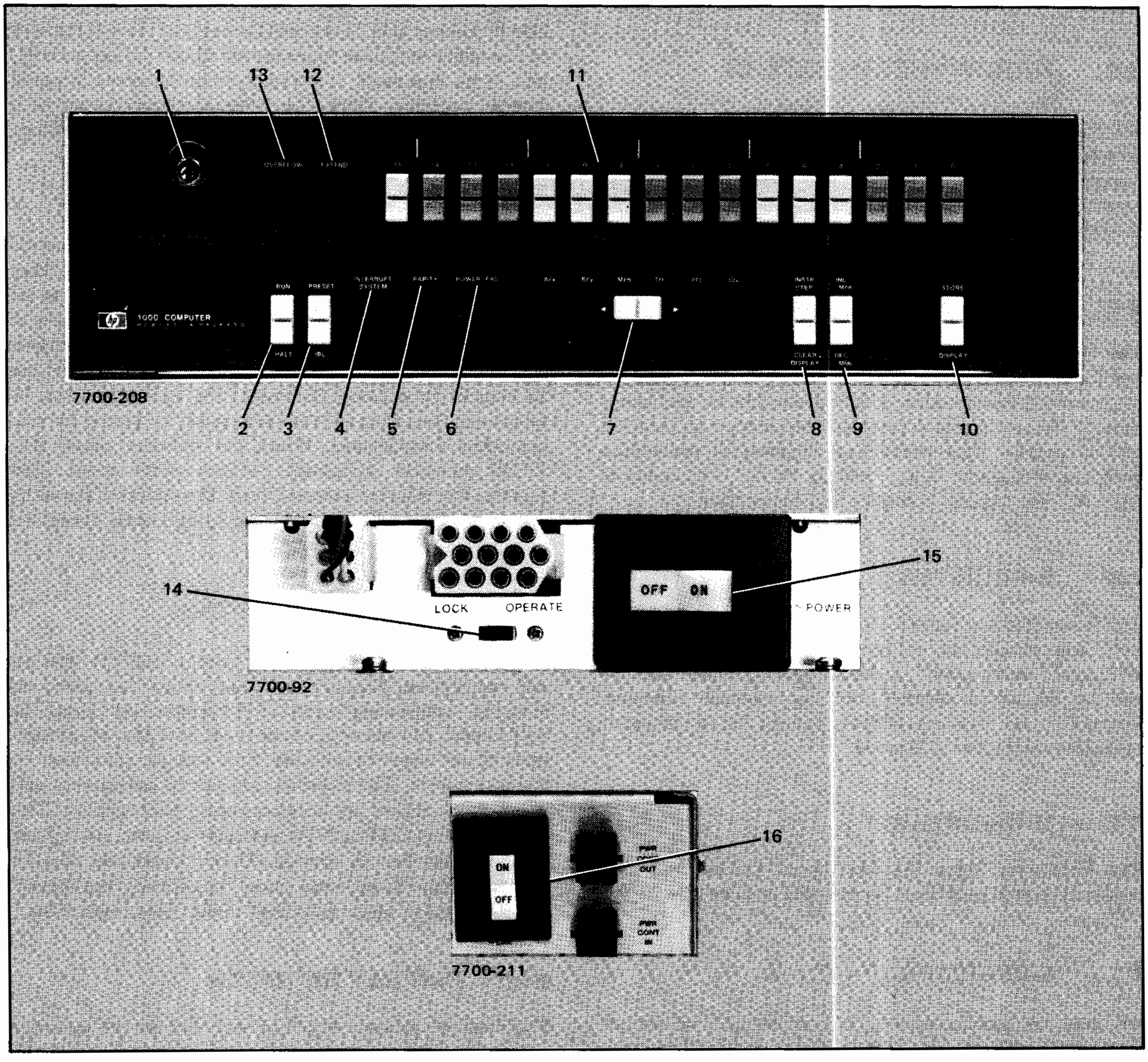


Figure 2-1. Operator Panel and Power Supply Controls and Indicators

Table 2-1. Operator Panel and Power Supply Control and Indicator Functions

<table>
<tr><th>FIG. 2-1, INDEX NO.</th><th>NAME</th><th>FUNCTION</th></tr>
<tr><td>1</td><td>Key</td><td>Secures the operator panel when access to the ~POWER OFF/ON and LOCK/OPERATE switches is not desired.</td></tr>
<tr><td>2</td><td>RUN/HALT</td><td>RUN. Starts CPU and lights the RUN indicator. All operator panel functions are disabled except Display Register, CLEAR DISPLAY, and HALT. Pressing RUN automatically causes the S-register contents to be displayed, and no other register can be selected during the run mode; thus, the Display Register effectively becomes the S-register, which may be addressed as select code 01 by the program.
HALT. Halts the computer at the end of the current instruction and turns off the RUN indicator. All other operator panel controls become enable. The T-register is selected automatically for display.</td></tr>
<tr><td>3</td><td>PRESET-IBL/TEST</td><td>PRESET. Disables the interrupt system and clears the parity indicator and overflow bit (if set). From I/O channel 06 up, clears control flip-flops and sets flags. Pressing and holding PRESET upon the restoration of power will force a ARS condition (see paragraph 6-1).
IBL (initial binary loader)/TEST. Causes the contents of the selected loader ROM to be written into the uppermost 64 memory locations and the automatic execution of firmware self-tests 1 and 2. (Refer to paragraph 2-31.) Bits 15 and 14 of the S-register select the desired loader ROM as follows:
<table>
<tr><th colspan="2">BITS</th><th rowspan="2">LOADER SELECTED</th></tr>
<tr><th>15</th><th>14</th></tr>
<tr><td>0</td><td>0</td><td>Loader ROM 00_2</td></tr>
<tr><td>1</td><td>0</td><td>Loader ROM 10_2</td></tr>
<tr><td>0</td><td>1</td><td>Loader ROM 01_2</td></tr>
<tr><td>1</td><td>1</td><td>Loader ROM 11_2</td></tr>
</table>
Bits 6 through 11 of the S-register must be set to the octal select code of the loading device.</td></tr>
<tr><td>4</td><td>INTERRUPT SYSTEM</td><td>Indicates the status of the interrupt system. When lighted, the interrupt system is enabled (Flag set); when turned off, the interrupt system is disabled (Flag clear).</td></tr>
<tr><td>5</td><td>PARITY</td><td>Lights when a parity error occurs as a result of reading from memory. In the halt mode, the light can be turned off by pressing the PRESET switch. With the memory protect or DMS installed and the parity error interrupt enabled, the indicator is turned off automatically by a parity error interrupt and is therefore not ordinarily lighted long enough to be visible.</td></tr>
</table>

Table 2-1. Operator Panel and Power Supply Control and Indicator Functions (Continued)

FIG. 2-1, INDEX NO.	NAME	FUNCTION
6	POWER FAIL	If the power fail/automatic restart feature is enabled (i.e., internal ARS/ARS switch is set to ARS position as described in Section VI) the indicator will light when power is restored. This light can be turned off by pressing the PRESET switch in the halt mode.
7	◀ Register Select ▶	In the halt mode, this switch allows any one of the working registers or special registers (A, B, M, T, P, S, x, y, m, t, f, or s) to be selected for display and/or modification. Pressing the left half (◀) of the switch moves the "dot" indicator left; pressing the right half (▶) of the switch moves the "dot" indicator right. The register currently selected is indicated by the appropriate indicator light. Note: During standard register display mode, the "dot" indicator is the lighted indicator; i.e., the selected register (A, B, M, T, P, or S) is indicated by the lighted indicator and the remaining indicators are not lighted. During special register display mode, the "dot" indicator is the unlighted indicator; i.e., the selected register (x, y, m, t, f, or s) is indicated by the indicator not lighted and the remaining indicators are lighted. After a programmed or manual halt, the T-register is selected automatically for display. In this case, the T-register holds the contents of the last accessed memory cell. In the case of a programmed halt, the halt instruction will be displayed.
8	INSTR STEP/CLEAR DISPLAY	INSTR STEP. Pressing and releasing this switch while in the halt mode advances the program to the next instruction. If the T-register indicator lights when the switch is released, infinite indirect addressing is indicated. Actuating this switch does not actually place the computer in the run mode. (See note for additional information.) CLEAR DISPLAY. In the run or halt mode, clears the Display Register; i.e., contents become 000000.
9	INC M/m-DEC M/m	INC M. In the halt mode, increments the M-register contents during standard display mode and the m-register contents during special display mode. (Refer to paragraph 2-25.) DEC M. In the halt mode, decrements the M-register contents during standard display mode and the m-register contents during the special display mode. (Refer to paragraph 2-25.) Note: Incrementing and decrementing occur even when the M- or m-register is not displayed.
10	STORE/MODE	STORE. In the halt and standard display modes, stores the contents of the Display Register into the selected working register (A, B, M, T, P, or S). If the Register Select "dot" is pointing to T and STORE is pressed, the contents of the Display Register will be loaded into memory cell m, the M-register will be incremented automatically to m + 1, and the Display Register will *not* be updated. This latter feature allows the same data to be stored in consecutive memory locations (e.g., halts in the trap cells, same word into a buffer, etc.). If the Register Select "dot" is pointing to any register other than T, only that one register will be updated when STORE is pressed. (Refer to paragraph 2-23 STORE functions during special register display mode.)

Table 2-1. Operator Panel and Power Supply Control and Indicator Functions (Continued)

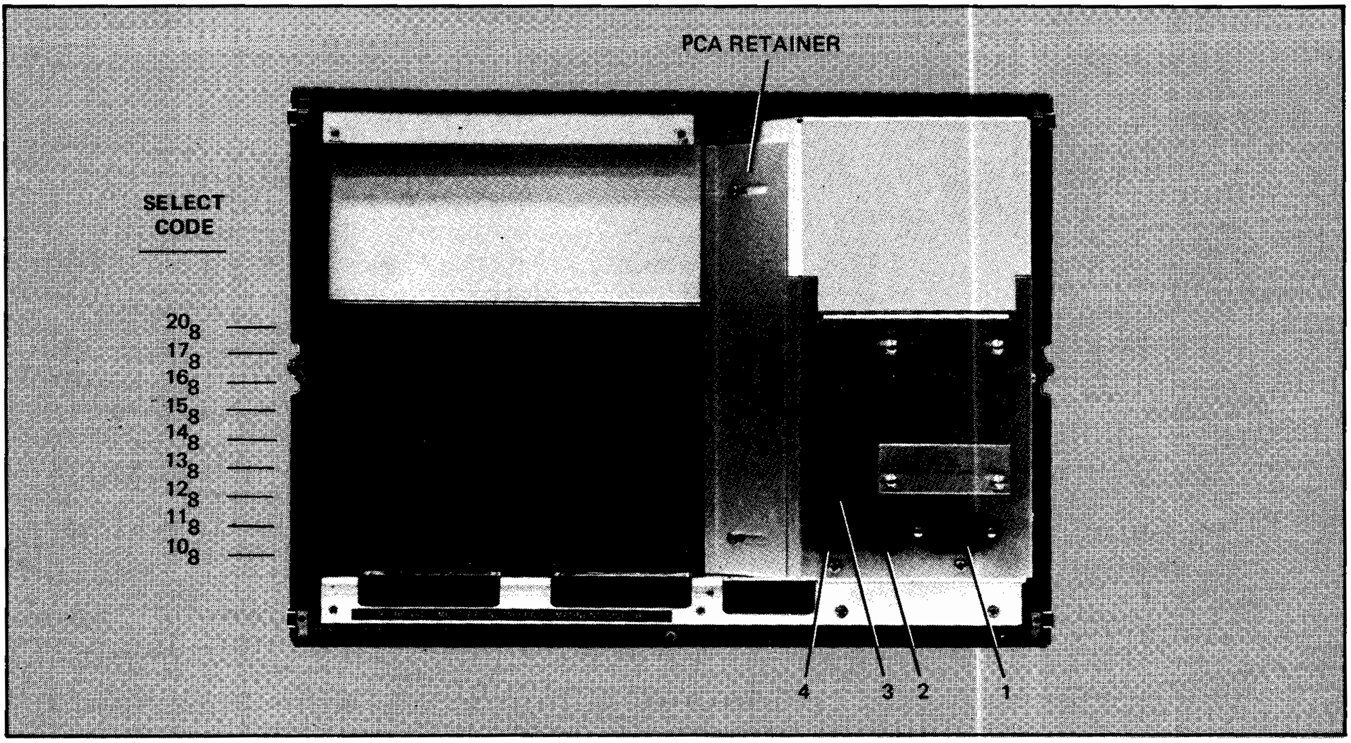
FIG. 2-1, INDEX NO.	NAME	FUNCTION
10	STORE/MODE	MODE. Selects either standard or special register display mode. If pressed when operating in standard register display mode, switches operator panel to special register display mode. If pressed when operating in special register display mode, switches operator panel back to standard register display mode.
11	Display Register	In the halt mode, displays the contents of the register currently pointed to by the Register Select "dot;" only the S-register is displayed during the run mode. A logic 1 is signified when the displayed bit indicator is lighted; a logic 0 is signified when the displayed bit indicator is not lighted. Pressing the upper half of the switch sets that bit to a logic 1; pressing the lower half of the switch sets that bit to a logic 0. The Display Register is cleared to all zeros when the CLEAR DISPLAY switch is pressed.
12	EXTEND	In both the run and halt modes, continuously displays the content of the extend register. When lighted, the extend bit is set (logic 1).
13	OVERFLOW	In both the run and halt modes, continuously displays the content of overflow register. When lighted, the overflow bit is set (logic 1).
Power Supply Front Panel Controls		
14	LOCK/OPERATE	LOCK. The RUN and HALT switches are disabled; all other functions are enabled (within the constraints of the run/halt modes). OPERATE. All switches are enabled.
15	~POWER ON/OFF	Two position circuit breaker. Controls application of ac line power to computer power supply and ventilating fans; provides protection against ac line power overload.
FPP (2117) only Rear Panel Control		
16	~LINE ON/OFF	Two position circuit breaker. Controls application of ac line power to Floating Point Processor power supply and ventilating fans; provides protection against ac line power overload.

NOTE

When pressing the INSTR STEP switch and performing a jump instruction while monitoring the T-, P-, or M-register, the following will be hoted:

a. The P-register will go to the operand target address.
b. The M-register will go to the operand target minus one (M=P−1).
c. The T-register will display the memory location prior to the current instruction to be executed.

Pressing the INSTR STEP switch will not cause an instruction step if there is a pending interrupt and the interrupt system is on; pressing PRESET or turning the interrupt system off will re-enable the INSTR STEP function.



7700-207

Figure 2-2. HP 2111F Rear Panel and I/O PCA Cage

Table 2-2. HP 2111F Rear Panel Features

FIG. 2-2, INDEX NO.	NAME	FUNCTION
1	~LINE connector	Three-input power connector; provides means of connecting ac line power to computer.
2	BAT INPUT connector	Nine-pin connector; provides means of connecting optional battery to memory sustaining circuits.
3 4	PWR CONT IN and PWR CONT OUT connectors	Two nine-pin connectors; provides means of connecting an external memory extender, I/O extender, or satellite computer (in any combination of two units) to main computer. The power supplies in all the interconnected units must be turned on before the CPU will start. When connected, the CPU monitors the ac and dc power in these units. An ac power failure in any one of the interconnected units will cause a power fail interrupt to be generated by the CPU and the CPU to halt. A dc power failure will cause the computer to stop. The interconnected units will remain inoperative until the failure has been corrected.

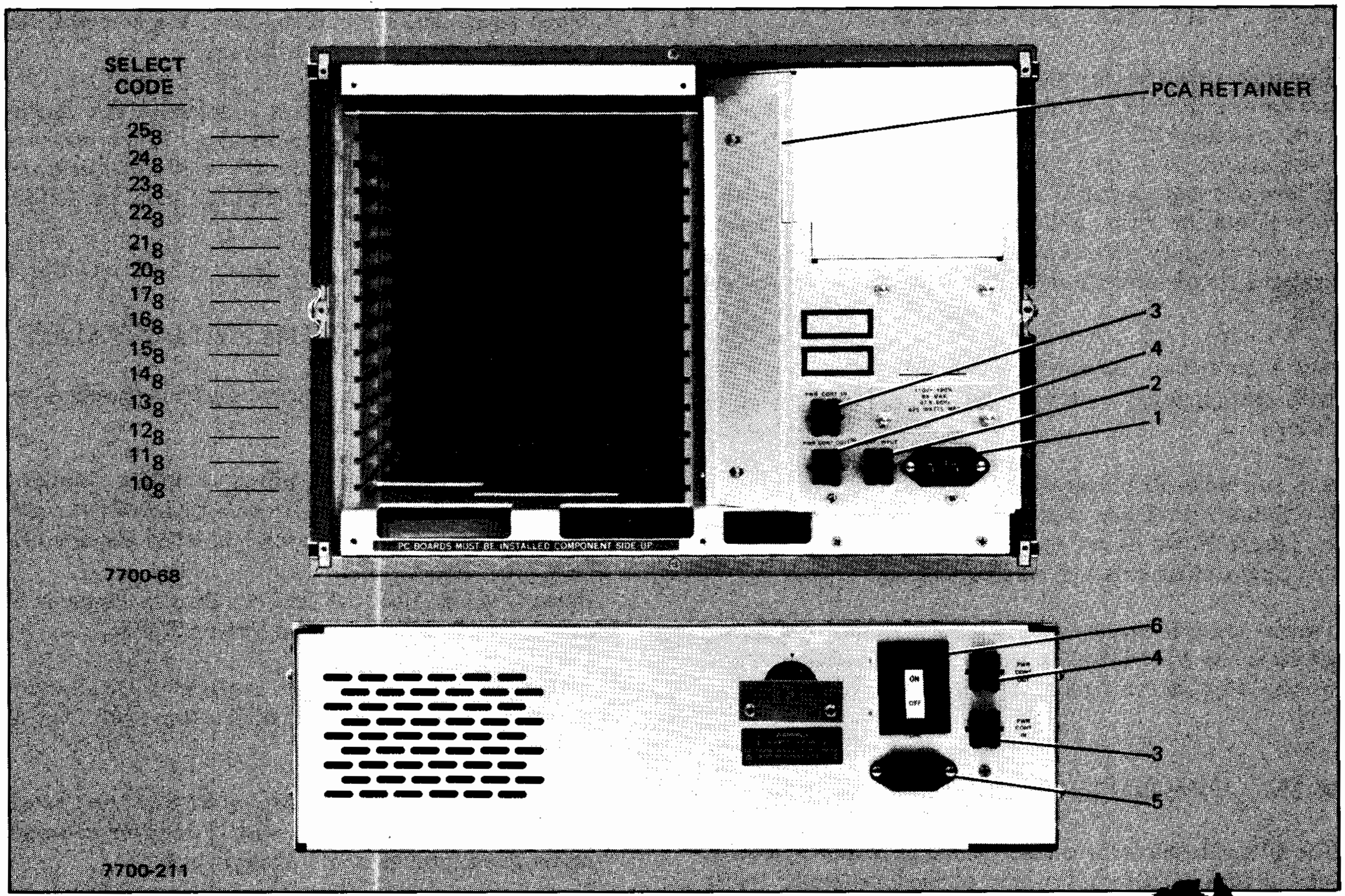


Figure 2-3. HP 2117F Rear Panel and I/O PCA Cage

Table 2-3. HP 2117F Rear Panel Features

FIG. 2-3, INDEX NO.	NAME	FUNCTION
1	~LINE connector	Three-input power connector; provides means of connecting ac line power to computer.
2	BAT. INPUT connector	Nine-pin connector; provides means of connecting optional batteries to memory sustaining circuits.
3 4	PWR CONT IN and PWR CONT OUT connectors	Two nine-pin connectors; provides means of connecting an external memory extender, I/O extender, or satellite computer (in any combination of two units) to main computer. The power supplies in all the interconnected units must be turned on before the CPU will start. When connected, the CPU monitors the ac and dc power in these units. An ac power failure in any one of the interconnected units will cause a power fail interrupt to be generated by the CPU and the CPU to halt. A dc power failure will cause the computer to stop. The interconnected units will remain inoperative until the failure has been corrected.
5	~LINE connector	Three-input power connector; provides means of connecting ac line power to Floating Point Processor.
6	~LINE ON/OFF	Two position circuit breaker. Controls application of ac line power to Floating Point Processor power supply and ventilating fans; provides protection against ac line power overload.

2-14. INTERNAL SWITCHES

Two toggle switches are mounted on the rear of the central processor unit (CPU) printed-circuit assembly (PCA). The setting of the ARS/$\overline{\text{ARS}}$ switch determines the action that the computer will take in the event of a primary power failure and the setting of the HLT /INT-IGNORE switch determines the action to be taken in the event of a parity error or memory protect violation. Programming considerations concerning these switches are given in Section VI.

Also mounted on the central processor unit PCA are the RPL configuration switches. These switches as used as follows:

a. To enable RPL operation.
b. Select one of two loader ROM's (10_2 and 11_2).
c. Patch in the select code of loading device.
d. Select subchannel of loading device.

Details concerning the configuration of these switches are given in the *HP 1000 F-Series Computer Installation and Service Manual,* part no. 02111-90002.

2-15. OPERATING PROCEDURES

The following procedures describe a cold power-up; how to load programs manually; how to load programs using punched tape, disc, magnetic tape, or other such media; how to verify and run programs; and how to enter the special register display mode.

2-16. COLD POWER-UP

Perform the cold power-up as follows:

a. Lower operator panel. Set the ~POWER OFF/ON switch to OFF. If computer is equipped with an optional power fail recovery system, set BATTERY switch to OFF. On the Floating Point Processor (if present), set the ~LINE switch to ON.

b. Set LOCK/OPERATE switch to OPERATE. Wait approximately three seconds and set ~POWER OFF/ON switch to ON.

c. Set BATTERY switch to ON. Raise and close the operator panel. Turn key fully counterclockwise.

d. The diagnostic will begin execution and the Display Register can be observed incrementing as each 64k byte block of memory is tested. When a cold power-up is performed, the computer automatically executes a firmware self-test that checks most of the computer registers and functions and all physical memory. (Refer to paragraphs 2-31 through 2-38 for additional firmware self-test information.) This test executes in approximately 10 seconds or less depending on memory size and clears memory upon completion.

e. Upon successful completion, the T-register will automatically be selected for display.

If a computer or memory failure is detected, the display register and all six working register indicators are lighted. In this case, refer to the *HP 1000 F-Series Computer Installation and Service Manual,* part no. 02111-90002 for further analysis.

2-17. LOADING PROGRAMS MANUALLY

Short programs can be loaded manually from the operator panel as follows:

a. Press MODE switch if required to obtain standard register display mode and press left half (◀) or right half (▶) of Register Select switch to select M-register.

b. Press CLEAR DISPLAY and set Display Register to starting address of program. Press STORE.

c. Select T-register and change contents of Display Register to binary code of first instruction to be loaded; press STORE.

d. Enter next instruction in Display Register and press STORE. (Pressing STORE with T-register selected automatically increments M-register.)

e. Repeat step d until entire program has been loaded.

2-18. LOADING PROGRAMS FROM PAPER TAPE READER

Use the following steps to first load the contents of the standard paper tape loader ROM into memory and then load your program by means of a tape reader. Proceed as follows:

a. Press MODE switch if required to obtain standard register display mode and press left half (◀) or right half (▶) of Register Select switch to select S-register.

b. Press CLEAR DISPLAY and set bits 6 through 11 to display octal select code of tape reader.

c. Press STORE, PRESET, and then press IBL/TEST. The paper tape loader is now loaded into the uppermost 64 locations of memory and the select code of the tape reader is patched according to the contents of the S-register. The P-register contains the address of the first instruction of the loader. (Starting addresses versus memory size are listed in table 2-4.)

d. A successful load is indicated if the OVERFLOW light remains off. An unsuccessful load is indicated if the OVERFLOW light is on; this will occur if the select code programmed in step b was less than 10 (octal) or if a memory hardware fault is detected.

e. Turn on tape reader and prepare it for reading. Press PRESET and then press RUN. The program will now be read into memory and the computer will halt with the T-register selected automatically. A successful load is indicated if the Display Register contents are 102077 (octal).

If the halt code displayed is not 102077 (octal), one of two possible error condition halt codes will be displayed. If the halt code displayed is 102055 (octal), an address error is indicated; check to ensure that the proper tape was used or that the tape was not installed backwards. If the halt code displayed is 102011 (octal), a checksum error is indicated; check for a possible defective or dirty tape or tape reader.

Table 2-4. Starting Address Vs Memory Size

MEMORY SIZE (in bytes)	STARTING ADDRESS (in octal) OF THE PAPER TAPE LOADER
32k	037700
64k and up	077700

2-19. LOADING PROGRAMS FROM DISC DRIVE

Use the following steps to first load the contents of the standard disc loader ROM into memory and then load your program by means of an HP 7900A, HP 7901A, HP 7905A, HP 7906A, or HP 7920A Disc Drive. Proceed as follows:

a. Press MODE switch if required to obtain standard register display mode and press left half (◀) or right half (▶) of Register Select switch to select S-register.

b. Press CLEAR DISPLAY and set bit 15 to logic 1 to select standard disc loader ROM.

c. Set bits 13 and 12 as shown below to select appropriate disc drive.

BITS		DISC SELECTED
13	12	
0	0	HP 7900A or HP 7901A
0	1	HP 7905A/7906, or HP 7920A

d. Set bits 11 through 6 to octal select code of disc drive interface PCA.

e. Set bits 2 through 0 as shown below to select corresponding disc subchannel.

BITS		DISC LOADING DEVICE
1	0	
0	0	HP 7900A (fixed disc)
0	1	HP 7900A or HP 7901A (removable disc)
0	0	HP 7905A/7906 (Head #0, top of removable disc) HP 7920A (Head #0)
0	1	HP 7905A/7906 (Head #1, bottom of removable disc) HP 7920A (Head #1)
1	0	HP 7905A/7906 (fixed disc) HP 7920A (Head #2)
1	1	HP 7920A/7906 (Head #3)

f. Press STORE, PRESET, and then IBL/TEST. The disc loader is now loaded into the uppermost 64 locations of memory and the select code of the disc drive is patched according to the contents of the S-register. The P-register contains the address of the first instruction of the loader. (Starting addresses versus memory size are listed in table 2-4.)

g. A successful load is indicated if the OVERFLOW light remains off. An unsuccessful load is indicated if the OVERFLOW light is on; this will occur if the select code programmed in step d was less than 10 (octal) or if a memory hardware fault is detected.

h. Turn on and prepare disc drive for operation and then press RUN. The program will now be read into memory and the computer will halt with the T-register selected automatically. A successful load is indicated if the Display Register contents are 102077 (octal).

2-20. LOADING PROGRAMS FROM OTHER LOADING DEVICES

The following procedure is used when loading programs from a disc, magnetic tape, or other such media. The contents of the optional loader ROM, associated with the loading device, must be loaded before the program can be loaded. Locations have been provided within the computer to accommodate two optional loaders; i.e., optional loader ROM 01_2 and 11_2. Each of these loaders is used to control the loading of programs from a particular type of loading device. It is assumed that the optional loader ROM, associated with the loading device to be used, is installed in the computer and that its location is known.

Use the following steps to first load the contents of one of the optional loader ROM's into memory and then load your program by means of a disc, magnetic tape, or other such media. The program must be in binary form and must contain absolute addresses. Assuming that the load-

S = 041201 B

ing device has been prepared for reading, proceed as follows:

a. Press MODE switch if required to obtain standard register display mode and press left half (◀) or right half (▶) of Register Select switch to select S-register.

b. Press CLEAR DISPLAY and set bit 6 through 11 to display octal select code of loading device.

c. Set bits 15 and 14 as listed in table 2-5 to select the optional loader corresponding to your loading device.

d. Set bits 0 through 5, 12, and 13 as outlined in the instructions provided with optional loader.

e. Press STORE and then press IBL/TEST. The optional loader is now loaded into the uppermost 64 locations of memory and the select code of the loading device is patched according to the contents of the S-register. The P-register is now pointing to the address of the first instruction of the optional loader. (Starting addresses versus memory size are listed in table 2-4.)

f. A successful load is indicated if the OVERFLOW light remains off. An unsuccessful load is indicated if the OVERFLOW light is on; this will occur if the select code programmed in step b was less than 10 (octal) or if a memory hardware fault is detected.

g. Verify that loading device is prepared for reading. Press PRESET and then press RUN. The program will now be read into memory and the computer will halt with the T-register selected automatically. A successful program load is typically indicated if the contents of the Display Register are 102077 (octal). Refer to the instructions included with each optional loader for the specific halt code used.

Table 2-5. Optional Loader Selection

BIT		LOADER SELECTED
15	14	
0	1	Loader ROM 01_2
1	1	Loader ROM 11_2

2-21. VERIFYING PROGRAMS

If desired, programs may be verified after loading by the following procedure:

a. Press MODE switch if required to obtain standard register display mode and press left half (◀) or right half (▶) of Register Select switch to select M-register.

b. Press CLEAR DISPLAY and set Display Register to the binary starting address of the program.

c. Press STORE. Select T-register and verify that the binary instruction code is displayed as desired for the first program instruction.

d. Press INC M to increment the contents of the M-register by one and verify that the binary instruction code displayed is as desired for next programmed instruction.

e. Repeat step d until all programmed instructions have been verified. Pressing DEC M permits the previous programmed instruction to be verified.

2-22. RUNNING PROGRAMS

To run a program after it has been loaded, proceed as follows:

a. Press MODE switch if required to obtain standard register display mode and press left half (◀) or right half (▶) of Register Select switch to select P-register.

b. Press CLEAR DISPLAY and set Display Register to starting address of program.

c. Press STORE, PRESET, and RUN.

The RUN indicator will remain lighted as long as the program is running. If the LOCK/OPERATE switch is set to OPERATE, all operator panel controls except the Display Register, CLEAR DISPLAY, and HALT switches are disabled.

During the run mode, the contents of the S-register are automatically selected for display in the Display Register and none of the other registers can be selected. Therefore, the Display Register effectively becomes the S-register and it can be directly addressed as I/O select code 01 (octal) by the program.

If the LOCK/OPERATE switch is set to LOCK, the functions of the RUN/HALT switch are disabled. All other operator panel controls are enabled within the constraints of the run or halt mode of operation.

NOTE

> If the computer has the Power Fail Recovery System installed, the BAT TEST switch *must not* be pressed unless the battery selector is set to INT.

2-23. SPECIAL REGISTER DISPLAY MODE

The special register display mode provides the capability of displaying and/or modifying the contents of the X- and Y-registers (paragraph 2-11), the Dynamic Mapping System (DMS) registers, the extend and overflow registers (paragraphs 2-8 and 2-9), the central interrupt register (CIR), and the interrupt system. To enter the special register display mode, use the MODE switch as discussed in table 2-1 and press the left half (◀) or right half (▶) of the

Register Select switch as required to move the unlighted "dot" indicator above the register (x, y, m, t, f, or s) to be displayed. It should be noted that the operator panel is switched back to the standard register display mode whenever the RUN switch is pressed. Each of the special register displays is discussed in the following paragraphs.

f, or s) to be displayed. It should be noted that the operator panel is switched back to the standard register display mode whenever the RUN switch is pressed. Each of the special register displays is discussed in the following paragraphs.

2-24. X- AND Y- (x and y) REGISTERS. When either of these registers is selected, the current contents of the register is indicated by the Display Register as discussed in paragraph 2-10 and table 2-1. If the STORE switch is pressed while either x or y is selected, the contents of the display are loaded into the selected register. The display is not altered.

2-25. DMS MAP (m, t, and f) REGISTERS. The m-register is a 7-bit register that holds the current memory map register number (0 - 177_8) being read from or written into by the CPU. When selected for display, bits 6 - 0 indicate the memory map register number on the Display Register. The memory map register number can be incremented or decremented with the INC M/m - DEC M/m switch as discussed in table 2-1. Any memory map can be quickly accessed when the m-register is selected for display by entering the memory map register number on the Display Register and pressing the STORE switch. To read the contents of the selected register number, select the t-register for display. The t-register is a 16-bit register that holds the selected register number contents and the read/write protect bits. When selected for display, bits 9 - 0 on the Display Register indicate the contents of the memory map register addressed by the m-register. If bit 15 is set (logic 1), the memory page is read-protected; if bit 14 is set, the memory page is write-protected. Bits 13 - 10 are always zero. If DMS is not installed and this register is selected for display, the display will be all 1's. To change the contents of the addressed memory map, enter the new contents on the Display Register with bit switches 9 - 0 and press the STORE switch. To read- or write-protect the memory map register, set the Display Register bit switches 15 or 14 to 1, respectively and press the STORE switch. If the INC M/m - DEC M/m switch is pressed when this register is selected for display, the contents of the next or previous memory map register address will be displayed.

The f-register is the 16-bit Memory Expansion Module status register and, when selected for display, indicates the address of the base page fence and DMS status as discussed in paragraph 4-2. If DMS is not installed and this register is selected for display, the display will be all 1's. To alter the address of the base page fence, enter the new address on the Display Register bit switches 10 - 0 and press the STORE switch. Status bits 15 - 11 cannot be altered. If Display Register bit switches 15 - 11 are changed from the original display and the STORE switch is pressed, no action takes place and the original contents of the f-register is again displayed.

2-26. STATUS (s) REGISTER. The s-register is a 16-bit status register. When selected for display, bit 15 indicates the status of the overflow register. (Refer to paragraph 2-9.) Bit 14 indicates the status of the extend register. (Refer to paragraph 2-8.) Bit 13 indicates the status of the interrupt system; a logic 1 indicates that the system is enabled. Bits 12 - 6 are always zero. Bits 5 - 0 indicate the current contents of the Central Interrupt Register (CIR) which is the octal select code of the device that last interrupted the computer. When the s-register is displayed, the overflow register bit, extend register bit, and interrupt system can be set as desired with Display Register bit switches 15, 14, and 13 respectively, and then pressing the STORE switch. The contents of CIR cannot be altered.

Bit 13 of the s-register will not display the status of the interrupt system correctly and cannot be changed if memory protect is enabled.

2-27. SHUTDOWN PROCEDURES

One of the following procedures should be used when the computer is shut down during periods of nonoperation. The first procedure should be used when it is necessary to sustain memory contents. The second procedure should be used when it is not necessary to sustain memory contents.

2-28. SHUTDOWN (MEMORY SUSTAINED). Use the following procedure to shut down the computer when it is necessary to sustain memory during periods of nonoperation:

a. The computer MUST BE equipped with the optional Power Fail Recovery system, or memory will be lost when power is removed from the computer.

b. Ensure that line (mains) power is available and that the BATTERY switch is set to INT and that the computer ~POWER ON/OFF switch is set to OFF. If the computer is housed in a system cabinet, ensure that the system power switch is set to provide power to rack-mounted units.

In the event of a power failure, the contents of memory will be sustained for a minimum of 1.6 hours by the Power Fail Recovery System.

2-29. SHUTDOWN (MEMORY NOT SUSTAINED). Use the following procedure to shut down the computer when it is not necessary to sustain memory during periods of nonoperation:

a. Set ~POWER OFF/ON switch to OFF. If computer is equipped with optional power fail recovery system, set BATTERY switch to OFF to prevent the battery from discharging.

b. Set computer ~POWER OFF/ON switch to OFF or, if the computer is housed in a system cabinet, set the system power switch to remove ac power.

All contents of memory and internal registers will be lost. When operation is to be resumed, the cold power-up procedure and program loading must be repeated.

2-30. REMOTE PROGRAM LOADING

The optional Remote Program Load (RPL) capability is a combination of hardware and firmware that allows loading and initiating execution of program to be controlled from a remote site. The hardware consists of a set of eight switches called the configuration switches mounted on the CPU board and typically, an appropriate communications interface board. (The HP 12966A or HP 12968A must be used if an HP standard interface is utilized.) The configuration switches are used to program the information normally entered into the S-register from the operator panel for Initial Binary Loader (IBL) operations. (Refer to the *HP 1000 F-Series Computer Installation and Service manual,* part no 02111-90002 for switch programming details.) The communications interface board provides the link through which remote devices initiate RPL operations. The firmware is a micro-routine that automatically reads the information from the configuration switches, calls the IBL routine specified by the configuration switches, and issues the RUN command from a remote computer site, a console device, or automatically when line power is applied to the computer.

The following conditions will cause execution of the RPL micro-routine if RPL is enabled and the LOCK/OPERATE switch is in the LOCK position:

a. Cold Power-up;

b. Programmed halt instructions (106XXX, 107XXX); or

c. Forced halts initiated via the I/O system. (This allows system consoles via their interface or remote systems via data communication links to initiate the RPL sequence.)

The following conditions will not cause execution of the RPL micro-routine, even though RPL is enabled:

a. LOCK/OPERATE switch is in the OPERATE position;

b. Auto-restart is enabled (A1S2 set to ARS), power coming up, and PRESET has not been pressed;

c. Programmed halt instructions (102XXX, 103XXX); or

d. Parity halt (A1S1 set to HALT).

A flow chart of the complete RPL sequence is shown in figure 2-4.

2-31. SELF-TEST FIRMWARE

The standard microprogrammed base set includes three tests that are designed to conveniently and quickly test the computer and memory without supplementary diagnostics. These firmware self-tests are not designed as a substitute for more complex software diagnostics and it may frequently be the case that you require a more thorough and detailed testing than provided by these standard self-test routines. Firmware test for the scientific instruction set (SIS) and the floating point processor (FFP) can be executed via the operator panel.

2-32. TEST DESCRIPTIONS

2-33. **TEST 1.** Test 1 tests most of the computer registers and functions. This test will not alter or destroy the contents of any working register or memory. An error condition will set all display registers, indicator bits, and the overflow register. The execution time is negligible.

2-34. **TEST 2.** Test 2 is a fast microprogrammed memory test that checks the presently enabled memory space (up to 64k bytes). The microprogram reads each memory location, complements the data and writes it back, it reads it, compares it to expected data, then complements it and writes it back into memory. The execution time is negligible and is non-destructive to memory data. An error condition is usually accompanied by a parity error indication and will set all display registers and indicator bits and clear the overflow register. The A-register will contain the expected (good) data, the B-register will contain the actual (bad) data, and the M-register will contain the logical memory location of the failure. If all display indicators and display registers do not light, but the PARITY indicator on the operator panel is lit, a check bit array or a parity bit on a memory module has failed.

2-35. **TEST 3.** Test 3 is a significantly more sophisticated microprogrammed memory test. All memory installed in the computer will be tested. Execution time is dependent on the amount of memory installed; approximately one second per 64k bytes. The display register will increment as memory in each 64k byte space is tested. Error reporting is the same as in Test 2 except the S-register will contain the number of the 64k byte space where the memory failure occurred. (Refer to the *HP 1000 F-Series Computer Installation and Service Manual,* part no. 02111-90002 for additional information.)

2-36. TEST EXECUTION

On a cold power-up (paragraph 2-16), Tests 1 and 3 will each be executed once.

Pressing the IBL-TEST switch on the operator panel will not only perform the loader function as described in paragraphs 2-18 and 2-19, it will also cause the execution of Tests 1 and 2.

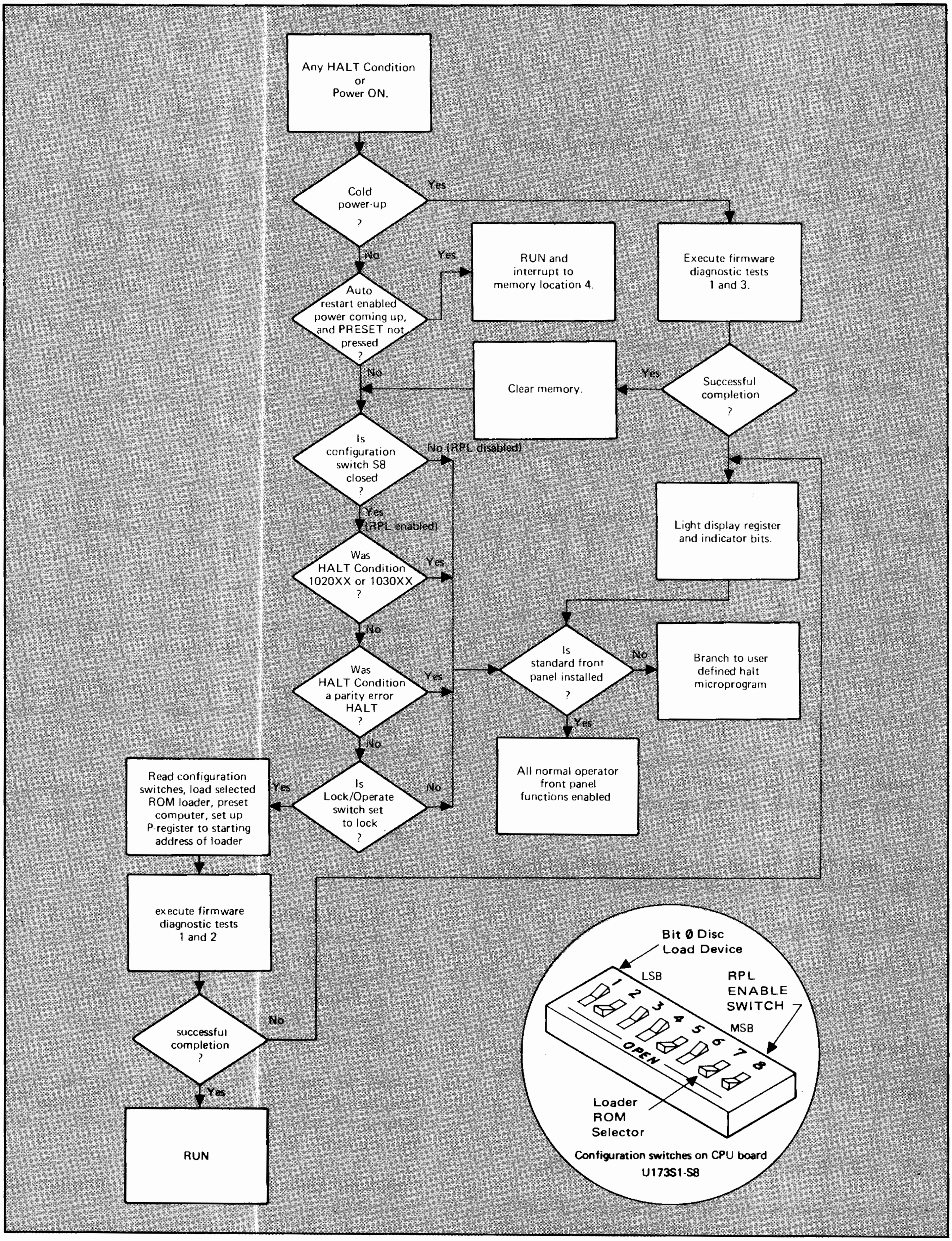


Figure 2-4. Remote Program Load Routine

Tests 1 and 3 may also be executed via the operator panel as follows:

a. Set P=0.

b. Set A = 100000 (octal).

c. At the end of the diagnostic (when executed once), memory will be filled with the value in the S-register. Set the S-register if desired.

d. Press PRESET.

e. To loop the diagnostic, set the LOCK/OPERATE switch to LOCK.

f. Press INSTR STEP, the diagnostic will now be executed.

g. While the diagnostic is being executed, if S-register bit 5 is set, the S-register will display the number of executions (bit 5 = LSB).

If the LOCK/OPERATE switch is set to LOCK position, the microprogrammed diagnostic will continuously loop.

2-37. FPP TEST. To execute the Floating Point Processor (FPP) test proceed as follows:

a. Store 105004 (octal) in the A-register.

b. Store 0 in the P-register and press PRESET. If the OVFL light remains on, refer to the F-Series computer installation and service manual for troubleshooting information.

c. Press INSTR STEP.

d. A 102077 (octal) in the display register indicates successful completion; otherwise, refer to the F-Series computer installation and service manual.

2-38. SIS TEST. To execute the Scientific Instruction Set (SIS) firmware test, proceed as follows:

a. Store 105337 (octal) in the A-register.

b. Store 0 in the P-register.

c. Press PRESET; then press INSTR STEP.

A 102077 (octal) in the display register indicates successful completion; otherwise, refer to the F-Series computer installation and service manual.

2-39. EXCHANGING I/O INTERFACES

Figures 2-2 and 2-3 show the HP 2111F and the HP 2117F I/O PCA cages and the select codes associated with each slot. Select code 10 (octal) has the highest priority in the interrupt structure and the highest numbered select code has the lowest priority. When it becomes necessary to install a new I/O interface PCA or change the location of an existing one, proceed as follows:

CAUTION

If the computer is not equipped with an optional power fail recovery system, the contents of memory will be lost when the line (mains) voltages are off. Therefore, store any contents of memory to be saved in another medium for later retrieval; then perform steps c and e through j below.

a. Check that the ~POWER switch is set to ON.

NOTE

If the computer is equipped with a power fail recovery system, ensure that there is a place to support the battery box with it connected to the computer.

b. Set the BATTERY switch to INT.

c. Loosen the four captive screws securing the I/O cage cover to the computer rear frame.

d. Place the I/O cage cover with the battery box on a support so that the battery cable remains connected.

e. Set the ~POWER switch to OFF.

f. Loosen the two screws holding the I/O PCA retainer and slide the retainer to permit the removal or installation of the I/O PCA's.

NOTE

If the I/O PCA associated with RPL is relocated, then the internal RPL configuration switches must be reset. Details on how to configure these switches are given in the HP 1000 F-Series Computer Installation and Service Manual (02111-90002).

g. Install the new I/O interface PCA or exchange I/O interface PCA's as required. If an HP 12979B I/O Extender is to be used, install its interface PCA in the first available lowest priority I/O slot.

h. Slide the I/O PCA retainer to the left and tighten the two screws.

i. Apply power to the computer by setting the ~POWER switch to ON.

j. Secure the I/O cage cover with the battery box to the rear frame of the computer by fastening the four captive screws.

2-40. HALT CODES

Table 2-6 provides a quick reference to those halt codes associated with the input device loader. These halt codes are displayed in the Display Register, when the computer is in the halt mode.

2-41. ABNORMAL INDICATIONS

Table 2-7 provides a quick reference to the operator panel indications that occur when an abnormal condition exists during operation in the normal register display mode. Abnormal indications encountered during the execution of the firmware tests are discussed in paragraphs 2-31 through 2-38.

Table 2-6. Halt Codes

HALT CODE (in octal)	COMMENTS
102077	Indicates a successful program load from paper tape and typically indicates a successful program load from disc, magnetic tape, or other such media.
102055	Indicates that an address error occurred while loading from input device.
102011	Indicates that a checksum error occurred while loading from input device.

Table 2-7. Abnormal Indications

INDICATION	ABNORMAL CONDITION	REMEDY
POWER FAIL light remains on.	Indicates that power has been restored after a power failure.	Press HALT; then PRESET or execute an STC 04 or CLC 04 instruction.
PARITY light is on.	Indicates that a parity error occurred while reading from memory.	Refer to *HP 1000F-Series Installation and Service manual,* part no. 02111-90002.
OVERFLOW light is on after IBL/TEST is pressed.	Indicates that: a. The presence of memory was not detected. b. The programmed select code was less than 10 (octal). c. The memory was defective.	 a. Check that memory modules are installed and programmed correctly. b. Check that the programmed select code is within range. c. Refer to *HP 1000 F-Series Installation and Service Manual,* part no. 02111-90002.
Operator panel indicators are irregular after cold power-up; Register Select switch cannot select registers.	Indicates that: a. Power supply or CPU PCA is defective. b. If Power Fail Recovery System is installed, battery cable is not connected or connector 12991-60002 is not connected to BAT INPUT connector.	 a. Refer to *HP 1000 F-Series Computer Installation and Service Manual,* part no. 02111-90002. b. Connect battery cable from battery box (or connect connector 12991-60002) to BAT INPUT connector.

PROGRAMMING INFORMATION

SECTION III

This section describes the software data formats and the base set machine-language instruction coding required to operate the computer and its associated input/output system. This section also describes the Scientific Instruction Set instructions and the Fast FORTRAN Processor instructions. Machine-language instruction coding for the Dynamic Mapping System is presented in Section VI.

3-1. DATA FORMATS

As shown in figure 3-1, the basic data format is a 16-bit word in which bit positions are numbered from 0 through 15 in order of increasing significance. Bit position 15 of the data format is used for the sign bit; a logic 0 in this position indicates a positive number and a logic 1 in this position indicates a negative number. The data is assumed to be a whole number and the binary point is therefore assumed to be to the right of the number.

The basic word can also be divided into two 8-bit bytes or combined to form a 32-bit double integer. The byte format is used for character-oriented input/output devices; packing two bytes of data into one 16-bit word is accomplished by software drivers. In I/O operations, the higher-order byte (byte 1) is the first to be transferred.

The double integer format is used for extended arithmetic in conjunction with the extended arithmetic instructions described under paragraphs 3-21 and 3-22. Bit position 15 of the most-significant word is the sign bit and the binary point is assumed to be to the right of the least-significant word. The integer value is expressed by the remaining 31 bits. When loaded into the accumulators, the B-register contains the most-significant word and the A-register contains the least-significant word (unless stated otherwise for a specific instruction).

The two floating point formats in figure 3-1 are used with floating point software. Bit position 15 of the most-significant word is the mantissa sign and bit position 0 of the least-significant word is the exponent sign. Bits 1 through 7 of the least-significant word express the exponent and the remaining bits express the mantissa. A single precision floating point number is made up of a 23-bit mantissa (fraction) and sign and a 7-bit exponent and sign, thus providing six significant decimal digits in the mantissa. An extended precision floating point number is made up of a 39-bit mantissa and a 7-bit exponent and sign, thus providing 11 significant decimal digits in the mantissa. If either the mantissa or the exponent is negative, that part must be stored in two's complement form. The number must be in the approximate range of 10^{-38} to 10^{+38}. When loaded into the accumulators, the A-register contains the most-significant word and the B-register contains the least-significant word.

Figure 3-1 also illustrates the octal notation for both single-length (16-bit) and double-length (32-bit) words. Each group of three bits, beginning at the right, is combined to form an octal digit. A single-length (16-bit) word can therefore be fully expressed by six octal digits and a double-length (32-bit) word can be fully expressed by 11 octal digits. Octal notation is not shown for byte or floating point formats, since bytes normally represent characters and floating point numbers are given in decimal form.

The range of representable numbers for single integer data is +32,767 to −32,768 (decimal) or +77,777 to −100,000 (octal). The range of representable numbers for double integer data is +2,147,483,647 to −2,147,483,648 (decimal) or +17,777,777,777 to −20,000,000,000 (octal).

3-2. ADDRESSING

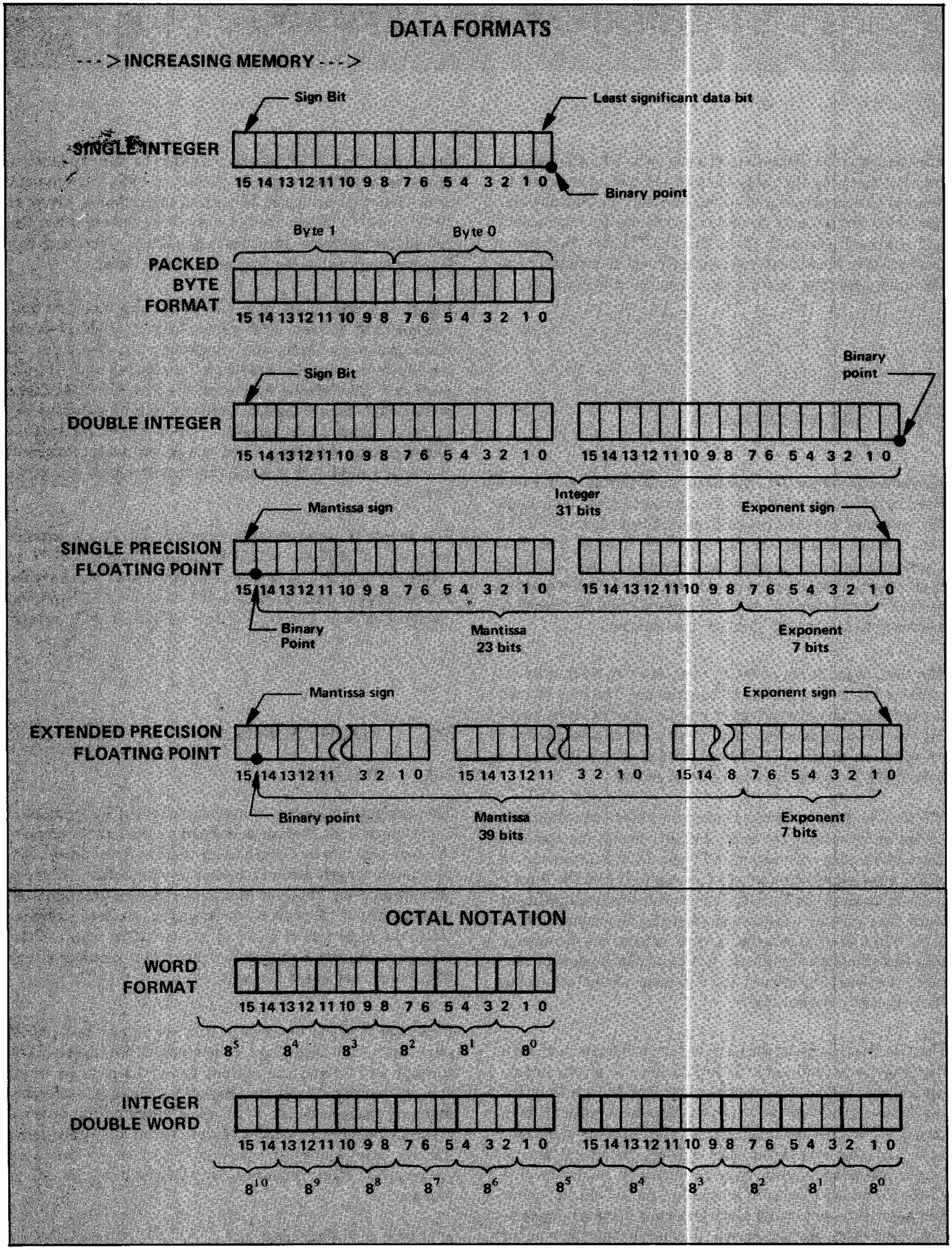
3-3. PAGING

The computer memory is logically divided into pages of 1,024 words each. A page is defined as the largest block of memory that can be directly addressed by the address bits of a single-length memory reference instruction. (Refer to paragraph 3-8.) These memory reference instructions use 10 bits (bits 0 through 9) to specify a memory address; thus, the page size is 1,024 locations (2000 octal). Octal addresses for each page, up to a maximum memory size of 32k words, are listed in table 3-1.

Provision is made to directly address one of two pages: page zero (the base page consisting of locations 00000 through 01777) and the current page (the page in which the instruction itself is located). Memory reference instructions reserve bit 10 to specify one or the other of these two pages. To address locations on any other page, indirect addressing is used as described in following paragraphs. Page references are specified by bit 10 as follows:

a. Logic 0 = Page Zero (Z).

b. Logic 1 = Current Page (C).



2270-2

Figure 3-1. Data Formats and Octal Notation

Table 3-1. Memory Paging

MEMORY SIZE	PAGE	OCTAL ADDRESSES
	0	00000 to 01777
	1	02000 to 03777
	2	04000 to 05777
	3	06000 to 07777
	4	10000 to 11777
	5	12000 to 13777
	6	14000 to 15777
	7	16000 to 17777
	8	20000 to 21777
	9	22000 to 23777
	10	24000 to 25777
	11	26000 to 27777
	12	30000 to 31777
	13	32000 to 33777
	14	34000 to 35777
16K ↓	15	36000 to 37777
	16	40000 to 41777
	17	42000 to 43777
	18	44000 to 45777
	19	46000 to 47777
	20	50000 to 51777
	21	52000 to 53777
	22	54000 to 55777
	23	56000 to 57777
	24	60000 to 61777
	25	62000 to 63777
	26	64000 to 65777
	27	66000 to 67777
	28	70000 to 71777
	29	72000 to 73777
	30	74000 to 75777
	31	76000 to 77777

3-4. DIRECT AND INDIRECT ADDRESSING

All memory reference instructions reserve bit 15 to specify either direct or indirect addressing. For single-length memory reference instructions, bit 15 of the instruction word is used; for extended arithmetic memory reference instructions, bit 15 of the address word is used. Indirect addressing uses the address part of the instruction to access another word in memory, which is taken as the new memory reference for the same instruction. This new address word is a full 16 bits long: 15 address bits plus another direct/indirect bit. The 15-bit length of the address permits access to any location in memory. If bit 15 again specifies indirect addressing, still another address is obtained; thus, multistep indirect addressing may be done to any number of levels. The first address obtained that does not specify another indirect level becomes the effective address for the instruction. Direct or indirect addressing is specified by bit 15 as follows:

a. Logic 0 = Direct (D).

b. Logic 1 = Indirect (I).

3-5. RESERVED MEMORY LOCATIONS

The first 64 memory locations of the base page (octal addresses 00000 through 00077) are reserved as listed in table 3-2. The first two locations are reserved as addresses for the two 16-bit accumulators (the A- and B-registers). If options or input/output devices corresponding to locations 00005 through 00077 are not included in the system configuration, these locations can be used for programming purposes.

The uppermost 64 locations of memory for any given configuration are reserved for the initial binary loader. The initial binary loader is permanently resident in a read-only memory (ROM) and loaded into the uppermost 64 memory locations by a pushbutton switch on the operator panel. These 64 locations are not protected and can therefore be used for temporary storage of data, trap cells, buffers, etc.

Table 3-2. Reserved Memory Locations

MEMORY LOCATION	PURPOSE
00000	A-register address.
00001	B-register address.
00002-00003	Exit sequence if contents of A-register and B-register are used as executable words.
00004	Power-fail interrupt (highest priority).
00005	Memory parity, memory protect, and DMS interrupt.
00006	Reserved for dual-channel port controller (DCPC) channel 1.
00007	Reserved for dual-channel port controller (DCPC) channel 2.
00010-00077	Interrupt locations in decreasing order of priority; e.g., location 00010 has priority over 00011.

3-6. NONEXISTENT MEMORY

Nonexistent memory is defined as those locations not physically implemented in the machine. Any attempt to write into a nonexistent memory location will be ignored (no operation). Any attempt to read from a nonexistent memory location will return an all-zeros word (000000 octal); no parity error occurs.

3-7. BASE SET INSTRUCTION FORMATS

The base set of instructions are classified according to format. The five formats used are illustrated in figure 3-2 and described in following paragraphs. In all cases where a single bit is used to select one of two cases (e.g., D/I), the choice is made by coding a logic 0 or logic 1, respectively.

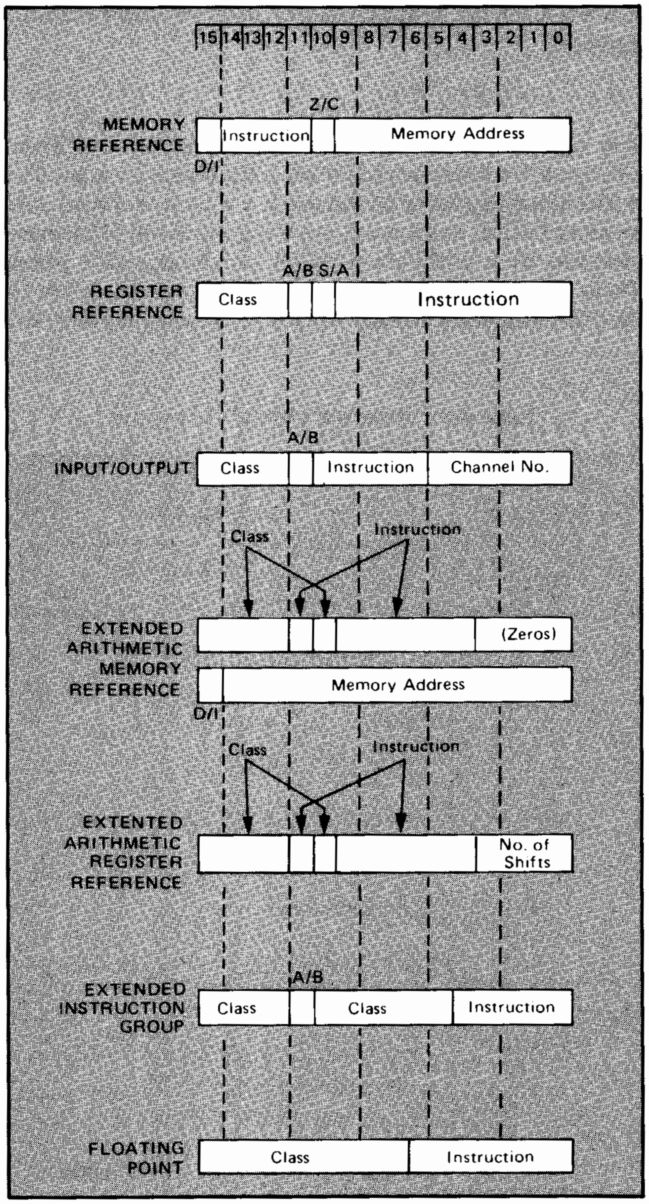


Figure 3-2. Base Set Instruction Formats

3-8. MEMORY REFERENCE INSTRUCTIONS

This class of instructions, which combines an instruction code and a memory address into one 16-bit word, is used to execute some function involving data in a specific memory location. Examples are storing, retrieving, and combining memory data to and from the accumulators (A- and B-registers) or causing the program to jump to a specified location in memory.

The memory cell referenced (i.e., the absolute address) is determined by a combination of 10 memory address bits (0 through 9) in the instruction word and 5 bits (10 through 14) assumed from the current contents of the M-register. This means that memory reference instructions can directly address any word in the current page; additionally, if the instruction is given in some location other than the base page (page zero), bit 10 (Z/C) of the instruction doubles the addressing range to 2,048 locations by allowing the selection of either page zero or the current page. (This causes bits 10 through 14 of the address contained in the M-register to be set to zero instead of assuming the current contents of the M-register.) This feature provides a convenient linkage between all pages of memory, since page zero can be reached directly from any other page.

As discussed under paragraph 3-4, bit 15 is used to specify direct or indirect memory addressing. Note also that since the A- and B-registers are addressable, any single-word memory reference instruction can apply to either of these registers as well as to memory cells. For example, an ADA 0001 instruction adds the contents of the B-register (address 0001) to the contents currently held in the A-register; specify page zero for these operations since the addresses of the A- and B-registers are on page zero.

3-9. REGISTER REFERENCE INSTRUCTIONS

In general, the register reference instructions manipulate bits in the A-register, B-register, and E-register; there is no reference to memory. This group includes 39 basic instructions which may be combined to form a one-word multiple instruction that can operate in various ways on the contents of the A-, B-, and E-registers. These 39 instructions are divided into two subgroups: the shift/rotate group (SRG) and the alter/skip group (ASG). The appropriate subgroup is specified by bit 10 (S/A). Typical operations are clear and/or complement a register, conditional skips, and register increment.

3-10. INPUT/OUTPUT INSTRUCTIONS

The input/output instructions use bits 6 through 11 for a variety of I/O instructions and bits 0 through 5 to apply the instructions to a specific I/O channel. This provides the means of controlling all peripherals connected to the I/O channels and for transferring data to and from these peripherals. Included also in this group are instructions to control the interrupt system, overflow bit, and computer halt.

3-11. EXTENDED ARITHMETIC MEMORY REFERENCE INSTRUCTIONS

As the single-word memory reference instruction described previously, the extended arithmetic memory reference instructions include an instruction code and a memory address. In this case, however, two words are required. The first word specifies the extended arithmetic class (bits 12 through 15 and 10) and the instruction code (bits 4 through 9 and 11); bits 0 through 3 are not needed and are coded with zeros. The second word specifies the memory address of the operand. Since the full 15 bits are used for the address, this type of instruction may directly address any location in memory. As with all memory reference instructions, bit 15 is used to specify direct or indirect addressing. Operations performed by this class of instructions are integer multiply and divide (using double-length product and dividend) and double load and double store.

3-12. EXTENDED ARITHMETIC REGISTER REFERENCE INSTRUCTIONS

This class of instructions provides long shifts and rotates on the combined contents of the A- and B-registers. Bits 12 through 15 and 10 identify the instruction class; bits 4 through 9 and 11 specify the direction and type of shift; and bits 0 through 3 control the number of shifts, which can range from 1 to 16 places.

3-13. EXTENDED INSTRUCTIONS

The extended instructions include index register instructions, bit and byte manipulation instructions, and move and compare instructions. Instructions comprising the extended instruction group are one, two, or three words in length. The first word is always the instruction code; operand addresses are given in the words following the instruction code or in the A- and B-registers. The operand addresses are 15 bits long, with bit 15 (most-significant bit) generally indicating direct or indirect addressing.

3-14. FLOATING POINT INSTRUCTIONS

The floating point instructions allow addition, subtraction, multiplication, and division of floating point quantities. Conversion routines are provided for transforming numerical integer representations to/from floating point representations.

3-15. BASE SET INSTRUCTION CODING

Machine language coding for the base set of instructions are provided in following paragraphs. Definitions for these instructions are grouped according to the instruction type: memory reference, register reference, input/output, extended arithmetic memory reference, and extended arithmetic register reference.

Directly above each definition is a diagram showing the machine language coding for that instruction. The gray shaded bits code the instruction type and the rose shaded bits code the specific instruction. Unshaded bits are further defined in the introduction to each instruction type. The mnemonic code and instruction name are included above each diagram.

In all cases where an additional bit is used to specify a secondary function (D/I, Z/C, or H/C), the choice is made by coding a logic 0 or logic 1, respectively. In other words, a logic 0 codes D (direct addressing), Z (zero page), or H (hold flag); a logic 1 codes I (indirect addressing), C (current page), or C (clear flag).

3-16. MEMORY REFERENCE INSTRUCTIONS

The following 14 memory reference instructions execute a function involving data in memory. Bits 0 through 9 specify the affected memory location on a given memory page or, if indirect addressing is specified, the next address to be referenced. Indirect addressing may be continued to any number of levels; when bit 15 (D/I) is a logic 0 (specifying direct addressing), that location will be taken as the effective address. The A- and B-registers may be addressed as locations 00000 and 00001 (octal), respectively.

If bit 10 (Z/C) is a logic 0, the memory address is on page zero; if bit 10 is a logic 1, the memory address is on the current page. If the A- or B-register is addressed, bit 10 must be a logic 0 to specify page zero, unless the current page is page zero.

ADA ADD TO A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	1	0	0	0	Z/C										

$\underbrace{\qquad\qquad\qquad\qquad}_{\text{Memory Address}}$

Adds the contents of the addressed memory location to the contents of the A-register. The sum remains in the A-register and the contents of the memory cell are unaltered. The result of this addition may set the extend bit or the overflow bit. (Extend and overflow examples are illustrated on page A-13.)

ADB ADD TO B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	1	0	0	1	Z/C										

$\underbrace{\qquad\qquad\qquad\qquad}_{\text{Memory Address}}$

Adds the contents of the addressed memory location to the contents of the B-register. The sum remains in the B-register and the contents of the memory cell are unaltered. The result of this addition may set the extend bit or the overflow bit. (Extend and overflow examples are illustrated on page A-13.)

AND "AND" TO A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	0	0	1	0	Z/C										

Memory Address

Combines the contents of the addressed memory location and the contents of the A-register by performing a logical "and" operation. The contents of the memory cell are unaltered.

CPA COMPARE TO A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	1	0	1	0	Z/C										

Memory Address

Compares the contents of the addressed memory location with the contents of the A-register. If the two 16-bit words are not identical, the next instruction is skipped; i.e., the P-register advances two counts instead of one count. If the two words are identical, the next sequential instruction is executed. Neither the A-register contents nor memory cell contents are altered.

CPB COMPARE TO B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	1	0	1	1	Z/C										

Memory Address

Compares the contents of the addressed memory location with the contents of the B-register. If the two 16-bit words are not identical, the next instruction is skipped; i.e., the P-register advances two counts instead of one count. If the two words are identical, the next sequential instruction is executed. Neither the B-register contents nor memory cell contents are altered.

IOR "INCLUSIVE OR" TO A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	0	1	1	0	Z/C										

Memory Address

Combines the contents of the addressed memory location and the contents of the A-register by performing a logical "inclusive or" operation. The contents of the memory cell are unaltered.

ISZ INCREMENT AND SKIP IF ZERO

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	0	1	1	1	Z/C										

Memory Address

Adds one to the contents of the addressed memory location. If the result of this operation is zero (memory contents incremented from 177777 to 000000), the next instruction is skipped; i.e., the P-register is advanced two counts instead of one count. If the result of this operation is not zero, the next sequential instruction is executed. In either case, the incremented value is written back into the memory cell.

JMP JUMP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	0	1	0	1	Z/C										

Memory Address

Transfers control to the addressed memory location. That is, a JMP causes the P-register count to set according to the memory address portion of the JMP instruction so that the next instruction will be read from that location.

JSB JUMP TO SUBROUTINE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	0	0	1	1	Z/C										

Memory Address

This instruction, executed in location P (P-register count), causes the computer control to jump unconditionally to the memory location (m) specified by the memory address portion of the JSB instruction. The contents of the P-register plus one (return address) is stored in memory location m, and the next instruction to be executed will be that contained in the next sequential memory location (m + 1). A return to the main program sequence at P + 1 will be effected by a JMP indirect through location m.

LDA LOAD A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	1	1	0	0	Z/C										

Memory Address

Loads the contents of the addressed memory location into the A-register. The contents of the memory cell are unaltered.

LDB LOAD B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	1	1	0	1	Z/C										

Memory Address

Loads the contents of the addressed memory location into the B-register. The contents of the memory cell are unaltered.

STA STORE A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	1	1	1	0	Z/C										

Memory Address

Stores the contents of the A-register in the addressed memory location. The previous contents of the memory cell are lost; the A-register contents are unaltered.

STB STORE B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	1	1	1	1	Z/C										

Memory Address

Stores the contents of the B-register in the addressed memory location. The previous contents of the memory cell are lost; the B-register contents are unaltered.

XOR "EXCLUSIVE OR" TO A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	0	1	0	0	Z/C										

Memory Address

Combines the contents of the addressed memory location and the contents of the A-register by performing a logical "exclusive or" operation. The contents of the memory cell are unaltered.

3-17. REGISTER REFERENCE INSTRUCTIONS

The 39 register reference instructions execute functions on data contained in the A-register, B-register, and E-register. These instructions are divided into two groups: the shift/rotate group (SRG) and the alter/skip group (ASG). In each group, several instructions may be combined into one word. Since the two groups perform separate and distinct functions, instructions from the two groups cannot be mixed. Unshaded bits in the coding diagrams are available for combining other instructions.

3-18. SHIFT/ROTATE GROUP. The 20 instructions in the shift/rotate group (SRG) are defined first; this group is specified by setting bit 10 to a logic 0. A comparison of the various shift/rotate functions are illustrated in figure 3-3. Rules for combining instructions in this group are as follows (refer to table 3-3):

a. Only one instruction can be chosen from each of the two multiple-choice columns.

b. References can be made to either the A-register or B-register, but not both.

c. Sequence of execution is from left to right.

d. In machine code, use zeros to exclude unwanted microinstructions.

e. Code a logic 1 in bit position 9 to enable shifts or rotates in the first position; code a logic 1 in bit position 4 to enable shifts or rotates in the second position.

f. The extend bit is not affected unless specifically stated. However, if a "rotate-with-E" instruction (ELA, ELB, ERA, or ERB) is coded but disabled by a logic 0 in bit position 9 and/or position 4, the E-register will be updated even though the A- or B-register contents are not affected; to avoid this situation, code a "no operation" (three zeros) in the first and/or second positions.

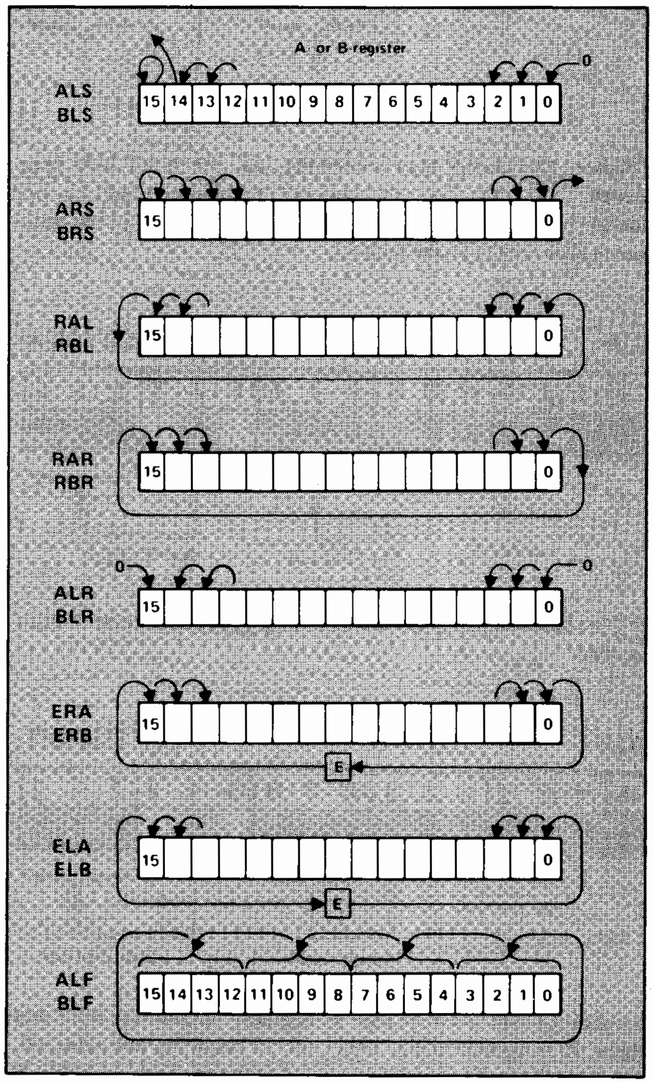
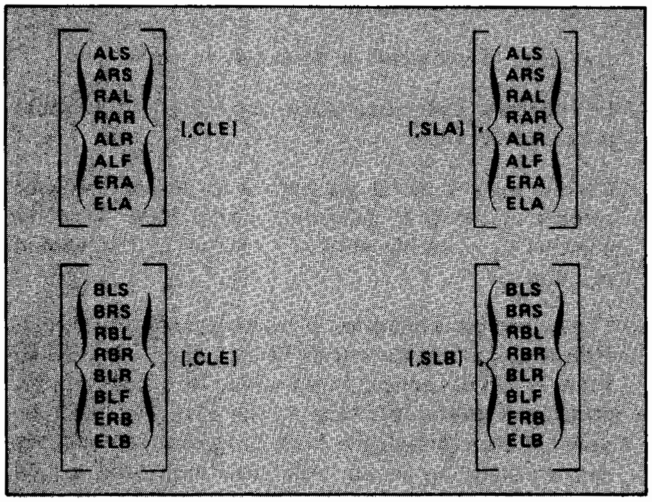


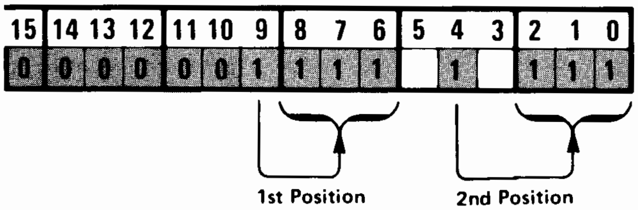
Figure 3-3. Shift and Rotate Functions

Table 3-3. Shift/Rotate Group Combining Guide



ALS			ALS
ARS			ARS
RAL			RAL
RAR	[,CLE]	[,SLA]	RAR
ALR			ALR
ALF			ALF
ERA			ERA
ELA			ELA
BLS			BLS
BRS			BRS
RBL			RBL
RBR	[,CLE]	[,SLB]	RBR
BLR			BLR
BLF			BLF
ERB			ERB
ELB			ELB

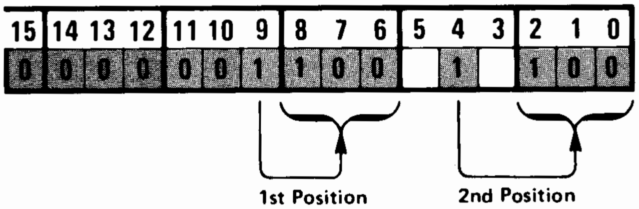
ALF ROTATE A LEFT FOUR



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

Rotates the A-register contents (all 16 bits) left four places. Bits 15, 14, 13, and 12 rotate around to bit positions 3, 2, 1, and 0, respectively. Equivalent to four successive RAL instructions.

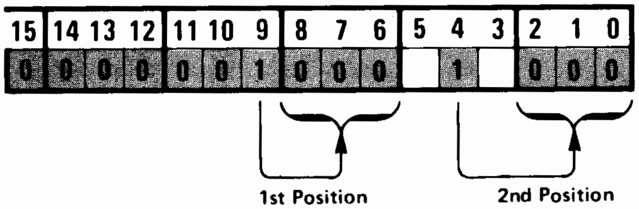
ALR A LEFT SHIFT, CLEAR SIGN



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

Shifts the A-register contents left one place and clears sign bit 15.

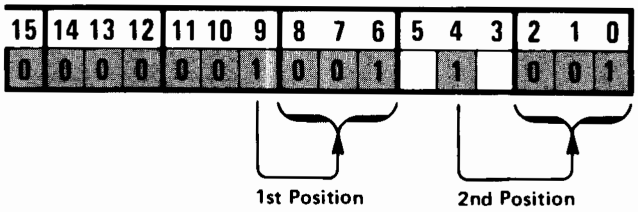
ALS A LEFT SHIFT



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

Arithmetically shifts the A-register contents left one place, 15 magnitude bits only; bit 15 (sign) is not affected. The bit shifted out of bit position 14 is lost; a logic 0 replaces vacated bit position 0.

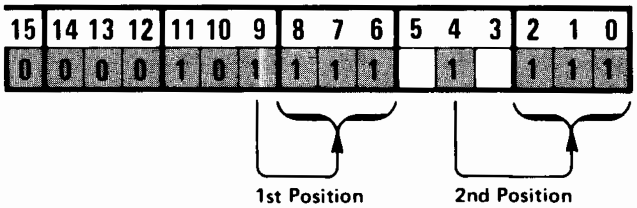
ARS A RIGHT SHIFT



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

Arithmetically shifts the A-register contents right one place, 15 magnitude bits only; bit 15 (sign) is not affected. A copy of the sign bit is shifted into bit position 14; the bit shifted out of bit position 0 is lost.

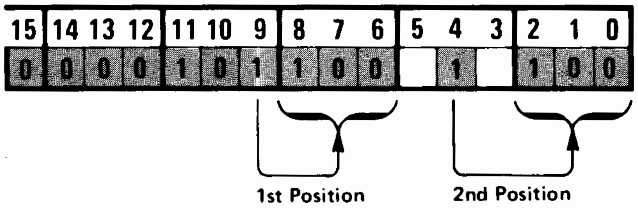
BLF ROTATE B LEFT FOUR



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

Rotates the B-register contents (all 16 bits) left four places. Bits 15, 14, 13, and 12 rotate around to bit positions 3, 2, 1, and 0, respectively. Equivalent to four successive RBL instructions.

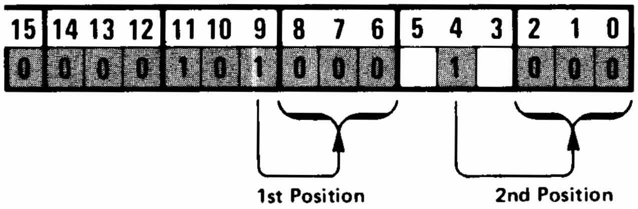
BLR B LEFT SHIFT, CLEAR SIGN



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

Shifts the B-register contents left one place and clears sign bit 15.

BLS B LEFT SHIFT



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

Arithmetically shifts the B-register contents left one place, 15 magnitude bits only; bit 15 (sign) is not affected. The bit shifted out of bit position 14 is lost; a logic 0 replaces vacated bit position 0.

BRS — B RIGHT SHIFT

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

1st Position 2nd Position

Arithmetically shifts the B-register contents right one place, 15 magnitude bits only; bit 15 (sign) is not affected. A copy of the sign bit is shifted into bit position 14; the bit shifted out of bit position 0 is lost.

CLE — CLEAR E

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

Clears the E-register; i.e., the extend bit becomes a logic 0.

ELA — ROTATE E LEFT WITH A

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

1st Position 2nd Position

Rotates the E-register content left with the A-register contents (one place). The E-register content rotates into bit position 0; bit 15 rotates into the E-register.

ELB — ROTATE E LEFT WITH B

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

1st Position 2nd Position

Rotates the E-register content left with the B-register contents (one place). The E-register content rotates into bit position 0; bit 15 rotates into the E-register.

ERA — ROTATE E RIGHT WITH A

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

1st Position 2nd Position

Rotates the E-register content right with the A-register contents (one place). The E-register content rotates into bit position 15; bit 0 rotates into the E-register.

ERB — ROTATE E RIGHT WITH B

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

1st Position 2nd Position

Rotates the E-register content right with the B-register contents (one place). The E-register content rotates into bit position 15; bit 0 rotates into the E-register.

NOP — NO OPERATION

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

This all-zeros instruction causes a no-operation cycle.

RAL — ROTATE A LEFT

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

1st Position 2nd Position

Rotates the A-register contents left one place (all 16 bits). Bit 15 rotates into bit position 0.

RAR — ROTATE A RIGHT

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

1st Position 2nd Position

Rotates the A-register contents right one place (all 16 bits). Bit 0 rotates into bit position 15.

RBL — ROTATE B LEFT

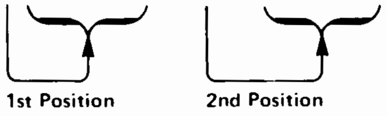
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	1	0		1		0	1	0

1st Position 2nd Position

Rotates the B-register contents left one place (all 16 bits). Bit 15 rotates into bit position 0.

RBR ROTATE B RIGHT

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



Rotates the B-register contents right one place (all 16 bits). Bit 0 rotates into bit position 15.

SLA SKIP IF LSB OF A IS ZERO

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

Skips the next instruction if the least-significant bit (bit 0) of the A-register is a logic 0.

SLB SKIP IF LSB OF B IS ZERO

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

Skips the next instruction if the least-significant bit (bit 0) of the B-register is a logic 0.

3-19. ALTER/SKIP GROUP. The 19 instructions comprising the alter/skip group (ASG) are defined next. This group is specified by setting bit 10 to a logic 1. Rules for combining instructions are as follows (refer to table 3-4):

a. Only one instruction can be chosen from each of the two multiple-choice columns.

b. References can be made to either the A-register or B-register, but not both.

c. Sequence of execution is from left to right.

d. If two or more skip functions are combined, the skip function will occur if either or both conditions are met. One exception exists: refer to the RSS instruction.

e. In machine code, use zeros to exclude unwanted instructions.

Table 3-4. Alter/Skip Group Combining Guide

[CLA CMA CCA]	[,SEZ]	[,CLE ,CME ,CCE]	[,SSA] [,SLA] [,INA] [,SZA] [,RSS]
[CLB CMB CCB]	[,SEZ]	[,CLE ,CME ,CCE]	[,SSB] [,SLB] [,INB] [,SZB] [,RSS]

CCA CLEAR AND COMPLEMENT A

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

Clears and complements the A-register contents; i.e., the contents of the A-register become 177777 (octal). This is the two's complement form of -1.

CCB CLEAR AND COMPLEMENT B

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

Clears and complements the B-register contents; i.e., the contents of the B-register become 177777 (octal). This is the two's complement form of -1.

CCE CLEAR AND COMPLEMENT E

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

Clears and complements the E-register content (extend bit); i.e., the extend bit becomes a logic 1.

CLA CLEAR A

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

Clears the A-register; i.e., the contents of the A-register become 000000 (octal).

CLB CLEAR B

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

Clears the B-register; i.e., the contents of the B-register become 000000 (octal).

CLE CLEAR E

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

Clears the E-register; i.e., the extend bit becomes a logic 0.

CMA COMPLEMENT A

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

Complements the A-register contents (one's complement).

CMB COMPLEMENT B

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

Complements the B-register contents (one's complement).

CME COMPLEMENT E

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

Complements the E-register content (extend bit).

INA INCREMENT A

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

Increments the A-register by one. The overflow bit will be set if an increment of the largest positive number (077777 octal) is made. The extend bit will be set if an all-ones word (177777 octal) is incremented.

INB INCREMENT B

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

Increments the B-register by one. The overflow bit will be set if an increment of the largest positive number (077777 octal) is made. The extend bit will be set if an all-ones word (177777 octal) is incremented.

RSS REVERSE SKIP SENSE

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

Skip occurs for any of the following skip instructions, if present, when the non-zero condition is met. An RSS without a skip instruction in the word causes an unconditional skip. If a word with RSS also includes both SSA and SLA (or SSB and SLB), bits 15 and 0 must both be logic 1's for a skip to occur; in all other cases, a skip occurs if one or more skip conditions are met.

SEZ SKIP IF E IS ZERO

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

Skips the next instruction if the E-register content (extend bit) is a logic 0.

SLA SKIP IF LSB OF A IS ZERO

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

Skips the next instruction if the least-significant bit (bit 0) of the A-register is a logic 0; i.e., skips if an even number is in the A-register.

SLB SKIP IF LSB OF B IS ZERO

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

Skips the next instruction if the least-significant bit (bit 0) of the B-register is a logic 0; i.e., skips if an even number is in the B-register.

SSA SKIP IF SIGN OF A IS ZERO

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

Skips the next instruction if the sign bit (bit 15) of the A-register is a logic 0; i.e., skips if a positive number is in the A-register.

SSB SKIP IF SIGN OF B IS ZERO

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

Skips the next instruction if the sign bit (bit 15) of the B-register is a logic 0; i.e., skips if a positive number is in the B-register.

SZA SKIP IF A IS ZERO

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

Skips the next instruction if the A-register contents are zero (16 zeros).

SZB SKIP IF B IS ZERO

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

Skips the next instruction if the B-register contents are zero (16 zeros).

3-20. INPUT/OUTPUT INSTRUCTIONS

The following input/output instructions provide the capability of setting or clearing the I/O flag and control bits, testing the state of the overflow and the I/O flag bits, and transferring data between specific I/O devices and the A- and B-registers. In addition, specific instructions in this group control the vectored priority interrupt system and can cause a programmed halt.

Bit 11, where relevant, specifies the A- or B-register or distinguishes between set control and clear control; otherwise, bit 11 may be a logic 0 or a logic 1 without affecting the instruction (although the assembler will assign zeros in this case). In those instructions where bit position 9 includes the letters H/C, the programmer has the choice of holding (logic 0) or clearing (logic 1) the device flag after executing the instruction. (Exception: the H/C bit associated with instructions SOC and SOS holds or clears the overflow bit instead of the device flag.) Bits 8, 7, and 6 specify the appropriate I/O instruction and bits 5 through 0 form a two-digit octal select code (address) to apply the instruction to one of up to 64 input/output devices or functions.

CLC CLEAR CONTROL

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

Select Code

Clears the control bit of the selected I/O channel or function. This turns off the specific device channel and prevents it from interrupting. A CLC 00 instruction clears all control bits from select code 06 upward, effectively turning off all I/O devices.

CLF CLEAR FLAG

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

Select Code

Clears the flag of the selected I/O channel or function. A CLF 00 instruction disables the interrupt system for all select codes except power fail (select code 04) and parity error (select code 05), which are always enabled; this does not affect the status of the individual channel flags.

CLO CLEAR OVERFLOW

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

Clears the overflow bit.

HLT HALT

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

Select Code

Halts the computer and holds or clears the flag of the selected I/O channel. The HLT instruction has the same effect as pressing the operator panel HALT pushbutton. The HLT instruction will be contained in the T-register, which is selected and displayed automatically when the computer halts. The P-register will contain the HLT location plus one.

LIA LOAD INTO A

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

Select Code

Loads the contents of the I/O buffer associated with the selected device into the A-register.

LIB LOAD INTO B

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

Select Code

Loads the contents of the I/O buffer associated with the selected device into the B-register.

MIA MERGE INTO A

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

Select Code

By executing a logical "inclusive or" function, merges the contents of the I/O buffer associated with the selected device into the A-register.

MIB MERGE INTO B

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

Select Code

By executing a logical "inclusive or" function, merges the contents of the I/O buffer associated with the selected device into the B-register.

OTA OUTPUT A

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

Select Code

Outputs the contents of the A-register to the I/O buffer associated with the selected device. If the I/O buffer is less than 16 bits in length, the least-significant bits of the A-register are normally loaded. (Some exceptions to this exist, depending on the type of output device.) The contents of the A-register are not altered.

OTB OUTPUT B

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

Select Code

Outputs the contents of the B-register to the I/O buffer associated with the selected device. If the I/O buffer is less than 16 bits in length, the least-significant bits of the B-register are normally loaded. (Some exceptions to this exist, depending on the type of output device.) The contents of the B-register are not altered.

SFC SKIP IF FLAG CLEAR

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

Select Code

Skips the next programmed instruction if the flag of the selected channel is clear (device busy).

SFS SKIP IF FLAG SET

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

Select Code

Skips the next programmed instruction if the flag of the selected channel is set (device ready). Used with "wait-for-flag" I/O programming, usually the interrupt system is off. If used with the interrupt system on, use a CLF instruction to clear the flag and eliminate the interrupt request.

SOC SKIP IF OVERFLOW CLEAR

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

Skips the next programmed instruction if the overflow bit is clear. Use the H/C bit (bit 9) to either hold or clear the overflow bit following the completion of this instruction (whether the skip is taken or not).

SOS SKIP IF OVERFLOW SET

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

Skips the next programmed instruction if the overflow bit is set. Use the H/C bit (bit 9) to either hold or clear the overflow bit following the completion of this instruction (whether the skip is taken or not).

STC SET CONTROL

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

Select Code

Sets the control bit of the selected I/O channel or function.

STF SET FLAG

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

Select Code

Sets the flag of the selected I/O channel or function. An STF 00 instruction enables the interrupt system for all select codes except power fail (select 04) which is always enabled and parity error (select code 05), which is selectively controllable.

STO SET OVERFLOW

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

Sets the overflow bit.

3-21. EXTENDED ARITHMETIC MEMORY REFERENCE INSTRUCTIONS

The four extended arithmetic memory reference instructions provide for integer multiply and divide and for loading and storing double-length words to and from the A- and B-registers. The complete instruction requires two words: one for the instruction code and one for the address. When stored in memory, the instruction word is the first to be fetched; the address word is in the next sequential location.

Since 15 bits are available for the address, these instructions can directly address any location in memory. As for all memory reference instructions, indirect addressing to any number of levels may also be used. A logic 0 in bit position 15 specifies direct addressing; a logic 1 specifies indirect addressing.

DIV DIVIDE

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

Memory Address

Divides a double-word integer in the combined B- and A-registers by a 16-bit integer in the addressed memory location. The result is a 16-bit integer quotient in the A-register and a 16-bit integer remainder in the B-register. Overflow can result from an attempt to divide by zero, or from an attempt to divide by a number too small for the dividend. In the former case (divide by zero), the division will not be attempted and the B- and A-register contents will be unchanged except that a negative quantity will be made positive. In the latter case (divisor too small), the execution will be attempted with unpredictable results left in the B- and A-registers. If there is no divide error, the overflow bit is cleared.

DLD DOUBLE LOAD

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

Memory Address

Loads the contents of addressed memory location m (and m + 1) into the A- and B-registers, respectively.

DST DOUBLE STORE

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

Memory Address

Stores the double-word quantity in the A- and B-registers into addressed memory locations m (and m + 1), respectively.

MPY MULTIPLY

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

Memory Address

Multiplies a 16-bit integer in the A-register by a 16-bit integer in the addressed memory location. The resulting double-length integer product resides in the B- and A-registers, with the B-register containing the sign bit and the most-significant 15 bits of the quantity. The A-register may be used as an operand (i.e., memory address 0), resulting in an arithmetic square. The instruction clears the overflow bit.

3-22. EXTENDED ARITHMETIC REGISTER REFERENCE INSTRUCTIONS

The six extended arithmetic register reference instructions provide various types of shifting operations on the combined contents of the B- and A-registers. The B-register is considered to be to the left (most-significant word) and the A-register is considered to be to the right (least-significant word). An example of each type of shift operation is illustrated in figure 3-4.

The complete instruction is given in one word and includes four bits (unshaded) to specify the number of shifts (1 to 16). By viewing these four bits as a binary-coded number, the number of shifts is easily expressed; i.e., binary-coded 1 = 1 shift, binary-coded 2 = 2 shifts . . . binary-coded 15 = 15 shifts. The maximum number of 16 shifts is coded with four zeros, which essentially exchanges the contents of the B- and A-registers.

The extend bit is not affected by any of the following instructions. Except for the arithmetic shifts, overflow also is not affected.

ASL ARITHMETIC SHIFT LEFT

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

Number of Shifts

Arithmetically shifts the combined contents of the B- and A-registers left n places. The value of n may be any number from 1 through 16. Zeros are filled into vacated low-order positions of the A-register. The sign bit is not affected, and data bits are lost out of bit position 14 of the B-register. If any one of the lost bits is a significant data bit ("1" for positive numbers, "0" for negative numbers), the overflow bit will be set; otherwise, overflow will be cleared during execution. See ASL example in figure 3-4. Note that two additional shifts in this example would cause an error by losing a significant '1'.

ASR ARITHMETIC SHIFT RIGHT

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

Number of Shifts

Arithmetically shifts the combined contents of the B- and A-registers right n places. The value of n may be any number from 1 through 16. The sign bit is unchanged and

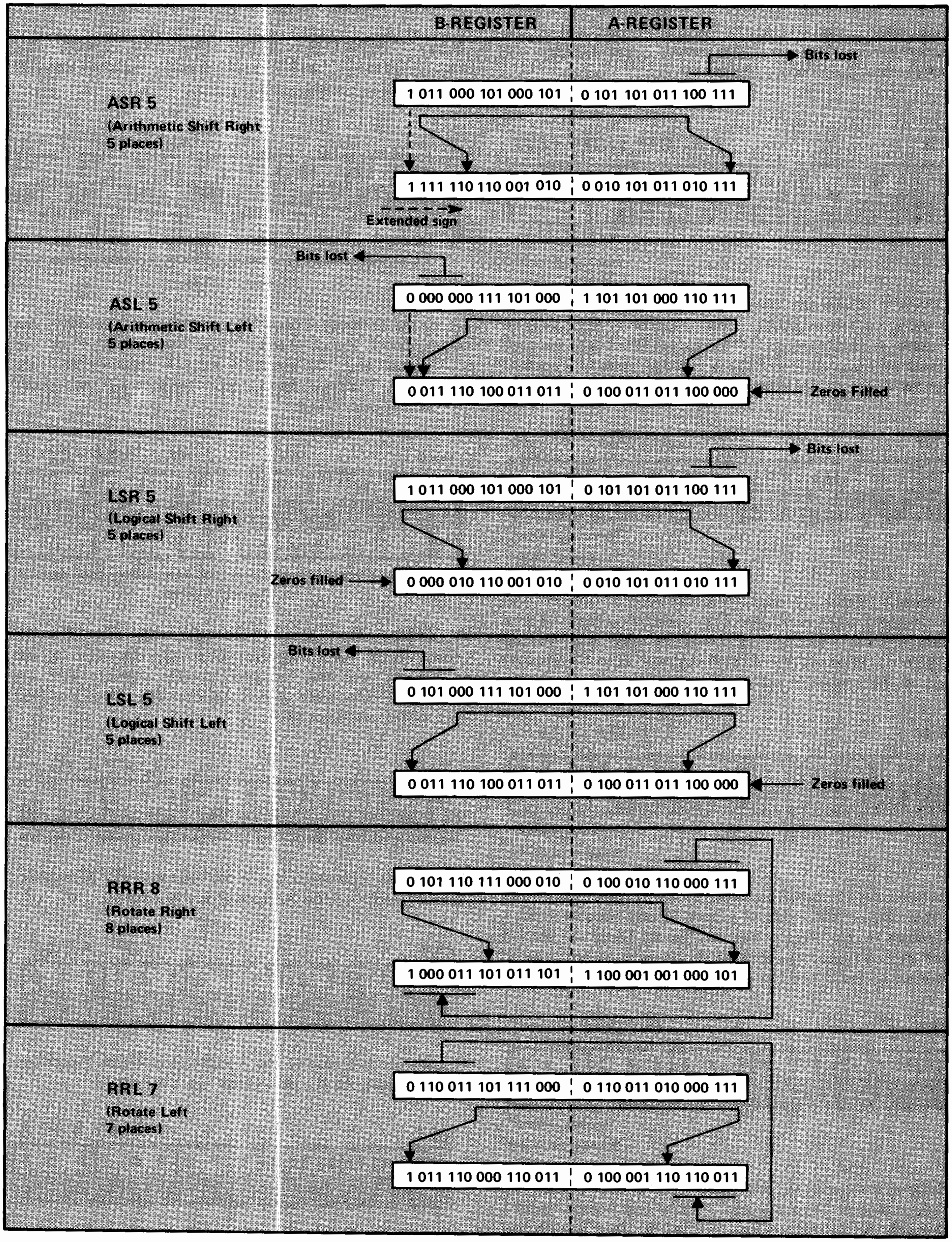


Figure 3-4. Examples of Double-Word Shifts and Rotates

is extended into bit positions vacated by the right shift. Data bits shifted out of the least-significant end of the A-register are lost. Overflow cannot occur because the instruction clears the overflow bit.

LSL LOGICAL SHIFT LEFT

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

Number of Shifts

Logically shifts the combined contents of the B- and A-registers left n places. The value of n may be any number from 1 through 16. Zeros are filled into vacated low-order bit positions of the A-register; data bits are lost out of the high-order bit positions of the B-register.

LSR LOGICAL SHIFT RIGHT

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

Number of Shifts

Logically shifts the combined contents of the B- and A-registers right n places. The value of n may be any number from 1 through 16. Zeros are filled into vacated high-order bit positions of the B-register; data bits are lost out of the low-order bit positions of the A-register.

RRL ROTATE LEFT

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

Number of Shifts

Rotates the combined contents of the B- and A-registers left n places. The value of n may be any number from 1 through 16. No bits are lost or filled in. Data bits shifted out of the high-order end of the B-register are rotated around to enter the low-order end of the A-register.

RRR ROTATE RIGHT

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

Number of Shifts

Rotates the combined contents of the B- and A-registers right n places. The value of n may be any number from 1 through 16. No bits are lost or filled in. Data bits shifted out of the low-order end of the A-register are rotated around to enter the high-order end of the B-register.

3-23. EXTENDED INSTRUCTION GROUP

3-24. INDEX REGISTER INSTRUCTIONS. The index registers (X and Y) are two 16-bit registers accessible by the following instructions.

ADX ADD MEMORY TO X

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

Memory Address

Adds the contents of the addressed memory location to the contents of the X-register. The sum remains in the X-register and the contents of the memory cell are unaltered. The result of this addition may set the extend bit or the overflow bit.

ADY ADD MEMORY TO Y

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

Memory Address

Adds the contents of the addressed memory location to the contents of the Y-register. The sum remains in the Y-register and the contents of the memory cell are unaltered. The result of this addition may set the extend bit or the overflow bit.

CAX COPY A TO X

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

Copies the contents of the A-register into the X-register. The contents of the A-register are unaltered.

CAY COPY A TO Y

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

Copies the contents of the A-register into the Y-register. The contents of the A-register are unaltered.

CBX COPY B TO X

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

Copies the contents of the B-register into the X-register. The contents of the B-register are unaltered.

CBY COPY B TO Y

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

Copies the contents of the B-register into the Y-register. The contents of the B-register are unaltered.

CXA COPY X TO A

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

Copies the contents of the X-register into the A-register. The contents of the X-register are unaltered.

CXB COPY X TO B

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

Copies the contents of the X-register into the B-register. The contents of the X-register are unaltered.

CYA COPY Y TO A

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

Copies the contents of the Y-register into the A-register. The contents of the Y-register are unaltered.

CYB COPY Y TO B

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

Copies the contents of the Y-register into the B-register. The contents of the Y-register are unaltered.

DSX DECREMENT X AND SKIP IF ZERO

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

Subtracts one from the contents of the X-register. If the result of this operation is zero (X-register decremented from 000001 to 000000), the next instruction is skipped; i.e., the P-register count is advanced two counts instead of one count. If the result is not zero, the next sequential instruction is executed.

DSY DECREMENT Y AND SKIP IF ZERO

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

Subtracts one from the contents of the Y-register. If the result of this operation is zero (Y-register decremented from 000001 to 000000), the next instruction is skipped; i.e., the P-register count is advanced two counts instead of one count. If the result is not zero, the next sequential instruction is executed.

ISX INCREMENT X AND SKIP IF ZERO

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

Adds one to the contents of the X-register. If the result of this operation is zero (X-register rolls over to 000000 from 177777), the next instruction is skipped; i.e., the P-register count is advanced two counts instead of one count. If the result is not zero, the next sequential instruction is executed.

ISY INCREMENT Y AND SKIP IF ZERO

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

Adds one to the contents of the Y-register. If the result of this operation is zero (Y-register rolls over to 000000 from 177777), the next instruction is skipped; i.e., the P-register count is advanced two counts instead of one count. If the result is not zero, the next sequential instruction is executed.

LAX LOAD A INDEXED BY X

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

Operand Address

Loads the A-register with the contents indicated by the effective address, which is computed by adding the contents of the X-register to the operand address. The effective address is loaded into the M-register; the X-register and memory contents are not altered. Indirect addressing is resolved before indexing; bit 15 of the effective address is ignored.

LAY LOAD A INDEXED BY Y

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

Operand Address

Loads the A-register with the contents indicated by the effective address, which is computed by adding the contents of the Y-register to the operand address. The effective address is loaded into the M-register; the Y-register and memory contents are not altered. Indirect addressing is resolved before indexing; bit 15 of the effective address is ignored.

LBX LOAD B INDEXED BY X

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

Operand Address

Loads the B-register with the contents indicated by the effective address, which is computed by adding the contents of the X-register to the operand address. The effective address is loaded into the M-register; the X-register and memory contents are not altered. Indirect addressing is resolved before indexing; bit 15 of the effective address is ignored.

LBY LOAD B INDEXED BY Y

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

Operand Address

Loads the B-register with the contents indicated by the effective address, which is computed by adding the contents of the Y-register to the operand address. The effective address is loaded into the M-register; the X-register and memory contents are not altered. Indirect addressing is resolved before indexing; bit 15 of the effective address is ignored.

LDX LOAD X FROM MEMORY

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

Memory Address

Loads the contents of the addressed memory location into the X-register. The A- and B-registers may be addressed as locations 00000 and 00001, respectively; however, if it is desired to load from the A- or B-register, copy instructions CAX or CBX should be used since they are more efficient.

LDY LOAD Y FROM MEMORY

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

Memory Address

Loads the contents of the addressed memory location into the Y-register. The A- and B-registers may be addressed as locations 00000 and 00001, respectively; however, if it is desired to load from the A- or B-register, copy instructions CAY or CBY should be used since they are more efficient.

SAX STORE A INDEXED BY X

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

Operand Address

Stores the contents of the A-register into the location indicated by the effective address, which is computed by adding the contents of the X-register to the operand address. The effective address is loaded into the M-register; the A- and X-register contents are not altered. Indirect addressing is resolved before indexing; bit 15 of the effective address is ignored.

SAY STORE A INDEXED BY Y

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

Operand Address

Stores the contents of the A-register into the location indicated by the effective address, which is computed by adding the contents of the Y-register to the operand address. The effective address is loaded into the M-register; the A- and Y-register contents are not altered. Indirect addressing is resolved before indexing; bit 15 of the effective address is ignored.

SBX STORE B INDEXED BY X

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

Operand Address

Stores the contents of the B-register into the location indicated by the effective address, which is computed by adding the contents of the X-register to the operand address. The effective address is loaded into the M-register; the B- and X;register contents are not altered. Indirect addressing is resolved before indexing; bit 15 of the effective address is ignored.

SBY STORE B INDEXED BY Y

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

Operand Address

Stores the contents of the B-register into the location indicated by the effective address, which is computed by adding the contents of the Y-register to the operand address. The effective address is loaded into the M-register; the B- and Y-register contents are not altered. Indirect addressing is resolved before indexing; bit 15 of the effective address is ignored.

STX STORE X TO MEMORY

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

Memory Address

Stores the contents of the X-register into the addressed memory location. The A- and B-registers may be addressed as locations 00000 and 00001, respectively. The X-register contents are not altered.

STY STORE Y TO MEMORY

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

Memory Address

Stores the contents of the Y-register into the addressed memory location. The A- and B-registers may be addressed as locations 00000 and 00001, respectively. The Y-register contents are not altered.

XAX EXCHANGE A AND X

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

Exchanges the contents of the A- and X-registers.

XAY EXCHANGE A AND Y

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

Exchanges the contents of the A- and Y-registers.

XBX EXCHANGE B AND X

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

Exchanges the contents of the B- and X-registers.

XBY EXCHANGE B AND Y

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

Exchanges the contents of the B- and Y-registers.

3-25. JUMP INSTRUCTIONS. The following two jump instructions involving the Y-register allow a program to either jump to or exit from a subroutine.

JLY JUMP AND LOAD Y

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

Memory Address

This instruction is designed for entering a subroutine. The instruction, executed in location P, causes computer control to jump unconditionally to the memory location specified in the memory address. Indirect addressing may be specified. The contents of the P-register plus two (return address) is loaded into the Y-register. A return to the main program sequence at P + 2 may be effected by a JPY instruction (described next). A memory protect check is performed by this instruction. The effective address may not be below the fence, including the addressable A- and B-registers.

JPY JUMP INDEXED BY Y

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

Operand Address

Transfers control to the effective address, which is computed by adding the contents of the Y-register to the operand address. Indirect addressing is not allowed. The effective address is loaded into the P-register; the Y-register contents are not altered. A memory protect check is performed by this instruction. The effective address may not be below the fence, including the addressable A- and B-registers.

3-26. BYTE MANIPULATION INSTRUCTIONS.

A byte address is defined as two times the word address plus zero or one, depending on whether the byte is in the high-order position (bits 8 through 15) or low-order position (bits 0 through 7) of the word containing it. If the byte of interest is in bit positions 8 through 15 of memory location 100, for example, then the address of that byte is 2* 100 + 0, or 200; the address of the low-order byte in the same location is 201 (2* 100 + 1). Because of the way byte addresses are defined, 16 bits are required to cover all possible byte addresses in a 32K-word memory configuration. Hence, for byte addressing, bit 15 does not indicate indirect addressing.

Byte addresses 000 through 003 reference bytes in the A- and B-registers. These addresses will not cause memory violations. The user should, however, be careful in referencing these byte addresses; for example, storing into byte address 002 or 003 would destroy the byte address originally contained in the B-register.

CBT COMPARE BYTES

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

Memory Address

Compares the bytes in string 1 with those in string 2. This is a three-word instruction where

Word 1 = Instruction code,

Word 2 = Address of word containing the string count, and

Word 3 = All-zeros word reserved for use by microcode.

The operand addresses are in the A- and B-registers. The A-register contains the first byte address of string 1 and the B-register contains the first byte address of string 2.

The number of bytes to be compared is given in the memory location addressed by Word 2 of the instruction; the number of bytes to be compared is restricted to a positive integer greater than zero. The strings are compared one byte at a time; the ith byte in string 1 is compared with the ith byte in string 2. The comparison is performed arithmetically; i.e., each byte is treated as a positive number. If all bytes in string 1 are identical with all bytes in string 2, the "equal" exit is taken. As soon as two bytes are compared and found to be different, the "less than" or "greater than" exit is taken, depending on whether the byte in string 1 is less than or greater than the byte in string 2. The three ways this instruction exits are as follows:

a. No skip if string 1 is equal to string 2; the P-register advances one count from Word 3 of the instruction. The A-register contains its original value incremented by the count stored in the address specified in Word 2.

b. Skips one word if string 1 is less than string 2; the P-register advances two counts from Word 3 of the instruction. The A-register contains the address of the byte in string 1 where the comparison stopped.

c. Skips two words if string 1 is greater than string 2; the P-register advances three counts from Word 3 of the instruction. The A-register contains the address of the byte in string 1 where the comparison stopped.

For all three exits, the B-register will contain its original value incremented by the count stored in the address specified in Word 2. This instruction is interruptible. The interrupt routine is expected to save and restore the contents of the A- and B-registers. During the interrupt, the remaining count is stored in Word 3 of the instruction.

LBT LOAD BYTE

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

This one word instruction loads into the A-register the byte whose address is contained in the B-register. The byte is right-justified with leading zeros in the left byte. The B-register is incremented by one.

MBT MOVE BYTES

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

Memory Address

Moves bytes in a left-to-right manner; i.e., the byte having the lowest address from the source is moved first. This is a three word instruction where

Word 1 = Instruction code,

Word 2 = Address of word containing the byte count, and

Word 3 = All-zeros word reserved for use by microcode.

The operand addresses are in the A- and B-registers. The A-register contains the first byte address source and the B-register contains the first byte address destination.

The number of bytes to be moved is given by a 16-bit positive integer greater than zero addressed by Word 2 of the instruction. The byte address in the A- and B-registers are incremented as each byte is being moved. Thus, at the end of the operation, the A- and B-registers are incremented by the number of bytes moved. Wraparound of the byte address would result from a carry out of bit position 15; therefore, if the destination became 000, 001, 002, or 003, the next byte would be moved into the A- or B-register and destroy the proper byte addresses for the move operation. For each byte move, a memory protect check is performed.

This instruction is interruptible. The interrupt routine is expected to save and restore the contents of the A- and B-registers. During the interrupt, the remaining count is stored in Word 3 of the instruction.

SBT STORE BYTE

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

Stores the A-register low-order (right) byte in the byte address contained in the B-register. The B-register is incremented by one. A memory protect check is performed before the byte is stored. The left byte in the A-register does not have to be zeros. The other byte in the same word of the stored byte is not altered.

SFB SCAN FOR BYTE

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

This is a one word instruction with the operands in the A- and B-registers. The A-register contains a termination byte (high-order byte) and a test byte (low-order byte). The B-register contains the first byte address of the string to be scanned.

A string of bytes is scanned starting at the byte address given in the B-register. Scanning terminates when a byte in the string matches either the test byte or the termination byte in the A-register. The manner in which the instruction exits depends on which byte is matched first. If a byte in the string matches the test byte, the instruction will not skip upon exit; the B-register will contain the address of the byte matching the test byte. If a byte in the string matches the termination byte, the instruction will skip one word upon exit; the B-register will contain the address of the byte matching the termination byte *plus one*.

The scanning operation will not continue indefinitely even if neither the termination byte nor test byte exists in memory. These bytes are in the A-register with byte addresses 000 and 001, respectively. Thus, if no match is made by the time the B-register points to the last byte in memory, the B-register will roll over to zero and the next test will match the termination byte in the A-register with itself.

This instruction is interruptible. The interrupt routine is expected to save and restore the contents of the A- and B-registers.

3-27. BIT MANIPULATION INSTRUCTIONS.

The following three instructions allow any number of bits in a specified memory location to be cleared, set, or tested.

CBS CLEAR BITS

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

Memory Address

Clears bits in the addressed location. This is a three-word instruction where

Word 1 = Instruction code,

Word 2 = Address of a 16-bit mask, and

Word 3 = Address of word where bits are to be cleared.

The bits to be cleared correspond to logic 1's in the mask. The bits corresponding to logic 0's in the mask are not affected. A memory protect check is performed prior to modifying the word in memory.

SBS SET BITS

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

Memory Address

Sets bits in the addressed location. This is a three-word instruction where

Word 1 = Instruction code,

Word 2 = Address of a 16-bit mask, and

Word 3 = Address of word where bits are to be set.

The bits to be set correspond to logic 1's in the mask. The bits corresponding to logic 0's in the mask are not affected. A memory protect check is performed prior to modifying the word in memory.

TBS TEST BITS

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

Memory Address

Tests (compares) bits in the addressed location. This is a three-word instruction where

Word 1 = Instruction code,

Word 2 = Address of a 16-bit mask, and

Word 3 = Address of word in which bits are to be tested.

The bits in the addressed memory word corresponding to logic 1's in the mask are tested. If all the bits tested are 1's, the instruction will not skip; otherwise the instruction will skip one word (i.e., the P-register will advance two counts from Word 3 of the instruction).

3-28. WORD MANIPULATION INSTRUCTIONS.

The following instructions facilitate the comparing and moving of word arrays.

CMW COMPARE WORDS

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

Memory Address

Compares the words in array 1 with those in array 2. This is a three-word instruction where

Word 1 = Instruction code,

Word 2 = Address of word containing the word count, and

Word 3 = All-zeros word reserved for use by microcode.

The operand addresses are in the A- and B-registers. The A-register contains the first word address of array 1 and the B-register contains the first word address of array 2. Bit 15 of the addresses in the A- and B-registers are ignored; i.e., no indirect addressing allowed.

The number of words to be compared is given in the memory location addressed by Word 2 of the instruction; the number of words to be compared is restricted to a positive integer greater than zero. The arrays are compared one word at a time; the ith word in array 1 is compared with the ith word in array 2. This comparison is performed arithmetically; i.e., each word is considered a two's complement number. If all words in array 1 are equal to all words in array 2, the "equal" exit is taken. As soon as two words are compared and found to be different, the "less than" or "greater than" exit is taken, depending on whether the word in array 1 is less than or greater than the word in array 2. The three ways this instruction exits are as follows:

a. No skip if array 1 is equal to array 2; the P-register advances one count from Word 3 of the instruction. The A-register contains its original value incremented by the word count stored in the address specified in Word 2.

b. Skips one word if array 1 is less than array 2; the P-register advances two counts from Word 3 of the instruction. The A-register contains the address of the word in array 1 where the comparison stopped.

c. Skips two words if array 1 is greater than array 2; the P-register advances three counts from Word 3 of the instruction. The A-register contains the address of the word in array 1 where the comparison stopped.

For all three exits, the B-register will contain its original value incremented by the word count stored in the address specified in Word 2. This instruction is interruptible. The interrupt routine is expected to save and restore the contents of the A- and B-registers. During the interrupt, the remaining count is stored in Word 3 of the instruction.

MVW MOVE WORDS

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

Memory Address

Moves words in a left-to-right manner; i.e., the word having the lowest address in the source is moved first. This is a three-word instruction where

Word 1 = Instruction code,

Word 2 = Address of word containing the count, and

Word 3 = All-zeros word reserved for use by microcode.

The operand addresses are in the A- and B-registers. The A-register contains the first word address source and the B-register contains the first word address destination. The number of words to be moved is a 16-bit positive integer greater than zero addressed by Word 2 of the instruction. The word addresses in the A- and B-registers are incremented as each word is being moved. Thus, at the end of the operation, the A- and B-registers are incremented by the number of words moved.

Wraparound of the word address would result from a carry into bit position 15 (i.e., at 32767). If the destination address became 000 or 001, the next word would be moved into the A- or B-register and destroy the proper word addresses for the move operation. For each word move, a memory protect check is performed.

This instruction is interruptible. The interrupt routine is expected to save and restore the contents of the A- and B-registers. During the interrupt, the remaining count is stored in Word 3 of the instruction.

3-29. FLOATING POINT INSTRUCTIONS

The floating point instructions allow addition, subtraction, multiplication, and division of both single precision (32-bit) and extended precision (48-bit) floating point quantities, and conversion of quantities from floating point format to integer format or vice versa. Data formats are shown in figure 3-1. Except for zero, all floating point operands must be normalized (i.e., sign of mantissa differs from most significant bit of mantissa).

For multiple-word instructions, indirect addressing to any number of levels is permitted for the word(s) indicated as memory address. A logic 0 in bit position 15 specifies direct addressing; a logic 1 specifies indirect addressing.

After initiating an operation in the Floating Point Processor (FPP), the floating point firmware waits for FPP completion. If completion is not signalled by the FPP within 25 microseconds, an FPP malfunction exists. The firmware indicates this condition by creating a memory protect violation (memory protect installed), setting the overflow flag, and setting the A-register to all ones (an unnormalized nunber).

The execution times of the floating point instructions are specified under paragraph 3-32. These instructions are noninterruptible; any attempted interrupt is held off for the full execution time of the currently active floating point instruction. However, data transfer via the dual-channel port controller (DCPC) is not held off.

Information required for direct user-microprogramming utilizing the Floating Point Processor is provided in the *HP Computer Microprogramming Reference Manual,* part no. 02109-90004.

3-30. SINGLE PRECISION OPERATIONS. Overflow for single precision operations occurs if the result lies outside the range of representable single precision floating point numbers $[-2^{127}, (1-2^{-23})\ 2^{127}]$. In such a case, the overflow flag is set and the result $(1-2^{-23})\ 2^{127}$ is returned to the A- and B-registers. Underflow occurs if the result lies inside the range $[-2^{-129}(1+2^{-22}), 2^{-129}]$. In such a case, the overflow flag is set and the result 0 is returned to the A- and B-registers.

FAD FLOATING POINT ADD

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

Memory Address

Adds the floating point quantity in the A- and B-registers to the floating point quantity in the specified memory locations. The floating point result is returned to the A- and B-registers.

FSB FLOATING POINT SUBTRACT

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

Memory Address

Subtracts the floating point quantity in the specified memory locations from the floating point quantity in the A- and B-registers. The floating point result is returned to the A- and B-registers.

FMP FLOATING POINT MULTIPLY

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

Memory Address

Multiplies the floating point quantity in the A- and B-registers by the floating point quantity in the specified memory locations. The floating point result is returned to the A- and B-registers.

FDV FLOATING POINT DIVIDE

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

Memory Address

Divides the floating point quantity in the A- and B-registers by the floating point quantity in the specified memory locations. The floating point result is returned to the A- and B-registers.

FIX FLOATING POINT TO SINGLE INTEGER

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

Converts the floating point quantity in the A- and B-registers to single integer format. The integer result is returned to the A-register. If the magnitude of the floating point number is <1, regardless of sign, the integer 0 is returned. If the magnitude of the exponent of the floating point number is ⩾16, regardless of sign, the integer 32767 (077777 octal) is returned as the result and the overflow flag is set.

FLT SINGLE INTEGER TO FLOATING POINT

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

Converts the single integer quantity in the A-register to single precision floating point format. The floating point result is returned to the A- and B-registers.

.FIXD* FLOATING POINT TO DOUBLE INTEGER

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

Converts the floating point quantity in the A- and B-registers to double integer format. The integer result is returned to the A- and B-registers. (The A-register contains the most-significant word and the B-register contains the least-significant word.) If the magnitude of the floating point number is <1, regardless of sign, the integer 0 is returned. If the magnitude of the floating point number is ⩾32, regardless of sign, the integer $2^{31}-1$ is returned as the result and the overflow flag is set.

.FLTD* DOUBLE INTEGER TO FLOATING POINT

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

Converts the double integer quantity in the A- and B-registers to single precision floating point format. The floating point result is returned to the A- and B-registers.

3-31. EXTENDED PRECISION OPERATIONS.

Overflow for extended precision operations occurs if the result lies outside the range of representable extended precision floating point numbers $[-2^{127}, (1-2^{-39})\ 2^{-127}]$. In such a case, the overflow flag is set and $(1-2^{-39})\ 2^{127}$ is returned as the result. Underflow occurs if the result lies inside the range $[-2^{-129}\ (1+2^{-38}), 2^{-129}]$. In such a case, the overflow flag is set and 0 is returned as the result.

.XADD* EXTENDED FLOATING POINT ADD

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

Memory Address

Adds two extended precision floating point quantities (augend plus addend). This is a four-word instruction where

Word 1 = Instruction code.
Word 2 = Address of result.
Word 3 = Address of augend.
Word 4 = Address of addend.

*For HP Assembly Language usage, refer to paragraph 3-37.

.XSUB* EXTENDED FLOATING POINT SUBTRACT

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

Memory Address

Subtracts one extended precision floating point quantity from another (minuend minus subtrahend). This is a four-word instruction where

Word 1 = Instruction code.
Word 2 = Address of result.
Word 3 = Address of minuend.
Word 4 = Address of subtrahend.

.XMPY* EXTENDED FLOATING POINT MULTIPLY

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

Memory Address

Multiplies one extended precision floating point quantity by another (multiplicand by multiplier). This is a four-word instruction where

Word 1 = Instruction code.
Word 2 = Address of result.
Word 3 = Address of multiplicand.
Word 4 = Address of multiplier.

*For HP Assembly Language usage, refer to paragraph 3-37.

.XDIV* EXTENDED FLOATING POINT DIVIDE

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

Memory Address

Divides one floating point quantity by another (dividend by divisor). This is a four-word instruction where

Word 1 = Instruction code.
Word 2 = Address of result.
Word 3 = Address of dividend.
Word 4 = Address of divisor.

.XFXS* EXTENDED FLOATING POINT TO SINGLE INTEGER

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

Memory Address

Converts the extended precision floating point quantity in the specified memory locations to single integer format. The integer result is returned to the A-register. If the magnitude of the floating point number is <1, regardless of sign, 0 is returned as the result. If the magnitude of the exponent of the floating point number is ⩾16, regardless of sign, the integer $2^{15}-1$ is returned as the result and the overflow flag is set.

.XFTS* SINGLE INTEGER TO EXTENDED FLOATING POINT

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

Memory Address

Converts the single integer quantity in the A-register to extended precision floating point format. The floating point result is returned to the specified memory locations.

*For HP Assembly Language usage, refer to paragraph 3-37.

.XFXD* EXTENDED FLOATING POINT TO DOUBLE INTEGER

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

Memory Address

Converts the extended precision floating point quantity in the specified memory locations to double integer format. The integer result is returned to the A- and B-registers. (The A-register contains the most-significant word and the B-register contains the least-significant word.) If the magnitude of the floating point number is <1, regardless of sign, 0 is returned as the result. If the magnitude of the exponent of the floating point number is ⩾32, regardless of sign, the integer $2^{31}-1$ is returned as the result and the overflow flag is set.

.XFTD* DOUBLE INTEGER TO EXTENDED FLOATING POINT

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

Memory Address

Converts the double integer quantity in the A- and B-registers to extended precision floating point format. The floating point result is returned to the specified memory locations.

3-32. INSTRUCTION EXECUTION TIMES

Table 3-5 lists the execution times required for the various base set instructions.

Table 3-5. Typical Base Set Instruction Execution Times[1]

INSTRUCTION	EXECUTION TIME (us)			
	High Performance Memory		High Performance Fault Control Memory	
	Non-DMS	DMS	Non-DMS	DMS
Memory Reference Group				
ADA/B,AND,IOR,LDA/B,XOR	0.910	0.910	0.910	0.980
1st Indirect Level	0.455	0.455	0.455	0.490
Each Additional Level	0.630	0.630	0.630	0.630
STA/B	1.260	1.260	1.330	1.400
1st Indirect Level	0.455	0.455	0.455	0.490
Each Additional Level	0.630	0.630	0.630	0.630
CPA/B (no skip)	1.085	1.085	1.085	1.120
(skip)	1.435	1.435	1.435	1.505
1st Indirect Level	0.455	0.455	0.455	0.490
Each Additional Level	0.630	0.630	0.630	0.630
ISZ (no skip)	1.540	1.540	1.540	1.610
(skip)	1.610	1.610	1.610	1.680
1st Indirect Level	0.455	0.455	0.455	0.490
Each Additional Level	0.630	0.630	0.630	0.630
JMP	0.735	0.735	0.735	0.735
1st Indirect Level	0.350	0.350	0.350	0.385
2nd Indirect Level	0.560	0.560	0.560	0.595
Each Additional Level	0.805	0.875	0.875	0.980
JSB	1.260	1.260	1.330	1.400
1st Indirect Level	0.350	0.420	0.420	0.490
2nd Indirect Level	0.560	0.490	0.490	0 490
Each Additional Level	0.805	0.875	0.875	0.980

*For HP Assembly Language usage, refer to paragraph 3-37.

*For HP Assembly Language usage, refer to paragraph 3-37.

Table 3-5. Typical Base Set Instruction Execution Times[1] (Continued)

INSTRUCTION	EXECUTION TIME (us)			
	High Performance Memory		High Performance Fault Control Memory	
	Non-DMS	DMS	Non-DMS	DMS
Shift/Rotate Group				
(no skip)	0.910	0.910	0.910	0.980
(skip)	1.260	1.330	1.330	1.470
Alter/Skip Group				
(no skip)	0.910	0.910	0.910	0.980
(skip)	1.260	1.330	1.330	1.470
Input/Output Group[2]				
SFS, SFC, SOS, SOC				
(no skip)	1.575	1.575	1.575	1.575
(skip)	1.855	1.855	1.855	1.890
Extended Arithmetic Group				
ASL	1.820 +0.175/shift	1.890 +0.175/shift	1.890 +0.175/shift	1.960 +0.175/shift
ASR	1.470 +0.175/shift	1.470 +0.175/shift	1.470 +0.175/shift	1.505 +0.175/shift
LSL,LSR,RRL,RRR	1.470 +0.175/shift	1.540 +0.175/shift	1.540 +0.175/shift	1.610 +0.175/shift
DLD	2.520	2.590	2.590	2.765
DST	2.695	2.695	2.765	2.905
MPY	5.320 to 6.055	5.320 to 6.125	5.320 to 6.125	5.425 to 6.300
DIV	7.665 to 9.065	7.665 to 9.065	7.665 to 9.065	7.770 to 9.205
Extended Instruction Group				
(Index Register Instructions)				
CAX.CBX,CAY,CBY,CXA,CXB,CYA,CYB	1.295	1.295	1.295	1.330
XAX,XBX,XAY,XBY	1.925	1.925	1.925	1.960
DSX,DSY (no skip)	1.750	1.750	1.750	1.820
(skip)	1.995	2.065	2.065	2.205
ISX,ISY (no skip)	1.750	1.750	1.750	1.820
(skip)	1.995	2.065	2.065	2.205
LDX,LDY	2.660	2.730	2.730	2.835
Each Indirect Level	0.805	0.875	0.875	0.980
STX,STY	2.940	2.940	2.940	3.010
Each Indirect Level	0.805	0.875	0.875	0.980
LAX,LBX,LAY,LBY	3.360	3.430	3.430	3.535
Each Indirect Level	0.805	0.875	0.875	0.980
SAX,SAY,SBX,SBY	3.465	3.465	3.465	3.500
Each Indirect Level	0.805	0.875	0.875	0.980
ADX,ADY	2.730	2.800	2.800	2.870
Each Indirect Level	0.805	0.875	0.875	0.980
JLY	2.660	2.660	2.660	2.695
Each Indirect Level	0.805	0.875	0.875	0.980
JPY	2.275	2.275	2.275	2.345

Table 3-5. Typical Base Set Instruction Execution Times[1] (Continued)

INSTRUCTION	EXECUTION TIME (us)			
	High Performance Memory		High Performance Fault Control Memory	
	Non-DMS	DMS	Non-DMS	DMS
Extended Instruction Group (Continued)				
(Byte Manipulation Instructions)				
LBT (from high byte)	3.640	3.640	3.640	3.675
(from low byte)	3.360	3.360	3.360	3.395
SBT (to high byte)	4.340	4.410	4.410	4.515
(to low byte)	3.885	4.025	4.025	4.165
MBT	3.745	3.815	3.815	3.955
	+4.042/byte	+4.078/byte	+4.078/byte	+4.130/byte
Each Indirect Level	0.805	0.875	0.875	0.980
CBT	3.745	3.815	3.815	3.955
	+4.480/byte	+4.480/byte	+4.480/byte	+4.480/byte
Each Indirect Level	0.805	0.875	0.875	0.980
SFB (if compare exit)	1.925	1.995	1.995	2.065
	+2.730/byte	+2.730/byte	+2.730/byte	+2.730/byte
(if terminal exit)	2.450	2.450	2.450	2.485
	+2.730/byte	+2.730/byte	+2.730/byte	+2.730/byte
(Word Manipulation Instructions)				
CMW	3.75	3.815	3.815	3.955
	+2.380/word	+2.520/word	+2.520/word	+2.660/word
Each Indirect Level	0.805	0.875	0.875	0.980
MVW	3.745	3.815	3.815	3.955
	+1.680/word	+1.680/word	+1.680/word	+1.680/word
Each Indirect Level	0.805	0.875	0.875	0.980
(Bit Manipulation Instructions)				
CBS,SBS	4.480	4.550	4.550	4.690
Each Indirect Level	0.805	0.875	0.875	0.980
TBS (no skip)	4.935	4.935	4.935	5.040
(skip)	5.005	5.075	5.075	5.250
Each Indirect Level	0.805	0.875	0.875	0.984
Floating Point Group				
FAD	4.865 to 7.735	4.935 to 7.805	4.935 to 7.805	5.075 to 7.945
FSB	4.865 to 7.735	4.935 to 7.805	4.935 to 7.805	5.075 to 7.945
FMP	6.125 to 6.475	6.195 to 6.545	6.195 to 6.545	6.335 to 6.685
FDV	6.125 to 9.345	6.195 to 9.415	6.195 to 9.415	6.335 to 9.555
FIX	3.640 to 5.250	3.640 to 5.250	3.640 to 5.250	3.675 to 5.285
FLT	3.395 to 4.550	3.395 to 4.550	3.395 to 4.550	3.430 to 4.585
FIXD	3.465 to 6.230	3.465 to 6.230	3.465 to 6.230	3.500 to 6.265
FLTD	3.290 to 5.705	3.290 to 5.705	3.290 to 5.705	3.325 to 5.740
.XADD	10.430 to 13.930	10.570 to 14.070	10.640 to 14.140	10.990 to 14.490
.XSUB	10.430 to 13.930	10.570 to 14.070	10.640 to 14.140	10.990 to 14.490
.XMPY	12.320 to 13.125	12.460 to 13.265	12.530.to 13.335	12.880 to 13.685
.XDIV	12.320 to 17.605	12.460 to 17.745	12.530 to 17.815	12.880 to 18.165
.XFXS	5.705 to 6.860	5.775 to 6.930	5.775 to 6.930	5.950 to 7.105
.XFTS	5.775 to 6.510	5.775 to 6.510	5.845 to 6.580	5.915 to 6.650
.XFXD	6.335 to 8.750	6.405 to 8.820	6.405 to 8.820	6.580 to 8.995
.XFTD	6.475 to 8.015	6.475 to 8.015	6.545 to 8.085	6.580 to 8.120

[1] Semiconductor memory refresh may increase program execution times by up to 3%.

[2] Depends on which I/O time period (T2,3,4,5,6) the instruction begins.

3-33. SCIENTIFIC INSTRUCTION SET

The Scientific Instruction Set (SIS) utilizes the Floating Point Processor (FPP) to perform nine trigonometric and logarithmic functions. After initiating an operation in the FPP, the SIS firmware waits for FPP completion. If completion is not signalled by the FPP within 25 microseconds, an FPP malfunction exists. The firmware indicates this condition by creating a memory protect violation (memory protect installed), setting the overflow flag, and setting the A-register to all ones (an unnormalized number).

The following paragraphs provide machine language coding and definitions for the SIS instructions. Error conditions and codes are given in table 3-6. Note that except for zero, all floating point operands must be normalized (i.e., sign of mantissa differs from most significant bit of mantissa).

TAN* TANGENT

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

Calculates the tangent of the single precision floating point quantity (in radians) contained in the A- and B-registers. The result is returned to the A- and B-registers. A normal return will skip the next instruction. An error return will execute the next instruction, set the overflow bit, and return an ASCII error code in the A- and B-registers.

SQRT* SQUARE ROOT

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

Calculates the square root of the single precision floating point quantity contained in the A- and B-registers. The result is returned to the A- and B-registers. A normal return will skip the next instruction. An error return will execute the next instruction, set the overflow bit, and return an ASCII error code in the A- and B-registers.

ALOG* NATURAL LOGARITHM

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

Calculates the natural logarithm of the single precision floating point quantity contained in the A- and B-registers. The result is returned to the A- and B-registers. A normal return will skip the next instruction. An error return will execute the next instruction, set the overflow bit, and return an ASCII error code in the A- and B-registers.

ATAN* ARCTANGENT

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

Calculates the arctangent of the single precision floating point quantity contained in the A- and B-registers. The result (in radians) is returned to the A- and B-registers.

COS* COSINE

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

Calculates the cosine of the single precision floating point quantity (in radians) contained in the A- and B-registers. The result is returned to the A- and B-registers. A normal return will skip the next instruction. An error return will execute the next instruction, set the overflow bit, and return an ASCII error code in the A- and B-registers.

SIN* SINE

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

Calculates the sine of the single precision floating point quantity (in radians) contained in the A- and B-registers. The result is returned to the A- and B-registers. A normal return will skip the next instruction. An error return will execute the next instruction, set the overflow bit, and return an ASCII error code in the A- and B-registers.

EXP* E TO THE POWER X

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

Calculates e to the power x of the single precision floating point quantity contained in the A- and B-registers. The result is returned to the A- and B-registers. A normal return will skip the next instruction. An error condition will execute the next instruction, set the overflow bit, and return an ASCII error code in the A- and B-registers.

*For HP Assembly Language usage, refer to paragraph 3-37.

*For HP Assembly Language usage, refer to paragraph 3-37.

ALOGT* COMMON LOGARITHM

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

Calculates the common logarithm of the single precision floating point quantity contained in the A- and B-registers. The result is returned to the A- and B-registers. A normal return will skip the next instruction. An error condition will execute the next instruction, set the overflow bit, and return an ASCII error code in the A- and B-registers.

TANH* HYPERBOLIC TANGENT

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

Calculates the hyperbolic tangent of the single precision floating point quantity contained in the A- and B-registers. The result is returned to the A- and B-registers.

Table 3-6. SIS Instruction Error Codes

INSTRUCTION	ERROR CODE
TAN	09OR if $|x| > 32768^*\pi/4$
SQRT	03UN if $x < 0$
ALOG	02UN if $x \leq 0$
ATAN	None
COS	05OR if $|x| > 32768^*\pi/4$
SIN	05OR if $|x| > 32768^*\pi/4$
EXP	07OF if $x > 88.029678$
ALOGT	02UN if $x \leq 0$
TANH	None

Where:
OF = Integer or floating point overflow.
OR = Out of range.
UN = Floating point underflow.

3-34. EXECUTION TIMES AND INTERRUPTS

Table 3-7 lists the typical execution times required for the SIS instructions. Also listed is the maximum period of non-interruptible instruction execution. If an instruction is interrupted, its execution restarts from the beginning.

*For HP Assembly Language usage, refer to paragraph 3-37.

Table 3-7. Typical Scientific Instruction Set Execution Times

INSTRUCTION	EXECUTION TIME (us)	MAXIMUM NON-INTERRUPTIBLE (us)
TAN	48.4	8
SQRT	30.9	11
ALOG	43.3	8
ATAN	42.4	12
COS	47.9	9
SIN	47.6	8
EXP	44.7	8
ALOGT	49.4	8
TANH	57.2	9

3-35. FAST FORTRAN INSTRUCTIONS

The Fast FORTRAN Processor instructions perform several frequently-used FORTRAN operations including parameter passing, array address calculations, and floating point conversion, packing, rounding and normalization operations.

For multiple-word instructions, indirect addressing to any number of levels is permitted for the word(s) indicated as a memory address. A logic 0 in bit position 15 specifies direct addressing; a logic 1 specifies indirect addressing.

The following paragraphs provide machine language coding and definitions for the fast FORTRAN instructions. Data formats are shown in figure 3-1.

DBLE* SINGLE FLOATING POINT TO EXTENDED FLOATING POINT

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

Memory Address

Converts the single precision floating point quantity in specified memory locations to an extended precision floating point quantity. The result is returned to other

*For HP Assembly Language usage, refer to paragraph 3-37.

specified memory locations. This is a four-word instruction where

Word 1 = Instruction code.
Word 2 = Return address.
Word 3 = Address of result.
Word 4 = Address of operand.

SNGL* EXTENDED FLOATING POINT TO SINGLE FLOATING POINT

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

Memory Address

Converts the extended precision floating point quantity in the specified memory locations to a single precision floating point quantity. The result is placed in the A- and B-registers. This is a three word instruction where

Word 1 = Instruction code.
Word 2 = Return address.
Word 3 = Address of operand.

.DFER* TRANSFER EXTENDED FLOATING POINT

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

Memory Address

Transfers an extended precision floating point quantity (three consecutive words) from one memory location to another. The source address +3 is returned to the A-register; the destination address +3 is returned to the B-register. This is a three word instruction where

Word 1 = Instruction code.
Word 2 = Destination address.
Word 3 = Source address.

.XFER* TRANSFER EXTENDED FLOATING POINT

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

Transfers an extended precision floating point quantity (three consecutive words) from one memory location to another. The A-register must contain the source address and the B-register must contain the destination address. The source address +3 is returned to the A-register; the destination address +3 is returned to the B-register.

.XPAK* NORMALIZE AND PACK EXTENDED FLOATING POINT

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

Memory Address

Normalizes, rounds, and packs with exponent the mantissa of an extended precision floating point quantity located in the specified memory locations. The value of the exponent must be in the A-register. The result is returned to the same specified memory locations.

.XCOM* COMPLEMENT EXTENDED FLOATING POINT ADD

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

Memory Address

Complements in place an unpacked extended precision floating point quantity located in the specified memory locations. The result is returned to the same specified memory locations. If the exponent is adjusted, the A-register equals 1; if not, the A-register equals 0.

*For HP Assembly Language usage, refer to paragraph 3-37.

*For HP Assembly Language usage, refer to paragraph 3-37.

..DCM* COMPLEMENT AND NORMALIZE EXTENDED FLOATING POINT

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

Memory Address

Complements and normalizes in place an unpacked extended precision floating point quantity located in the specified memory locations. The result is returned to the same specified memory locations. If the exponent is adjusted, the A-register equals 1; if not, the A-register equals 0.

DDINT* EXTENDED FLOATING POINT TO DOUBLE INTEGER

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

Memory Address

Converts the extended precision floating point quantity in specified memory locations to double integer format. The integer result is returned to other specified memory locations. This is a four-word instruction where

Word 1 = Instruction code.
Word 2 =Return address.
Word 3 = Address of result.
Word 4 = Address of operand.

If the magnitude of the floating point quantity is <1, regardless of sign, 0 is returned as the result. If the magnitude of the exponent of the floating point quantity is ⩾32, regardless of sign, the integer $2^{31}-1$ is returned as the result and the overflow flag is set.

.GOTO* TRANSFER CONTROL

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

Transfers control to the location indicated by a FORTRAN computed GO TO statement: GO TO (K1,K2, . . . , Kn),J. If J<1, then result address is K1; if J>n, address is Kn; otherwise, address is KJ. The succeeding instructions must define the return address, the value of J, and the parameters.

..MAP* COMPUTE ARRAY ELEMENT ADDRESS

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

Computes the address of a specified element of a 2 or 3 dimensional array; returns the address to the A-register. The A-register must equal 0 for a 2 dimensional array or 1 for a 3 dimensional array. The B-register must contain the number of words per variable. Succeeding instructions must define base address, element subscripts, and length of first (or first and second) dimension.

.ENTR* TRANSFER PARAMETER ADDRESSES

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

Transfers the true addresses of parameters from a calling sequence into a subroutine; adjusts return address to the true return point. A true address is determined by eliminating all indirect references.

.ENTP* TRANSFER PARAMETER ADDRESSES

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

Transfers the true addresses of parameters from a calling sequence into a subroutine; adjusts return address to the true return point. A true address is determined by eliminating all indirect references.

*For HP Assembly Language usage, refer to paragraph 3-37.

*For HP Assembly Language usage, refer to paragraph 3-37

.PWR2*

X TIMES 2 TO THE POWER N

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

Calculates for floating point x and integer n: y = x*2^n. The floating point quantity must be in the A- and B-registers; the succeeding instruction must define integer n. The first word of the two word floating point result is returned to the A-register; the second word, to the B-register.

.FLUN*

UNPACK FLOATING POINT QUANTITY

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

Unpacks a floating point quantity. The lower part of the floating point quantity must be in the B-register. The exponent is returned to the A-register; the lower part of the mantissa is returned to the B-register.

.SETP*

SET A TABLE

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

Sets a table of increasing numbers in consecutive memory locations. The A-register must contain the initial number and the B-register must contain the initial memory address (direct only); the succeeding instruction must define the number of memory locations (count ⩾ 0). Entries in the table are established by incrementing the initial address and number by one (1) for each successive entry until the last number, initial number +COUNT−1, is reached. On return, the B-register equals the initial address +COUNT−1.

NOTE

> If the initial address +COUNT −1 results in an address which is beyond the end of logical memory, addresses within the base page are destroyed.

.PACK*

NORMALIZE FLOATING POINT QUANTITY

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

Converts the signed mantissa of a floating point quantity into a normalized format. The floating point quantity must be in the A- and B-registers. The succeeding instruction must reserve one word of memory for temporary storage of the exponent. The first word of the two word floating point result is returned to the A-register; the second word, to the B-register.

3-36. EXECUTION TIMES AND INTERRUPTS

Table 3-8 lists the typical execution times required for the Fast FORTRAN Processor instructions. Also listed is the maximum period of non-interruptible instruction execution time where applicable. If an instruction is interrupted, it restarts execution from the beginning.

Table 3-8. Typical Fast FORTRAN Instruction Execution Times

INSTRUCTION	EXECUTION TIME (us)	MAXIMUM NON-INTERRUPTIBLE (us)
DBLE	13.02	
SNGL	18.2	
.DFER	12.81	
.XFER	8.96 to 12.7	
.XPAK	18.9 to 29.5	11.6
.XCOM	11.7 to 12.1	
..DCM	22.1 to 33.4	12.2
DDINT	23.9 to 58.6	30.6
.GOTO	10.6	
..MAP	17.7 to 27.2	
.ENTR	13.9 + 3.7*NP'	
.ENTP	13.6 + 3.7*NP'	
.PWR2	8.4	
.FLUN	3.1	
.SETP	6.4 + 1.2*COUNT	42.4 (COUNT = 30)
.PACK	19.2 to 27.2	

*For HP Assembly Language usage, refer to paragraph 3-37.

*For HP Assembly Language usage, refer to paragraph 3-37.

3-37. ASSEMBLY LANGUAGE

New instructions not recognized by the HP Assembler require different handling in HP Assembly Language programming. These instructions are asterisked in the preceding paragraphs and must be used in the form: JSB x where x is the instruction. (The instruction, x, must be declared as an external at the beginning of the assembly language program.) Most of these instructions correspond to library subroutines* and must be implemented into HP RTE systems (as described in the following paragraph) to enable their execution in hardware-firmware instead of in software.

3-38. RTE IMPLEMENTATION

New instructions are implemented in an RTE-M, RTE-II, or RTE-IV system simply by changing library entry points during the parameter input phase of system generation. (Refer to the appropriate RTE manual for the system generation procedure.) Using the list of entry point opcodes given in table 3-9, make the entry changes as indicated below:

```
.XADD,RP,105001
.XSUB,RP,105021
 .
 .
 .
.PACK,RP,105230
```

Alternatively, entry points may be changed by loading (via LOADR) a "replacement" program when user programs are loaded. An example replacement program is shown below.

```
ASMB,L,R
       NAM REPLA
       ENT .XADD, . XSUB, . . . . , . PACK
.XADD RPL 105001B
.XSUB RPL 105021B
 .
 .
 .
.PACK RPL 105230B
       END
```

Table 3-9. Instructions and Opcodes

INSTRUCTION MNEMONIC	OCTAL OPCODE	INSTRUCTION MNEMONIC	OCTAL OPCODE
.XADD	105001	TANH	105330
.XSUB	105021	.DBLE	105201
.XMPY	105041	.SNGL	105202
.XDIV	105061	.DFER	105205
.XFXS	105101	.XFER	105220
.FIXD	105104	.XPAK	105206
.XFXD	105105	.XCOM	105215
.XFTS	105121	..DCM	105216
.FLTD	105124	DDINT	105217
.XFTD	105125	.GOTO	105221
TAN	105320	..MAP	105222
SQRT	105321	.ENTR	105223
ALOG	105322	.ENTP	105224
ATAN	105323	.PWR2	105225
COS	105324	.FLUN	105226
SIN	105325	.SETP	105227
EXP	105326	.PACK	105230
ALOGT	105327		

*Refer to the Relocatable Library Reference Manual, part no. 24998-90001.

DYNAMIC MAPPING SYSTEM

SECTION IV

The basic addressing space of the HP 1000 F-Series Computer is 32,768 words, which is referred to as *logical* memory. The amount of MOS memory actually installed in the computer system is referred to as *physical* memory. An HP 1000 F-Series Computer with the Dynamic Mapping System (DMS) has an addressing capability for one million words of physical memory. The DMS allows logical memory to be mapped into physical memory through the use of four dynamically alterable memory maps. (The DMS is standard for the HP 2117F and optional for the HP 2111F.)

4-1. MEMORY ADDRESSING

The basic memory addressing scheme provides for addressing 32 pages of logical memory, each of which consists of 1,024 words. This memory is addressed through a 15-bit memory address bus shown in figure 4-1. The upper 5 bits of this bus provide the page address and the lower 10 bits provide the relative word address within the page.

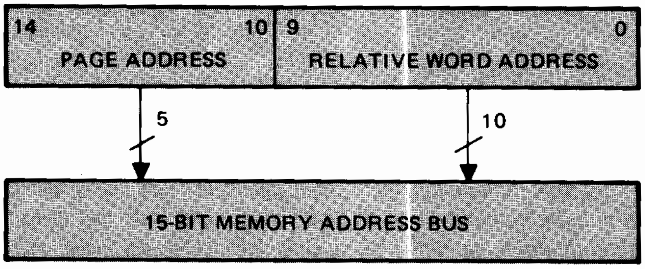


Figure 4-1. Basic Memory Addressing Scheme

The Memory Expansion Module (MEM), which is part of the DMS option, converts the 5-bit page address into a 10-bit page address and thereby allows 1,024 (2^{10}) pages to be addressed. This conversion is accomplished by allowing the original 5-bit address to identify one of the 32 12-bit registers within a "memory map." Each of these map registers contains the new user-specified 10-bit page address. This new page address is combined with the original 10-bit relative address to form a 20-bit memory address bus as shown in figure 4-2.

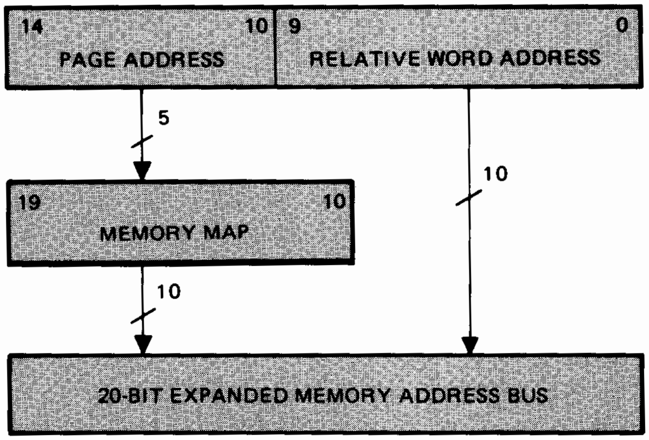


Figure 4-2. Expanded Memory Addressing Scheme

4-2. MAP REGISTER LOADING

Conversion of the basic 16-bit word data format to and from the map register 12-bit word data format is shown in figure 4-3. Bits 13 through 10 of the basic data format are not used by the memory map registers. Read and write memory protect violations are discussed in paragraph 4-3.

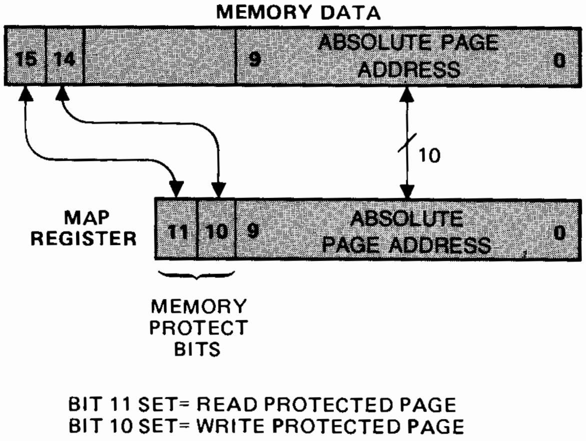


Figure 4-3. Basic Word Format Vs Map Register Format

4-3. STATUS AND VIOLATION REGISTERS

The MEM also includes a status register and a violation register. As shown in table 4-1, the MEM status register contents enable the programmer to determine whether the MEM was enabled or disabled at the time of the last interrupt and the address of the base page fence. The MEM violation register contents enable the programmer to determine whether a fault occurred in the hardware or the software so that the proper corrective steps may be taken. Refer to table 4-2.

Table 4-1. MEM Status Register Format

BIT	SIGNIFICANCE
15	0 = MEM disabled at last interrupt 1 = MEM enabled at last interrupt
14	0 = System map selected at last interrupt 1 = User map selected at last interrupt
13	0 = MEM disabled currently 1 = MEM enabled currently
12	0 = System map selected currently 1 = User map selected currently
11	0 = Protected mode disabled currently 1 = Protected mode enabled currently
10	Portion mapped*
9	Base page fence bit 9
8	Base page fence bit 8
7	Base page fence bit 7
6	Base page fence bit 6
5	Base page fence bit 5
4	Base page fence bit 4
3	Base page fence bit 3
2	Base page fence bit 2
1	Base page fence bit 1
0	Base page fence bit 0

*Bit 10	Mapped Address (M)
0	$\text{Fence} \leqslant M < 2000_8$
1	$1 < M < \text{Fence}$

Note: The base page fence separates the reserved (mapped) memory from the shared (unmapped) memory. Bit 10 specifies which area is reserved (mapped). (Refer to LFA and LFB instructions contained in paragraph 4-6.)

Table 4-2. MEM Violation Register Format

BIT	SIGNIFICANCE
15	Read violation*
14	Write violation*
13	Base page violation*
12	Privileged instruction violation*
11	Reserved
10	Reserved
9	Reserved
8	Reserved
7	0 = ME bus disabled at violation 1 = ME bus enabled at violation
6	0 = MEM disabled at violation 1 = MEM enabled at violation
5	0 = System map enabled at violation 1 = User map enabled at violation
4	Map address bit 4
3	Map address bit 3
2	Map address bit 2
1	Map address bit 1
0	Map address bit 0

*Significant when associated bit is set.

Any attempt to read from a read-protected page will result in a read violation and the memory read will not occur. Any attempt to write into a write-protected page will result in a write violation and the memory will not be altered. In addition, if a page is write protected, a jump or jump indirect instruction to that page will cause a write violation and the jump will not occur. It should be noted that all violation rules are ignored for DCPC signals.

If a read or write violation occurs, the MEM signals the memory protect logic that a violation has occurred which causes the memory protect logic to generate an interrupt. As discussed in paragraph 6-3, memory violations are interrupted to select code 05 and a DMS violation can be distinguished from a memory protect violation by executing an SFS 05 instruction. If the skip occurs, DMS is in violation; if no skip occurs, memory protect is in violation.

4-4. MAP SEGMENTATION

All registers within the memory map are dynamically alterable. To maximize the system performance capability, the MEM includes four separate memory maps: the User Map, System Map, and two Dual-Channel Port Controller (DCPC) Maps. (See figure 4-4.) These maps, which are manipulated through the use of 38 machine-language instructions, are addressed as a contiguous register block. It should be noted that the base page fence applies to both the System Map and the User Map.

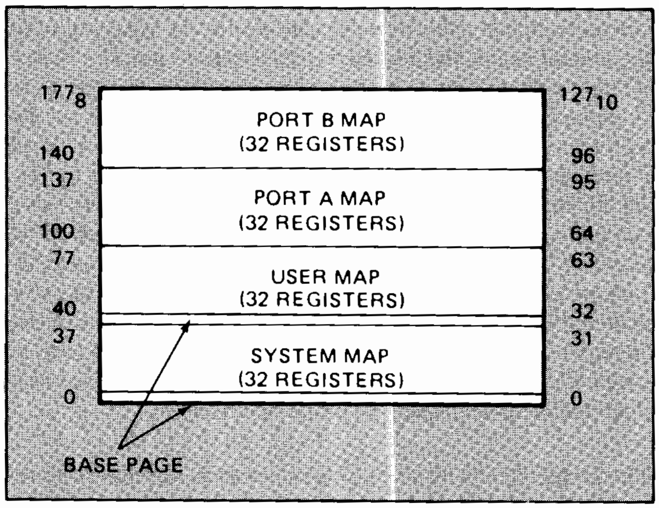


Figure 4-4. Map Segmentation

4-5. POWER FAIL CHARACTERISTICS

A power failure automatically enables the System Map, and a minimum of 500 microseconds is assured the programmer for executing a power fail routine. Since all maps are disabled and none are considered valid upon the restoration of power, the power fail routine should include instructions to save as many maps as desired.

4-6. DMS INSTRUCTION CODING

Machine language coding and definitions of the 38 Dynamic Mapping System instructions are provided on this and following pages. A sample map load and enable routine is given in paragraph 4-8.

DJP DISABLE MEM AND JUMP

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

Memory Address

Disables the translation and protection features of the MEM hardware. Prior to disabling, the P-register is set to the effective memory address. As a result of executing this instruction, normal I/O interrupts are held off until the first opportunity following the fetch of the next instruction, unless three or more levels of indirect addressing are used.

This instruction will normally generate an MEM violation when executed in the protected mode. In this case, the status of the MEM is not affected and the jump will not occur; however, if the System map is enabled, the instruction is allowed.

DJS DISABLE MEM AND JUMP TO SUBROUTINE

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

Memory Address

Disables the translation and protection features of the MEM hardware. Prior to disabling, the P-register is set one count past the effective memory address (m + 1) and the return address is stored in location m. As a result of executing this instruction, normal I/O interrupts are held off until the first opportunity following the fetch of the next instruction, unless three or more levels of indirect addressing are used.

This instruction will normally generate an MEM violation when executed in the protected mode. In this case, the status of the MEM is not affected and the jump will not occur; however, if the System map is enabled, the instruction is allowed.

JRS JUMP AND RESTORE STATUS

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

Memory Address

Causes the status of the MEM to be restored. This is a three-word instruction where

Word 1 = Instruction code,

Word 2 = Status word address, and

Word 3 = Jump address.

Only bits 15 and 14 of the status word are used; the remaining bits (13-0) of the status word are ignored. Bits 15 and 14 restore the MEM status as follows:

Bit 15 = 0 = MEM will be disabled
= 1 = MEM will be enabled

Bit 14 = 0 = System map will be selected
= 1 = User map will be selected

As a result of executing this instruction, normal I/O interrupts are held off until after the fetch of the next instruction, unless a total of three or more levels of indirect addressing are used in Word 2 (status word address) and Word 3 (jump address). For example, if Word 2 contains one level of indirect addressing and Word 3 contains two levels of indirect addressing, interrups will not be held off past the fetch of the next instruction.

This instruction will normally generate an MEM violation when executed in the protected mode. In this case, the status of the MEM is not affected and the jump will not occur; however, if the System map is enabled, the instruction is allowed.

LFA LOAD FENCE FROM A

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

Loads the contents of the A-register into the base page fence register. Bits 9-0 of the A-register specify the address in page zero where shared (unmapped) memory is separated from reserved (mapped) memory. Bit 10 is used as follows to specify which portion is mapped:

Bit 10	Mapped Address (M)
0	Fence ⩽ M < 2000_8
1	1 < M < Fence

This instruction will normally generate an MEM violation when executed in the protected mode; however, it is allowed if the System map is enabled. When an MEM violation does occur, the fence is not altered.

LFB LOAD FENCE FROM B

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

Loads the contents of the B-register into the base page fence register. Bits 9-0 of the B-register specify the address in page zero where shared (unmapped) memory is separated from reserved (mapped) memory. Bit 10 is used as follows to specify which portion is mapped:

Bit 10	Mapped Address (M)
0	Fence ⩽ M < 2000_8
1	1 < M < Fence

This instruction will normally generate an MEM violation when executed in the protected mode; however, it is allowed if the System map is enabled. When an MEM violation does occur, the fence is not altered.

MBF MOVE BYTES FROM ALTERNATE MAP

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

Moves a string of bytes using the alternate program map for source reads and the currently enabled map for destination writes. The A-register contains the source byte address and the B-register contains the destination byte address. The initial byte addresses in the A- and B-registers must be even byte addresses. The byte in bits 15 through 8 of a word is the even byte. The X-register contains the octal number of bytes to be moved. The number of bytes to be moved is restricted to a positive integer greater than zero. If the contents of the X-register is zero, the instruction will be a NOP. If the contents of the X-register is a negative integer, a large indeterminate block of memory will be transferred. Both the source and destination must begin on word boundaries.

The instruction is interruptible on an even number of byte transfers, thus maintaining the even word boundaries in the A- and B-registers. The interrupt routine is expected to save and restore the current contents of the A-, B-, and X-registers to allow continuation of the instruction at the next entry. When the byte string move is completed, the X-register will always be zero and the A- and B-registers will contain their original value incremented by the number of bytes moved.

This instruction can cause an MEM violation only if read or write protection rules are violated.

MBI MOVE BYTES INTO ALTERNATE MAP

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

Moves a string of bytes using the currently enabled map for source reads and the alternate program map for destination writes. The A-register contains the source byte address and the B-register contains the destination byte address. The initial byte addresses in the A- and B-registers must be even byte addresses. The byte in bits 15 through 8 of a word is the even byte. The X-register contains the octal number of bytes to be moved. The number of bytes to be moved is restricted to a positive integer greater than zero. If the contents of the X-register is zero, the instruction will be a NOP. If the contents of the X-register is a negative integer, a large indeterminate block of memory will be transferred. Both the source and destination must begin on word boundaries.

The instruction is interruptible on an even number of byte transfers, thus maintaining the even word boundaries in the A- and B-registers. The interrupt routine is expected to save and restore the current contents of the A-, B-, and X-registers to allow continuation of the instruction at the next entry. When the byte string move is completed, the X-register will always be zero and the A- and B-registers will contain their original value incremented by the number of bytes moved.

This instruction will always cause an MEM violation when executed in the protected mode and no bytes will be transferred.

MBW

MOVE BYTES WITHIN ALTERNATE MAP

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

Moves a string of bytes with both the source and destination addresses established through the alternate program map. The A-register contains the source byte address and the B-register contains the destination byte address. The initial byte addresses in the A- and B-registers must be even byte addresses. The byte in bits 15 through 8 of a word is the even byte. The X-register contains the octal number of bytes to be moved. The number of bytes to be moved is restricted to a positive integer greater than zero. If the contents of the X-register is zero, the instruction will be a NOP. If the contents of the X-register is a negative integer, a large indeterminate block of memory will be transferred. Both the source and destination must begin on word boundaries.

The instruction is interruptible on an even number of byte transfers, thus maintaining the even word boundaries in the A- and B-registers. The interrupt routine is expected to save and restore the current contents of the A-, B-, and X-registers to allow continuation of the instruction at the next entry. When the byte string move is completed, the X-register will always be zero and the A- and B-registers will contain their original value incremented by the number of bytes moved.

This instruction will always cause an MEM violation when executed in the protected mode and no bytes will be transferred.

MWF

MOVE WORDS FROM ALTERNATE MAP

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

Moves a string of words using the alternate program map for source reads and the currently enabled map for destination writes. The A-register contains the source address and the B-register contains the destination address. The X-register contains the octal number of words to be moved. The number of words to be moved is restricted to a positive integer greater than zero. If the contents of the X-register is zero, the instruction will be a NOP. If the contents of the X-register is a negative integer, a large indeterminate block of memory will be transferred.

The instruction is interruptible. The interrupt routine is expected to save and restore the current contents of the A-, B-, and X-registers to allow continuation of the instruction at the next entry. When the word string move is completed, the X-register will always be zero and the A- and B-registers will contain their original value incremented by the number of words moved.

This instruction can cause an MEM violation only if read and write protection rules are violated.

MWI

MOVE WORDS INTO ALTERNATE MAP

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

Moves a string of words using the currently enabled map for source reads and the alternate program map for destination writes. The A-register contains the source address and the B-register contains the destination address. The X-register contains the octal number of words to be moved. The number of words to be moved is restricted to a positive integer greater than zero. If the contents of the X-register is zero, the instruction will be a NOP. If the contents of the X-register is a negative integer, a large indeterminate block of memory will be transferred.

The instruction is interruptible. The interrupt routine is expected to save and restore the current contents of the A-, B-, and X-registers to allow continuation of the instruction at the next entry. When the word string move is completed, the X-register will always be zero and the A- and B-registers will contain their original value incremented by the number of words moved.

This instruction will always cause an MEM violation when executed in the protected mode and no words will be transferred.

MWW

MOVE WORDS WITHIN ALTERNATE MAP

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

Moves a string of words with both the source and destination addresses established through the alternate program map. The A-register contains the source address and the B-register contains the destination address. The X-register contains the octal number of words to be moved. The number of words to be moved is restricted to a positive integer greater than zero. If the contents of the X-register is zero, the instruction will be a NOP. If the contents of the X-register is a negative integer, a large indeterminate block of memory will be transferred.

The instruction is interruptible. The interrupt routine is expected to save and restore the current contents of the A-, B-, and X-registers to allow continuation of the instruction at the next entry. When the word string move is completed, the X-register will always be zero and the A- and B-registers will contain their original value incremented by the number of words moved.

This instruction will always cause an MEM violation when executed in the protected mode and no words will be transferred.

PAA LOAD/STORE PORT A MAP PER A

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

Transfers the 32 Port A map registers to or from memory. If bit 15 of the A-register is clear, the Port A map is *loaded* from memory starting from the address specified in bits 14-0 of the A-register. If bit 15 of the A-register is set, the Port A map is *stored* into memory starting at the address specified in bits 14-0 of the A-register. When the load/store operation is complete, the A-register will be incremented by 32 to allow multiple map instructions.

An attempt to load any map register when in the protected mode will cause an MEM violation. An attempt to store the Port A map is allowed within the constraints of write protected memory.

PAB LOAD/STORE PORT A MAP PER B

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

Transfers the 32 Port A map registers to or from memory. If bit 15 of the B-register is clear, the Port A map is *loaded* from memory starting from the address specified in bits 14-0 of the B-register. If bit 15 of the B-register is set, the Port A map is *stored* into memory starting at the address specified in bits 14-0 of the B-register. When the load/store operation is complete, the B-register will be incremented by 32 to allow multiple map instructions.

An attempt to load any map register when in the protected mode will cause an MEM violation. An attempt to store the Port A map is allowed within the constraints of write protected memory.

PBA LOAD/STORE PORT B MAP PER A

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

Transfers the 32 Port B map registers to or from memory. If bit 15 of the A-register is clear, the Port B map is *loaded* from memory starting from the address specified in bits 14-0 of the A-register. If bit 15 of the A-register is set, the Port B map is *stored* into memory starting at the address specified in bits 14-0 of the A-register. When the load/store operation is complete, the A-register will be incremented by 32 to allow multiple map instructions.

An attempt to load any map register when in the protected mode will cause an MEM violation. An attempt to store the Port B map is allowed within the constraints of write protected memory.

PBB LOAD/STORE PORT B MAP PER B

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

Transfers the 32 Port B map registers to or from memory. If bit 15 of the B-register is clear, the Port B map is *loaded* from memory starting from the address specified in bits 14-0 of the B-register. If bit 15 of the B-register is set, the Port B map is *stored* into memory starting at the address specified in bit 14-0 of the B-register. When the load/store operation is complete, the B-register will be incremented by 32 to allow multiple map instructions.

An attempt to load any map register when in the protected mode will cause an MEM violation. An attempt to store the Port B map is allowed within the constraints of the write protected memory.

RSA READ STATUS REGISTER INTO A

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

Reads the contents of the MEM status register into the A-register. This instruction can be executed at any time. The format of the MEM status register is given in table 4-1.

RSB READ STATUS REGISTER INTO B

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

Reads the contents of the MEM status register into the B-register. This instruction can be executed at any time. The format of the MEM status register is given in table 4-1.

RVA READ VIOLATION REGISTER INTO A

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

Reads the contents of the MEM violation register into the A-register. This instruction can be executed at any time. The format of the MEM violation register is given in table 4-2.

RVB READ VIOLATION REGISTER INTO B

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

Reads the contents of the MEM violation register into the B-register. This instruction can be executed at any time. The format of the MEM violation register is given in table 4-2.

SJP ENABLE SYSTEM MAP AND JUMP

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

Memory Address

Causes the MEM hardware to use the set of 32 map registers, referred to as the System map, for translating all programmed memory references. Prior to enabling the System map, the P-register is set to the effective memory address. As a result of executing this instruction, normal I/O interrupts are held off until the first opportunity following the fetch of the next instruction, unless three or more levels of indirect addressing are used.

This instruction will normally generate an MEM violation when executed in the protected mode. In this case, the status of the MEM is not affected and the jump will not occur; however, if the System map is enabled, the instruction is allowed and effectively executes a JMP *+1,I.

SJS ENABLE SYSTEM MAP AND JUMP TO SUBROUTINE

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

Memory Address

Causes the MEM hardware to use the set of 32 map registers, referred to as the System map, for translating all programmed memory references. Prior to enabling the System map, the P-register is set one count past the effective memory address (m + 1). After enabling the System map, the return address is stored in m. As a result of executing this instruction, normal I/O interrupts are held off until the first opportunity following the fetch of the next instruction, unless three or more levels of indirect addressing are used.

This instruction will normally generate an MEM violation when executed in the protected mode. In this case, the status of the MEM is not affected and the jump will not occur; however, if the system map is enabled, the instruction is allowed and effectively executes a JSB *+1,I.

SSM STORE STATUS REGISTER INTO MEMORY

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

Memory Address

Stores the 16-bit contents of the MEM status register into the address memory location. The status register contents are not altered. This instruction is used in conjunction with the JRS instruction to allow easy processing of interrupts, which always select the System map (if the MEM is enabled). The format of the MEM status register is listed in table 4-1.

This instruction can cause an MEM violation only if write protection rules are violated.

SYA LOAD/STORE SYSTEM MAP PER A

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

Transfers the 32 System map registers to or from memory. If bit 15 of the A-register is clear, the System map is *loaded* from memory starting from the address specified in bits 14-0 of the A-register. If bit 15 of the A-register is set, the System map is *stored* into memory starting at the address specified in bits 14-0 of the A-register. When the load/store operation is complete, the A-register will be incremented by 32 to allow multiple map instructions.

Note: If not in the protected mode, the MEM provides no protection against altering the contents of maps while they are currently enabled.

An attempt to load any map in the protected mode will cause an MEM violation. An attempt to store the System map is allowed within the constraints of write protected memory.

SYB LOAD/STORE SYSTEM MAP PER B

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

Transfers the 32 System map registers to or from memory. If bit 15 of the B-register is clear, the System map is *loaded* from memory starting from the address specified in bits 14-0 of the B-register. If bit 15 of the B-register is set, the System map is *stored* into memory starting at the address specified in bits 14-0 of the B-register. When the load/store operation is complete, the B-register will be incremented by 32 to allow multiple map instructions.

Note: If not in the protected mode, the MEM provides no protection against altering the contents of maps while they are currently enabled.

An attempt to load any map in the protected mode will cause an MEM violation. An attempt to store the System map is allowed within the constraints of write protected memory.

UJP ENABLE USER MAP AND JUMP

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

Memory Address

Causes the MEM hardware to use the set of 32 map registers, referred to as the User map, for translating all programmed memory references. Prior to enabling the User map, the P-register is set to the effective memory address. As a result of executing this instruction, normal I/O interrupts are held off until the first opportunity following the fetch of the next instruction, unless three or more levels of indirect addressing are used.

This instruction will normally generate an MEM violation when executed in the protected mode. In this case, the status of the MEM is not affected and the jump will not occur; however, if the System map is enabled, the instruction is allowed.

UJS ENABLE USER MAP AND JUMP TO SUBROUTINE

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

Memory Address

Causes the MEM hardware to use the set of 32 map registers, referred to as the User map, for translating all programmed memory references. Prior to enabling the User map, the P-register is set one count past the effective memory address (m + 1). After enabling the System map, the return address is stored in m. As a result of executing this instruction, normal I/O interrupts are held off until the first opportunity following the fetch of the next instruction, unless three or more levels of indirect addressing are used.

This instruction will normally generate an MEM violation when executed in the protected mode. In this case, the status of the MEM is not affected and the jump will not occur; however, if the System map is enabled, the instruction is allowed.

USA LOAD/STORE USER MAP PER A

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

Transfers the 32 User map registers to or from memory. If bit 15 of the A-register is clear, the User map is *loaded* from memory starting from the address specified in bits 14-0 of the A-register. If bit 15 of the A-register is set, the User map is *stored* into memory starting at the address specified in bits 14-0 of the A-register. When the load/store operation is complete, the A-register will be incremented by 32 to allow multiple map instructions.

Note: If not in the protected mode, the MEM provides no protection against altering the contents of maps while they are currently enabled.

An attempt to load any map in the protected mode will cause an MEM violation. An attempt to store the User map is allowed within the constraints of write protected memory.

USB LOAD/STORE USER MAP PER B

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

Transfer the 32 User map registers to or from memory. If bit 15 of the B-register is clear, the User map is *loaded* from memory starting from the address specified in bits 14-0 of the B-register. If bit 15 of the B-register is set, the User map is *stored* into memory starting at the address specified in bits 14-0 of the B-register. When the load/store operation is complete, the B-register will be incremented by 32 to allow multiple map instructions.

Note: If not in the protected mode, the MEM provides no protection against altering the contents of maps while they are currently enabled.

Any attempt to load any map in the protected mode will cause an MEM violation. An attempt to store the User map is allowed within the constraints of write protected memory.

XCA CROSS COMPARE A

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

Memory Address

Compares the contents of the A-register with the contents of the addressed memory location. If the two 16-bit words are not identical, the next instruction is skipped; i.e., the P-register advances three counts instead of two counts. If the two words are identical, the next instruction is executed. Neither the A-register contents nor memory cell contents are altered.

This instruction uses the alternate program map to determine the addressed memory location. If the MEM is currently disabled, then a compare directly with physical memory occurs.

This instruction will cause an MEM violation only if read protection rules are violated.

XCB CROSS COMPARE B

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

Memory Address

Compares the contents of the B-register with the contents of the addressed memory location. If the two 16-bit words are not identical, the next instruction is skipped; i.e., the P-register advances three counts instead of two counts. If the two words are identical, the next instruction is executed. Neither the B-register contents nor memory cell contents are altered.

This instruction uses the alternate program map to determine the addressed memory location. If the MEM is currently disabled, then a compare directly with physical memory occurs.

This instruction will cause an MEM violation only if read protection rules are violated.

XLA CROSS LOAD A

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

Memory Address

Loads the contents of the specified memory address into the A-register. The contents of the memory cell are not altered.

This instruction uses the alternate program map to fetch the operand. If the MEM is currently disabled, then a load directly from physical memory occurs.

This instruction will cause an MEM violation only if read protection rules are violated.

XLB CROSS LOAD B

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

Memory Address

Loads the contents of the specified memory address into the B-register. The contents of the memory cell are not altered.

This instruction uses the alternate program map to fetch the operand. If the MEM is currently disabled, then a load directly from physical memory occurs.

This instruction will cause an MEM violation only if read protection rules are violated.

XMA TRANSFER MAPS INTERNALLY PER A

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

Transfers a copy of the entire contents (32 map registers) of the System map or the User map to the Port A map or the Port B map as determined by the control word in the A-register:

Bit*	Significance
15	0 = System Map 1 = User Map
0	0 = Port A Map 1 = Port B Map

*Bits 14-1 are ignored.

This instruction will always generate an MEM violation when executed in the protected mode.

XMB TRANSFER MAPS INTERNALLY PER B

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

Transfers a copy of the entire contents (32 map registers) of the System map or the User map to the Port A map or the Port B map as determined by the control word in the B-register:

Bit*	Significance
15	0 = System Map 1 = User Map
0	0 = Port A Map 1 = Port B Map

*Bits 14-1 are ignored.

This instruction will always generate an MEM violation when executed in the protected mode.

XMM TRANSFER MAPS OR MEMORY

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

Transfers a number of words either from sequential memory locations to sequential map registers or vice versa. The A-register points to the first map register to be accessed and the B-register points to the first word of a group of words (table) in sequential memory locations. The X-register indicates the number of maps (0 to 127_{10}) to be transferred. If the content of the X-register is a positive integer, words are moved from memory to map registers; if the content is a negative integer, words are moved from map registers to memory. Map registers are addressed as a contiguous space and a wraparound count from 127 to 0 can and will occur. It is the programmer's responsibility to avoid this error.

The contents of the maps are transferred in blocks of 16 registers or less. This instruction is interruptible only after each block has been completely transferred. The A-, B-, and X-registers are then reset to allow reentry at a later time. The X-register will always be zero at the completion of the instruction an the A- and B-registers will be advanced by the number of registers moved.

An attempt to load any map register in the protected mode will generate an MEM violation. An attempt to store map registers is allowed within the constraints of write protected memory.

XMS TRANSFER MAPS SEQUENTIALLY

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

Transfers a number of words to sequential map registers. The A-register points to the first register to be accessed, the B-register contains the base quantity, and the X-register indicates the number of maps (0 to 127_{10}) to be loaded. If the contents of the X-register is a positive integer, the contents of the B-register will be used as the base quantity to be loaded into the first map register. The second register will be loaded with the base quantity plus one, the third register will be loaded with the base quantity plus two, and so forth up to the number of map registers specified in the X-register. If the contents of the X-register is less than or equal to zero, an effective NOP will occur, leaving the contents of the A-, B-, and X-registers unaltered.

This instruction is interruptible after each group of 16 registers has been transferred. The A-, B-, and X-registers are then reset to allow reentry at a later time. The X-register will always be zero at the completion of the instruction and the A- and B-registers will be advanced by the number of registers moved.

An attempt to load any map register in the protected mode will generate an MEM violation.

XSA CROSS STORE A

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

Memory Address

Stores the contents of the A-register into the addressed memory location. The previous contents of the memory cell are lost; the A-register contents are not altered.

This instruction uses the alternate program map for the write operation. If the MEM is currently disabled, then a store directly into physical memory occurs.

This instruction will always cause an MEM violation when executed in the protected mode.

XSB CROSS STORE B

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

Memory Address

Stores the contents of the B-register into the addressed memory location. The previous contents of the memory cell are lost; the B-register contents are not altered.

This instruction uses the alternate program map for the write operation. If the MEM is currently disabled, then a store directly into physical memory occurs.

This instruction will always cause an MEM violation when executed in the protected mode.

4-7. INSTRUCTION EXECUTION TIMES

Table 4-3 lists the execution times required for the various DMS instructions.

4-8. SAMPLE MAP LOAD/ENABLE ROUTINE

Table 4-4 provides a sample DMS map load and enable routine. This routine begins by loading 32 registers for the System map and 32 registers for the User map and continues by setting the Port A map to the area for User number one. The Port B map is then set to point into a new area where a third User's program would be loaded. Next, the Base Page Fence is set so that the System Fence value is used. Finally, the mapping functions of the DMS are enabled and program control is transferred to the System area beginning at address 1000_8.

4-9. ADDITIONAL DMS DEFINTIONS

The following paragraphs further define the terms "alternate map" and "protected mode" and contain definitive discussions for MEM violations and DCPC operation in a DMS environment.

4-10. ALTERNATE MAP

If the system map is currently enabled, the user map is the alternate map. If the user map is currently enabled, the system map is the alternate map. The DCPC maps are never the alternate maps.

4-11. PROTECTED MODE

If the DMS and memory protect are enabled, the computer is in the protected mode. DMS will operate in the unprotected mode (DMS enabled, memory protect disabled), but none of the DMS safeguards will be operative.

4-12. MEM VIOLATIONS

The MEM violations are designed to safeguard DMS. The four types of violations are read protect, write protect, base page, and privileged instruction. Throughout the following paragraphs, references to logical memory refers to the memory address before mapping and references to physical memory refers to the memory address after mapping.

If the computer is in the protected mode and bit 11, the read protect bit, of a system or user map register equals 1, any attempt by the system or user to read from the associated memory page causes a read protect violation and the read does not occur. If the computer is in the unprotected mode, the read occurs. In either case, bit 15 of the MEM violation register will be set to 1. For example, suppose the computer is in the protected mode and the system or user map register 3 contains 4043_8. Any attempt by the system or user to read from page 43_8 using map register 3 (i.e., read from physical addresses in the 106000_8 to 107777_8 range), causes a read protect violation.

If the computer is in the protected mode and bit 10, the write protect bit, of a system or user map register equals 1, any attempt by the system or user to write onto the associated memory page causes a write protect violation and the write does not occur. If the computer is in the unprotected mode, the write occurs. In either case, bit 14 of the MEM violation register will be set to 1. For example, suppose the computer is in the protected mode and the system or user map register 3 contains 2043_8. Any attempt by the system or user to write onto page 43_8 using map register 3 (i.e., write onto physical addresses in the 106000_8 to 107777_8 range), causes a write protect violation.

If the computer is in the protected mode, any attempt by the system or user to write onto the physical base page causes a base page violation and the write does not occur. If the computer is in the unprotected mode, the write occurs. In either case, bit 13 of the MEM violation register will be set to 1. For example, suppose the computer is in protected mode, the system or user map register 0 contains 0040_8, the base page fence is set at 1000_8, and bit 10 of the MEM status register equals 1 (i.e., logical addresses below the base page fence are mapped). If the system or user attempts to write to a logical memory address of 1500_8, MEM detects that the base page addresses above the base page fence are not mapped and begins to access physical memory address 1500_8. However, MEM then detects that a write to physical base page is being attempted which causes a base page violation and the write does not occur. If the computer was in the unprotected mode, the write would have occurred. In either case, bit 13 of the MEM violation register will be set to 1. If the system or user attempts to write to a logical memory address of 500_8, MEM detects that addresses below the base page fence are mapped and begins to access physical memory address $100000_8 + 500_8 = 100500_8$ where the write will occur providing standard memory protect is not violated. Note that standard memory protect checks the logical address (i.e., 500_8), not the physical address (i.e., 100500_8). Reading from logical or physical base page will not generate a base page violation. From the previous discussion, it can be seen that a DMS memory space has its base page in two pieces which may or may not be contiguous. Regardless, the total base page available for any DMS memory space is 1024 locations. The part of the physical base page accessible by all memory spaces is also referred to as the unmapped or shared part of the base page. Note that the logical addresses of 0 and 1 access the A and B registers, respectively.

If the computer is in the protected mode, any attempt by the user to load into any MEM register, except the MEM address register, will cause a privileged instruction violation and the load will not occur. Any attempt by the system to load into any of the MEM map registers will cause a privileged instruction violation and the load will not occur. If the computer is in the unprotected mode, the load occurs. In either case, bit 12 of the MEM violation register will be set to 1. The system can always load into the MEM state register or MEM fence register. Under microprogrammed control the user can always load into the MEM state register or MEM fence register. The system or user can always load into the MEM address register, and can always read the MEM map registers. All MEM violations cause an interrupt to select code 5. Instruction SFS 5 will skip only for an MEM violation, allowing DMS interrupts to be differentiated from memory protect or parity error interrupts.

4-13. DCPC OPERATION IN A DMS ENVIRONMENT

DCPC activity disables the MEM violation logic. Therefore, the DCPC's can read or write physical memory without generating MEM violations. Note that mapping remains enabled during DCPC activity and that the base page partitioning is the same. For example, if a DCPC input transfer were aimed at logical memory addresses 0 to 77777_8, which happened to map to physical addresses 100000_8 to 177777_8, and the conditions cited in the base page examples prevailed, then the input data would be written into physical addresses 100000_8 to 100777_8, 1000_8 to 1777_8, and 102000_8 to 177777_8.

Table 4-3. Typical DMS Instruction Execution Times

INSTRUCTION	EXECUTION TIME (us)	
	High Performance Memory	High Performance Fault Control Memory
DJP,SJP	3.290	3.360
Each Indirect Level	0.875	0.980
DJS	3.710	3.850
Each Indirect Level	0.875	0.980
JRS	4.62 to 5.215	4.760 to 5.355
Each Indirect Level	0.875	0.980
LFA/B	2.100	2.100
Each Indirect Level	0.875	0.980
MBF	4.900 to 7.490	4.900 to 7.700
	+1.680/byte	+1.680/byte
MBI	4.725 to 7.385	4.725 to 7.595
	+1.680/byte	+1.680/byte
MBW	4.900 to 6.965	4.900 to 7.105
	+1.750/byte	+1.820/byte
MWF	2.520	2.590
	+1.680/word	+1.680/word
MWI	2.345	2.415
	+1.680/word	+1.680/word
MWW	2.345	2.415
	+1.750/word	+1.820/word
PAA/B (load)	42.420	44.730
(store)	35.665	36.890
PBA/B (load)	42.595	44.905
(store)	35.840	37.065
SYA/B (load)	42.245	44.555
(store)	34.510	34.580
RSA/B	1.715	1.820
RVA/B	1.680	1.820
SJS	4.130	4.235
Each Indirect Level	0.875	0.980
SSM	3.360	3.430
Each Indirect Level	0.875	0.980
UJP	3.010	3.080
Each Indirect Level	0.875	0.980
UJS	3.430	3.535
Each Indirect Level	0.875	0.980
USA/B (load)	42.315	44.625
(store)	34.580	34.650
XCA/B (no skip)	3.080	3.185
(skip)	3.500	3.675
Each Indirect Level	0.875	0.980
XLA/B	3.255	3.325
Each Indirect Level	0.875	0.980
XMA/B	28.735 to 29.225	28.805 to 29.295
XMM (from memory to map)	3.920	3.990
	+1.575/word	+1.645/word
	+0.175/16 words	+0.175/16 words
(from map to memory)	4.025	4.095
	+1.330/word	+1.330/word
	+0.175/16 words	+0.175/16 words
XMS	3.850	3.920
	+1.330/word	+1.330/word
	+0.175/16 words	+0.175/16 words
XSA/B	3.080	3.150
Each Indirect Level	0.875	0.980

Table 4-4. Sample DMS Load/Enable Routine

LABEL	OPCODE	OPERAND	COMMENTS
DMS	NOP		DMS Load/Enable Routine
	LDA	S.TABL	Load address of System Map Table
	LDB	U.TABL	Load address of User Map Table
	SYA		Load System Map from memory
	USB		Load User Map from memory
	LDA	A.TABL	Load address of Port A Map Table
	LDB	B.TABL	Load address of Port B Map Table
	PAA		Load Port A Map from memory
	PBB		Load Port B Map from memory
	LDA	FNC,I	Load fence value
	LFA		Load Base Page Fence register
	SJP	SYSTRT,I	Enable System Map and jump to operating system entry point
SYSTRT	OCT	1000	Operating system begins at 1000$_8$
FNC	DEF	SYSF	Points to System Fence
S.TABL	DEF	SYSTEM	Points to System Table
U.TABL	DEF	US01	Points to first User Table
A.TABL	DEF	US01	Points to first User Table
B.TABL	DEF	US03	Points to third User Table
SYSF	OCT	100	Fence for operating system
SYSTEM	OCT	0	System Map Table
	OCT	1	System Map Table
	OCT	2	System Map Table
⋮			
US01F	OCT	1000	Fence for first user
US01	OCT	40	First User Map Table
	OCT	41	First User Map Table
	OCT	42	First User Map Table
⋮			
US02F	OCT	2044	Fence for second user
US02	OCT	100	Second User Map Table
	OCT	101	Second User Map Table
	OCT	102	Second User Map Table
⋮			
US03F	OCT	0	Fence for third user
US03	OCT	200	Third User Map Table
	OCT	201	Third User Map Table
	OCT	202	Third User Map Table
⋮			

MICROPROGRAMMING

SECTION V

This section contains an introductory discussion of Hewlett-Packard's microprogramming techniques and development. For additional information, refer to the *HP E-Series Computer Microprogramming Reference Manual,* part no. 02109-90004.

5-1. THE MICROPROGRAMMED COMPUTER

The control section of a computer is the portion of the computer that directs and controls the other sections; i.e., the memory section, input-output section, and the arithmetic-logic section. In totally hardwired computers, the control section logic is normally "spread out" physically throughout the computer. This design approach makes it impossible to enhance the computer's instruction set without redesign. In contrast, F-Series computers have a fully microprogrammed control section, which means that the sequence in which the control functions are performed are made programmable through the use of a technique called microprogramming.

The action taken when any one of the F-Series base set of 128 assembly language instructions is executed is determined by a microprogram associated with the assembly language instruction (these microprograms reside in a special memory called control store); the control section oversees the translation and controls the execution of the microprogram. With this design approach, instruction set enhancements can be made by changing or adding to the set of microprograms that control the machine's execution. Many computers are microprogrammed; however, Hewlett-Packard has taken the concept one step further to offer the power of microprogramming to the user.

5-2. THE MICROPROGRAMMABLE COMPUTER

F-Series computer users can more fully take advantage of the computer's power by utilizing microprogramming. The microprogrammer has more instructions, a more flexible word format, more registers, and faster execution times to work with than does the assembly language programmer. The microinstruction word length is 24 bits which enables concurrent operations to be performed in a single instruction. Microprogrammers can access 12 scratch pad registers in addition to those available to the assembly language programmer and have up to 16,384 24-bit words of memory (termed control store) in which to store microprograms. Up to three levels of nested subroutines are possible in F-Series computers. The microprogrammer works in a much faster environment than does the assembly language programmer for two reasons. One, since microinstructions have access to most of the internal parts of the computer's architecture, fewer memory fetches are required to accomplish most tasks. Two, the microinstruction execution time of 175 or 280 nanoseconds is much faster than the typical assembly instruction execution time of 1 to 2 microseconds.

These capabilities are easily taken advantage of by F-Series computer users through the extensive support provided by Hewlett-Packard. Some of the more important benefits of Hewlett-Packard's microprogramming are given in the following paragraphs.

5-3. CUSTOMIZED INSTRUCTIONS

Through the use of microprogramming, the computer's assembly language instruction set can be expanded with instructions tailored for specific applications. By adding special purpose instruction sets, the general purpose computer can be uniquely adapted for a certain job and thus become very efficient at that job. F-Series users can easily design their own instructions or purchase HP-supplied instruction sets such as the Dynamic Mapping System instructions. Applications that may be profitably microcoded include arithmetic calculations, I/O device driver programs, sorts and table searches, pseudo-DCPC operations, and special IBL loaders.

Microprogramming is very similar to assembly language programming, although it is more powerful in many ways. Some knowledge of the internal structure of the computer is required, but once this knowledge is attained, the increased power and flexibility of microprogramming can ease the solution of many programming tasks. Microprograms are easily callable by assembly or higher level language programs. An extensive set of debugging aids, software analysis aids, and documentation is available to make microprogramming easy and efficient.

5-4. SYSTEM SPEED

Microprogramming often-used routines will typically decrease program execution time by factors of two to ten and sometimes by as much as twenty or more. Software routines can be made to execute at the hardware speeds of the microprogram environment and the additional registers available to the microprogrammer can serve to eliminate many time-consuming memory fetches.

5-5. MEMORY SPACE AND SECURITY

By converting software routines into microprograms, space in main memory that would normally be required for time-critical routines can be freed for other uses. The routines remain instantly callable, as opposed to routines stored in a peripheral device. Microprograms are also less accessible than conventional software which affords a higher degree of security to microcoded routines.

5-6. DEVELOPING MICROPROGRAMS

Developing microprograms is similar to developing assembly language programs; assembling and interactive debugging of microprograms is done with the aid of the HP Micro Assembler and Micro Debug Editor. Since the user will not normally want to microcode all of a certain program, some analysis is required to determine which segment(s) of the assembly language program can be most profitably converted to microcode. This analysis is easily done with the use of an HP contributed library program called the Acitvity Profile Generator (APG). The APG enables the user to determine where in a program the CPU is spending most of its time; by substituting this section of code with a microprogrammed subroutine that is callable by the assembly or higher level program, overall execution time may often be reduced.

Once the microprogrammer has determined what segment to implement in microcode, the microprogram is developed as shown in figure 5-1. The Micro Assembler program (in main memory) is used to assemble the source microprogram into an object program. Then, the object microprogram is loaded into Writable Control Store (WCS) with the aid of the Micro Debug Editor program. Interactive debugging may be performed with the aid of the Debug Editor while the object microprogram resides in WCS.

When the microprogram is fully checked out, the user may choose to have his program reside permanently in programmable Read-Only Memory (pROM) or in WCS where it may be altered programmatically. Implementation in ROM is accomplished by programming the pROM's with a pROM writer and installing the programmed ROM's in the computer. The mask tapes shown in figure 5-1 are required by the pROM writer and are generated by the software at the user's command. ROM-resident microprograms are permanent and do not have to be reloaded each time the computer is powered up; this implementation also prevents users from erroneously destroying the microprogram. The user who does not require such permanence for microprogram storage may execute his microcode from WCS. Microprograms used in this manner may be loaded with the WCS I/O utility routine and may be altered under program control to suit a variety of users.

User-written microprograms are easily accessed by assembly or higher level programs. Once the microprogram is developed and loaded into control store, it may be called in a very similar manner to a software subroutine.

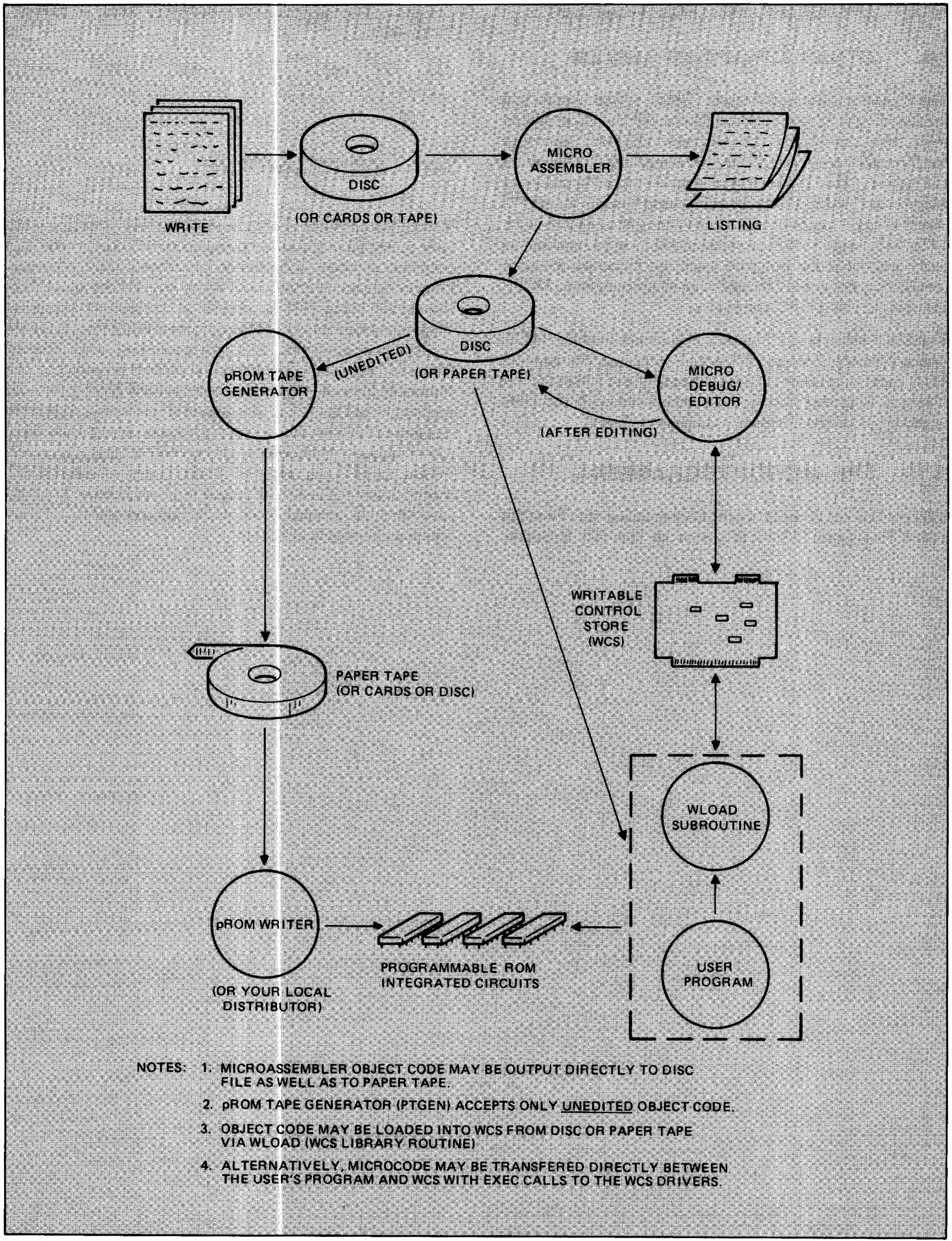
5-7. SUPPORT FOR THE MICROPROGRAMMER

Hewlett-Packard provides a comprehensive set of hardware manuals, software manuals, and training courses to make user microprogramming easy to learn and implement. For permanent implementation of microprograms, programmed pROM's may be installed in the Firmware Accessory Board or in the HP 13047A 2K User Control Store Board. The Firmware Accessory Board mounts under the main CPU board of the F-Series computer and is used to house Hewlett-Packard standard or optional instruction sets such as the Dynamic Mapping System and the D/S 1000 firmware instructions. Additional space is available on this board for mounting 4K bit and 1K bit pROM's. Up to 2,048 24-bit words of control store in the form of 1K bit pROM's may be installed in the optional 2K User Control Store Board which occupies a slot in the I/O section of the computer mainframe.

The 1K Writable Control Store (WCS) option provides a read-write control store module which can be used for the development and execution of user-supplied microprograms. Microprograms in WCS are executed at the same speed as those in the read-only control store. Each WCS module consists of a single card which plugs into the I/O PCA cage, thus eliminating the need for extensive cabling or an additional power supply. A WCS card contains 1,024 24-bit locations of Random-Access-Memory (RAM), including all necessary address and read/write circuits. WCS can be written into or read under computer control using standard input/output instructions. An I/O utility routine makes it possible for FORTRAN and ALGOL programs to write into or read from a WCS module using a conventional subroutine call. A WCS module is read at full speed by way of a flat cable connecting it to the control section of the processor.

Available microprogramming software includes the Micro Assembler and Micro Debug Editor as well as diagnostics, driver program, and I/O utility routine for use with the Writable Control Store module. These software aids operate under the Hewlett-Packard Real Time Executive (RTE) operating systems.

A course is offered at HP facilities in Cupertino, California for customer training. Requiring only a knowledge of F-Series assembly language as a prerequisite, the course features in-depth coverage of microprogram development and implementation, and provides hands-on experience for the microprogrammer. The F-Series microprogrammer may also take advantage of other user-written microprograms via the HP Contributed Library, which contains many tested and documented microprograms.



NOTES:

1. MICROASSEMBLER OBJECT CODE MAY BE OUTPUT DIRECTLY TO DISC FILE AS WELL AS TO PAPER TAPE.
2. pROM TAPE GENERATOR (PTGEN) ACCEPTS ONLY UNEDITED OBJECT CODE.
3. OBJECT CODE MAY BE LOADED INTO WCS FROM DISC OR PAPER TAPE VIA WLOAD (WCS LIBRARY ROUTINE)
4. ALTERNATIVELY, MICROCODE MAY BE TRANSFERED DIRECTLY BETWEEN THE USER'S PROGRAM AND WCS WITH EXEC CALLS TO THE WCS DRIVERS.

7114-2

Figure 5-1. Microprogram Development Cycle

5-8. OPTIONAL INSTRUCTION SETS

5-9. DYNAMIC MAPPING SYSTEM

The Dynamic Mapping System (DMS), which is standard for the HP 2117F and optional for the HP 2111F, gives the user the capability to address physical memory configurations larger than the standard 32,768 word limitation. The DMS provides a 20-bit-wide memory address bus which allows an addressing space of 1,048,576 words of main memory and allows the user to specify each 1,024-word page within physical memory to be read and/or write protected for program security. Separate memory translation maps provide isolation of system, user, DCPC channel 1, and DCPC channel 2.

The DMS consists of a Memory Expansion Module (MEM) and a Memory Protect PCA which plug into the memory PCA cage; microcode for implementing the additional 38 machine language instructions associated with the DMS is mounted in the Firmware Accessory Board.

5-10. FPP MICROPROGRAMMING

Information on directly microprogramming the Floating Point Processor (FPP) is given in the HP E-Series Computer Microprogramming Reference Manual, part no. 02109-90004.

5-11. CONCLUSION

Microprogramming is a very powerful tool that gives the user many advantages in terms of speed, flexibility, and program security. Experience has shown that microprogramming once learned, is in many cases much more flexible while being just as simple in concept as assembly language programming. Microprogramming does have its limitations however, and the potential user should examine very closely the extent of support provided by the computer manufacturer. Hewlett-Packard has by far sold and supported the greatest number of microprogrammable computers in the world, and provides world-wide customer support. Customer training courses and documentation have been refined from years of customer-contributed feedback and actual implementation is made easy through extensive software support packages and inexpensive hardware tools.

INTERRUPT SYSTEM

SECTION VI

The vectored priority interrupt system has up to 60 distinct interrupt levels, each of which has a unique priority assignment. Each interrupt level is associated with a numerically corresponding interrupt location in memory.

Of the 60 interrupt levels, the two highest priority levels are reserved for hardware faults (power fail and parity error), the next two are reserved for Dual-Channel Port Controller completion interrupts, and the remaining levels are available for I/O device channels. Tables 6-1 and 6-2 list the interrupt levels in priority order for the HP 2111F and HP 2117F Computers, respectively.

Table 6-1. HP 2111F Interrupt Assignments

CHANNEL (Octal)	INTERRUPT LOCATION	ASSIGNMENT
04	00004	Power Fail Interrupt
05	00005	Memory Parity/Memory Protect/ DMS Interrupt
06	00006	DCPC Channel 1 Completion Interrupt
07	00007	DCPC Channel 2 Completion Interrupt
10	00010	I/O Device (highest priority)
11 - 20	00011-00020	I/O Device (Mainframe)
21 - 42	00021-00042	I/O Device (Extender No. 1)
43 - 64	00043-00064	I/O Device (Extender No. 2)

Table 6-2. HP 2117F Interrupt Assignments

CHANNEL (Octal)	INTERRUPT LOCATION	ASSIGNMENT
04	00004	Power Fail Interrupt
05	00005	Memory Parity/Memory Protect/ DMS Interrupt
06	00006	DCPC Channel 1 Completion Interrupt
07	00007	DCPC Channel 2 Completion Interrupt
10	00010	I/O Device (highest priority)
11 - 25	00011-00025	I/O Device (Mainframe)
26 - 47	00026-00047	I/O Device (Extender No. 1)
50 - 71	00050-00071	I/O Device (Extender No. 2)

As an example of the simplicity of the interrupt system, an interrupt request from I/O channel 12 will cause an interrupt to memory location 00012. This request for service will be granted on a priority basis higher than afforded to channel 13 but lower than that afforded to channel 11. Thus, a transfer in progress via channel 13 would be suspended to allow channel 12 to proceed. On the other hand, a transfer in progress via channel 11 cannot be interrupted by channel 12.

Any device can be selectively enabled or disabled under program control, thus switching the device into or out of the interrupt structure. In addition, the entire interrupt system, except power fail and parity error interrupts, can be enabled or disabled under program control using a single instruction.

Interrupt requests received while the computer is in the halt mode will be processed, in order of priority, when the computer is placed in the run mode. Input/output priority is covered in more detail in Section VII.

6-1. POWER-FAIL INTERRUPT

The computer is equipped with power-sensing circuits. When primary line power fails or drops below a predetermined operating level while the computer is running, an interrupt to memory location 00004 is automatically generated. This interrupt is given the highest priority in the system and cannot be turned off or otherwise disabled. Memory location 00004 is intended to contain a jump-to-subroutine (JSB) instruction referencing the entry point of a power fail subroutine; however, location 00004 may alternatively contain a halt (HLT) instruction. The interrupt cabability of lower-priority operations is automatically inhibited while a power fail subroutine is in process.

A minimum of 500 microseconds is available between the detection of a power failure and the loss of usable power supply power to execute a power fail subroutine; the purpose of such a subroutine is to transfer the current state of the computer system into memory and then halt the computer. A sample power fail subroutine is given in table 6-3. The optional battery will supply enough power to preserve the contents of memory for a sustained line power outage of up to 2 hours.

If the Dynamic Mapping System (DMS) is installed and a power failure occurs, the System Map is automatically enabled just prior to fetching the instruction in location 00004. Since all maps are disabled and none are considered valid upon the restoration of power, the power

Table 6-3. Sample Power Fail Subroutine

LABEL	OPCODE	OPERAND	COMMENTS
PFAR	NOP		Power Fail/Auto Restart Subroutine
	SFC	4B	Skip if interrupt was caused by a power failure
	JMP	UP	Power is being restored, reset state of computer system
DOWN	STA	SAVA	Save A-register contents
	CCA		Set switch indicating that the computer was running
	STA	SAVR	when power failed
	STB	SAVB	Save B-register contents
	ERA,ALS		Transfer E-register content to A-register bit 15
	SOC		Increment A-register if Overflow
	INA		is set
	STA	SAVEO	Save E- and O-register contents
	LDA	PFAR	Save contents of P-register at time of
	STA	SAVP	power failure
	LIA	1B	Save contents of
	STA	SAVS	S-register
	STX	SAVX	Save contents of X-register
	STY	SAVY	Save contents of Y-register
	.		Insert user-written routine to save I/O
	:		device states
	CLC	4B	Turn on restart logic so computer will restart when power is restored
			after momentary power failure
	HLT	4B	Shutdown
UP	LDA	SAVR	Was computer running
	SZA,RSS		when power failed?
	HLT	4B	No
	CLA		Yes, reset computer Run switch to
	STA	SAVR	initial state
	LDA	FENCE	Restore the memory protect
	OTA	5B	fence register contents
	.		Insert user-written routine to restore
	:		I/O device states
	LDA	SAVEO	Restore the contents
	CLO		of the
	SLA,ELA		E-register and
	STF	1B	O-register
	LDA	SAVS	Restore the contents of the
	OTA	1B	S-register
	LDA	SAVA	Restore A-register contents
	LDB	SAVB	Restore B-register contents
	LDX	SAVX	Restore X-register contents
	LDY	SAVY	Restore Y-register contents
	STC	4B	Reset power fail logic for next power failure
	JMP	SAVP,I	Transfer control to program in execution at time of power failure
FENCE	OCT	2000	Fence address storage (must be updated each time fence is changed)
SAVEO	OCT	0	Storage for E and O
SAVA	OCT	0	Storage for A
SAVB	OCT	0	Storage for B
SAVS	OCT	0	Storage for S
SAVX	OCT	0	Storage for X
SAVY	OCT	0	Storage for Y
SAVP	OCT	0	Storage for P
SAVR	OCT	0	Storage for Run switch

fail subroutine should include the necessary instructions to save as many maps as desired and restore them prior to enabling the DMS.

Since the computer might be unattended by an operator, the user has a switch-selectable option of what action the computer will take upon the restoration of primary power. When the switch (A1S2) is set to the $\overline{\text{ARS}}$ position, the computer will halt when power is restored regardless of whether the computer was running or halted when the failure occurred. (No operator panel indication is given.)

Note: Switch A1S2 is mounted on the CPU and is not considered an operator control. The setting of this switch is normally determined prior to or during system installation.

When A1S2 is in the ARS position, the automatic restart feature is enabled. After a built-in delay of about half a second following the return to normal power levels, another interrupt to location 00004 occurs. This time the power-down portion of the subroutine is skipped and the power-up portion begins. (Refer to table 6-3). If the computer was not running when the power failure occurred, the computer is halted immediately. If the computer was running, those conditions existing at the time of the power fail interrupt are restored and the computer continues the program from the point of the interruption. Alternatively, if location 00004 contains a HLT instruction instead of a JSB instruction, the computer will halt and light the POWER FAIL indicator.

To allow for the possibility of a second power failure occurring while the power-up portion of the subroutine is in process, the user should limit the combined power-down and power-up instructions to less than 100. If the computer memory does not contain a subroutine to service the interrupt, location 00004 should contain a HLT 04 instruction (102004 octal).

A Set Control instruction (STC 04) must be given at the end of any restart routine. This instruction re-initializes the power-fail logic and restores the interrupt capability to the lower priority functions. Pressing the PRESET switch on the operator panel performs the same function as the STC 04 instruction. Pressing and holding the PRESET switch will force a halt when the LOCK/OPERATE switch is set to OPERATE.

The optional battery sustains the contents of memory when the line power is off. If the battery becomes discharged when the line power is off, the contents of memory will be lost. When power is restored, the computer will initiate a "Cold Start-up" and clear memory.

6-2. PARITY ERROR INTERRUPT

Parity checking of memory is a standard feature in the computer. The parity logic continuously generates correct parity for all words written into memory and monitors the parity of all words read out of memory. Correct parity is defined as having the total number of "1" bits in a 17-bit memory word (16 data bits plus the parity bit) equal to an odd value. If a "1" bit (or any odd number of "1" bits) is either dropped or added in the transfer process, a Parity Error signal is generated when that word is read out of memory.

The Parity Error signal may either halt the computer or cause the computer to take some other action as determined by an internal switch (A1S1) mounted on the CPU. When the switch is in the HALT position and a parity error occurs, the computer will halt and light the PARITY indicator. The PARITY indicator will remain lighted until the PRESET switch is pressed.

Note: Switch A1S1 is mounted on the CPU and is not considered an operator control. The setting of this switch is normally determined prior to or during installation or when the memory protect PCA is installed at the user's site.

If switch A1S1 is in the INT/IGNORE position, the action that the computer will take when a parity error occurs is as follows:

a. If the memory protect PCA is installed and the parity error logic has not been disabled by a CLF 05 instruction, an interrupt to memory location 00005 is generated. This location may contain a JSB instruction referencing the entry point of a user-written memory protect subroutine, or alternatively contain a HLT instruction.

b. If the memory protect PCA is not installed, or if the memory protect option is installed but the parity error logic has been disabled by a CLF 05 instruction, the parity error will be ignored and the PARITY indicator will light.

In conjunction with memory protect, it is possible to determine the memory address containing the parity error. The error address will be loaded automatically into the violation register of the memory protect logic and from there it is accessible to the user by programming an LIA 05 or LIB 05 instruction.

When a parity error occurs, it is recommended that the entire program or set of data containing the error location be reloaded. However, by knowing the address and the contents of the error location, the user may be able to determine what operations have taken place as a result of reading the erroneous word. For example, if the erroneous word was an instruction, several other locations may be affected. By individually checking and correcting the contents of all affected memory locations, the user may resume running the program without the necessity of a complete reload. If software is being generated, this may also need correcting.

6-3. MEMORY PROTECT/DMS INTERRUPT

The memory protect option provides the capability of protecting a selected block of memory of any size, from a settable fence address downward, against alteration or entry by programmed instructions.

The memory protect logic, when enabled by an STC 05 instruction, also prohibits the execution of all I/O instructions (including HLT 01) except those referencing I/O select code 01 (the S-register and the overflow register). This feature limits the control of I/O operations to interrupt control only. Thus, an executive program residing in protected memory can have exclusive control of the I/O system.

The memory protect logic is disabled automatically by any interrupt (except when the interrupt location contains an I/O instruction) and must be re-enabled by an STC 05 instruction at the end of each interrupt subroutine.

The optional DMS hardware includes additional memory protect features, which are enabled or disabled simultaneous with the memory protect hardware. When enabled by an STC 05 instruction, the DMS hardware provides the capability of read/write protecting memory on a 1024-word page basis. Included in the DMS are several privileged instructions which are not allowed when the memory protect logic is enabled. Upon detection of a violation, an interrupt to location 00005 is generated. Since the DMS will set the flag on channel 05, executing either an SFS 05 or an SFC 05 instruction will permit the programmer to know whether the DMS or memory protect interrupted.

Programming rules pertaining to the use of memory protect are as follows (assuming that an STC 05 instruction has been given):

a. The upper protected memory boundary address is loaded into the fence register from the A- or B-register by an OTA 05 or OTB 05 instruction, respectively. Memory addresses below but not including this address are protected.

b. Execution will be inhibited and an interrupt to location 00005 will occur if one of the following instructions either directly or indirectly modifies or enters a location in protected memory, or if any I/O instruction is attempted (including HLT but excluding those I/O instructions addressing select code 01).

DST	ISZ	JLY	JMP	JPY	JSB
MVB	MVW	SAX	SAY	SBX	SBY
STA	STB	STX	STY		

c. Location 00002 is normally the lower boundary of protected memory. (Locations 00000 and 00001 are the A- and B-register addresses and may be freely addressed.) JMP, JLY, and JPY instructions may not reference the A- or B-register.

d. After three successive levels of indirect addressing, the memory protect logic will allow a pending I/O interrupt if the memory protect logic is installed.

e. Any instructions not mentioned in step b of this paragraph is legal even if the instruction directly references a protected memory address. In addition, indirect addressing through protected memory by those instructions listed in step b is legal provided that the ultimate effective address is outside the protected memory area.

Following a memory protect interrupt, the address of the illegal instruction will be present in the violation register. This address is made accessible to the programmer by an LIA 05 or LIB 05 instruction, which loads the address into the A- or B-register.

Since parity error and memory protect share the same interrupt location, it is necessary to distinguish which type of error is responsible for the interrupt. A parity error is indicated if, after the LIA (or LIB) 05 instruction is executed, bit 15 of the selected register is a logic 1; a memory protect violation is indicated if bit 15 is a logic 0. In either case, the remaining 15 bits of the selected register contains the logical address of the error location.

Table 6-4 illustrates a sample memory protect, DMS, and parity error subroutine. An assumption made for this example is that the location immediately following the error location is an appropriate return point. This may not always be the case, however, because it may be deemed advisable to abort the program in process and return to a supervisory program.

Table 6-4. Sample Memory Protect, Parity Error, and DMS Subroutine

LABEL	OPERCODE	OPERAND	COMMENTS
MPEDM	NOP		Memory Protect/Parity Error/DMS Subroutine
	CLF	0B	Turn off interrupt system
	STA	SAVA	Save A-register contents
	STB	SAVB	Save B-register contents
	LIA	5B	Get contents of violation register
	CLF	5B	Turn off parity error interrupts
	SFC	5B	Check flag for DMS violation
	JMP	DMS	If flag is set, then DMS interrupted
	SSA		Check bit 15 of violation register
	JMP	PE	If bit 15 is set, then parity error occurred
	JMP	MP	If bit 15 is clear, then memory protect interrupted
MP	—		User's routine for memory protect violation
	—		
	—		
	etc.		
	—		
	—		
	JMP	REST	
PE	—		User's routine for parity error condition
	—		
	—		
	etc.		
	—		
	—		
	JMP	REST	
DMS	—		User's routine for DMS violation
	—		
	—		
	etc.		
	—		
	—		
	JMP	REST	
REST	LDA	SAVA	Restore A-register
	LDB	SAVB	Restore B-register
	STF	0B	Enable interrupt system
	STF	5B	Enable parity error interrupt
	STC	5B	Turn on memory protect
	JMP	MPEDM,I	Exit
SAVA	OCT	0	Storage for A
SAVB	OCT	0	Storage for B

6-4. DUAL-CHANNEL PORT CONTROLLER INTERRUPT

The optional Dual-Channel Port Controller (DCPC) allows high-speed block transfer of data between input/output devices and memory. For the most part, the DCPC operates independently of the interrupt system in that the only time that a DCPC interrupt occurs is when the specified block of data has been transferred. Since there are two DCPC channels, two interrupt locations are reserved for this purpose; location 00006 is reserved for channel 1 and location 00007 is reserved for channel 2. Channel 1 interrupt has priority over the channel 2 interrupt. Because DCPC interrupts are primarily completion signals to the programmer, and are therefore application dependent, no interrupt subroutine example is considered necessary.

6-5. INPUT/OUTPUT INTERRUPT

The remaining interrupt locations (00010 through 00077 octal) are reserved for I/O devices; this represents a total of 56 (decimal) locations, one for each I/O channel. In a typical I/O operation, the computer issues a programmed command such as Set Control/Clear Flag (STC,C) to one or more external devices to initiate an input (read) or an output (write) operation. Each device will then either put data into or accept data from an input/output buffer on its associated interface PCA. During this time, the computer may continue running a program or may be programmed into a waiting loop to wait for a specific device to complete a read or write operation. Upon the completion of a read or write operation, each device returns a Flag signal to the computer. These Flag signals are passed through a priority network which allows only one device to be serviced regardless of the number of Flag signals present at that time. The Flag signal with the highest priority generates an Interrupt signal at the end of the current machine cycle except under the following circumstances:

a. Interrupt system disabled or interface PCA interrupt disabled.

b. JMP indirect or JSB indirect instruction not sufficiently executed. These instructions inhibit all interrupts except power fail or memory protect until the succeeding instruction is executed. After three successive levels of indirect addressing, the memory protect logic will allow a pending I/O interrupt if the memory protect logic is installed.

c. Instruction in an interrupt location not sufficiently executed, even if that interrupt is of lower priority. Any interrupt inhibits the entire interrupt system until the succeeding instruction is executed.

d. Optional dual-channel port controller in the process of transferring data.

e. Current instruction is one that may affect the priorities of I/O devices; e.g., STC, CLC, STF, CLF, SFS, and SFC. The interrupt in this case must wait until the succeeding instruction is executed. the SFS instruction used with the interrupt system on produces special conditions. Since the SFS instruction holds off interrupts until the next instruction is executed, if the next instruction clears the device flag, then it should also remove the interrupt request. Therefore, a CLF instruction should be used rather than appending ,C (sets bit 9 in some I/O instructions) to the instruction.

After an interface PCA has been issued a Set Control command and its Flag flip-flop becomes set, all interrupt requests from lower-priority devices are inhibited until this Flag flip-flop is cleared by a Clear Flag (CLF) instruction. A service subroutine in process for any device can be interrupted only by a higher-priority device; then, after the higher-priority device is serviced, the interrupted service subroutine may continue. In this way it is possible for several service subroutines to be in the interrupt state at one time; each of these service subroutines will be allowed to continue after the higher-priority device is serviced. All such service subroutines normally end with a JMP indirect instruction to return the computer to the point of the interrupt.

6-6. CENTRAL INTERRUPT REGISTER

Each time an interrupt occurs, the address of the interrupt location is stored in the central interrupt register. The contents of this register are accessible at any time by executing an LIA 04 or LIB 04 instruction. This loads the address of the most recent interrupt into the A- or B-register. As described in section II, the central interrupt register contents can be accessed from the operator panel.

6-7. INTERRUPT SYSTEM CONTROL

I/O address 00 is the master control address for the interrupt system. An STF 00 instruction enables the entire interrupt system and a CLF 00 disables the interrupt system. The two exceptions to this are the power fail interrupt, which cannot be disabled, and parity error interrupt, which can only be selective enabled or disabled by an STF 05 or CLF 05, respectively. Whenever power is initially applied, the interrupt system is disabled.

INPUT/OUTPUT SYSTEM

SECTION VII

The purpose of the input/output system is to transfer data between the computer and external devices. As shown in figure 7-1, data is normally transferred through the A- or B-register. An input transfer of this type occurs in three distinct steps: (1) between the external device and its interface PCA in the computer, (2) between the interface PCA and the A- or B-register via the I/O bus and CPU, and (3) between the A- or B-register and memory via the S-bus and memory controller. This three-step process also applies to an output transfer except in reverse order. This type of transfer, which is executed under program control, allows the computer logic to manipulate the data during the transfer process.

Also shown in figure 7-1, data may be transferred automatically under control of the Dual-Channel Port Controller (DCPC) option. Once the DCPC has been initialized, no programming is involved and the transfer is reduced to a two-step process: (1) between the external device and its interface PCA in the computer and (2) between the interface PCA and memory via the I/O bus, S-bus, and memory controller. The two DCPC channels are assignable to operate with any two device interface PCA's.

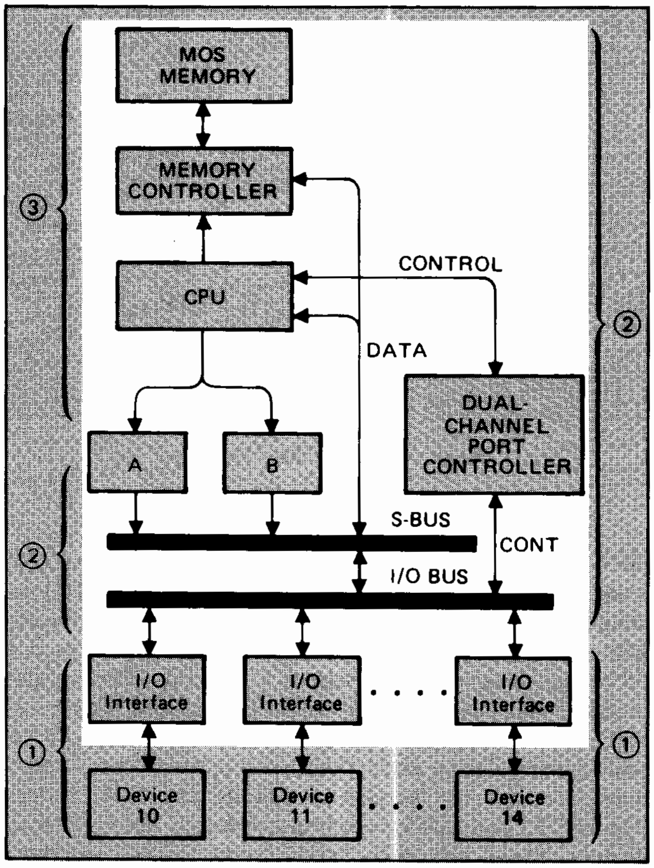


Figure 7-1. Input/Output System

Since a DCPC transfer eliminates programmed loading and storing via the accumulators, the time involved is very short. Thus, the DCPC is used with high-speed devices. Further information on the DCPC option is given under paragraph 7-13.

7-1. INPUT/OUTPUT ADDRESSING

As shown in figure 7-2, an external device is connected by cable directly to an interface PCA located inside the computer mainframe. The interface PCA, in turn, plugs into one of the input/output slots, each of which is assigned a fixed address commonly referred to as the device select code. The computer can then communicate with a specific device on the basis of its select code.

Figure 7-2 shows an interface PCA inserted in the I/O slot having the highest priority; this channel is assigned select code 10 (octal). If it is decided that the associated device should have lower priority, its interface PCA and cable may simply be exchanged with those occupying some other I/O slot. This will change both the priority and the I/O address; however, due to priority chaining (refer to paragraph 7-2), there can be no vacant slots from select code 10 to the highest used select code (if the interrupt mode is to be used).

Only select codes 10 through 77 (octal) are available for input/output devices; the lower select codes (00 through 07) are reserved for other features. Figure 7-2 illustrates the I/O select codes available in the HP 2111F and HP 2117F Computer mainframes.

Select codes (channels) higher than those shown in figure 7-2 are available through the use of one or two I/O extenders. Each I/O extender provides an additional 16 I/O channels, which are an extension of the computer's vectored priority interrupt system. Select codes in the extender(s) operate at the same speed and with the same versatility as those in the computer mainframe.

7-2. INPUT/OUTPUT PRIORITY

When a device is ready to be serviced, it causes its interface PCA to request an interrupt so that the computer will interrupt the current program and service the device. Since many device interface PCA's will be requesting service at random times, it is necessary to establish an orderly sequence for granting interrupts. Secondly, it is desirable that high-speed devices should not have to wait for low-speed device transfers. Both of these requirements are met by a series-linked priority structure illustrated by

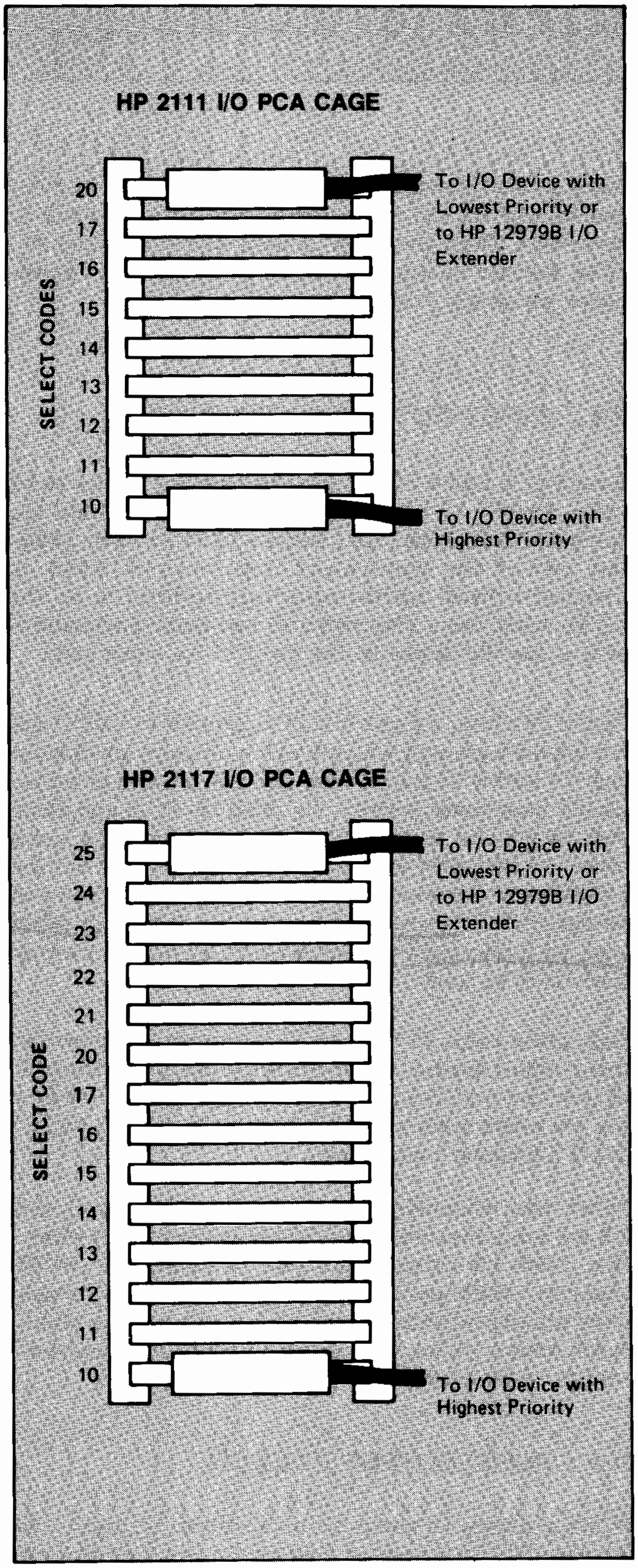

HP 2111 I/O PCA CAGE
SELECT CODES
20
17
16
15
14
13
12
11
10
To I/O Device with Lowest Priority or to HP 12979B I/O Extender
To I/O Device with Highest Priority
HP 2117 I/O PCA CAGE
SELECT CODE
25
24
23
22
21
20
17
16
15
14
13
12
11
10
To I/O Device with Lowest Priority or to HP 12979B I/O Extender
To I/O Device with Highest Priority

Figure 7-2. I/O Address Assignments

figure 7-3. The bold line, representing a priority enabling signal, is routed in series through each PCA capable of causing an interrupt. The PCA cannot interrupt unless this enabling signal is present at its input.

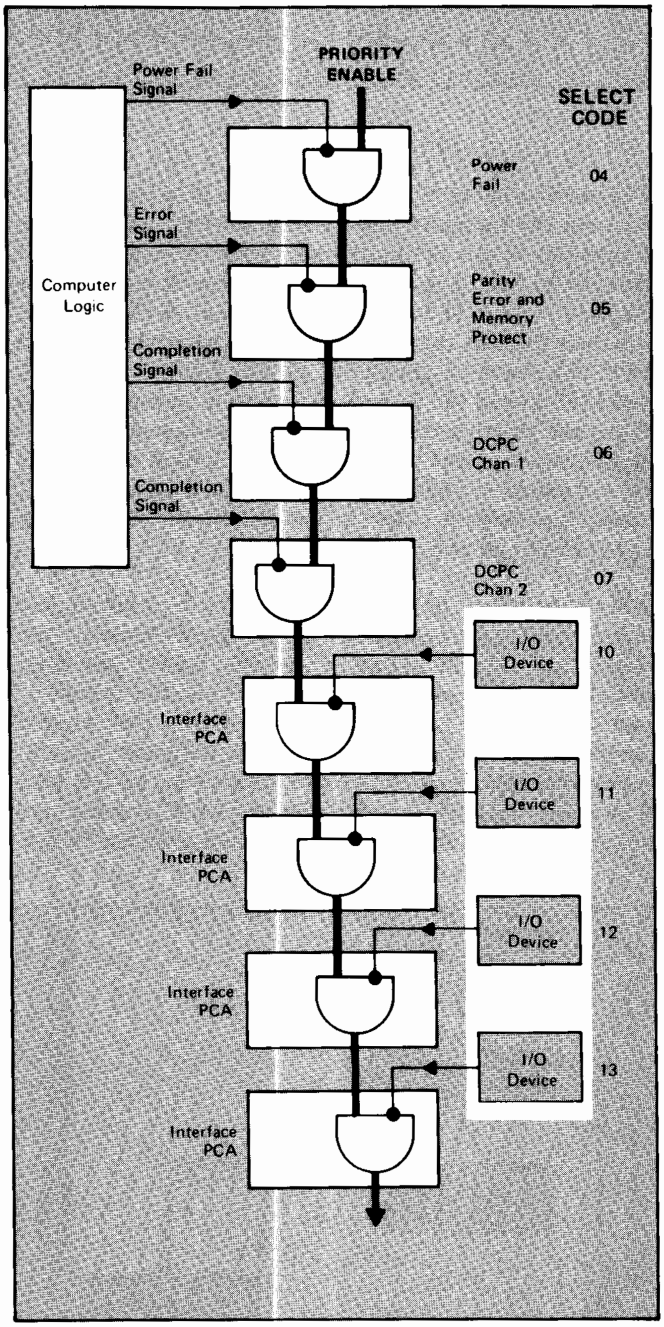


Figure 7-3. Priority Linkage

Each device (or other interrupt function) can break the enabling line when it requests an interrupt. If two devices simultaneously request an interrupt, obviously the device with the lowest select code will be the first one that can interrupt because it has broken the enable line for the higher select code. The other device cannot begin its service routine until the first device is finished; however, a still higher priority device (one with a lower select code) may interrupt the service routine of the first device. Figure 7-4 illustrates a hypothetical case in which several devices require service by interrupting a CPU program. Both simultaneous and time-separated interrupt requests are considered.

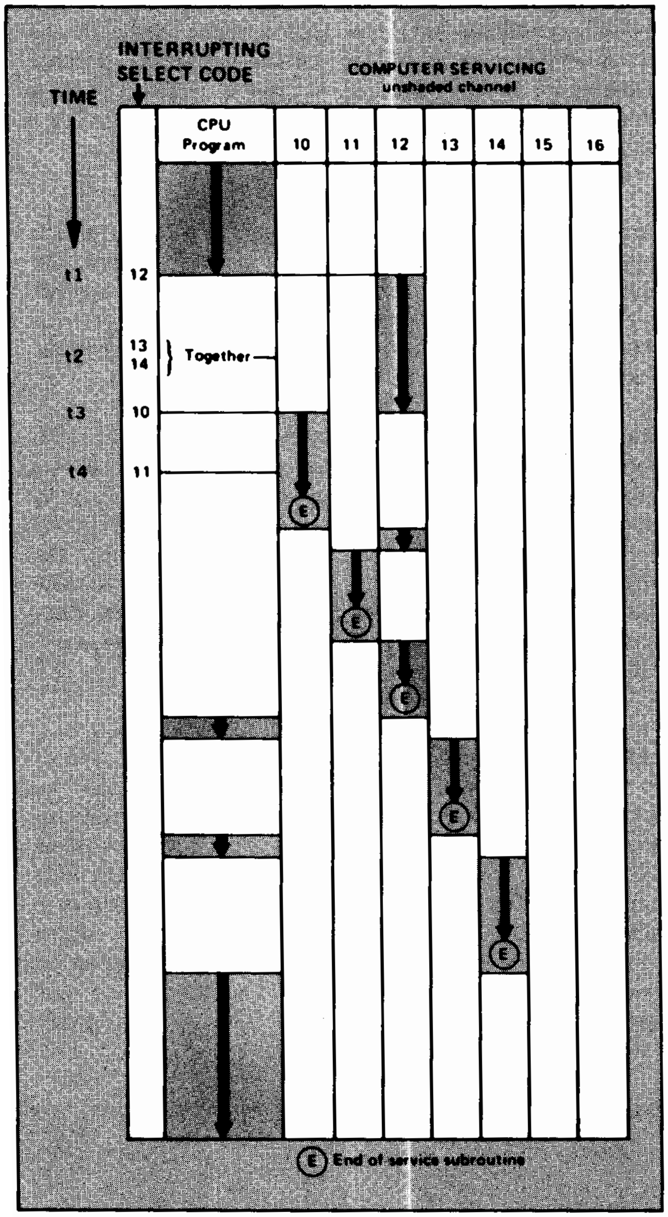


Figure 7-4. Interrupt Sequences

Assume that the computer is running a CPU program when an interrupt from I/O channel 12 occurs (at reference time t1). A JSB instruction in the interrupt location for select code 12 causes a program jump to the service routine for the channel 12 device. The JSB instruction automatically saves the return address (in a location which the programmer must reserve in his routine) for a later return to the CPU program.

The routine for channel 12 is still in progress when several other devices request service (set flag). First, channels 13 and 14 request simultaneously at t2; however, since neither one has priority over channel 12, their flags are ignored and channel 12 continues its transfer. But at t3, a higher priority device on channel 10 requests service. This request interrupts the channel 12 transfer and causes the channel 10 transfer to begin. The JSB instruction saves the return address for return to the channel 12 routine. During the channel 10 transfer, device 11 sets the channel 11 flag (t4). Since it has lower priority than channel 10, device 11 must wait until the end of the channel 10 routine. And since the channel 10 routine, when it ends, contains a return address to the channel 12 routine, program control temporarily returns to channel 12 (even though the waiting channel 11 has higher priority). The JMP,I instruction used for the return inhibits all interrupts until fully executed. At the end of this short interval, the channel 11 interrupt request is granted.

When channel 11 has finished its routine, control is returned to channel 12, which at last has sufficient priority to complete its routine. Since channel 12 has been saving a return address in the main CPU program, it returns control to this point.

The two waiting interrupt requests from channels 13 and 14 are now enabled. Channel 13 has the higher priority and goes first. At the end of the channel 13 routine, control is temporarily returned to the CPU program. Then, the lowest priority channel (channel 14) interrupts and completes its transfer. Finally, control is returned to the CPU program, which resumes processing.

7-3. INTERFACE ELEMENTS

The interface PCA provides the communication link between the computer and an external device. The interface PCA includes three basic elements which either the computer or the device can control in order to effect the necessary communication. These three elements are the control bit, flag bit, and buffer.

7-4. CONTROL BIT

This is a one-bit register used by the computer to turn on the device channel. When set, the control bit generates a start command to the device, allowing it to perform one operation cycle (e.g., read or write one character or word). The interface PCA cannot interrupt unless the control bit is set. The control bit is set by an STC (set control) instruction and cleared by a CLC (clear control) instruction, both of which must be accompanied by a specific select code (e.g., STC 12 or CLC 12). The device cannot affect the control bit.

7-5. FLAG BIT

This is a one-bit register primarily used by the device to indicate (when set) that a transmission between the device and the interface PCA buffer has been completed. Computer instructions can also set the flag (STF), clear the flag (CLF), test if it is set (SFS), and test if it is clear (SFC). The device cannot clear the flag bit. If the corresponding control bit is set, priority is high, and the interrupt system is enabled, setting the flag bit will cause an interrupt to the location corresponding to the device select code.

7-6. BUFFER

The buffer register is used for intermediate storage of data. Typically, the data capacity is 8 or 16 bits, but this is entirely dependent on the type of device.

7-7. INPUT/OUTPUT DATA TRANSFER

The following paragraphs describe how data is transferred between memory and input/output devices. A summary of I/O group instructions pertinent to the computer interrupt and control functions is provided in the appendix. The sequences presented for interrupt and noninterrupt methods of data transfer are highly simplified in order to present an overall view without the involvement of software operating systems and device drivers. For more detailed information, refer to the documentation supplied with the appropriate software system or I/O subsystem.

7-8. INPUT DATA TRANSFER (INTERRUPT METHOD)

Figure 7-5 illustrates the sequence of events required to input data using the interrupt method. Note that some operations are under control of the computer program (programmer's responsibility) and some of the operations are automatic. Note also that the interface PCA (device controller) is installed in the slot assigned to select code 12.

The operations begins (1) with the programmed instruction STC 12,C which sets the Control flip-flop and clears the Flag flip-flop on the interface PCA. Since the next few operations are under control of the hardware, the computer program may continue the execution of other instructions. Setting the Control flip-flop causes the PCA to output a Start signal (2) to the device, which reads out a data character and asserts the Done signal (3).

The device Done signal sets the PCA Flag flip-flop, which in turn generates an interrupt (4) assuming that the interrupt conditions are met; i.e., the interrupt system must be on (STF 00 previously given), no higher priority interrupt is pending, and the Control flip-flop is set (done in step 1).

The interrupt causes the current computer program to be suspended and control is transferred to a service subroutine (5). It is the programmer's responsibility to provide the linkage between the interrupt location (00012 in this case) and the service subroutine. It is also the programmer's responsibility to include in his service subroutine the instructions for processing the data (loading into an accumulator, manipulating if necessary, and storing into memory).

The subroutine may then issue further STC 12,C commands to transfer additional data characters. One of the final instructions in the service subroutine must be CLC 12. This step (6) restores the interrupt capability to lower priority devices and returns the interface PCA to its static "ready" condition (Control clear and Flag set). This condition is initially established by the computer at power turn-on and it is the programmer's responsibility to return the interface PCA to the same condition on the completion of each data transfer operation. At the end of the subroutine, control is returned to the interrupted program via previously established linkages.

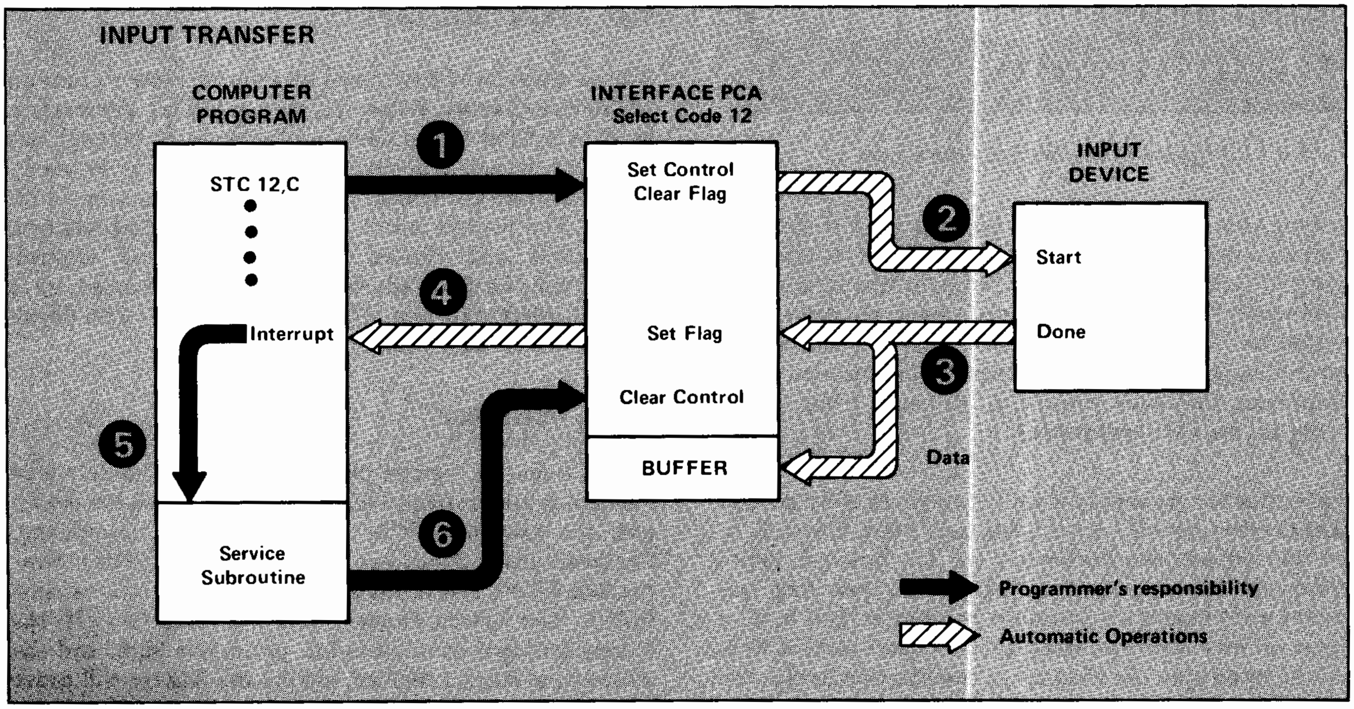


Figure 7-5. Input Data Transfer (Interrupt Method)

7-9. OUTPUT DATA TRANSFER (INTERRUPT METHOD)

Figure 7-6 illustrates the sequence of events required to output data using the interrupt method. Again note the distinction between programmed and automatic instructions. It is assumed that the data to be transferred has been loaded into the A-register and is in a form suitable for output. The interface PCA in this example is assumed to be in the slot assigned to select code 13.

The output operation begins with a programmed instruction (OTA 13) to transfer the contents of the A-register to the interface PCA buffer (1). This is followed (2) by the instruction STC 13,C which sets the Control flip-flop and clears the Flag flip-flop on the interface PCA. Since the next few operations are under control of the hardware, the computer program may continue the execution of other instructions. Setting the Control flip-flop causes the PCA to output the buffered data and a Start signal (3) to the device, which writes (e.g., punches, stores, etc.) the data character and asserts the Done signal (4).

The device Done signal sets the PCA Flag flip-flop, which in turn generates an interrupt (5) provided that the interrupt system is on, priority is high, and the Control flip-flop is set (done in step 2). The interrupt causes the current computer program to be suspended, and control is transferred to a service subroutine (6). It is the programmer's responsibility to provide the linkage between the interrupt location (00013 in this case) and the service subroutine. The detailed contents of the subroutine are also the programmer's responsibility, and the contents will vary with the type of device. The subroutine may then output further data to the interface PCA and reissue the STC 13,C command for additional data character transfers. One of the final instructions in the service subroutine must be a clear control (CLC 13). This step (7) allows lower priority devices to interrupt, and restores the channel to its static "ready" condition (Control clear and Flag set). At the end of the subroutine, control is returned to the interrupted program via previously established linkages.

7-10. NONINTERRUPT DATA TRANSFER

It is also possible to transfer data without using the interrupt system. This involves a "wait-for-flag" method in which the computer commands the device to operate and then waits for the completion response. In using this method to transfer data, it is assumed that the computer time is relatively unimportant. The programming is very simple, consisting of only four words of in-line coding as shown in table 7-1. Each of these routines will transfer, one word or character of data. It is also assumed that the interrupt system is turned off (STF 00 not previously given).

7-11. INPUT. As described under paragraph 7-8, an STC 12,C instruction begins the operation by commanding the device to read one word or character. The computer then goes into a waiting loop, repeatedly checking the status of the flag bit. If the Flag flip-flop is not set, the JMP *-1 instruction causes a jump back to the SFS instruction. (The *-1 operand is assembler notation for "this location minus one.") When the Flag flip-flop is set, the skip condition for SFS is met and the JMP instruction is skipped. The computer thus exits from the waiting loop

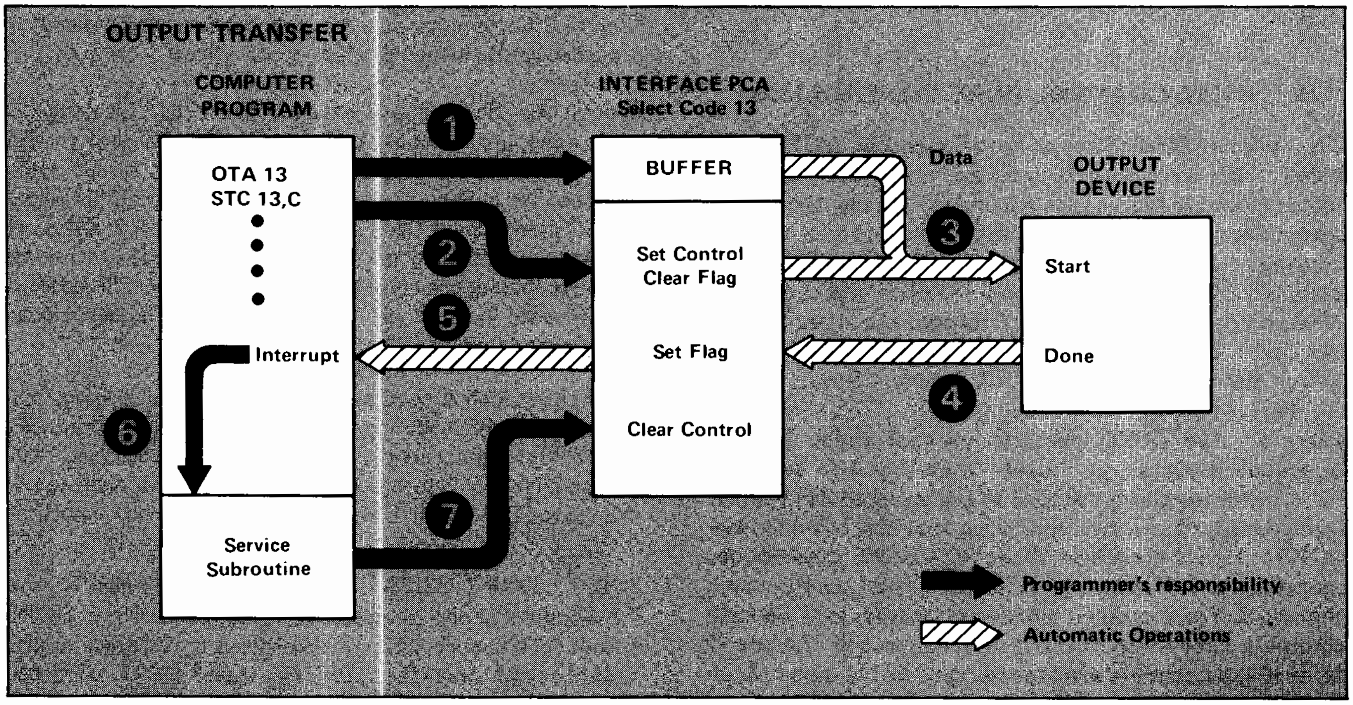


Figure 7-6. Output Data Transfer (Interrupt Method)

and the LIA 12 instruction loads the device input data into the A-register.

Table 7-1. Noninterrupt Transfer Routines

INPUT

INSTRUCTIONS	COMMENTS
STC 12,C	Start device
SFS 12	Is input ready?
JMP *-1	No, repeat previous instruction
LIA 12	Yes, load input into A-register

OUTPUT

INSTRUCTIONS	COMMENTS
OTA 13	Output A-register to buffer
STC 13,C	Start device
SFS 13	Has device accepted the data?
JMP *-1	No, repeat previous instruction
NOP	Yes, proceed

7-12. OUTPUT. The first step, which is to transfer the data to the interface PCA buffer, is the OTA 13 instruction. Then STC 13,C commands the device to operate and accept the data. The computer then goes into a waiting loop as described in the preceding paragraph. When the Flag flip-flop becomes set, indicating that the device has accepted the output data, the computer exits from the loop. (The final NOP is for illustration purposes only.)

7-13. DUAL-CHANNEL PORT CONTROLLER

The optional Dual-Channel Port Controller (DCPC) provides a direct data path, software assignable, between memory and a high-speed peripheral device; the DCPC accomplishes this by stealing an I/O cycle instead of interrupting to a service subroutine. The DCPC logic is capable of stealing every consecutive I/O cycle and can transfer input data at rates up to 1.0 million words per second; see Direct Memory Access specifications in table 1-1 for output data rates and the DCPC latency times.

There are two DCPC channels, each of which may be separately assigned to operate with any I/O interface PCA, including those installed in the optional HP 12979B Input/Output Extender (assuming that the I/O extender DCPC option is installed). When both DCPC channels are operating simultaneously, channel 1 has priority over channel 2. The combined maximum transfer rate for both channels operating concurrently can reach the above rates if the channels are transferring in the same direction. Channels working in opposite directions will achieve an effective transfer rate between these rates.

Since the memory cycle rate is somewhat faster than the I/O cycle rate, it is possible for the CPU to interleave memory cycles while the DCPC is operating.

Transfers via the DCPC are on a full-word basis; hardware packing and unpacking of bytes are not provided. The word count register is a full 16 bits in length, and data transfers are accomplished in blocks. The transfer is initiated by an initialization routine, and from then on the operation is under automatic control of the hardware. The initialization routine specifies the direction of the data transfer (in or out), where in memory to read or write, which I/O channel to use, and how much data to transfer. Completion of the block transfer is signalled by an interrupt to location 00006 (for channel 1) or to location 00007 (for channel 2) if the interrupt system is enabled. It is also possible to check for completion by testing the status of the flag for select code 06 or 07, or by interrogating the word count register with an LIA/B to select code 02 (for channel 1) or to select code 03 (for channel 2). A block transfer in process can be aborted with an STF 06 or 07 instruction.

7-14. DCPC INPUT DATA TRANSFER. Figure 7-7 illustrates the sequence of operations for a DCPC input data transfer. A comparison with the conventional interrupt method (figure 7-5) shows that much more of the DCPC operation is automatic. Remember that the procedure in figure 7-5 must be repeated for each word or character. In figure 7-7, the automatic DCPC operation will input a block of data of any size limited only by the available memory space. The initialization routine sets up the control registers on the DCPC (1) and issues the first start command (STC 12,C) directly to the interface PCA. The DCPC logic is now turned on and the computer program continues with other instructions.

Setting the Control and clearing the Flag flip-flops (2) causes the interface PCA to send a Start signal to the external device (3). The device goes through a read cycle and returns a Done signal with a data word. The Done signal (4) sets the PCA Flag flip-flop which, regardless of priority, immediately requests the DCPC logic to steal an I/O cycle (5) and transfer a word into memory. The process now repeats back to the beginning of this paragraph to transfer the next word.

After the specified number of words have been transferred, the interface PCA Control flip-flop is cleared (7) and the DCPC logic generates a completion interrupt (8). The program control is now forced to a completion routine (9), the content of which is the programmer's responsibility.

7-15. DCPC OUTPUT DATA TRANSFER. Figure 7-8 illustrates the sequence of operations for a DCPC output data transfer. The initialization routine sets up the control registers on the DCPC (1), starts DCPC (STC 6B, C), clears the Control flip-flop (CLC 12), and sets the Flag flip flop (STF 12) on the interface PCA (2). The DCPC logic

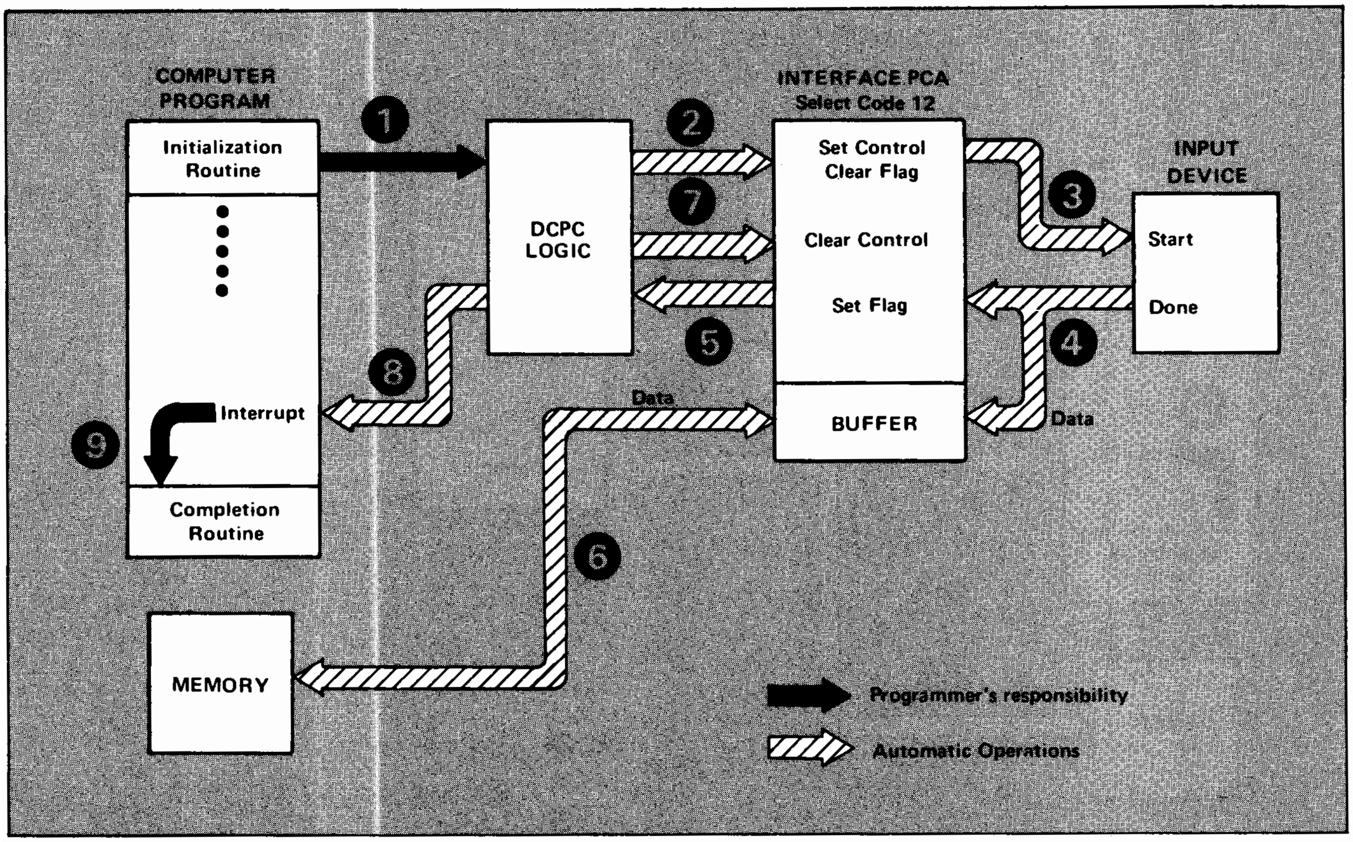


Figure 7-7. DCPC Input Data Transfer

is now enabled and the computer program continues with other instructions. Setting the Flag flip-flop causes the interface PCA to issue an SRQ signal to the DCPC (3). The DCPC transfers a word from memory to the interface PCA buffer (4), sets the Control flip-flop, and clears the Flag flip-flop (5). Setting the Control flip-flop causes a command to be sent to the device (6). The device acknowledges the command by taking the data from the interface PCA (7) and returning a Done signal (8). The Done signal sets the Flag flip-flop which causes another SRQ to be issued to the DCPC (3). This process repeats until all data words are transferred, at which time the DCPC generates an interrupt (9) and program control is transferred to the user-written completion routine (10).

Note that the initial DCPC output data transfer should *not* be started by setting the Control and clearing the Flag flip-flops (STC 12,C). If this is done the device takes a word on receipt of the device command generated by setting the Control flip-flop. Thus, the device receives an erroneous first word or two copies of the first word if the word was output to the interface PCA before the DCPC was initiated. Instead, SRQ should initiate the DCPC. Therefore, on initialization, it is important that the Flag flip-flop be set to generate the first SRQ (STF 12).

For DCPC output transfers, the clear control option of Control Word 1 (bit 13 = 1, figure 7-9) is I/O-card dependent. A condition might exist such that the last data word is not received by the device because the device command is not issued. This can occur if the CLC signal overrides the STC signal (Control Word 1, bit 15 = 1), consequently eliminating device command for the last word transfer. The relationship between the STC signal, the CLC signal, an the device command determines whether the CLC option can be used.

7-16. DCPC INITIALIZATION.

The information required to initialize the DCPC (direction, memory allocation, I/O channel assignment, and block length) are given by three control words. These three words must be addressed specifically to the DCPC. Figure 7-9 illustrates the format of the three control words. Control Word 1 (CW1) identifies the I/O channel to be used and provides two options selectable by the programmer:

Bit 15
1 = give STC (in addition to CLF) to I/O channel at end of each DCPC cycle (except on last cycle, if input)

0 = no STC

Bit 13
1 = give CLC to I/O channel at end of block transfer

0 = no CLC

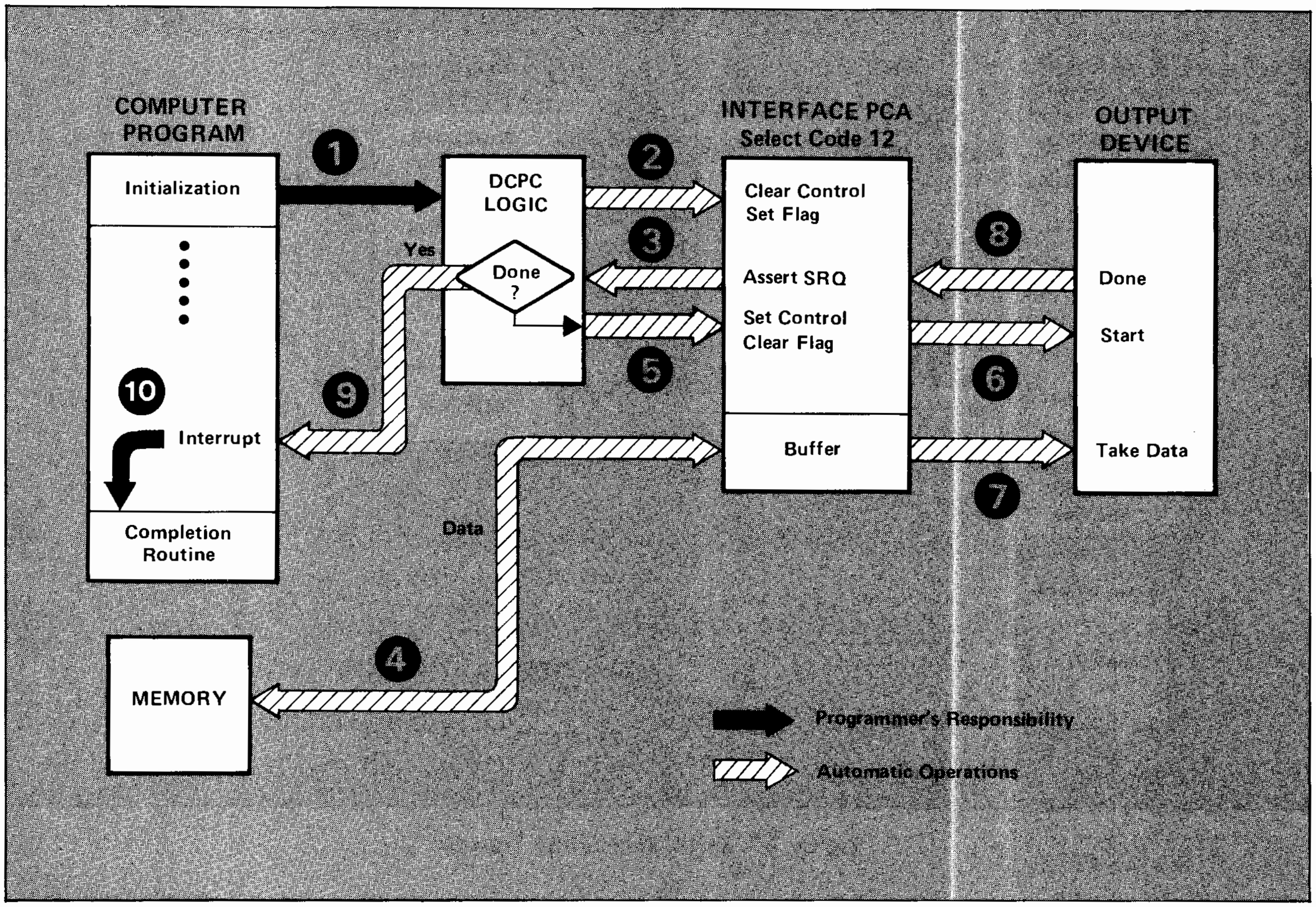


Figure 7-8. DCPC Output Data Transfer

Control Word 2 (CW2) gives the starting memory address for the block transfer and bit 15 determines whether data is to go into memory (logic 1) or out of memory (logic 0). Control Word 3 (CW3) is the two's complement of the number of words to be transferred into or out of memory (i.e., the block length). This number can be from 1 to 32,768, although it is limited in the practical case by available memory.

Table 7-2 gives the basic program sequence for outputting the control words to the DCPC. As shown in this table, CLC 2 and STC 2 perform switching functions to prepare the logic for either CW2 or CW3. The device is assumed to be in I/O slot channel 10, and it is also assumed that its start command is STC 10B, C. The sample values of CW1, CW2, and CW3 will read a block of 50 words and store these in locations 200 through 261 (octal). The STC 06B,C instruction starts the DCPC operation. A flag-status method of detecting the end-of-transfer is used in this example; an interrupt to location 00006 could be substituted for this test. The program in table 7-2 could easily be changed to operate on channel 2 by changing select codes 2 to 3 and 6 to 7.

One important difference should be noted when doing a DCPC input operation from a disc or a drum. Due to the synchronous nature of disc or drum memories and the design of the interface PCA, the order of starting must be reversed from the order given; i.e., start the DCPC first and then start the disc (or drum).

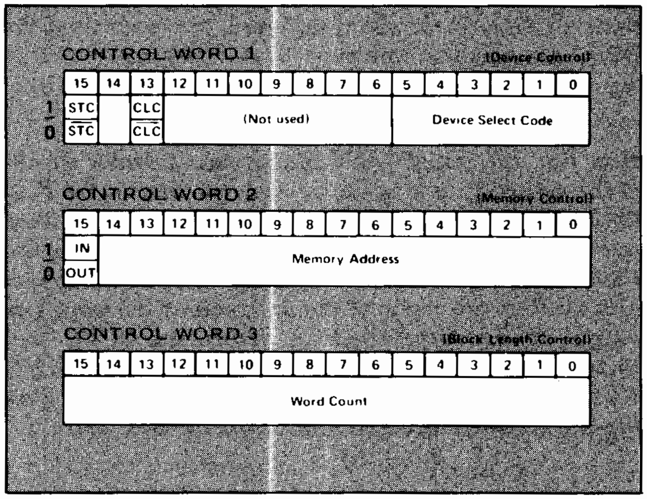


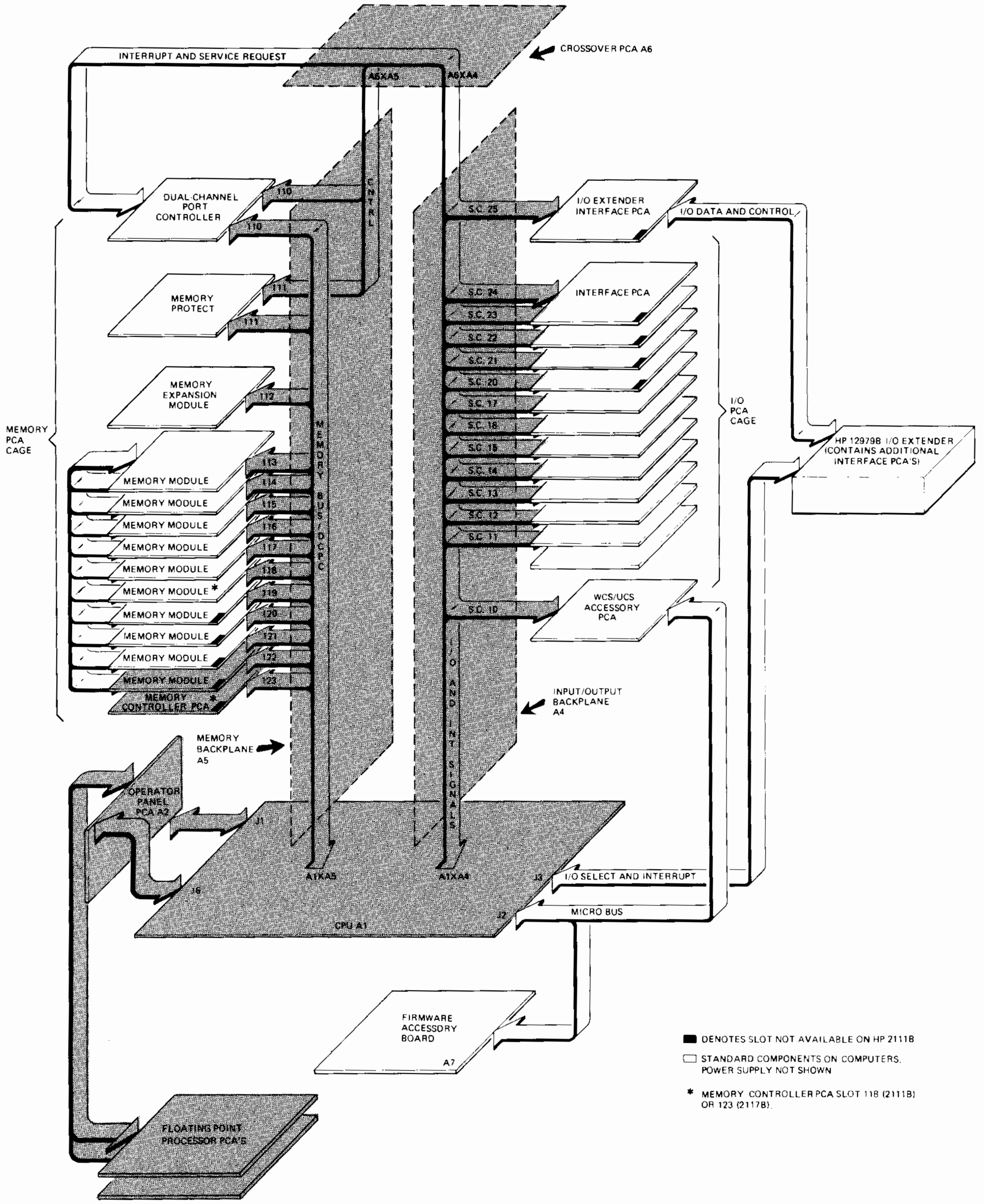
Figure 7-9. DCPC Control Word Formats

Table 7-2. DCPC Initialization Program

LABEL	OPCODE	OPERAND	COMMENTS
ASGN1	LDA	CW1	Fetches control word 1 (CW1) from memory and loads it in A-register.
	OTA	6B	Outputs CW1 to DCPC Channel 1.
MAR1	CLC	2B	Prepares Memory Address Register to receive control word 2 (CW2).
	LDA	CW2	Fetches CW2 from memory and loads it in A-register.
	OTA	2B	Outputs CW2 to DCPC Channel 1.
WCR1	STC	2B	Prepares Word Count Register to receive control word 3 (CW3).
	LDA	CW3	Fetches CW3 from memory and loads it in A-register.
	OTA	2B	Outputs CW3 to DCPC Channel 1.
STRT1	STC	10B,C	Start input device.
	STC	6B,C	Activate DCPC Channel 1.
	SFS JMP	6B *-1	Wait while data transfer takes place or, if interrupt processing is used, continue program.
⋮	⋮	⋮	
	HLT		Halt
CW1	OCT	120010	Assignment for DCPC Channel 1 (ASGN1); specifies I/O channel select code address (10_8), STC after each word is transferred, and CLC after final word is transferred.
CW2	OCT	100200	Memory Address Register control. DCPC Channel 1 (MAR1); specifies memory input operation and starting memory address (200_8).
CW3	DEC	-50	Word Count Register control. DCPC Channel 1 (WCR1); specifies the 2's complement of the number of character words in the block of data to be transferred (50_{10}).
DCPC OUTPUT DATA INITIALIZATION SEQUENCE			
WRTE1	CLC	12B	Idle device. (Equivalent to STRT1 above.)
	STF	12B	Assert SRQ.
	STC	6B,C	Activate DCPC channel 1.
	SFS JMP	6B *-1	Wait while data transfer occurs, or continue program if interrupt processing is used.

APPENDIX

COMPUTER PHYSICAL LAYOUT

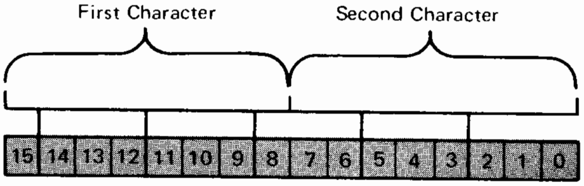


7700-202

CHARACTER CODES

ASCII Character	First Character Octal Equivalent	Second Character Octal Equivalent
A	040400	000101
B	041000	000102
C	041400	000103
D	042000	000104
E	042400	000105
F	043000	000106
G	043400	000107
H	044000	000110
I	044400	000111
J	045000	000112
K	045400	000113
L	046000	000114
M	046400	000115
N	047000	000116
O	047400	000117
P	050000	000120
Q	050400	000121
R	051000	000122
S	051400	000123
T	052000	000124
U	052400	000125
V	053000	000126
W	053400	000127
X	054000	000130
Y	054400	000131
Z	055000	000132
a	060400	000141
b	061000	000142
c	061400	000143
d	062000	000144
e	062400	000145
f	063000	000146
g	063400	000147
h	064000	000150
i	064400	000151
j	065000	000152
k	065400	000153
l	066000	000154
m	066400	000155
n	067000	000156
o	067400	000157
p	070000	000160
q	070400	000161
r	071000	000162
s	071400	000163
t	072000	000164
u	072400	000165
v	073000	000166
w	073400	000167
x	074000	000170
y	074400	000171
z	075000	000172
0	030000	000060
1	030400	000061
2	031000	000062
3	031400	000063
4	032000	000064
5	032400	000065
6	033000	000066
7	033400	000067
8	034000	000070
9	034400	000071
NUL	000000	000000
SOH	000400	000001
STX	001000	000002
ETX	001400	000003
EOT	002000	000004
ENQ	002400	000005

ASCII Character	First Character Octal Equivalent	Second Character Octal Equivalent
ACK	003000	000006
BEL	003400	000007
BS	004000	000010
HT	004400	000011
LF	005000	000012
VT	005400	000013
FF	006000	000014
CR	006400	000015
SO	007000	000016
SI	007400	000017
DLE	010000	000020
DC1	010400	000021
DC2	011000	000022
DC3	011400	000023
DC4	012000	000024
NAK	012400	000025
SYN	013000	000026
ETB	013400	000027
CAN	014000	000030
EM	014400	000031
SUB	015000	000032
ESC	015400	000033
FS	016000	000034
GS	016400	000035
RS	017000	000036
US	017400	000037
SPACE	020000	000040
!	020400	000041
"	021000	000042
#	021400	000043
$	022000	000044
%	022400	000045
&	023000	000046
'	023400	000047
(	024000	000050
)	024400	000051
*	025000	000052
+	025400	000053
,	026000	000054
-	026400	000055
.	027000	000056
/	027400	000057
:	035000	000072
;	035400	000073
<	036000	000074
=	036400	000075
>	037000	000076
?	037400	000077
@	040000	000100
[	055400	000133
\	056000	000134
]	056400	000135
∧	057000	000136
_	057400	000137
`	060000	000140
{	075400	000173
|	076000	000174
}	076400	000175
~	077000	000176
DEL	077400	000177



OCTAL ARITHMETIC

ADDITION

TABLE

0	01	02	03	04	05	06	07
1	02	03	04	05	06	07	10
2	03	04	05	06	07	10	11
3	04	05	06	07	10	11	12
4	05	06	07	10	11	12	13
5	06	07	10	11	12	13	14
6	07	10	11	12	13	14	15
7	10	11	12	13	14	15	16

EXAMPLE

Add: 3677 octal
+ 1331 octal
(111-) carries
5230 octal

MULTIPLICATION

TABLE

1	02	03	04	05	06	07
2	04	06	10	12	14	16
3	06	11	14	17	22	25
4	10	14	20	24	30	34
5	12	17	24	31	36	43
6	14	22	30	36	44	52
7	16	25	34	43	52	61

EXAMPLE

Multiply: 657 octal
X 54 octal
3274
4153
45024 octal

(Reminder: add in octal)

COMPLEMENT

To find the two's complement form of an octal number. (Same procedure whether converting from positive to negative or negative to positive.)

RULE

1. Subtract from the maximum representable octal value.

2. Add one.

EXAMPLE

Two's complement of 556_8:

177777
- 000556
177221
+ 1
177222_8

OCTAL/DECIMAL CONVERSIONS

OCTAL TO DECIMAL

TABLE

OCTAL	DECIMAL
0- 7	0- 7
10-17	8-15
20-27	16-23
30-37	24-31
40-47	32-39
50-57	40-47
60-67	48-55
70-77	56-63
100	64
200	128
400	256
1000	512
2000	1024
4000	2048
10000	4096
20000	8192
40000	16384
77777	32767

EXAMPLE

Convert 463_8 to a decimal integer.

$400_8 = 256_{10}$

$60_8 = 48_{10}$

$3_8 = \underline{3_{10}}$

307 decimal

DECIMAL TO OCTAL

TABLE

DECIMAL	OCTAL
1	1
10	12
20	24
40	50
100	144
200	310
500	764
1000	1750
2000	3720
5000	11610
10000	23420
20000	47040
32767	77777

EXAMPLE

Convert 5229_{10} to an octal integer.

$5000_{10} = 11610_8$

$200_{10} = 310_8$

$20_{10} = 24_8$

$9_{10} = \underline{11_8}$

12155_8

↑

(Reminder: add in octal)

NEGATIVE DECIMAL TO TWO'S COMPLEMENT OCTAL

TABLE

DECIMAL	2's COMP
-1	177777
-10	177766
-20	177754
-40	177730
-100	177634
-200	177470
-500	177014
-1000	176030
-2000	174060
-5000	166170
-10000	154360
-20000	130740
-32768	100000

EXAMPLE

Convert -629_{10} to two's complement octal.

$-500_{10} = 177014_8$

$-100_{10} = 177634_8$

$-20_{10} = 177754_8$ (Add in octal)

$-9_{10} = \underline{177767_8}$

176613_8

For reverse conversion (two's complement octal to negative decimal):

1. Complement, using procedure on facing page.
2. Convert to decimal, using OCTAL TO DECIMAL table.

MATHEMATICAL EQUIVALENTS

$2^{\pm n}$ IN DECIMAL

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.0625
32	5	0.03125
64	6	0.01562 5
128	7	0.00781 25
256	8	0.00390 625
512	9	0.00195 3125
1 024	10	0.00097 65625
2 048	11	0.00048 82812 5
4 096	12	0.00024 41406 25
8 192	13	0.00012 20703 125
16 384	14	0.00006 10351 5625
32 768	15	0.00003 05175 78125
65 536	16	0.00001 52587 89062 5
131 072	17	0.00000 76293 94531 25
262 144	18	0.00000 38146 97265 625
524 288	19	0.00000 19073 48632 8125
1 048 576	20	0.00000 09536 74316 40625
2 097 152	21	0.00000 04768 37158 20312 5
4 194 304	22	0.00000 02384 18579 10156 25
8 388 608	23	0.00000 01192 09289 55078 125
16 777 216	24	0.00000 00596 04644 77539 0625
33 554 432	25	0.00000 00298 02322 38769 53125
67 108 864	26	0.00000 00149 01161 19384 76562 5
134 217 728	27	0.00000 00074 50580 59692 38281 25
268 435 456	28	0.00000 00037 25290 29846 19140 625
536 870 912	29	0.00000 00018 62645 14923 09570 3125
1 073 741 824	30	0.00000 00009 31322 57461 54785 15625
2 147 483 648	31	0.00000 00004 65661 28730 77392 57812 5
4 294 967 296	32	0.00000 00002 32830 64365 38696 28906 25

$10^{\pm n}$ IN OCTAL

10^n	n	10^{-n}
1	0	1.000 000 000 000 000 000 00
12	1	0.063 146 314 631 463 146 31
144	2	0.005 075 341 217 270 243 66
1 750	3	0.000 406 111 564 570 651 77
23 420	4	0.000 032 155 613 530 704 15
303 240	5	0.000 002 476 132 610 706 64
3 641 100	6	0.000 000 206 157 364 055 37
46 113 200	7	0.000 000 015 327 745 152 75
575 360 400	8	0.000 000 001 257 143 561 06
7 346 545 000	9	0.000 000 000 104 560 276 41
112 402 762 000	10	0.000 000 000 006 676 337 66
1 351 035 564 000	11	0.000 000 000 000 537 657 77
16 432 451 210 000	12	0.000 000 000 000 043 136 32
221 411 634 520 000	13	0.000 000 000 000 003 411 35
2 657 142 036 440 000	14	0.000 000 000 000 000 264 11
34 327 724 461 500 000	15	0.000 000 000 000 000 022 01
434 157 115 760 200 000	16	0.000 000 000 000 000 001 63
5 432 127 413 542 400 000	17	0.000 000 000 000 000 000 14
67 405 553 164 731 000 000	18	0.000 000 000 000 000 000 01

MATHEMATICAL EQUIVALENTS

2^x IN DECIMAL

x	2^x	x	2^x	x	2^x
0.001	1.00069 33874 62581	0.01	1.00695 55500 56719	0.1	1.07177 34625 36293
0.002	1.00138 72557 11335	0.02	1.01395 94797 90029	0.2	1.14869 83549 97035
0.003	1.00208 16050 79633	0.03	1.02101 21257 07193	0.3	1.23114 44133 44916
0.004	1.00277 64359 01078	0.04	1.02811 38266 56067	0.4	1.31950 79107 72894
0.005	1.00347 17485 09503	0.05	1.03526 49238 41377	0.5	1.41421 35623 73095
0.006	1.00416 75432 38973	0.06	1.04246 57608 41121	0.6	1.51571 65665 10398
0.007	1.00486 38204 23785	0.07	1.04971 66836 23067	0.7	1.62450 47927 12471
0.008	1.00556 05803 98468	0.08	1.05701 80405 61380	0.8	1.74110 11265 92248
0.009	1.00625 78234 97782	0.09	1.06437 01824 53360	0.9	1.86606 59830 73615

$n \log_{10} 2$, $n \log_2 10$ IN DECIMAL

n	$n \log_{10} 2$	$n \log_2 10$	n	$n \log_{10} 2$	$n \log_2 10$
1	0.30102 99957	3.32192 80949	6	1.80617 99740	19.93156 85693
2	0.60205 99913	6.64385 61898	7	2.10720 99696	23.25349 66642
3	0.90308 99870	9.96578 42847	8	2.40823 99653	26.57542 47591
4	1.20411 99827	13.28771 23795	9	2.70926 99610	29.89735 28540
5	1.50514 99783	16.60964 04744	10	3.01029 99566	33.21928 09489

MATHEMATICAL CONSTANTS IN OCTAL SCALE

π =	$(3.11037\ 552421)_{(8)}$	e =	$(2.55760\ 521305)_{(8)}$	γ =	$(0.44742\ 147707)_{(8)}$
π^{-1} =	$(0.24276\ 301556)_{(8)}$	e^{-1} =	$(0.27426\ 530661)_{(8)}$	$\ln \gamma$ =	$-(0.43127\ 233602)_{(8)}$
$\sqrt{\pi}$ =	$(1.61337\ 611067)_{(8)}$	$\sqrt{e}$ =	$(1.51411\ 230704)_{(8)}$	$\log_2 \gamma$ =	$-(0.62573\ 030645)_{(8)}$
$\ln \pi$ =	$(1.11206\ 404435)_{(8)}$	$\log_{10} e$ =	$(0.33626\ 754251)_{(8)}$	$\sqrt{2}$ =	$(1.32404\ 746320)_{(8)}$
$\log_2 \pi$ =	$(1.51544\ 163223)_{(8)}$	$\log_2 e$ =	$(1.34252\ 166245)_{(8)}$	$\ln 2$ =	$(0.54271\ 027760)_{(8)}$
$\sqrt{10}$ =	$(3.12305\ 407267)_{(8)}$	$\log_2 10$ =	$(3.24464\ 741136)_{(8)}$	$\ln 10$ =	$(2.23273\ 067355)_{(8)}$

OCTAL COMBINING TABLES

MEMORY REFERENCE INSTRUCTIONS

Indirect Addressing

Refer to octal instruction codes given on the following page.
To combine code for indirect addressing, merge "100000" with octal instruction code.

REGISTER REFERENCE INSTRUCTIONS

Shift-Rotate Group (SRG)

1. select to operate A or B.
2. select 1 to 4 instructions, not more than one from each column.
3. combine octal codes (leading zeros omitted) by inclusive or.
4. order of execution is from column 1 to column 4.

A Operations

1	2	3	4
ALS (1000)	CLE (40)	SLA (10)	ALS (20)
ARS (1100)			ARS (21)
RAL (1200)			RAL (22)
RAR (1300)			RAR (23)
ALR (1400)			ALR (24)
ERA (1500)			ERA (25)
ELA (1600)			ELA (26)
ALF (1700)			ALF (27)

B Operations

1	2	3	4
BLS (5000)	CLE (4040)	SLB (4010)	BLS (4020)
BRS (5100)			BRS (4021)
RBL (5200)			RBL (4022)
RBR (5300)			RBR (4023)
BLR (5400)			BLR (4024)
ERB (5500)			ERB (4025)
ELB (5600)			ELB (4026)
BLF (5700)			BLF (4027)

Alter-Skip Group (ASG)

1. select to operate on A or B.
2. select 1 to 8 instructions, not more than one from each column.
3. combine octal codes (leading zeros omitted) by inclusive or.
4. order of execution is from column 1 to column 8.

A Operations

1	2	3	4
CLA (2400)	SEZ (2040)	CLE (2100)	SSA (2020)
CMA (3000)		CME (2200)	
CCA (3400)		CCE (2300)	

5	6	7	8
SLA (2010)	INA (2004)	SZA (2002)	RSS (2001)

B Operations

1	2	3	4
CLB (6400)	SEZ (6040)	CLE (6100)	SSB (6020)
CMB (7000)		CME (6200)	
CCB (7400)		CCE (6300)	

5	6	7	8
SLB (6010)	INB (6004)	SZB (6002)	RSS (6001)

INPUT/OUTPUT INSTRUCTIONS

Clear Flag

Refer to octal instruction codes given on the following page.
To clear flag after execution (instead of holding flag), merge "001000" with octal instruction code.

INSTRUCTION CODES IN OCTAL

Memory Reference

ADA	04(0XX)---
ADB	04(1XX)---
AND	01(0XX)---
CPA	05(0XX)---
CPB	05(1XX)---
IOR	03(0XX)---
ISZ	03(1XX)---
JMP	02(1XX)---
JSB	01(1XX)---
LDA	06(0XX)---
LDB	06(1XX)---
STA	07(0XX)---
STB	07(1XX)---
XOR	02(0XX)---
	↑ Binary

Shift-Rotate

ALF	001700
ALR	001400
ALS	001000
ARS	001100
BLF	005700
BLR	005400
BLS	005000
BRS	005100
CLE	000040
ELA	001600
ELB	005600
ERA	001500
ERB	005500
NOP	000000
RAL	001200
RAR	001300
RBL	005200
RBR	005300
SLA	000010
SLB	004010

Alter-Skip

CCA	003400
CCB	007400
CCE	002300
CLA	002400
CLB	006400
CLE	002100

CMA	003000
CMB	007000
CME	002200
INA	002004
INB	006004
RSS	002001
SEZ	002040
SLA	002010
SLB	006010
SSA	002020
SSB	006020
SZA	002002
SZB	006002

Input/Output

CLC	1067--
CLF	1031--
CLO	103101
HLT	1020--
LIA	1025--
LIB	1065--
MIA	1024--
MIB	1064--
OTA	1026--
OTB	1066--
SFC	1022--
SFS	1023--
SOC	102201
SOS	102301
STC	1027--
STF	1021--
STO	102101

Extended Arithmetic

ASL	1000(01X)-
ASR	1010(01X)-
• DIV	100400
• DLD	104200
• DST	104400
LSL	1000(10X)-
LSR	1010(10X)-
• MPY	100200
RRL	1001(00X)-
RRR	1011(00X)-
	↑ Binary

*Assuming: no indirect addressing
no combined instructions
shifts taken in first position only
hold flag after I/O execution

Refer to preceding page for octal combining tables

Ext. Inst. Group

ADX	105746
ADY	105756
CAX	101741
CAY	101751
CBS	105774
CBT	105766
CBX	105741
CBY	105751
CMW	105776
CXA	101744
CXB	105744
CYA	101754
CYB	105754
DSX	105761
DSY	105771
ISX	105760
ISY	105770
JLY	105762
JPY	105772
LAX	101742
LAY	101752
LBT	105763
LBX	105742
LBY	105752
LDX	105745
LDY	105755
MBT	105765
• MVW	105777
SAX	101740
SAY	101750
SBS	105773
SBT	105764
SBX	105740
SBY	105750
SFB	105767
STX	105743
STY	105753
TBS	105775
XAX	101747
XAY	101757
XBX	105747
XBY	105757

Floating Point

FAD	105000
FDV	105060
FIX	105100
FLT	105120
FMP	105040
FSB	105020
• .FIXD	105104
• .FLTD	105124
• .XADD	105001
• .XDIV	105061
• .XFTD	105125

• .XFTS	105121
• .XFXD	105105
• .XFXS	105101
• .XMPY	105041
• .XSUB	105021

Scientific Inst. Set

• ALOG	105322
• ALOGT	105327
• ATAN	105323
• COS	105324
• EXP	105326
• SIN	105325
• SQRT	105321
• TAN	105320
• TANH	105330

Fast FORTRAN

*• DBLE	105201
• DDINT	105217
*• SNGL	105202
• .DFER	105205
• .ENTP	105224
• .ENTR	105223
• .FLUN	105226
• .GOTO	105221
• .PACK	105230
• .PWR2	105225
• .SETP	105227
• .XCOM	105215
• .XFER	105220
• .XPAK	105206
• ..DCM	105216
• ..MAP	105222

Dynamic Mapping System

DJP	105732
DJS	105733
JRS	105715
LFA	101727
LFB	105727
MBF	105703
MBI	105702
MBW	105704
MWF	105706
MWI	105705
MWW	105707
PAA	101712
PAB	105712
PBA	101713
PBB	105713
RSA	101730
RSB	105730

RVA	101731
RVB	105731
SJP	105734
SJS	105735
SSM	105714
SYA	101710
SYB	105710
UJP	105736
UJS	105737
USA	101711
USB	105711
XCA	101726
XCB	105726
XLA	101724
XLB	105724
XMA	101722
XMB	105722
XMM	105720
XMS	105721
XSA	101725
XSB	105725

BASE SET INSTRUCTION CODES IN BINARY

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	AND	001		0	Z/C	← Memory Address →									
D/I	XOR	010		0	Z/C										
D/I	IOR	011		0	Z/C										
D/I	JSB	001		1	Z/C										
D/I	JMP	010		1	Z/C										
D/I	ISZ	011		1	Z/C										
D/I	AD*	100		A/B	Z/C										
D/I	CP*	101		A/B	Z/C										
D/I	LD*	110		A/B	Z/C										
D/I	ST*	111		A/B	Z/C										

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	SRG	000		A/B	0	D/E	*LS		000	†CLE	D/E	‡SL*	*LS		000
				A/B	0	D/E	*RS		001		D/E		*RS		001
				A/B	0	D/E	R*L		010		D/E		R*L		010
				A/B	0	D/E	R*R		011		D/E		R*R		011
				A/B	0	D/E	*LR		100		D/E		*LR		100
				A/B	0	D/E	ER*		101		D/E		ER*		101
				A/B	0	D/E	EL*		110		D/E		EL*		110
				A/B	0	D/E	*LF		111		D/E		*LF		111
				NOP	000				000		000				000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	ASG	000		A/B	1	CL*	01	CLE	01	SEZ	SS*	SL*	IN*	SZ*	RSS
				A/B		CM*	10	CME	10						
				A/B		CC*	11	CCE	11						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	IOG	000			1	H/C	HLT		000	← Select Code →					
					1	0	STF		001						
					1	1	CLF		001						
					1	0	SFC		010						
					1	0	SFS		011						
				A/B	1	H/C	MI*		100						
				A/B	1	H/C	LI*		101						
				A/B	1	H/C	OT*		110						
				0	1	H/C	STC		111						
				1	1	H/C	CLC		111						
					1	0	STO		001		000			001	
					1	1	CLO		001		000			001	
					1	H/C	SOC		010		000			001	
					1	H/C	SOS		011		000			001	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	EAG	000		MPY**		000		010			000			000	
				DIV**		000		100			000			000	
				DLD**		100		010			000			000	
				DST**		100		100			000			000	
				ASR		001		000		0	1				
				ASL		000		000		0	1				
				LSR		001		000		1	0			number	
				LSL		000		000		1	0	←		of	→
				RRR		001		001		0	0			bits	
				RRL		000		001		0	0				

Notes: * = A or B, according to bit 11.
D/I, A/B, Z/C, D/E, H/C coded: 0/1.
**Second word is Memory Address.

†CLE: Only this bit is required.
‡SL*: Only this bit and bit 11 (A/B as applicable) are required.

BASE SET INSTRUCTION CODES IN BINARY (CONT)

EXTENDED INSTRUCTION GROUP	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAX/SAY/SBX/SBY	1	0	0	0	A/B	0	1	1	1	1	1	0	X/Y	0	0	0
CAX/CAY/CBX/CBY	1	0	0	0	A/B	0	1	1	1	1	1	0	X/Y	0	0	1
LAX/LAY/LBX/LBY	1	0	0	0	A/B	0	1	1	1	1	1	0	X/Y	0	1	0
STX/STY	1	0	0	0	1	0	1	1	1	1	1	0	X/Y	0	1	1
CXA/CYA/CXB/CYB	1	0	0	0	A/B	0	1	1	1	1	1	0	X/Y	1	0	0
LDX/LDY	1	0	0	0	1	0	1	1	1	1	1	0	X/Y	1	0	1
ADX/ADY	1	0	0	0	1	0	1	1	1	1	1	0	X/Y	1	1	0
XAX/XAY/XBX/XBY	1	0	0	0	A/B	0	1	1	1	1	1	0	X/Y	1	1	1
ISX/ISY/DSX/DSY	1	0	0	0	1	0	1	1	1	1	1	1	X/Y	0	0	I/D
JUMP INSTRUCTIONS	1	0	0	0	1	0	1	1	1	1	1	1	▨	0	1	0
JLY =													0			
JPY =													1			
BYTE INSTRUCTIONS	1	0	0	0	1	0	1	1	1	1	1	1	0	▨	▨	▨
LBT =														0	1	1
SBT =														1	0	0
MBT =														1	0	1
CBT =														1	1	0
SFB =														1	1	1
BIT INSTRUCTIONS	1	0	0	0	1	0	1	1	1	1	1	1	1	▨	▨	▨
SBS =														0	1	1
CBS =														1	0	0
TBS =														1	0	1
WORD INSTRUCTIONS	1	0	0	0	1	0	1	1	1	1	1	1	1	1	1	▨
CMW =																0
MVW =																1

BASE SET INSTRUCTION CODES IN BINARY (CONT)

15	14 13 12	11 10 9	8 7	6	5 4	3	2 1 0
1	FLTPT 000	101	00	FAD	000	0	000
				FSB	001		
				FMP	010		
				FDV	011		
				FIX	100		
				FLT	101		
				.XADD	000		001
				.XSUB	001		
				.XMPY	010		
				.XDIV	011		
				.XFXS	100		
				.XFTS	101		
				.FIXD	100		100
				.FLTD	101		
				.XFXD	100		101
				.XFTD	101		

SCIENTIFIC AND FAST FORTRAN INSTRUCTION CODES IN BINARY

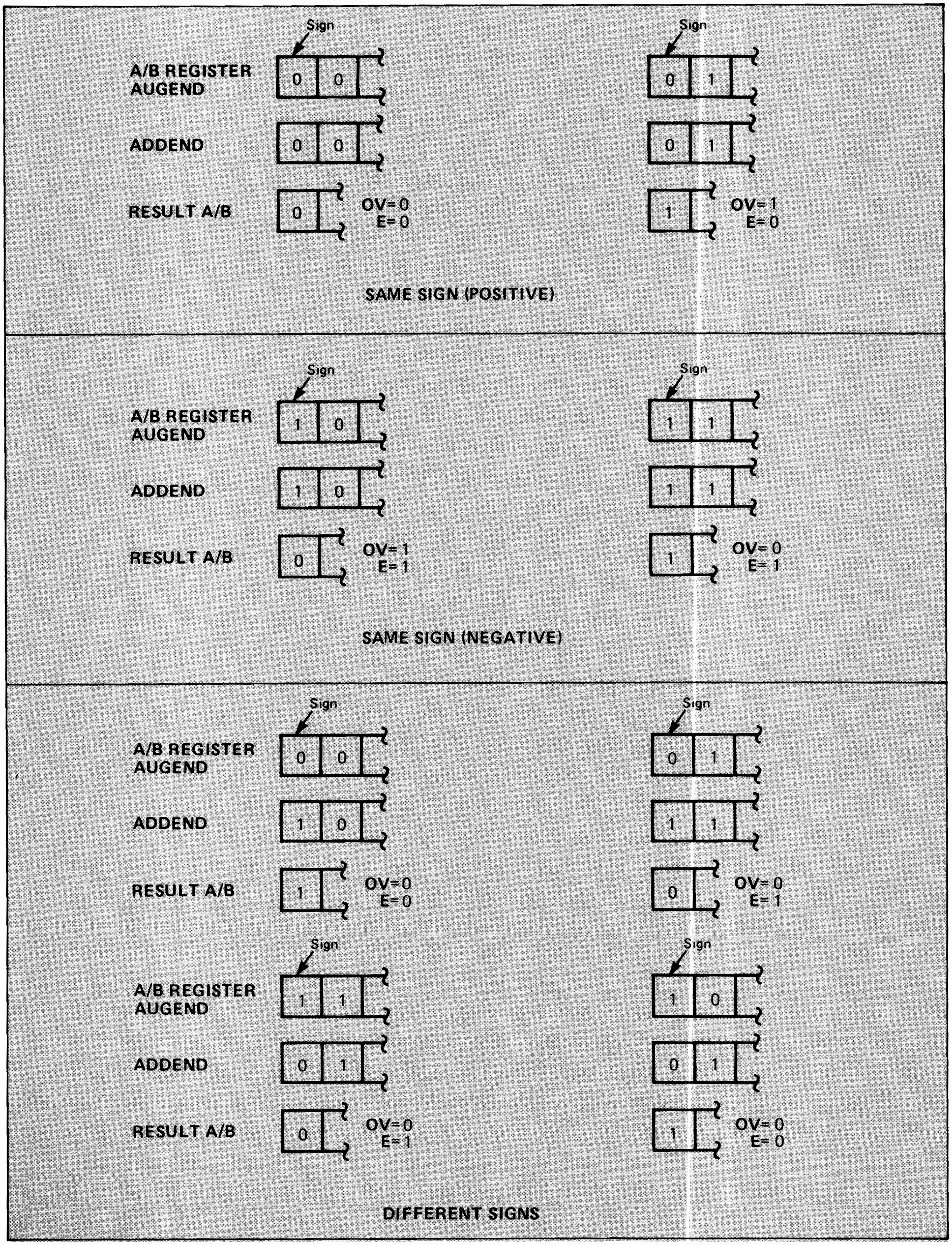
15	14 13 12	11 10 9	8 7 6	5	4	3		2 1 0
1	SCIENTIFIC INST. SET 000	101	011	0	1	0	TAN	000
							SQRT	001
							ALOG	010
							ATAN	011
							COS	100
							SIN	101
							EXP	110
							ALOGT	111
						1	TANH	000

15	14 13 12	11 10 9	8 7 6	5	4	3		2 1 0
1	FAST FORTRAN 000	101	010	0	0	0	DBLE	001
							SNGL	010
							.DFER	101
							.XPAK	110
					0	1	.XCOM	101
							..DCM	110
							DDINT	111
					1	0	.XFER	000
							.GOTO	001
							..MAP	010
							.ENTR	011
							.ENTP	100
							.PWR2	101
							.FLUN	110
							.SETP	111
					1	1	.PACK	000

DYNAMIC MAPPING SYSTEM INSTRUCTION CODES IN BINARY

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DJP/DJS/UJP/UJS	1	0	0	0	1	0	1	1	1	1	0	1	1	D/U	1	P/S
JRS	1	0	0	0	1	0	1	1	1	1	0	0	1	1	0	1
LFA/LFB	1	0	0	0	A/B	0	1	1	1	1	0	1	0	1	1	1
MBI/MBF	1	0	0	0	1	0	1	1	1	1	0	0	0	0	1	I/F
MBW	1	0	0	0	1	0	1	1	1	1	0	0	0	1	0	0
MWF	1	0	0	0	1	0	1	1	1	1	0	0	0	1	1	0
MWI/MWW	1	0	0	0	1	0	1	1	1	1	0	0	0	1	I/W	1
PAA/PAB	1	0	0	0	A/B	0	1	1	1	1	0	0	1	0	1	0
PBA/PBB	1	0	0	0	A/B	0	1	1	1	1	0	0	1	0	1	1
RSA/RSB/RVA/RVB	1	0	0	0	A/B	0	1	1	1	1	0	1	1	0	0	S/V
SJP/SJS	1	0	0	0	1	0	1	1	1	1	0	1	1	1	0	P/S
SSM	1	0	0	0	1	0	1	1	1	1	0	0	1	1	0	0
SYA/SYB	1	0	0	0	A/B	0	1	1	1	1	0	0	1	0	0	0
USA/USB	1	0	0	0	A/B	0	1	1	1	1	0	0	1	0	0	1
XCA/XCB/XLA/XLB	1	0	0	0	A/B	0	1	1	1	1	0	1	0	1	L/C	0
XMA/XMB	1	0	0	0	A/B	0	1	1	1	1	0	1	0	0	1	0
XMM/XMS	1	0	0	0	1	0	1	1	1	1	0	1	0	0	0	M/S
XSA/XSB	1	0	0	0	A/B	0	1	1	1	1	0	0	1	1	0	1

EXTEND AND OVERFLOW EXAMPLES



SAME SIGN (POSITIVE)

SAME SIGN (NEGATIVE)

DIFFERENT SIGNS

This page is intentionally blank.

INTERRUPT AND I/O CONTROL SUMMARY

INST	S.C. 00	S.C. 01	S.C. 02	S.C. 03
STC	NOP	NOP	Prepares DCPC channel 1 to receive and store the block length in 2's complement form.	Prepares DCPC channel 2 to receive and store the block length in 2's complement form.
CLC	Clears all Control FF's from S.C. 06 and up; effectively turns off all I/O devices.	NOP	Prepares DCPC channel 1 to receive and store the direction of data flow and the starting memory address.	Prepares DCPC channel 2 to receive and store the direction of data flow and the starting memory address.
STF	Turns on interrupt system.	STO sets overflow bit.	NOP	NOP
CLF	Turns off interrupt system except power fail (S.C. 04) and parity error (S.C. 05).	CLO clears overflow bit.	NOP	NOP
SFS	Skip if interrupt system is on.	SOS	NOP	NOP
SFC	Skip if interrupt system is off.	SOC	NOP	NOP
LIA/B	Loads A/B register with all zeros. (Equivalent to CLA/B instruction.)	Loads display register contents into A/B register.	Loads present contents of DCPC channel 1 word count register into A/B register.	Loads present contents of DCPC channel 2 word count register into A/B register.
MIA/B	Equivalent to a NOP.	Merges display register contents into A/B register.	Merges present contents of DCPC channel 1 word count register into A/B register.	Merges present contents of DCPC channel 2 word count register into A/B register.
OTA/B	NOP	Outputs A/B register contents into display register.	1. Outputs to DCPC channel 1 the block length in 2's complement form (previously prepared by an STC 02 instruction). 2. Outputs to DCPC channel 1 the direction of data flow and the starting memory address (previously prepared by a CLC 02 instruction).	1. Outputs to DCPC channel 2 the block length in 2's complement form (previously prepared by an STC 03 instruction). 2. Outputs to DCPC channel 2 the direction of data flow and the starting memory address (previously prepared by a CLC 03 instruction).

S.C. 04	S.C. 05	S.C. 06	S.C. 07	S.C. 10-77
Re-initializes power-fail logic and restores interrupt capability to lower priority functions.	Turns on memory protect.	Sets Control FF on DCPC channel 1 (activates DMA).	Sets Control FF on DCPC channel 2 (activates DMA).	Sets PCA Control FF and turns on device on channel specified by S.C.
Re-initialize power-fail logic and restores interrupt capability to lower priority functions.	NOP	Clears Control FF on DCPC channel 1 (reestablishes priority with STF; does not turn off DCPC).	Clears Control FF on DCPC channel 2 (reestablishes priority with STF; does not turn off DCPC).	Clears PCA Control FF and turns off device.
Flag FF sets automatically when power comes up. (No program control possible.)	Turns on parity error interrupt capability.	Aborts DCPC channel 1 data transfer.	Aborts DCPC channel 2 data transfer.	Sets PCA Flag FF.
Flag FF clears automatically when power fail occurs. (No program control possible.)	Turns off parity error interrupt capability and clears violation register bit 15.	Clears Flag FF on DCPC channel 1.	Clears Flag FF on DCPC channel 2.	Clears PCA Flag FF.
NOP	Skip if Dynamic Mapping System (DMS) interrupt.	Tests if DCPC channel 1 data transfer is complete.	Skip if DCPC channel 2 data transfer is complete.	Skip if I/O channel Flag FF is set.
Skip if power fail has occurred.	Skip if memory protect interrupt.	Tests if DCPC channel 1 data transfer is still in progress.	Skip if DCPC channel 2 data transfer is still in progress.	Skip if I/O channel Flag FF is clear.
Loads contents of central interrupt register (S.C. of last interrupting device) into least-significant bits of A/B register.	Loads contents of violation register into A/B register: Bit 15 = 1 = PE Bit 15 = 0 = MPV	Loads A/B register with all ones. (Equivalent to CCA/CCB instruction.)	Loads A/B register with all ones. (Equivalent to CCA/CCB instruction.)	Loads contents of PCA data buffer into A/B register.
Merges contents of central interrupt register into least-significant bits of A/B register.	Merges contents of violation register into A/B register.	Same as LIA/B 06 above.	Same as LIA/B 07 above.	Merges contents of PCA data buffer into A/B register.
NOP	Outputs first address of unprotected memory to fence register.	Outputs to DCPC channel 1 the S.C. of I/O channel. Specify STC after each word; CLC after block.	Outputs to DCPC channel 2 the S.C. of I/O channel. Specify STC after each word; CLC after block.	Outputs data from A/B register into PCA data buffer.

NOTES

NOTES

NOTES

HEWLETT hp PACKARD

SALES & SERVICE OFFICES

AFRICA, ASIA, AUSTRALIA

ANGOLA

Telectra
Empresa Técnica de
Equipamentos
Eléctricos, S.A.R.L.
R. Barbosa Rodrigues, 42-I DT.
Caixa Postal, 6487
Luanda
Tel: 35515/6
Cable: TELECTRA Luanda

AUSTRALIA

Hewlett-Packard Australia
Pty. Ltd.
31-41 Joseph Street
Blackburn, Victoria 3130
P.D. Box 36
Doncaster East, Victoria 3109
Tel: 89-6351
Telex: 31-024
Cable: HEWPARD Melbourne

Hewlett-Packard Australia
Pty. Ltd.
31 Bridge Street
Pymble
New South Wales, 2073
Tel: 449-6566
Telex: 21561
Cable: HEWPARD Sydney

Hewlett-Packard Australia
Pty. Ltd.
153 Greenhill Road
Parkside, S.A., 5063
Tel: 272-5911
Telex: 82536 AOEL
Cable: HEWPARD ADELAID

Hewlett-Packard Australia
Pty.Ltd.
141 Stirling Highway
Nedlands, W.A. 6009
Tel: 86-5455
Telex. 93859 PERTH
Cable: HEWPARD PERTH

Hewlett-Packard Australia
Pty. Ltd.
121 Wollongong Street
Fyshwick, A.C.T. 2609
Tel: 95-2733
Telex: 62650 Canberra
Cable: HEWPARD CANBERRA

Hewlett Packard Australia
Pty. Ltd.
5th Floor
Teachers Union Building
495-499 Boundary Street
Spring Hill, 4000 Queensland
Tel: 229-1544
Cable: HEWPARD Brisbane

GUAM

Medical/Pocket Calculators Dnly
Guam Medical Supply, Inc.
Jay Ease Building, Room 210
P.O. Box 8947
Tamuning 96911
Tel: 646-4513
Cable: EARMED Guam

HONG KONG

Schmidt & Co.(Hong Kong) Ltd.
P.O. Box 297
Connalight Centre
39th Floor
Connaught Road, Central
Hong Kong
Tel: H-255291-5
Telex: 74766 SCHMC HX
Cable: SCHMIDTCO Hong Kong

INDIA

Blue Star Ltd.
Kasturi Buildings
Jamshedji Tata Rd
Bombay 400 020
Tel: 29 50 21
Telex: 001-2156
Cable BLUEFROST

Blue Star Ltd.
Sahas
414/2 Vir Savarkar Marg
Prabhadevi
Bombay 400 025
Tel: 45 78 87
Telex: 011-4093
Cable: FROSTBLUE

Blue Star Ltd.
Band Box House
Prabhadevi
Bombay 400 025
Tel: 45 73 01
Telex: 011-3751
Cable: BLUESTAR

Blue Star Ltd.
7 Hare Street
P.O. Box 506
Calcutta 700 001
Tel: 23-0131
Telex: 021-7655
Cable: BLUESTAR

Blue Star Ltd.
7th & 8th Floor
Bhandari House
91 Nehru Place
New Delhi 110024
Tel: 634770 & 635166
Telex: 031-2463
Cable: BLUESTAR

Blue Star Ltd.
Blue Star House
11/11A Magarath Road
Bangalore 560 025
Tel: 55668
Telex: 043-430
Cable: BLUESTAR

Blue Star Ltd.
Meeakshi Mandiran
xxx/1678 Mahatma Gandhi Rd.
Cochin 682 016
Tel: 32069,32161,32282
Telex: 0885-514
Cable: BLUESTAR

Blue Star Ltd.
1-1-117/1
Sarojini Devi Road
Secunderabad 500 003
Tel: 70126, 70127
Cable: BLUEFROST
Telex: 015-459

Blue Star Ltd.
2/34 Kodambakkam High Road
Madras 600034
Tel: 82056
Telex: 041-379
Cable: BLUESTAR

INDONESIA

BERCA Indonesia P.T.
P.O. Box 496/Jkt.
JLN•Abdul Muis 62
Jakarta
Tel: 40369, 49886,49255,356038
JKT.42895
Cable: BERCACON

BERCA Indonesia P.t.
63 JL. Raya Gubeng
Surabaya
Tel: 44309

ISRAEL

Electronics & Engineering Div.
of Motorola Israel Ltd.
17, Kremenetski Street
P.O. Box 25016
Tel-Aviv
Tel: 38973
Telex: 33569
Cable: BASTEL Tel-Aviv

JAPAN

Yokogawa-Hewlett-Packard Ltd.
Ohashi Building
59-1 Yoyogi 1-Chome
Shibuya-ku, **Tokyo** 151
Tel: 03-370-2281/92
Telex: 232-2024YHP MARKET
TDK 23-724
Cable: YHPMARKET

Yokogawa-Hewlett-Packard Ltd.
Chuo Bldg., 4th Floor
4-20, Nishinakajima 5-chome
Yodogawa-ku, Osaka-shi
Osaka,532
Tel: 06-304-6021

Yokogawa-Hewlett-Packard Ltd.
Nakamo Building
24 Kami Sasajima-cho
Nakamura-ku, **Nagoya** , 450
Tel: (052) 571-5171

Yokogawa-Hewlett-Packard Ltd.
Tanigawa Building
2-24-1 Tsuruya-cho
Kanagawa-ku
Yokohama, 221
Tel: 045-312-1252
Telex: 382-3204 YHP YOK

Yokogawa-Hewlett-Packard Ltd.
Mito Mitsu Building
105, Chome-1,San-no-maru
Mito, Ibaragi 310
Tel:: 0292-25-7470

Yokogawa-Hewlett-Packard Ltd.
Inoue Building
1348-3, Asahi-cho, 1-chome
Ataugi, Kanagawa 243
Tel: 0462-24-0452

Yokogawa-Hewlett-Packard Ltd.
Kumagaya Asahi
Hackijuni Building
4th Floor
3-4, Tsukuba
Kumagaya, Saitama 360
Tel: 0485-24-6563

KENYA

Technical Engineering
Services(E.A.)Ltd.,
P.O. Box 18311
Nairobi
Tel: 557726/556762
Cable: PROTON

Medical Only
International Aeradio(E.A.)Ltd.,
P.O. Box 19012
Nairobi Airport
Nairobi
Tel: 336055/56
Telex: 22201/22301
Cable: INTAERIO Nairobi

KOREA

Samsung Electronics Co., Ltd.
20th Fl. Dongbang Bldg. 250, 2-KA
C.P.O. Box 2775
Taepyung-Ro, Chung-Ku
Seoul
Tel: (23) 6811
Telex: 22575
Cable: ELEKSTAR Seoul

MALAYSIA

Teknik Mutu Sdn. Bhd.
2 Lorong 13/6A
Section 13
Petaling Jaya,**Selangor**
Tel: 54994/54916
Telex: MA 37605

Protel Engineering
P.O. Box 1917
Lot 259, Satok Road
Kuching, **Sarawak**
Tel: 2400
Cable: PROTEL ENG

MOZAMBIQUE

A.N. Goncalves, Lta.
162, 1 Apt. 14 Av. D. Luis
Caixa Postal 107
Lourenco Marques
Tel: 27091, 27114
Telex: 6-203 NEGON Mo
Cable: NEGON

NEW ZEALAND

Hewlett-Packard (N.Z.) Ltd.
P.O. Box 9443
Courtenay Place
Wellington
Tel: 877-199
Cable: HEWPACK Wellington

Hewlett-Packard (N.Z.) Ltd.
Pakuranga Professional Centre
267 Pakuranga Highway
Box 51092
Pakuranga
Tel: 569-651
Cable: HEWPACK,Auckland

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
Scientific Division
79 Carlton Gore Rd., Newmarket
P.O. Box 1234
Auckland
Tel: 75-289
Cable:DENTAL Auckland

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
P.O. Box 1994
147-161 Tory St.
Wellington
Tel: 850-799
Telex: 3858
Cable: DENTAL, Wellington

Analytical/Medical Only
Medical Supplies N.Z. Ltd
P.O. Box 309
239 Stanmore Road
Christchurch
Tel: 892-019
Cable: DENTAL, Christchurch

Analytical/Medical Only
Medical Supplies N.Z. Ltd
303 Great King Street
P.D. Box 233
Dunedin
Tel: 88-817
Cable: OENTAL, Dunedin

NIGERIA

The Electronics
Instrumentations Ltd.
N6B/770 Oyo Road
Oluseun House
P.M.B. 5402
Ibadan
Tel: 61577
Telex: 31231 TEIL Nigeria
Cable. THETEIL Ibadan

The Electronics Instrumentations Ltd.
144 Agege Motor Road, Mushin
P.O. Box 6645
Lagos
Cable: THETEIL Lagos

PAKISTAN

Mushko & Company, Ltd.
Dosman Chambers
Abdullah Haroon Road
Karachi-3
Tel: 511027, 512927
Telex: 2894
Cable: COOPERATDR Karachi

Mushko & Company, Ltd.
38B, Satellite Town
Rawalpindi
Tel: 41924
Cable: FEMUS Rawalpindi

PHILIPPINES

The Online Advanced
Systems Corporation
Rico House
Amorsolo cor. Herrera Str.
Legaspi Village, Makati
Metro **Manila**
Tel: 85-35-81, 85-34-91
Telex: 3274 QNLINE

RHODESIA

Field Technical Sales
45 Kelvin Road North
P.O. Box 3458
Salisbury
Tel: 705231 (5 lines)
Telex: RH 4122

SINGAPORE

Hewlett-Packard Singapore
(Pte.) Ltd.
1150 Depot Road
Alexandra P.O. Box 58
Singapore 4
Tel: 270-2355
Telex: HPSG RS 21486
Cable: HEWPACK, Singapore

SOUTH AFRICA

Hewlett-Packard South Africa
(Pty.). Ltd.
Private Bag Wendywood
Sandton, Transvaal 2144
Hewlett-Packard Centre
Daphne Street, Wendywood,
Sandton, Transvaal 2144
Tel: 802-10408
Telex: 8-4782
Cable: HEWPACK JOHANNESBURG

Service Department
Hewlett-Packard South Africa
(Pty.). Ltd.
P.O. Box 39325
Gramley, Sandton, 2018
451 Wynberg Extension 3,
Sandton, 2001
Tel: 636-8188/9
Telex: 8-2391

Hewlett-Packard South Africa
(Pty.), Ltd.
P.O. Box 120
Howard Place, Cape Province, 7450
Pine Park Centre, Forest Orive,
Pinelands, Cape Province, 7405
Tel: 53-7955 thu 9
Telex: 57-0006

Service Department
Hewlett-Packard South Africa
(Pty.). Ltd.
P.O. Box 37099
Dverport, Durban 4067
Braby House
641 Ridge Road
Durban, 4001
Tel: 88-7478
Telex: 6-7954

TAIWAN

Hewlett-Packard Far East Ltd.,
Taiwan Branch
39 Chung Hsiao West Road
Sec. 1, 7th Floor
Taipei
Tel: 3819160-4
Cable: HEWPACK TAIPEI

Hewlett-Packard Far East Ltd.
Taiwan Branch
68-2, Chung Cheng 3rd. Road
Kaohsiung
Tel: (07) 242318-Kaohsiung

Analytical Dnly
San Kwang Instruments Co., Ltd.,
No. 20, Yung Sui Road
Taipei
Tel: 3715171-4 (5 lines)
Telex: 22894 SANKWANG
Cable: SANKWANG TAIPEI

TANZANIA

Medical Dnly
International Aeraoio (E.A.). Ltd.
P.O. Box 861
Dar es Salaam
Tel: 21251 Ext. 265
Telex: 41030

THAILAND

UNIMESA Co., Ltd.
Elcom Research Building
2538 Sukumvit Ave.
Bangkok
Tel: 3932387, 3930338
Cable: UNIMESA Bangkok

UGANDA

Medical Only
International Aeradio(E.A.), Ltd.,
P.O. Box 2577
Kampala
Tel: 54388
Cable: INTAERIO Kampala

ZAMBIA

R.J. Tilbury (Zambia) Ltd.
P.O. Box 2792
Lusaka
Tel: 73793
Cable: ARJAYTEE, Lusaka

OTHER AREAS NOT LISTED, CONTACT:
Hewlett-Packard Intercontinental
3200 Hillview Ave.
Palo Alto, California 94304
Tel: (415) 493-1501
TWX: 910-373-1267
Cable: HEWPACK Palo Alto

CANADA

ALBERTA

Hewlett-Packard (Canada) Ltd.
11620A - 168th Street
EdmontonT5M 3T9
Tel: (403) 452-3670
TWX: 610-831-2431

Hewlett-Packard (Canada) Ltd.
210,7220 Fisher St. S.E.
Calgary T2H 2H8
Tel: (403) 253-2713
Twx: 610-821-6141

BRITISH COLUMBIA

Hewlett-Packard (Canada) Ltd.
837 E. Cordova Street
Vancouver V6A 3R2
Tel: (604) 254-0531
TWX: 610-922-5059

MANITOBA

Hewlett-Packard (Canada) Ltd.
513 Century St.
St. James
Winnipeg R3H OL8
Tel: (204) 786-7581
TWX: 610-671-3531

NOVA SCOTIA

Hewlett-Packard (Canada) Ltd.
800 Windmill Road
Dartmouth 83B 1L1
Tel: (902) 469-7820
TWX: 610-271-4482 HFX

ONTARIO

Hewlett-Packard (Canada) Ltd.
1020 Morrison Dr.
Ottawe K2H 8K7
Tel: (613) 820-6483
TWX: 610-563-1636

Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
Mississauga L4V 1M8
Tel: (416) 678-9430
TWX: 610-492-4246

QUEBEC

Hewlett-Packard (Canada) Ltd.
275 Hymus Blvd.
Pointe Claire H9R 1G7
Tel: (514) 697-4232
TWX: 610-422-3022
TLX: 05-821521 HPCL

FOR CANADIAN AREAS NOT LISTED:
Contact Hewlett-Packard (Canada)
Ltd. in Mississauga.

CENTRAL AND SOUTH AMERICA

ARGENTINA

Hewlett-Packard Argentina
S.A.
Av. Leandro N. Alem 822 - 12'
1001**Buenos Aires**
Tel: 31-6063,4.5,6 and 7
Telex: 122443 AR CIGY
Cable: HEWPACK ARG

BOLIVIA

Casa Kavlin S.A.
Calle Potosi' 1130
P.O. Box 500
La Paz
Tel: 41530,53221
Telex: CWC BX 5298,ITT 3560082
Cable: KAVLIN

BRAZIL

Hewlett-Packard do Brasil
I.e.C. Ltda.
Avenida Rio Negro, 980
Alphaville
06400**Barueri** SP
Tel: 429-3222

Hewlett-Packard do Brasil
I.e.C. Ltda.
Rua Padre Chagas, 32
90000-**Pôrto Alegre**-RS
Tel: (0512) 22-2998, 22-5621
Cable: HEWPACK Potto Alegre

Hewlett-Packard do Brasil
I.E.C. Ltda.
Rua Siqueira Campos, 53
Copacabana
20000-**Rio de Janeiro**
Tel: 257-80-94-DDD (021)
Telex: 391-212-1905 HEWP-BR
Cable: HEWPACK
Rio de Janeiro

CHILE

Calcagni y Metcalfe Ltda.
Alameda 580-01. 807
Casilla 2118
Santiago, 1
Tel: 398613
Telex: 3520001 CALMET
Cable: CALMET Santiago

COLOMBIA

Instrumentacidn
Henrik A. Langebaek & Kier S.A.
Carrera 7 No. 48-75
Apartado Aéreo 6287
Bogotá, I D.E.
Tel: 69-88-77
Cable: AARIS Bogotá
Telex: 044-400

COSTA RICA

Cientifica Costarricense S.A.
Avenida 2, Calle 5
San Pedro de Montes de Oca
Apartado 10159
San Jose
Tel: 24-38-20, 24-08-19
Telex: 2367 GALGUR CR
Cable: GALGUR

ECUADOR

Calculators Only
Computadoras y Equipos
Electrdnicos
P.O. Box 6423 CCI
Eloy Alfaro #1824,3 Piso
Quito
Tel: 453482
Telex: 02-2113 Sagita Ed
Cable: Sagita-Ouito

EL SALVADOR

Instrumentacion y Procesamiento
Electronico de el Salvador
Bulevar de los Heroes 11-48
San Salvador
Tel: 252787

GUATEMALA

IPESA
Avenida La Reforma 3-48,
Zona 9
Guatemala City
Tel: 63627, 64786
Telex: 4192 Teletro Gu

MEXICO

Hewlett-Packard Mexicana,
S.A. de C.V.
Av. Periférico Sur No. 6501
Tepepan, Xochimilco
Mexico 23, D.F.
Tel: 905-676-4600

Hewlett-Packard Mexicana,
S.A. de C.V.
Ave. Constitucidn No. 2184
Monterrey, N.L.
Tel: 48-71-32, 48-71-84
Telex: 038-410

NICARAGUA

Roberto Terán G.
Apartado Postal 689
Edificio Terán
Managua
Tel: 25114, 23412,23454
Cable: RDTERAN Managua

PANAMA

Electrdnico Balboa, S.A.
P.O. Box 4929
Calle Samuel Lewis
Cuidad de Panama
Tel: 64-2700
Telex: 3483103 Curunda,
Canal Zone
Cable: ELECTRON Panama

PERU

Compañla Electro Médica S.A.
Los Flamencos 145
San Isidro Casilla 1030
Lima 1
Tel: 41-4325
Cable: ELMED Lima

PUERTO RICO

Hewlett-Packard Inter-Americas
Puerto Rico Branch Office
Calle 272,
No. 203 Urb. Country Club
Carolina 00924
Tel: (809) 762-7255
Telex: 345 0514

URUGUAY

Pablo Ferrando S.A.
Comercial e Industrial
Avenida Italia 2877
Casilla de Correo 370
Montevideo
Tel: 40-3102
Cable: RADIUM Montevideo

VENEZUELA

Hewlett-Packard de Venezuela
C.A.
P.O. Box 50933
Caracas 105
Los Ruices Norte
3a Transversal
Edificio Segre
Caracas 107
Tel: 35-00-11 (20 lines)
Telex: 25146 HEWPACK
Cable: HEWPACK Caracas

FOR AREAS NOT LISTED, CONTACT:
Hewlett-Packard
Inter-Americas
3200 Hillview Ave.
Palo Alto, California 94304
Tel: (415) 493-1501
TWX: 910-373-1260
Cable: HEWPACK Palo Alto
Telex: 034-8300, 034-8493

EUROPE, NORTH AFRICA AND MIDDLE EAST

AUSTRIA
Hewlett-Packard Ges.m.b.H.
Handelskai 52
P.O. box 7
A-1205 **Vienna**
Tel: (0222) 351621 to 27
cable: HEWPAK Vienna
Telex: 75923 hewpak a

BELGIUM
Hewlett-Packard Beneiux
S.A./N.V
Avenue de Col-Vert, 1.
(Groenkraaglaan)
B-1170 **Brussels**
Tel: (02) 672 22 40
Cable: PALOBEN Brussels
Telex: 23 494 paloben bru

CYPRUS
Kypronics
19. Gregorios & Xenopoulos Rd.
P.O. 8ox 1152
CY-**Nicosia**
Tel: 45628/29
Cable: KYPRONICS PANDEHIS
Telex: 3018

CZECHOSLOVAKIA
Vyvojova a Provozni Zakladna
Vyzkumnych Ustavu v Bechovicich
CSSR-25097 **Bechovice u Prahy**
Tel: 89 93 41
Telex: 121333

Institute of Medical Bionics
Vyskumny Ustav Lekarskej Bioniky
Jedlova 6
CS-88346 **Brstislava-Kramare**
Tel: 44-551/45-541

DDR
Entwicklungslabor der TU Oresden
Forschungsinstitut Meinsberg
OOR-7305
Waldheim/Meinsberg
Tel: 37 667
Telex: 112145

Export Contact AG Zuerich
Guenther Forgber
Schlegelstrasse 15
1040 **Berlin**
Tel: 42-74-12
Telex. 111889

DENMARK
Hewlett-packard A/S
Datavej 52
DK-3460 **Birkerød**
Tel: (02) 81 66 40
Cable: HEWPACK AS
Telex: 37409 hpas dk

Hewlett-Packard A/S
Navervej 1
DK-8600 **Silkeborg**
Tel: (06) 82 71 66
Telex: 37409 hpas dk
Cable: HEWPACK AS

FINLAND
Hewlett-Packard OY
Nahkahousuntie 5
P.O. Box 6
SF-00211 **Helsinki** 21
Tel: (90) 6923031
Cable: HEWPACKOY Helsinki
Telex: 12-1563 HEWPA SF

FRANCE
Hewlett-Packard France
Quartier de Courtaboeuf
Boite Postale No. 6
F-91401 **Orsay** Cédex
Tel: (1) 907 78 25
Cable: HEWPACK Orsay
Telex: 600048

Hewlett-Packard France
Agency Régionale
"Le Saquin"
Chemin des Mouilles
B.P. 162
F-69130 **Ecully**
Tel: (78) 33 81 25,
Cable: HEWPACK Eculy
Telex: 31 06 17

Hewlett-Packard France
Agence Régionale
Péricentre de la Cépière
Chemin de la Cépière, 20
F-31300 **Toulouse-Le Mirail**
Tel:(61) 40 11 12
Cable: HEWPACK 51957
Telex: 510957

Hewlett-Packard France
Agence Régionale
Aéroport principal de
Marseille-Marignane
F-13721**Marignane**
Tel: (91) 89 12 36
Cable: HEWPACK MARGN
Telex: 410770

Hewlett-Packard France
Agence Régionale
63, Avenue de Rochester
B.P. 1124
F-35014 **Rennes** Cédex
Tel: (99) 36 33 21
Cable: HEWPACK 74912
Telex: 740912

Hewlett-Packard France
Agence Régionale
74, Allée de la Robertsau
F-67000 **Strasbourg**
Tel: (88) 35 23 20/21
Telex: 890141
Cable: HEWPACK STRBG

Hewlett-Packard France
Agence Régionale
Centre Vauban
201, rue Colbert
Entrée A2
F-59000 **Lille**
Tel: (20) 51 44 14
Telex: 820744

Hewlett-Packard France
Centre d' Affaires Paris-Nord
Bâtiment Ampère
Rue de La Commune de Paris
B.P. 300
F-93153 **Le Blanc Mesnil Cédex**
Tel: (01) 931 88 50

GERMAN FEDERAL REPUBLIC
Hewlett-Packard GmbH
Vertriebszentrale Frankfurt
Bernerstrasse 117
Postfach 560 140
0-6000 **Frankfurt** 56
Tel: (0611) 50 04-1
Cable: HEWPACKSA Frankfurt
Tel: (0611) 50 04-1
Cable: HEWPACKSA Frankfurt
Telex: 04 13249 hpffmd

Hewlett-Packard GmbH
Technisches Buero Böblingen
Herrenbergerstrasse 110
D-7030 **Böblingen**, Württemberg
Tel: (07031) 667-1
Cable: HEPAK Böblingen
Telex: 07265739 bbn

Hewlett-Packard GmbH
Technisches Buero Düsseldort
Emanuel-Leutze-Str.1 (Seestern)
0-4000 **Düsseldorf** 11
Tel: (0211) 59711
Telex: 085/86 533 hpdd d

Hewlett-Packard GmbH
Technisches Buero Hamburg
Wendenstrasse 23
0-2000 **Hamburg** 1
Tel: (040) 24 13 93
Cable: HEWPACKSA Hamburg
Telex: 21 63 032 hphh d

Hewlett-Packard GmbH
Technisches Buero Hannover
Am Grossmarkt 6
D-3000 **Hannover 91**
Tel: (0511) 46 60 01
Telex: 092 3259

Hewlett-Packard GmbH
Werk Groetzingen
Ohmstrasse 6
D-7500 **Karlsruke 41**
Tel: (0721) 69 40 06
Telex. 07-825707

Hewlett-Packard GmbH
Technisches Buero Nuremberg
Neumeyer Str. 90
D-8500 **Nuremberg**
Tel: (0911) 56 30 83/85
Telex: 0623 860

Hewlett-Packard GmbH
Technisches Buero München
Unterhachinger Strasse 28
ISAR Center
D-8012 **Ottobrunn**
Tel: (089) 601 30 61/7
Cable: HEWPACKSA München
Telex: 0524985

Hewlett-Packard GmbH
Technisches Buero Berlin
Keith Strasse 2-4
0-1000 **Berlin** 30
Tel: (030) 24 90 86
Telex: 18 3405 hpbln d

GREECE
Kostas Karayannis
08. Omirou Street
GR-**Athens** 133
Tel: 3237731
Cable: RAKAR Athens
Telex: 21 59 62 rkar gr

Analytical Only
"INTECO" G. Papathanassiou & Co.
Marni 17
GR - **Athens** 103
Tel: 522 1915
Cable: INTEKNIKA Athens
Telex: 21 5329 INTE GR

Medical Only
Technomed Hellas Ltd.
52,Skooula Street
GR - **Athens** 135
Tel: 362 6972, 363 3830
Cable.ETALAK athens
Telex: 21-4693 ETAL GR

HUNGARY
MTA
Müszerügyi és Méréstechnikai
Szolgalata
Lenin Krt. 67
1391 Budapest VI
Tel: 42 03 38
Telex: 22 51 14

ICELAND
Medical Only
Elding Trading Company Inc.
Hafnarhvoli - Tryggvatotu
IS-**Reykjavik**
Tel: 1 58 20
Cable. ELDING Reykjavik

IRAN
Hewlett-Packard Iran Ltd.
No. 13, Fourteenth St
Miremad Avenue
P.O. Box 41/2419
IR-**Tehran**
Tel. 851082-7
Telex: 21 25 74 khrm ir

IRAQ
Hewlett-Packard Trading Co.
4/1/8 Mansoor City
Baghdad
Tel. 5517827
Telex: 2455 hepairaq ik
Cable: HEWPACDAO,
Baghdad Iraq

IRELAND
Hewlett-Packard Ltd.
King Street Lane
GB-**Winnersh,**Wokingham
Berks, RG11 5AR
Tel: (0734) 78 47 74
Telex: 847178/848179

ITALY
Hewlett-Packard Italiana S.p.A.
Via Amerigo Vespucci 2
Casella postale 3645
I-20100 **Milano**
Tel: (2) 6251 (10 lines)
Cable: HEWPACKIT Milano
Telex: 32046

Hewlett-Packard Italiana S.p.A.
Via Pietro Maroncelli 40
(ang. Via Visentin)
I-35100 **Padova**
Tel: (49) 66 48 88
Telex. 41612 Hewpacki

Medical only
Hewlett-Packard Italiana S.p.A.
Via d'Aghiardi, 7
I-56100 **Pisa**
Tel: (050) 2 32 04
Telex: 32046 via Milano

Hewlett-Packard Italiana S.p.A
Via G. Armellini 10
I-00143 **Roma**
Tel: (06) 54 69 6l
Telex: 61514
Cable: HEWPACKIT Roma

Hewlett-Packard Italiana S.p.A.
Corso Giovanni Lanza
I-1031 **Torino**
Tel: (011) 682245/659308

Medical/Calculators Only
Hewlett-Packard Italiana S.p.A.
Via Principe Nicola 43 G/C
I-95126 **Catania**
Tel:(095) 37 05 04

Hewlett-Packard Italiana S.p.A.
Via Amerigo Vespucci, 9
I-80142 **Napoli**
Tel: (081) 33 77 11

Hewlett-Packard Italiana S.p.A.
Via E. Masi, 9/B
I-40137 **Bologna**
Tel: (051) 30 78 87

KUWAIT
Al-Khaldiya Trading &
Contracting Co.
P.O. Box 830-Satt
Kuwait
Tel: 424910-411726
Telex: 2481 areeg kt
Cable: VISCOUNT

LUXEMBURG
Hewlett-Packard Benelux
S.A./N.V.
Avenue du Col-Vert, 1.
(Groenkraaglaan)
B-1170 **Brussels**
Tel: (02) 672 22 40
Cable: PALOBEN Brussels
Telex: 23 494

MOROCCO
Gerep
190, Blvd. Brahim Roudani
Casablanca
Tel: 25-16-76/25-90-99
Cable: Gerep-Casa
Telex. 23739

NETHERLANDS
Hewlett-Packard Benelux N.V.
Van Heuven Goedhartlaan 121
P.O. Box 667
NL-1134 **Amstelveen**
Tel: (020) 47 20 21
Cable: PALOBEN Amsterdam
Telex: 13 216 hepa nl

NORWAY
Hewlett-Packard Norge A/S
Nesveien 13
Box 149
N-1344 **Haslum**
Tel: (02) 53 83 60
Telex: 16621 hpnas n

POLAND
Biuro Informacji Technicznej
Hewlett-Packard
U1 Stawki 2, 6P
00-950**Warszawa**
Tel: 395962/395187
Telex: 81 24 53 hepa pl

UNIPAN
Zaklad Doswiadczalny
Budowy Aparatury Naukowej
U1. Krajowej Rady Narodowej 51/55
00-800 **Warszawa**
Tel: 36190
Telex: 81 46 48

Zaklady Naprawcze Sprzetu
Medycznego
Plac Komuny Paryskiej 6
90-007 **Lódź**
Tel: 334-41, 337-83

PORTUGAL
Telectra-Empresa Técnica de
Equipamentos Eléctricos S.a.r.l.
Rua Rodrigo da Fonseca 103
P.O. Box 2531
P-Lisbon 1
Tel: (19) 68 60 72
Cable: TELECTRA Lisbon
Telex: 12598

Medical only
Mundinter
Intercambio Mundial de Comércio
S.a.r.l.
Av.A.A.de Aguiar 138
P.O. Box 2761
P - **Lisbon**
Tel: (19) 53 21 31/7
Cable: INTERCAMBIO Lisbon

RUMANIA
Hewlett-Packard Reprezentanta
Bd.N. 8alcescu 16
Bucharest
Tel: I58023/138885
Telex: 10440

I.I.R.U.C.
Intreprinderea Pentru
Intretinerea
Si Repararea Utilajelor de Calcul
B-dul prof. Dimitrie Pompei 6
Bucharest-Sectorul 2
Tel: 12 64 30
Telex: 11716

SAUDI ARABIA
Modern Electronic Establishment
King Abdul Aziz str.(Head office)
P.O. Box 1228
Jeddah
Tel: 31173-332201
Cable: ELECTRA

P.D. Box 2728 (Service center)
Riysdh
Tel. 62596-66232
Cable: RAOUFCO

SPAIN
Hewlett-Packard Española, S.A.
Jerez, Calle 3
E-**Madrid** 16
Tel:(1) 458 26 00 (10 lines)
Telex: 23515 hpe

Hewlett-Packard Española, S.A.
Milanesado 21-23
E-**Barcelona** 17
Tel: (3) 203 6200 (5 lines)
Telex: 52603 hpbe e

Hewlett-Packard Española, S.A.
Av Ramdn y Cajal, 1
Edificio Sevilla, planta 9',
E-**Seville** 5
Tel: 64 44 54/58

Hewlett-Packard Española S.A.
Edificio Albia II 7 B
E-**Bilbao**-1
Tel: 23 83 06/23 82 06

Calculators Only
Hewlett-Packard Española S.A.
Gran Via Fernando El Catdlico, 67
E-**Valencia**-8
Tel: 326 67 28/326 85 55

SWEDEN
Hewlett-Packard Sverige AB
Enighetsvägen 1-3
Fack
S-161 20 **Bromma** 20
Tel: (08) 730 05 50
Cable: MEASUREMENTS
Stockholm
Telex: 10721

Hewlett-Packard Sverige AB
Ostra Vintergatan 22
S-702 40 **Orabro**
Tel: (019) 14 07 20

Hewlett-Packard Sverige AB
Frötallsgatan 30
S-421 32 **Västra Frölunda**
Tel: (031) 49 09 50
Telex: 10721 Via Bromma Office

SWITZERLAND
Hewlett-Packard (Schweiz) AG
Zürcherstrasse 20
P.O. Box 307
CH-8952 **Schlieren-Zurich**
Tel: (01) 730 52 40/730 18 21
Cable: HPAG CH
Telex. 53933 hpag ch

Hewlett-Packard (schweiz) AG
Château Bloc 19
CH-1219 **Le Lignon-Geneva**
Tel: (022) 96 03 22
Cable: HEWPACKAG Geneva
Telex: 27 333 hpag ch

SYRIA
Medical/Calculator only
Sawah & Co.
Place Azmé
B.P. 2308
SYR-**Damascus**
Tel: 16367, 19697, 14268
Cable: SAWAH, Oamascus

TURKEY
Telekom Engineering Bureau
P.O. Box 437
Beyoğlu
TR-**Istanbul**
Tel: 49 40 40
Cable: TELEMATION Istanbul
Telex: 23609

Medical only
E.M.A.
Muhendislik Kollektif Sirketi
Adakale Sokak 41/6
TR-**Ankara**
Tel: 175622

Analytical only
Yilmaz Ozyurek
Milli Mudafaa Cad No. 16/6
Kizilay
TR-**Ankara**
Tel: 25 03 09
Telex: 42576 ozek tr

UNITED KINGDOM
Hewlett-Packard Ltd.
King Street Lane
GB-**Winnersh,** Wokingham
Berks. RG11 5AR
Tel: (0734) 78 47 74
Cable: Hewpie London
Telex:847178/9

Hewlett-Packard Ltd.
Trafalgar House,
Navigation Road
Altrincham
Cheshire WA14 1NU
Tel: (061) 928 6422
Telex: 668068

Hewlett-Packard Ltd.
Lygon Court
Hereward Rise
Oudley Road
Halesowen,
West Midlands B62 8SD
Tel: (021) 550 9911
Telex: 339105

Hewlett-Packard Ltd.
Wedge House
799, London Road
GB-**Thornton Heath**
Surrey CR4 6XL
Tel: (01) 684 0103/8
Telex: 946825

Hewlett-Packard Ltd.
c/o Makro
South Service wholesale Centre
Wear Industrial Estate
Washington
GB-**New Town**, County Ourham
Tel Washington 464001 ext. 57/58

Hewlett-Packard Ltd
10, Wesley St.
GB-**Castleford**
West Yorkshire WF10 1AE
Tel: (09775) 50402
Telex: 557355

Hewlett-Packard Ltd
1. Wallace Way
G8-**Hitchin**
Herts
Tel: (0462) 52824/56704
Telex: 825981

Hewlet-Packard Ltd
2C, Avonbeg Industrial Estate
Long Mile Road
Dublin 12
Tel: Oublin 509458
Telex: 30439

USSR
Hewlett-Packard
Representative Office USSR
Pokrovsky Boulevard 4/17-KW 12
Moscow 101000
Tel 294-2024
Telex. 7825 hewpak su

YUGOSLAVIA
Iskra-standard/Hewlett-Packard
Miklosiceva 38/VII
61000 **Ljubljans**
Telb: 31 58 79/32 16 74
Telex: 31583

SOCIALIST COUNTRIES NOT SHOWN PLEASE CONTACT:
Hewlett-Packard Ges.m.b.H
P.D. Box 7
A-1205 **Vienna**, Austria
Tel: (0222) 35 16 21 to 27
Cable: HEWPAK Vienna
Telex: 75923 hewpak a

MEDITERRANEAN AND MIDDLE EAST COUNTRIES NOT SHOWN PLEASE CONTACT:
Hewlett-Packard S.A.
Mediterranean and Middle
East Operations
35, Kolokotroni Street
Platia Kefallariou
GR-Kifissia-**Athens**, Greece
Tel: 8080337/359/429
Telex: 21-6588
Cable: HEWPACKSA Athens

FOR OTHER AREAS NOT LISTED CONTACT
Hewlett-Packard S.A.
7, rue du Bois-du-Lan
P.O. Box
CH-1217 Meyrin 2 - **Geneva**
Switzerland
Tel: (022) 82 70 00
Cable: HEWPACKSA Geneva
Telex: 2 24 86

UNITED STATES

ALABAMA
8290 Whitesburg Dr., S.E.
P.D. Box 4207
Huntsville 35802
Tel: (205) 881-4591

Medical Only
228 W. Valley Ave.
Room 220
Birmingham 35209
Tel: (205) 942-2081/2

ARIZONA
2336 E. Magnolia St.
Phoenix 85034
Tel: (602) 244-1361

2424 East Aragon Rd.
Tucson 85706
Tel: (602) 294-3148

*ARKANSAS
Medical Service Only
P.O. Box 5646
Brady Station
Little Rock 72215
Tel: (501) 376-1844

CALIFORNIA
1430 East Orangethorpe Ave.
Fullerton 92631
Tel: (714) 870-1000

3939 Lankershim Boulevard
North Hollywood 91604
Tel: (213) 877-1282
TWX: 910-499-2671

5400 West Rosecrans Blvd.
P.O. Box 92105
World Way Postal Center
Los Angeles 90009
Tel: (213) 970-7500

***Los Angeles**
Tel: (213) 776-7500

3003 Scott Boulevard
Santa Clara 95050
Tel: (408) 249-7000
TWX: 910-338-0518

***Ridgecrest**
Tel: (714) 446-6165

646 W. North Market Blvd
Sacramento 95834
Tel: (916) 929-7222

9606 Aero Drive
P.O. Box 23333
San Diego 92123
Tel: (714) 279-3200

COLORADO
5600 South Ulster Parkway
Englewood 80110
Tel: (303) 771-3455

CONNECTICUT
12 Lunar Drive
New Haven 06525
Tel: (203) 389-6551
TWX: 710-465-2029

FLORIDA
P.D. Box 24210
2806 W. Oakland Park Blvd.
Ft. Lauderdale 33311
Tel: (305) 731-2020

***Jacksonville**
Medical Service only
Tel: (904) 398-0663

P.D. Box 13910
6177 Lake Ellenor Dr.
Orlando 32809
Tel: (305) 859-2900

P.D. Box 12826
Pensacola 32575
Tel: (904) 476-8422

GEORGIA
P.O. Box 105005
Atlanta 30348
Tel: (404) 955-1500
TWX:810-766-4890

Medical Service Only
***Augusta** 30903
Tel: (404) 736-0592

P.O. Box 2103
Warner Robins 31098
Tel: (912) 922-0449

HAWAII
2875 So. King Street
Honolulu 96814
Tel: (808) 955-4455
Telex: 723-705

ILLINOIS
5201 Tollview Dr.
Rolling meadows 60008
Tel: (312) 255-9800
TWX: 910-687-2260

INDIANA
7301 North Shadeland Ave.
Indianapolis46250
Tel: (317)842-1000
TWX: 810-260-1797

IOWA
2415 Heinz Road
Iowa City 52240
Tel: (319) 338-9466

KENTUCKY
Medical Only
Atkinson Square
3901 Atkinson Dr..
Suite 407 Atkinson Square
Louisville 40218
Tel: (502) 456-1573

LOUISIANA
P.D. Box 840
3229-39 Williams Boulevard
Kenner 70063
Tel: (504) 443-6201

MARYLAND
6707 Whitestone Road
Baltimore 21207
Tel: (301) 944-5400
TWX: 710-862-9157

2 Choke Cherry Road
Rockville 20850
Tel: (301) 948-6370
TWX: 710-828-9684

MASSACHUSETTS
32 Hartwell Ave.
Lexington 02173
Tel: (617) 861-8960
TWX: 710-326-6904

MICHIGAN
23855 Research Drive
Farmington Hills 48024
Tel: (313) 476-6400

724 West Centre Ave.
Kalamazoo 49002
Tel: (606) 323-8362

MINNESOTA
2400 N. Prior Ave.
St. Paul 55113
Tel: (612) 636-0700

MISSISSIPPI
***Jackson**
Medical Service only
Tel: (601) 982-9363

MISSOURI
11131 Colorado Ave.
Kansas City 64137
Tel: (816) 763-8000
TWX: 910-771-2087

1024 Executive Parkway
St. Louis 63141
Tel: (314) 878-0200

NEBRASKA
Medical Only
7171 Mercy Road
Suite 110
Omaha 68106
Tel: (402) 392-0948

NEW JERSEY
W. 120 Century Rd.
Paramus 07652
Tel: (201) 265-5000
TWX: 710-990-4951

Crystal Brook Professional
Building
Eatontown 07724
Tel:(201) 542-1384

NEW MEXICO
P.D. Box 11634
Station E
11300 Lomas Blvd., N.E.
Albuquerque 87123
Tel: (505) 292-1330
TWX: 910-989-1185

156 Wyatt Drive
Las Cruces 88001
Tel: (505) 526-2484
TWX: 910-9983-0550

NEW YORK
6 Automation Lane
Computer Park
Albany 12205
Tel: (518) 458-1550

201 South Avenue
Poughkeepsie 12601
Tel: (914) 454-7330
TWX: 510-253-5981

650 Perinton Hill Office Park
Fairport 14450
Tel: (716) 223-9950

5858 East Molloy Road
Syracuse 13211
Tel: (315) 454-2486
TWX: 710-541-0482

1 Crossways Park West
Woodbury 11797
Tel: (516) 921-0300
TWX: 710-990-4951

NORTH CAROLINA
P.O. Box 5188
1923 North Main Street
High Point 27262
Tel: (919) 885-8101

OHIO
16500 Sprague Road
Cleveland 44130
Tel: (216) 243-7300
TWX: 810-423-9430

330 Progress Rd.
Dayton 45449
Tel: (513) 859-8202

1041 Kingsmill Parkway
Columbus 43229
Tel: (614) 436-1041

OKLAHOMA
P.O. Box 32008
Oklahoma City 73132
Tel: (405) 721-0200

OREGON
17890 SW Lower Boones
Ferry Road
Tualatin 97062
Tel: (503) 620-3350

PENNSYLVANIA
111 Zeta Drive
Pittsburgh 15238
Tel: (412) 782-0400

1021 8th Avenue
King of Prussia Industrial Park
King of Prussia 19406
Tel: (215) 265-7000
TWX: 510-660-2670

SOUTH CAROLINA
6941-0 N. Trenholm Road
Columbia 29260
Tel: (803) 782-6493

TENNESSEE
***Knoxville**
Medical Service only
Tel: (615) 523-5022

3027 Vanguard Dr.
Director's Plaza
Memphis 38131
Tel: (901) 346-8370

Nashville
Medical Service only
Tel: (615) 244-5448

TEXAS
P.O. Box 1270
201 E. Arapaho Rd.
Richardson 75080
Tel: (214) 231-6101

10535 Harwin Dr.
Houston 77036
Tel: (713) 776-6400

205 Billy Mitchell Road
San Antonio 78226
Tel: (512) 434-8241

UTAH
2160 South 3270 West Street
Salt Lake City 84119
Tel: (801) 972-4711

VIRGINA
P.O. Box 12778
No. 7 Koger Exec. Center
Suite 212
Norfolk 23502
Tel:(804) 461-4025/6

P.O.Box 9669
2914 Hungary Springs Road
Richmond 23228
Tel: (804) 285-3431

WASHINGTON
Bellefield Office Pk.
1203-114th Ave. S.E.
Bellevue 98004
Tel: (206) 454-3971
TWX: 910-443-2446

*WEST VIRGINIA
Medical/Analytical Only
Charleston
Tel: (304) 345-1640

WISCONSIN
9004 West Lincoln Ave.
West Allis 53227
Tel: (414) 541-0550

FOR U.S. AREAS NOT LISTED:
Contact the regional office nearest you: Atlanta, Georgia... North Hollywood, California.. Rockville, Maryland...Rolling Meadows, Illinois. Their complete addresses are listed above.

***Service Only**

1/78

[illegible]
PRINTED IN U.S.A.

HEWLETT hp PACKARD

[illegible]
[illegible]