HEWLETT **hp** *PACKARD*

HP 3000 SERIES III
COMPUTER SYSTEM

REFERENCE/TRAINING MANUAL

Manual Part No. 30000-90143

Printed in U.S.A. 6/79

LIST OF EFFECTIVE PAGES

The List of Effective Pages gives the date of the current edition and of any pages changed in updates to that edition. Within the manual, any page changed since the last edition is indicated by printing the date the changes were made on the bottom of the page. Changes are marked with a vertical bar in the margin. If an update is incorporated when an edition is reprinted, these bars are removed but the dates remain. No information is incorporated into a reprinting unless it appears as a prior update.

All pages in this manual are original issue.

PRINTING HISTORY

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The date on the title page and back cover of the manual changes only when a new edition is published. When an edition is reprinted, all the prior updates to the edition are incorporated. No information is incorporated into a reprinting unless it appears as a prior update. The edition does not change.

First Edition. . . . . Jun 1979

This manual contains hardware-oriented reference information for the HP 3000 Series III Computer Systems. Specifically, this manual contains reference reading material for all persons that are to attend Hewlett-Packard's 3000 Series III Computer System Hardware Training Courses. Since the information contained in this manual is approximately the same as that presented during classroom lectures, this manual should be used for classroom reference, note taking purposes, and post school reference.

The HP 3000 Series III Computer Systems are divided into two product lines; the HP 32421A Series III and the HP 32435A Series III. Unless otherwise stated, the content of this manual applies equally to both product lines.

CONTENTS

CONTENTS (continued)

CONTENTS (continued)

CONTENTS (continued)

CONTENTS (continued)

CONTENTS (continued)

CONTENTS (continued)

CONTENTS (continued)

CONTENTS (continued)

CONTENTS (continued)

ILLUSTRATIONS (continued)

ILLUSTRATIONS (continued)

TABLES

The HP 3000 Series III Computer Systems are general purpose computers with true multiprogramming and multilingual capabilities. They can simultaneously handle many interactive and batch operations; each in any of several programming languages. The HP 3000 Series III Computer Systems feature hardware stack architecture, variable-length code segmentation in a hardware-assisted virtual memory scheme, user protection, dynamic storage allocation, and integrated hardware/software design. The hardware and software work together in an interrelated manner with the hardware performing many operations conventionally performed by software. The HP 3000 Series III Computer Systems have a single, comprehensive operating system, the Multiprogramming Executive (MPE). MPE is a general-purpose, disc-based software system that supervises the processing of user programs. MPE relieves the user of many program control, input/output, and other housekeeping responsibilities by monitoring and controlling the compilation, run preparation, loading, execution, and output of user programs. MPE also controls the order in which programs are executed and allocates the hardware and software resources they require.

1-1. SYSTEM FEATURES

The HP 3000 Series III Computer Systems incorporate many features usually found only on very large computer systems. These features are summarized in paragraphs 1-2 through 1-10.

1-2. Stack Architecture

The system's stack architecture provides private, hardware-protected data storage for each user as well as an automatic method for moving this data to and from the central processor registers. The major operating features derived from this design are:

a. Fast execution

b. Code compression

c. Hardware-protected execution

d. Dynamic allocation of subprogram data space

e. Ease of parameter passing

f. Efficient subprogram linkage

g. Rapid interruption and restoration of user environments

h. Subprograms being able to call themselves (recursion)

1-3. Microprogrammed Operations

Many system operations that were previously programmed in software are now microprogrammed. These operations are requested by machine instructions which in turn execute multiple microinstructions built into the central processor hardware. Microprogramming eliminates repetitive coding otherwise required for recurring operations.

1-4. Data Base Management Facilities

The computer systems provide software facilities that allow the user to create, access, and maintain large data bases. The information in these bases can be accessed both interactively from a keyboard terminal and programmatically from user programs written in any of the the available programming languages.

1-5. Five Programming Languages

The computer systems provide the user with a true multilingual programming environment. The six available languages are COBOL, RPG, FORTRAN, BASIC, SPL (a language developed especially for the HP 3000 Series Computers), and APL.

1-6. Virtual Memory

The operating system's hardware-assisted virtual memory scheme offers each user program a memory space that exceeds the maximum main memory size of 1024K words. Virtual memory consists of both main memory and a flexible storage area on disc. Virtual memory is implemented using a segment trap frequency algorithm that ensures the automatic presence in main memory of only those segments of code and data which are currently required by the executing program. Main memory is thus efficiently shared by the users in a manner that gives each programmer the impression of working with a much larger computer system.

1-7. Fault Control Memory

The computer systems employ high-speed semiconductor memory modules that provide automatic fault detection and single-bit correction with no loss in performance.

1-8. Concurrent I/O and CPU Operations

Many I/O operations can be performed concurrently with Central Processor Unit (CPU) and memory operations. This is possible because, in addition to the CPU, the computer has an Input/Output Processor (IOP) with its own dedicated data transfer path (IOP bus) to which are connected a Multiplexer Channel(s) and one or more Asynchronous Terminal Controllers. All of this hardware operates under control of the MPE operating system which handles all queuing and device scheduling.

1-9. Reentrant Code and Private Data

Within the MPE environment, many user and system functions can be
active concurently without interferring with each other because
the hardware provides protection of programs and guarantees the
privacy of user data areas. The hardware keeps code and data
physically separate by organizing them into reentrant code seg-
ments and data segments. (The code segments can be shared among
users, but not altered. The data segments cannot be shared, but
can be altered by the creating user.) This segmenting ability
facilitates the operation of virtual memory in that 1; code seg-
ments need never be swapped out since an identical copy always
exists on disc, and 2; code segments can be swapped indirectly
from wherever the program file resides on disc without having to
be copied first to a special swapping disc.

1-10. Operating System

A single, comprehensive operating system (MPE) supervises the
processing of all user programs and provides the user with an
extensive set of system functions. The major features of MPE are:

a. Interchangeable batch and interactive processing

b. Uniform, device-independent, and language-independent file
    system

c. File coordination and security

d. Input and output spooling (concurrent usuage of I/O devices)

e. Console job control

f. Automatic scheduling (under control of the installation's
    management)

g. System back-up facility

h. Power fail/auto restart

i. System tailoring (under control of installation's management)

j. System logging facility

1-11. HARDWARE FEATURES

The hardware design of the HP 3000 Series III Computer System
will be discussed in detail throughout the remainder of this man-
ual. Briefly, the hardware features are:

a. Up to 1024K words of high-speed, fault correcting, semicon-
    ductor memory

b. High-speed selector channels for block transfers between main
    memory and high-speed I/O devices such as discs

c. I/O multiplexer channels for word transfers between main memory and low- to medium-speed I/O devices such as card readers, line printers, and magnetic tape units

d. Asynchronous terminal controllers for data transmissions between main memory and interactive terminals

e. High-speed disc storage devices that provide storage capacities from 15 to 120 million bytes and data transfers of nearly one megabyte per second

f. 800 or 1600 character-per-inch magnetic tape units

g. Line printers with operating speeds from 165 to 1800 lines per minute

h. CRT display terminals

i. Card readers and high-speed punched tape equipment

1-12. SOFTWARE FEATURES

The HP 3000 Series III Computer Systems offer a wide range of software including the MPE operating system, six programming languages, a text editor, a flexible file copier, a fast sort/ merge package, two libraries of commonly used mathematical, statistical, and utility procedures, data base management facility, and data communications products. Currently available software is shown in figure 1-1.

1-13. SYSTEM CONFIGURATIONS

The HP 3000 Series III Computer Systems are available in two product lines; the 32421A Series III and the 32435A Series III. The 32421A Series III is available in two hardware models; a standard 2-bay model and an optional 3-bay model (Option 200). The 32435A Series III is also available in two hardware models; a standard 1-bay model and an optional 2-bay model (Option 200). All models use the same operating system, language processors, utility programs, data base management programs, and data communications programs. All models operate in both batch and interactive modes with full spooling capabilities. Rack layouts for the four models are shown in figures 1-2 through 1-5. (HP 29425A Cabinets that contain the system discs are not shown.) The printed circuit assembly (PCA) slot assignments for the models are listed in tables 1-1 through 1-3.

OPERATING SYSTEM

Configurator	Initiator	System Console Manager	Command Interpreter	File Management System	Input/ Output System
Virtual Memory Manager	Disc Space Manager	Private Volumes Facility	Serial Disc Interface	Tape Labels Facility	
Spooling Facility	Job/Session Scheduler	Process Dispatcher	Segmenter	Loader	User Trap Manager
Utility Intrinsics	Accounting Facility	Logging Facility	Backup/ Restore Facility	Power Fail/ Auto Restart	

LANGUAGES

COBOL	RPG	FORTRAN	BASIC	SPL	APL

UTILITIES

Text Editor	File Copier	Sort/Merge	Compiler Library	Scientific Library	Data Entry Library

DATA MANAGEMENT

KSAM	DBMS (Image & Query)	FORMS

DATA COMMUNICATIONS

DS	RJE	MRJE	MTS

Figure 1-1. HP 3000 Series III Computer System Software
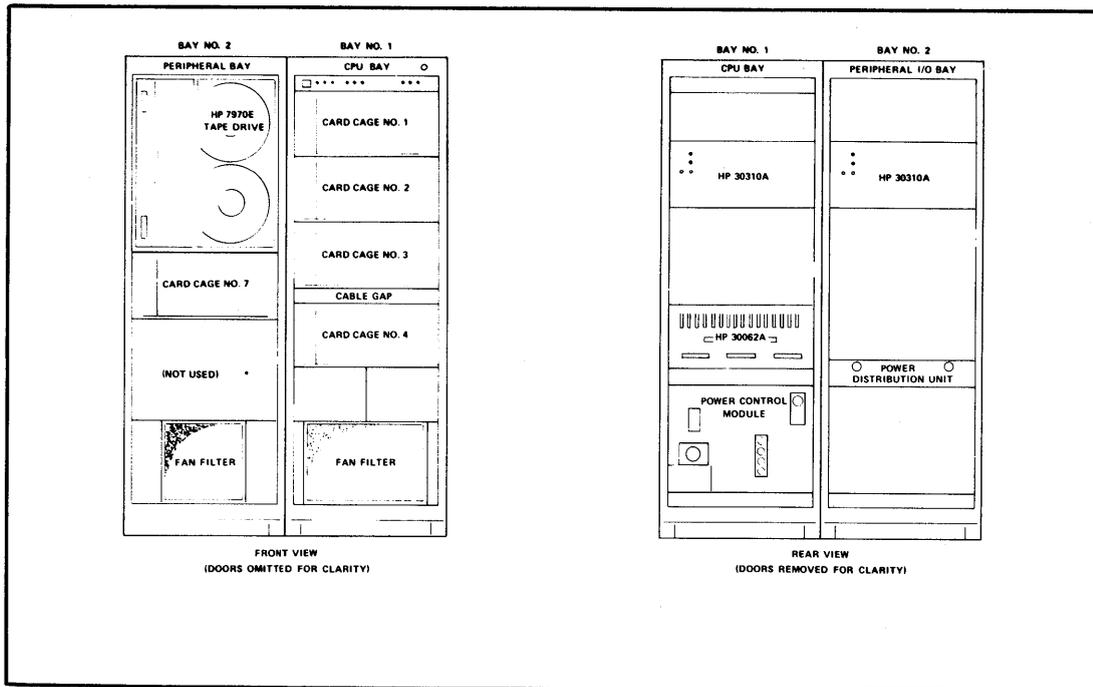
Figure 1-2. HP 32421A Series III Computer System, 2-Bay Model
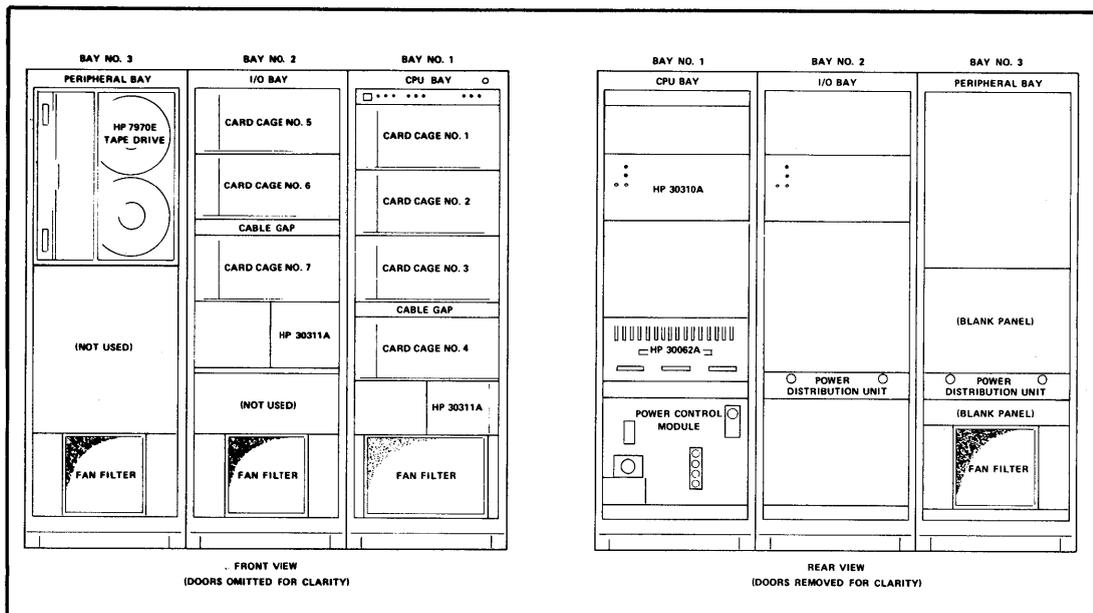
Figure 1-3. HP 32421A Series III Computer System, 3-Bay Model
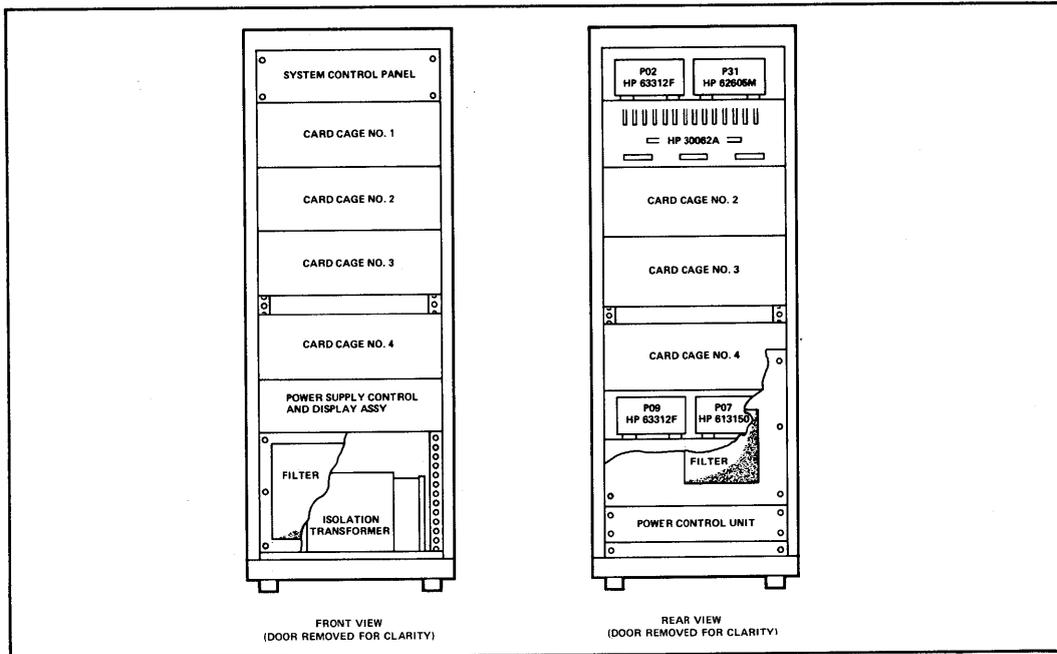
Figure 1-4. HP 32435A Series III Computer System, 1-Bay Model
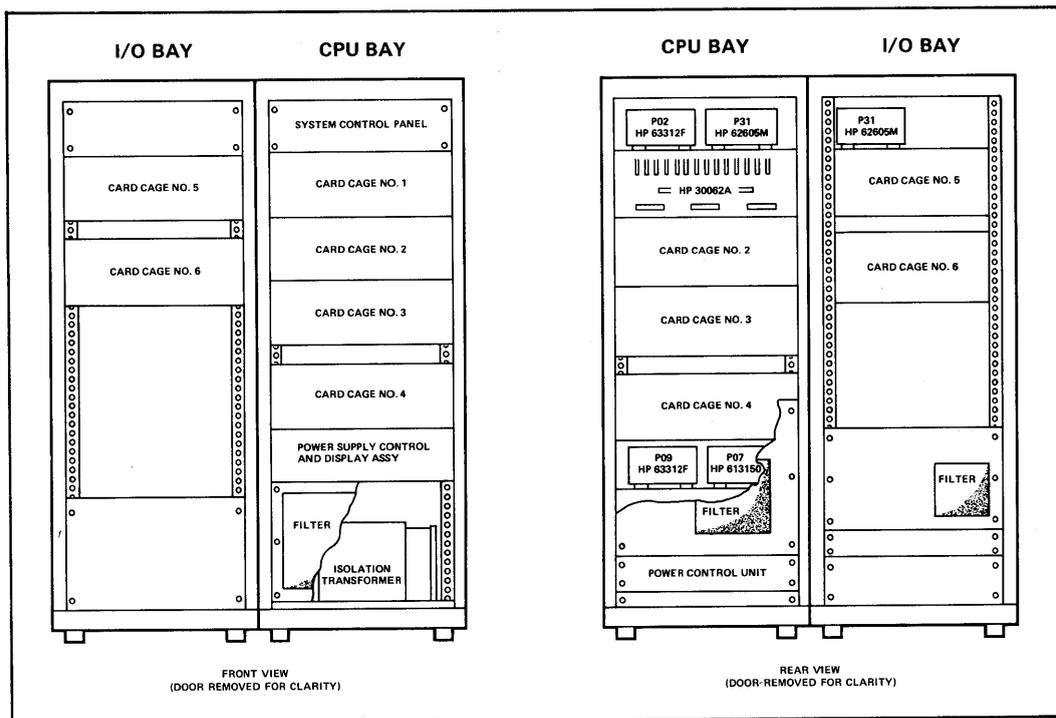
Figure 1-5. HP 32435A Series III Computer System, 2-Bay Model

SLOT	PRINTED CIRCUIT ASSEMBLY

	SLOT	PRINTED CIRCUIT ASSEMBLY
	A1	Reserved for maintenance panel PCA.
	A2	30012-60001 Expanded Read Only Memory
	A3	30003-60021 Read Only Memory
	A4	30003-60022 Skip and Special Field
CARD	A5	30003-60003 Arithmetic and Logic Unit
CAGE	A6	30003-60004 R Bus
NO.1	A7	30003-60025 S Bus
	A8	30003-60006 Current Instruction Register
	A9	30003-60007 Module Control Unit
	A10	30003-60028 Input Output Processor
	A1	30032-60001 Terminal Data Interface
	A2	30061-60001 Terminal Control Interface
	A3	Reserved for 204 Modem capability
	A4	30009-60002 Fault Logging Interface
CARD	A5	
CAGE	A6	30008-60003 Memory Array (128K)
NO.2	A7	Available to add 128K
	A8	Available to add 128K
	A9	Available to add 128K
	A10	30007-60005 Memory Control and Logging
	A1	Available for add-on memory
	A2	Available for add-on memory
	A3	Available for add-on memory
	A4	Available for add-on memory
CARD	A5	Available for add-on memory
CAGE	A6	
NO.3	A7	30030-60020 Selector Channel Port Controller
	A8	30030-60021 Selector Channel Register
	A9	30030-60003 Selector Channel Control
	A10	30030-60011 Selector Channel Sequencer
	A1	30036-60002 Multiplexer Channel
	A2	Available for programmed (SIO) or direct I/O.
	A3	Available for programmed (SIO) or direct I/O.
	A4	Reserved for Selector Channel maintenance.
CARD	A5	30031-60001 System Clock
CAGE	A6	Reserved for maintenance.
NO.4	A7	30229-60001 7905A/20A/25A Interface
	A8	13037-60028 Disc Controller
	A9	13037-60024 Error Correction
	A10	13037-60001 Microprocessor

	SLOT	PRINTED CIRCUIT ASSEMBLY
	A1	Available for programmed (SIO) or direct I/O
	A2	30215-60002 Magnetic Tape Controller Processor
	A3	30215-60006 Magnetic Tape Controller
	A4	Available for programmed (SIO) or direct I/O
CARD	A5	Available for programmed (SIO) or direct I/O
CAGE	A6	Available for programmed (SIO) or direct I/O
NO.7	A7	Available for programmed (SIO) or direct I/O
	A8	Available for programmed (SIO) or direct I/O
	A9	Available for programmed (SIO) or direct I/O
	A10	Available for programmed (SIO) or direct I/O

Table 1-1. HP 32421A Series III 2-Bay Model PCA Slot Assignments

	SLOT	PRINTED CIRCUIT ASSEMBLY		SLOT	PRINTED CIRCUIT ASSEMBLY
CARD CAGE NO.5	A1	30036-60002 Multiplexer Channel	CARD CAGE NO.1	A1	Reserved for maintenance panel PCA.
	A2	30215-60002 Magnetic Tape Controller Processor		A2	30012-60001 Expanded Read Only Memory
	A3	30215-60006 Magnetic Tape Controller		A3	30003-60021 Read Only Memory
	A4	30031-60001 System Clock		A4	30003-60022 Skip and Special Field
	A5	Available for programmed (SIO) or direct I/O		A5	30003-60003 Arithmetic and Logic Unit
	A6	Available for programmed (SIO) or direct I/O		A6	30003-60004 R Bus
	A7	Available for programmed (SIO) or direct I/O		A7	30003-60025 S Bus
	A8	Available for programmed (SIO) or direct I/O		A8	30003-60006 Current Instruction Register
	A9	Available for programmed (SIO) or direct I/O		A9	30003-60007 Module Control Unit
	A10	Available for programmed (SIO) or direct I/O		A10	30003-60028 Input Output Processor
CARD CAGE NO.6	A1	Available for programmed (SIO) or direct I/O	CARD CAGE NO.2	A1	30032-60001 Terminal Data Interface
	A2	Available for programmed (SIO) or direct I/O		A2	30061-60001 Terminal Control Interface
	A3	Available for programmed (SIO) or direct I/O		A3	Reserved for 203 Modem capability
	A4	Available for programmed (SIO) or direct I/O		A4	30009-60002 Fault Logging Interface
	A5	Available for programmed (SIO) or direct I/O		A5	
	A6	Available for programmed (SIO) or direct I/O		A6	30008-60003 Memory Array (128K)
	A7	Available for programmed (SIO) or direct I/O		A7	Available to add 128K.
	A8	Available for programmed (SIO) or direct I/O		A8	Available to add 128K.
	A9	Available for programmed (SIO) or direct I/O		A9	Available to add 128K.
	A10	Available for programmed (SIO) or direct I/O		A10	30007-60005 Memory Control and Logging
CARD CAGE NO.7	A1	Available for programmed (SIO) or direct I/O	CARD CAGE NO.3	A1	Available for add-on memory
	A2	Available for programmed (SIO) or direct I/O		A2	Available for add-on memory
	A3	Available for programmed (SIO) or direct I/O		A3	Available for add-on memory
	A4	Available for programmed (SIO) or direct I/O		A4	Available for add-on memory
	A5	Available for programmed (SIO) or direct I/O		A5	Available for add-on memory
	A6	Available for programmed (SIO) or direct I/O		A6	
	A7	Available for programmed (SIO) or direct I/O		A7	30030-60020 Selector Channel Port Controller
	A8	Reserved for second disc controller		A8	30030-60021 Selector Channel Register
	A9	Reserved for second disc controller		A9	30030-60003 Selector Channel Control
	A10	Reserved for second disc controller		A10	30030-60011 Selector Channel Sequencer
			CARD CAGE NO.4	A1	Reserved for second Selector Channel
				A2	Reserved for second Selector Channel
				A3	Reserved for second Selector Channel
				A4	Reserved for Selector Channel maintenance.
				A5	Reserved for second 7905A/20A/25A Interface
				A6	Reserved for maintenance.
				A7	30229-60001 7905A/20A/25A Interface
				A8	13037-60028 Disc Controller
				A9	13037-60024 Error Correction
				A10	13037-60001 Microprocessor

Table 1-2. HP 32421A Series III 3-Bay Model PCA Slot Assignments

	Slot	PRINTED CIRCUIT ASSEMBLY
CARD CAGE NO. 5	A1	Available for programmed (SIO) or direct I/O
	A2	Available for programmed (SIO) or direct I/O
	A3	Available for programmed (SIO) or direct I/O
	A4	Available for programmed (SIO) or direct I/O
	A5	Available for programmed (SIO) or direct I/O
	A6	Available for programmed (SIO) or direct I/O
	A7	Available for programmed (SIO) or direct I/O
	A8	Available for programmed (SIO) or direct I/O
	A9	Available for programmed (SIO) or direct I/O
	A10	Available for programmed (SIO) or direct I/O
CARD CAGE NO. 5	A1	Available for programmed (SIO) or direct I/O
	A2	Available for programmed (SIO) or direct I/O
	A3	Available for programmed (SIO) or direct I/O
	A4	Available for programmed (SIO) or direct I/O
	A5	Available for programmed (SIO) or direct I/O
	A6	Available for programmed (SIO) or direct I/O
	A7	Available for programmed (SIO) or direct I/O
	A8	Available for programmed (SIO) or direct I/O
	A9	Available for programmed (SIO) or direct I/O
	A10	Available for programmed (SIO) or direct I/O

I/O BAY

	Slot	PRINTED CIRCUIT ASSEMBLY
CARD CAGE NO. 1	A1	Reserved for maintenance panel PCA.
	A2	30012-60001 Expanded Read Only Memory.
	A3	30003-60021 Read Only Memory
	A4	30003-60022 Skip and Special Field
	A5	30003-60003 Arithmetic and Logic Unit
	A6	30003-60004 R Bus
	A7	30003-60025 S Bus
	A8	30003-60006 Current Instruction Register
	A9	30003-60007 Module Control Unit
	A10	30003-60028 Input Output Processor
CARD CAGE NO. 2	A1	30008-60003 Memory Array (128K)
	A2	Available to add 128K
	A3	Available to add 128K
	A4	Available to add 128K
	A5	30007-60005 Memory Control and Logic #1
	A6	Available to add Memory Control and Logic #2
	A7	Available to add 128K
	A8	Available to add 128K
	A9	Available to add 128K
	A10	Available to add 128K
CARD CAGE NO. 3	A1	30135-60063 System Clock/FLI
	A2	30032-60001 Terminal Data Interface
	A3	30061-60001 Terminal Control Interface
	A4	30030-60020 Selector Channel Port Controller
	A5	30030-60021 Selector Channel Register
	A6	30030-60003 Selector Channel Control
	A7	30030-60011 Selector Channel Sequencer
	A8	Available for programmed (SIO) or direct I/O
	A9	Available for programmed (SIO) or direct I/O
	A10	Available for programmed (SIO) or direct I/O
CARD CAGE NO. 4	A1	Available for programmed (SIO) or direct I/O
	A2	Available for programmed (SIO) or direct I/O
	A3	Available for programmed (SIO) or direct I/O
	A4	Available for programmed (SIO) or direct I/O
	A5	Available for programmed (SIO) or direct I/O
	A6	Available for programmed (SIO) or direct I/O
	A7	30215-60002 Magnetic Tape Controller Processor
	A8	30215-60006 Magnetic Tape Controller
	A9	30036-60002 Multiplexer Channel
	A10	30229-60001 Disc Control Interface

CPU BAY

Table 1-3. HP 32435A Series III PCA Slot Assignments

NOTES

NOTES

This section contains a brief description of the computer system's hardware organization and detailed discussions of the system's operating environment, instruction formats, addressing conventions, and CPU operations. The topics that are summarized in this section are discussed in more detail throughout the remainder of this manual. In addition, this section contains principles of operation and servicing information for the CPU.

2-1 HARDWARE ORGANIZATION

The hardware elements of the computer system are organized as shown in figure 2-1. This basic structure of independent modules organized around a Central Data (CTL) Bus permits high-speed internal data rates. When not communicating over the CTL Bus, each module can run independently at its own speed. This structure also allows new equipment to be added without going through a major hardware reconfiguration. The separate Input/ Output Processor (IOP) Bus is totally dedicated to input/output (I/O) data transfers which allows the computer system to immediately respond to I/O device needs regardless of what transfers are currently in progress between the various system modules. The IOP Bus also permits many I/O operations to be handled concurrently with CPU, Main Memory, and Selector Channel operations. Data can be transferred directly between Main Memory (Bank 0 through Bank 15) and high-speed I/O devices in block mode via the Selector Channel Bus, Selector Channel, Port Controller, and CTL Bus. For lower-speed I/O devices, data can be multiplexed on a word-by-word basis via the IOP, IOP Bus, Multiplexer Channel, and Multiplexer Channel Bus. In both cases, the I/O channels operate in parallel with CPU operations. In addition, I/O devices attached to the IOP Bus can be directly controlled through the use of the CPU's direct I/O instructions.

2-2. Bus System

The computer's bus system is a network of data, control, and power lines necessary to effect the transfer of data between computer modules and between I/O devices and memory. The individual buses are discussed in paragraphs 2-3 through 2-8.

2-3. CTL BUS. The CTL Bus provides the communications path between all computer modules. This bus consists of a 50-conductor flat cable and connectors and is connected to each Module Control Unit (MCU) and Port Controller in the computer system. (Refer to paragraph 2-15.)
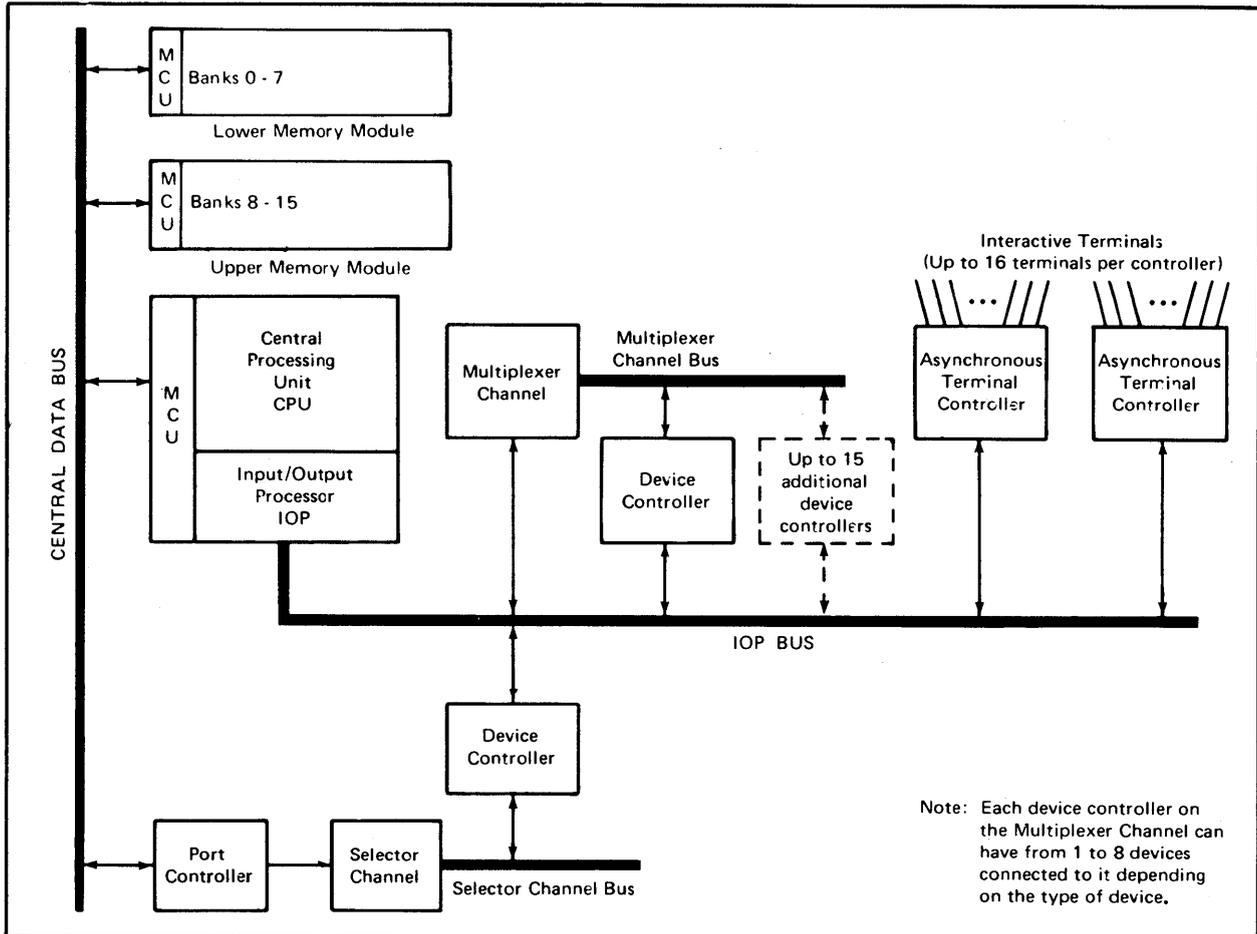
Figure 2-1. HP 3000 Series III Computer System Hardware
Organization

2-4. IOP BUS. The IOP Bus provides the means for the IOP to
send control signals and control words to any Device Controller
and for the IOP to accept interrupts from any Device Controller.
(For multiplexed I/O devices, all data transmissions also occur
via the IOP Bus.   For high-speed devices connected to the Selec-
tor Channel Bus, data transmissions for direct I/O instructions
also occur via the IOP Bus.) This bus consists of a 50-conductor
flat cable and connectors and connects the IOP to every Device
Controller and Multiplexer Channel in the computer system.

2-5. SELECTOR CHANNEL BUS. The Selector Channel Bus (one for
each Selector Channel)   provides the communication path for the
Selector Channel to select one of up to eight I/O devices for
transmission.   Data transmissions on the Selector Channel Bus,
occuring as a result of an SIO instruction, are by block transfer
(data burst). This bus consists of a 50-conductor flat cable and
connectors and connects a Selector Channel to each of its as-
sociated high-speed Device Controllers.

2-6. PORT CONTROLLER BUS. The Port Controller Bus (not shown in figure 2-1) provides the communication path between each Selector Channel and the Port Controller which interfaces with the CTL Bus. This bus consists of a 50-conductor flat cable and connectors and connects each Selector Channel to the Port Controller.

2-7. MULTIPLEXER CHANNEL BUS. Except for minor signal nomenclature differences, the Multiplexer Channel Bus (one per Multiplexer Channel) is identical to the Selector Channel Bus. This allows certain high-speed I/O devices to be connected interchangeably to either bus. The major difference is that data transmissions are under control of a Multiplexer Channel instead of a Selector Channel. All data transmissions in this case are via the IOP Bus and are multiplexed among the I/O devices on a word-by-word basis. (The equivalent data lines on the Selector Channel Bus are used as service request lines on the Multiplexer Channel Bus.) This bus consists of a 50-conductor flat cable and connectors and connects each Multiplexer Channel to each of its associated Device Controllers.

2-8. POWER BUS. The Power Bus (not shown in figure 2-1), unlike the previously discussed flat-cable buses, is a rigid PCA with fixed 56-pin connectors. The Power Bus provides dc power and and some IOP Bus related signals for each PCA mounted in a particular card cage module. There is one Power Bus for each card cage module and each Power Bus is individually wired to the computer's power supply. Although dc power is not distributed from card cage module to card cage module via the Power Bus, a 20-conductor flat cable is connected between the Power Buses for the distribution of the IOP Bus related signals. In addition, each Power Bus contains connector pins reserved for the data poll, interrupt poll, and system clock signals.

2-9. Functional Hardware Elements

Brief descriptions of the principal hardware elements illustrated in figure 2-1 are contained in paragraphs 2-10 through 2-14.

2-10. CENTRAL PROCESSOR MODULE. The Central Processor Module determines the basic characteristics of the computer system's hardware and consists of the MCU, CPU, and IOP. Significant features of the module are listed in table 2-1.

The MCU resolves CTL Bus priority conflicts between the CPU and IOP and interfaces both to the CTL Bus. Refer to paragraph 2-15. A detailed discussion of the MCU is contained in Section VI.

The CPU translates received instruction words into microprogram starting addresses, decodes microprograms into fixed control signal sequences, executes various arithmetic functions, and either transfers the results out of the Central Processor Module or stores the results in various internal registers for future use. The CPU shares the MCU with the IOP. A detailed discussion of the CPU is contained in paragraphs 2-71 through 2-133.

The IOP provides the I/O control link for the computer system and resolves priority conflicts for I/O interrupts and multiple Multiplexer Channel access to the CTL Bus. The IOP performs different functions for each of the three I/O transfer modes. (Refer to Section VII.) During direct I/O transfer mode and depending on received direct I/O instructions (RIO, WIO, TIO, CIO, SIN, and SMSK), the IOP transfers either data, device status, or control information between the CPU and a Device Controller. During programmed I/O transfer mode via a Multiplexer Channel, the IOP transfers I/O program words between memory and the Multiplexer Channel, and transfers data between memory and a Multiplexer-Channel-selected Device Controller. During programmed I/O transfer mode via a Selector Channel, the IOP only transfers initialization information to a Device Controller associated with the Selector Channel; it does not become involved in any part of the I/O program execution. During all I/O transfer modes, the IOP interrupts the CPU on behalf of the Device Controllers. The IOP shares the MCU with the CPU.

Physically, the Central Processor Module consists of nine PCA's contained in slots A2 through A10 of Card Cage No. 1 of all HP 3000 Series III Computer System models. Card Cage No. 1 is a dedicated card cage module and the nine PCA's must be installed exactly as shown in tables 1-1 through 1-3.

2-11. MAIN MEMORY. Main Memory is a high-speed, semiconductor, random access memory that provides high-speed storage for the computer system. Main Memory operates as an error correcting memory with single-bit fault correction and some double-bit detection. (Main Memory can operate as a non-error correcting memory with a parity bit, but this is not the normal operating mode.) Main Memory can vary in size from 128K (K=1024) words to 1024K words and, due to its modular design, it can easily be expanded from one size to another.

A maximum word capacity system consists of 16 64K-word memory banks (Bank 0 through Bank 15) divided into two 512K-word memory modules. Each 512K-word memory module contains its own MCU which controls word transfers between the module and the other system modules connected to the CTL Bus. The word length transmitted over the CTL Bus is 17 bits; 16 bits of data (one word or two bytes) and one parity bit. (Within the memory modules, word length is expanded to 22 bits; 16 bits of data and six bits for automatic fault detection and correction.) A detailed discussion of Main Memory is contained in Section VI.

Physically, Main Memory consists of three basic PCA's configured as shown in table 2-2. It should be noted that each Semiconductor Memory Array (SMA) PCA contains 128K words of memory, that one Memory Control and Logging (MCL) PCA can support up to four SMA PCA's (512K), and that one Fault Logging Interface (FLI) PCA can support the computer system's maximum memory capacity of eight SMA PCA's (1024K). The Main Memory PCA's are arranged in Card Cages No. 2 and No. 3 as shown in tables 1-1 through 1-3. Conventionally, card cage slots 2A6 through 2A9 (HP 32421A Series III)

Table 2-1. Central Processor Module Features

```
ARCHITECTURE

   Hardware-implemented stack
   Separate code and data
   Non-modifiable reentrant code
   Variable-length code segmentation
   Virtual memory for code
   Dynamic relocatability of programs

IMPLEMENTATION

   Microprogrammed CPU
   175 nanosecond microinstruction time
   Automatic restart after power failure
   CTL Bus
   Bus parity checking
   Concurrent CPU and I/O operations

INSTRUCTIONS

   209 instructions
   All instructions except stack operations are 16 bits
      in length. (Stack operations can be packed two per
      per word.)
   16- and 32-bit integer arithmetic
   32- and 64-bit floating point arithmetic
   28-digit packed decimal arithmetic
   Special instructions that optimize operating system
      efficiency
```

or 2A1 through 2A4 (HP 32435A Series III) are reserved for the Lower Memory Module (Banks 0 - 7) and card cage slots 3A2 through 3A5 (HP 32421A Series III) or 2A7 through 2A10 (HP 32435A Series III) are reserved for the Upper Memory Module (Banks 8 - 15).

2-12. MULTIPLEXER CHANNEL. The Multiplexer Channels are designed to operate with moderate-speed I/O devices. Each Multiplexer Channel can handle up to 16 Device Controllers. The Multiplexer Channel, in conjunction with the IOP, allows its associated Device Controllers to run concurrently, interleaving their transfers to or from Main Memory on a word-by-word basis. The Multiplexer Channel resolves priority conflicts between its associated Device Controllers for access to the IOP, translates I/O program doubleword instructions into operating commands for its Device Controllers, and maintains the operating status of each Device Controller. Physically, the Multiplexer Channel consists of one PCA which is conventionally installed in Card Cage No. 4 or 5 (depending on the computer system model) as shown in tables 1-1 through 1-3. A detailed discussion of the Multiplexer Channels is contained in Section VII.

Table 2-2. Main Memory Configurations

Bank No.	System Word Capacity	PCA's Required			
		MCL	SMA	FLI	Total
2	128K	1	1	1	3
4	256K	1	2	1	4
6	384K	1	3	1	5
8	512K	1	4	1	6
12	768K	2	6	1	9
16	1024K	2	8	1	11

2-13. PORT CONTROLLER/SELECTOR CHANNEL. The Port Controller and Selector Channels are designed to operate with high-speed I/O devices. The Port Controller contains the MCU logic required to interface the Selector Channels with Main Memory via the CTL Bus and also resolves priority conflicts between Selector Channels for accessing the CTL Bus. (Although the Port Controller contains three selector channel ports, only two Selector Channels can be installed in the computer system at one time.) Physically, the Port Controller consists of one PCA and, as shown in tables 1-1 through 1-3, is conventionally installed in Card Cage No. 3. Detailed discussions of the Port Controller and Selector Channel are contained in Section VII.

Each Selector Channel can handle up to eight Device Controllers. Unlike the Multiplexer Channel which switches between Device Controllers on demand (based on hardware priority), the Selector Channel uses only one Device Controller at a time and that Device Controller monopolizes the channel until the device's I/O program is complete. Thus, only one I/O program is current at a given time for any one Selector Channel. Also, the Selector Channel directly accesses Main Memory for data and I/O program word transfers rather than indirectly as the Multiplexer Channel does through the IOP. Physically, each Selector Channel consists of three PCA's and, as shown in tables 1-1 through 1-3, are conventionally installed in Card Cage No. 3 and Card Cage No. 4 depending on the computer system model.

2-14. DEVICE CONTROLLERS. The computer system can handle up to 125 Device Controllers. Device Controllers provide the hardware I/O linkage between the computer system and external I/O devices. Primarily, a Device Controller translates programmed I/O commands (from a Multiplexer or Selector Channel) or direct I/O commands (from the IOP) into unique control signals required by its associated external I/O device(s). A Device Controller also generates the interrupts required by its associated I/O device(s) and the interrupts required by direct or programmed commands.

A Device Controller consists of one or more PCA's and, depending on the particular type of controller, can drive one or several external I/O devices. There are three types of Device Controllers; controllers used only for direct I/O, controllers used only for programmed I/O, and controllers used for both direct and programmed I/O. Regardless of the type, every Device Controller can accept all or some direct I/O instructions, can generate interrupts, and has a unique device number for addressing. Device Controllers can be installed in any of the available card cage slots designated in tables 1-1 through 1-3.

2-15. CTL Bus Priority

All computer system modules contain MCU logic that interfaces each module to the others via the CTL Bus. Each module gains access and control of the CTL Bus on a priority basis via its MCU logic. (The CTL Bus is only available to one module at a time.) For example; if two modules attempt to gain access to the CTL Bus simultaneously, the module with the higher priority will get the bus and the module with the lower priority will not get the bus until it is released by the higher-priority module. CTL Bus priority is resolved by assigning priority numbers to each system module with jumper switches located in each module's MCU logic. The system modules assigned the lowest priority numbers have the highest priority for accessing the CTL Bus.

Figure 2-2 illustrates the CTL Bus priority number assignments for each module in a typical computer system. It should be noted that the highest CTL Bus priorities (lowest priority numbers) are reserved for Main Memory and that the lowest CTL Bus priority (priority number 5) is reserved for the Central Processor Module. The lower memory module responds to both priority numbers 0 and 1. The upper memory module responds to both priority numbers 2 and 3. The required MCU logic for Main Memory is contained on the MCL PCA(s). (Priority for Memory Banks 0 through 7 is controlled by one MCL PCA and priority for Memory Banks 8 through 15 is controlled by a second MCL PCA.) If installed, the Selector Channel(s) has the next highest CTL Bus priority (priority number 4) after Main Memory. The required MCU logic for the Selector Channel(s) is contained on the Port Controller PCA. As previously discussed, the Central Processor's MCU resolves CTL Bus priority conflicts between the IOP and CPU. The IOP always has higher priority than the CPU. Therefore, the CPU always has a lower CTL Bus priority than any other module in the computer system.

2-16. OPERATING ENVIRONMENT

2-17. Virtual Memory

Virtual memory is a memory management scheme that uses semiconductor Main Memory and disc storage secondary memory. Due to a technique called memory segmentation, many programs stored in secondary memory can concurrently access the computer system and share its Main Memory. The system organizes programs into variable-length segments of code and data in secondary memory which

```
┌──────────────────────────────────────────────────────────────────────────────┐
│  ┌──────────────────────────────────────────────────────────────────────────┐ │
│  │                              CTL BUS                                      │ │
│  └──────────────────────────────────────────────────────────────────────────┘ │
│        │                          │                    │           │           │
│   ┌─────────────┐          ┌─────────────┐      ┌──────────┐  ┌──────────┐     │
│   │ PRIORITY    │          │ PRIORITY    │      │ PRIORITY │  │ PRIORITY │     │
│   │ NO. Ø AND 1 │          │ NO. 2 AND 3 │      │ NO. 5    │  │ NO. 4    │     │
│   │             │          │             │      │          │  │          │     │
│   │ ┌─────────┐ │          │ ┌─────────┐ │      │          │  │  PORT    │     │
│   │ │512 K WORDS│        │ │512 K WORDS│      │ CPU/IOP  │  │ CONTROL- │     │
│   │ ├─────────┤ │          │ ├─────────┤ │      │          │  │  LER     │     │
│   │ │LOWER MEMORY│        │ │UPPER MEMORY│     │          │  │          │     │
│   │ │ MODULE   │ │          │ │ MODULE   │ │      │          │  │          │     │
│   │ │BANKS Ø-7 │ │          │ │BANKS 8-15│ │      └──────────┘  └──────────┘     │
│   │ └─────────┘ │          │ └─────────┘ │                                     │
│   └─────────────┘          └─────────────┘                                     │
│        _____/                                   │
│                            │                                                   │
│                      MAIN MEMORY                                               │
└──────────────────────────────────────────────────────────────────────────────┘
```
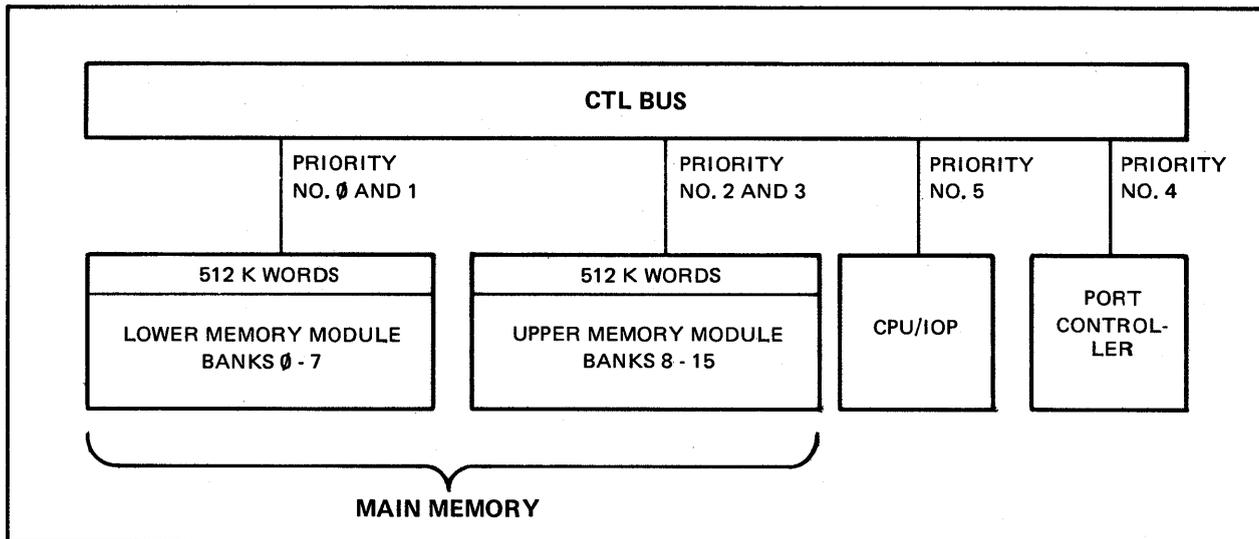
Figure 2-2.   CTL Bus Priority Number Assignments

can be transferred in and out of Main Memory on demand.   (Code
consists of executable instructions and unchanging constants of a
program or subprogram.   As the code is executed, the manipulated
values are referred to as data.) When a program is executed, only
those segments of code and data required at a particular time
actually reside in Main Memory and all other related segments
remain in secondary memory until they in turn are required.   When
a particular code segment is no longer needed, it is overlayed in
Main Memory by the next required code segment. (Code segments
are non-modifiable and reentrant.) If a code segment is needed
again, it is again copied from the secondary memory disc where it
resides.   Data segments, however, are dynamic and their contents
can be changed during the programs execution.   Therefore, when a
particular data segment is no longer needed, it is copied back
into the secondary memory disc and replaces the original data
segment version.   The vacated Main Memory space is then available
for other segments.   This process of transferring segments be-
tween secondary memory and Main Memory is referred to as swapping
and permits large programs that actually exceed Main Memory's
word capacity to be executed concurrently and still allow Main
Memory space for additional user programs.

2-18. Variable-Length Segmentation

Variable-length segmentation of code and data is used to facili-
tate multiprogramming.   It minimizes waste of memory resources
due to internal fragmentation and allows the operating system to
deal with logical rather than physical entities. This means that
a particular subprogram can be contained within one segment
rather than arbitrarily divided between two physical pages, thus
minimizing the amount of swapping that need be accomplished while
executing the subprogram. The location and size of all executing
code segments are maintained in a Code Segment Table and the lo-
cation and size of all associated data segments are maintained in

a Data Segment Table. These tables are known to both software and hardware. Software uses the tables for dynamic memory management by the operating system. Hardware uses the tables to perform references and transfers between segments and to make sure that all the segments required for current execution are present in Main Memory. (Refer to paragraph 2-24.) Code segments can be up to 16K words in length and data segments can be up to 32K words in length.

2-19. Processes

In an MPE environment, programs are run on the basis of processes created and handled by the operating system. A process is the basic executable entity in MPE. A process is not just a particular program; it is the unique execution of a particular program by a particular user at a particular time. When a user requests the execution of program, the system creates a private, hardware-protected data segment called a stack for that particular execution. Data segments separate from the stack can be obtained dynamically during process execution. Data segments can also be expanded and contracted by the operating system as required. This includes system handling of the stack overflow interrupt (paragraph 2-69) during which the data segment may automatically be expanded to accommodate operation of the stack. The program's changing set of code segments operating on the data stack constitute the process. (The code segments used by a particular process can be shared with other processes, but each individual process data stack is private.) In order for a process to execute, its data stack and code segment containing the procedure currently in execution by the CPU must be present in Main Memory.

2-20. Data Stacks

As previously discussed in paragraph 2-19, data for each user is organized into a data stack. In general, a stack is a storage area where the last item stored in is usually the first item taken out. In actual use, programs have direct access to all items in the stack by specifying addresses relative to several CPU registers. (Refer to paragraph 2-21.) All features of the stack, including the automatic transferring of data to and from CPU registers and checking for stack overflow and underflow, are implemented in the hardware. When programming in high-level languages such as COBOL or RPG, all stack manipulations are accomplished automatically by the language processor. The user can, however, manipulate the stack directly by writing programs in SPL. Figure 2-3 illustrates the general structure of a data stack as viewed from a subprogram. The white areas represent locations filled with valid data and the shaded area represents available unfilled locations. The stack area is delimited by the locations defined as Data Base (DB) and Stack Pointer (S-pointer). The DB and S-pointer addresses are retained in dedicated CPU registers. (Refer to paragraph 2-21.) The Q-minus relative addressing area contains the parameters passed by the calling program. The area between the S-pointer and Q contains the subprogram's local and temporary variables and intermediate results.
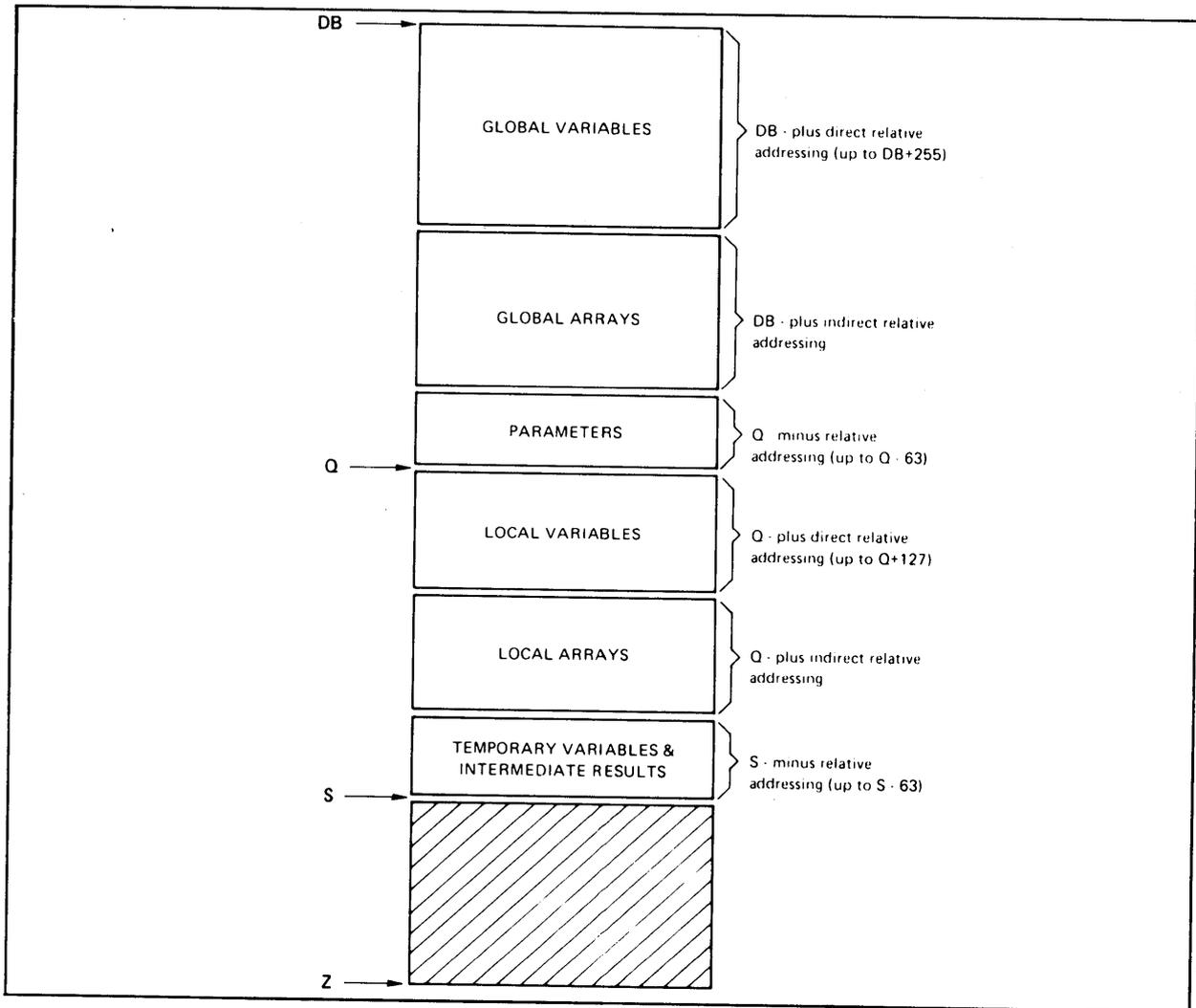
Figure 2-3.   Typical Data Stack

The data in the DB location is the oldest element on the stack.
The data in the S-pointer location is the most current element on
the stack. (The S-pointer location is referred to as the Top of
the Stack (TOS). Conventionally, TOS is represented downward
from DB to correspond to the normal progression of writing soft-
ware programs where the most recently written statement is fur-
ther down the page than previously written statements.   The area
from S+1 to Z is available for adding elements to the stack.
When a data word is added to the stack, it is stored in the next
available location and and the S-pointer is automatically incre-
mented by one to reflect the new TOS. (This process is said to
"push" a word onto the stack.) When data is deleted from the
stack, the S-pointer is decremented which puts the deleted word
in an undefined area.   S-minus relative addressing is used to
refer to recently stacked elements of data and is one of the
standard addressing conventions. Under this convention, S-1 is
the second element on the stack, S-2 is the third element on the
stack, etc.   The other standard addressing conventions are
DB-plus relative addressing, Q-minus relative addressing, and

Q-plus relative addressing. (Q separates the data of a calling program or subprogram from the data of a called subprogram.)

Since the four TOS elements are the most frequently used, there are four corresponding CPU registers (RA, RB, RC, and RD) that can at various times contain these four elements. The use of the four CPU registers increases stack operation execution speed by reducing the number of memory references needed when manipulating data at or near TOS. The four CPU registers are implicitly accessed by many of the machine instructions and whenever stack locations S, S-1, S-2, or S-3 are specifically referenced. (Refer to paragraphs 2-96 and 2-97.) During execution, data stacks are automatically expanded by the operating system up to a maximum of 32K words.

The system is also capable of operating in a split-stack mode. (Refer to paragraph 2-64.) In split-stack mode, the DB Register points to the current extra data segment and the other stack registers continue to point to the stack data segment. This is particularly efficient for system routines with tables in system data segments. In split-stack mode, these data segments can be accessed relative to the DB Register while using the other stack registers for computation. In addition to split-stack mode, the system contains instructions for moving data between data segments. These instructions cause an "absence trap" if either of the required data segments is not present in Main Memory. Therefore, the system can access very large address spaces outside of the stack and can provide buffering and other data storage facilities without having to reserve space for these functions within the stack data segment.

2-21. CPU Registers

The computer system contains 38 special purpose registers which perform the specific functions summarized in table 2-3. Since all addressing of code and data segments is accomplished relative to hardware address registers, the segments can be dynamically relocated in memory by simply changing the register base addresses. (The few instances where absolute addresses are required are privileged operations handled by the operating system.) Several of the hardware registers are used for defining the limits and operating elements of the code and data segments. As shown in figure 2-4, four of the CPU registers point to locations in a code segment and eight of the CPU registers point to locations in a data segment. It should be noted that there will normally be several segments in Main Memory at one time, but only one code segment and one data segment will be active at any given time. The CPU registers always point to the currently active segment. The functions of the CPU segment pointer registers are discussed in paragraphs 2-22 and 2-23. The remaining special purpose registers will be discussed later in this manual.

Table 2-3. Machine Registers

Register	Function	Register	Function
PB		SWCH	Switch Register
P	Code Segment		
PL	Pointers	PCLK	Process Clock
PB-Bank			Register
CIR	Current Instruction	SP0	
	Register	SP1	
		SP2	
NIR	Next Instruction	SP3	Scratch Pad, Flag,
	Register	CTR	and Interrupt
		ABS-Bank	Registers
DL		CPX1	
DB		CPX2	
Q		MOD	
SM	Data Segment		
SR	(Stack) Pointers	IOA	I/O Registers
Z		IOD	
DB-Bank			
S-Bank		ACOR	Memory Address and
		DCOR	Data Registers
RA		OPND	
RB	Top Of Stack (TOS)		
RC	Registers	RAR	Firmware Address
RD		SAVE	Registers
X	Index Register	STA	Status Register

2-22. CODE SEGMENT REGISTERS. The functions of the CPU code segment registers are as follows:

The PB Register defines the program base of the code segment being executed. The PB Register contains a 16-bit absolute address pointing to the first memory location of the code segment.

The PB-Bank Register is a 4-bit register used in conjunction with the PB Register to define in which memory bank the code segment resides.

The PL Register defines the program limit of the code segment being executed. The PL Register contains a 16-bit absolute address pointing to the last memory location of the code segment.

The P Register is the program counter. The P Register contains a 16-bit absolute address pointing to the memory location of the instruction being executed. The P Register can never point to a location beyond the limits defined by the PB and PL Registers.
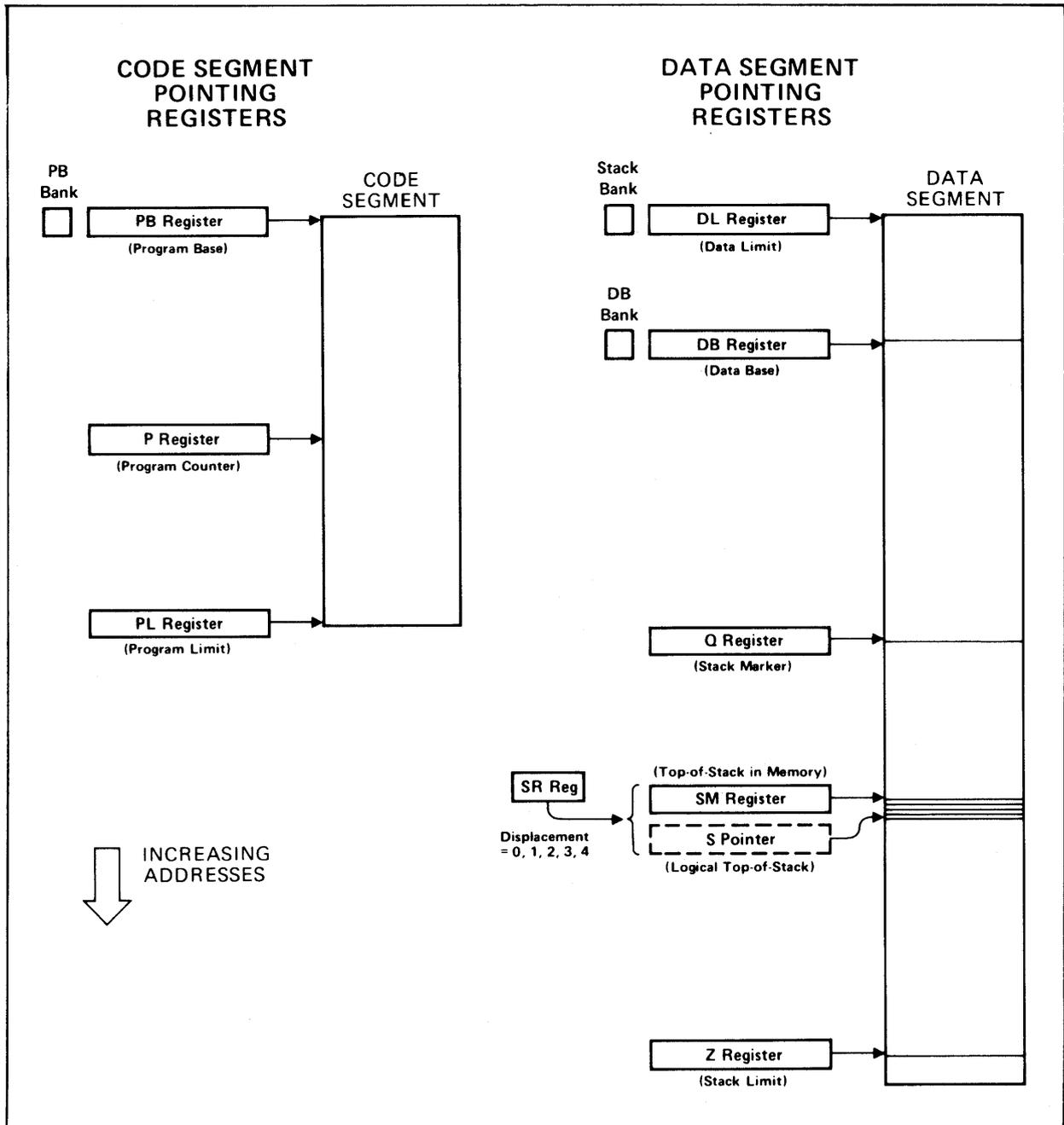
Figure 2-4. CPU Segment Pointer Registers

2-23. DATA SEGMENT REGISTERS. The functions of the CPU data
segment (stack) registers are as follows:

The DL Register defines the data limit of the current data seg-
ment. The DL Register contains a 16-bit absolute address point-
ing to the first word of memory available to the user's data
space.

The DB Register defines the data base of the current user's stack. The DB Register contains a 16-bit absolute address pointing to the first memory location of the directly addressable global area of the stack.

The DB-Bank Register is a 4-bit register used in conjunction with the DB Register to define in which memory bank the stack or split stacks (paragraph 2-64) reside.

The Q Register defines the current stack marker in the current data segment. The area of the stack between Q and S represents data that is incurred by the current procedure or routine. The Q Register contains a 16-bit absolute address pointing to the fourth word of the current stack marker being used within the stack. The location pointed to by the Q Register must be within the limits defined by the DB Register and Z Register. (During privileged mode (paragraph 247), Q can be moved below DB.

The SM Register defines the last memory location of the current stack. The SM Register contains a 16-bit absolute address pointing to the last accessed data location in memory. It should be noted that the contents of the SM Register may not necessarily point to the actual (or logical) TOS. The location pointed to by the SM Register must be within the limits defined by the DB Register and Z Register.

The SR Register defines the number of TOS elements that are in the CPU stack registers. The SR Register contains a 3-bit number that has a value from 0 and 4. This number is a positive displacement which, when added to the address contained in the SM Register, indicates the logical TOS. (The contents of the SM Register plus the contents of the SR Register always defines the S-pointer.)

The S-pointer is not a physical register, but is logically composed by adding together the contents of the SM Register and SR Register. The S-pointer always defines the logical TOS. (The principle of using two physical registers to create the S-pointer is employed for hardware convenience in achieving fast execution times.) The following relationship exists between the S-pointer and the CPU stack registers:

$$
\begin{aligned}
RA &= \text{S-pointer} = \text{SR Register} + \text{SM Register} \\
RB &= \text{S-pointer} - 1 \\
RC &= \text{S-pointer} - 2 \\
RD &= \text{S-pointer} - 3
\end{aligned}
$$

The Z Register defines the stack limit of the current user's stack. The Z Register contains a 16-bit absolute address pointing to the last memory location available to the stack. (Actually, each data segment has several locations beyond Z that are used for bounds checks (paragraph 2-65) and stack markers due to an interrupt (paragraph 2-28).

The S-Bank Register is a 4-bit register used in conjunction with the S-pointer and DL, Q, and Z Registers to define which memory the S-Bank is not necessarily equal to the DB-Bank.

2-24. Basic Table Structures

The first few locations of Main Memory are reserved for the system pointers listed in table 2-4. During system cold load, memory location 0 is set to point to the location of the Code Segment Table (CST) as shown in (1), figure 2-5. The CST contains a single four-word entry for each Segmented Library segment currently in use in the system. (Segmented Libraries permit separate programs to share procedures.) Memory location 1 (2), figure 2-5 points to the Code Segment Table Extension (CSTX) area allocated to the program being executed by the CPU. The CSTX is used to keep track of the code segments in the various program files being executed. Therefore, the contents of memory location 1 will shift to point to various sections of the CSTX as different programs are executed by the CPU. For example, figure 2-5 shows that Program X is currently being executed by user process A. Also during system cold load, memory locations 2 and 3 are set to point to the Data Segment Table (DST) and Process Control Block (PCB) Base respectively. See (3) and (4), figure 2-5. There is a four-word DST entry for each data segment in use in the system as discussed in paragraph 2-26. There is a PCB allocated to each process running in the system. The PCB entry for a process points to the DST entry for its stack data segment although, for simplicity, this is not shown in figure 2-5. Memory location 4 is set by the software to point to the PCB of the currently executing process (5), figure 2-5. The linkage from the PCB to the CSTX area (6) is used to set memory location 1. It should be noted that if process B and process C happen to be executing the same program (7), the program file segments will be shared. The CPU Status Register (STA Register) then points to the current segment of the current process holding CPU resources.

2-25. CODE SEGMENT TABLE AND CODE SEGMENT TABLE EXTENSION. The CST contains a list of code segments that are being referenced by executing programs. Its length is determined at system generation time. The actual number of entries in use at any time is variable, limited only by the length of the table. Entries are dynamically allocated by the operating system as programs are loaded and unloaded. Each entry contains control information about the segment and gives its length and starting address in the format shown in figure 2-6. The first %300 entries are reserved for Segmented Library segments. The CST entry for segment 0 contains control information. Segment 1 contains the routines needed to service internal interrupts. Segments 2 through 191 (%277) contain code such as service routines for external interrupts, system intrinsics, and library procedures. The remainder of the CST entries fall in the CSTX area and keep track of program segments. Each program can have up to 63 segments. The table is accessed via the PCAL, EXIT, IXIT, and DISP instructions (Section IV) and is completely invisible to the user.

Table 2-4. Reserved Low Main Memory Locations

Memory Location	Contents
0	Code Segment Table Base
1	Code Segment Table Extension
2	Data Segment Table Base
3	Process Control Block Base
4	Current Process Control Block
5	Interrupt Stack Base
6	Interrupt Stack Limit
7	Interrupt Mask
%10-%13	Reserved
%14-%777(max.)	Device Reference and External Interrupt Table
%1000	System Global Table (Pointers to resident tables, etc.)

Note: The % symbol preceding a number indicates an octal value.

2-26. DATA SEGMENT TABLE. The DST contains a list of the various data segments currently in use by the operating system and user programs. These segments include I/O buffers, system and user process stacks, and extra data segments. The DST length is determined at system generation time and it contains a four-word entry for each data segment in the format shown in figure 2-7. The actual number of entries in use at any one time is variable, limited only by the length of the table. Entries are dynamically allocated by the operating system as programs are loaded and unloaded or as special capability processes request or release additional data segments.

2-27. Code Segment Linkage

During the execution of one user process, there will usually be several code segments in memory and a single data segment. Assume that the current process presently has two code segments in memory as shown in figure 2-8. The purpose of figure 2-8 is to illustrate how the system keeps track of where code segments are and to show how references can be made from one segment to another. Although figure 2-8 illustrates hardware, it is the responsibility of the MPE operating system to control the tables shown.

The CST Pointer is permanently resident in Location 0 and it contains an absolute address pointing to the starting location of the CST (1), figure 2-8. The CST tells where each code segment (present or absent) is located. If the segment is a program segment, Location 1 is used. Each entry in the CST has a unique number (code segment number) that identifies the particular seg-

Figure 2-5. Basic Data Structures

CODE SEGMENT TABLE Doubleword

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	M	R	T						L/4						
Reserved															
Reserved													B		
Address															

A Absence Bit = 1 if segment is absent from main memory.

M Mode Bit = 1 if segment executes in Privileged Mode (Code only).

R Reference Bit = 1 if segment has been referenced (set by microcode).

T Trace Bit = 1 if trace feature is used. Checked by PCAL instruction.

L Length Field = segment length divided by 4.

B Bank Address. Points to memory bank (if resident in main memory) in which segment resides.

ADDRESS   Absolute address of PB within B if the segment is present, otherwise the 3rd and 4th words contain the absolute disc address.

SEGMENT TRANSFER TABLE Words

STT Length

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

U Uncallable bit

LENGTH Maximum = 255 (Calls from external segments may reference only the first **127** entries. PL-1 thru PL-127.)(PL-0= STTL.)

Local Program Label

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

U Uncallable bit

ADDRESS PB relative, + only

External Program Label

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

STT # STT entry number in target segment, maximum = 127

SEG # Target segment

STATUS Word

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
M	I	T	R	O	C	CC		SEGMENT #							

M Mode bit (=1 for privileged mode)

I Interrupt enable (1)/disable(0), external

T Traps enable(1)/disable(0), user

R Right Stack Opcode bit (pending = 1)

O Overflow bit

C Carry bit

CC Condition Code

SEGMENT # currently executing

90020-7

Figure 2-6. Formats Associated With Code Segments

```
       0 ' 1 ' 2 ' 3 ' 4 ' 5 ' 6 ' 7 ' 8 ' 9 '10 '11 '12 '13 '14 '15
      ┌───┬───┬───┬───────────────────────────────────────────────┐
      │ A │ C │ R │                    L/4                        │
      ├───┴───┴───┴───────────────────────────────────────────────┤
      │                       Reserved                            │
      ├───────────────────────────────────────────────┬───────────┤
      │                    Reserved                    │     B     │
      ├───────────────────────────────────────────────┴───────────┤
      │                       Address                             │
      └───────────────────────────────────────────────────────────┘
```

A = Absence Bit = 1 if segment is absent from main memory.

C = Clean bit. Used to eliminate unnecessary output swapping.

R = Reference Bit = 1 if segment has been referenced (set by microcode).

L = Length Field = segment length divided by 4.

B = Bank Address. Points to memory bank (if resident in main memory) in which segment resides.

ADDRESS = Absolute Segment Address within given bank in third word of segment. If segment is
         absent, words 3 and 4 contain absolute disc address.

90020-6

Figure 2-7. Data Segment Table Entry Format

ment.   Each entry consists of a four-word descriptor which in-
cludes the absolute address of the related segment and its
length.   Entry number 0 in the table is unique in that it simply
points to the final entry in the table (2). This defines the
length of the table for the benefit of the operating system in
allocating space for the table itself. Segment number 0 does not
exist.   Assume that one user is executing a process which re-
quires code segments 22 through 25. Also assume that segments 22
and 23 are in Main Memory and that segments 24 and 25 are not
presently needed and, therefore, are on disc. The process is
presently executing instructions in code segment 23. This means
that the address value contained in the fourth word of CST entry
23 has been loaded into the PB Register. Therefore, the PB
Register is pointing to PB(a) as shown in (3), figure 2-8. The
PL Register, using a value derived from the segment length, is
pointing to PL(a). The P Register is advancing from PB(a) toward
PL(a).

The last nine locations of segment 23 are not part of the seg-
ment's code, but are added by the operating system when the seg-
ment is loaded into virtual memory. These locations contain
linking references for every procedure call (PCAL) in the Segment
Transfer Table (STT). A PCAL is an instruction that references a
set of instructions elsewhere in the code segment. This set of
instructions is structured as a procedure to perform a standard-
ized operation or computation and then return control to the
instruction immediately following the call instruction. It
should be noted that entries in the STT are numbered from the end
back towards the code. Entry number 0 gives the STT length as
shown in figure 2-6. This indicates the number of the last STT
entry (4), figure 2-8 , so that the hardware can make validity

2-19
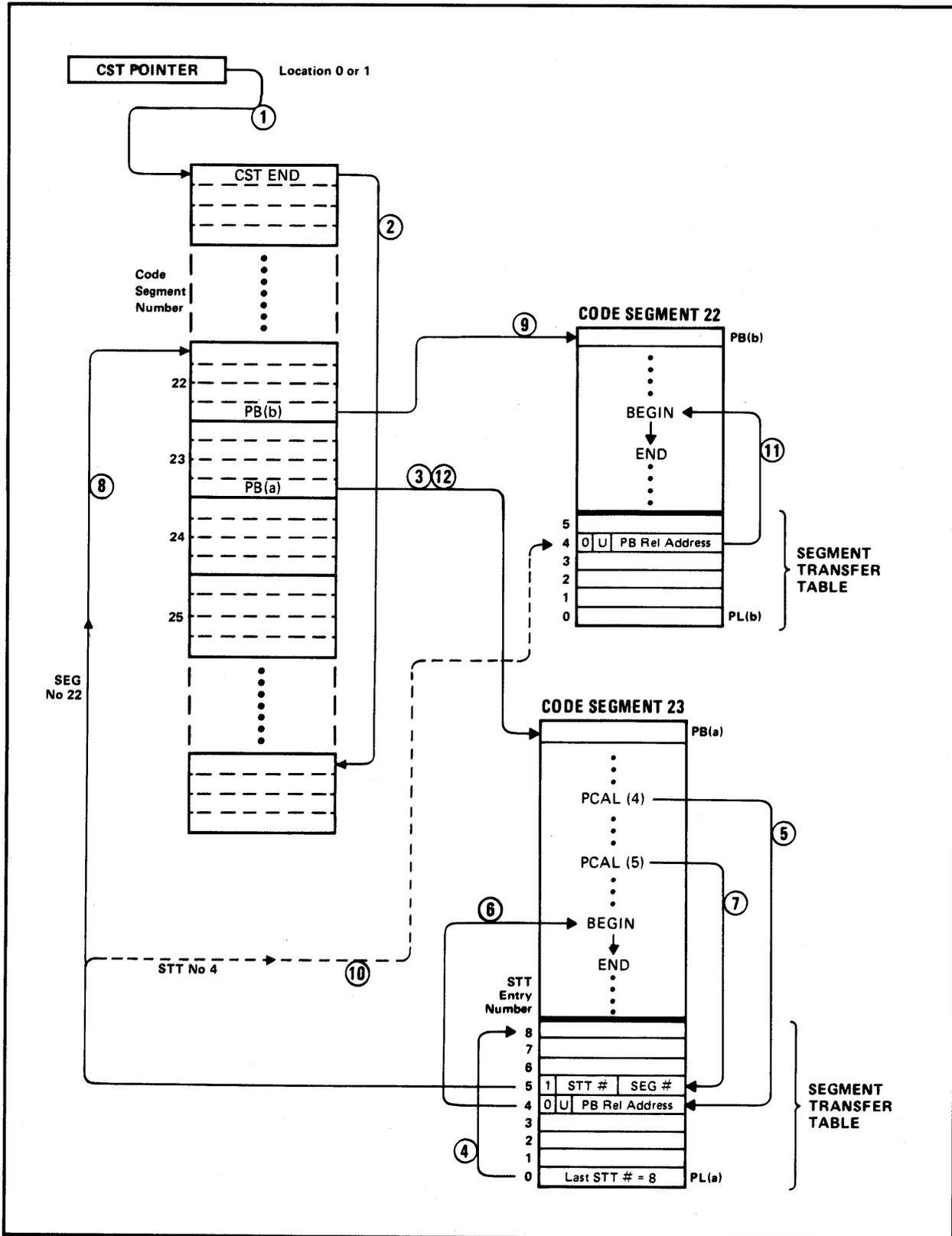
Figure 2-8. Code Segment Linkage

checks on PCAL references.   For example, a call to entry to num-
ber 9 would be invalid. (If a call from within the segment is
made to entry 0, the reference will be taken from the TOS instead
of from the STT.   A call from outside a segment to entry 0 starts
execution at P = PB after checking the U bit.) When the execution
sequence reaches the first PCAL instruction, a reference (5) is
made to the fourth entry of the STT.   (Since the PCAL instruction
uses PL-addressing, the instruction references cell PL -4.) This
location contains a local program label (figure 2-6) which im-
plies that the called procedure is located within the same seg-
ment. The reference is a PB-relative address pointing to the
beginning of a procedure or block (6), figure 2-8.   After some
preparatory operations, which include saving the return address
on the stack, the PCAL instruction transfers control to the pro-
cedure.   When an EXIT instruction is encountered in the proced-
ure, control is returned to the instruction immediately following
the first PCAL. In this example there were no references outside
the current segment. In the following example, an external ref-
erence is made.

When the execution sequence reaches the second PCAL, another call
(7), figure 2-8 is made to the STT.   The call requests the fifth
entry in the table which happens to be an external program label
(indicated by a logical 1 in bit 0). This implies that the called
procedure is in some other segment.   The label contents tell
which segment and also give the STT number in that segment which
must contain the local reference. The PCAL instruction, after
the usual preparatory operations (which include bringing the seg-
ment into Main Memory if it is absent), transfers control to the
called procedure as follows.   The segment number given in the
external program label (8) points to a specific entry in the CST,
assumed to be entry number 22. A value for PB is picked up in
the fourth word of this entry and loaded into the PB Register.
This causes the PB Register to point (9) to the starting location
of code segment 22; PB(b).   The limit, PL(b), is also establish-
ed. Meanwhile, the STT value given in the external program label
is pointing to entry number 4 (10) of the STT.   This causes a
PB-relative address to be picked up for the P Register. The P
Register now points to the starting address of the procedure or
block (11) and execution begins. If an STT number of 0 was giv-
en, execution would begin at PB(b). Calling procedures outside
the segment in this manner is subject to a number of rules,
checks, and safeguards to ensure that the call is allowable and
that other users are fully protected from invasions of privacy.
The manner in which the operating system sets up the STT ensures
that all transfers are legal for that process. At the conclusion
of the called procedure, control is returned to the original seg-
ment by the EXIT instruction. This instruction restores the STA
Register which gives the caller's segment number, and the PB Reg-
ister value (12) returns back to PB(a).   The saved P-relative
address on the stack reestablishes the return point and execution
continues at the location immediately following the second PCAL
instruction.

2-28. Stack Operation

Note

When the letters P, Q, DB, etc., are
used alone in the following paragraphs,
the letter is interpreted to mean "the
location pointed to by the P Register,
Q Register, DB Register, etc."

Figure 2-9 illustrates the basic construction of the stack area
and how the CPU stack registers delimit the various areas. It
should be noted that there will normally be several stacks in
memory (one for each process), but only one stack will be active
at a given time. The CPU stack registers always point to the
currently active stack.

As shown in figure 2-9, the stack area is bounded at the low end
by the DL Register and bounded at the high end by the Z Register.
The DB Register points to the base location of the stack and di-
vides the stack area into two major parts. (The area between DB
and Z is the actual user stack. The area between DB and DL is
not part of the stack itself, but is closely associated with the
stack. This area provides a dynamic area for such applications
as dynamic arrays, symbol tables, etc.) The SM Register points to
the current top-of-stack (TOS) location in memory.

Whereas the contents of the DB Register and Z Register are stat-
ic, the content of the SM Register is constantly changing as the
program progresses, moving up and down the stack area. The area
between DB and SM is always filled with valid data and the area
between SM and Z is always available for additional data. (If the
quantity of data should exceed the available space, the attempt
to move SM past Z will cause an interrupt to the operating system
which may then grant additional space (new Z value) one or more
times.

Unlike the cell-at-a-time movement of SM, Q moves sporadically in
jumps. The purpose of the Q Register is to retain the starting
point of data relating to the current procedure. Therefore, when
a new procedure begins, Q jumps ahead to establish a new starting
point at the current TOS. Conversely, when a procedure ends, Q
jumps back to the place it had marked previously for the preced-
ing procedure. As far as the current procedure is concerned,
stack data consists of the locations from a base of Q to the
current TOS.

In the previous discussion, the SM Register was assumed to point
at the absolute TOS. This is true only for the portion of the
stack in memory. Actually, as many as four of the top words of
the stack can spill over into TOS registers RA, RB, RC, and RD.
Figure 2-10 illustrates where three of the topmost words are in
TOS registers RA, RB, and RC. It should be noted that the SM
Register points to the last stack element in memory, but that

```
CPU                              MEMORY
Stack Registers              Stack and Array Area

DL Register  ───────────►

                                              Own
                                              Array
                                              Area

DB Register  ───────────►

             Base
             of
             Stack
                                              Filled

Q Register   ───────────►

SM Register  ───────────►

             Top
             of
             Stack
             Increasing
             Addresses
                                              Avail-
                                              able

Z Register   ───────────►

90020-35
```
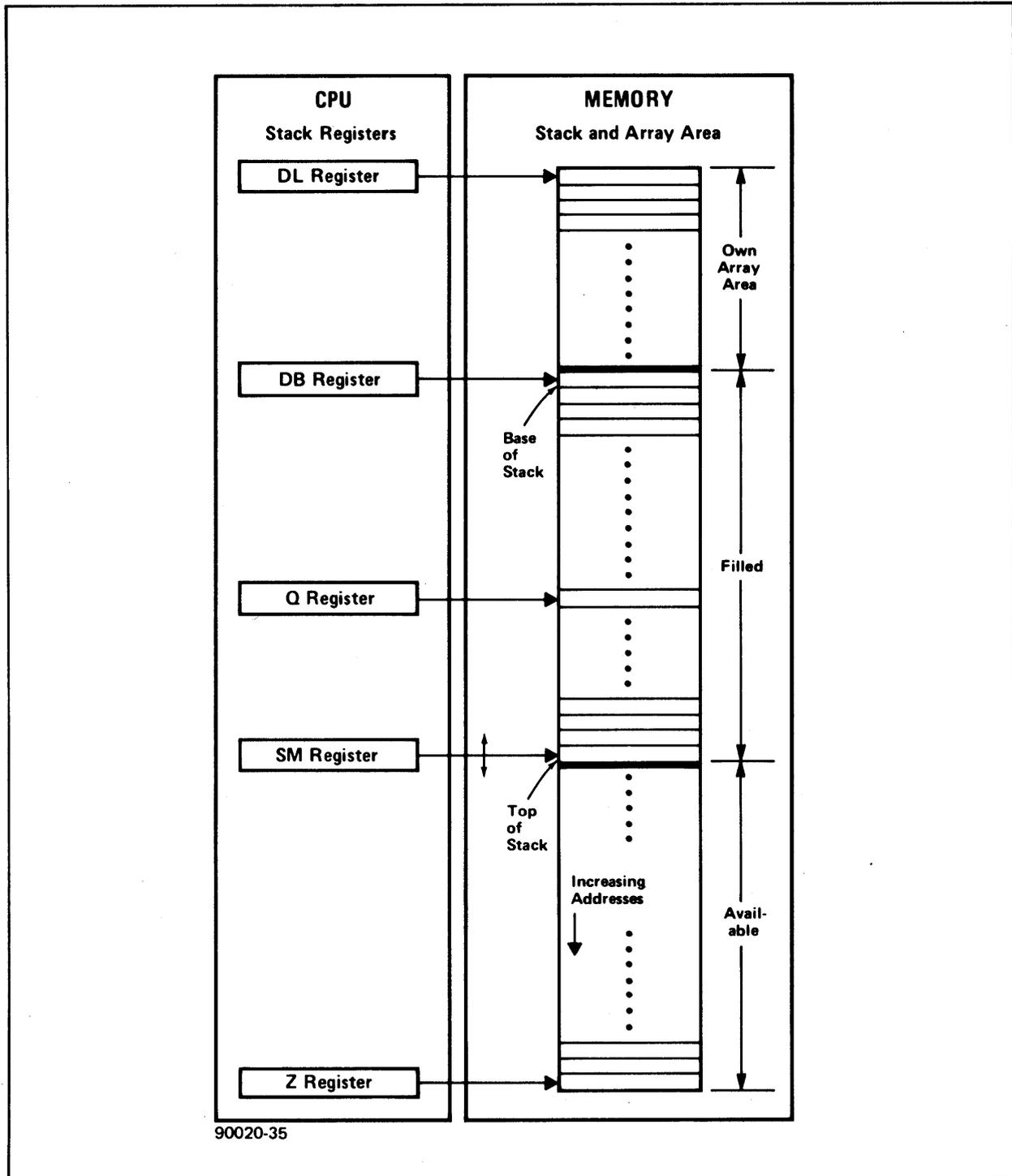
Figure 2-9.   CPU Registers and Stack Basic Operations

the logical TOS is in the third CPU register and is defined by the S-pointer (S). The four TOS registers are reserved for the four topmost words of the stack and are employed only by CPU hardware. The TOS registers cannot be addressed externally. Externally, the programmer is interested only in location S and and the hardware defines this address for him. Using figure 2-10
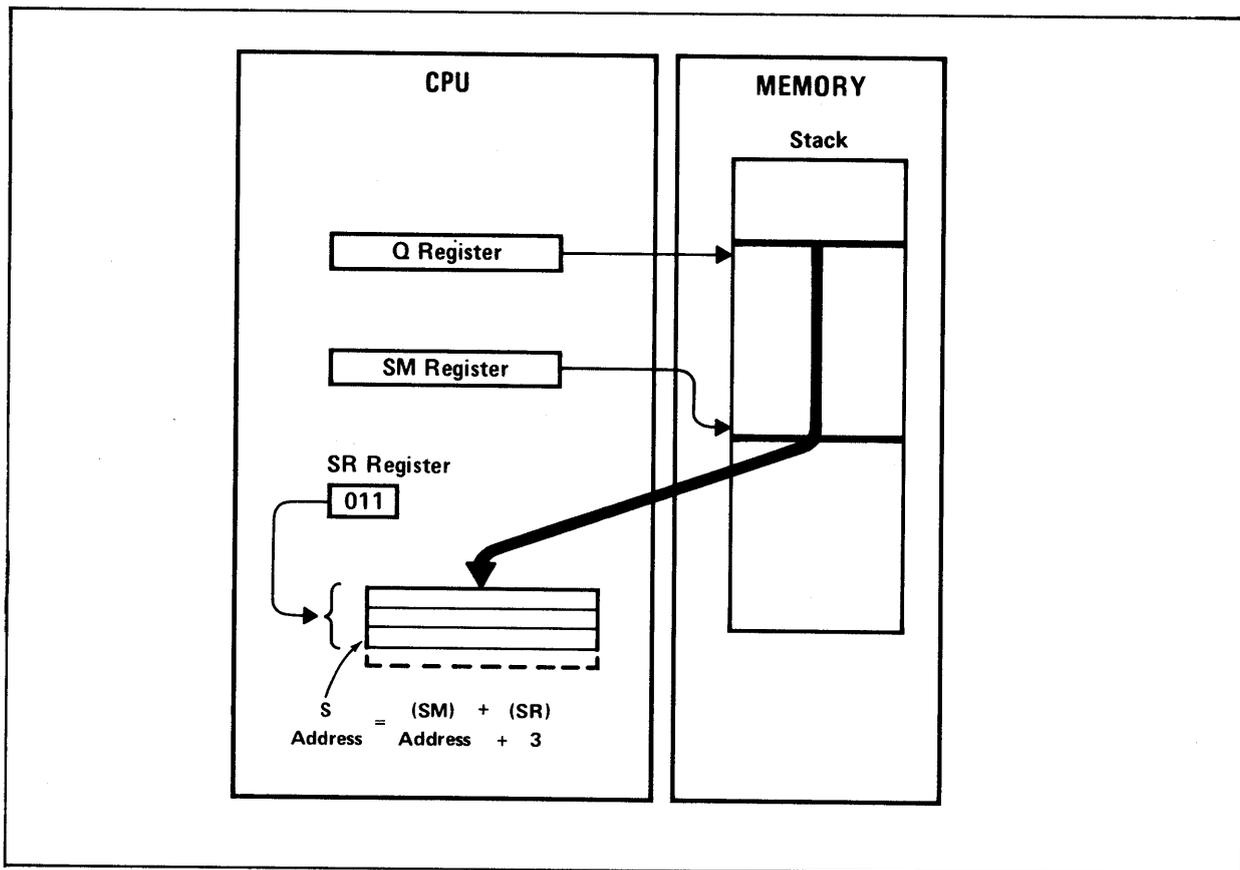
Figure 2-10.   CPU TOS Registers

as an example, the hardware will define address S as being equal to the SM Register value plus three. The value three is obtained from the SR Register which, as previously discussed, retains the number of TOS elements that are in the TOS registers. (S = SM + SR; RA = S, RB = S - 1, etc.) The address value S obtained by adding the SR Register contents to the SM Register contents is a completely valid address. In fact, when the CPU must be cleared for some other operation (e.g., a new procedure or an interrupt), the register contents are physically transferred to the numerically corresponding memory locations. In this example, SM would move up by three and the SR Register contents would become zero.

Figure 2-11 illustrates the actions of the Q Register in marking the starting location for each procedure's data. Figure 2-11 shows that the currently executing code segment was working with data in the temporary storage area immediately following the First Q area. At that time, the Q Register was pointing at First Q, S was defining TOS, and the Z Register was pointing to the end of the data segment. (If the executing code segment never called another procedure, the stack would never get more complicated.) As illustrated however, the code called a procedure at some point by means of a Procedure Call (PCAL) instruction that caused additions to the stack as indicated by Procedure A. New data was incurred as the procedure began and S pointed to the top of that data as it was generated. Then, Procedure A called Procedure B
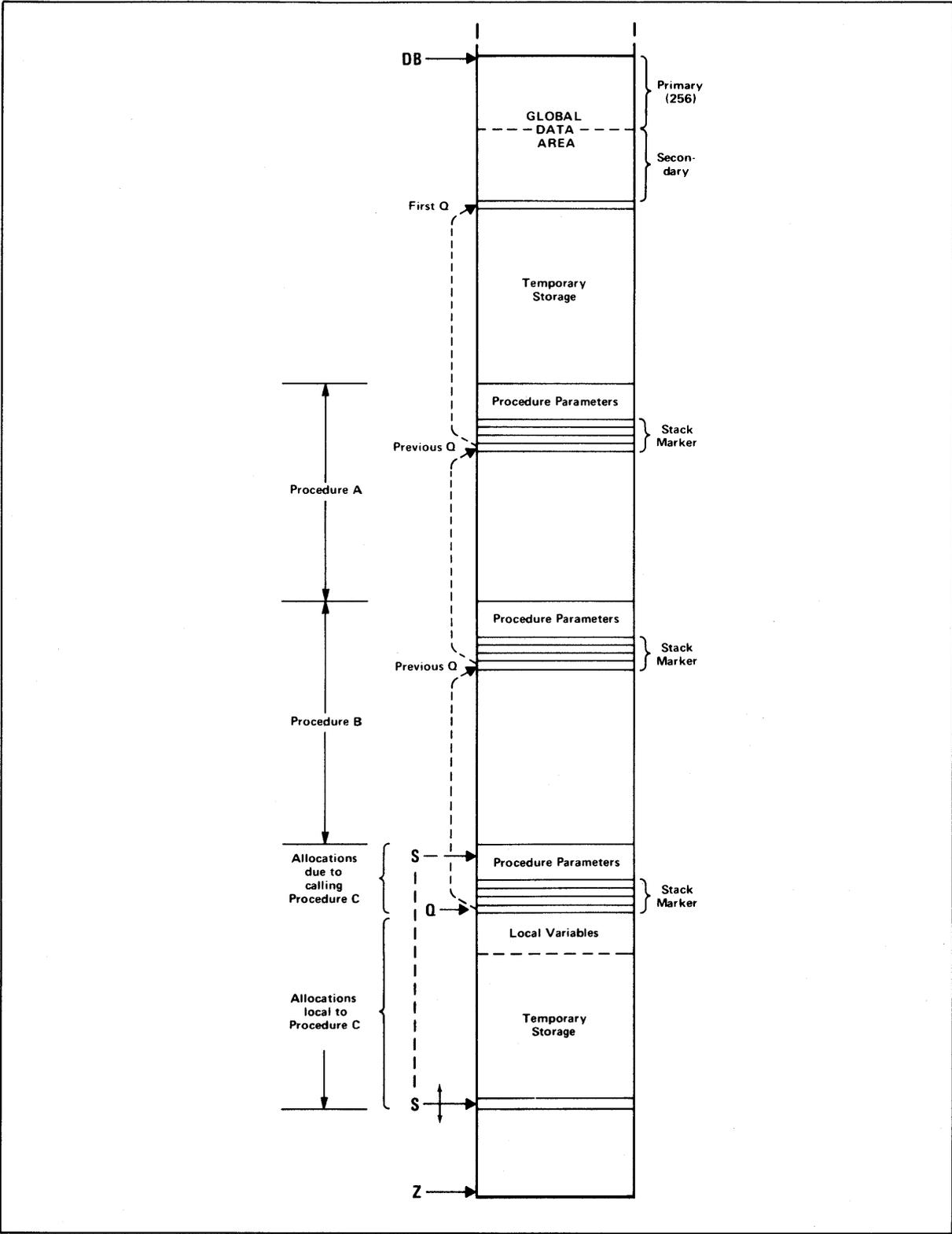
Figure 2-11. Stack Mark Chain

and caused new additions to the stack as indicated. Next, Procedure B called Procedure C and caused the final stack picture as shown.

As the program progresses, Procedure C will end and, after saving its answer in a convenient place for Procedure B to access, issue an EXIT instruction. Then, all the other stack additions due to Procedure C will be eliminated by moving S and Q back and Procedure B will continue its computations on its own data stack. In the same manner, Procedure B will end, save its data, and exit. This eliminates the data stack for Procedure B. Finally, Procedure A will exit and return the net answer to the new TOS on the main temporary storage area.

Each time control is returned from the called procedure to the caller's procedure (within the code segment), the stack registers also return to the caller's data area. Thus, the stack marker chain virtually eliminates system overhead in keeping track of nested procedures. For example, the simple return sequence previously discussed (C-to-B-to-A-to main program) is not imperative. Procedure C could have been called again before the return to the main program was complete or other procedures could have been called. Regardless, the return for both code and data will always remain perfectly in step; from the called to the caller.

Note that the area between DB and First Q in figure 2-11 is the global data area. The locations in this area are reserved by the process for variables which the process has declared to be global for all procedures called by that process. That is, any procedure using this particular data segment may reference the variables in this area. The individual locations in the global area can contain an actual value or an indirect address pointing to some other location that either contains the value or is the start of an array. Since DB-relative addressing (paragraph 2-49) is restricted to a maximum of DB+255, only the first 256 locations of this area can be addressed directly. These areas are called the primary global data area. If the number of entries exceed 256, indirect addressing (paragraph 2-50) must be used. These locations are called the secondary global data area.

When the operating system has completed assigning space for the global variables, it points Q at the next succeeding location (First Q, figure 2-11). This is the actual start of the data stack. Since there is not data on the stack, S also points to this location. As the executing code segment proceeds to obtain, manipulate, and generate data for the stack, S moves away from Q, always indicating the top of such data. At some time during the execution of the code segment, it is assumed that Procedure A is called. Usually, a set of procedure parameters accompany the call and these parameters are placed on the stack just prior to the issuance of the PCAL instruction. These are actual parameters to be substituted for formal parameters in the procedure and are referenced by Q addressing. (Refer to paragraph 2-49.)

Calling the procedure causes a four-word stack marker to be placed on the stack as shown in figure 2-11. The marker format is shown in figure 2-12. The first word saves the current contents of the Index Register (X Register). The second word saves the return address for the code segment (P Register address plus one relative to the PB Register contents). The third word saves the STA Register contents (M, I, T, R, O, C, and CC) and the code segment number of the caller in case the called procedure is external to the current code segment. The fourth word contains a value called Delta Q which designates how far back it is to the previous location to which the Q Register was pointing. In this case, Delta Q is pointing to First Q. The Q Register now points at this Delta Q location.

The previously described sequence of events are repeated when Procedure B (figure 2-11) and Procedure C are called. Each time, the Q Register will point to the Delta Q location of the current stack marker and the contents of that location will point to the previous Q setting. Therefore, when Procedure C is executing, there is a chain of Delta Q stack marks linking the present Q setting back to the First Q.

The links are used and eliminated as the procedures are exited the same as they were established when the procedures were called. When Procedure C ends, the EXIT instruction returns S to equal Q, essentially placing the Delta Q value temporarily on the TOS. This allows the EXIT instruction to compute a new value for the Q Register (Previous Q) and it appropriately moves Q back. The EXIT instruction causes S to decrement step-by-step through the stack marker, restoring status, P Register contents, and X Register contents for Procedure B.

Lastly, S is moved back to eliminate the unwanted parameters of Procedure C. Presumably, one or more parameters will be answers computed by procedure C and, therefore, S is only moved back so far as to preserve the desired answers which are now on the TOS. The sequence of events discribed in the last two paragraphs is repeated until all stack marks are eliminated and only the final answer is on the TOS. For additional information on stack operations, refer to paragraph 4-17.

2-29. INSTRUCTION AND STATUS WORD FORMATS

2-30. Instruction Formats

The machine instruction set is designed for maximum efficiency of bit usage in the instruction words and, therefore, the instruction formats do not necessarily fall into rigid field boundries. There are 23 distinct instruction set formats. In addition to the instruction formats, there are 13 instruction groups as shown in figure 2-13. The formats of the individual instruction groups are discussed in paragraphs 2-31 through 2-44. For additional information, refer to Section IV.

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

X Register Contents

PB Relative Return Address for P Reg

M	I	T	R	O	C	CC	Code Segment #

Delta Q
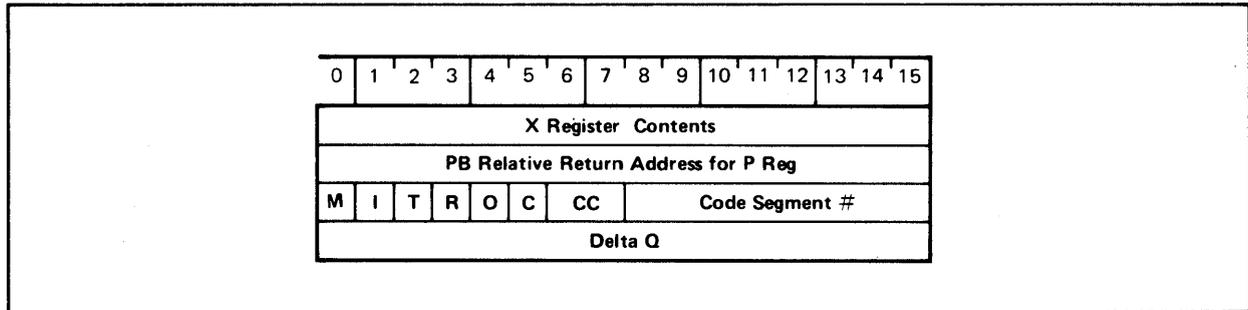
Figure 2-12. Standard Stack Marker Format

2-31. GENERAL FORMAT. The first format shown in figure 2-13 is the general scheme for dividing the instruction word into code fields. Only the first field is rigidly adhered to. This field (bits 0 through 3) defines either a specific instruction code in the memory address group (or loop control group) or one of the sub-opcode groups. There are four sub-opcode groups; 1, 2, 3, and stack ops. The field for the sub-opcodes varies. For sub-opcode groups 2 and 3, bits 4, 5, 6, and 7 are used as shown. For subopcode group 1, bits 5 through 9 are used and, for stack ops, the remainder of the word is used. In some cases, the sub-opcode will enable a third field (mini-opcode or special opcode) in bits 8 through 11. The remainder of the word has a variety of special uses and commonly is part of an argument field.

2-32. STACK OP. The stack op format is defined by four 0's in the first four bits. The remaining 12 bits are divided into two fields; stack op A and stack op B. Either or both of these fields may contain any of the 63 stack op instruction codes. Execution sequence is from left to right (A first, then B). Interrupts may occur between the execution of A and B. It should also be noted that indicators (Carry, Overflow, and Condition Code) are set by the last executed stack op. If using only one of the two stack op fields, it is more efficient to use stack op A since the hardware always looks ahead to see if stack op B is a NOP. This permits the hardware to ignore the second field which results in saving time.

2-33. SHIFT. The shift instructions use about half of the sub-opcode group 1 codes. Sub-opcode group 1 is defined by 0001 in the first four bits. If the index bit (bit 4) is 1, the contents of the Index register (X Register) is added to the shift count in bits 10 through 15 to specify the number of places each data bit is shifted. Bits 5 through 9 encode the specific shift instruction.

2-34. BRANCH. The branch instructions use 11 of the sub-opcode group 1 codes. Bit 4 is used as an indirect bit (indirect if bit 4 is 1 and direct if bit 4 is 0). Bits 5 through 9 encode the specific branch instruction. Bits 11 through 15 give a P-relative displacement from 0 through 31 and bit 10 specifies whether the displacement is + or - relative to P (0 = +, 1 = -).
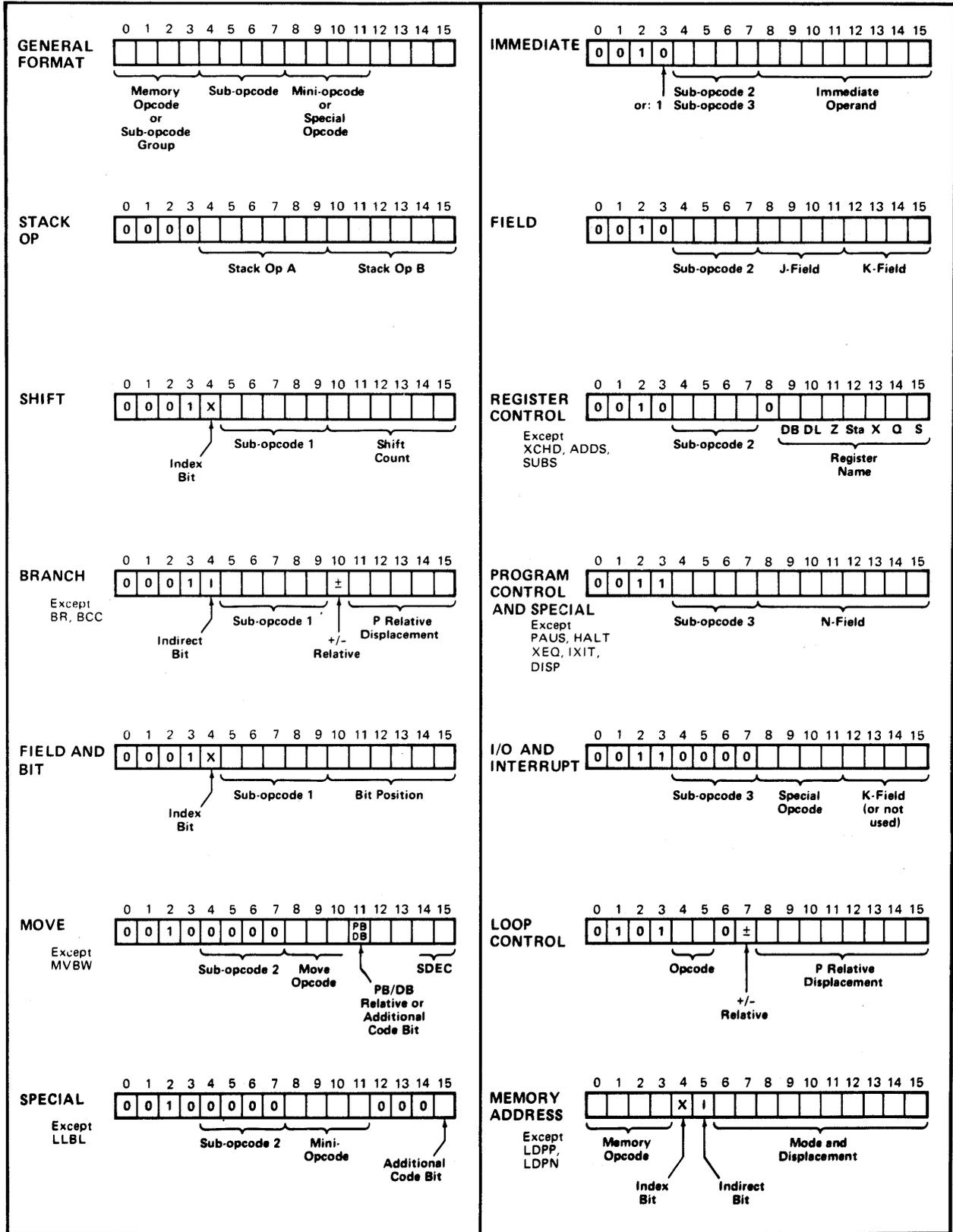
Figure 2-13.   Instruction Groups

2-35. BIT TEST. The bit test instructions are also in subop-
code group 1 and use bits 5 through 9 to encode the specific in-
struction. Bits 10 through 15 specify a bit position in the TOS
word for testing. If the index bit (bit 4) is 1, the contents of
the X Register is added to the specified bit position.

2-36. MOVE. The move instructions use 12 of the codes specified
by the sub-opcode group 2 code 0000. Sub-opcode group 2 is de-
fined by 0010 in the first four bits. Bits 8 through 10 encode
the specific move instruction. Bit 11 is used by some instruc-
tions to specify whether the source of the moved data is PB-rela-
tive (bit 11 = 0) or DB-relative (bit 11 = 1). In some cases,
bit 11 is also used as an additional code bit for specifying the
instruction. Bits 12 and 13 are not used. Bits 14 and 15 are
used to specify an S-decrement value to delete, if desired, the
move parameters from the TOS.

2-37. SPECIAL. The special instructions use four mini-opcodes.
The mini-opcode group is also specified by the sub-opcode group 2
code 0000. Bits 8 through 11 plus bit 15 encode the specific
special instruction. Bits 12 through 14 are not used.

2-38. IMMEDIATE. The immediate instructions use codes in both
sub-opcode group 2 (coded 0010) and sub-opcode group 3 (coded
0011). Bits 4 through 7 encode the specific immediate instruc-
tion. Bits 8 through 15 are used for the immediate operand.

2-39. FIELD. The field deposit and extract instructions are
specified by two of the sub-opcode group 2 codes. Bits 4 through
7 encode the specific field instruction. Bits 8 through 15 are
divided into a J-field and a K-field. The J-field specifies the
starting bit number and the K-field specifies the number of bits.

2-40. REGISTER CONTROL. The register control instructions use
bits 9 through 15 to name a register. Bits 4 through 7 encode
the specific register control instruction.

2-41. PROGRAM CONTROL. The program control instructions use four
of the sub-opcode group 3 codes. Sub-opcode group 3 is specified
by 0011 in the first four bits. Bits 4 through 7 encode the spe-
cific program control instruction. The N-field (bits 8 through
15) is used for either a PL-displacement (PCAL and SCAL) or to
specify a number of parameters to be deleted on return from a
procedure or subroutine (EXIT or SXIT).

2-42. I/O AND INTERRUPT. The I/O and interrupt instructions use
11 of the special opcodes (bits 8 through 11) defined by the
subopcode group 3 code 0000. The K-field (bits 12 through 15) is
used by some of the instructions for an S-displacement to locate
a device number given in the stack.

2-43. LOOP CONTROL. The loop control instructions are defined by
a special coding of bits 4, 5, and 6 for memory opcode 05 (which
is otherwise defined as the STOR instruction). Bits 8 through 15
give a P-relative displacement for a branch address and bit 7

specifies whether the displacement is + (bit 7 = 0) or - (bit 7 = 1) relative to P.

2-44. MEMORY ADDRESS. Bits 0 through 3 encode the specific memory address. Bits 6 through 15 give both an addressing mode and a displacement. (Refer to paragraph 2-48.) Bit 5 is used to specify direct or indirect addressing (1 = indirect, 0 = direct). Bit 4 is used to specify indexing (1 = indexing), if desired. If both indirect addressing and indexing are specified, post-indexing (paragraph 2-54) will occur.

2-45. Status Word Format

There is a status word for each code segment in the system. At all times, the status word associated with a given process indicates the machine status following the execution of the most recent instruction in that segment. The status for the currently executing segment is resident in the STA Register and is constantly being updated as each instruction is executed. For segments that are not current (suspended by either an interrupt or procedure call), the status word exists in a stack marker in a data stack as discussed in paragraph 2-28. As shown in figure 2-6, status word bits 8 through 15 indicate the segment number of the currently executing code segment (when the particular status word is resident in the STA Register). Therefore, when a status word is pushed into a stack marker by an interrupt or procedure call, bits 8 through 15 identify the segment that is to be returned to when execution is later resumed. The following descriptions of the status bits assume that the status word under discussion is resident in the STA Register. All references to "current" conditions can also be inferred as "then current" conditions in the case of suspended segments or procedures.

Bit 0 is used to indicate whether the current segment is running in privileged mode (bit 0 = 1) or user mode (bit 0 = 0). (Refer to paragraph 2-47.) The state of this bit cannot be changed by machine instructions while resident in the STA Register except in privileged mode. (The PCAL, IXIT, and EXIT instructions include checks to prevent illegal mode changes by altering the noncurrent status mode bits.)

Bit 1 is used to enable or disable external interrupts. This bit cannot be changed in user mode while current and the EXIT instruction invokes a trap if a non-privileged user illegally alters the bit while non-current. The state of bit 1 can only be changed in privileged mode.

Bit 2 is used to enable or disable user traps. The state of this bit can be changed in any mode while current or non-current with a SETR instruction. (The state of this bit is not affected by the EXIT instruction.)

Bit 3 is normally used only by the hardware which sets this bit to 1 if the right stack opcode (bits 10 through 15) contains a valid instruction other than NOP. The hardware requires this information in case an interrupt occurs between the execution of the left and right stack ops. The state of bit 3 cannot be changed in user mode while current.

Bit 4 is the overflow bit and is one of the three indicators which are set or cleared as an incidental operation by many of the machine instructions. In general, bit 4 is used only by signed integer and floating-point computations. If bit 4 is set (1), it indicates that the result of the computation is too large to be represented in the number of available bits in the data format. (For floating point, it can also indicate that the result is too small.) If user traps are enabled (bit 2 = 1), an interrupt to segment 1 will occur in lieu of setting bit 4; except for integer overflow which causes both bit 4 to be set and an interrupt to segment 1. This permits the system to generate a message to the user which indicates which type of overflow or underflow occurred. All user traps will set bit 4 if traps are disabled.

Bit 5 is the carry bit and is one of the three indicators which are set or cleared as an incidental operation by many of the machine instructions. Bit 5 is used primarily by logical and integer arithmetic and usually indicates a carry (bit 5 =1) or lack of carry (bit 5 = 1) out of the most significant bit during a computation. Bit 5 is also used by some instructions as an indicator for special purposes which are stated in the individual machine instruction definitions. (Refer to Section IV.)

Bits 6 and 7 are used to encode the condition codes discussed in paragraph 2-46 and are one of the three indicators which are set or cleared as an incidental operation by many of the machine instructions.

2-46. Condition Codes

Although several instructions make special use of the condition code bits (status word bits 6 and 7), the condition code typically indicates the state of an operand or a comparison result with two operands. The operand can be a word, byte, double word, or triple word and can be located on the TOS, in the X Register, or in a specified memory location. Three codings are used; 00, 01, and 10. (Code 11 is not used.) Except for special interpretations, there are four basic patterns for interpreting the codes. The four patterns (CCA, CCB, CCC, and CCD) are summarized in table 2-5 and discussed in the following paragraphs.

Table 2-5. Condition Codes

```
| CCA sets CC = CCG (00) if operand > 0              |
|              CCL (01) if operand < 0              |
|              CCE (10) if operand = 0              |
|---------------------------------------------------|
| CCB sets CC = CCG (00) if numerical (%060 - 071)  |
|              CCL (01) if special character (all   |
|                       other octal values)         |
|              CCE (10) if alphabetic (uppercase,   |
|                       %101 - 132; lowercase, %141 -|
|                       172)                        |
|---------------------------------------------------|
| CCC sets CC = CCG (00) if operand 1 > 2           |
|              CCL (01) if operand 1 < 2           |
|              CCE (10) if operand 1 = 2           |
|---------------------------------------------------|
| CCD sets CC = CCG (00) if device is not ready     |
|              CCL (01) if non-responding Device    |
|                       Controller                  |
|              CCE (10) if responding Device        |
|                       Controller and/or external  |
|                       device is ready             |
|---------------------------------------------------|
| Notes: CC  = Condition Code                       |
|        CCG = Condition Code Greater               |
|        CCL = Condition Code Less                  |
|        CCE = Condition Code Equal                 |
```

The most common condition code pattern is pattern A (CCA). In CCA, the condition code is set to 00 if the operand is greater than zero; to 01 if the operand is less than zero; and to 10 if the operand is exactly zero.

Condition code pattern B (CCB) is used with byte oriented instructions. In CCB, the condition code is set to 00 if the operand byte is an ASCII numerical character which would be represented by octal values 060 through 071. The code is set to 10 if the byte is an ASCII alphabetic character which would be represented by octal values 101 through 132 for upper-case letters and 141 through 172 for lower-case letters. The code is set to 01 if the byte is an ASCII special character which would be represented by the remaining octal values.

Condition code pattern C (CCC) is used with comparison instructions. In CCC, the condition code is set to 00 if operand 1 is greater than operand 2; to 01 if operand 1 is less than operand 2; and to 10 if operand 1 is equal to operand 2.

Condition code pattern D (CCD) is used with some (not all) I/O instructions. In CCD, the condition code is set to 00 if the external device is not ready. (This condition is usually caused by the device being busy.) This code is only used with instructions that will (WIO and RIO) or could (SIO) require data to be moved. The code is set to 01 if the Device Controller does not respond. (This condition can be caused by loss of power in the external device or Device Controller, a malfunction in the external device or Device Controller or, more normally, the external device or Device Controller waiting for a response to an interrupt request.) The condition code is set to 10 if the external device or Device Controller has responded normally and the instruction has been completed properly.

2-47. OPERATING MODES

The computer system can be operated in either privileged or user mode and has the capability of switching from one mode to the other depending on the type of operation being executed at a given instant. The operating mode currently in effect is indicated at all times by the STA Register's bit 0 as discussed in paragraph 2-45.

Privileged mode operation is characterized by the ability to execute privileged instructions and to call segments that have been declared uncallable. (The method of declaring a code segment uncallable involves the use of the uncallable bit (bit 1) in the associated STT local program label shown in figure 2-6.) Privileged operations such as I/O operations are performed by the operating system operating in privileged mode. For an non-privileged user to perform such operations, it is necessary to call one of the callable intrinsics of the operating system which, in turn, will call the uncallable intrinsics that will perform the operation on behalf of the nonprivileged user. However, a privileged mode user can use the computer as if he were the operating system.

```
CAUTION
```

The normal checks and limitations that apply to the standard non-privileged users in MPE are bypassed during privileged mode. It is possible for a privileged mode program to destroy file integrity including the MPE operating system software. Hewlett-Packard cannot be responsible for system integrity when programs written by users operate in the privileged mode.

2-48. ADDRESSING CONVENTIONS

2-49. Memory Addressing

Throughout this manual the terms "displacement", "effective address", "relative address", and "base" are used in connection with memory addressing. These terms are defined as follows:

Displacement is a positive number given in the instruction word pointing to a location plus or minus that number of locations from a given reference cell which is also given in the instruction word.

Effective address is always an absolute address. It may or may not be the location indicated by the displacement number. The effective address is the final computed address after displacement calculation, indirect addressing (if any), and indexing (if any) have all been resolved.

Relative address is the address obtained by subtracting the base from the effective address.

Base is either the program base (PB) address or the data base (DB) address.

The computer system uses relative addressing almost exclusively. Addressing can be relative to the location pointed to by the P Register, DB Register, Q Register, or S-pointer. As shown in figure 2-14, memory address instructions (paragraph 2-44) use bits 6 through 15 for "mode and displacement" and addressing can be + or - relative to P or Q, but only + relative to DB and - relative to S. The relative addressing displacement ranges for the various modes are also shown in figure 2-14. (It should be noted that these ranges apply only to direct, unindexed addressing. Indirect addressing and indexing are discussed separately in paragraphs 2-50 and 2-53.) The variety of displacement ranges is due to the particular coding required to specify a given mode. For example, only two bits (6 and 7) are required to specify the P+, P-, and DB+ relative modes. This leaves bits 8 through 15 available for displacement which, therefore, can be any value from 0 through 255. For Q+ relative mode, bits 9 through 15 give a displacement range from 0 through 127. For Q- and S-relative modes, bits 10 through 15 give a displacement range from 0 through 63. In order to provide the most efficient usage of bits, the mode codes are assigned according to the respective needs of each displacement range.

Referring to figure 2-14, note that the DB+, Q-, Q+, and S-ranges can overlap. Also, DB+, Q+, and S- can actually address words currently held in the TOS registers. P+ and P- addressing modes are typically used for branches and referencing of literals. The DB+ mode is used for referencing global variables and pointers (i.e., indirect addresses). The Q+ and Q- modes are useful for local variable storage and passing of procedure parameters respectively. The S- mode is typically used for accessing param-
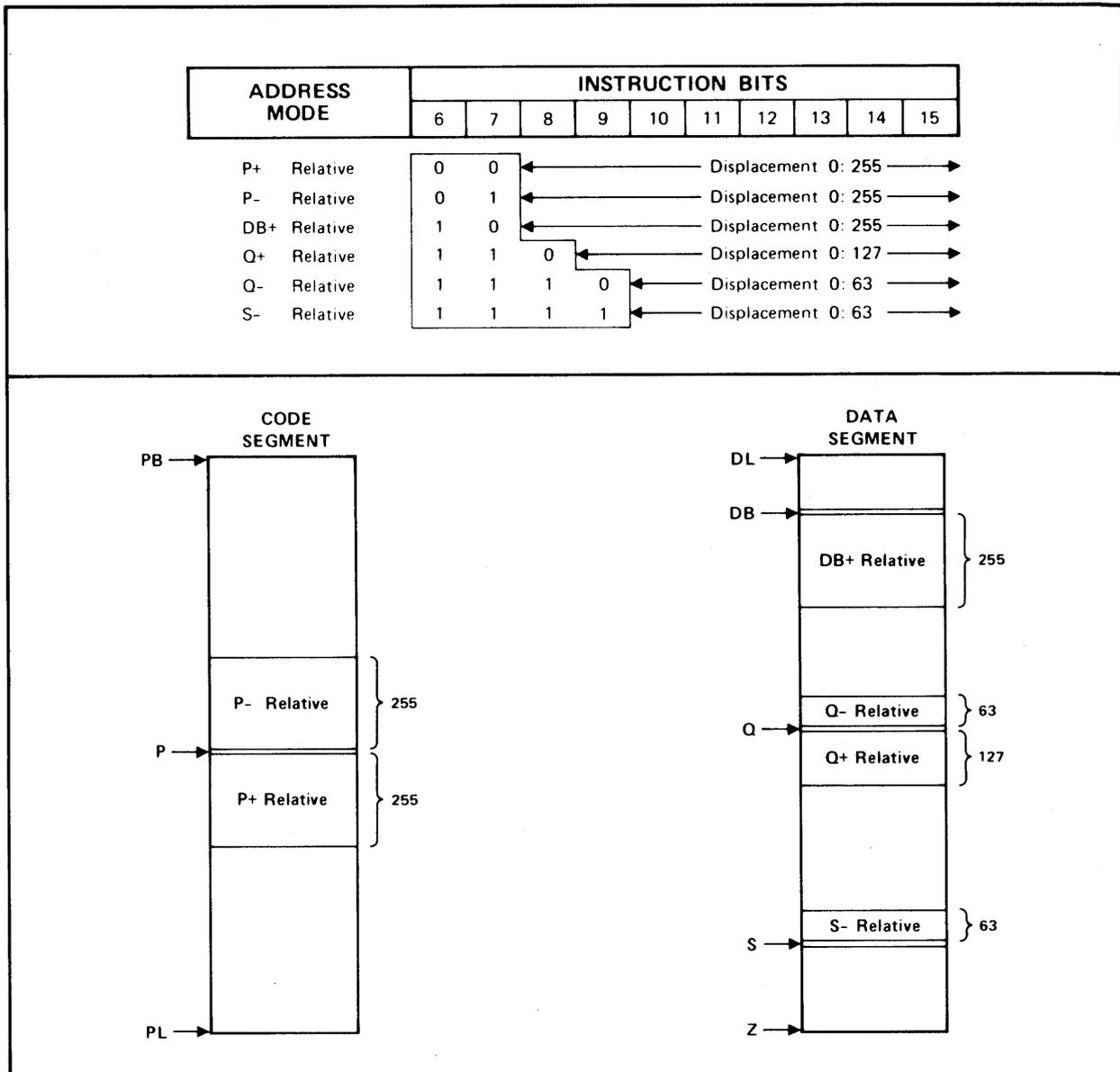
Figure 2-14. Memory Addressing Modes

eters in subroutines. Not all memory address instructions are capable of using all six relative modes. Each instruction defi- nition (Section IV) will specify which modes are applicable for each instruction.

2-50. Indirect Addressing

As shown in figure 2-15, indirect addressing uses the location referenced by the initial displacement (the Indirect Cell) to specify another location within the same code or data segment. For code references, the Indirect Cell contains a self-relative address. For data references, the Indirect Cell contains a DB+ relative address. For memory address instructions (paragraph 2-44), indirect addressing is specified by bit 5 of the instruc-
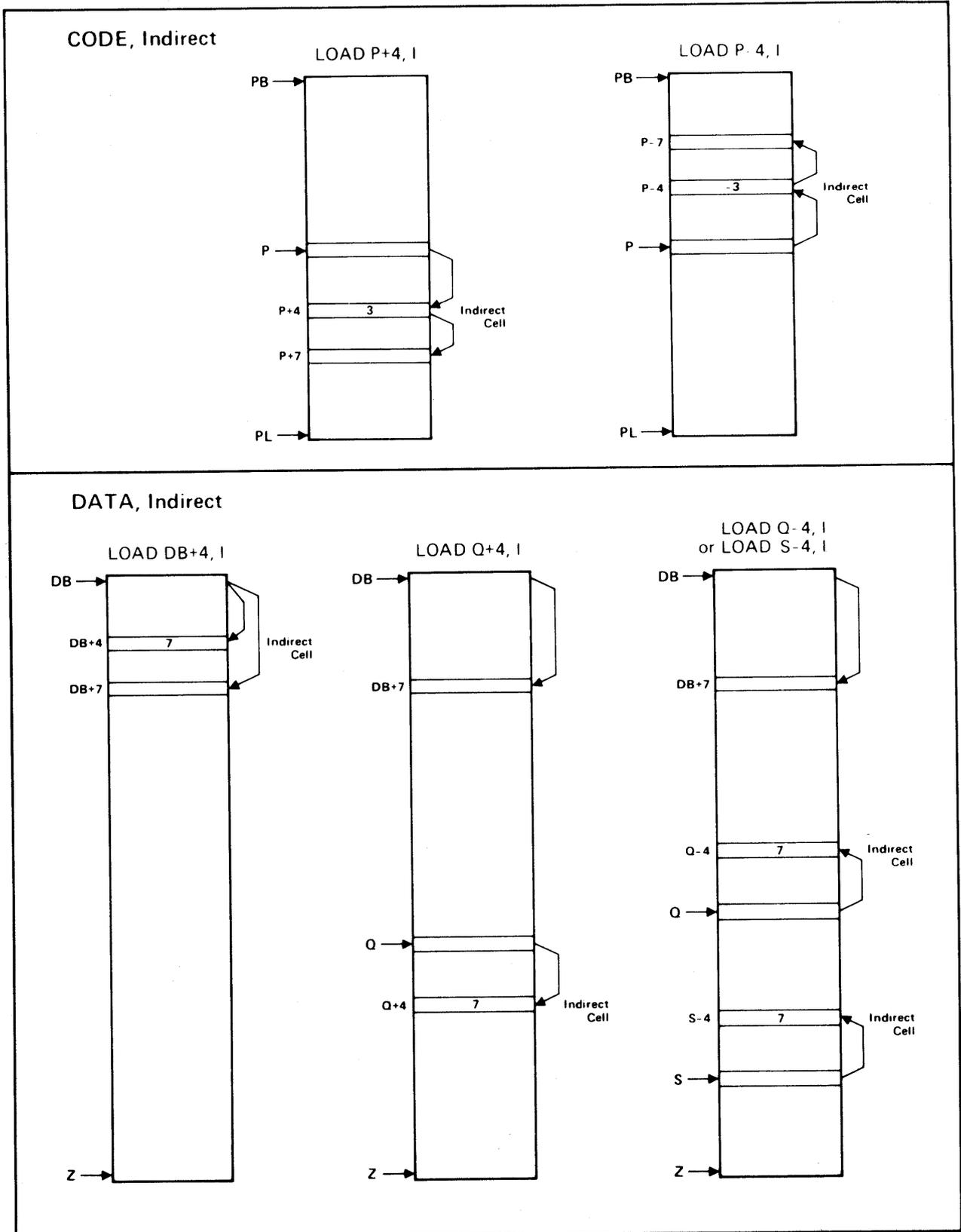
Figure 2-15.   Indirect Addressing Examples

tion word. (A logic 1 specifies indirect addressing.) For most branch instructions (paragraph 2-34), indirect addressing is specified by bit 4.

2-51. CODE INDIRECT. Both P+ and P- examples of indirect addressing in a code segment are shown in figure 2-15. The first example shows the actions occuring for an assumed "LOAD P+4, I" instruction. The displacement (+4) points to the Indirect Cell at P+4. The Indirect Cell contains a +3 self-relative address. This points to a location three addresses higher, or P+7. It is the contents of P+7 that will be loaded onto the TOS by the "LOAD P+4, I" instruction. The second example shows the actions occuring for an assumed "LOAD P-4, I" instruction. The displacement (-4) points to the Indirect Cell at P-4. The Indirect Cell contains a -3 self-relative address. This points to location P-7 which is the effective address for th "LOAD P-4, I" instruction.

2-52. DATA INDIRECT. The first example in figure 2-15 of indirect addressing in a data segment shows the actions occuring for an assumed "LOAD DB+4, I" instruction. The displacement (+4) points to the Indirect Cell at DB+4. The Indirect Cell contains a DB+7 relative address. This is not a self-relative address and, therefore, the effective address is at location DB+7. It should be noted that it is possible for the effective address to be below as well as above the Indirect Cell. The second example shows the actions occuring for an assumed "LOAD Q+4, I" instruction. The displacement (+4) points to the Indirect Cell at Q+4. The Indirect Cell contains a DB+7 relative address and, therefore, the effective address is again at location DB+7. The third example shows the actions occuring for an assumed "LOAD S-4, I" or "LOAD Q-4, I" instruction. The displacement (-4) points to the Indirect Cell at either S-4 or Q-4 depending on the instruction and, since the contents of the Indirect Cell is assumed to be+7, the effective address for both instructions is again DB+7.

2-53. Indexing

The memory address instructions (paragraph 2-44) use indexing to modify an operand address. Indexing is specified by bit 4 of the instruction word. (A logical 1 specifies indexing.) Figure 2-16 shows examples of indexing when combined with positive and negative addressing modes (both direct) and an example of indirect, indexed addressing (positive mode only) for both code and data segments. It should be noted that in each example, the index is assumed to be 5. This is established by the "LDXI5" instruction that preceds each LOAD instruction used in the examples. This instruction loads the value 5 in the Index Register (X Register).

2-54. CODE INDEXING. The first example in figure 2-16 shows the actions occuring for an assumed "LOAD P+4, X" instruction. The displacement (+4) would by itself point to location P+4. However, by adding the index of 5 to the displacement, the location P+11 (octal) is addressed. It is the contents of this location that will be loaded onto the TOS by the "LOAD P+4, X" instruction. The second example shows the actions occuring for an
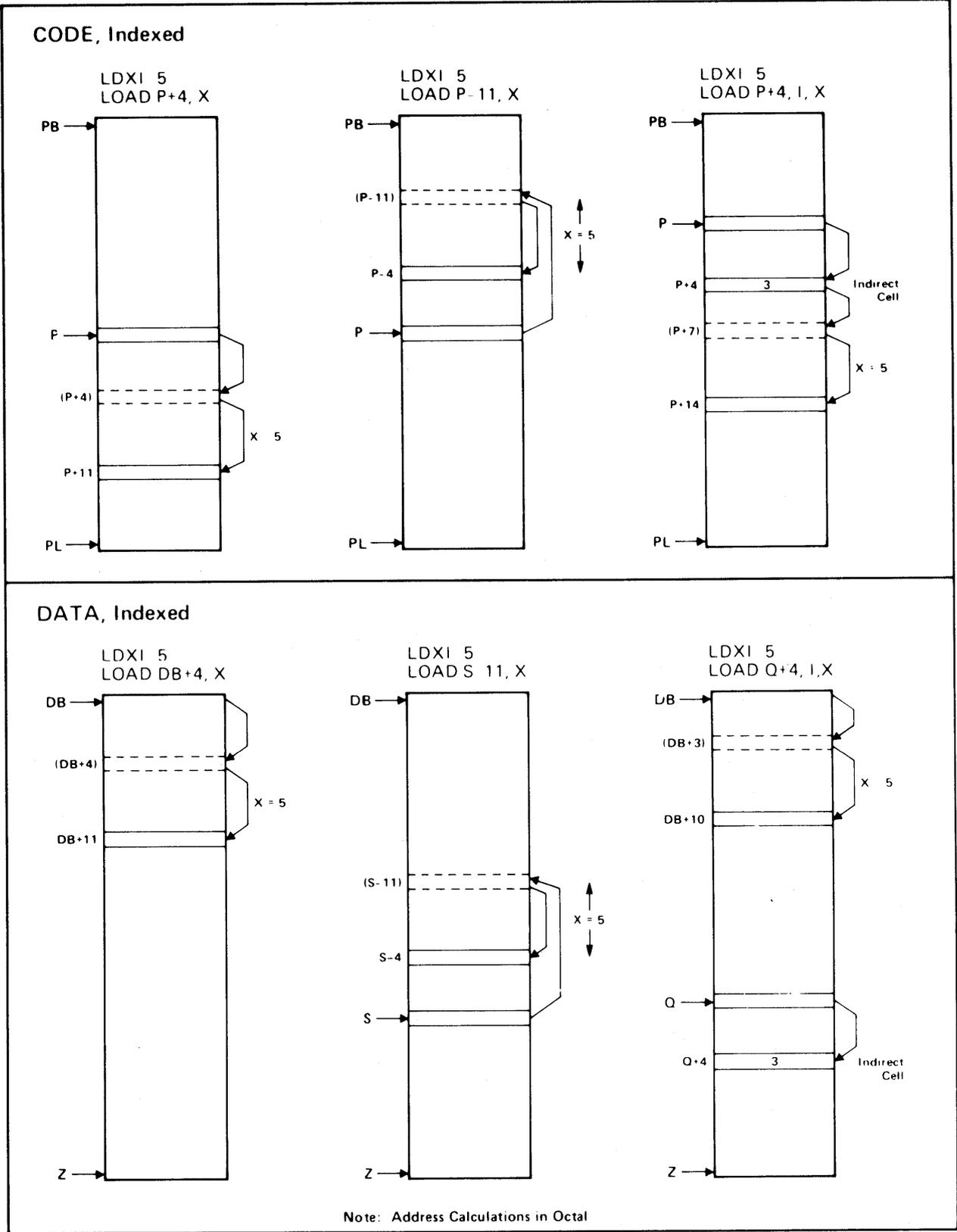
Figure 2-16. Indexing Examples

assumed "LOAD P-11, X" instruction. The displacement (-11) is added to the positive index of 5 and the final address is P-4. The third example shows code indexing combined with indirect addressing. In all cases, post-indexing is used; i.e., the indirect addressing is accomplished first (either positive or negative direction), and then indexing proceeds in a positive or negative direction from the indicated location. As shown in the example for the "LOAD P+4, I, X" instruction, the displacement of +4 points to the Indirect Cell at P+4. The contents of P+4 is a self-relative address of 3 that points to P+7. However, indexing adds 5 to this value and the final effective address becomes P+14 (octal).

2-55. DATA INDEXING. The first example in figure 2-16 shows the actions occuring for an assumed "LOAD DB+4, X" instruction. The displacement (+4) points at DB+4 which is then modified by the index of 5 to point at DB+11. The second example shows the actions occuring for an asumed "LOAD S-11, X" instruction which is similar to the actions occuring for the "LOAD P-11, X" instruction discussed in paragraph 2-54. Since a positive index is specified, indexing proceeds in a positive direction from the location indicated by the displacement. The third example shows data indexing combined with indirect addressing. Again, post-indexing is used. The displacement (+4) points to the Indirect Cell at Q+4 which contains the value 3. Since indirect addresses for data are always DB+ relative, this points at location DB+3. This is modified by the addition of the index 5 and the final effective address becomes DB+10 (octal).

2-56. Byte Addressing

The Load Byte (LDB), Store Byte (STB), and five Move Instructions (Section IV) use the byte addressing convention. Since the CPU is not specifically organized as a byte processor, the byte addressing convention uses the contents of the X Register, an indirect cell, or a stack word to specify the desired byte. For memory addressing (Load Byte and Store Byte instructions), the displacement value remains a word displacement. The byte data label in an indirect cell is an inflated value of two times the word displacement from DB. The contents of the X Register and/or an indirect cell indicate the desired byte in a byte array. For Move instructions, one or two of the TOS locations give a PB+ or DB+ relative byte index. The byte addressing range is therefore restricted to 32K words; 15 bits for word addresses and one bit for byte number. Four examples of byte addressing for memory address instructions (LDB and STB) are shown in figure 2-17. (The convention for the Move Instructions corresponds to the Direct, Indexed example in figure 2-17. The difference is that the byte index would be obtained from a TOS word rather than the X Register.)

2-57. DIRECT BYTE ADDRESSING. For direct, unindexed byte addressing, the displacement value given in the instruction word is strictly a word displacement and only the left byte of each word is addressable. As shown in figure 2-17, an "STB DB+7" instruc-
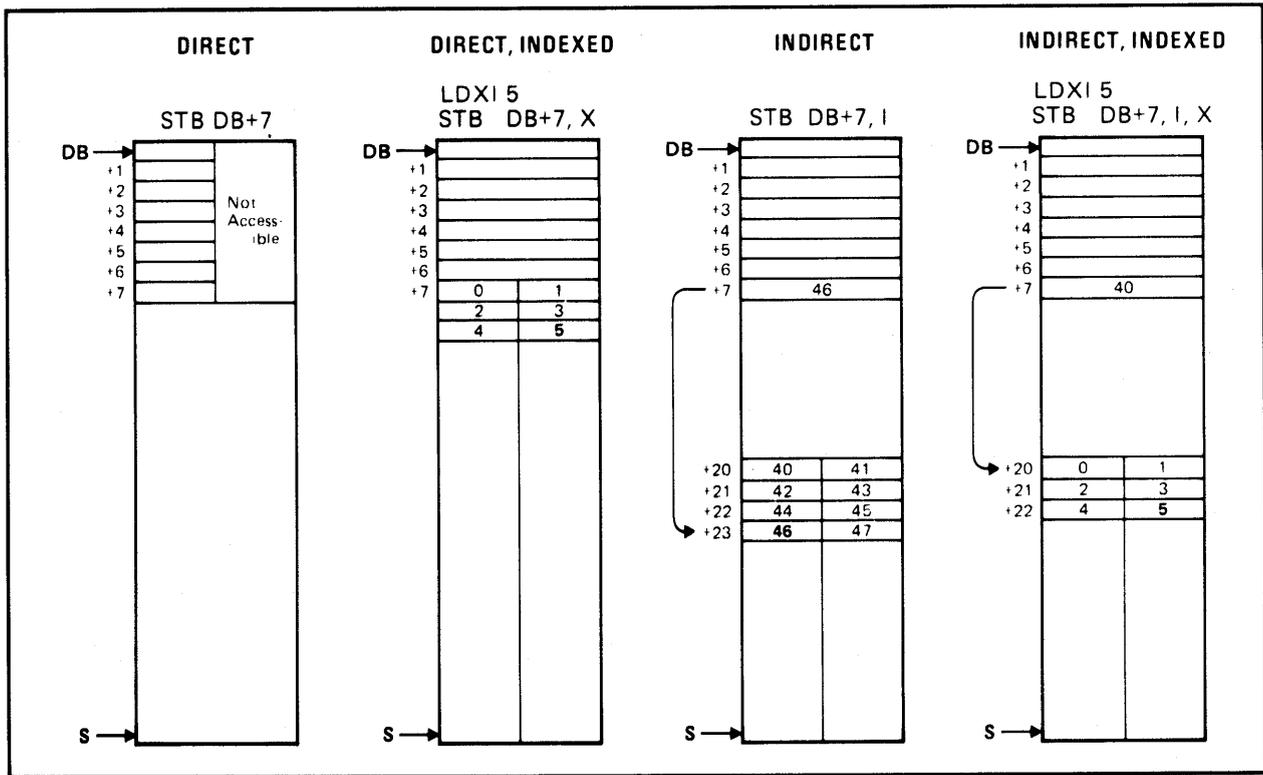
Figure 2-17. Byte Addressing Examples

tion would store a byte from the TOS into the left byte of the
DB+7 location.

2-58. DIRECT, INDEXED BYTE ADDRESSING. In the examples shown in
figure 2-17, the index is assumed to be 5. This is established
by the "LDXI5" instruction that precedes each STB instruction.
The "STB DB+7, X" instruction directly addresses location DB+7
and the index of 5 accesses the sixth byte. It should be noted
that the byte index starts at zero and that all even indexes are
left bytes and all odd indexes are right bytes.

2-59. INDIRECT BYTE ADDRESSING. For indirect, unindexed byte
addressing, the byte index is given in the indirect cell. As in
all indirect data addressing, the indirect reference is relative
to DB. Therefore, the "STB DB+7, I" instruction shown in figure
2-17 initially addresses the 47th byte in respect to DB. This
will be the left byte of DB+23. (Since there are two bytes per
word, divide the byte index by two to identify the word location;
a remainder of zero indicates the left byte and a remainder of
one indicates the right byte.)

2-60. INDIRECT, INDEXED BYTE ADDRESSING. For indirect, indexed
byte addressing, the displacement points to the indirect cell,
the indirect cell points to the start of the byte array, and the
index in the X Register points to the desired byte in the array.
This is shown by the "STB DB+7, I, X" instruction in figure 2-17.
The index in the X Register is again assumed to be 5. The dis-

2-41

placement points to the indirect cell at location DB+7 that contains the value 40. Dividing 40 by two gives the starting word address of the array as DB+20. Since the index is five, the location accessed is the sixth byte of the array. In this manner, the X Register acts as a byte index for ease of stepping through byte strings or byte arrays.

2-61. Double-Word Indexing

The Load Double Word Onto Stack and Store Double On TOS Into Memory instructions (Section IV) permit double-word indexing. When indexing is specified for these instructions, the hardware automatically multiplies the X Register contents by two during computation of the effective address. Therefore, an index value of 4 would imply the fifth double word in a double-word array.

2-62. Accessing DB- Area

The area between DB and DL can be accessed through indirect addressing and indexing. Figure 2-18 illustrates the technique of indirect addressing to access this area using both word addressing and byte addressing.

2-63. WORD ADDRESSING. The first example in figure 2-18 shows how to load the contents of the location at DB-10 onto the stack assuming that location DB+4 can be used for the indirect cell. The "LOAD DB+4, I" instruction initially references the indirect cell at DB+4. Instead of the usual positive number, location DB+4 contains the two's complement of the desired DB displacement. In octal, the two's complement of 10 is 177770. Remember that the contents of an indirect cell in a data segment is always DB+ relative displacement. Therefore, since addressing arithmetic is modulo 65K, adding 177770 to DB causes wrap-around and addresses the desired DB-10 location. From this point, indexing via the X Register can be applied.

2-64. BYTE ADDRESSING. The second example in figure 2-18 shows how the DB-10 byte can be loaded onto the stack assuming that location DB+4 can again be used for the indirect cell. The "LDB DB+4, I" instruction initially references the indirect cell which contains the two's complement (177770) of the desired byte displacement (-10) from DB. Remember that byte indexes are converted to word indexes by dividing by two. This would indicate location DB+77774 (left byte) which may or may not exceed the upper limit of memory, depending on the current absolute value of DB. (To allow for byte addressing in additional data segments where DB may not be between DL and Z, a check for this condition is made. Refer to paragraph 2-65.) If DB is not between DL and Z (this should happen only during privileged mode and is then called split stack), the byte will then be accessed without further bounds checking. If DB is between DL and Z, then the LDB instruction (or any other byte addressing instruction) tests this address to see if it is within the required DL to Z range. If the address is not within the range (which should be the case whether or not wrap-around has already occured), the instruction
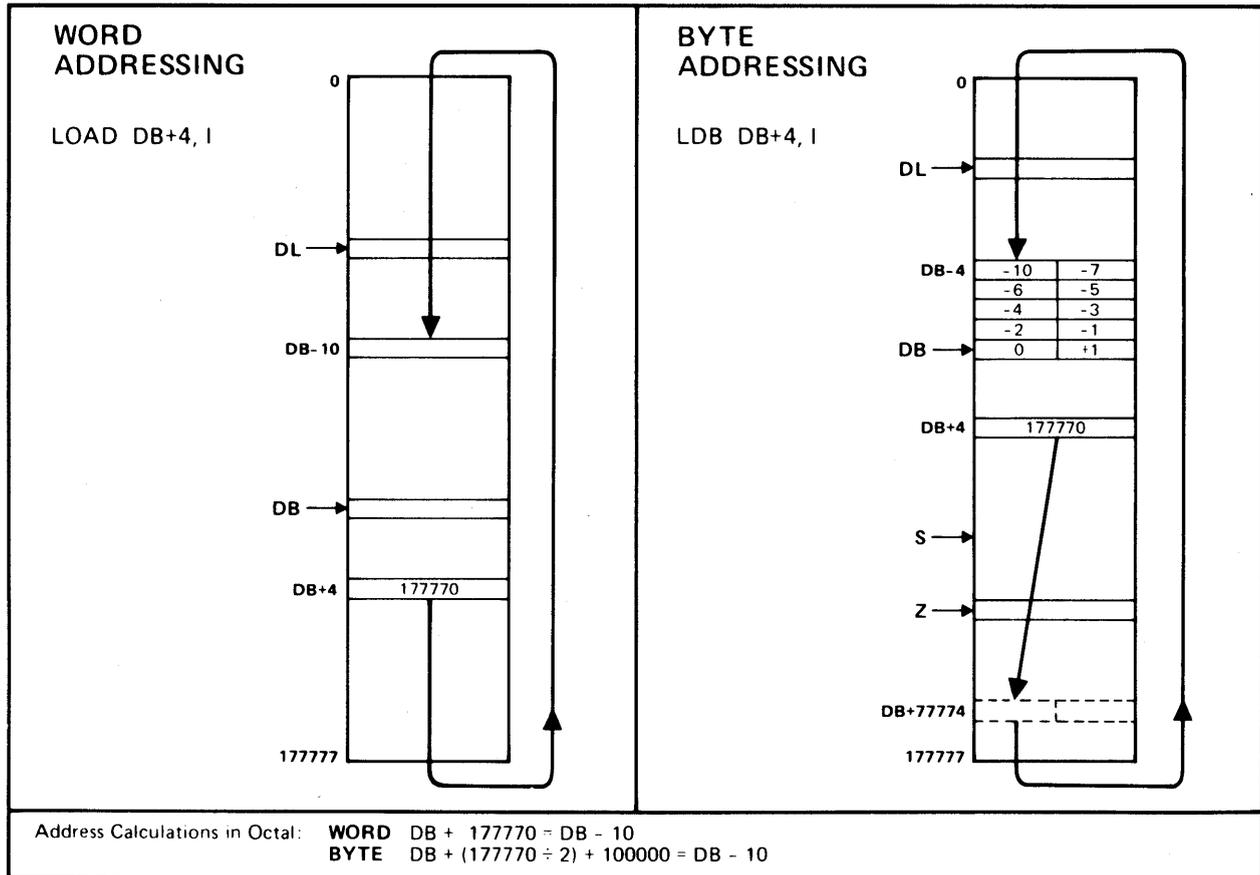
Figure 2-18. Accessing DB- Area

will add 32K (%100000) to the DB+77774 value. Assuming that wrap-around has not yet occured, this addition will cause it to occur and thus address the byte at byte address DB-10 (left byte in location DB-4). At this time, a second test is made to check if the effective address is within the DL to Z range. If the technique has been applied properly, the test will be affirmative and the byte will be transferred. If the test fails during user mode, there will be a bounds violation interrupt. If the test fails during privileged mode, the test results will be ignored and execution will continue (even if out of bounds), using the second referenced byte.

2-65. Bounds Checking

The CPU routinely checks all address references and TOS movements to ensure that such operations remain within legal bounds. Sufficient checks are made for all machine instructions to ensure that a nonprivileged user cannot adversely affect other users or the operating system. The basic bounds checks that are made for the applicable instruction types are discussed in paragraphs 2-66 through 2-70 and summarized in table 2-6. The boundry limits checked are illustrated in figure 2-19. If any of the bounds check fail during non-privileged user mode, there will be a bounds violation interrupt. Those checks whose results are ignored during privileged mode are so indicated.

CODE

PB →

Legal
Addresses
(for Program
Transfers
and User
References)

P →

PL →

DATA

Stack
Underflow

DL →

DB →

Legal
Addresses
for
User
References

User
Stack
Area

SM →
S →

Z →

INCREASING
ADDRESSES
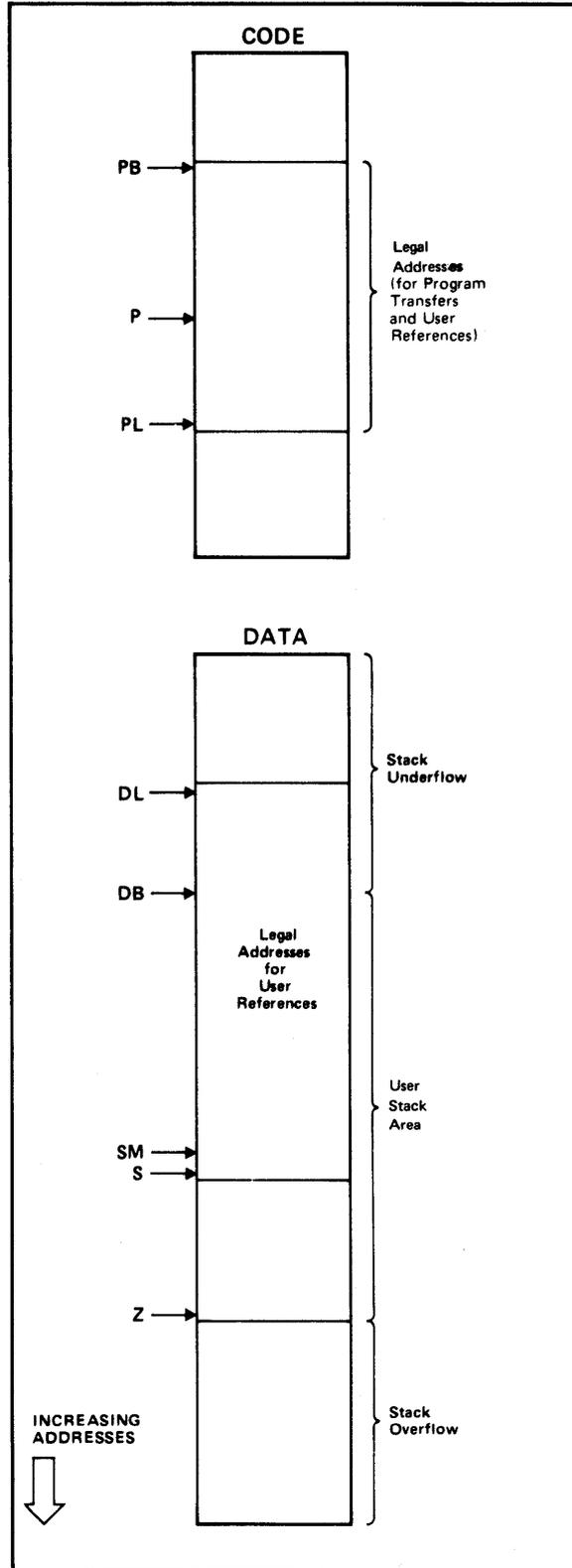
Stack
Overflow

Figure 2-19.   Addressing and Stack Bounds

2-66.   PROGRAM TRANSFER LIMIT. Program control cannot be passed
to any location beyond the limits defined by the contents of the
PB and PL Registers. For indirect branches, both the indirect and
direct references must be within limits. This also applies when
branching indirect via the data stack, except that the initial
reference must be within data stack limits DB and S rather than
code segment limits PB and PL.

2-67.   PROGRAM REFERENCE LIMITS. Some of the Memory Address in-
structions, all Loop Control instructions, and some Move instruc-
tions are capable of addressing locations in the code segment.
During privileged mode, these references can be made as desired.
During non-privileged user mode however, these references (both
direct and indirect) must be within the limits defined by PB and
PL.

2-68.   DATA REFERENCE LIMITS. During privileged mode, data ref-
erences are not subject to bounds checking. During non-privileged
user mode however, these references (both direct and indirect)
must be within the user's area defined by DL and S.

2-69.   STACK OVERFLOW LIMIT. Stack overflow is defined as moving
the S-pointer beyond the stack limit. Stack overflow occurs when
SM exceeds Z. Since SM is not necessarily the actual TOS (SM may
equal S or be up to four locations lower) and to allow marker
space for the remote possibility of a procedure call and an in-
terrupt while SM is at Z, there is a zone of approximately 128
locations beyond Z which could be filled with stack related data.
A stack overflow causes an interrupt which, under discretion of
the operating system, may extend the stack limit.

2-70.   STACK UNDERFLOW LIMIT. Stack underflow is defined as mov-
ing the S-pointer below the data base or, more strictly, moving
SM below DB. Since SM may not equal S, underflow can occur even
though S is up to three locations above DB.   During privileged
mode, stack underflow is not subject to checking.   During non-
privileged user mode however, stack underflow will cause an in-
terrupt. Users can access the area between DL and DB by indirect
addressing or indexing (paragraph 2-62) as long as SM does not
become less than DB. Although the hardware does address arithme-
tic modulo 64K, code segments and data stacks can not cross mem-
ory bank boundries. This restriction is handled by the operating
system.

2-71. CPU OVERVIEW

Operation of the CPU is controlled by the software set of in-
structions and the microprogram. Logically, the CPU (figure
2-20) consists of three sections; a microprocessor, processor
registers, and an arithmetic logic unit (ALU). The microproces-
sor receives an instruction word from Main Memory and translates
it into a microprogram starting address.   The microprogram is
then read out of read-only memory (ROM) and is decoded into a set
sequence of control signals. The processor registers are flip-
flop registers that can be loaded from the U-Bus (i.e., output of

Table 2-6. Bounds Checks Summary

Check	Definition	Mode
Program Transfer	$PB \leq E \leq PL$	Privileged, User
Program References	$PB \leq E \leq PL$	User only (except moves)
Data References	$DL \leq E \leq S$	User only
Stack Overflow	$SM > Z$	Privileged, User
Stack Underflow	$SM < DB$	User only
E = effective address of memory address		

the ALU) and read onto the R-Bus and/or S-Bus (inputs to ALU). The ALU executes various functions (add, subtract, etc.) on the R- and S-Bus inputs (with or without a shift) and outputs the result to either of the CPU registers for transmission out of the Central Processor Module or to the U-Bus for storage in one of the internal registers. For a more detailed discussion of the CPU logical components, refer to paragraph 2-75.

2-72. Pipelines

There are two pipelines in the CPU; a microcode pipeline and a data pipeline. Basically, the microcode pipeline consists of the Current Instruction Register (CIR), CMUX, Mapper, Look Up Table (LUT), VBUS MUX, ROM, ROR1, and ROR2. See figure 2-20. The data pipeline basically consists of the Store Logic, various registers, R- and S-Bus Logic, ALU, Shifter, and Decimal Corrector.

2-73. DATA PIPELINE. In general, the data pipeline picks up two operands via the R- and S-Bus Logic and R- and S-Bus Registers (figure 2-20) and inputs them to the ALU where a mathematical calculation can be performed. The result is then outputed to either the Shifter or Decimal Corrector where it can be either shifted (shift left 1, shift right 1, or swap bytes with or without clearing either byte), or its decimal arithmetic corrected. The final result is then put on the U-Bus and either stored in any one of the registers or input to the ALU a second time for additional calculations.

To give the data time to propagate through the entire pipeline, the data is stepped through in two steps. The first step is to read the operands from the two source registers to the input lines for the R- and S-Bus Registers. This is accomplished in one 175-nanosecond clock cycle. The second step is for the data to go through the ALU, Shifter or Decimal Corrector, and Store Logic and then be on the input to the selected store register. This is accomplished by the next 175-nanosecond clock cycle.
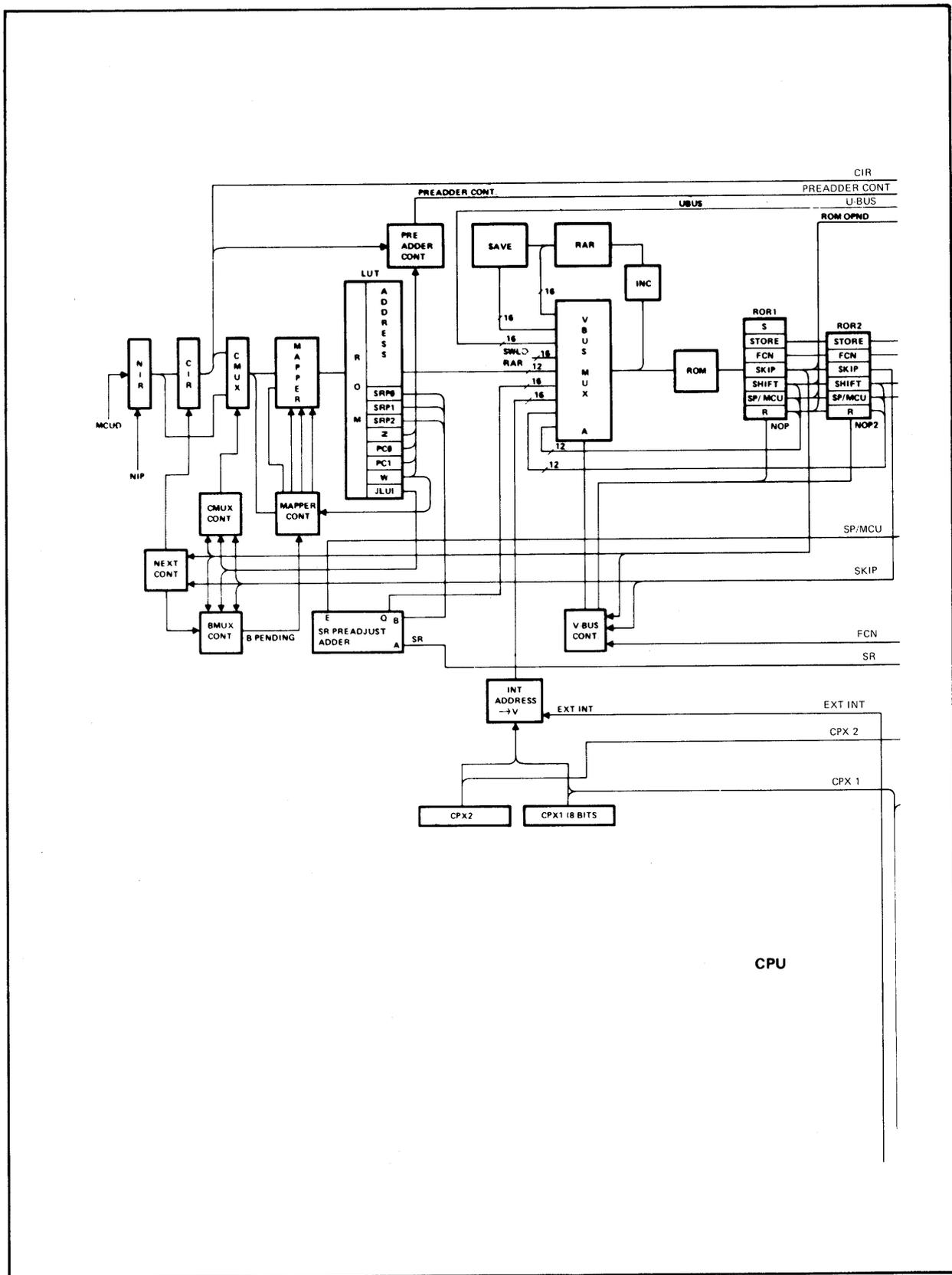
Figure 2-20. CPU Simplified Logic Diagram (Sheet 1 of 2)
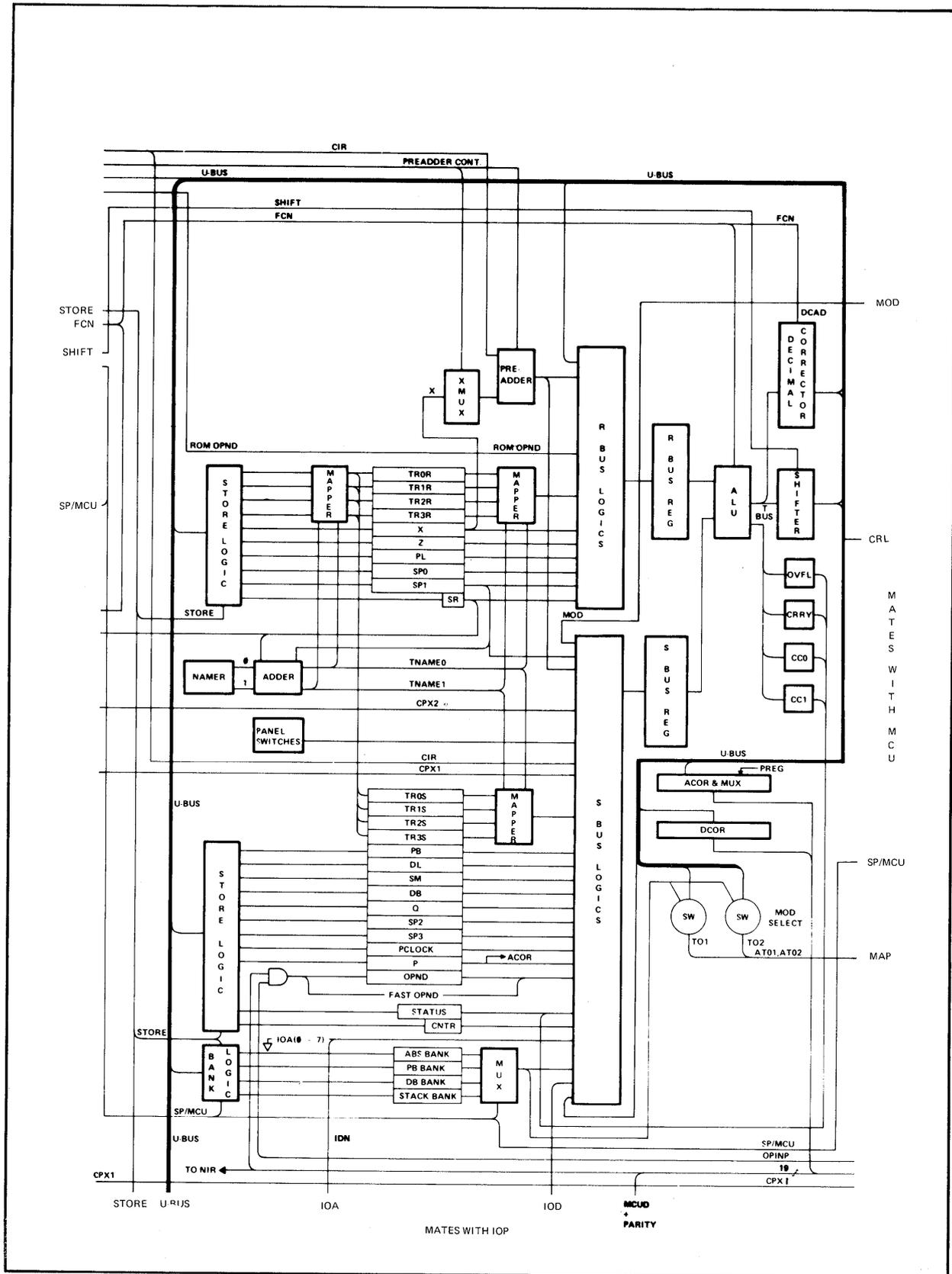
Figure 2-20. CPU Simplified Logic Diagram (Sheet 2 of 2)

The entire data calculation is accomplished by one microcode instruction which is also executed in two steps. During the first step, the microcode instruction is held in ROR1. Effectively, the only two microcode instruction fields being decoded during this clock cycle are the R- and S-Bus fields. (Refer to Section V for microcode instruction format descriptions.) These two fields cause the R- and S-Bus Logic to select the correct registers for the two operands and gate the operands to the R- and S-Bus Registers. The same clock cycle that gates the operands into the R- and S-Bus Registers also gates the current microcode instruction into ROR2 and gates the next microcode instruction into ROR1 as discussed in paragraph 2-74. It also gates the previous microcode instructions's final result into the register specified by the instruction's Store field. During the second step (current microcode instruction in ROR2), the instruction's Function field specifies what calculation is to be accomplished by selecting either the Shifter or Decimal Corrector and the instruction's Shift field specifies what the Shifter or Decimal Corrector is to accomplish. Also, the instruction's Store field specifies to the Store Logic which register to select to gate the final result appearing on the U-Bus. During the next clock cycle, the now completed microcode instruction is discarded by loading the next microcode instruction into ROR2 and the final result of the executed instruction is gated into the register specified by the Store Logic.

Each microcode instruction also contains two other fields that are decoded during execution; a Skip field and a Special field. The Special field controls the hardware that performs such operations as setting condition codes, popping the stack, and incrementing and decrementing the stack's SR Register. A complete listing of the operations specified by the Special field is contained in Section V. The Skip field specifies test conditions such as the status of internal flags, the contents of the SR Register as compared to zero through four, and operand results that appear on the T-Bus as compared to zero, non-zero, odd, and even. A complete listing of the test conditions specified by the Skip field is contained in Section V. The Skip field determines which condition will be tested for a possible skip. If the condition is met, ROR2 executes a No Operation (NOP), effectively skipping one microinstruction word. Other signals, such as NEXT, also come from the Skip field.

2-74. MICROCODE PIPELINE. In general, the microcode pipeline receives a requested instruction from Main Memory via the CTL Bus, MCU, and Next Instruction Register (NIR). See figure 2-20. The instruction is clocked into the CIR and then into the CMUX. If the pipeline has not been previously filled, the NIR output is clocked into the CIR and CMUX simultaneously, thus saving one clock cycle. Ten bits of the CMUX output go to the Mapper and 8 bits go to the Mapper Control. The 8-bit output of the Mapper goes to the Look Up Table (LUT) ROM. The LUT ROM produces a 12-bit microprogram starting address from the received instruction and also eight special use bits. The SRP0, SRP1, and SRP2 special use bits go to the SR Preadjust Adder. The Z, PC0, PC1,

and W special use bits go to the Preadder Control. (The W bit also goes to the Mapper Control.) The JULI special use bit goes to the BMUX Control and CMUX Control.

The 12-bit microprogram starting address from LUT is applied to the VBUS MUX. The VBUS MUX outputs 16 bits to the ROM and Increment (INC). The 16 bits applied to the ROM is the starting address for the microcode instruction providing no special conditions such as stack pre-adjust are needed. The 32-bit ROM output is clocked into ROR1. At the same time that the ROM is being accessed, the starting address is being sent to the INC circuit. During the same clock cycle that clocks the ROM output to ROR1, the address-plus-one is applied to the Address Register (RAR). The output of RAR goes back to the VBUS MUX. When, doing the next clock cycle, the incremented address goes to ROM, the new microcode instruction goes to ROR1 and the original microcode instruction goes from ROR1 to ROR2. The microcode pipeline is now packed, functioning, and incrementing one step at a time through the microcode. (Refer to paragraph 2-86 for microcode jump information.)

2-75. CPU Component Descriptions

The logical components of the CPU shown in figure 2-20 are described in paragraphs 2-76 through 2-128.

2-76. NIR. The NIR is a 16-bit register that is loaded with an instruction from Main Memory and provides storage for that instruction until the current instruction has been executed. This allows an instruction to be fetched from memory concurrently with the execution of the current instruction. The NIR is loaded by an NIP signal from the MCU operation decoder. The NIP signal is generated as a result of a microcode instruction Skip field code NEXT or the MCU field code NIR as described in Section V.

2-77. CIR. The CIR is a 16-bit register that contains the instruction currently being executed by the CPU. The CIR is loaded by an NIRTOCIR signal from the Next Control. The NIRTOCIR signal is generated as a result of a microcode instruction Skip field code NEXT or by the clock cycle after a Special field code CCPX as described in Section V. As previously discussed, if the pipeline has not been filled, the contents of the NIR goes directly to both the CIR and CMUX to save one clock cycle. The NIR and CIR allow one clock cycle to fetch one instruction from memory while the previous clock cycle is still executing an instruction. Instruction translation is accomplished from the CIR two clock cycles after the execution has begun until the execution is complete unless it is the right instruction of a stack-op. In the case of a Right Stack-Op instruction, the entire translation is accomplished from the CIR. The controlling factor concerning the execution of a Right Stack-Op instruction is the BMUX Control.

2-78. CMUX AND CMUX CONTROL. The CMUX is controlled by the Next Control and CMUX Control to determine whether the instruction from the NIR or CIR goes into the Mapper.

2-79. MAPPER AND MAPPER CONTROL. The Mapper combines the inputs from the CMUX and Mapper Control and generates an 8-bit output that addresses a specific location in the LUT ROM.

2-80. LUT ROM. The LUT ROM outputs a 12-bit address and eight special use bits as determined by the Mapper. The 12-bit address is applied to the VBUS MUX and the VBUS MUX generates a 16-bit output that addresses the initial microcode instruction that starts the accomplishment of the instruction from the NIR or CIR. The eight special use bits specify the mode of addressing being utilized for the memory reference instructions. The SP0, SP1, and SP2 bits are applied to the SR Preadjust Adder to define how many TOS registers must be valid before execution of the instruction can begin. Data bit 0 in the LUT ROM is the W-bit and bits 1 through 12 contain the starting address of the microprogram for the instruction to be executed. When a new instruction is to be executed, the W-bit is stored in the W-Bit Register. The W-bit has different meanings for different instructions and has a fixed, known value for every instruction as follows:

a. For STACKOPS (CIR (0:3) = %00) instructions, the W-bit has no meaning; it is set to logic 1 merely for convenience.

b. For SUBOP 1 (CIR (0:3) = %01) instructions:

   (1) The W-bit is set to logic 1 for instructions regarding P-relative addresses (some branches). In this case, CIR (10) is treated as a sign bit for the P-relative displacement in CIR (11:15). This bit controls the function of the Pre-Adder (add or subtract) so that a positive or negative number can be obtained from it.

   (2) The W-bit is set equal to logic 0 for shift instructions. In this case, the pre-added output is CIR (10:15), a 6-bit shift count, with zeros in all other bit positions.

c. For SUBOP 2 (CIR (0:3) = %02) instructions, the W-bit controls the function of the Pre-Adder. In all cases, the input to the Pre-Adder is CIR (8:15). When the W-bit is logic 0, the Pre-Adder is set to add. Since the second input to the Pre-Adder is logic 0 (no indexing), the output is -CIR (8:15) (= 317 - CIR (8:15)), a negative number.

d. For SUBOP 3 (CIR (0:3) = %03) instructions:

   (1) For SPECOP 00 (CIR (0:3) = %03), the W-bit is set to logic 0 which forces the Pre-Adder to the add function. In addition, only CIR (12:15) is applied to the Pre-Adder input. Therefore, the output is the K-field CIR (12:15).

(2) For SPECOP 01 through 17 (CIR (4:7) = %01 - %17), the W-bit causes the same action as in paragraph c above.

e. SUBOP %04 through %17 (CIR (0:3) = %04 - %17) instructions generally reference an operand in memory. The operations necessary to obtain the effective address is this operand are common to most of the instructions and, therefore, one microprogram is used for this calaculation. When one of these instructions is to be executed, it maps through the LUT to this microprogram to obtain the operand address. When this is done, the instruction then jumps to the microprogram that executes the specified instruction and the W-bit now becomes effective. The W-bit is set to logic 1. When the foregoing address calculation routine has been completed, a micro-operation (JLUI) in the ROM Skip field is executed. If the instruction does not specify indirect addressing or if one level of indirect addressing has been completed, the execution of JULI forces a microprogram jump to an address contained in the LUT. Since the contents of the CIR have not changed, the LUT would normally still be pointing to the address of the foregoing address calculation routine and an infinite loop would result. However, the W-bit now modifies the LUT entry address to a different, but related, address. This LUT address contains the microprogram address of the desired instruction to be executed.

2-81. VBUS MUX AND VBUS CONTROL. One of the nine inputs to the VBUS MUX is selected by the VBUS Control to be fed through the VBUS MUX which becomes a 16-bit address for the ROM. This address is also applied to the Incrementor (INC) which increments the address by one and applies this new address to the ROM Address Register (RAR).

2-82. RAR. The RAR is a 16-bit register that holds the address of the next microinstruction to be executed if no preempting conditions (interrupt, jump, etc.) occur. The RAR is loaded with the ROM address incremented by one and is automatically incremented every 175 nanoseconds by the INC until the end of the microprogram for the instruction is reached. Normally, the RAR is loaded from the INC. However, if a repeat is specified, the contents of the RAR does not change until the repeat is terminated. In addition to the 12-bit output from the LUT ROM, the RAR can be loaded from the ROM Output Register Rank 2 (ROR2), by a JMPGATE signal generated in response to a Function field code Jump (JMP) or Jump To Subroutine (JSB), by the interrupt logic due to an interrupt or power failure, from the U-Bus in response to an RAR store specified, or from the Hardware Maintenance Panel.

2-83. SAVE REGISTER. When a JSB is decoded by the Function Field Decoder, a JSB1 signal is generated and the contents of the RAR is loaded into the Save Register until a Return from Subroutine (RSB) is decoded by the Skip Field Decoder. The RSB signal loads the contents of the Save Register back into the VBUS MUX and from there back into the ROM which continues executing the microprogram with the microinstruction following the JSB.

2-84. ROM. The ROM accepts 16-bit addresses from the VBUS MUX and outputs 32-bit microinstructions of a microprogram to the ROM Output Registers (ROR1 and ROR2). The ROM contains 4096 (%7777), 32-bit instruction words. Each instruction generally calls several microinstructions from the ROM. For example, instructions that affect TOS will first call a microprogram routine to check that there are enough filled or vacant TOS registers to carry out the operation. Then, after one or more memory transfers to adjust the stack, the remaining microinstructions called by the instruction will begin. Updated addresses for succeeding microinstructions called by the instruction are furnished to the ROM every 175 nanoseconds by the RAR.

2-85. ROR1 AND ROR2. The 32-bit microinstruction words from ROR1 and ROR2 are divided into eight fields, each field containing from three to five bits. Each field, when decoded, produces a set of microcode signals that control the operation of the CPU. (Refer to Section V of this manual.) The 32-bit output of the ROM is loaded into ROR1 on each 175-nanosecond clock cycle. On the next clock cycle, six of the seven microinstruction word fields are transferred from ROR1 into ROR2 while ROR1 is receiving the next microinstruction word. (Initially, it takes two clock cycles to fill the pipeline, but thereafter ROR2 receives a new microinstruction word on each successive clock cycle.) Two ROM output registers allow the S- and R-Bus fields to be decoded in advance of the rest of the instruction word. Therefore, the S- and R-bus selection occurs is ROR1 and the selected data will be ready and waiting on the U-Bus by the time the rest of the word is decoded in ROR2. Each field of the ROM output word is separately decoded as discussed in Section V. The S-Bus field selects one of 31 registers or sets of lines to be loaded into the S-Bus Register. The R-Bus field selects one of one of 15 processor registers or sets of lines to be loaded into the R-Bus Register. The Store field selects one of 29 registers in which to store the U-Bus data. The Function field specifies the function that the ALU is to perform on the two operands in the S- and R-Bus Registers. The Shift field specifies how the T-Bus data will be shifted onto the U-Bus. The Special field performs many varied operations including the generation of POP and memory opcode and CTL Bus request signals.

The Skip field specifies a test condition, which if true, causes the microcode instruction in the next ROM address not to be executed. (A complete list of test conditions that can be specified by the Skip field is contained in Section V.) However, if the current instruction is a microcode jump instruction, the jump will be executed only if the condition being tested is true. In the case where the next microcode instruction is not to be executed, the skip condition is tested while the microcode instruction is in ROR2. This means that the instruction to be skipped is in ROR1. The clock cycle that moved the instruction to be skipped from ROR1 into ROR2 also sets the NOP2 flip-flop. This causes the ALU to add, forces the Shift field to a pass function, and the Store field not to be decoded. However, the operands specified by the R- and S-Bus fields of the instruction to be

skipped were already clocked into the R- and S-Bus Registers so that the data on the U-Bus at the end of the NOP cycle is the sum of the contents of the source registers.

2-86. Microcode Jumps. Microcode jumps can be taken from either ROR1 or ROR2. The jumps can be taken from ROR1 only under the condition that the jump has an unconditional skip code and the instruction in ROR2 meets one or more of the following conditions: is cancelled by NOP2; is a ROM Immediate type instruction without a data-dependent skip; contains a NOP skip function; and/or contains a non-data-dependent skip test (skip codes 14 through 27, 32, 33, and 34) which is not met or if an ROR1 jump has just been completed. All other microcode jumps will be executed from ROR2.

An unconditional jump is a jump that occurs without regard to the data. If the microcode calls for an unconditional jump, a jump target address is selected out of the Shift, Special, and R-Bus fields of the microcode instruction in ROR1 (ROR2 if previous microcode instruction contained a data-dependent skip condition) and applied back to the VBUS MUX so that the new ROM microcode instruction is sent to ROR1. The target address goes to the INC, is incremented by one, and the new target address plus one is stored in the RAR until the next clock cycle when it is applied to the VBUS MUX for consecutive addressing of the microcode instructions.

The jumps that are executed from ROR2 because none of the fast-jump conditions were present for ROR1 and the conditional jumps that are always executed from ROR2 behave as follows: Not Taken, next line in sequence executed on next clock; Non-Data-Dependent Taken, one overhead clock required (NOP2 effective) before target line executed; Data-Dependent Taken, two overhead clocks required (FREEZE, NOP2) before target line executed. Execution of jumps in ROR2 inhibit any fast jumps from ROR1 being executed. Therefore, if there are two consecutive lines of microcode containing jumps, the jump in ROR2 will be taken and the jump in ROR1 will be ignored.

The microcode instruction calling for a jump comes out of ROM and into ROR1 which decodes the R- and S-Bus fields as discussed in paragraph 2-85. The R- and S-Bus field information is sent through the R- and S-Bus Logic and is waiting at the inputs of the R- and S-Bus Registers. On the next clock cycle, the jump instruction goes to ROR2 and the R- and S-Bus field data is clocked through the R- and S-Bus Registers. The T-Bus data is loaded from ROR2 to feed ROM so that, on the next clock cycle, the address of the jump-to-microcode instruction goes to ROM. As the new instruction is clocked into ROR2, the jump-to-microcode address plus one goes into the RAR and the operation resumes stepping through the microcode.

2-87. S-Bus Field Decoder (S). The S-Bus Field Decoder (bits 0 through 4) selects one of 32 registers or sets of lines to be loaded into the S-Bus Register. S-Bus field code definitions are contained in Section V.

2-88. Store Field Decoder (STORE). The Store Field Decoder (bits 5 through 9) selects one of the Store Logic registers or other destinations outside the CPU for the U-Bus data. Store field code definitions are contained in Section V.

2-89. Function Field Decoder (FCN). The Function Field Decoder (bits 10 through 14) specifies the function to be performed by the ALU on the two operands in the R- and S-Bus Registers. Function field code definitions are contained in Section V.

2-90. Skip Field Decoder (SKIP). The Skip Field Decoder (bits 15 through 19) determines which condition will be tested for a possible skip. If the condition is met, ROR2 will execute a NOP, effectively skipping one microinstruction word. The Skip field also specifies the conditions under which a JMP or JSB will be executed if coded in the microinstruction. Other signals, such as NEXT which calls the next instruction from memory, are also decoded from the Skip field. Skip field code definitions are contained in Section V.

2-91. Shift Field Decoder (SHIFT). The Shift Field Decoder (bits 20 through 22) specifies how the T-Bus data will be shifted. In addition, the Shift field generates the Scratch Pad 1 and Scratch Pad 3 Register shift signals used in conjunction with the Function field. The Shift field code definitions are contained in Section V.

2-92. Special Field Decoder (SP). The Special Field Decoder (bits 23 through 27) performs varied operations such as generating memory operation code signals and the POP signal. Special field code definitions are contained in Section V.

2-93. MCU Option Field Decoder (MCU). The MCU Option Field Decoder (bits 23 through 27) uses the same bits as the Special Field Decoder. The Special Field Decoder is disabled and the MCU Option Field Decoder is enabled when executing an S-Bus field code RBR or a Store field code BUS, BSP0, BSP1, or SBR. The MCU Option Field Decoder initiates transfers to or from memory and transfers from ACOR to the Operand, Next Instruction, or Command Registers via the CTL Bus. MCU Option field code definitions are contained in Section V.

2-94. R-Bus Field Decoder (R). The R-Bus Field Decoder (bits 28 through 31) selects one of 16 registers or sets of lines for loading into the R-Bus Register. R-Bus field code definitions are contained in Section V.

2-95. PROCESSOR REGISTERS. Except for the Operand (OPND), I/O Address, I/O Direct Data In, CPX1, and CPX2 Registers, the processor registers can be selectively loaded from the U-Bus and selectively read into the R- and/or S-Bus Registers. The processor registers are illustrated in similar readout groups in figure 2-20. For example, the X, Z, PL, SP0, and SR Registers can be read out only to the R-Bus Register. The SP1 Register can be read out to either the R- and/or S-Bus Registers. Similarily, the PB, DL, SM, DB, Q, SP2, SP3, PCLOCK, and OPND Registers can be read out only to the S-Bus Register. Descriptions of the individual processor registers, including the renamer logic, are contained in paragraphs 2-96 through 2-115. In addition, the actions of many of the processor registers in an operating environment are discussed in paragraphs 2-16 through 2-70.

2-96. Renamer Logic. The renamer logic consists the Namer, Adder, three Mappers, four TOS registers (TR0 through TR3), and the SR Register. These components are designated as the TOS register renamer, or simply, the renamer. The renamer permits fast access to the TOS elements by renaming the registers when stack elements are added or deleted (rather than transferring data from register to register). The ROM microprograms know TR0 through TR3 only by the names RA (top), RB, RC, and RD. The namer includes a 2-bit Namer Register that tells the Mappers which of the four TOS registers (TR0 through TR3) is RA, RB, RC, and RD as listed in table 2-7. The Namer Register is decremented each time a stack element is added (PUSH) and incremented each time a stack element is deleted (POP). To keep track of how many elements are in the TR0 through TR3 registers, the 3-bit SR Register is incremented by PUSH and decremented by POP in step with the Namer Register. When the SR Register count is zero, there are no elements in the TR0 through TR3 registers. This would indicate to a ROM microprogram not to look for the TOS in the CPU and that one or more memory fetches may be required. The Adder combines the outputs of the Namer Register, SR Register, and Scratch Pad 1 Register (SP1) and generates the TNAME signals (bits 0 and 1) for the Mappers. (Refer to table 2-7.) The Mappers use the TNAME code to control access to the TOS registers (TR0 through TR3). The TNAME code specifies which of the TOS registers is RA, RB, RC, and RD as listed in table 2-7.

Table 2-7. TOS Namer Relationships

TNAME Code =	00	01	10	11
RA =	TR0	TR1	TR2	TR3
RB =	TR1	TR2	TR3	TR0
RC =	TR2	TR3	TR0	TR1
RD =	TR3	TR0	TR1	TR2

2-97. TOS Registers. The TOS registers consist of eight 16-bit registers designated TR0R through TR3R and TR0S through TR3S. The two groups of registers always contain the same data (i.e., TR0R = TR0S, TR1R = TR1S, etc.). The registers contain up to four of the top elements of the current data stack. The TOS registers are read by R-Bus field codes RA, RB, RC, RD, and MREG and by S-Bus field codes RA, RB, RC, RD, and QDWN as discussed in Section V. The TOS registers are loaded by Store field codes RA, RB, RC, RD, PUSH, and QUP as discussed in Section V.

2-98. Index Register (X). The Index Register (X Register) is a 16-bit register that contains the index word to be used by memory reference instructions if indexing is specified. Certain other instructions use the X Register for parameters or addresses. (Refer to paragraph 2-48.) The X Register is read by R-Bus field codes X and XC and loaded by Store field code X.

2-99. Stack Limit Register (Z). The Stack Limit Register (Z Register) is a 16-bit register that contains an absolute address pointing to the top memory location available to the current data stack. Although there are 128 word locations above the stack limit, they are reserved for stack markers in the event of an interrupt. (Refer to paragraph 2-28.) The Z Register is read by R-Bus field code Z and loaded by Store field code Z.

2-100. Program Limit Register (PL). The Program Limit Register (PL Register) is a 16-bit register that contains the absolute address of the upper location of the current program segment. (Refer to paragraphs 2-24 through 2-28.) The PL Register is read by R-Bus field code PL and loaded by Store field code PL.

2-101. Scratch Pad 0 Register (SP0). The Scratch Pad 0 Register (SP0 Register) is a 16-bit register that is used by the CPU to store partial results during various CPU routines and as addresses during memory transfers. The SP0 Register is read by R-Bus field code SP0 and loaded by Store field codes SP0 and BSP0.

2-102. Scratch Pad 1 Register (SP1). The Scratch Pad 1 Register (SP1 Register) is a 16-bit register that is used by the CPU to store partial results during various microprogram routines. The SP1 Register can be left shifted and provides serial data input to bit 15 and output from bit 0. The SP1 Register is read by R-Bus field code SP1, loaded by Store field code SP1, and shifted by Function field codes CTSD, DVSB, and QASL. In addition, the SP1 Register can be read onto the S-Bus by S-Bus field code SP1 (code is not the same as R-Bus field code SP1).

2-103. Stack Register (SR). The Stack Register (SR Register) is a 3-bit register counter that provides the number of TOS registers that are currently in use. The SR Register works in conjunction with the Namer Register to locate and access any of the top four elements of the data stack. (Refer to paragraph 2-21.) The SR Register is read by R-Bus field code SR and modified by Store field code PUSH and Special field codes INSR, DCSR, POPA, CLSR, and POP.

2-104. Program Base Register (PB). The Program Base Register (PB Register) is a 16-bit register that contains the absolute address of the bottom location of the current program segment. (Refer to paragraphs 2-21 through 2-28.) The PB Register is read by S-Bus field code PB and loaded by Store field code PB.

2-105. Data Limit Register (DL). The Data Limit Register (DL Register) is a 16-bit register that contains the absolute address of the bottom usable location in the current data stack. (Refer to paragraphs 2-21 through 2-28.) The DL Register is read by S-Bus field code DL and loaded by Store field code DL.

2-106. Stack Memory Register (SM). The Stack Memory Register (SM Register) is a 16-bit register that contains the absolute address of the top element of the data stack in memory. Depending on the number of TOS registers in use (specified by contents of SR Register), this address can be from zero to four locations below the actual TOS. (Refer to paragraphs 2-21 through 2-28.) The SM Register is read by S-Bus field code SM and loaded by Store field code SM.

2-107. Data Base Register (DB). The Data Base Register (DB Register is a 16-bit register that is one of the stack limit reg- isters. The DB Register contains the absolute address of the first location of directly addressable storage in the current data stack. (Refer to paragraphs 2-21 through 2-28.) The DB Register is read by S-Bus field code DB and loaded by Store field code DB.

2-108. Q Register (Q). The Q Register is a 16-bit stack marker register that contains the absolute address of the current stack marker being used within the data stack. (Refer to paragraphs 2-21 through 2-28.) The Q Register is read by S-Bus field code Q and loaded by Store field code Q.

2-109. Scratch Pad 2 Register (SP2). The Scratch Pad 2 Register (SP2 Register) is a 16-bit register that is used by the CPU to store partial results during various microprogram routines. The SP2 Register is read by S-Bus field code SP2 and loaded by Store field code SP2.

2-110. Scratch Pad 3 Register (SP3). The Scratch Pad 3 Register (SP3 Register) is a 16-bit register used by the CPU to store partial results during various microprogram routines. The SP3 Register can be right shifted and provides serial data input to bit 0 and output from bit 15. The SP3 Register is read by S-Bus field code SP3, loaded by Store field code SP3, and shifted by Function field codes CTSD, MPAD, and TASR.

2-111. Process Clock Register (PCLOCK). The Process Clock Reg- ister (PCLOCK Register) is a 16-bit register counter. The PCLOCK Register is loaded and read by software instructions and is continuously incremented as long as the CPU is not executing on the Interrupt Control Stack (ICS FLAG = 0) or is not halted.

The clocking interval is 1.001 ms. The maximum range of the clock before rollover is approximately 65.5 seconds.

2-112. Program Counter Register (P). The Program Counter Register (P Register) is a 16-bit register that contains the absolute address of the next program instruction to be fetched from memory. During execution of Skip field code NEXT, the P Register and PB-Bank Register are used to select a memory module and prefetch the instruction following the one which is about to be executed. (Refer to paragraphs 2-21 through 2-28.) The P Register is read by S-Bus field code P and loaded by Store field code P.

2-113. Operand Register (OPND). The Operand Register (OPND Register is a 16-bit register that provides storage for data read from memory by the CPU. The OPND Register is loaded by an OPINP signal from the Operand In Process (OPINP) flip-flop in the MCU operation decoder as a result of MCU options OPND, RNWA, RWA, and RWAN. The OPND Register is read by an RDOPND signal from the S-Bus Decoder as a result of S-Bus field code OPND. When the CPU freezes for an operand, the operand from memory goes directly to the S-Bus Logic as well as into the OPND Register. It is then loaded into the S-Bus Register to await CPU operation.

2-114. Status Register (STA). The Status Register (STA Register) is a 16-bit register that indicates the current status of the CPU hardware. (The status word format is discussed in paragraph 2-45.) The STA Register is read by S-Bus field code STA and loaded by Store field code STA. Status bits are also affected by Function field codes CADO, SUBO, INCO, and ADDO; and by Special field codes CCB, SCRY, CCRY, POPA, SOV, CLO, CCZ, CCL, CCG, CCE, and CCA.

2-115. Counter Register (CNTR). The Counter Register (CNTR Register is a 6-bit register that is used as a repeat counter by the CPU. The two's complement of the desired count is loaded into the CNTR Register and the register is then incremented for each repeated execution until it contains all ones as indicated by a CTRM code from the Skip field. The CNTR Register is affected or referenced by S-Bus field codes CTRI and CTRH, Function field code REPN, Store field codes CTRL and CTRH, Special field code INCT, and Skip field code CTRM. Additionally, the CNTR Register saves the contents of the SR Register when the CPU is put in the Halt Mode. Therefore, after a halt has occured, the CNTR Register can be displayed to show what the contents of the SR Register was just prior to the halt.

2-116. OVERFLOW FLIP-FLOP (OVFL). The Overflow flip-flop controls the status word overflow bit (bit 4) and stores the state of the Overflow signal from the ALU when the OFCENB signal is true. The Overflow flip-flop is set and cleared by Special field codes SOV and CLO respectively. Refer to paragraph 2-45.

2-117. CARRY FLIP-FLOP (CRRY). The Carry flip-flop controls the status word carry bit (bit 5) and stores the state of the Carry signal from the ALU when the OFCENB signal is true. The Carry flipflop is set and cleared by Special field codes SCRY and CCRY respectively. Refer to paragraph 2-45.

2-118. CONDITION CODE LOGIC (CC0 AND CC1). The condition code logic controls the condition code. Refer to paragraph 2-46.

2-119. PRE-ADDER. The Pre-Adder is used to gain a speed increase for instructions that use or perform computations on CIR bits. For example, when executing indexed memory reference instructions (not indirect), the proper CIR displacement field is pre-added to the X Register contents. Therefore, the final absolute address can be computed in only one clock cycle by adding the output of the Pre-Adder to the contents of the base register (PB, DB, Q, or Z).

2-120. R-BUS REGISTER. The R-Bus Register is a 16-bit register that provides buffer storage between the R-Bus and the ALU. The R-Bus Register can be left-shifted one bit position (refer to Function field code QASL, Section V) and is loaded from the R-Bus. Refer to R-Bus field code definitions.

2-121. S-BUS REGISTER. The S-Bus Register is a 16-bit register that provides buffer storage between the S-Bus and the ALU. The S-Bus Register can be right-shifted one bit position (refer to Function field code QASR, Section V) and is loaded from the S-Bus. Refer to S-Bus field code definitions.

2-122. ALU. The ALU combines the R- and S-Bus data and generates functions that are divided into two modes or groups; arithmetic functions and logic functions. The 16-bit output of the ALU is placed on the T-Bus for either the Shifter or Decimal Corrector.

2-123. SHIFTER. The Shifter performs all shifts and rotates (left shift, right shift, right-left swap, etc.) on the T-Bus data as directed by the Shift Field Decoder. The output of the Shifter is placed on the U-Bus for storage in one of the U-Bus registers.

2-124. DECIMAL CORRECTOR. The Decimal Corrector adds six to each group of four bits in the output from the ALU and generates carries to the next group as required to yield a correct decimal addition. Each group of four bits in the source operands must be in the range of 0 to 9. If an invalid digit is detected during the add cycle, overflow will be true.

2-125. ADDRESS COMPUTER OUTPUT REGISTER (ACOR). The ACOR is a 16-bit register that functions as a memory address buffer between the U-Bus and the CTL Bus.

2-126. DATA COMPUTER OUTPUT REGISTER (DCOR). The DCOR is a 16-bit register that functions as a buffer for memory bound data and operand address transfers between the U-Bus and the CTL Bus.

2-127. INTERRUPT STATUS REGISTER 1 (CPX1). The Interrupt Status 1 Register (CPX1 Register) provides 16 bits that are used to monitor the system Run Mode interrupt status. When a Run Mode interrupt occurs, the CPU reads the CPX1 Register and checks its contents for the cause of the interrupt. The CPX1 Register is read by S-Bus field code CPX1 and is affected by Special field code CCPX as discussed in Section V. Each of the CPX1 Register's 16 bits (when true) signifies a specific Run Mode interrupt as follows:

Bit 0: Integer Overflow   Bit 8: External Interrupt
Bit 1: Bounds Violation   Bit 9: Power Fail Interrupt
Bit 2: Illegal Address   Bit 10: 0
Bit 3: CPU Timer   Bit 11: ICS Flag
Bit 4: System Parity Error   Bit 12: DISP Flag
Bit 5: Address Parity Error   Bit 13: Emulator
Bit 6: Data Parity Error   Bit 14: I/O Timer
Bit 7: Module Interrupt   Bit 15: Option Present

2-128. INTERRUPT STATUS REGISTER 2 (CPX2). The Interrupt Status 2 Register (CPX2 Register) is used to monitor the system's Halt Mode interrupt status. When a Halt Mode interrupt occurs, the CPU reads the CPX2 Register and checks its contents for the cause of the interrupt. The CPX2 Register is read by S-Bus field code CPX2 and is affected by Special field code CCPX as discussed in Section V. Each of the CPX2 Register's 16 bits (when true) signifies a specific Halt Mode interrupt as follows:

Bit 0: Run Switch   Bit 8: Execute Switch
Bit 1: Dump Switch   Bit 9: Increment Address
Bit 2: Load Switch   Bit 10: Decrement Address
Bit 3: Load Register   Bit 11: 0
Bit 4: Load Address   Bit 12: 0
Bit 5: Load Memory   Bit 13: Inhibit PFARS
Bit 6: Display Memory   Bit 14: System Halt
Bit 7: Single Instruction   Bit 15: Run Flip-Flop

2-129. CPU Servicing Information

Physically, the basic CPU consists of the nine PCA's contained in slots A2 through A10 of Card Cage No. 1 as shown in tables 1-1 through 1-3. All CPU PCA's are nonrepairable PCA's and must be replaced if found defective. No repair procedures are required. However, four of the six CPU PCA's contain jumpers or switches that must be properly configured as discussed in paragraphs 2-130 through 2-133.

2-130. READ-ONLY MEMORY (ROM) PCA. The ROM PCA contains four jumpers (W5 through W8) that must be installed to reflect the type of ROM's loaded on the PCA. If the ROM PCA is loaded with ROM's having a capacity of 1K words, install the four jumpers nearest the 1K marking as shown in figure 2-21. If the ROM PCA is loaded with ROM's having a capacity of 2K words, install the four jumpers nearest the 2K marking.

TOP 1, 3, ←————————→ 79   1, 3, ←————————→ 79
BOTTOM 2, 4, ←————————→ 80   2, 4, ←————————→ 80
P1

ROM PCA
30003-60021

J1
TOP 2, 4, ←————→ 50   J2   2, 4, ←————→ 50   J3   2, 4, ←————→ 50
BOTTOM 1, 3, ←————→ 49   1, 3, ←————→ 49   1, 3, ←————→ 49

1K

} 1K POSITIONS

W5
W6   } 2K POSITIONS
W7
W8

2K
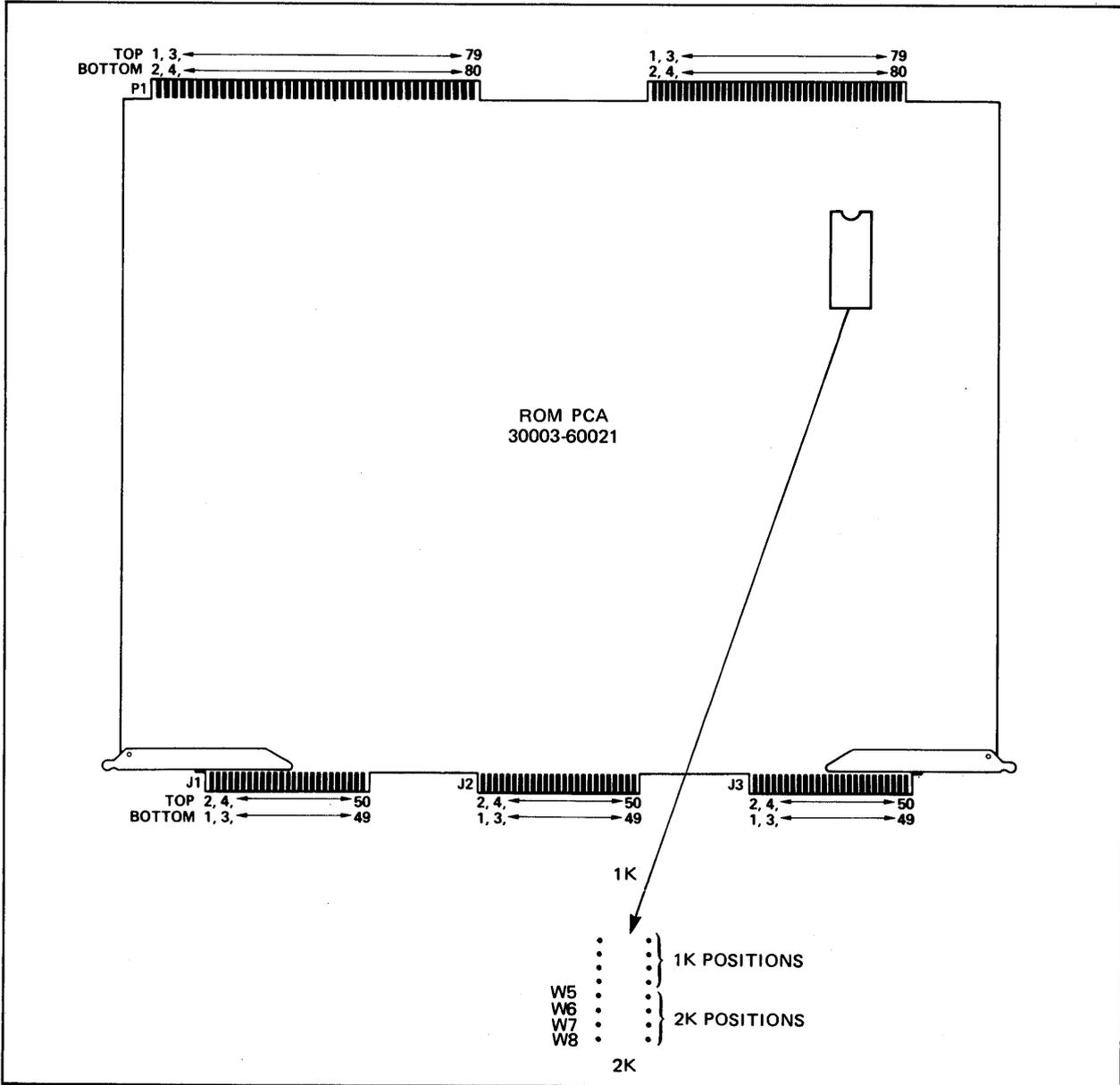
Figure 2-21.   ROM PCA Jumper Locations

2-131. SKIP AND SPECIAL FIELD (SSF) PCA. The SSF PCA contains
two synchronizing jumpers (W1 and W2) that must be installed to
reflect that there is only one CPU in the system.   Ensure that
jumpers W1 and W2 are installed exactly as shown in figure 2-22.

```
TOP 1, 3, ◄──────────────► 79        1, 3, ◄──────────────► 79
BOTTOM 2, 4, ◄──────────────► 80      2, 4, ◄──────────────► 80
P1 ▐█████████████████████████▌    ▐█████████████████████████▌
```

U18 U28 U38

U17 U27 U37

W1

SKIP AND SPECIAL
FIELD PCA
30003-60022

U51 U61 U71

W2 — M
• S

```
J1 ▐██████████████▌   J2 ▐██████████████▌   J3 ▐██████████████▌
TOP 2, 4, ◄──────► 50       2, 4, ◄──────► 50       2, 4, ◄──────► 50
BOTTOM 1, 3, ◄──────► 49     1, 3, ◄──────► 49       1, 3, ◄──────► 49
```
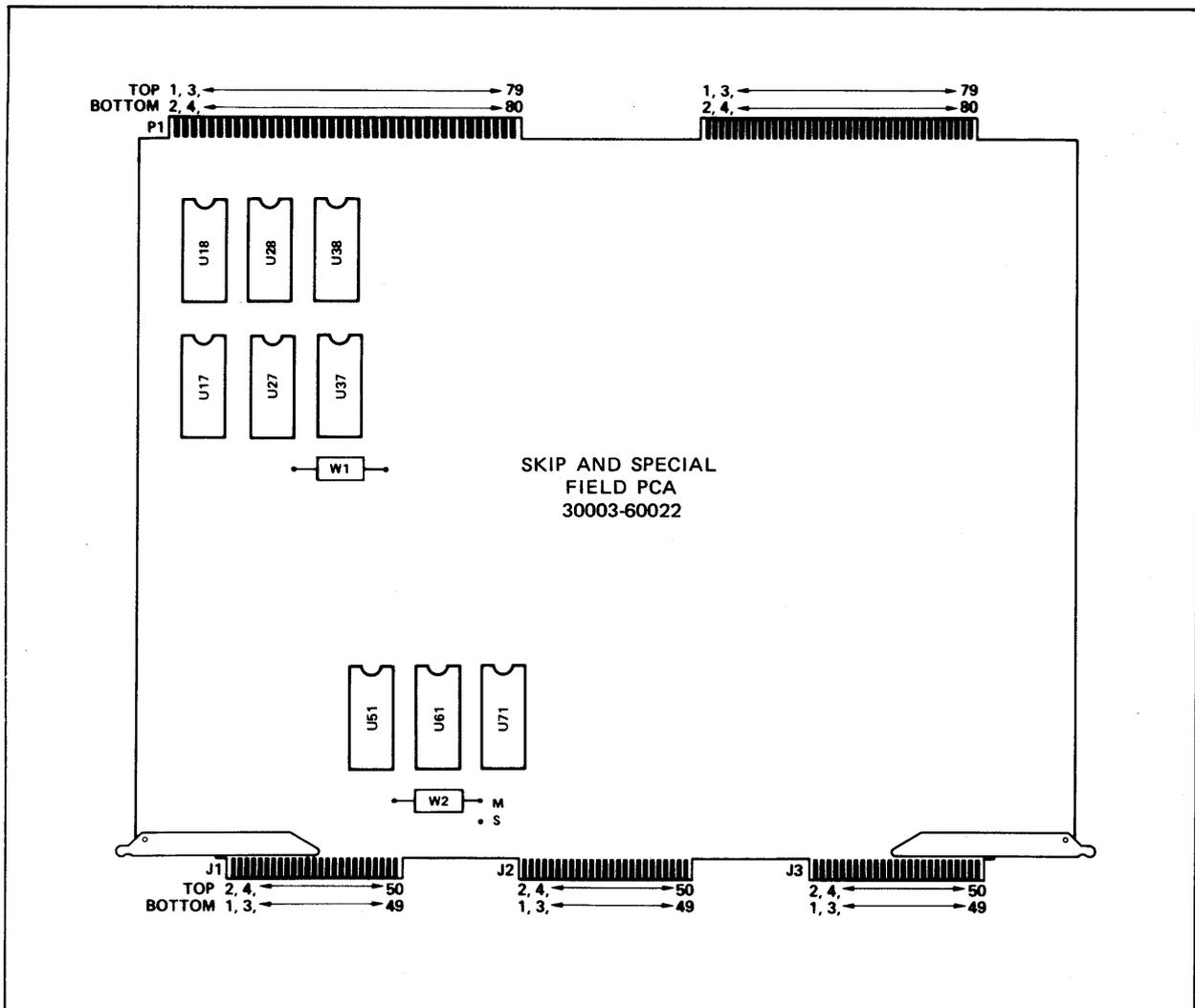
Figure 2-22.   SSF PCA Jumper Locations

2-132.   S-BUS PCA.   The S-Bus PCA contains three selector switch-
es (S1, S2, and S3) as shown in figure 2-23.   Set switch S3 to
match the computer system's Main Memory size.   Switches S1 and S2
are used for memory interleaving.   At present, memory interleav-
ing is not factory supported and switches S1 and S2 must be con-
figured for non-interleaving in accordance with table 2-8.

Memory interleaving requires two Memory Control and Logging PCA's
be installed in the system, each supporting one, two, or four
Semiconductor Memory Array PCA's.   The memory sizes that can be
interleaved are limited to 256K, 512K, and 1024K words.   The re-
quired switch configurations of switches S1 and S2 on the S-Bus,
IOP, and Selector Channel Register PCA's for memory interleaving
are listed in table 2-8.

Table 2-8. Memory Interleaving Switch Configurations

Mode	S1						S2					
	1	2	3	4	5	6	1	2	3	4	5	6
Non Interleaving 1 to 4 PCA's/MCL	C	C	C	C								C*
Interleaving 1024K 4 PCA's/MCL			C	C	C	C						C*
Interleaving 512K 2 PCA's/MCL			C		C		C	C				C*
Interleaving 256K 1 PCA/MCL				C	C				C	C	C*	

*Applies to IOP PCA only.

On the S-Bus, IOP Bus, and Selector Channel Register PCA's, open all switch positions of S1 and S2. Then, close those switch positions indicated with a C on all three PCA's for the applicable mode.

2-133. CURRENT INSTRUCTION REGISTER (CIR) PCA. The CIR PCA contains eight jumpers (W1 through W8) as shown in figure 2-24. If neither the HP 32105A APL (A Programming Language), HP 32233A COBOL '74 or, the Extended Instruction Set (EIS) PCA, part no. 30012-60001 are installed in the system, W1 through W8 are all installed. If the EIS PCA is installed in the system, remove jumpers W1 and W8 from the CIR PCA. Removing jumper W1 enables the floating point instructions and removing jumper W8 enables the decimal instruction set. If the HP 32105A APL ROM's are installed on the EIS PCA, remove jumper W2 from the CIR PCA to enable the APL instructions. If the HP 32233A COBOL '74 ROM's are installed on the EIS PCA, remove jumper W4 to enable the COBOL '74 instructions.

TOP 1, 3, → 79
BOTTOM 2, 4, → 80
P1

1, 3, → 79
2, 4, → 80

S-BUS PCA

S3

S1 S2

J1
TOP 2, 4, → 50
BOTTOM 1, 3, → 49

J2
2, 4, → 50
1, 3, → 49

J3
2, 4, → 50
1, 3, → 49

MEMORY SIZE

(WORDS)
S3

1 - 128K
2 - 256K
3 - 384K
4 - 512K
5 - 768K
6 - 1024K
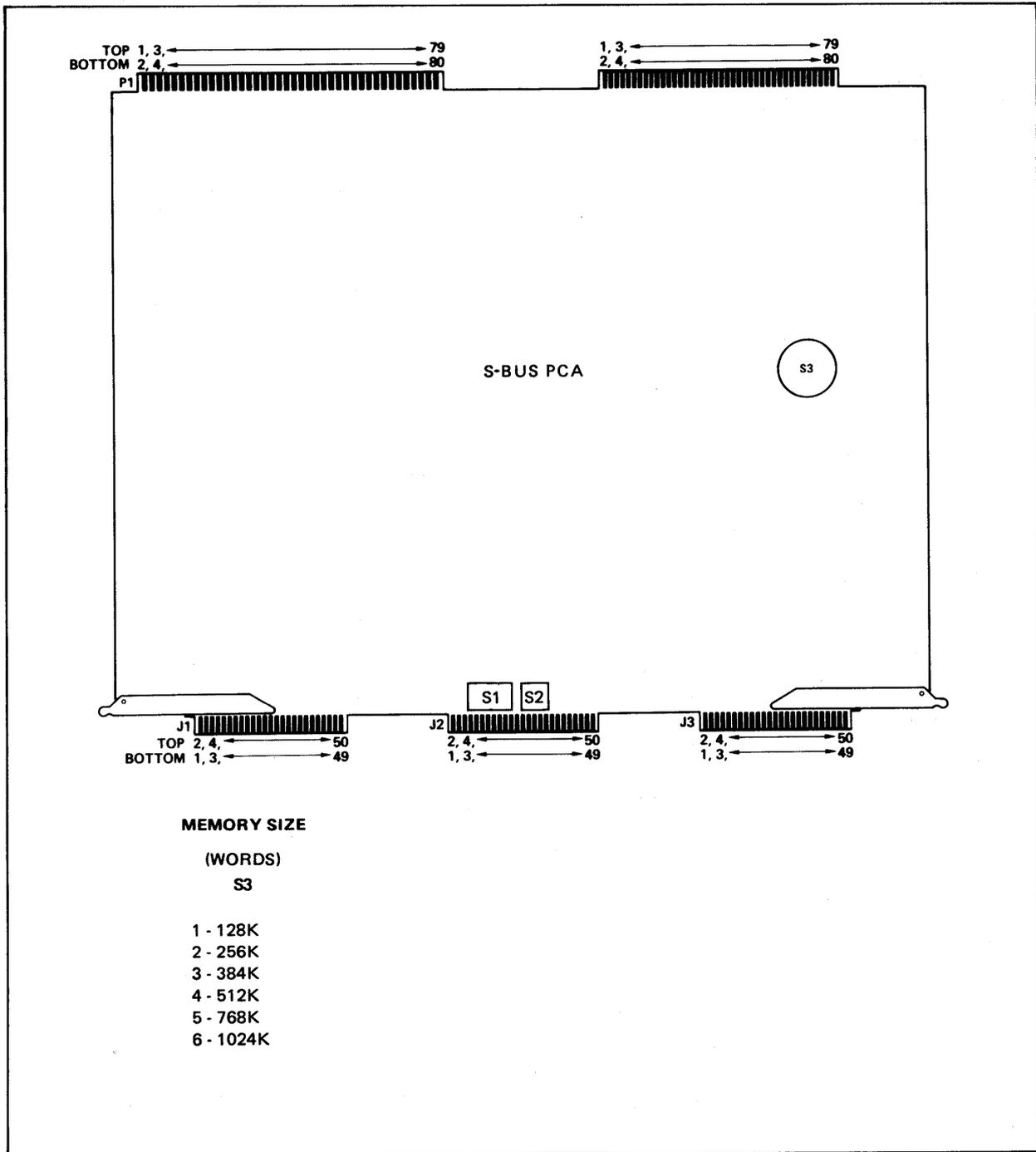
Figure 2-23. S-Bus PCA Switch Locations
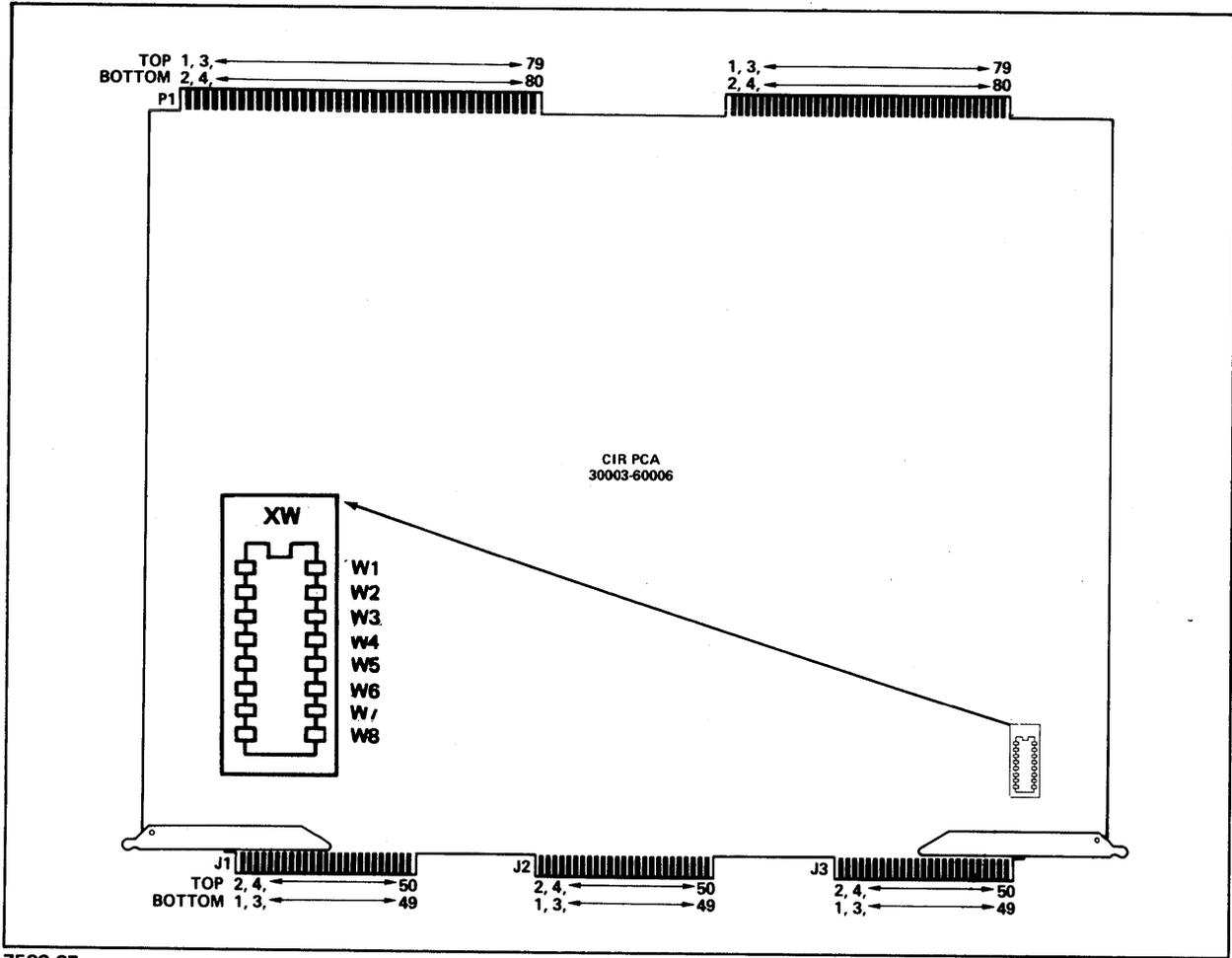
Figure 2-24. CIR PCA Jumper Locations

NOTES

NOTES

SYSTEM VERIFICATION AND TROUBLESHOOTING

This section contains a brief discussion of available verification procedures that can be used to determine if the computer system is operating properly, a brief discussion of system troubleshooting procedures, and a discussion of how to use the System Control Panel and the HP 30354A Maintenance Panel.

3-1. DIAGNOSTIC AND VERIFICATION PROGRAMS

The computer system uses three types of test programs; on-line verification programs, stand-alone diagnostic programs, and microdiagnostics.

3-2. On-Line Verification Programs

The on-line verification programs are used to confirm proper operation of peripheral devices (i.e., printer, terminals, readers, punches etc). These programs run concurrently with other programs under control of the Multiprogramming Executive Operating System (MPE) and permit uninterrupted system operation. If the minimum hardware configuration required for MPE is inoperable, on-line verification programs cannot be run and the stand-alone diagnostics must then be used. For detailed information on the use and functions of the on-line verification programs, refer to the individual on-line verification program manuals.

3-3. Stand-Alone Diagnostic Programs

The stand-alone diagnostic programs allow Customer Engineers to run maintenance and troubleshooting tests on system hardware and peripheral devices. Each of these programs is independently operated and runs directly on the central processor. MPE is not required and the operating system is shut-down while stand-alone programs are running. When a problem occurs that prevents the use of both on-line or stand-alone programs, then the microdiagnostics must be used. The stand-alone diagnostic tapes are created under control of SDUPII (Stand-Alone Diagnostic Utility Program II). Updating stand-alone diagnostics is also accomplished under control of SDUPII. For detailed information on the use and functions of SDUPII, refer to the Diagnostic Utility Program II Manual, part no. 03000-90125. For detailed information on the use and functions of stand-alone diagnostic programs, refer to the individual stand-alone diagnostic program manuals.

3-4. Microdiagnostics

The microdiagnostics are microprograms that are built into the system. These are microprograms that replace the instruction set microprograms in the central processor and in some controllers.

They identify problems by checking the hardware from the most basic level. The operating procedures for the built in micro-diagnostics are contained in the HP 3000 Computer System Install-ation Manual, part no. 30000-90147. The program is listed in the HP 3000 Series III Computer System Microprogram Listing, part no. 30000-90136.

3-5. SLEUTH 3000

SLEUTH 3000 is a stand-alone utility written in SPL/3000. It is designed to give Customer Engineers the capability of generating unique I/O test programs that run under the control of SLEUTH 3000. These programs allow isolation of I/O problems and ease the troubleshooting of these problems. SLEUTH 3000 has the abil-ity to run up to five different types of I/O devices concurrent-ly. It can also write and execute SIO programs, store and restore programs on magnetic tape, and edit the programs. Peripheral devices that do not have on-line and/or stand-alone verification programs require that SLEUTH 3000 programs be writ-ten to test these devices. For additional information, refer to the Stand-Alone SLEUTH Diagnostic D411A, manual part no. 03000-90123.

3-6. SYSTEM TROUBLESHOOTING AND REPAIR

The HP 3000 Series III CE Handbook, part no. 30000-90172 con-tains system troubleshooting procedures that are designed to iso-late malfunctions to specific functional areas of the system. Repair of a defective functional area is usually accomplished by replacing the defective PCA. Only the main memory PCA´s are re-paired to the component level.

In order to run the available diagnostic and verification pro-grams and to be able to perform the system troubleshooting pro-cedures, it is mandatory that Customer Engineers know how to use both the System Control Panel and the Maintenance Panel. De-tailed information on how to use these panels and "hands-on" ex-perience will be obtained while attending the hardware training course. For reference purposes, the panels are described in par-agraphs 3-7 and 3-8.

3-7. SYSTEM CONTROL PANEL

The System Control Panel (figure 3-1) is located at the top front of the CPU Equipment Bay and provides the switches and lamps re-quired to perform the following operations:

a. Cold load and run diagnostics.
b. Load and run user programs.
c. Halt programs.
d. System dump.
e. Observe Current Instruction Register.
f. Reset CPU.
g. Enable and disable auto restart function after power failure.
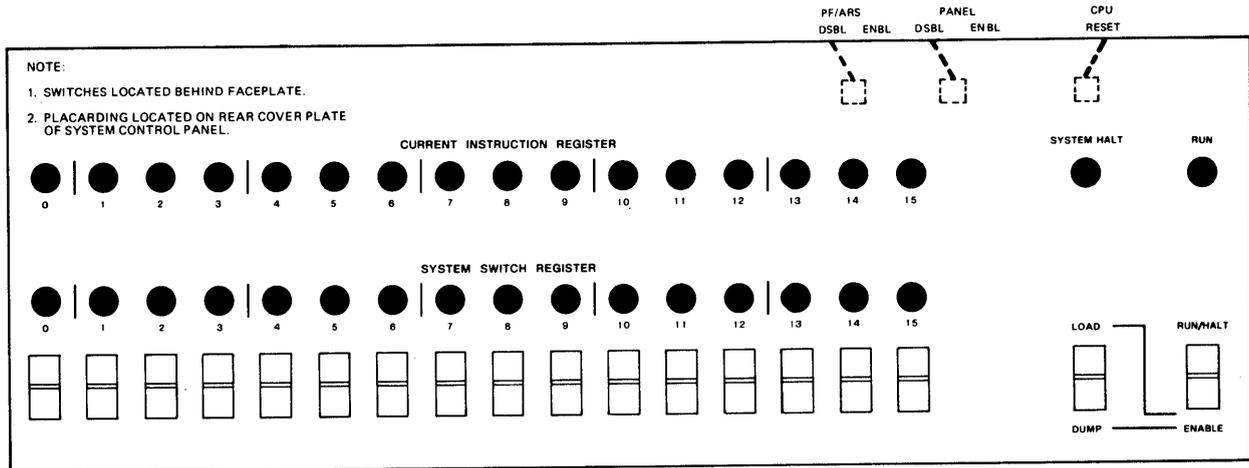
Figure 3-1.   System Control Panel

All front panel switches are three-position, spring-return, rock-er switches with a center-off position.   To perform a specific operation, press either the top or bottom half of the appropriate switch as indicated by the placarding.   When released, the switch will return to its center-off position.   The switches and lamps shown in Figure 3-1 are identified and described in table 3-1.

In addition to the switches described in table 3-1, there are three switches locted behind the upper-right corner of the panel that are accessible when the cabinet door is opened.   See figure 3-1.   The CPU RESET switch is a two-position, spring-return switch that resets the CPU circuits.   The PANEL DSBL/ENBL switch is a two-position switch that disables or enables the System Con-trol Panel for use.   The PF/ARS DSBL/ENBL switch is a two-posi-tion switch that disables or enables the auto restart program in the event of a power failure.

Table 3-1. System Control Panel Switches and Lamps

Panel Marking	Function
CURRENT INSTRUCTION REGISTER (lamps)	Displays the contents of the CIR.
SYSTEM SWITCH REGISTER (lamps and switches)	Displays the contents of the Switch Register as determined by pressing the switches on or off. Switches provide a 16-bit word to be used as a device number and control byte for cold load procedure.
RUN (lamp)	Indicates the system is executing a program.

Table 3-1. System Control Panel Switches and Lamps (Continued)

Panel Marking	Function
SYSTEM HALT (lamp)	Indicates a system halt caused by an irrecoverable error detected by the firmware.
RUN/HALT (switch)	Reverses the run/halt condition of the system.
ENABLE (switch)	Must be held in the ENABLE position to permit the LOAD or DUMP switch function to become active.
DUMP (switch)	Sends the contents of memory and CPU registers to the system magnetic tape unit.
LOAD (switch)	Used to load memory from a device specified by the SYSTEM SWITCH REGISTER contents.

3-8. MAINTENANCE PANEL

The Maintenance Panel (figure 3-2) is a troubleshooting aid for the computer system. When the panel is connected to the system, switches on the panel are used to select specific registers whose content may be observed or changed to assist in localizing system faults. Additionally, lamps on the panel show the contents of many computer registers and the state of principal signals, allowing analysis of system functioning. (For the most part, the visual displays are used only when the computer is halted.) Operating power is provided by the computer system. An interface PCA, installed in the CPU card cage, is required for the Maintenance Panel.

The names of switches and indicators on the Maintenance Panel are marked on an overlay which installs on the face of the unit. A smaller overlay (the I/O overlay) can be placed over a certain row of lamp names on the main overlay to extend the display function of those lamps. A switch permits display of the signals named on the small overlay; other displays remain unchanged. The small overlay can be turned over to provide another set of names; these signals are displayed by making an additional cable connection to the computer. The Maintenance Panel also has a self-test capability which allows the operability of most panel circuits to be verified without the use of test equipment.

(REF. ONLY)

CENTRAL DATA BUS

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

| MCUD PARITY | MCUD PE | DATA PE | SYSTEM PE | SYSTEM PARITY | TO 0 | TO 1 | TO 2 | FROM 0 | FROM 1 | FROM 2 | MOP 0 | MOP 1 | CPU LORQ | CPU HIRQ | CPU SELECT |

| READY 0 | READY 1 | READY 2 | READY 3 | READY 4 | READY 5 | CB14 | ENABLE 0 | ENABLE 1 | ENABLE 2 | ENABLE 3 | ENABLE 4 | CB15 | I/O LORQ | I/O HIRQ | I/O SELECT |

| FLAG 1 | FLAG 2 | FLAG 3 | TNAME 0 | TNAME 1 | SKIP | NOP 2 | REPEAT | ALU CARRY | ALU OVFL | CPU TIMER | OPND WAIT | INSTR WAIT | I/O WAIT | PANEL FREEZE | FREEZE |

| ICS FLAG | DISP FLAG | C MUX | NEXT +1 | INTRPT FF | JSB FLAG | NOP | BNDV | LUT GATE | INTRPT GATE | U GATE | RAR GATE | SAVE GATE | JUMP 1 GATE | JUMP 2 GATE | JUMP FREEZE |

| DIRECT ACTIVE | SERVICE OUT | SERVICE IN | DATA POLL | SIO ACTIVE | INBOUND | DRT REQ | JUMP | IOB ENABLE | DRT STORE | I/O TIMER | XFER ERROR | INTRPT REQ | INTRPT POLL | INTRPT ACK | EXT INTRPT |

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |

S		STORE		FUNCTION		SKIP		SHIFT		SPECIAL		R
				JMP. JSB					JUMP TARGET		
				ANY ROM				ROM CONSTANT		

V BUS

| B12 | B13 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

V BUS COMPARE REGISTER

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

V BUS JUMP REGISTER

| B12 | B13 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

V BUS				SINGLE CYCLE REGISTER DISPLAY		TIMERS	ERROR FREEZE	INTRPT	IOP SINGLE STEP		RESET		CLOCK	
COMPARE														
ENABLE	HALT	HALT EXIT	EXECUTE JUMP	REGISTER	ALT	INHIBIT	ENABLE	INHIBIT	ENABLE	EXECUTE	CPU	I/O	EXT INT	INHIBIT FREE RUN	SINGLE CYCLE
INHIBIT	JUMP			U BUS	NORMAL	ENABLE	INHIBIT	ENABLE	INHIBIT						

REGISTER SELECTION

| SP2 | SP3 | OPND | PADD | CPX1 | CPX2 | SR | MOD NO. | IDN | CMD DEV NO. | IOD | IOP | I/O DATA | I/O MAP | PCLK | TEST | R RGTR | S RGTR |
| MEM ADRS | MEM DATA | STATUS | X | PB | P | PL | DL | DB | Q | S | Z | RA | RB | RC | RD | CNTR | CIR |

REGISTER DISPLAY

| B14 | B15 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

SWITCH REGISTER

| B14 | B15 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

| | | | | | | | | BKPT HALT | SYSTEM HALT | RUN |

LOAD REGISTER	ADDRESS CONTROL	MEMORY	BREAKPOINT	EXECUTE			
	DECR ENABLE / INCR ENABLE		READ ENABLE / WRITE ENABLE				
FROM SW RGTR / MEM ADRS FROM DSPL	INHIBIT / INHIBIT	DISPLAY / STORE	INHIBIT / INHIBIT	SW RGTR / SINGLE INSTR	SYSTEM RESET	LOAD	RUN/HALT

Figure 3-2.   Maintenance Panel

3-9. Switch/Lamp Identification and Description

Figure 3-2 illustrates Maintenance Panel switches and lamps. The shaded numbers on the right side of figure 3-2 identify the row number of lamps or switches and are used in table 3-2 as an aid to locating the switch or lamp. The I/O overlay is shown in figure 3-3. When referring to a switch or lamp, this manual uses the name physically marked on the equipment. The name is quoted in capital letters to indicate it is an equipment marking. There are three types of switches on the panel as follows:

a. Bistable switches. These switches have two positions, and can remain in either the up or the down position. In the down position they have no effect on normal computer functioning, and they are left in this position except when their particular function is required. In figure 3-2, the bistable switches can be identified by the fact that they are in the down position.

b. Two-position spring-return switches. These switches are pressed down when their function is required. When released, they return to the up position.

c. Three-position spring-return switches. These switches have a center-off position. They are pressed up or down to produce the desired function. When released, they return to the center position. All switches of this type are in row 12 of figure 3-2.

Lamps which display register contents are lighted when the particular position of the register contains a binary 1. Lamps which display the state of a signal are lighted when the signal is in the asserted state. That is, a lamp is lighted when a "not" signal is low; for other signals, a lamp is lighted when the signal is high.

3-10. Operating Precautions

The operating system, if in use, may cause unexpected changes in computer functioning when the Maintenance Panel switches are actuated. These unexpected changes result from such factors as stack overflow, etc. Therefore, the operator should be thoroughly familiar with the operating system before attempting to use the Maintenance Panel.

Table 3-2. Maintenance Panel Switches and Lamps

Panel Row	Panel Marking	Use
1	CENTRAL DATA BUS 0 through 15 (lamps)	These lamps display the data word which is on the CTL-Bus (MCUD 0:15).
2	MCUD PARITY (lamp)	Indicates the state of the CTL-Bus parity bit.
2	MCUP PE (lamp)	Indicates a parity error has been detected on the CTL-Bus.
2	DATA PE (lamp)	Indicates there was a parity error detected in the data received by the CPU from memory.
2	SYSTEM PE (lamp)	Indicates a parity error was detected in the information transferred on the TO, FROM, MOP, and SYSTEM PARITY lines.
2	SYSTEM PARITY (lamp)	Indicates the state of the parity bit generated from the TO, FROM, and MOP codes.
2	TO 0, TO 1, TO 2 (lamps)	Display the address for which the word on the CTL-Bus is intended.
2	FROM 0, FROM 1 FROM 2 (lamps)	Display the address of the module from which the word on the CTL-Bus is being sent.
2	MOP 0, MOP 1 (lamps)	Display the memory operation code. This code is used by the addressed memory module.
2	CPU LORQ (lamp)	Indicates the CPU is issuing a low priority request for a transfer to a module.
2	CPU HIRQ (lamp)	Indicates the CPU is issuing a high priority request for use of the CTL-Bus.
2	CPU SELECT (lamp)	Indicates the CPU is currently selected to use the CTL-Bus.
3	READY 0 through 5 (lamps)	Display the module ready lines. Each line is associated with a like numbered module and, when true, indicates the module is ready to receive a transfer.

Table 3-2. Maintenance Panel Switches and Lamps (Continued)

Panel Row	Panel Marking	Use
3	ENABLE 0 through 4 (lamps)	Display the module enable lines. Each line is associated with a like numbered module and, when true, indicates a module is transferring data. The enable lines are monitored by the modules to resolve priorities.
3	CB14, CB15 (lamps)	Display MSBs of address on CTL-Bus.
3	I/O LORQ (lamp)	Indicates the IOP is issuing a low priority request for use of the CTL-Bus.
3	I/O HIRQ (lamp)	Indicates the IOP is issuing a high priority request for use of the CTL-Bus.
3	I/O SELECT (lamp)	Indicates the IOP is selected to use the CTL-Bus.
4	FLAG 1, FLAG 2 FLAG 3 (lamps)	Indicate the states of the three Flag flip-flops controlled primarily by the Special Field microinstructions.
4	TNAME 0, TNAME 1 (lamps)	Indicate the states of the TOS namer bits. These bits specify the mapping between the TOS registers RA, RB, RC, RD, and the associated physical registers.
4	SKIP (lamp)	Indicates a skip condition is met during the current clock cycle.
4	NOP 2 (lamp)	Indicates the state of the NOP 2 bit. When true, causes a "no operation" by Rank 2 of the ROM Output Register.
4	REPEAT (lamp)	Indicates the Repeat bit is true, causing the microporcessor to repeat the current microinstruction until the skip condition is met.
4	ALU CARRY (lamp)	Indicates the carry signal from the microprocessor ALU is true.
4	ALU OVFL (lamp)	Indicates the overflow signal from the microprocessor is true.

Table 3-2. Maintenance Panel Switches and Lamps (Continued)

Panel Row	Panel Marking	Use
4	CPU TIMER (lamp)	Indicates that a module did not respond to the CPU within a specified time.
4	OPND WAIT (lamp)	Indicates the CPU is waiting for an operand from memory.
4	INSTR WAIT (lamp)	Indicates the CPU is waiting for an instruction from memory.
4	I/O WAIT (lamp)	Indicates a multiplexed I/O operation is fetching a word from memory.
4	PANEL FREEZE (lamp)	Lighted when a microprogram halt is in effect or when the V-Bus carries the same number as the V BUS COMPARE REGISTER switches.
4	FREEZE (lamp)	Indicates the CPU freezable clock has stopped.
5	ICS FLAG (lamp)	Indicates the Interrupt Stack Flag flip-flop is set.
5	DISP FLAG (lamp)	Indicates the dispatcher is executing.
5	C MUX (lamp)	Indicates which instruction is being decoded. Lighted = current instruction; off = next instruction.
5	NEXT + 1 (lamp)	Indicates the sequence is in the "next sequence" state.
5	INTRPT FF (lamp)	Indicates the state of the Interrupt flip-flop. When lighted, an external or internal interrupt is pending.
5	JSB FLAG (lamp)	Indicates the microcode is executing a subroutine.
5	NOP (lamp)	Indicates one of the normal inputs of the V-Bus are turned off caused by any one of the following: CPU Reset PWR ON UGATE on (RAR in the Store Field)

Table 3-2. Maintenance Panel Switches and Lamps (Continued)

Panel Row	Panel Marking	Use
5	NOP (lamp) (Cont)	INTG on (A CPU interrupt is forcing the V-Bus to address 3). The execution of a V-Bus jump with panel switches.
5	BNDV (lamp)	Indicates a memory instruction references an address outside the limit registers.
5	LUT GATE (lamp)	Indicates an instruction target address is being sent to the V-Bus.
5	INTRPT GATE (lamp)	Indicates when doing a NEXT + 1 cycle and a microcode interrupt is pending or when a bounds violation has been detected.
5	U GATE (lamp)	Indicates the U-Bus is gated onto the V-Bus.
5	RAR GATE (lamp)	Indicates the current address + 1 is put on the V-Bus.
5	SAVE GATE (lamp)	Indicates the microcode return address is being gated onto the V-Bus.
5	JUMP 1 GATE (lamp)	Indicates the jump target from Rank 1 is being gated onto the V-Bus.
5	JUMP 2 GATE (lamp)	Indicates the jump target from Rank 2 is being gated onto the V-Bus.
5	JUMP FREEZE (lamp)	Indicates a one cycle freeze is taking place to allow a new V-Bus address.
6	DIRECT ACTIVE (lamp)	Indicates the IOP is sending out a direct I/O command.
6	SERVICE OUT (lamp)	Indicates to device controller: For direct commands; the command code, device address, and data on on the bus are valid. For SIO transfers inbound; data on the bus is anticipated. For SIO transfers outbound; data on bus is valid.

Table 3-2. Maintenance Panel Switches and Lamps (Continued)

Panel Row	Panel Marking	Use
6	SERVICE IN (lamp)	Indicates a response to an IOP Service Out or Data Poll.
6	DATA POLL (lamp)	Indicates the IOP has received a request for a transfer to or from memory.
6	SIO ACTIVE (lamp)	Indicates a multiplexed I/O operation is in progress.
6	INBOUND (lamp)	Indicates the IOP is executing an in-bound memory transfer.
6	DRT REQ (lamp)	Indicates the IOP has received a request from an SIO multiplexer to fetch a DRT entry.
6	JUMP (lamp)	Indicates the IOP is currently updating the DRT pointer during the execution of a jump order.
6	IOB ENABLE (lamp)	Indicates the outbound data is on the IOP Bus.
6	DRT STORE (lamp)	Indicates the IOP is updating the DRT pointer.
6	I/O TIMER (lamp)	Indicates the Service In signal has failed to occur within a period of time after a Service Out signal.
6	XFER ERROR (lamp)	Indicates an I/O data parity exists.
6	INTRPT REQ (lamp)	Indicates the state of the Int Req line from the devices.
6	INTRPT POLL (lamp)	Indicates the state of the Int Poll signal from the IOP to the devices.
6	INTRPT ACK (lamp)	Indicates that an Interrupt Acknowledge signal has been received in response to an interrupt poll.
6	EXT INTRPT (lamp)	Indicates an external interrupt has been acknowledged by the IOP.

Table 3-2. Maintenance Panel Switches and Lamps (Continued)

Panel Row	Panel Marking	Use
7	ROM 0 through 31 (lamps)	Displays the contents of the ROM output registers. Lamps (0:4) and (28:31) display ROR1. Lamps (5:27) display ROR2.
7	B12, B13 (lamps)	Display two MSB bank bits.
8	V BUS 2 through 15 (lamps)	Display the address of the ROM data currently being accessed. Since the ROM is two levels removed from the actual microinstruction being executed out of ROR2, the address is normally two ahead of the address being executed.
9	V BUS COMPARE REGISTER 0 through 15 (bistable switches)	These switches specify the microprogram address at which a V TRIG pulse will be supplied. (The pulse is available at test point E3 at the front of the MPI PCA. It is also available at J3, pin 3 on the MPI PCA.) These switches also specify a microprogram jump address or halt address when the V BUS COMPARE ENABLE/INHIBIT switch is at ENABLE. The V TRIG pulse or breakpoint halt takes place at the completion of a particular clock cycle. To bring about the effect at the desired clock cycle, the microinstruction address set into the V BUS COMPARE REGISTER switches should be as follows: Address +1 for completion of execution of the R-Bus and S-Bus fields. Address +2 for completion of execution of the remaining microinstruction fields.
10	B12, B13 (bistable switches)	Used to set bank bits B12 and B13.
10	V BUS JUMP REGISTER 2 through 15	These switches specify the jump target for:

Table 3-2. Maintenance Panel Switches and Lamps (Continued)

Panel Row	Panel Marking	Use
	(bistable switches)	Jump resulting from the V-Bus contents being equal to the contents of the V BUS COMPARE REGISTER switches. Jump resulting from the V BUS EXECUTE JUMP switch being pressed.
11	V BUS COMPARE ENABLE/INHIBIT (bistable switch)	Enables the V BUS COMPARE HALT/JUMP switch.
11	V BUS COMPARE HALT/JUMP (bistable switch)	When enabled by the switch listed above, selects halt or jump when the V-Bus contents are the same as the V BUS COMPARE REGISTER switches.
11	V BUS COMPARE HALT EXIT (spring-return switch)	When pressed, starts the microprogram after: A halt brought about by the V BUS COMPARE REGISTER switches. A freeze brought about a CCPX-14 Special field microinstruction.
11	V BUS EXECUTE JUMP (spring-return switch)	When this switch is pressed, the microprogram jumps to the address in the V BUS JUMP REGISTER switches. This function should be used only when the computer is halted.
11	SINGLE CYCLE REGISTER DISPLAY, REGISTER/U BUS (bistable switch)	Selects the REGISTER DISPLAY lamp readout as follows: With the switch at the REGISTER position, the display is identified by the lighted REGISTER SELECTION lamp. With the switch at the U BUS position the U-Bus is displayed. For this function, the CLOCK INHIBIT/ FREE RUN switch (panel row 11) must be at INHIT (single cycle operation).

Table 3-2. Maintenance Panel Switches and Lamps (Continued)

Panel Row	Panel Marking	Use
11	SINGLE CYCLE REGISTER DISPLAY, ALT/ NORMAL (bistable switch)	With this switch at the NORMAL position, RA, RB, RC, and RD on the S-Bus PCA can be displayed by the REGISTER DISPLAY lamps. With the switch at ALT, RA, RB, RC, and RD on the R-Bus PCA can be displayed. Also, with the switch at NORMAL, SP1 and Pre-adder are displayed from the S-Bus; with the switch at ALT, SP1 and the Pre-adder are displayed from the R-Bus. This switch must be at NORMAL to store into RA, RB, RC, RD, or SP1 from the Maintenance Panel.
11	TIMERS (bi- stable switch)	Enables or disables the CPU, memory, IOP, and Selector Channel timers.
11	ERROR FREEZE (bistable switch)	In the ENABLE position, this switch causes a freeze when any of the following occurs: Illegal memory address Memory address parity error MCUD parity error System parity error I/O data parity error I/O address parity error To end the freeze, the ERROR FREEZE switch is set to the down position.
11	INTRPT (bi- stable switch)	When the computer is running, setting this switch to INHIBIT causes all internal and external interrupts to be ignored, with the exception of the power fail interrupt. When the switch is returned to the ENABLE position, the previously ignored interrupts are processed. The switch performs no function when the computer is halted.
11	IOP SINGLE STEP ENABLE/ INHIBIT (bistable switch)	Enables or disables the IOP SINGLE STEP EXECUTE switch.

Table 3-2. Maintenance Panel Switches and Lamps (Continued)

Panel Row	Panel Marking	Use
11	IOP SINGLE STEP EXECUTE (spring-return switch)	When enabled by the IOP SINGLE STEP ENABLE/INHIBIT switch, the IOP executes one step each time this switch is used.
11	RESET CPU (spring-return switch)	The CPU and MCU are reset when this switch is pressed. To avoid improperly changing the contents of registers, the switch should be pressed only when the computer is halted.
11	RESET I/O (spring-return switch)	All I/O subsystems are reset when this switch is pressed.
11	CLOCK EXT/INT (bistable switch)	At the INT position, this switch allows the CPU to use the clock pulse generated within the CPU. At the EXT position, the switch selects a clock pulse produced by an external pulse generator.
		The external clock pulse must have the following characteristics:
		Source impedance: 50 ohms or less
		Source must sink up to 60 ma.
		High level: +2.5V to +5.0V
		Low level: 0.0V to +0.4V
		Maximum rise time: 10 nsec
		Maximum fall time: 10 nsec
		High time: 20 nsec to infinite time
		Low time: 20 nsec to infinite time
		Maximum frequency: 25.0 MHz
		Minimum frequency: 0 Hz
		To equal the internal clock-pulse rate, the external clock-pulse frequency must be 22.8571 MHz. This corresponds to a period of 43.75 nsec, which, because of a divide-by-four action in the CPU, provides a 175-nsec computer clock cycle.
		The external clock pulse is supplied to a BNC-type connector on the CPU backplane. The connector is labeled EXT - CLOCK. A 50-ohm termination impedance is provided in the CPU.

Table 3-2. Maintenance Panel Switches and Lamps (Continued)

Panel Row	Panel Marking	Use
11	CLOCK INHIBIT/ FREE RUN (bistable switch)	In the INHIBIT position, this switch permits the CPU to execute one machine cycle each time the CLOCK SINGLE CYCLE switch is pressed. In the FREE RUN position, the CPU operates continuously using either internal or external clock pulses.
11	CLOCK SINGLE CYCLE (spring-return switch)	When enabled by the CLOCK INHIBIT/FREE RUN switch, pressing this switch causes execution of one CPU machine cycle.
12	REGISTER SELECTION (lamps and center-off spring-return switches)	When pressed up or down, each switch lights the lamp above or below it, and turns off any other lighted lamp in the group. The lighted lamp identifies the register displayed by the REGISTER DIS-PLAY lamps. The B14 and B15 lamps remain extinguished except as stated below.

The lighted REGISTER SELECTION lamp also identifies the register which will be loaded by either of the LOAD REGISTER switches.

The following registers cannot be loaded by the LOAD REGISTER FROM SW RGTR switch; these are identified below as "display only" registers. Special comments are as follows:

    OPND, display only.
    PADD, display only.
    CPX1, display only.
    CPX2, display only.
    SR, display only

    MOD NO., display only. The module number appears in positions 5, 6, and 7 of the REGISTER DISPLAY lamps. REGISTER DISPLAY lamp 13 is lighted indicating CPU No. 1.

    IDN, display only. |

Table 3-2. Maintenance Panel Switches and Lamps (Continued)

Panel Row	Panel Marking	Use
		CMD DEV NO., display only. The I/O command is displayed in positions 5 through 7; the device number is in positions 8 through 15. Positions 0 through 4 of the display do not light
		IOD, display only. Displays (via the S-Bus) the contents of the IOD Input Register.
		IOP, display only. Displays the contents of the Data-In Register.
		I/O DATA, display only. Displays the data on the IOD (0:15) bits. On the IOP bus, these bits are in "not" form; however, the "not" bits are inverted before display. Thus, there is no overbar over the mnemonic.
		I/O MAP, display only. To identify the I/O Map bits, the I/O overlay (part no. 30354-80012) is placed over the REGISTER DISPLAY lamps.
		PCLK, display only.
		TEST, display only. Displays any 16 bits applied to J3 on the Maintenance Panel Interface PCA.
		R RGTR, display only.
		S RGTR, display only.
		MEM ADRS displays the memory address (SPO) in REGISTER DISPLAY (0:15), and the ABS-Bank Register in B14 and B15. This lamp is called MEM ADRS because SPO contains the address when memory is accessed by means of the MEMORY STORE or MEMORY DISPLAY switches.
		MEM DATA displays the memory data (SP1) in REGISTER DISPLAY (0:15). B14 and B15 will be zero. This lamp is called MEM DATA because SP1 re-

Table 3-2. Maintenance Panel Switches and Lamps (Continued)

Panel Row	Panel Marking	Use
		ceives the data when memory is displayed by means of the MEMORY DISPLAY switch.
		SP2 displays the contents of the Scratch Pad 2 Register, used by the microcode. B14 and B15 will be zero.
		SP3 displays the contents of the Scratch Pad 3 Register, used by the microcode. B14 and B15 will be zero.
		RA displays the contents of the TOS register. B14 and B15 will be zero.
		RB displays the contents of the 2nd stack register. B14 and B15 will zero.
		RC displays the contents of the 3rd stack register. B14 and B15 will be zero.
		RD displays the contents of the 4th stack register. B14 and B15 will be zero.
		STATUS displays the CPU Status Register.
		PB displays the PB Status Register in REGISTER DISPLAY (0:15). B14 and B15 display the PB-Bank Register.
		P displays the P Register in REGISTER DISPLAY (0:15). B14 and B15 display the PB-Bank Register.
		PL displays the PL Register in REGISTER DISPLAY (0:15). B14 and B15 display the PB-Bank Register; these two bits are display only.
		DL displays the contents of the DL Register in REGISTER DISPLAY (0:15). B14 and B15 display the stack register; these two bits are display only.

Table 3-2. Maintenance Panel Switches and Lamps (Continued)

Panel Row	Panel Marking	Use
		DB displays the Data Base Register in REGISTER DISPLAY (0:15). B14 and B15 display the DB-Bank Register. Q displays the Q Register in REGISTER DISPLAY (0:15). B14 and B15 display the Stack-Bank Register; these two bits are display only. S displays the S Register in REGISTER DISPLAY (0:15). B14 and B15 display the Stack-Bank Register; these two bits are display only. Z displays the Z Register in REGISTER DISPLAY (0:15). B14 and B15 display the Stack-Bank Register. CNTR, display bits 10:15. CIR, display only. Display valid only when the CLOCK INHIBIT/FREE RUN switch is at INHIBIT position (single cycle operation).
13	REGISTER DISPLAY 0 through 15, B15, and B14 (lamps)	Display the register identified by the lighted REGISTER SELECTION lamp. The REGISTER DISPLAY lamps also indicate the data which will be loaded by the LOAD REGISTER, MEM ADDRS FROM DISPL switch.
14	SWITCH REGISTER 0 through 15, B15, and B14 (bistable switches)	Switches 0 through 15 provide a 16-bit word to: Load in a selected register by means of the LOAD REGISTER FROM SWITCH RGTR switch. Store in memory by means of the MEMORY STORE switch. Use as an instruction word when the EXECUTE SW RGTR switch is pressed. Match with a word read from memory to cause a read breakpoint halt (using the BREAKPOINT READ ENABLE switch).

Table 3-2. Maintenance Panel Switches and Lamps (Continued)

Panel Row	Panel Marking	Use

		Match with a word stored in a memory storage operation to cause a store breakpoint halt (using the BREAKPOINT STORE ENABLE switch).
		
		Switches B14 and B15 (row 14) and B12 and B13 (row 10) are used to change the contents of the following bank registers by means of the LOAD REGISTER FROM SW RGTR switch:
		
		  ABS-Bank Register (MEM ADRS Register selected)
		
		  PB-Bank Register (PB Register selected)
		
		  Stack-Bank Register (Z Register selected)
		
		  DB-Bank Register (DB Register selected)
		
		Except as listed above, switches B14 and B15 produce no effect when registers are manually loaded.
		
		A further use of switches B14 and B15 is to specify the memory module number for breakpoint halts (using the BREAKPOINT STORE ENABLE/INHIBIT switch).

15	BKPT HALT (lamp)	Lighted during a breakpoint halt caused by either of the following switches:
		
		  BREAKPOINT READ ENABLE/INHIBIT
		
		  BREAKPOINT STORE ENABLE/INHIBIT

15	SYSTEM HALT (lamp)	Lighted during a system halt (caused by an irrecoverable error).
15	RUN (lamp)	Lighted when the CPU is running.
16	LOAD REGISTER FROM SW RGTR (spring-return	When pressed, the register indicated by the lighted REGISTER SELECTION lamp is loaded with the contents of the SWITCH

Table 3-2. Maintenance Panel Switches and Lamps (Continued)

Panel Row	Panel Marking	Use
	switch)	REGISTER switches. The bank registers may also be loaded, as explained in the SWITCH REGISTER description above.
16	LOAD REGISTER, MEM ADDRS FROM DISPL (spring-return switch)	When pressed, the Memory Address Register (SPO) is loaded with the bits displayed by the REGISTER DISPLAY (0:15) lamps. Also, the ABS-Bank Register is loaded with the bits displayed by the REGISTER DISPLAY B14 and B15 lamps. The MEMORY DISPLAY or MEMORY STORE switch can then display or store at the 18-bit address.
16	ADDRESS CONTROL, DECR ENABLE/INHIBIT (bistable switch)	When this switch is at ENABLE, the Memory Address Register (SPO) is decremented by 1 each time the MEMORY DISPLAY or MEMORY STORE switch is pressed. (The ADDRESS CONTROL INCR ENABLE/INHIBIT switch should be at INHIBIT.
16	ADDRESS CONTROL INCR ENABLE/INHIBIT (bistable switch)	When this switch is at ENABLE, the Memory Address Register (SPO) is incremented by 1 each time the MEMORY DISPLAY or MEMORY STORE switch is pressed. (The ADDRESS CONTROL DECR ENABLE/INHIBIT switch should be at INHIBIT.
16	MEMORY DISPLAY (spring-return switch)	When this switch is pressed, the following takes place: The REGISTER SELECTION MEM DATA lamp lights. Any other lighted lamp in this group goes out. The REGISTER DISPLAY (0:15) lamps show the contents of the memory address specified by the ABS-Bank Register and the Memory Address Register (SPO). Lamps B14 and B15 remain extinguished. The Memory Address Register (SPO) is incremented or decremented by 1 if one of the ADDRESS CONTROL switches is at ENABLE. A carry from SPO does not enter the ABS-Bank Register.

Table 3-2. Maintenance Panel Switches and Lamps (Continued)

Panel Row	Panel Marking	Use
16	MEMORY STORE (spring-return switch)	When this switch is pressed, the following takes place. The contents of the SWITCH REGISTER (0:15) switches are stored in memory at the address indicated by the ABS-Bank Register and the Memory Address Register (SPO). The Memory Address Register (SPO) is incremented or decremented by 1 if one of the ADDRESS CONTROL switches is at ENABLE. A carry from SPO does not enter the ABS-Bank Register.
16	BREAKPOINT READ ENABLE/ INHIBIT (bistable switch)	In the ENABLE position, this switch halts the CPU when a 16-bit word read from memory is the same as the word in the SWITCH REGISTER (0:15) switches.
16	BREAKPOINT WRITE ENABLE/ INHIBIT (bistable switch)	In the ENABLE position, this switch halts the CPU when memory storage takes place at an address which is in the SWITCH REGISTER B14, B15, and (0:15) switches.
16	EXECUTE SW RGTR (spring-return switch)	When this switch is pressed, the CPU executes the instruction in the SWITCH REGISTER (0:15) switches. The P Register contents do not change when the instruction is performed unless the instruction is a branch type; in this case, the P Register receives the target address if the branch condition is met. If a stack operation is performed, stack operation B should be NOP. (That is, positions 10 through 15 of the SWITCH REGISTER must contain zeros.
16	EXECUTE SINGLE INSTR (spring-return switch)	When this switch is pressed, the instruction indicated by the P Register is executed. The P Register is incremented by 1, or it is loaded with a target address if the instruction is a branch type and the branch condition is met.

Table 3-2. Maintenance Panel Switches and Lamps (Continued)

Panel Row	Panel Marking	Use
16	EXECUTE SINGLE INSTR (Cont)	If the instruction is of the double stack type, the switch must be pressed twice to execute both halves of the instruction. The first depression causes execution of stack operation A. The second depression causes execution of stack operation B. If operation B is other than NOP, bit 3 of the CPU Status Register is set to 1 during stack operation A. This bit is cleared at the start of stack operation B. The P Register is incremented during stack operation B unless B is NOP, in which case it is incremented during A.
16	SYSTEM RESET (spring-return switch)	When pressed, this switch resets the CPU and the I/O system.
16	LOAD (spring-return switch)	Stores in memory from an I/O device specified in the SWITCH REGISTER switches.
16	RUN HALT (spring-return switch)	Pressing this switch halts the CPU if it is running or starts the computer if it is halted. Both halves of a stack op instruction are always completed before the halt takes place. If the CPU halts while an SIO operation is in progress, the SIO operation continues to its normal completion.

3-11. Preparation for Use

To use the Maintenance Panel, make preparation as follows:

a. Ensure that the computer system is not in use.

b. Inform potential users that the system is unavailable until further notice.

c. For HP 32421A Series III systems, set the SYSTEM DC POWER switch (inside top of CPU equipment bay door) to STANDBY. For HP 32435A Series III systems, set the DC POWER switch (on Power Control and Display Panel) to DISABLE.
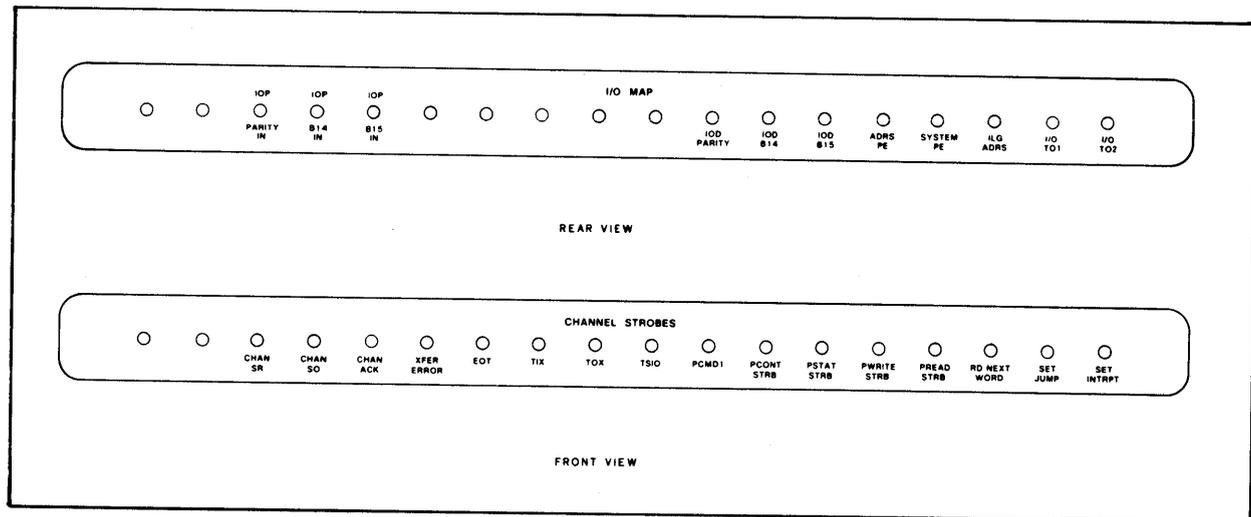
Figure 3-3.   Maintenance Panel I/O Overlay

d.   Install Maintenance Panel Interface PCA, part no. 30354-60003 in CPU equipment bay card cage slot 1A1.

e.   At the front of the MPI PCA, set switch S1 to the NORMAL position.   (This switch is labelled NORMAL/LAMP TEST/SWITCH TEST.)

f.   Make the connections shown in figure 3-4.

g.   If required, install the 30354-80012 I/O overlay over the REGISTER DISPLAY lamps. If using the CHANNEL STROBES side of this overlay, connect jack J3 on the MPI PCA to J1 on the Selector Channel Maintenance PCA.

h.   Set all Maintenance Panel bistable switches to the down position.

i.   Reapply DC power to the computer system.

3-12. General Operating Method

When using the Maintenance Panel, operate switches to achieve the desired results, observing lamp indications. Refer as necessary to table 3-2 and the applicable program documentation. To prevent unauthorized use of the Maintenance Panel, set switch S1 on the MPI PCA to the SWITCH TEST position. This prevents Maintenance Panel switch information from entering the computer system. Note that if a power failure occurs with S1 at SWITCH TEST, the switch must be returned to the NORMAL position to permit initialization of the computer system for restart after power is restored.
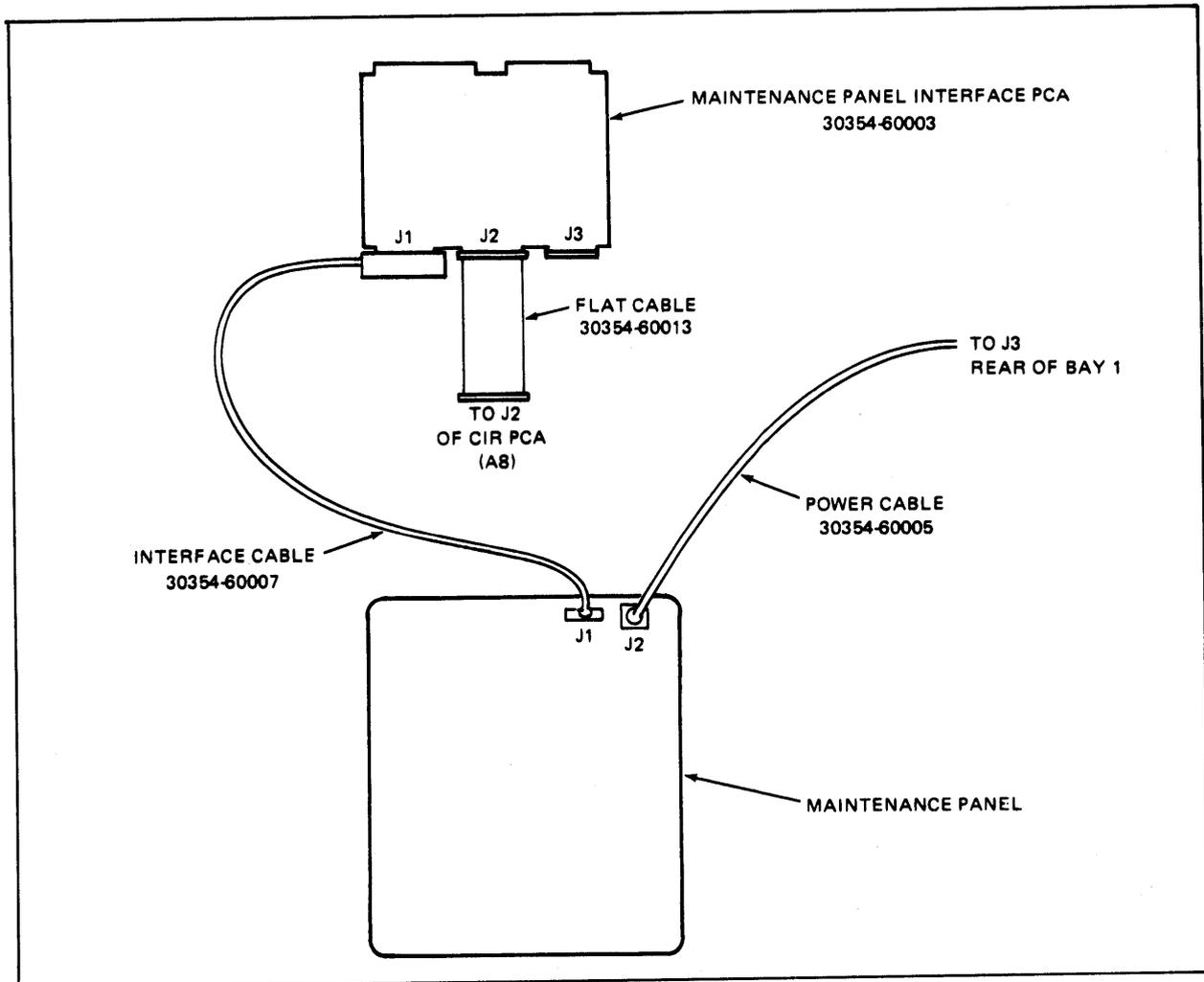
Figure 3-4. Maintenance Panel Operating Connections

3-13. Using Maintenance Panel and System Control Panel

When the Maintenance Panel is in use, switches and indicators on the System Control Panel (figure 3-1) function in the normal manner. The following points apply:

a. The CPU can be started or halted either by the RUN-HALT switch on the Maintenance Panel or by the RUN/HALT switch on the System Control Panel.

b. If the LOAD switch on the System Control Panel is used, the appropriate information must be set into the SWITCH REGISTER on the System Control Panel. Similarly, use of the LOAD switch on the Maintenance Panel requires that the SWITCH REGISTER on the Maintenance Panel be used.

c. The RSW instruction acquires the 16-bit word which is in the SWITCH REGISTER on the System Control Panel, as in normal operation.

3-14. Stack Register Loading

Because of the queue-down function which occurs when the CPU halts, stack registers must be loaded at their location in memory. For test purposes it is possible to load the stack registers which are on the S-Bus PCA and R-Bus PCA. However, when queue-up takes place after the CPU is started, the register contents will be destroyed.

3-15. CPU Register Displays

When CPU registers are displayed, the register contents are acquired either from the computer S-Bus or R-Bus. A switch on the Maintenance Panel permits display from either the S-Bus or the R-Bus. This switch is titled SINGLE CYCLE REGISTER DISPLAY, ALT/NORMAL (panel row 11).

3-16. General-Use Display

When the REGISTER SELECTION TEST lamp is lighted, any 16 bits or 16 binary signals can be applied for display to jack J3 of the MPI PCA (figure 3-4). The bits or signals must have TTL silicon logic levels. They are displayed in the REGISTER DISPLAY (0:15) lamps on the Maintenance Panel. The pins in J3 to which connection is made are listed in table 3-3. The table also shows two trigger pulses and a +5 volt source. These are supplied by the MPI PCA. The MR signal is an input to the PCA that is used for factory test purposes. For any particular application of the general-use display, a cardboard overlay can be made for the REGISTER DISPLAY lamps with appropriate signal names marked on the overlay. (A ticket punch is a suitable device for making the holes in the cardboard.)

3-17. Maintenance Panel Test

The following tests check the operability of most circuits in the Maintenance Panel. Most circuits in the MPI PCA are not checked. If performed as described, and if step a in paragrah 3-18 has previously been completed, the test can be executed without interfering with normal computer functioning. (Switch information from the Maintenance Panel does not enter the computer system when switch S1 on the MPI PCA is at the SWITCH TEST position.)

3-18. LAMP TEST. Perform the lamp test as follows:

a. Connect the Maintenance Panel to the computer system in the normal manner (figure 3-4). Before making connections be sure +5 volts has been removed from the system and ensure that all bistable switches on the Maintenance Panel are down.

Table 3-3. MPI PCA J3 Pin Connections

Pin	Signal
1	MCU TRIG
3	V TRIG
5	$\overline{\text{MR}}$
7	+5 volts (100 source)
19	REGISTER DISPLAY 0
21	REGISTER DISPLAY 1
23	REGISTER DISPLAY 2
25	REGISTER DISPLAY 3
27	REGISTER DISPLAY 4
29	REGISTER DISPLAY 5
31	REGISTER DISPLAY 6
33	REGISTER DISPLAY 7
35	REGISTER DISPLAY 8
37	REGISTER DISPLAY 9
39	REGISTER DISPLAY 10
41	REGISTER DISPLAY 11
43	REGISTER DISPLAY 12
45	REGISTER DISPLAY 13
47	REGISTER DISPLAY 14
49	REGISTER DISPLAY 15

Note: All even-numbered pins are ground.

b. At the MPI PCA, set switch S1 to the LAMP TEST position. Every lamp on the Maintenance Panel should light.

c. Set S1 to the SWITCH TEST position. Some lamps should go out.

CAUTION

Before performing step d below, be sure S1 has been set to the SWITCH TEST position. Otherwise, switch information may enter the computer system. The switch information is the result of voltage spikes produced when the plug is removed from J1.

d. Remove the plug from receptacle J1 on the Maintenance Panel.
All lamps should go out. Replace the plug.

3-19. SWITCH TEST. The switches on the Maintenance Panel can be
tested by placing the switch on the MPI PCA to the SWITCH TEST
position. This routes the switch information back to the Main-
tenance Panel to be displayed by lighting or extinguishing one or
more lamps for each switch. The data from the switches will be
displayed by nine groups of lamps. See figure 3-5.

The information for the REGISTER SELECTION switches is coded into
six binary bits. These bits are displayed twice by the lamps in
group five. Figure 3-5 illustrates the individual groups of
lamps and also indicates by each switch which lamp(s) will illum-
inate or extinguish when the switch is exercised.

Note

Although twelve groups of lamps are
designated in figure 3-5, not all
of the groups of lamps will be ex-
ercised during the switch test.

The bistable switches, shown in the down position, will illumi-
nate the lamp/lamps indicated by the numbers above the switch
when the switch is placed up. The two-position spring return
switches, shown in the up position, will extinguish the lamps
indicated by the numbers below the switch when the switch is held
down. The three-position spring return switches (REGISTER
SELECTION switches) have a code above and below the switch that
will be displayed when the switch is held in that position.
Stenciling on the panel has not been shown for clarity. Where
two pairs of numbers are present, two lamps will be illuminated.
For example, 1,8 and 9,8 signify that group 1 lamp 8 and group 9
lamp 8 will be illuminated when the switch is exercised; the
lamps will extinguish when the switch is off. The reverse condi-
tion applies to the two-position spring return switches, extin-
guishing the lamps when the switch is exercised. When exercising
the REGISTER SELECTION switches, a binary coded value is present-
ed by the lamps in group five. The coded value appears twice, in
bits 0-5 and bits 8-13. The indications presented by bits 8-13
should be ignored.

CAUTION

After the switch test is complete, be sure all bistable
switches on the Maintenance Panel are in the down posi-
tion before restoring S1 on the MPI PCA to the NORMAL
position. This will prevent information from entering
the computer when the switch is returned to NORMAL.
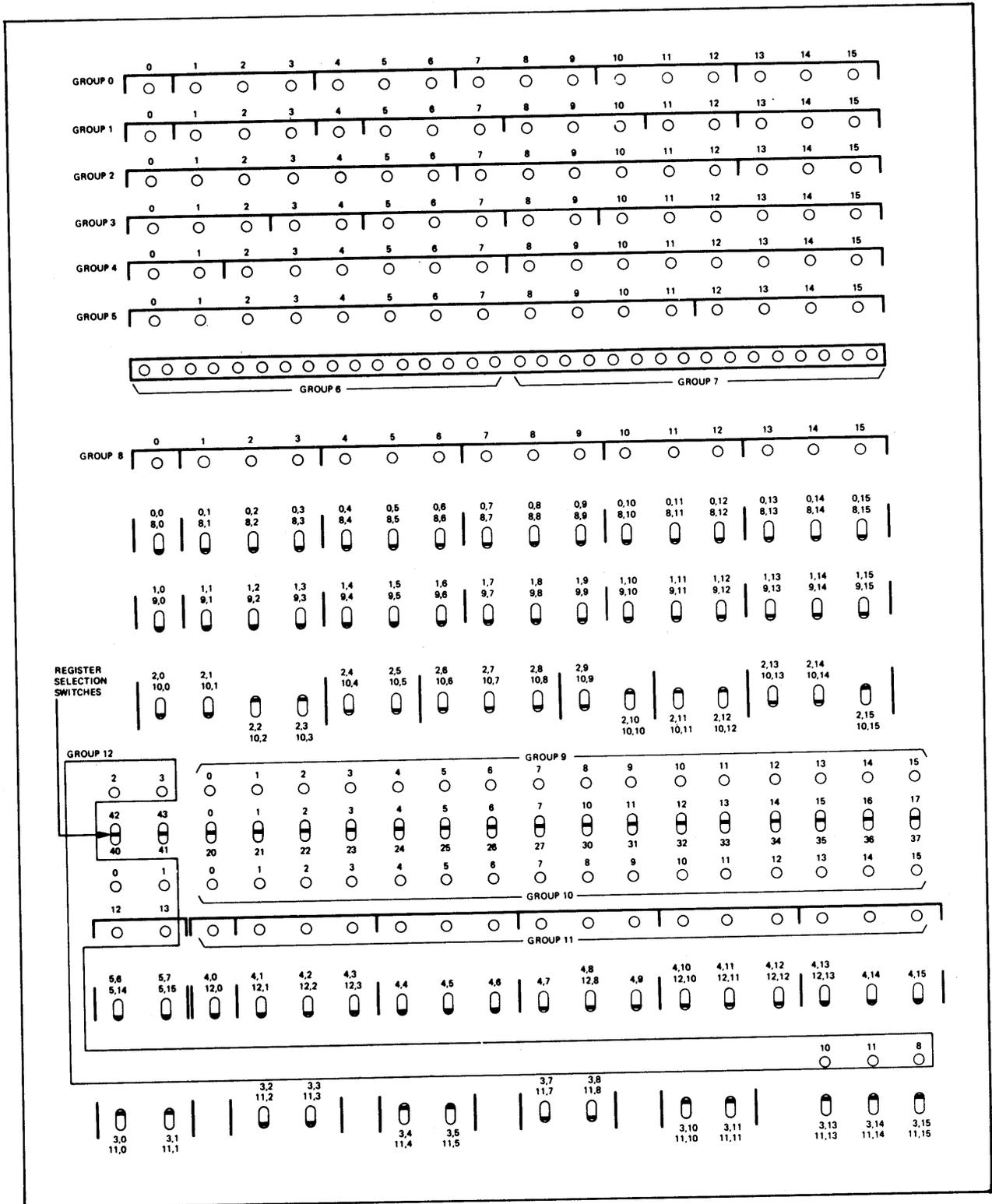
Figure 3-5. Switch Test Lamp Indications

NOTES

MACHINE INSTRUCTIONS AND STACK OPERATIONS

This section contains information on the computer system's basic instruction set and describes representative instructions for most of the 13 instruction groups and how some of these instructions affect stack operations. For complete descriptions of all the machine instructions, including extended instruction sets refer to the HP 3000 Series II/III Computer System Machine Instruction Set Manual, part no. 30000-90022.

4-1. INSTRUCTION DECODING

As the CPU executes a user program, it fetches the required machine instructions from memory. A ROM address of a microprogram stored in a microprogram ROM is generated for the instructions. There is a microprogram in ROM for each of the machine instructions. The ROM address is stored in the ROM Address Register (RAR). The RAR is used first to access the initial microinstruction and is then incremented to point to the next microinstruction. Thus, the entire microprogram for a particular machine instruction is called and executed by the CPU as discussed in Section II.

4-2. TRAPS AND INTERRUPTS

Only those traps and interrupts which occur as a result of instruction execution over which the user has some control are used in the instruction descriptions provided in this section. They are defined here by segment #1 Segment Transfer Table (STT) number.

a. STT #1; BNDV - Bounds Violation. An operand or instruction is outside of the legal bounds for a particular mode of addressing.

b. STT #17; STTV - Segment Transfer Table Violation. A variety of conditions can force this trap as follows:

   (1) The STT number in an external progam label is greater than the STT length pointed to by PL in the referenced segment. This error can occur in PCAL, LLBL, and the firmware interrupt handler while attempting to set up a new segment.

   (2) In LLBL, the label fetched from PL-N is an internal label and N is greater than 128 (%177). This would require too large an STT number when creating the external label.

(3) In PCAL and interrupt handler when setting up a new segment, the STT number in the external program label points to an external program label in the new segment.

(4) In SCAL, (PL-N) is an external label.

c. STT #18; CSTV - Code Segment Table Violation. An attempt is made to transfer to Segment 0 or 192, or a segment number is greater than the CST length.

d. STT #19; DSTV - Data Segment Table Violation. The data segment number referenced by MFDS, MTDS, or MDS is greater than the DST length or is 0.

e. STT #20; STUN - Stack Underflow. The process being executed or being transferred to is non-privileged and SM is less than DB.

f. STT #21; MODE - Privileged Mode Violation. The code segment being executed is non-privileged (bit 0 of the Status Register is 0) and an attempt is made to execute a privileged instruction. This violation also occurs in EXIT if an attempt is made to exit from user to privileged mode or, if exiting from user mode, the External Interrupts bit in the Status Register has been altered.

g. STT #24; STOV - Stack Overflow. SM is greater than Z or may become greater as a result of the current instruction.

h. STT #25; ARITH - Arithmetic. All User Traps will be executed in the segment #1 routine pointed to by STT #25. The error conditions and their parameters are as follows:

Interrupt Type	Octal Parameters
Integer Overflow	000001
Floating Point Overflow	000002
Floating Point Underflow	000003
Integer Divide-by-Zero	000004
Floating Point Divide-by-Zero	000005

i. STT #31; ABS CST - Absent Code Segment. The absence bit in the CST entry for the referenced segment is set. The interrupt handler and PCAL stack a (second) marker; others including EXIT, IXIT, etc., do not.

j. STT #32; TRACE - Code Segment Trace. Code segment is being traced.

k. STT #33; UNCALL - Uncallable STT Entry. The uncallable bit in a local label or, if the STT number is 0, in (PL) is set. This trap does not stack a (second) marker.

1. STT #34; ABS DST - Absent Data Segment. The absence bit in the DST entry for the referenced segment is set.

4-3. CONDITION CODE

Bits 6 and 7 of the CPU Status Register are used for the condition code. Although several instructions make special use of the condition code, the condition code typically indicates the state of an operand (or a comparison result with two operands). The operand may be a byte, word, doubleword, tripleword, or quadrupleword and may be located on the TOS, in the Index Register, or in a specified memory location. Refer to paragraph 2-46 for condition code interpretations.

4-4. INSTRUCTION FORMATS

The machine instruction formats are shown in figures 4-1 through 4-5. For a general discussion of the formats, refer to paragraph 2-30.

4-5. INSTRUCTION DEFINITIONS

Paragraphs 4-6 through 4-15 contain definitions for 36 of the 191 machine instructions. The definitions are arranged in mnemonic alphabetical order within each of the instruction groups. When additional information is required to fully define a particular instruction, an Instruction Commentary number reference is made immediately following the instruction's definition. In such cases, refer to the corresponding reference number in paragraph 4-16. Also, some of the instruction definitions refer to the first five elements of the stack as A, B, C, D, and E. With this convention, A is the TOS (S), B is S-1, C is S-2, D is S-3, and D is S-4.

4-6. Stack Op Instructions

ADD   Add

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

Alternate
Position
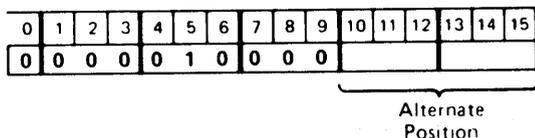
The top two words of the stack are added in integer form and are then deleted. The resulting sum is pushed onto the stack.

Stack opcode: 20
Indictors: CCA, Carry, Overflow
Traps: STUN, ARITH

SUB OP CODE 00, STACK OP CODE 00 - 77

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

0 0

Mnemonic	Bits 4 - 9 or 10 - 15
NOP	00
DELB	01
DDEL	02
ZROX	03
INCX	04
DECX	05
ZERO	06
DZRO	07
DCMP	10
DADD	11
DSUB	12
MPYL	13
DIVL	14
DNEG	15
DXCH	16
CMP	17
ADD	20
SUB	21
MPY	22
DIV	23
NEG	24
TEST	25
STBX	26
DTST	27
DFLT	30
BTST	31
XCH	32
INCA	33
DECA	34
XAX	35
ADAX	36
ADXA	37

Mnemonic	Bits 4 - 9 or 10 - 15
DEL	40
ZROB	41
LDXB	42
STAX	43
LDXN	44
DUP	45
DDUP	46
FLT	47
FCMP	50
FADD	51
FSUB	52
FMPY	53
FDIV	54
FNEG	55
CAB	56
LCMP	57
LADD	60
LSUB	61
LMPY	62
LDIV	63
NOT	64
OR	65
XOR	66
AND	67
FIXR	70
FIXT	71
INCB	73
DECB	74
XBX	75
ADBX	76
ADXB	77

Figure 4-1.   Sub-Opcode 00 Formats

SUBOPCODE 01, OPCODES 00 - 17

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

01

Mnemonic	4	Bits 5 - 9	Bits 10 - 15
ASL	x	00	SHIFT COUNT
ASR	x	01	SHIFT COUNT
LSL	x	02	SHIFT COUNT
LSR	x	03	SHIFT COUNT
CSL	x	04	SHIFT COUNT
CSR	x	05	SHIFT COUNT
SCAN	x	06	ALL ZEROS
IABZ	I	07	± DISPLACEMENT
TASL	x	10	SHIFT COUNT
TASR	x	11	SHIFT COUNT
IXBZ	I	12	± DISPLACEMENT
DXBZ	I	13	± DISPLACEMENT
BCY	I	14	± DISPLACEMENT
BNCY	I	15	± DISPLACEMENT
TNSL	x	16	ALL ZEROS
QASL	0·	17	SHIFT COUNT
QASR	1	17	SHIFT COUNT

X = INDEX BIT
I = INDIRECT BIT
SHADED BITS ARE RESERVED BITS

SUBOPCODE 01, OPCODES 20 - 37

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

01

Mnemonic	4	Bits 5 - 9	Bits 10 - 15
DASL	x	20	SHIFT COUNT
DASR	x	21	SHIFT COUNT
DLSL	x	22	SHIFT COUNT
DLSR	x	23	SHIFT COUNT
DCSL	x	24	SHIFT COUNT
DCSR	x	25	SHIFT COUNT
CPRB	I	26	± DISPLACEMENT
DABZ	I	27	± DISPLACEMENT
BOV	I	30	± DISPLACEMENT
BNOV	I	31	± DISPLACEMENT
TBC	x	32	BIT POSITION
TRBC	x	33	BIT POSITION
TSBC	x	34	BIT POSITION
TCBC	x	35	BIT POSITION
BRO	I	36	± DISPLACEMENT
BRE	I	37	± DISPLACEMENT

Figure 4-2.   Sub-Opcode 01 Formats

CMP   Compare

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

Alternate
Position
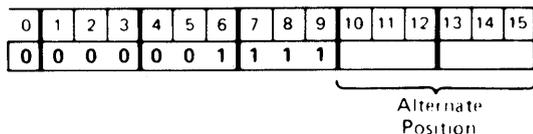
The Condition Code is set to pattern C as a result of the integer comparison of the second word of the stack with the TOS. Both words are deleted.

Stack opcode: 17
Indicators: CCC
Traps: STUN

SUBOPCODE 02, MOVE OPCODES 00, 0 - 5

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

02   00

Mnemonic	Bits 8 - 10	Bits 11 - 15				
MOVE	0	B	0	0		SDEC
MVB	1	B	0	0		SDEC
MVBL	2	0	0	0		SDEC
MABS	2	0	1		SDEC	
SCW	2	1	0	0		SDEC
MTDS	2	1	1		SDEC	
MVLB	3	0	0	0		SDEC
MDS	3	0	1		SDEC	
SCU	3	1	0	0		SDEC
MFDS	3	1	1		SDEC	
MVBW	4	N	A	U		SDEC
CMPB	5	B	0	0		SDEC

SUBOPCODE 02, MINI OPCODES 00, 14 - 17

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

02   00

Mnemonic	Bits 8 - 11	Bits 12 - 15			
RSW	14	0	0	0	0
LLSH	14	0	0	0	1
PLDA	15	0			0
PSTA	15	0	0	0	1
LSEA	16	0	0	0	0
SSEA	16	0	0	0	1
LDEA	16	0	0	1	0
SDEA	16	0	0	1	1
IXIT	17	0	0	0	0
PCN	17	0	0	1	0

SUBOPCODES 02, OPCODES 01 - 17

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

02

Mnemonic	Bits 4 - 7	Bits 8 - 15	
DMUL	01	CIR (8:15) = % 170	
DDIV	01	CIR (8:15) = % 171	
LDI	02	IMMEDIATE OPERAND	
LDXI	03	IMMEDIATE OPERAND	
CMPI	04	IMMEDIATE OPERAND	
ADDI	05	IMMEDIATE OPERAND	
SUBI	06	IMMEDIATE OPERAND	
MPYI	07	IMMEDIATE OPERAND	
DIVI	10	IMMEDIATE OPERAND	
PSHR	11	‡	
LDNI	12	IMMEDIATE OPERAND	
LDXN	13	IMMEDIATE OPERAND	
CMPN	14	IMMEDIATE OPERAND	
EXF	15	START BIT #	# OF BITS
DPF	16	START BIT #	# OF BITS
SETR	17	‡	

‡BIT 8 = STACK BANK REGISTER
 BIT 9 = DB-BANK, DB REGISTER
 BIT 10 = DL REGISTER
 BIT 11 = Z REGISTER
 BIT 12 = STATUS REGISTER
 BIT 13 = X REGISTER
 BIT 14 = Q REGISTER
 BIT 15 = S REGISTER

Shaded bits are reserved

A = Alphabetic
B = PB/DB
N = Numeric
SDEC = S Decrement
U = Upshift

Figure 4-3.   Sub-Opcode 02 Formats

SUBOPCODE 03, SPECIAL OPCODES 00

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

03   00

Mnemonic	Bits 8 - 11	Bits 12 - 15
LST	00	K FIELD
PAUS	01	0 0 0 0
SED	02	0 0 0 X
XCHD	03	0 0 0 0
PSDB	03	0 0 0 1
DISP	03	0 0 1 0
PSEB	03	0 0 1 1
SMSK	04	0 0 0 0
SCLK	04	0 0 0 1
RMSK	05	0 0 0 0
RCLK	05	0 0 0 1
XEQ	06	K FIELD
SIO	07	K FIELD
RIO	10	K FIELD
WIO	11	K FIELD
TIO	12	K FIELD
CIO	13	K FIELD
CMD	14	K FIELD
SST	15	K FIELD
SIN	16	K FIELD
HALT	17	K FIELD

SUBOPCODE 03, OPCODES 01 - 17

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

03

Mnemonic	Bits 4 - 7	Bits 8 - 15
SCAL	01	N FIELD
PCAL	02	N FIELD
EXIT	03	N FIELD
SXIT	04	N FIELD
ADXI	05	IMMEDIATE OPERAND
SBXI	06	IMMEDIATE OPERAND
LLBL	07	PL – DISPLACEMENT
LDPP	10	P+ DISPLACEMENT
LBPN	11	P– DISPLACEMENT
ADDS	12	IMMEDIATE OPERAND
SUBS	13	IMMEDIATE OPERAND
ORI	15	IMMEDIATE OPERAND
XORI	16	IMMEDIATE OPERAND
ANDI	17	IMMEDIATE OPERAND

Shaded bits are reserved and ignored.

x = 1 or 0.

Figure 4-4.   Sub-Opcode 03 Formats

DDUP   Double Duplicate

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

Alternate
Position

The double word in the top two words of the stack is duplicated
by pushing a copy of it onto the stack.

Stack opcode: 46
Indicators: CCA on new TOS double word
Traps: STUN, STOV

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

Mnemonic	Bits 0 - 3	Bits 4 - 15				
LOAD	04	X	I	MODE AND DISPLACEMENT		
TBA	05	0	0	0	±	DISPLACEMENT
MTBA	05	0	1	0	±	DISPLACEMENT
TBX	05	1	0	0	±	DISPLACEMENT
MTBX	05	1	1	0	±	DISPLACEMENT
STOR	05	x	I	1	MODE AND DISPLACEMENT	
CMPM	06	x	I	MODE AND DISPLACEMENT		
ADDM	07	x	I	MODE AND DISPLACEMENT		
SUBM	10	x	I	MODE AND DISPLACEMENT		
MPYM	11	x	I	MODE AND DISPLACEMENT		
INCM	12	x	I	0	MODE AND DISPLACEMENT	
DECM	12	x	1	I	MODE AND DISPLACEMENT	
LDX	13	x	I	MODE AND DISPLACEMENT		
BR	14	x	I	0	±	DISPLACEMENT
BR	14	x	1	1	MODE AND DISPLACEMENT	
BCC	14	I	0	1	> = < ±	DISPLACEMENT
LDB	15	x	I	0	MODE AND DISPLACEMENT	
LDD	15	x	I	1	MODE AND DISPLACEMENT	
STB	16	x	I	0	MODE AND DISPLACEMENT	
STD	16	x	I	1	MODE AND DISPLACEMENT	
LRA	17	x	I	MODE AND DISPLACEMENT		

X = INDEX BIT
I = INDIRECT BIT

Figure 4-5. Sub-Opcode 04 thru 17 Formats

DEL   Delete A

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

Alternate
Position

The top word of the stack is deleted.

Stack opcode: 40
Indicators: Unaffected
Traps: STUN

DIV   Divide

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

The integer in the second word of the stack is divided by the in-
teger on the TOS. The two words are then deleted. The second
word is replaced by the quotient, and the top word is replaced by
the remainder.

Stack opcode: 23
Indicators: CCA on quotient, Overflow
Traps: STUN, ARITH

DUP   Duplicate A

```
 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |10|11|12|13|14|15|
 0 | 0   0   0 | 1   0   0 | 1   0   1 |       |      |
```
Alternate
Position
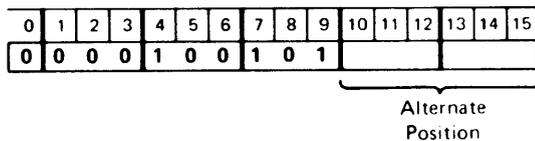
The top word of the stack is duplicated by pushing a copy of the
TOS onto the stack.

Stack opcode: 45
Indicators: CCA
Traps: STUN, STOV

INCA   Increment A

```
 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |10|11|12|13|14|15|
 0 | 0   0   0 | 0   1   1 | 0   1   1 |       |      |
```
Alternate
Position
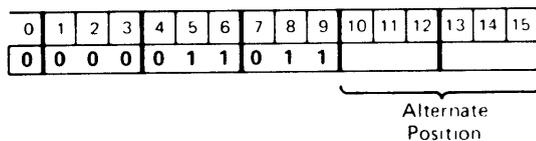
The TOS is incremented by one in integer form.

Stack opcode: 33
Indicators: CCA, Carry, Overflow
Traps: STUN, ARITH

MPY   Multiply

```
 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |10|11|12|13|14|15|
 0 | 0   0   0 | 0   1   0 | 0   1   0 |       |      |
```
Alternate
Position
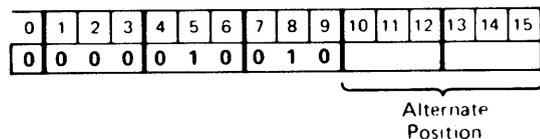
The top two words of the stack are multiplied in integer form.The
two words are deleted and the least significant word of the doub-
le length product is pushed onto the stack. If the high order 17
bits of the double length product (including the sign bit of the
second word) are not all zeros or all ones, Overflow is set.

Instruction Commentary 1.
Stack opcode: 22
Indicators: CCA, Overflow
Traps: STUN, ARITH

TEST   Test TOS

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

Alternate
Position
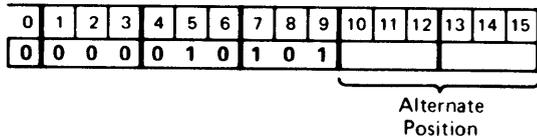
The Condition Code is set to pattern A according to the   content
of the TOS word.

Stack opcode: 25
Indicators: CCA
Traps: STUN

XOR   Logical Exclusive-OR

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

Alternate
Position
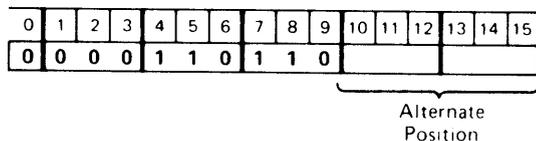
The top two words of the stack are combined by a logical
exclusive-OR. The two words are deleted and the result is pushed
onto the stack.

Stack opcode: 65
Indicators: CCA on the new TOS
Traps: STUN

4-7. Shift Instructions

ASL   Arithmetic Shift Left

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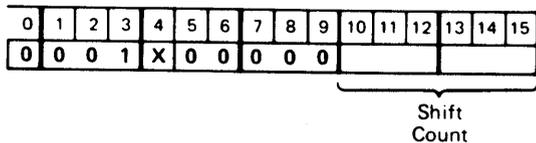
Shift
Count

The TOS is shifted left n bits, preserving the sign bit.   The
value of n (modulo 64) is the number specified in the argument
field plus, if X is specified (bit 4), the content of the Index
Register.

Instruction Commentary 2.
Sub-opcode 1: 00
Indicators: CCA
Traps: STUN

CSL   Circular Shift Left

```
 0  1  2  3  4  5  6  7  8  9  10 11 12  13 14 15
 0  0  0  1  X  0  0  1  0  0  |           |
                               _____/
                                   Shift
                                   Count
```
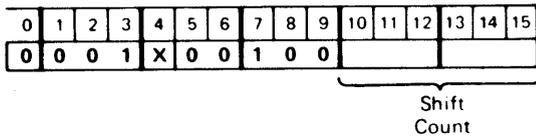
The TOS is shifted left n bits circularly. The value of n (modulo 64) is the number specified in the argument field plus, if X is specified, the content of the Index Register.

Instruction Commentary 2.
Sub-opcode 1: 04
Indicators: CCA
Traps: STUN

4-8. Branch Instructions

BCC   Branch On Condition Code

```
 0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15
 1  1  0  0  I  0  1  G  E  L  ±
                      \_____/  _____/
                        CCF     Displacement
```
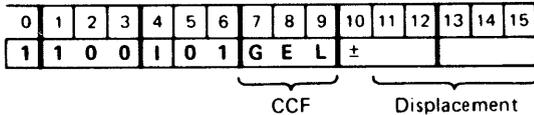
The Condition Code in the Status Register is compared with conditions named in the CCF field of the instruction. If the named conditions are met, control is transferred to P +/- displacement; otherwise to P+1. The displacement is limited to +/-31. Control is transferred to the branch address under the following conditions:

```
    If CCF = 0, never branch
           = 1, branch if CC = CCL
           = 2, branch if CC = CCE
           = 3, branch if CC = CCL or CCE
           = 4, branch if CC = CCG
           = 5, branch if CC = CCG or CCL
           = 6, branch if CC = CCG or CCE
           = 7, always branch
```

Memory opcode: 14, bits 5,6 = 01
Indicators: Unaffected
Addressing modes: P relative (+/-), direct or indirect
Traps: BNDV if user or privileged

BR   Branch Unconditionally

```
 0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15
 1  1  0  0  X  I  0  ±  |                      |
                        _____/
                              Displacement
                              P Relative
```
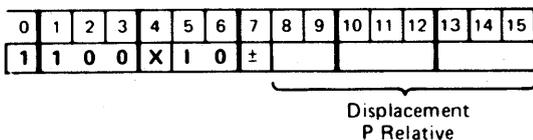
For P relative mode, control is transferred unconditionally to P +/- displacement, plus (if specified) the value in X; may be indirect. For DB, Q, and S relative modes, control is transferred indirectly (only) via the location specified by DB, Q, or S +/- this displacement; the content of the location so specified is added to PB (plus post-indexing if X is specified) to obtain the effective address for P.

Instruction Commentary 3.
Memory opcode: 14, bits 5,6 = 00, 10, or 11
Indicators: Unaffected

Addressing modes: P relative (+/-), direct or indirect
                   DB+ relative, indirect
                   Q+ relative, indirect
                   Q- relative, indirect
                   S- relative, indirect
                   Indexing available
Traps: BNDV, BNDV on P and P relative if user or privileged

4-9. Move Instructions

<div align="center">Note</div>

All Move instructions are interruptable after each word (or byte) transfer and will continue from the point of interrupt when control is returned to the instruction.

CMPB   Compare Bytes

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

PB/DB   SDEC

This instruction scans two byte strings simultaneously until the compared bytes are unequal or until a specified number of comparisons have been made. CMPB expects a signed byte count in A, a DB or PB relative displacement for a source byte address in B, and a DB relative displacement for a target byte address in C. As long as the word count in A has not been counted to zero, the comparison proceeds as follows: The content of the byte address location specified by DB + B or PB + B is compared with the content of the byte address location specified by DB + C. If the byte count in A is positive, the source and target displacement values in B and C are incremented by one after each comparison, and the byte count is decremented by one. If the byte count in A is negative, the source and target displacement values in B and C are decremented by one after each comparison, and the byte count is incremented by one. Note that the byte count is always changed by one toward zero. The instruction terminates when either a comparison fails or the byte count in the TOS reaches zero. The Condition Code is set to a special pattern to indicate the terminating condition. On termination, the instruction deletes from the stack the number of words (0, 1, 2, or 3) specified by the

SDEC field of the instruction.

Instruction Commentary 4.
Move opcode: 5
Indicators: CCE if byte count = 0
            CCG if target byte > source byte (final)
            CCL if target byte < source byte (final)

Addressing modes: Byte addressing
                  DB+ or PB+ for source
                  DB+ for target
Traps: STUN, STOV, BNDV, BNDV on P relative if user or privileged

4-10. Privileged Memory Reference Instructions

LSEA   Load Single Word From Extended Address

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

A bank address is in B and a 16-bit absolute address of a loca-
tion in that bank is in A.   The word at that address is pushed
onto the stack.

Mini-opcode: 16, bits 14, 15 = 00
Indicators: CCA
Addressing mode: Absolute
Traps: STUN, STOV, MODE
This is a privileged instruction.

PLDA   Privileged Load From Absolute Address

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

The content of the Index Register is a 16-bit absolute address in
bank 0; the content of this address is pushed onto the stack.

Mini-opcode: 15, bit 15 = 0
Indicators: CCA
Addressing mode: Absolute
Traps: STOV, MODE
This is a privileged instruction.

4-11. Immediate Instructions

ADDI   Add Immediate

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

Immediate Operand

The immediate operand N is added to the TOS in integer form, and
the sum replaces the TOS. The value of N is given in the argu-
ment field of the instruction, and is expressed as a positive
integer in the range 0 through 255.

Sub-opcode 2: 05
Indicators: CCA on the new TOS,
Carry, Overflow
Traps: STUN, ARITH

LDI   Load Immediate

```
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
┌─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬──┬──┬──┬──┬──┬──┐
│0│0│1│0│0│0│1│0│ │ │  │  │  │  │  │  │
└─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴──┴──┴──┴──┴──┴──┘
                  └──────────┬──────────┘
                      Immediate Operand
```
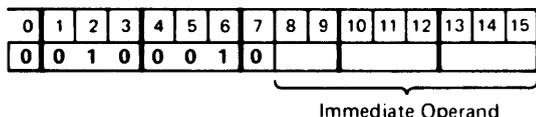
The immediate operand N is pushed onto the stack. The value of N
is given in the argument field of the instruction, and is expres-
sed as a positive integer in the range 0 through 255.

Sub-opcode 2: 02
Indicators: CCA on the new TOS
Traps: STOV

4-12. Register Control Instructions

PSHR   Push Registers

```
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
┌─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬──┬──┬──┬──┬──┬──┐
│0│0│1│0│1│0│0│1│ │ │  │  │  │  │  │  │
└─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴──┴──┴──┴──┴──┴──┘
                   ↑ DB DL  Z Sta  X  Q  S
                   └─S-Bank
```
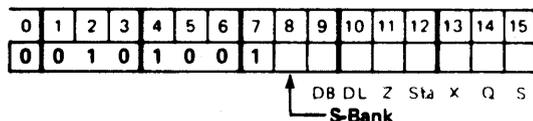
The content of a register (or the displacement it represents)
specified by any bit 8 through 15 is pushed onto the stack. If
more than one register (or displacement) is specified, the con-
tents will be stacked in the order shown below, such that if all
nine were specified, S-Bank would be on the TOS after execution,
DB next, etc. Note that when S-DB is pushed, the value stacked
will be as it existed before the execution of this instruction.
Stack overflow occurs if the original S+9 exceeds Z, regardless
of the number of registers pushed.

     If bit 15 = 1, push S-DB
     If bit 14 = 1, push Q-DB
     If bit 13 = 1, push Index Register
     If bit 12 = 1, push Status Register
     If bit 11 = 1, push Z-DB
     If bit 10 = 1, push DL-DB
    *If bit 9 = 1, push DB-Bank and DB Register
    *If bit 8 = 1, push S-Bank

Sub-opcode 2: 11
Indicators: Unaffected
Traps: STOV, MODE
*These are privileged operations.

4-13. Program Control and Special Instructions

DISP   Dispatch

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

This instruction is used to transfer to the Dispatcher's entry point; or to request such a transfer if executed while on the ICS or within the range of a PSDB-PSEB pair.

Instruction Commentary 7.
Special opcode: 03, bits 12-15 = 0010
Indicators: See instruction commentary.
Traps: MODE, CSTV, TRACE, ABS CST, BNDV if user or privileged
This is a privileged instruction.

EXIT   Exit From Procedure

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

N

This instruction is used to return from a procedure called by the PCAL instruction or by some interrupts. A normal exit occurs by restoring the return address to P, restoring the previous contents of the Index and Status Registers, and deleting all stack variables incurred by the called routine plus its marker, plus N number of procedure parameters. The value of N may be any number from 0 to 255 for exits from PCAL routines; it must be 0 for exits from interrupt routines. If bit 0 of the return-P marker word is a "1", control is transferred to Trace, segment #1, STT #32 (decimal).

Instruction Commentary 6.
Sub-opcode 3: 03
Indicators: Restored to values before PCAL
Traps: STUN (going to user mode),STOV, MODE, CSTV,TRACE, ABS CST,
       BNDV if user or privileged

IXIT   Interrupt Exit

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

This instruction is used to exit from those interrupt service routines which always run on the Interrupt Control Stack (ICS). This results in a return to the interrupted process (which may be another interrupt or the Dispatcher) or a transfer to the Dispatcher's entry point. The action taken depends in part on the sequence of DISP, PSDB, and PSEB instructions which have been executed. IXIT is also used by the Dispatcher to exit to a process being launched.

Instruction Commentary 7.
Mini-opcode: 17, bits 12-15 = 0000
Indicators: Restored to those before interrupt or as specified
             for the Dispatcher
Traps: MODE, STOV,CSTV,TRACE,ABS CST, BNDV if user or privileged
This is a privileged instruction.

PAUS   Pause

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

Not Used

The computer hardware pauses; interrupts may occur.   Bits 12
through 15 are ignored.

Special opcode: 01
Indicators: Unaffected
Traps: MODE
This is a privileged instruction.

PCAL   Procedure Call

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

N

Control is transferred to the location pointed to by the evalua-
tion of the program label at PL-N, unless N is zero, the program
label is taken from the TOS and then deleted. Then a four word
stack marker is placed on the stack, and Q and S are updated to
point at this new marker. The program label may be local or ex-
ternal.   If the Trace bit is on in the target CST entry, a call
will be made to Trace, segment #1, STT #32 (decimal). If a priv-
ileged user is calling a user segment, it will run in privileged
mode.

Instruction Commentary 6.
Sub-opcode 3: 02
Indicators: Unaffected
Addressing modes:
    Indirect via: PL - N (if N does not equal 0)
                  TOS (if N = 0)
    Local Label: PB+
    External Label: via CST to local label in target
                    segment
Traps: STUN, STOV, CSTV, STTV, ABS CST, TRACE, UNCALL, BNDV if
        user or privileged

4-14. I/O Instructions

CIO   Control I/O

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

K

This instruction assumes that the TOS contains a control word and expects a device number to be given in the stack at S-K. CIO transmits the TOS to the specified device controller, along with a CIO signal. If the device controller acknowledges receiving the word, the TOS is deleted and the Condition Code is set to CCE. If the device controller does not respond, the Condition Code is set to CCL and the instruction is terminated.

Special opcode: 13
Indicators: CCE = responding device controller
            CCL = non-responding device controller
Traps: STUN, MODE
This is a privileged instruction.

CMD   Command

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

K

This instruction assumes that the TOS contains a 16-bit data word to be sent to a system hardware module and expects a command word in the stack at S-K. Bits 13 through 15 of the command word specify the module number, and bits 10 and 11 are used to specify a module command. (The four possible commands are interpreted by the target module and do not form a part of this instruction's definition.) CMD sends the 16-bit data word and the 2-bit command over the Central Data Bus to the specified module, and then deletes the TOS. (Note: If the destination module is not ready, the CPU will not proceed until that module becomes ready.)

Special opcode: 14
Indicators: Unaffected
Traps: STUN, MODE
This is a privileged instruction.

RIO   Read I/O

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

K

This instruction expects a device number to be given in the stack at S-K. RIO first checks if the device is ready by checking bit 1 of the device controller's Status Register. If it is ready (bit = "1"), the 16-bit data word from the device is pushed onto the stack and the Condition Code is set to CCE. If it is not

ready (bit = "0"), the content of the device controller's Status Register is pushed onto the stack and the Condition Code is set to CCG. If the device controller does not respond to the readiness test, the Condition Code is set to CCL and the instruction is terminated.

Special opcode: 10
Indicators: CCL = non-responding device controller
            CCE = device ready
            CCG =device not ready
Traps: STOV, MODE
This is a privileged instruction.

SIO   Start I/O

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

                                              K

The SIO instruction expects the absolute starting address of an I/O program to be on the TOS, and a device number to be in the stack at S-K. The instruction first checks if the device is ready for an SIO by checking bit 0 of the device controller's Status Register. Bit 0 is the "SIO OK" bit. If it is ready (bit = "1"), the TOS is stored into the first location of the DRT entry for the device specified at S-K; an SIO command is then issued to the device controller to begin execution of its I/O program. If the device is not ready (bit 0 of the device status = "0"), the content of the device controller's Status Register is pushed onto the stack and the Condition Code is set to CCG. If the device controller does not respond, the Condition Code is set to CCL and the instruction is terminated. If the device is ready, the TOS is deleted and the Condition Code is set to CCE.

Instruction Commentary 8.
Special opcode: 07
Indicators: CCL = non-responding device controller
            CCE = device ready
            CCG = device not ready
Traps: STUN, STOV, MODE
This is a privileged instruction.

TIO   Test I/O

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

                                              K

This instruction expects a device number to be given in the stack at S-K. TIO obtains a copy of the device status word from the device controller, pushes it onto the stack, and sets the Condition Code to CCE. If the device controller does not respond, the Condition Code is set to CCL and the instruction is terminated.

Special opcode: 12
Indicators: CCE = responding device controller
            CCL = non-responding device controller
Traps: STOVE, MODE
This is a privileged instruction.

WIO Write I/O

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

K
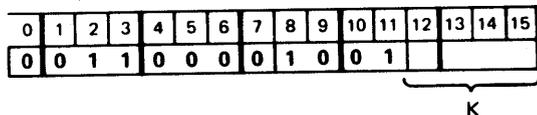
This instruction assumes that the TOS contains a data word and
expects a device number to be given in the stack at S-K. WIO
first checks if the device is ready by checking bit 1 of the de-
vice controller´s Status Register. If it is ready (bit = "1"),
the word is transmitted to the specified device and then deleted
from the stack; the Condition Code is set to CCE. If it is not
ready (bit = "0"), the content of the device controller´s Status
Register is pushed onto the stack and the Condition Code is set
to CCG. If the device controller does not respond, the Condition
Code is set to CCL and the instruction is terminated.

Special opcode: 11
Indicators: CCL = non-responding device controller
            CCE = device ready
            CCG = device not ready
Traps: STUN, STOV, MODE
This is privileged instruction.

4-15. Memory Address Instructions

ADDM Add Memory To TOS

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

Mode and Displacement
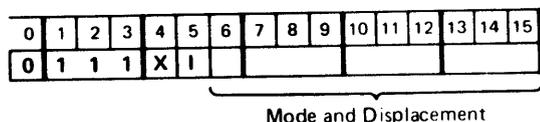
The content of the effective address memory location is added in
integer form to the TOS. The result replaces the operand on the
TOS.

Memory opcode: 07
Indicators: CCA, Carry, Overflow
Addressing modes: P+, P-, DB+, Q+, Q-, S- relative
                  Direct or indirect
                  Indexing available
Traps: STUN, BNDV, ARITH

CMPM Compare TOS With Memory
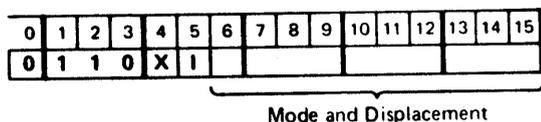
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	0	X	I										

Mode and Displacement

4-19

The Condition Code is set to pattern C as a result of the comparison of the TOS with the content of the effective address location. The TOS is then deleted.

Memory opcode: 06
Indicators: CCC
Addressing modes: P+, P-, DB+, Q+, Q-, S- relative
                              Direct or indirect
                              Indexing available
Traps: STUN

LDD   Load Double

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

Mode and Displacement

The contents of the effective address memory location (E) and the succeeding location (E + 1) are pushed onto the stack. The content of E, the most significant word, is loaded into B; the content of E + 1, the least significant word, is loaded into A. If indirect addressing is used, the word referenced by the initial address (base + displacement) contains a DB+ relative word address. If indexing is used, the effective address is obtained by adding twice the contents of the Index Register to the relative word address.

Memory opcode: 15, bit 6 = 1
Indicators: CCA
Addressing modes: DB+, Q+, Q-, S- relative
                              Direct or indirect
                              (for final indirect: DB+ only)
                              Doubleword Indexing available
Traps: STOV, BNDV

LOAD   Load Word Onto Stack

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

Mode and Displacement

The content of the effective address location is pushed onto the stack.

Memory opcode: 04
Indicators: CCA
Addressing modes: P+, P-, DB+, Q+, Q-, S- relative
                              Direct or indirect
                              Indirect Indexing available
Traps: STOV, BNDV

STOR   Store TOS Into Memory

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

Mode and Displacement
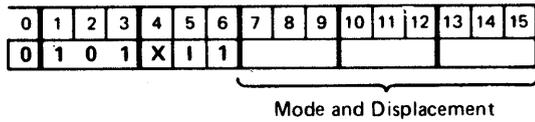
The content of the TOS is stored into the effective address memory location, and is then deleted from the stack.

Memory opcode: 05, bit 6 = 1
Indicators: Unaffected
Addressing modes: DB+, Q+, Q-, S- relative
                  Direct or indirect
                  Indexing available
Traps: STUN, BNDV

4-16. Instruction Commentary

1. MPYL, MPY, DTST, FIXR, FIXT, LMPY.   These six instructions
provide for the deletion of the most significant word of a
doubleword result. The assumption is that the result of the in-
struction (e.g., multiplication product) does not require more
than 16 bits to represent it. The MPY instruction deletes auto-
matically during execution; the remaining five instructions
simply test the result and provide an indication (Carry bit) to
note whether or not the low order word fully represents the true
result. Thus, for these five, the programmer may choose to in-
sert a delete sequence (see figure 4-6) to delete the high order
word if it is insignificant.
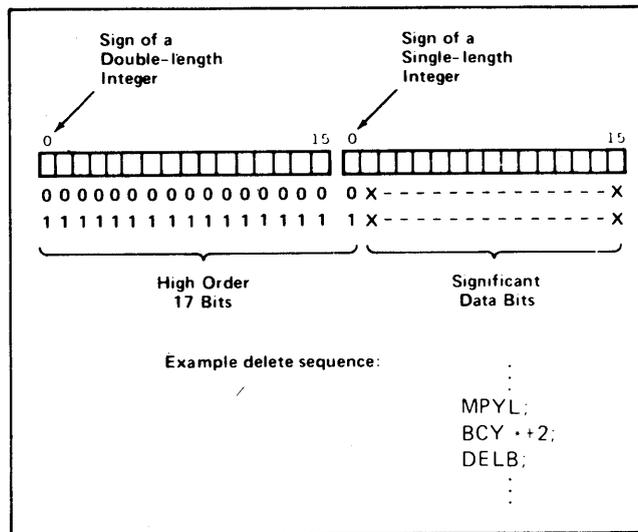
Figure 4-6.   Deleting A High Order Word

For MPYL, DTST, FIXR, FIXT, and LMPY, the Carry bit is cleared if
the high order 17 bits are all zeros or all ones. This test en-
sures that the sign bit of the single-length result will be the
same as the sign of the double-length result. If this is not the

case, Carry is set, and the most significant word should not be deleted. For MPY, Overflow will be set if the test fails, meaning that MPYL should have been used instead of MPY.

2. ASL, ASR, LSL, LSR, CSL, CSR. The actions of the six-single word Shift instructions are shown in figure 4-7. It is assumed that the shift count specified in the argument field of the instruction, is 3 in each case. The before and after conditions of the TOS word are shown for each example.

In the case of arithmetic shifts, the sign bit is always preserved. When shifting left, the bits shifted out of bit 1 (most significant bit next to the sign bit) are lost; zeros are filled into the vacated low order bit positions. When shifting right, the sign bit is copied into the vacated high order bit positions, and bits shifted out of bit 15 (least significant bit) are lost.

In the case of logical shifts, all bits are shifted. Bits are lost out of the high end when shifting left and out of the low end when shifting right. Zeros are filled into the vacated bit positions.

In the case of circular shifts, no bits are lost. Bits shifted out of the high end when shifting left are filled into the vacated low order bit positions. When shifting right, bits shifted out of the low end are filled into the vacated high order bit positions.

Note that, for all Shift instructions, the number of shifts is determined either by the value specified in the argument field of the instruction or, if X is specified ("1" in bit 4), by adding the argument field value to the Index Register contents. This permits the number of shifts to be computed as well as explicitly specified.

All Shift instructions except TNSL use the shift count in a modulo 64 manner. Thus if the final shift count is 100 octal (64 decimal), the data is not shifted at all. Furthermore, if the number of shifts equals or exceeds the number of magnitude bits (whether single, double, or triple word), the following will occur: for left arithmetic shifts and all logical shifts, the magnitude will be all-zero; for right arithmetic shifts, all magnitude bits will be the same as the sign bit; for circular shifts, the circular shifting will continue until the specified number of shifts (up to 63) have been achieved. Except for TNSL, the execution of shift instructions does not alter the contents of the Index Register.

3. BR. The P-relative mode of BR, the unconditional branch instruction, is a conventional P relative branch except for the indexing capability and the extended displacement range. Bits 8 through 15 are available to specify displacement, which therefore can be up to +/-255.

Arithmetic Shift Left   ASL 3

Arithmetic Shift Right   ASR 3

Logical Shift Left   LSL 3

Logical Shift Right   LSR 3

Circular Shift Left   CSL 3

Circular Shift Right   CSR 3
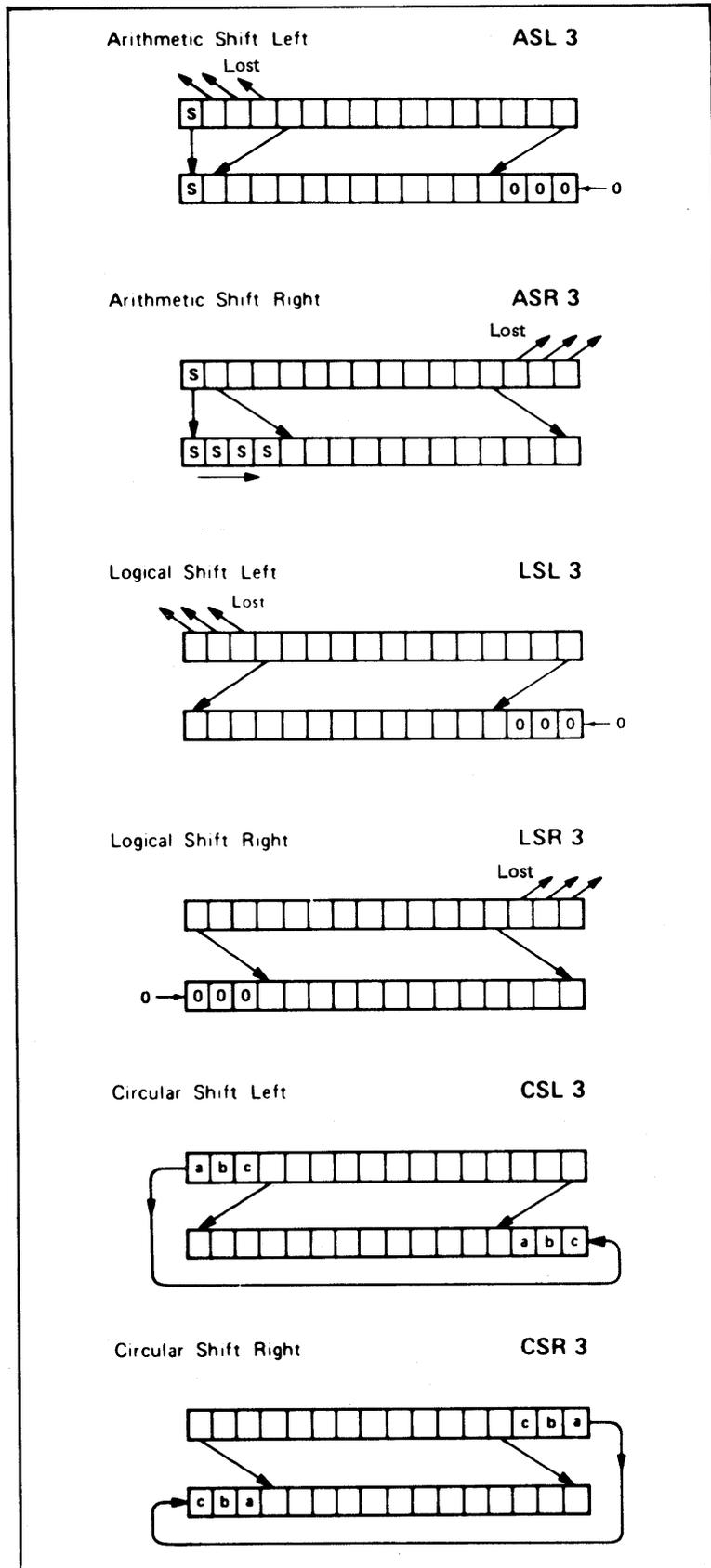
Figure 4-7. Single Word Shifts

The DB, Q, and S relative mode, however, are unconventional in that they permit indirect branches through the data stack. (It is both illegal and impossible to have a direct branch to the stack; the coding of "01" for bits 5 and 6 encodes the BCC instruction.)

Figure 4-8 shows an example of the S- relative mode. Assume that the instruction in location P specifies the S- relative mode, with a displacement of 4, and indexing. This causes an indirect branch to S-4 in the data stack. The content of S-4 is then ad-ded to PB, thus pointing at location "a" in the code segment. Since indexing is specified, the value contained in the Index Register is also added to the address being computed. Thus the ultimate effective address for the branch (next P) is location "a" displaced by the index value.
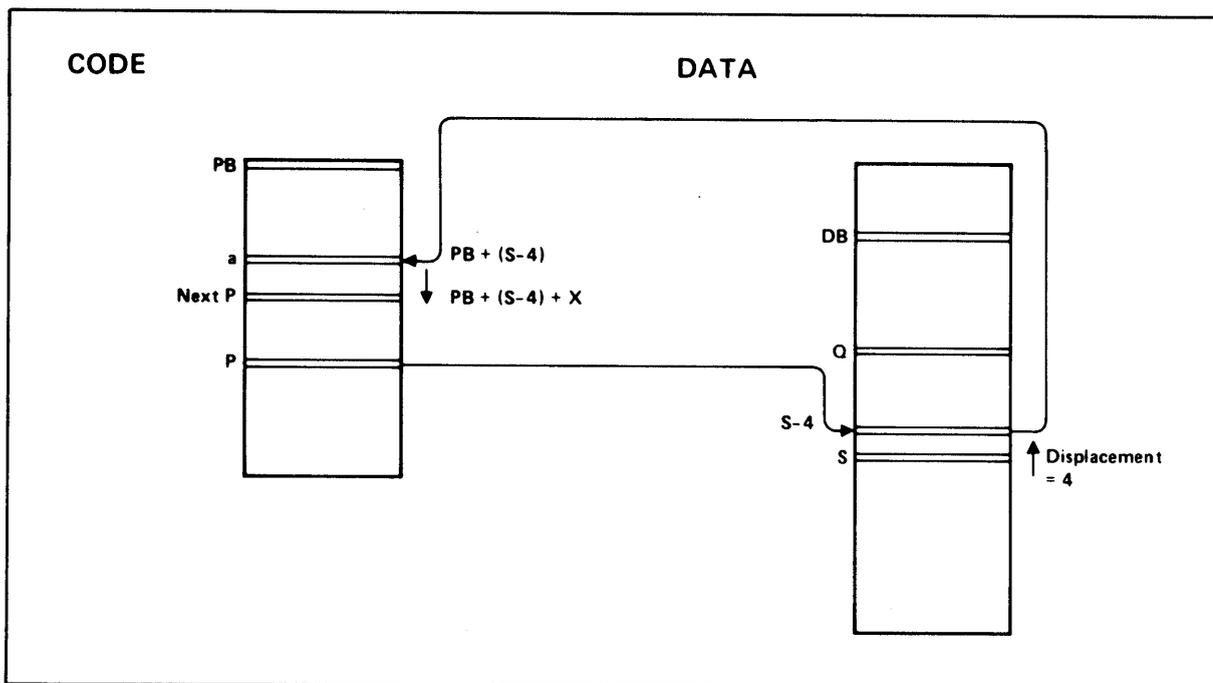
Figure 4-8.   Indirect Branch Via Stack

Note particularly that the indirect address given in the stack is relative to the program base, PB, not to P as is usually the case.   Also note that the displacement is relative to a location in the stack (DB, Q, or S), and that indexing is applied after the indirect addressing has been accomplished.

The displacement range for the DB, Q, and S modes depends on which mode is selected.   For DB+, bits 8 through 15 provide a range of 0 through +255.   For Q+, bits 9 through 15 provide a range of 0 through +127.   For Q- and S-, bits 10 through 15 pro-vide a range of 0 through -63.

4. MOVE, MVB, MVBW, CMPB, SCU, SCW. These six instructions are members of the move group and as such deal with strings of words or bytes. The first three physically move a word or byte string from one block of locations in primary memory to another. The CMPB instruction does not move data but compares data in two complete strings, byte by byte. The last two also do not move data but scan a data string testing the string byte by byte against a test character and a terminal character.

SOURCES. The MOVE, MVB, and CMPB instructions may take source data from either the code segment or the data segment. (For reference purposes, "source" and "target" terminology is retained for CMPB, even through there is no move operation.) If bit 11 of the instruction is a "0", source addresses are PB+ relative; i.e., from the code segment. If bit 11 is a "1", source addresses are DB+ relative; i.e., from the data segment. Figure 4-9 illustrates both cases. Note that the target for either case is in the DB+ area. (Disregard move-direction arrows for CMPB.) Both source and target (MVBW) addresses are DB relative for MVBW, SCU, and SCW. The target need not be "higher" than the source; figure 4-9 shows examples only.
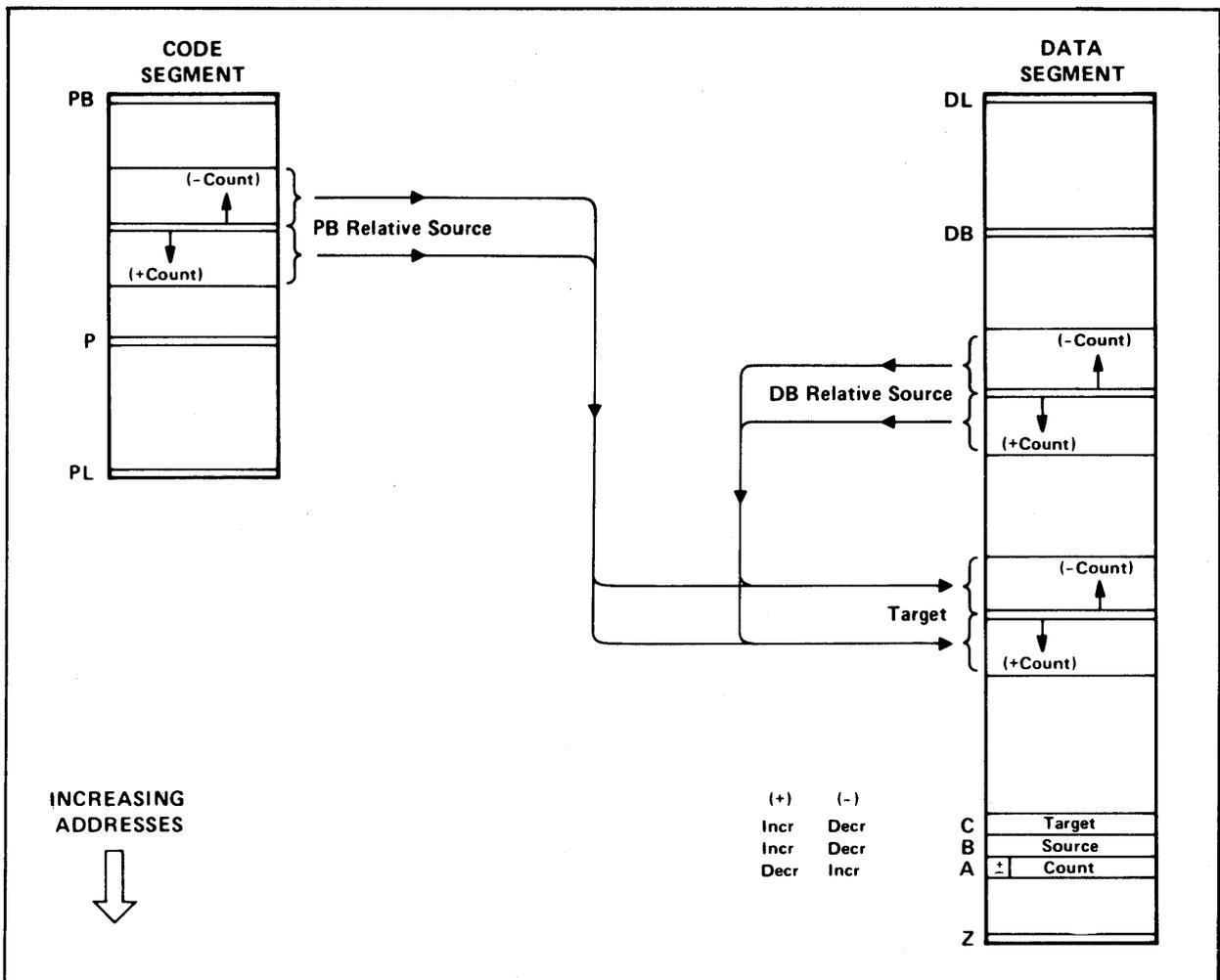
Figure 4-9. Move Examples

ASCENDING/DESCENDING ADDRESSES. The MOVE MVB, and CMPB instructions have the capability of generating ascending or descending addresses for source and target locations. The direction is established by the sign of the count word, which is bit 0 of A, as shown in figure 4-9. If this bit is a "0", the sign is "+", and successive addresses are ascending (B and C incremented). If this bit is a "1" the sign is "-", and successive addresses are descending (B and C decremented). Note the +Count and -Count arrows in figure 4-9. The MVBW instruction uses only ascending addresses; this instruction does not use a count word, and the source and target pointers are in A and B instead of B and C. SCU and SCW also only use ascending addresses; terminal and test characters are in A, the source pointer is in B.

METHOD OF TERMINATION. The MOVE and MVB instructions are terminated only when the word or byte count becomes zero. The MVBW instruction is terminated only when a character of a specified type, either alphabetic or numeric, is encountered. The CMPB instruction has two methods of termination; when the byte count becomes zero, or when any two bytes being compared are unequal. SCU scans until the terminal or test character is found; SCW scans while the string equals the test character.

SPECIAL FEATURES. The MVBW instruction includes an "upshift" bit (bit 13). This bit, when set ("1"), will transpose any lower case source characters to upper case during the transfer. If not set ("0"), the source characters are unaltered by the instruction.

MOVES BEYOND TOS. In the event that the source or target of any move instruction advances into the instruction parameters on the TOS or beyond, the paramters (top four if more than four) will not be affected since these values are contained in the TOS registers. The memory locations directly corresponding to these registers will be used for the move (or comparison). However, this situation is normally a software error.

INTERRUPTS. All Move instructions are interruptable and will continue their operation after return from the interrupt. To do this, the count, source, and target addresses are kept updated and deleted from the stack (if specified) only upon completion of the instruction.

5. SCAL, SXIT. Figure 4-10 illustrates the operations for calling and exiting from a subroutine. Since only local labels may be used, operation is entirely within the current code segment. Assume that the system is executing instructions in the code segment shown in figure 4-10. At some point, P will encounter the "SCAL N" instruction, where N is some value 0 through 255. If the value of N is not 0, e.g., 8, this value will be subtracted from PL (i.e., PL-8), thus pointing at the ninth cell counting backward from PL. This must be within the Segment Transfer Table, whose first entry is PL-1. The eight entry, in this case, contains a local program label (bit 0 = 0), which is a PB relative address pointing to the start of the subroutine. This address is converted to absolute (add to PB) and is loaded into

the P Register, while the former value of P, plue one, is stored in the TOS as the return address. However, if N were 0, it would be assumed that the TOS contains the local label (subroutine starting address). This address, then, (made absolute) would be loaded into the P Register, while the former value of P, plus one, replaces the label on the TOS as the return address. In either case, once the P Register has its new address, the location so indicated will be fetched and subroutine execution begins.
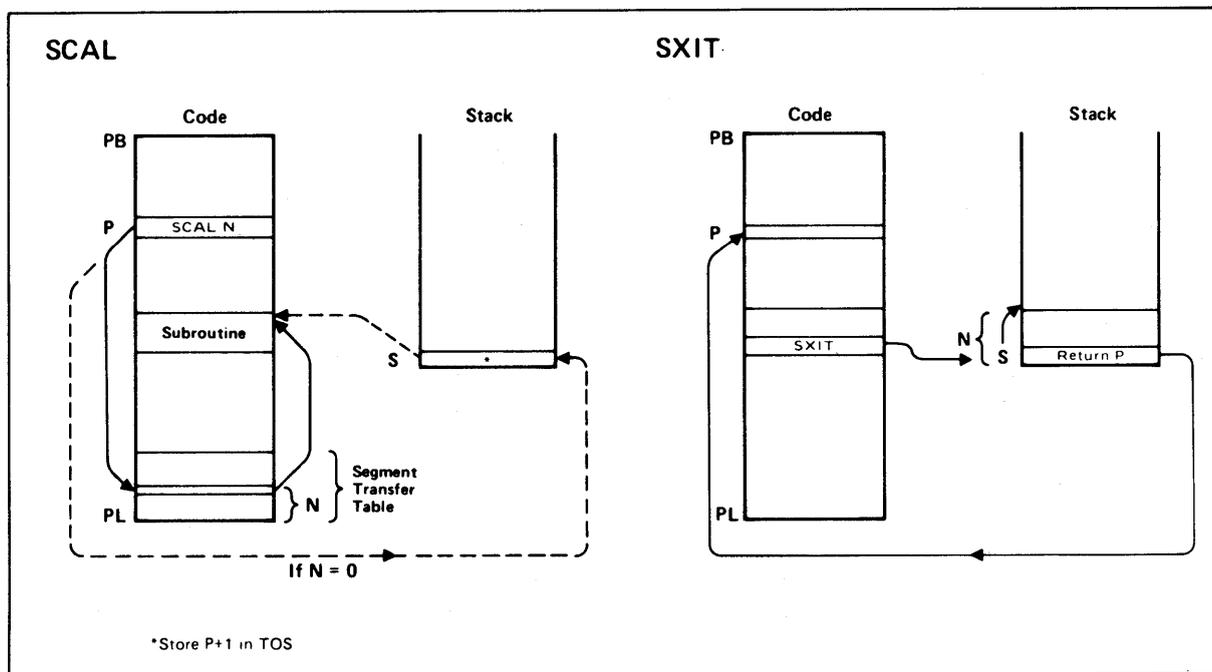
Figure 4-10. Subroutine Call and Exit

The final instruction of the subroutine is SXIT. At this time the return address, pushed onto the stack by SCAL, is assumed to be on the TOS. It is the responsibility of the subroutine to provide this condition, which normally means deleting all variables incurred by the subroutine. The SXIT instruction simply takes the address contained in the TOS and puts it in the P-register, thus effecting a return to the calling routine. As a final step, SXIT deletes the TOS, since the return address is no longer needed, and may additionally move S back some number of locations specified by N. This would typically be used for deleting some of the parameters passed to the subroutine.

6. PCAL, EXIT. These two instructions perform basically the same function as the SCAL and SXIT instructions described above (Instruction Commentary 5). That is, to call a routine and return from it to the point where it was called. However, since the routines in the case of PCAL/EXIT may be external to the current segment, possibly not even present in main memory, the operation is somewhat more complex.

The following paragraphs describe the operations of PCAL and EXIT on a step-by-step basis, referring to flowcharts. It will frequently be assumed that the reader has a working knowledge of the intents and purposes of the various steps.

PCAL Sequence. Figure 4-11 illustrates the operations of the PCAL instruction. If the call is within the current segment (local label), only the steps shown on the left side of the diagram are performed. For calls outside the current segment, the steps on the right side are added.

The first step is to fetch the program label. From the PCAL instruction definition, we see that the label can be obtained from one of two places: from the TOS if N is zero, or from PL-N if N is not zero. This operation can be seen in the SCAL operation of figure 4-10, where the label is fetched from either the Segment Transfer Table, at PL-N, or from the TOS.

Thus, referring to figure 4-11, PCAL initially checks N to see if the label is on the TOS. If not (block 1), the label is fetched from PL-N and a check is made to see if that location is actually within the bounds of the Segment Transfer Table. (N must be <STTL value in the PL location.) If out of STT bounds, an STT violation is incurred; otherwise, the PCAL sequence continues. If the label is on the TOS (block 2), the label is put into temporary storage in the CPU and S is decremented to delete the label from the stack. At this time, the CPU has the label but does not know whether it is local or external, or if it is valid.

The next step is to place a standard four-word stack marker onto the stack (block 3) and update the Q pointer by loading it with the content of S (block 4). Both Q and S are now pointing at the last word (Delta Q) of the new stack marker.

Now the label is checked to see if it is a local label (bit 0=0). If it is, the sequence goes to block 8 (skip to paragraph starting "Block 8 sets").

If the label is external (bit 0 = 1), bits 8 through 15 are checked to see if the segment number specified is valid. If the segment number does not have an entry in the Code Segment Table, a CST violation is incurred. Otherwise, the PCAL sequence continues. Next, absolute addresses for PB and PL are calculated from the CST entry and loaded into these two registers (block 5).

Block 6 sets the privileged mode in the Status Register if the mode bit in the CST entry indicates privileged mode, or if the caller was executing in privileged mode (i.e., if the privileged mode bit in Status already was set). (Although not shown, the Reference bit in the CST is set at this time for statistical purposes.)
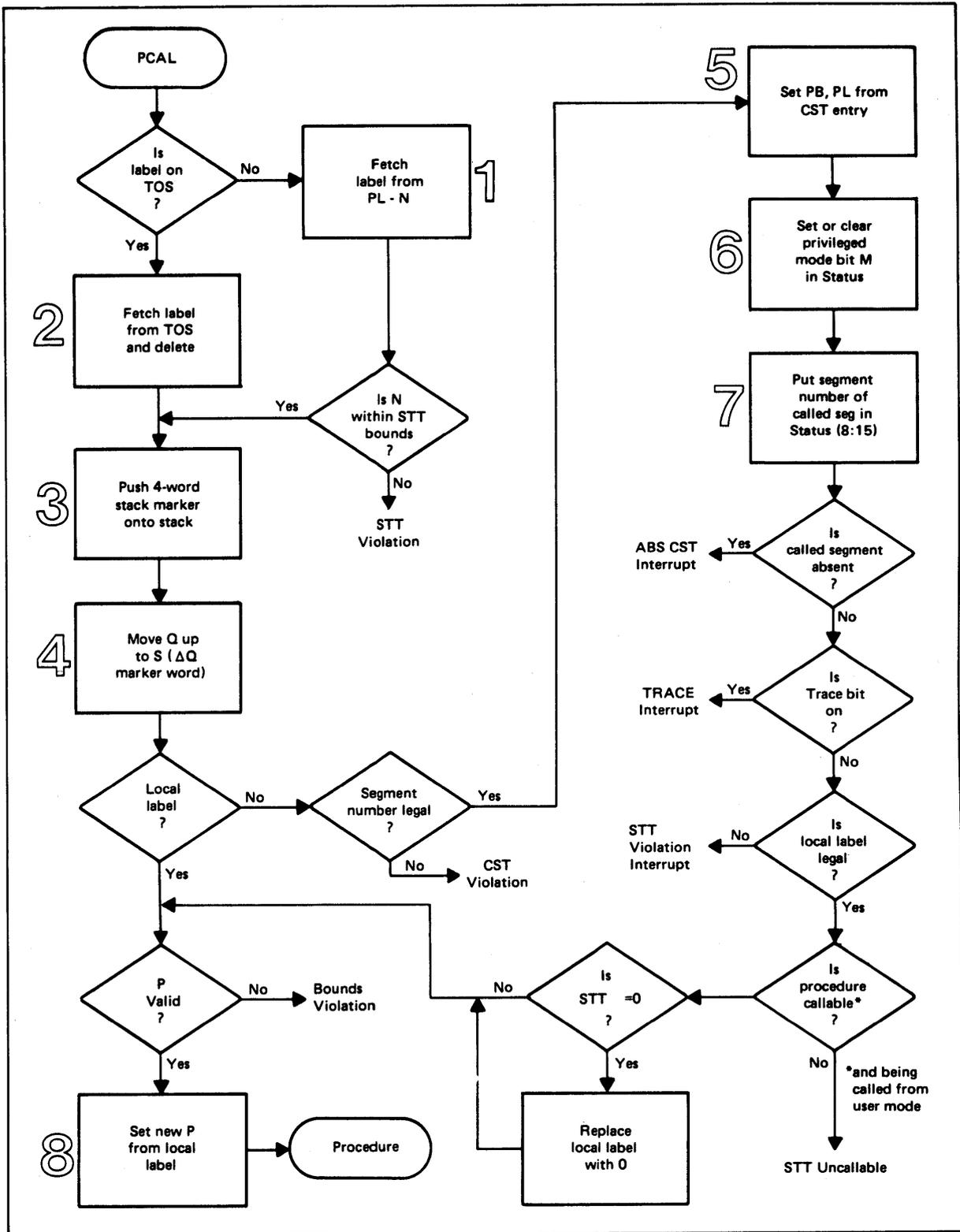
Figure 4-11. PCAL Instruction Flowchart

Block 7 stores bits 8 through 15 of the label into bits 8 through 15 of the Status Register. This indicates to the system that we are now operating in the called segment. A check is then made to see if the called segment is absent from main memory. If it is, an absent code segment trap is incurred. A similar check is made for TRACE by checking the CST entry for the called segment.

The next check is to see if bits 1 through 7 of the label are 0. These bits specify which STT entry in the target segment contains the desired local label. Since a value of 0 would point at the STTL word in PL, the value of 0 is specially defined to indicate that P should be set to PB of the called segment; i.e., the local label equals 0. A check is then made to see if the PB entry is callable if it is being called from user mode. Assuming that bits 1 through 7 of the external label are not 0, the value so indicated will point to one entry in the Segment Transfer Table. If it does not (i.e., if the value exceeds the STTL value), or if the entry pointed to is not a local label (i.e., if bit 0 = 1), there will be an STT Violation. But if the label is valid, it is then checked to see if the procedure is callable if being called from user mode by checking bit 1 (must be 0).

Block 8 sets the P Register to the starting address of the procedure. The CPU at this point has a local label, whether it is in the same segment as the PCAL or in a segment external to the calling segment. The value for P is calculated by adding the contents of bits 2 through 15 of the local label to the contents of PB. As a final check, this value for P is checked to see if it is between PB and PL. The resultant absolute value is then loaded into the P Register, and the location so indicated is fetched and execution of the procedure begins.

EXIT Sequence. Figure 4-12 illustrates the operation of the EXIT instruction. If the exit is within the current segment, only the steps on the left side of the diagram are performed. For returns to another segment the right side is also executed.

The first step (1) is to fetch the 4-word marker pointed to by Q, which was placed on the stack when the current procedure was called. S is set equal to Q, deleting any local storage being used by the current procedure. If the current procedure is executing in user mode, the privileged and external interrupt enable bits in the marker status are compared with the current status to ensure that the user has not modified these in the marker. Then X is restored from the marker.

In step 2, if the current segment and the segment in the marker are the same, steps 3 through 6 are omitted; otherwise continue.

Steps 3 and 4 are similar to the equivalent steps in PCAL (figure 4-11).
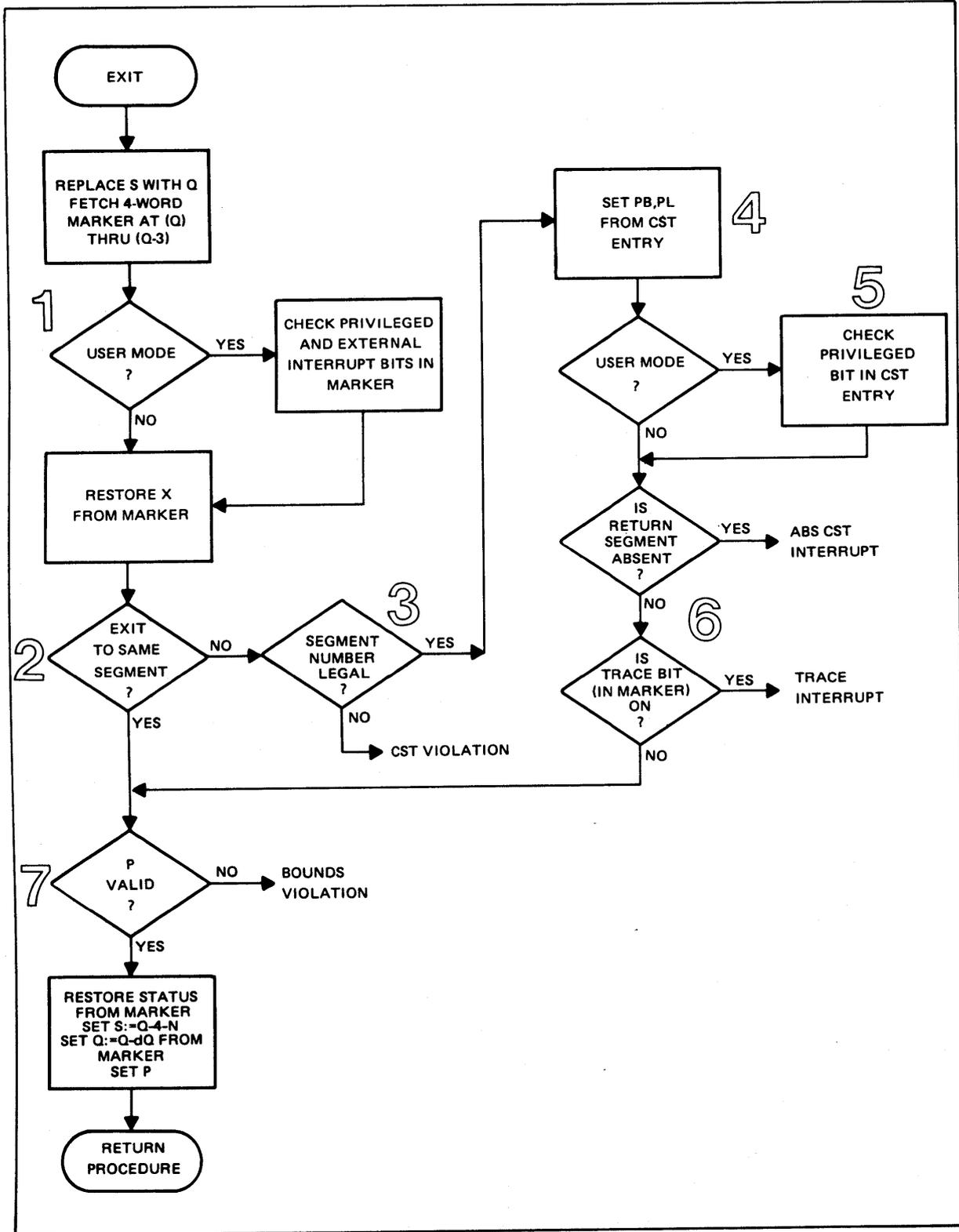
Figure 4-12. EXIT Instruction Flowchart

In step 5, if in user mode, the privileged bit in the CST entry for the return segment must be off. (Although not shown, the reference bit in the CST entry is set at this time for statistical purposes.)

An absent code segment trap occurs following step 5 if the return segment is absent. A trace trap occurs in step 6 if bit 0 of Delta P in the marker is set. This bit is normally set by the trace routine which would have been called when the current procedure was entered.

At step 7, return "P = P-Delta P" from the marker must be between PB and PL. The Status Register is restored from the marker; Q is set pointing to the previous marker, then S is decremented by 4 to delete the marker on the top of the stack and by N (specified in the EXIT instruction) to delete any parameters passed to the procedure being exited. P is set to return P and execution begins within the return procedure.

7. DISP, IXIT, PSDB, PSEB. The Dispatcher, external interrupts, and some internal interrupts execute on the Interrupt Control Stack (ICS). Normally the Dispatch (DISP) instruction is used to enter the Dispatcher and the Interrupt Exit (IXIT) is used to exit from the Dispatcher. Also, when "ICS" type interrupt service routines are entered in response to appropriate events, the instruction IXIT is used to exit from these. The exit may be from the Dispatcher to the process being launched or from interrupt service routines to the interrupted procedure or, in certain cases, to the Dispatcher entry point. The instructions Pseudo Interrupt Disable (PSDB) and Pseudo Interrupt Enable (PSEB) are used to prevent entry to the Dispatcher during critical sections of code.

The instruction DISP causes a transfer to the Dispatcher's entry point unless it is executed while on the ICS or while the Dispatcher is disabled. The Dispatcher is disabled when the Dispatcher Flag is non-zero, (QI-18) not 0. The address of QI is located at 4 times the CPU number plus 1. Condition code CCE is set when the Dispatcher is entered; the Status Register is set as specified for the Dispatcher. The transfer is executed in a manner similar to an ICS interrupt. If a DISP instruction is executed on the ICS or while the Dispatcher is disabled, bit 0 of (QI) is set and CCG is set in the Status Register. This bit is checked by those instructions (IXIT and PSEB) which may remove the conditions inhibiting the Dispatcher.

The instruction PSDB increments (QI-18); PSEB decrements (QI-18). Starting the Dispatcher is disabled unless this location is zero. Outside the Dispatcher and not on the ICS, a PSEB which decrements (QI-18) to zero effectively does a DISP instruction if bit 0 of (QI) is set.

Within the dispatcher, a PSEB which decrements (QI-18) to zero, clears (QI) eliminating any pending Start Dispatcher requests. PSDB and PSEB are used at the beginning of the Dispatcher to prevent any interrupts which request a dispatch from causing the first portion of the Dispatcher to be unnessarily repeated. PSEB instructions which do not transfer to the Dispatcher set CCG in the Status Register.

Figure 4-13 is a simplified flowchart of IXIT operation. IXIT operates in one of two manners. The first, (1) in the figure, is by the dispatcher to transfer to a process being launched; the second, (2) through (6), is to exit from ICS interrupt service routines.

If an interrupt service routine is not in segment #1, it is assumed to be an external interrupt routine and a "Reset Interrupt" is sent to the device whose device number is at Q+3. (Q+3) is assumed to be valid in memory, which is normally the case since the device number supplied to external interrupt routines as a parameter is written into memory.

If bit 0 of (Q) is zero, (Q(0))= 0, then if Q=QI, the return is to the interrupted process (2). Otherwise the return is to a lower priority interrupt which was interrupted (3).

If (Q(0))= 1 and (QI(0))= 0, the return is to the Dispatcher which was interrupted (4).

If (Q(0))= 1 and (QI(0))= 1, a DISP instruction has been executed and the request to start the Dispatcher is still pending. If (QI-18)= 0, the Dispatcher is not disabled, QI is cleared, and a transfer is made to the Dispatcher's entry point (5) or (6). It doesn't matter whether a process, Q=QI, or the Dispatcher, Q not =QI was interrupted. If (QI-18) not 0, starting the Dispatcher is disabled and the DISP request cannot be carried out at this time. Instead IXIT returns to the interrupted Dispatcher, Q not =QI(4a), or to the interrupted process, Q=QI (2a). The "Start Dispatcher" request is still pending, (QI(0))= 1.

8. SIO. The I/O instructions in the HP 3000 Command System instruction set are as follows:

    SIO   Start I/O
    RIO   Read I/O
    WIO   Write I/O
    TIO   Test I/O
    CIO   Control I/O

Additional information for these instructions is contained in Section VII. The distinction to note here is that the SIO instruction is used in conjunction with an I/O program and the others are not. That is, the SIO instruction commands a device controller to begin executing its associated I/O program, which effects a block transfer of data between an I/O device and memory. This is termed as "SIO transfer" mode. The other instruc-
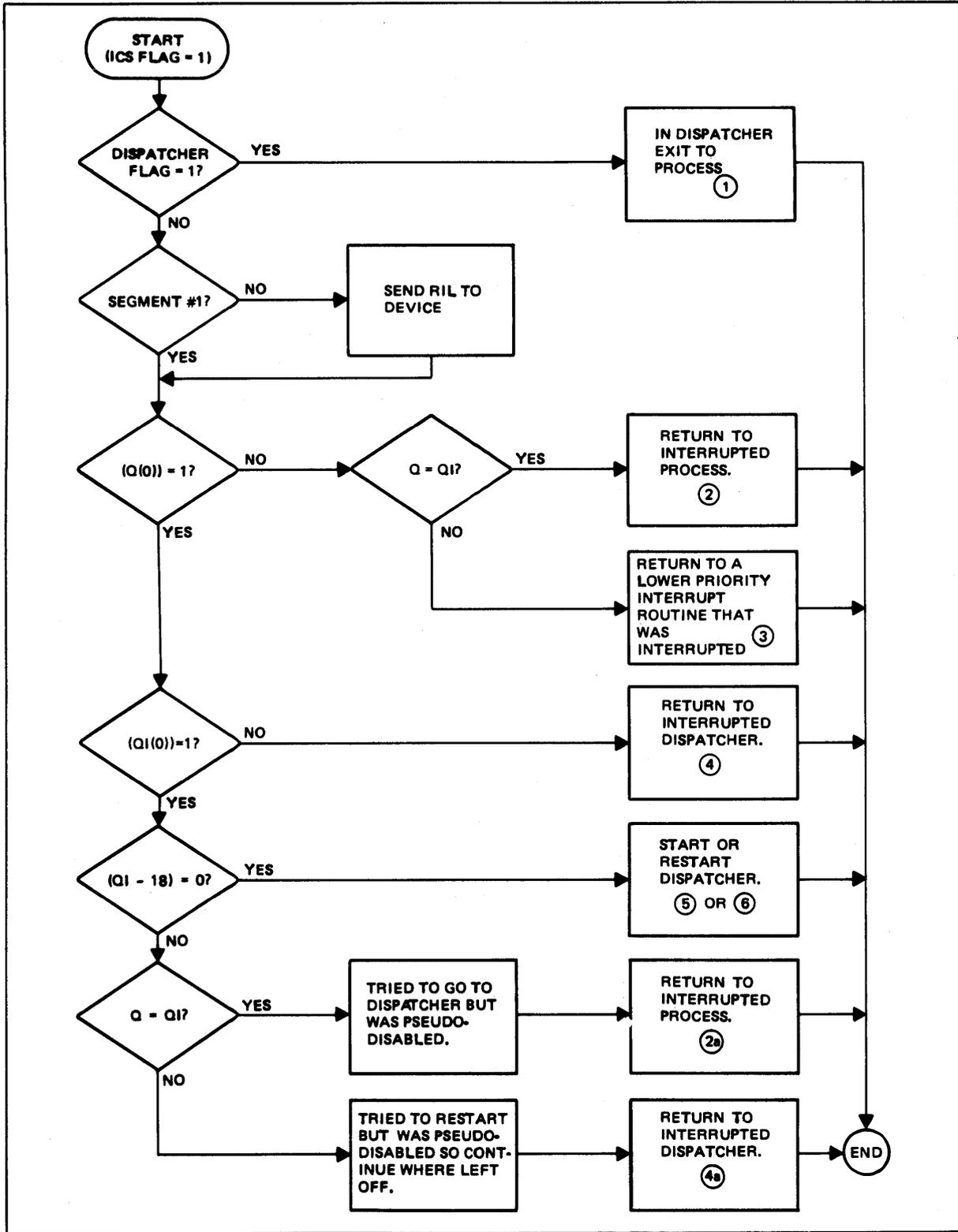
Figure 4-13. IXIT Instruction Flowchart

tions, on the other hand, transfer only one word per instruction, between the device and the TOS in the CPU.

An SIO type data transfer is initiated by the CPU executing a Start I/O instruction for a particular device. The instruction assumes that there is an I/O program stored in main memory. The hardware I/O system executes the I/O program independently of the CPU. The CPU is then free to continue processing in parallel with the I/O operations.

Figure 4-14 illustrates the order pair format of the double words which are used in I/O programs. The general format is shown at the top of the figure and then the actual format of each of the nine orders is shown beneath. The first word of an order pair is designated as the I/O Command Word, or IOCW, and the second word is designated as the I/O Address Word, or IOAW. The IOAW does not necessarily always contain an address, as the figure shows.

The nine I/O orders are defined as follows:

JUMP. If bit 4 of the IOCW is a "1", a conditional jump of I/O program control is made to the address given by the IOAW at the discretion of the device controller. If bit 4 of the IOCW is a "0", an unconditional jump is made.

RETURN RESIDUE. This causes the residue of the count to be returned to the IOAW. The residue is obtained from the Multiplexer or Selector Channel. Each Multiplexer or Selector Channel has its own count. The count is initialized from the least significant 12 bits of all IOCWs except Return Residue and Set Bank.

SET BANK. This instruction loads the Bank Register of the Multiplexer or Selector Channel with bits 12 through 15 of its IOAW. The execution of an SIO instruction automatically clears the Bank Register. Therefore, if the data buffer for this device resides in some bank other than 00, the I/O program must contain a SET BANK order prior to a READ or WRITE order.

INTERRUPT. This order pair causes the device controller to set its Interrupt Request flip-flop and, therefore, to interrupt the CPU.

END. End of the I/O program. If bit 4 of the IOCW is a "1", the device controller also interrupts the CPU. Returns device status to the IOAW.

CONTROL. This causes transfer of a 16-bit control word in the IOAW to the device controller, as well as the 12 low order bits of the IOCW.

SENSE. This causes transfer of a 16-bit status word from the device controller to the IOAW.
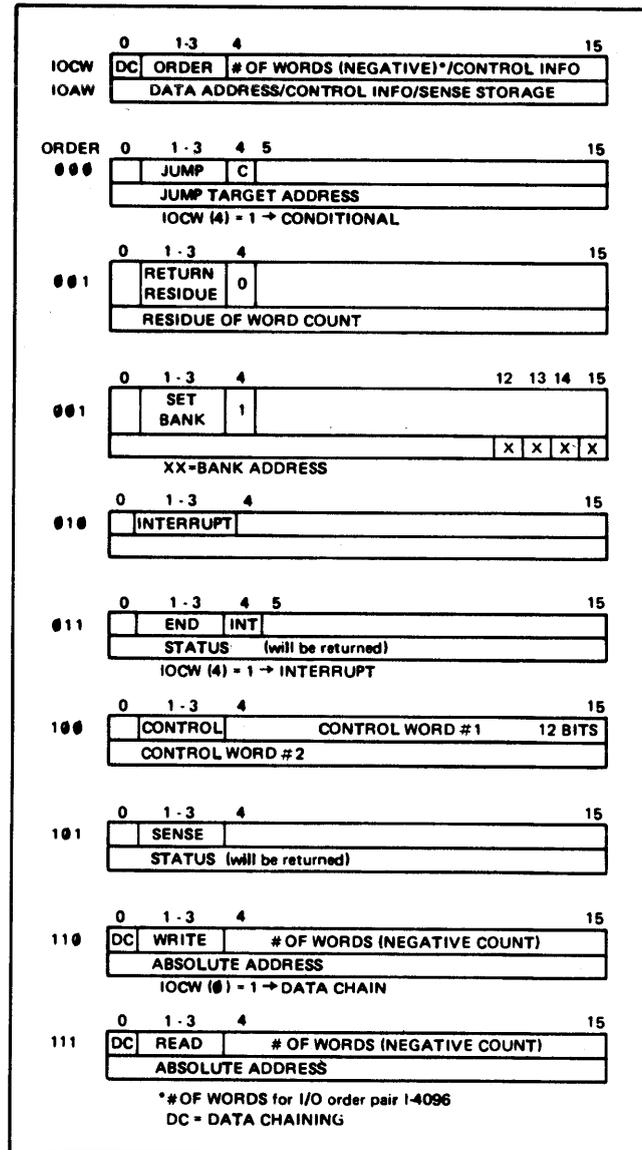
Figure 4-14. I/O Order Pairs

WRITE. This causes "count" words of data to be transferred between main memory and the device, starting at the address given by the IOAW, within a given bank.

READ. This causes "count" words of data to be transferred between the device and main memory, starting at the address given by the IOAW, within a given bank.

Data chaining occurs for Write and Read orders if bit 0 of the IOCW is a "1". This bit may be a "1" for a Write order followed by a Write or for a Read order followed by a Read.   This will permit the hardware to treat the counts of each order as a continuous chained count, without re-initializing for each order. The DC bit should be "0" for all other orders.

The count field for Read and Write orders contains the least significant 12 bits of a negative two's complement count value. The count is a word count, independent of the particular recording format (bytes, words, or records). For a Control order, these 12 bits are used for control information in addition to the 16 bits in the IOAW (a total of 28 bits).

4-17. STACK OPERATION EXAMPLES

4-18. Basic Arithmetic

Figure 4-15 shows a sequence of basic instructions being executed on some data which is presumed to exist in the stack. The upper row shows the most elementary method of adding and removing data to and from the stack via load and delete instructions. The lower row shows the effects of four arithmetic instructions. As shown for the initial stack condition (A), the data consists of six numbers in six consecutive locations. The Q Register points to the oldest element of the group, and S points to the element currently on the TOS. A Delete instruction (DEL), executed between (A) and (B), causes the number 44 to be removed from the stack. This is accomplished by simply decrementing the S-pointer by one. Then, between (B) and (C), a LOAD instruction causes the number 37 to be loaded onto the stack. This is accomplished by storing the number 37 (from another memory location) into the location formerly occupied by the number 44 and then incrementing the S-pointer by one.

Between (C) and (D) an ADD instruction is executed. This instruction adds the two top elements of the stack together, deletes both from the stack, places the answer (100) on the TOS, and points S at the answer.

Note

As previously discussed, up to four of the top stack elements may exist in CPU registers. Obviously, to execute the ADD instruciton, at least the two top elements must exist in the CPU. To ensure that this is the case, the hardware checks the contents of the SR Register. If the contents of the SR Register is not at least two, one or more memory fetches will be made so that the instruction can be carried out.

Between (D) and (E) a Multiply instruction (MPY) is executed. This instruction multiplies the two top elements of the stack together, deletes both from the stack, places the answer (700) on the TOS and points S at the answer.
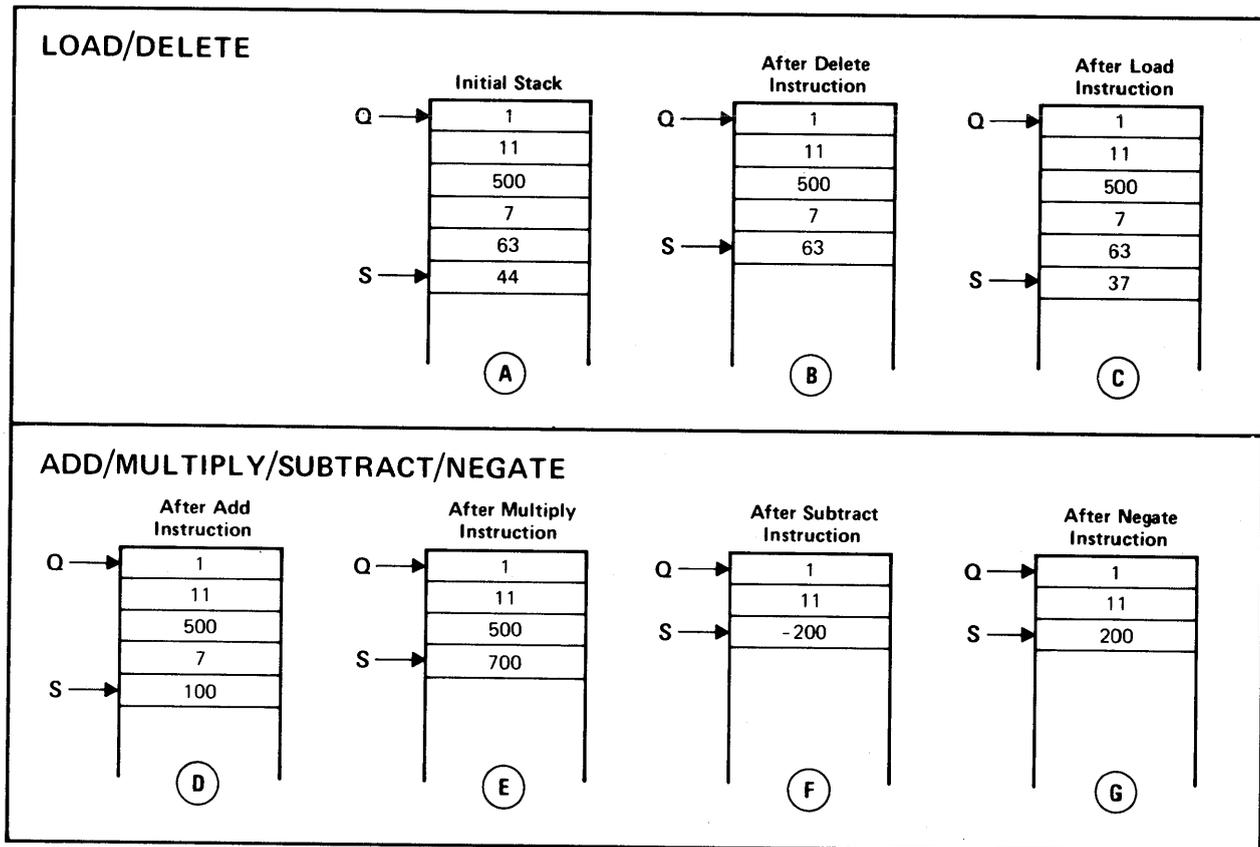
Figure 4-15. Basic Arithmetic Stack Operations

To subtract (SUB), the top element is subtracted from the next-to-top element. Thus the answer at (F) is the result of 500-700, or -200. (As before, only the answer remains after computation is performed.) Finally, at (G), negation is performed. This simply reverses the sign of the number of the TOS. In binary form, a two's complement operation is performed.

Although the sequence (A) through (G) in figure 4-15 is a very simple series of operations, it does illustrate the advantages of the stack technique in computation. First, note that regardless of how many elements of data there are or what memory cells they occupy, the operand for each instruction is consistently the same ; the TOS. This permits implicit addressing; i.e., since the operand is understood to be the TOS, it is not necessary to give an operand address in the instruction word. Thus (except for LOAD which must specify a relative address to load from), the instruction can simply say "add", or "multiply", etc. The immediate benefit of this is that it allows code compression. Two instructions can be given in a single word. The sequence (D) through (G) for example, can be given in two instruction words. Since this reduces the number of memory fetches, the speed of computation is considerably increased. A second point to note is that temporary storage of intermediate results is automatically provided. For example, once the parameters 63 and 37 (C) have been added, they are no longer required and are deleted. The answer however, which is substituted on the TOS, is automatically

in position (adjacent to 7) for the ensuing multiplication. Therefore, there is no need to provide a dedicated location to save the temporary quantity 100 or any of the other intermediate results.

4-19. Procedure Calls

Figures 4-16 and 4-17 illustrate the operations involved in a procedure call. Figure 4-16 shows programmatically how a procedure is set up and called, and figure 4-17 shows what happens to the stack when the procedure is called and executed. As shown in the bottom block of figure 4-16, the calling of a procedure has an equivalency in mathematical terms. That is, a procedure is like a request to solve the equation for the specific values of 25 for J and 10 for K. Executing the procedure is to perform the computation; in this case getting an answer of 2. (To keep things simple, the example procedure will be made to work strictly with integer numbers; thus the fractional remainder 5/10 will automatically be discarded.) The upper two boxes in figure 4-16 list two forms of the program that will accomplish the example procedure. The top box shows how the program is written in the source programming language. The middle box shows the machine language code that will be emitted by the compiler. The machine language code is shown both in assembly or mnemonic form, and in an octal form of the actual binary machine code.

As shown in figure 4-16, line 1 begins the source language program block and line 9 ends it. Although the entire program consists of only one procedure and a call to that procedure, it is necessary to enclose the program between a BEGIN statement and an END statement. These statements define a program. ANSWER is declared to be a global variable for this program by giving its name within the BEGIN statement. This will cause the variable ANSWER to reside in the global data area and thus allow its access by another procedure; such as an output routine to print out the result. The type declaration INTEGER specified that ANSWER will always be an integer and tells the compiler to reserve one word for the result rather than two or three. ANSWER is allocated the word at DB+0. Lines 2 through 7 comprise the procedure declaration which includes the procedure head (lines 2, 3, and 4) and the procedure body (lines 5, 6, and 7). The procedure declaration in a program cannot cause execution by itself, but it must be called before any execution can take place. Therefore, the procedure declaration is always separate and distinct from the procedure call. They need not be immediately adjacent, however, as in this example. Line 2 gives the procedure name, QUO-TIENT, and declares that the procedure is of type INTEGER. This means that the result will be in integer form. It also gives the names of the formal parameters, J and K. Line 3 is the value part of the procedure declaration. Declaring J and K as values means that a value (rather than a pointer) will be passed as a procedure parameter in both cases. This permits working with a copy and eliminates any need to change the actual parameter. Line 4 declares that actual parameters for J and K must be integers. If any other type is given (e.g., floating point), a com-

SOURCE LANGUAGE

	1	BEGIN INTEGER ANSWER;

Pro-cedure	2	INTEGER PROCEDURE QUOTIENT (J,K);
	3	VALUE J,K;
	4	INTEGER J,K;
	5	BEGIN
	6	QUOTIENT ⟵ J/K;
	7	END;
Call	8	ANSWER ⟵ QUOTIENT (25,10);
	9	END;

MACHINE LANGUAGE

		Assembly	Octal
Pro-cedure	10	LOAD Q-5	041605
	11	LOAD Q-4	041604
	12	DIV, DEL	002340
	13	STOR Q-6	051606
	14	EXIT, 2	031402
Call	15	ZERO, NOP	000600
	16	LDI, 31	021031
	17	LDI, 12	021012
	18	PCAL, 20	031020
	19	STOR DB+0	051000
	20	PCAL (to system)	031xxx

MATHEMATICAL LANGUAGE

Procedure: ANSWER = J/K

Call: Solve ANSWER for
J = 25 and K = 10

Execution: ANSWER = 25/10
= 2, remainder 5

Note: Decimal 25 = Octal 31
Decimal 10 = Octal 12

Figure 4-16.   Declaring and Calling A Procedure

pilation error will result.   Line 5 begins the procedure body.
Actually, since this procedure consists of only one statement,
the BEGIN statement and END statement (line 7) are superfluous.
They are included here, however, to illustrate the common form
for a procedure (normally involving a compound statement). Line

6 is the procedure statement which is the executable part of the procedure body. This is the statement that will cause the division of J by K and temporarily store the quotient as a procedure result, identified by the procedure name QUOTIENT. The call to the procedure is given at line 8. This is an executable statement as opposed to a procedure declaration. When this statement is encountered in the program, it causes the procedure named QUOTIENT to be executed, passing actual parameter of 25 and 10 to the procedure, and causes the global variable ANSWER to assume the value of the result. This completes the program.

Lines 10 through 19 show the machine language code that the compiler emits for the two executable statements in the program (i.e., line 6 causes line 10 through 14 to be generated and line 8 causes lines 15 through 19 to be generated). In order to explain the operation of the program in machine language, it is necessary to examine what is happening on the stack. It is assumed that the user has logged onto the system, has compiled the program, and is ready to run (or is running a program that will shortly encounter the statement in line 8). Loading the program causes space to be allocated for the one global variable, AN-SWER, which is at DB+0 as shown in (A), figure 4-17. Since there are no other global variables, Q and S initially point at the immediately following location. (The content of that location will never be signifiant; in essence it is a dummy Delta Q location.) Additionally, during program loading, the operating system evaluates the program in order to set the Z Register appropriately for an initial estimated stack size. Also, since no dynamic arrays are declared, DL is set coincident with DB, therefore DL is not shown. (Refer to paragraph 2-28.)

It is assumed that the user has issued a system command to execute the procedure call given in line 8 of figure 4-16. This causes control to be passed to line 15 in the machine language program where the sequence to call the procedure begins. The first instruction is a ZERO,NOP. Executing this instruction puts a "0" on the stack and increments the S-pointer, (A), figure 4-17. This reserves a location for the procedure result. Next at (B) and (C) (lines 16 and 17), the parameter values 31 and 12 are passed directly from the instruction words to the stack (area reserved for procedure parameters). Octal notation is used for these values. Then at (D), a procedure call instruction (PCAL) causes a four-word stack marker to be placed on the stack. The S- and Q-pointers point to the Delta Q location of the marker which now indicates 7 (the number of locations back to the initial Q location). It is assumed that entry number 20 in the Segment Transfer Table will direct the call to the correct procedure starting point. (Refer to paragraph 2-24.)

Execution of the procedure now begins (line 10). The first two instructions (lines 10 and 11) load copies of the procedure parameters onto the TOS (E) and (F), using Q-relative addressing. The next instruction (line 12) divides the top-of-stack parameter into the next-to-top parameter and substitutes the quotient "2" and the remainder "5" on the TOS as shown at (G). The second

CALLING THE PROCEDURE

Stack After ZERO Instruction

DB → (Answer)
Q →
S → 0

(A)

After LDI 31

DB →
Q →
0
S → 31

(B)

After LDI 12

DB →
Q →
0
31
S → 12

(C)

After PCAL 20

DB →
0
31
12
X
Return P
Status
S,Q → 7

(D)

EXECUTING THE PROCEDURE

After LOAD Q-5

DB →
0
31
12

Q →
S → 31

(E)

After LOAD Q-4

DB →
0
31
12

Q →
31
S → 12

(F)

After DIV

DB →
0
31
12

Q →
2
S → 5

(G)

After DEL

DB →
0
31
12

Q →
S → 2

(H)

SAVING PROCEDURE RESULTS

After STOR Q-6

DB →
2
31
12
X
Return P
Status
S,Q → 7

(I)

After EXIT 2

DB →
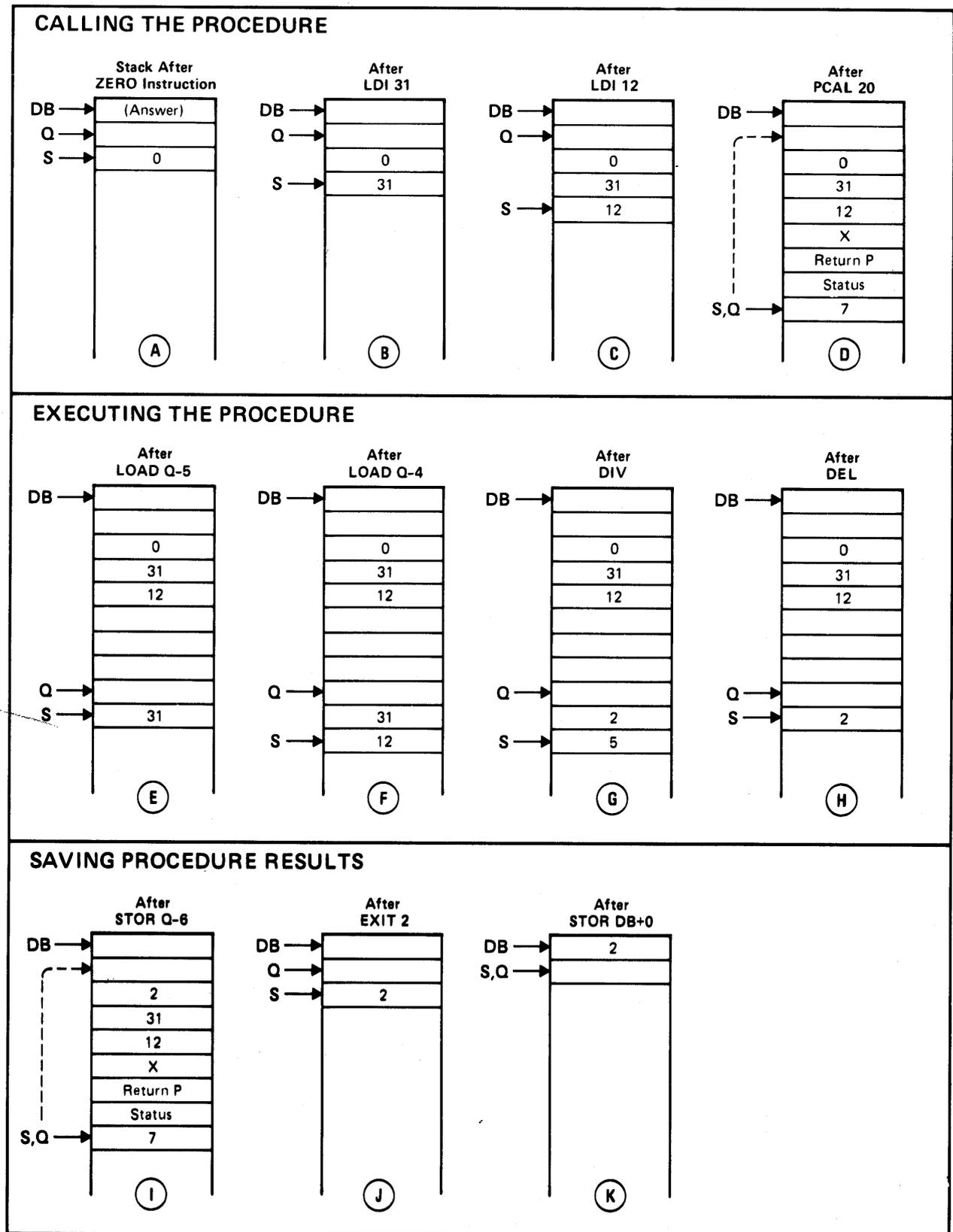Q →
S → 2

(J)

After STOR DB+0

DB → 2
S,Q →

(K)

Figure 4-17. Executing A Simple Procedure

half of the same instruction (DEL) discards the remainder word by decrementing S as shown at (H). To save the result, the STOR Q-6 (line 13) first copies the TOS into the location reserved for the procedure result, formerly occupied by a 0, as shown at (I). Then it is possible to exit from the procedure. The EXIT instruction (line 14) restores Q to its initial setting, and the "2" included with the instruction causes S to move back two locations past the stack marker. As shown at (J), this leaves the result, 2, in the location reserved for QUOTIENT (now on the TOS). The EXIT returns program control to line 19 which causes the content for QUOTIENT to be stored in the location for ANSWER in the global data area. This produces the final result shown at (K). Finally (line 20), a procedure call to the system returns control back to the system.

4-20. Recursion

This example demonstrates the stack principles involved in a recursive procedure. A recursive procedure is one which calls itself one or more times during execution. The form of the source language program for this example (figure 4-18) is nearly identical to that of the preceding example in figure 4-16. The procedure simply computes N! (N factorial), where N is the formal parameter. The procedure will be called with an actual parameter of 4 so that computation of 4! will be: 1 x 2 x 3 x 4 = 24.

This problem consists of repetitively multiplying the previous product by a parameter which is incremented by one on each repetition. To provide a starting point (initial previous product), the value 1 is automatically given. The procedure is designed to perform this multiplication sequence by repetitively calling itself after it has been called once by the main program. Thus for any N, the procedure will be called N+1 times. In this example there will be one call by the main program and four recursive calls. Figure 4-18 lists the source and machine language forms of a program block to solve this problem. Since the source language program is similar to the preceding example, it need not be discussed at this point. The machine language form has been slightly changed to more closely resemble an actual program listing. Some assumed PB-relative addresses are given for each instruction, beginning at address 00114. The assumption is that this program block is embedded in a larger main program. (Note that the assigned STT entry for this procedure is assumed to be 026 and that the global assignment for Y is DB+15.) The starting point for execution is at address 00130.

Figure 4-19 illustrates the program in flowchart form. Box 1 in the diagram calls the procedure (boxes 2 through 9), box 10 saves the result, and then control reverts to the main program at box 11. The procedure consists of two phases. The call phase begins when the procedure is called by the program and is repeated four times. In this phase, N values are placed on the stack along with a space for intermediate answers. The N values are decremented to zero and then the exit phase begins. This phase successively multiplies an accumulating product by each of the N

4-43

SOURCE LANGUAGE

```
        .
        .
        .
    BEGIN INTEGER Y;
        INTEGER PROCEDURE FACTORIAL (N);
        VALUE N;
        INTEGER N;
            FACTORIAL := IF N = 0 THEN 1 ELSE N * FACTORIAL (N-1);
        Y := FACTORIAL (4)
    END;  .
        .
        .
```

MACHINE LANGUAGE

PB Relative Addresses	Instructions	Octal Code	Comments
00114	LOAD Q- 004	041604	Load parameter
00115	CMPI, 000	022000	Test it for zero
00116	BNE P+ 003	141503	If not zero, branch to 00121
00117	LDI, 001	021001	If zero, load 1 as initial multiplicand
00120	BR 006	140006	Branch to 00126 (to Exit loops)
00121	ZERO, NOP	000600	Save space for intermediate product
00122	LOAD Q- 004	041604	Load parameter
00123	SUBI, 001	023001	Decrement for use as new parameter
00124	PCAL, 026	031026	Recursive call
00125	MPYM Q- 004	111604	Multiply parameter by TOS
00126	STOR Q- 005	051605	Store this recursion's product
00127	EXIT, 001	031401	Save the product and exit
00130	ZERO, NOP	000600	Save space for final product
00131	LDI, 004	021004	Load initial actual parameter
00132	PCAL, 026	031026	Main program's call to the procedure
00133	STOR DB 015	051015	Save final product in global area
00134	PCAL, XXX	031xxx	Return to system

Figure 4-18. Recursive Program

values loaded on the stack in the call phase, in the reverse or-
der. On each loop, unneeded stack information is deleted, saving
only the answer for that loop, until only the final answer is
left. At that time (box 9) the final EXIT instruction finds that
its return address points back to the calling block and the final
answer is stored in the global area. Control then reverts to the
main program. As will be shown in the following detailed discus-
sion, the return address check at box 9 is not literally a test
for a specific address. Rather it specifies a return to the ad-
dress given in each stack marker. Obviously, the last return
(first one placed on the stack) will be a return to the outer
block. Figures 4-20 and 4-21 show the overall process of build-
ing up the stack by recursive calls, and then reducing it with
recursive exits. These two figures are used in the following
discussions. Also, the machine language program in figure 4-18
will be referred to. Individual lines will be identified by PB-
relative addresses, omitting the leading zeros.

Figure 4-19.   Recursive Procedure Flowchart

4-21. MAIN PROGRAM CALL. As in previous examples,the main program has already reserved global space for the final answer (Y) before the procedure is called. When the call is given, the ZERO,NOP instruction at address 130 reserves space for the procedure result, FACTORIAL. (Compare stacks (a), figure 4-20 and (Z), figure 4-21. This is the first stack addition due to calling the procedure. Next, the actual parameter 4 is loaded on (B), and then the PCAL instruction is issued. This causes the first stack marker to be loaded (C). This marker differs from the ones that follow in that is contains return information to the outer block which called the present procedure (i.e., the return P word is a P-relative address for return to the caller in the code segment and Delta Q points back to the Q value that the caller was using earlier in the stack). Now, S and Q are both pointing at the last word of the first marker for this procedure.

4-22. TEST FOR ZERO. At addresses 114 and 115 (stack (D) and (E), figure 4-20), the procedure parameter is first tested for zero. This is done by copying it onto the top of the stack (LOAD Q-4) and a CMPI 0 instruction. This instruction sets the condition code according to comparison results and deletes the tested word (E). Since the first test is non-zero (i.e., 4), the branch instruction at line 116 transfers control to address 121 (i.e., P+ 4). This test and branch will be repeated in each of the following recursion loops until the parameter has become zero.

4-23. FIRST RECURSIVE CALL. The branch to address 121 causes the procedure to call itself. As usual, the first action of the call is to load the procedure parameters onto the stack. The parameters in this case are the variable FACTORIAL and a decremented form of the original passed parameter. Thus the ZERO, NOP instruction reserves a location for FACTORIAL (F), figure 4-20 strictly for use by this recursion (i.e., distinct from the final FACTORIAL location reserved at (A). Then (G and H), the new parameter, is obtained by copying the preceding value to the top of the stack (LOAD Q-4) and decrementing with a SUBI 1 instruction. After loading parameters for the new call, another PCAL instruction is issued. This causes a new stack marker (I), figure 4-20 and, via the Segment Transfer Table, control is transferred back to the starting point of the procedure at address 114. The new stack marker gives as its return P value the address immediately following the PCAL which is 125. (This will be important to remember when the exit sequence is discussed.) Also, the Delta Q value is 6 since the previous Delta Q was six locations back.

4-24. SUCCESSIVE RECURSIONS. Next, all the previously described steps are repeated, beginning with paragraph 4-22. Since the parameter is 3 on the second recursion, the branch to address 121 again occurs. The first actions, again, are to reserve a location for this recursion's answer (J), figure 4-20 and to load a decremented parameter value of 2 (K) and (L). After this, the procedure call back to the beginning is made again which results in another stack marker (M) that is identical to the one generated on the first recursion. The third and fourth recursions repeat the entire process again, loading parameters of 1 and 0
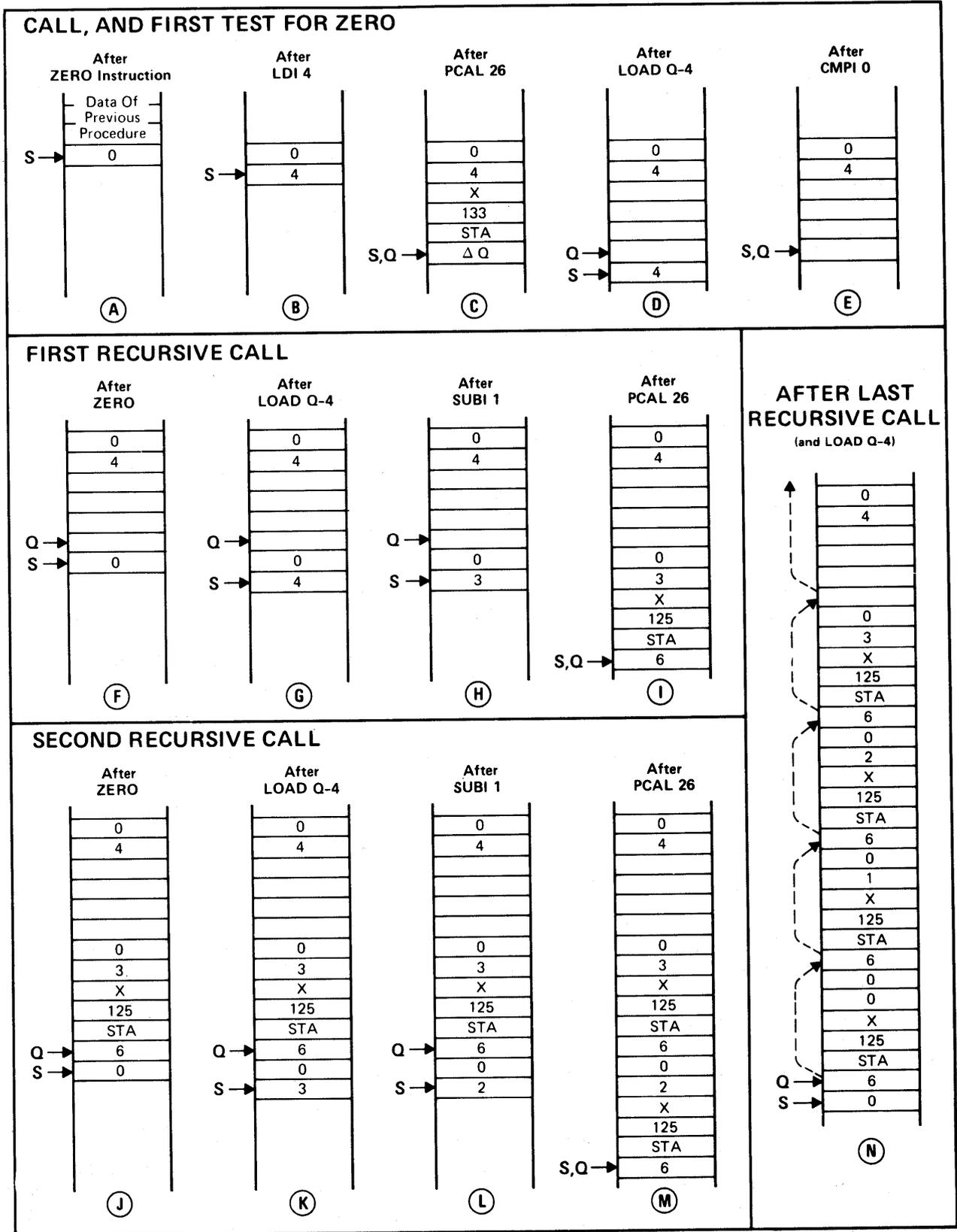
CALL, AND FIRST TEST FOR ZERO

After ZERO Instruction

```
┌─ Data Of ─┐
│─ Previous ─│
│  Procedure │
│     0      │  ← S
│            │
│            │
│            │
```
(A)

After LDI 4

```
│     0      │
│     4      │  ← S
│            │
│            │
```
(B)

After PCAL 26

```
│     0      │
│     4      │
│     X      │
│    133     │
│    STA     │
│    Δ Q     │  ← S,Q
```
(C)

After LOAD Q-4

```
│     0      │
│     4      │
│            │
│            │  ← Q
│     4      │  ← S
```
(D)

After CMPI 0

```
│     0      │
│     4      │
│            │
│            │  ← S,Q
│            │
```
(E)

FIRST RECURSIVE CALL

After ZERO

```
│     0      │
│     4      │
│            │
│            │  ← Q
│     0      │  ← S
```
(F)

After LOAD Q-4

```
│     0      │
│     4      │
│            │
│            │  ← Q
│     0      │
│     4      │  ← S
```
(G)

After SUBI 1

```
│     0      │
│     4      │
│            │
│            │  ← Q
│     0      │
│     3      │  ← S
```
(H)

After PCAL 26

```
│     0      │
│     4      │
│            │
│     0      │
│     3      │
│     X      │
│    125     │
│    STA     │
│     6      │  ← S,Q
```
(I)

AFTER LAST RECURSIVE CALL
(and LOAD Q-4)

```
│     0      │
│     4      │
│            │
│     0      │
│     3      │
│     X      │
│    125     │
│    STA     │
│     6      │
│     0      │
│     2      │
│     X      │
│    125     │
│    STA     │
│     6      │
│     0      │
│     1      │
│     X      │
│    125     │
│    STA     │
│     6      │
│     0      │
│     0      │
│     X      │
│    125     │
│    STA     │
│     6      │  ← Q
│     0      │  ← S
```
(N)

SECOND RECURSIVE CALL

After ZERO

```
│     0      │
│     4      │
│            │
│     0      │
│     3      │
│     X      │
│    125     │
│    STA     │
│     6      │  ← Q
│     0      │  ← S
```
(J)

After LOAD Q-4

```
│     0      │
│     4      │
│            │
│     0      │
│     3      │
│     X      │
│    125     │
│    STA     │
│     6      │  ← Q
│     0      │
│     3      │  ← S
```
(K)

After SUBI 1

```
│     0      │
│     4      │
│            │
│     0      │
│     3      │
│     X      │
│    125     │
│    STA     │
│     6      │  ← Q
│     0      │
│     2      │  ← S
```
(L)

After PCAL 26

```
│     0      │
│     4      │
│            │
│     0      │
│     3      │
│     X      │
│    125     │
│    STA     │
│     6      │
│     0      │
│     2      │
│     X      │
│    125     │
│    STA     │
│     6      │  ← S,Q
```
(M)

Figure 4-20. Stack Operations During Recursive Calls

followed each time by a stack marker. Thus, when the final LOAD Q-4 occurs in preparation for the zero test, the stack appears as shown at (N).

4-25. FIRST EXIT. The check at address 115 now finds that the parameter is zero. The checked copy of the parameter is deleted from the stack (P), figure 4-21 and the branch at address 116 transfers control to address 117 (rather than 121). As mentioned earlier in paragraph 4-20, an assumed value of 1 is necessary as an initial previous product in order to begin the multiplication loops. This is accomplished by a LDI 1 instruction (address 117), which puts a 1 on the top of the stack (Q). Then an unconditional branch at address 120 transfers control to address 126, where the "1" on the top of the stack is stored into the location reserved for this recursion's answer as shown at (R). The next instruction is the EXIT 1 instruction at address 127. This causes Q to move back six locations (Delta Q = 6) and S five locations (EXIT 1 deletes one of the two parameters), as shown at (S). The return address for the P Register is the MPYM Q-4 instruction at address 125. This causes the parameter at Q-4 (1) to be multiplied by the 1 on the top of the stack, leaving the answer as the new TOS element. Since 1x1 = 1 there is no apparent change from (S) to (T) but, in fact, a multiplication has occurred.

4-26. FIRST RECURSIVE EXIT. The answer of the first multiplication is now stored in the location reserved for it (Q-5) as shown at (U), figure 4-21 by the STOR Q-5 instruction at address 126. The next instruction, at 127, is again the EXIT 1 instruction which moves back the stack as shown at (V) and returns the P Register to the MPYM Q-4 instruction at address 125. The parameter for multiplication (at Q-4) is now 2 and, therefore, the multiplication result at (W) is 2. Again, this is stored back in the location reserved for it (Q-5) as shown at (X).

4-27. SUCCESSIVE EXITS. After saving the result, the next EXIT 1 is encountered again, causing the S and Q stack pointers to move back to the next marker, leaving the answer 2 on the TOS. The return for the P Register is again 125, the MPYM Q-4 instruction multiplies 2x3, and the following STOR Q-5 puts the answer 6 into the reserved location as shown at (Y), figure 4-21. Similarly, the last recursive exit causes the value 6 to be left on the TOS when the last return to address 125 is made. Then the final multiplication multiplies 6x4 and the last STOR Q-5 instruction puts the answer 24 into the location originally reserved for the end result FACTORIAL. The last EXIT instruction finds the return for the Q Register (Delta Q) pointing back to the origin of an earlier procedure and therefore, is not shown in the stack diagram at (Z). However, since one parameter is saved, the final answer remains on the TOS as shown. Meanwhile, the P Register returns to the next instruction in the outer block which is the STOR DB 15 instruction at address 133. This saves the answer in the global area and a final PCAL returns control to the system.
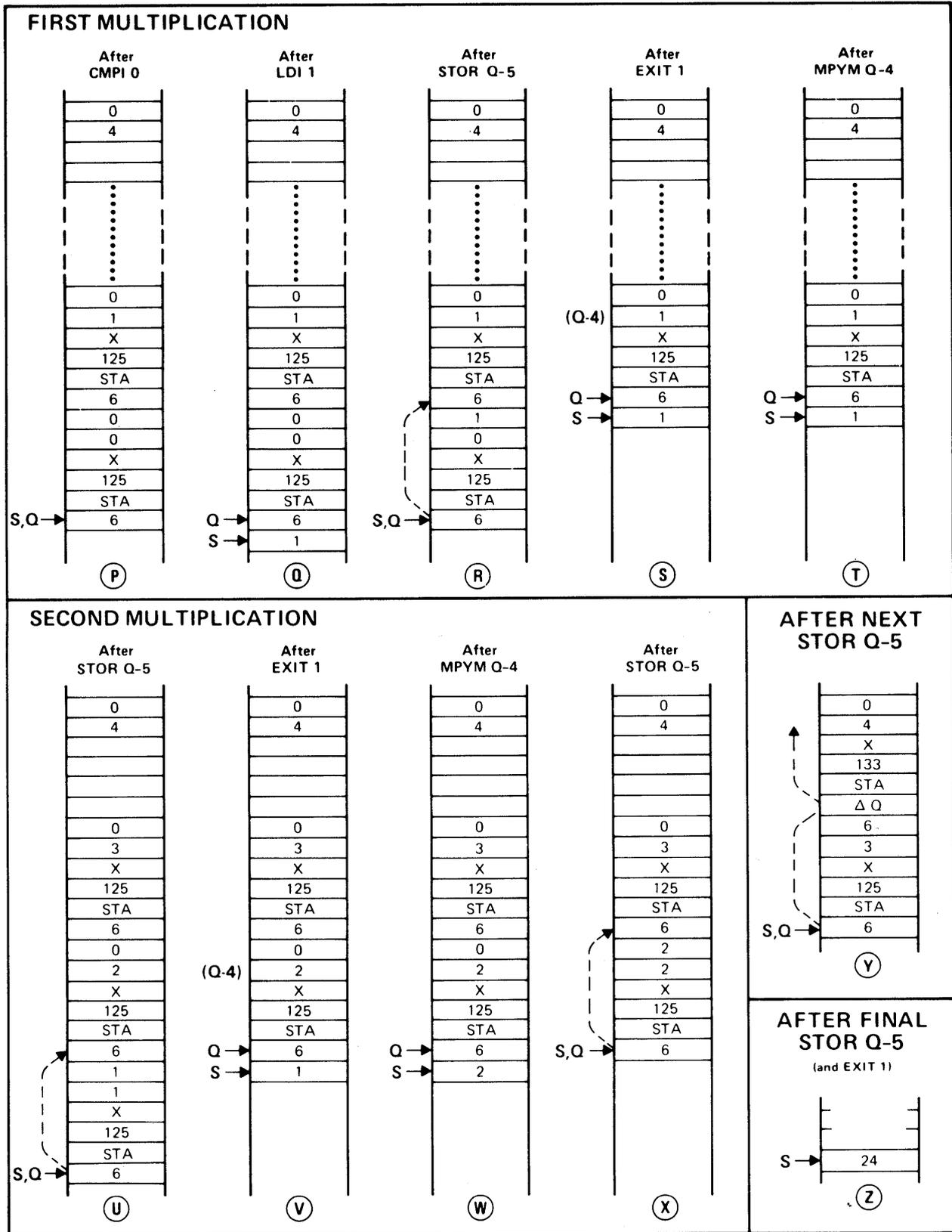
FIRST MULTIPLICATION

After CMPI 0	After LDI 1	After STOR Q-5	After EXIT 1	After MPYM Q-4
0	0	0	0	0
4	4	4	4	4
⋮	⋮	⋮	⋮	⋮
0	0	0	0	0
1	1	1	1	1
X	X	X	X	X
125	125	125	125	125
STA	STA	STA	STA	STA
6	6	6	6	6
0	0	1	1	1
0	0	0		
X	X	X		
125	125	125		
STA	STA	STA		
6	6	6		
	1			

(Q-4) at EXIT 1; Q → 6, S → 1 at EXIT 1 and MPYM Q-4

S,Q → 6 (P) · Q → 6, S → 1 (Q) · S,Q → 6 (R) · (S) · (T)

SECOND MULTIPLICATION

After STOR Q-5	After EXIT 1	After MPYM Q-4	After STOR Q-5
0	0	0	0
4	4	4	4
0	0	0	0
3	3	3	3
X	X	X	X
125	125	125	125
STA	STA	STA	STA
6	6	6	6
0	0	0	2
2	2	2	2
X	X	X	X
125	125	125	125
STA	STA	STA	STA
6	6	6	6
1	1	2	
1			
X			
125			
STA			
6			

(Q-4) at EXIT 1; Q → 6, S → 1 (V); Q → 6, S → 2 (W); S,Q → 6 (X)

S,Q → 6 (U)

AFTER NEXT STOR Q-5

0
4
X
133
STA
ΔQ
6
3
X
125
STA
6

S,Q → 6 (Y)

AFTER FINAL STOR Q-5
(and EXIT 1)

24

S → 24 (Z)

Figure 4-21. Stack Operations During Recursive Exits

NOTES

This section contains a description of the computer system's microcode and an introduction on how to read the system's Look Up Table (LUT) and Microprogram Listings. During the hardware training course, detailed instructions on how to read the LUT and Microprogram Listings will be presented as well as instructions on how to use the microcode in conjunction with the Maintenance Panel as a troubleshooting aid. (Complete LUT and Microprogram Listings are contained in the HP 3000 Series III Computer System Microprogram Listing Manual, part no. 30000-90136 and are not repeated in this manual.)

5-1. GENERAL INFORMATION

The microcode causes the CPU's hardware to execute the functions required to perform the operations specified by the machine instruction set. Therefore, in order to fully understand the microcode definitions contained in this section, it is first necessary to be completely familiar with the stack and pipeline operations discussed in Section II. Specifically, review paragraphs 2-16 through 2-28 and 2-71 through 2-128.

5-2. Stack Element Locations

The stack has a topmost element (TOS) which is logically quantity A. Similarly, the stack has logical quantities B, C, and D that correspond to the second (TOS-1), third (TOS-2), and fourth (TOS-3) words of the stack, respectively. The logical quantities A, B, C, and D can be either in the CPU TOS registers or in memory as determined by the Stack Register (SR Register). If the SR Register's contents equal zero, none of the logical quantities (A, B, C, or D) are in the TOS registers, but are located in some memory locations SM, SM-1, SM-2, and SM-3, respectively. At all times however, there are four TOS registers (RA, RB, RC, and RD Registers) that are named by the renamer logic as discussed in paragraph 2-96. In the microprogram field codes, RA, RB, RC, and RD refer to the hardware RA, RB, RC, and RD Registers and not to the logical quantities A, B, C, and D. There is a relationship however. For any of the logical quantities (A, B, C, and D), the state of the SR Register indicates where the quantity is as listed in table 5-1. For example; if the SR Register contents equal zero (table 5-1), then the logical quantity B is in memory location SM and, if the field code RB is used, the content of the register named RB is affected and not the logical quantity B. That is, for this case, the RB, RC, and RD Registers can be used as scratch pads without affecting the logical quantities B, C, and D. Manipulation of the TOS registers is discussed further in paragraph 5-3 through 5-6.

Table 5-1. Stack Element Locations

SR Register Contents	Locations			
	A	B	C	D
0	SM	SM-1	SM-2	SM-3
1	RA	SM	SM-1	SM-2
2	RA	RB	SM	SM-1
3	RA	RB	RC	SM
4	RA	RB	RC	RD

5-3. PUSH. The microinstruction Store Field code PUSH accomplishes three things:

a. It stores the output of the Shifter in the RD Register.

b. It increments the SR Register.

c. It renames the TOS registers so that the register named the RA Register becomes (:=) the RB Register; the register named the RB Register:= the RC Register; the register named the RC Register := the RD Register; and the register named the RD Register := the RA Register. Effectively, combining steps a and c, PUSH stores the contents of the U-Bus in the TOS location.

5-4. POP. The microinstruction Special Field code POP accomplishes two things.

a. It decrements the SR Register.

b. It renames the TOS registers so that the register named the RA Register:= the RD Register; the register named the RB Register := the RA Register; the register named the RD Register := the RB Register; and the register named the RD Register := the RC Register. Effectively, POP removes or "pops" the top element from the stack.

5-5. QUP. The microinstruction Store Field code QUP (Stack Marker Pointer Up) effectively stores the contents of the U-Bus in the stack at location SM+1. For example, if stack locations S and S-1 are in the TOS registers and locations S-2 is the first stack element in memory (SM),QUP places the contents of the U-Bus in a TOS register at stack location S-2 and SM := S-3. (To maintain stack integrity, the SR Register must be incremented with the Special Field code INSR to indicate the addition of the TOS register element.) When QUP is executed, TNAME (output of the Adder for the Mapper) := NAME (output of Namer Register) plus the contents of the SR Register to rename the TOS registers. (Refer to paragraph 2-96.) The register temporarily named RA := U-Bus as

follows:

```
If TNAME = 00 then TR0 := U-Bus
        = 01 then TR1 := U-Bus
        = 10 then TR2 := U-Bus
        = 11 then TR3 := U-Bus
```

TOS registers referenced in the R- and S-Bus fields (RA, RB, RC, and RD) of the following microinstruction will assume the temporary name.

5-6. QDWN. The microinstruction S-Bus field code QDWN (Stack Marker Pointer Down) effectively stores the contents of the lowest valid TOS register in the S-Bus Register. (To maintain stack integrity, the SR Register must be decremented with the Special Field code DCSR to indicate the deletion of the TOS register element.) When QDWN is executed, TNAME (output of the Adder for the Mapper) := NAME (output of Namer Register) plus the contents of the SR Register to rename the TOS registers. (Refer to paragraph 2-96.) The register temporarily named RD := S-Bus Register as follows:

```
If TNAME = 00 then TR3S := S-Bus Register
        = 01 then TR0S := S-Bus Register
        = 10 then TR1S := S-Bus Register
        = 11 then TR2S := S-Bus Register
```

The TOS registers are returned to their former names on the following clock cycle. TOS registers referenced in the Store Field (RA, RB, RC, and RD) of the previously executed microinstruction will assume the temporary name.

5-7. Reading Microprogram Listings

Briefly, to determine what hardware operations are required to execute a specific machine instruction, it is first necessary to locate the instruction's mnemonic in the LUT Listing's INSTR (instruction) column and then to read the instruction's associated microprogram starting address in the RAR column. (Additional information that can be obtained from the LUT Listing and how it can be used will be discussed during the training course.) After the microprogram's starting address is obtained, refer to that address in the Microprogram Listing's ADDRESS column and read the associated microinstructions in accordance with the following paragraphs.

5-8. MICROINSTRUCTION DESCRIPTIONS

As previously discussed in paragraphs 2-84 through 2-94, each microinstruction is a 32-bit word divided into eight fields. Each field, when coded with a particular micro-order, causes the hardware to perform a specific operation. Figure 5-1 lists the micro-orders that can be coded into each field and illustrates the bit number assignments for each field. The eight fields of a microinstruction are discussed in paragraphs 5-9 through 5-16.

	R-BUS	S-BUS	FCN	SHIFT	STORE	SPECIAL	SKIP	MCU	
00	PL	CIR	QASL	LRZ	PCLK	CCB	ZERO T	ABS	00
01	SR	SP1	QASR	LLZ	IOA	CCPX	NZRO T	CRL	01
02	Z	PADD	ROMX	SL1	IOD	CLSR	EVEN	CMD	02
03	MREG	RBR *	ROMN	SR1	MREG	SF3	ODD		03
04	PADD	CPX1	JSB	RRZ	BSP1 *	SIFG	NSME T		04
05	RBUS	MOD	CAND	RLZ	BSP0 *	SDFG	BIT6		05
06	X	CPX2	XOR	SWAB	SBR *	CTF	BIT8	WRA	06
07	XC	SWCH	AND	NOP	BUS *	CF3	NOFL	ROA	07
10	RD	QDWN	DVSB		PUSH	INSR	CRRY	PB	10
11	RC	IOA	UBNT		PL	DCSR	NCRY	NIR	11
12	RB	IOD	CADO T		Z	INCN	POS		12
13	RA	PCLK	SUBO T		QUP	INCT	NEG	RONP	13
14	SP1	CTRL	JMP		SP1	HBF	F1	RNP	14
15	SP0	CTRH	BNDT		SP0	FHB	NF1		15
16	UBUS	UBUS	CAD		CTRL	CLIB	F2		16
17	NOP	SBUS	SUB		CTRH	LBF	NF2	ROP	17
20		P	PNLR+		P	SF2	SRZ	DB	20
21		Q	PNLS+		Q	CF2	SRNZ	DATA	21
22		DB	ROMI		DB	CF1	SR4	DPOP	22
23		SM	ROM +		SM	SF1	SRN4	ROND	23
24		STA	REPC+		STA	SCRY	INDR	RND	24
25		SP3	REPN+		SP3	CCRY	SRL2		25
26		OPND	IOR		X	POPA T	NPRV	WRD	26
27		CC	CTSD+		RAR	POP	SRL3	ROD	27
30		RD	MPAD+		RD	SOV	RSB	S	30
31		RC	INCO+T		RC	CLO	JLUI	OPND	31
32		RB	CRS +		RB	CCZ T	TEST		32
33		RA	ADDO+T		RA	CCL	CTRM	RONS	33
34		DL	CTSS+		DL	CCG	F3	RNS	34
35		SP2	INC +		SP2	CCE	NEXT		35
36		PB	DCAD+		PB	CCA T	UNC	WRS	36
37		NOP	ADD +		NOP	NOP	NOP	ROS	37

* These options inhibit execution of the Special field options and enable the MCU field options in their place.

+ These functions cause an "ADD".

T Test is made on the T-Bus.

0 1	2 3 4	5 6 7 8 9	10	11 12 13	14	15 16 17 18 19	20 21 22	23 24 25	26 27 28	29 30 31
				REPN		COUNT		SPECIAL		
S-BUS		STORE		FUNCTION		SKIP	SHIFT	MCU		R-BUS
				JMP, JSB				JUMP TARGET		
				ANY ROM	0	SKIP 00-17		ROM CONSTANT		
					1					

Figure 5-1. Microinstruction Summary

5-9. R-Bus Field

The R-Bus field (bits 28 through 31) selects one of 16 register contents to be placed into the R-Bus Register. The R-Bus field code definitions are contained in table 5-2.

5-10. S-Bus Field

The S-Bus field (bits 0 through 4) selects one of 32 register contents to be placed into the S-Bus Register. The S-Bus field code definitions are contained in table 5-3.

5-11. Function Field

The Function field (bits 10 through 14) specifies the function to be performed by the ALU on the two operands in the R- and S-Bus Registers or a special function. The Function field code definitions are contained in table 5-4.

5-12. Shift Field

The Shift field (bits 20 through 22) specifies how the T-Bus data will be shifted. The Shift field code definitions are contained in table 5-5.

5-13. Store Field

The Store field (bits 5 through 9) selects one of the Store Logic registers or other destinations outside the CPU for the U-Bus data. The Store field code definitions are contained in table 5-6.

5-14. Special Field

The Special field (bits 23 through 27) codes perform the various operations listed in table 5-7.

5-15. MCU Option Field

The MCU Option field (bits 23 through 27) is executed in place of the Special field when the S-Bus field contains RBR or the Store field contains BUS, BSP0, BSP1, or SBR. The MCU Option field codes perform the various operations listed in table 5-8.

5-16. Skip Field

The Skip field (bits 15 through 19) determines which condition will be tested for a possible skip and specifies the conditions under which a JMP or JSB will be executed. The Skip field code definitions are contained in table 5-9.

5-17. MICRODIAGNOSTICS

The system's microcode also contains diagnostics to test the CPU registers, Main Memory, I/O channels, magnetic tape subsystem, and the Asynchronous Terminal Data PCA's. These diagnostics are executed from the system's Control Panel. Complete information for these diagnostics is contained in the HP 3000 Computer System Installation Manual, part no. 30000-90147.

Table 5-2. R-Bus Field Code Definitions

Label and Name	Field Code	Description
NOP (No Operation)	1111	No Operation.
MREG	0011	The MREG R-Bus field code is used to fetch a memory element that happens to lie in a TOS register (i.e., E is greater than SM). Prior to executing MREG, the value of S minus E must be placed in the SP1 Register. During execution of MREG, TNAME becomes the sum of NAME and SP1(14:15) and the R-Bus Register is loaded as follows: If TNAME = 00 then R-BUS := TR0R If TNAME = 01 then R-BUS := TR1R If TNAME = 10 then R-BUS := TR2R If TNAME = 11 then R-BUS := TR3R Due to the pipeline affect, a TOS register referenced in the Store field of the preceding microinstruction assumes the above TNAME.
PADD (Pre-Adder)	0100	The 16-bit content of the Pre-Adder is loaded into the R-Bus Register.
PL	0000	The 16-bit content of the PL Register is loaded into the R-Bus Register.
RA	1011	The RA R-Bus field code is used to read the content of the first TOS register (location S). SR must be greater than 0.* During execution, TNAME becomes NAME and the R-Bus Register is loaded as follows: If TNAME = 00 then R-BUS := TR0R If TNAME = 01 then R-BUS := TR1R If TNAME = 10 then R-BUS := TR2R If TNAME = 11 then R-BUS := TR3R

*True only if RA:RD are being used as part of the stack. RA:RD are used by the microprogram as scratch pad registers when not used otherwise.

Table 5-2. R-Bus Field Code Definitions (Continued)

Label and Name	Field Code	Description
RB	1010	The RB R-Bus field code is used to read the second TOS register (location S-1). SR must be greater than 1.* During execution, TNAME becomes NAME and the R-Bus Register is loaded as follows: If TNAME = 00 then R-BUS := TR1R If TNAME = 01 then R-BUS := TR2R If TNAME = 10 then R-BUS := TR3R If TNAME = 11 then R-BUS := TR0R
R-BUS	0101	The RBUS R-Bus field code causes R-Bus Register to remain unchanged.
RC	1001	The RC R-Bus field code is used to read the third TOS register (location S-2). SR must be greater than 2.* During execution, TNAME becomes NAME and the R-Bus Register is loaded as follows: If TNAME = 00 then R-BUS := TR2R If TNAME = 01 then R-BUS := TR3R If TNAME = 10 then R-BUS := TR0R If TNAME = 11 then R-BUS := TR1R
RD	1000	The RD R-Bus field code is used to read the fourth TOS register (location S-3). SR must be equal to 4.* During execution, TNAME becomes NAME and the R-Bus Register is loaded as follows: If TNAME = 00 then R-BUS := TR3R If TNAME = 01 then R-BUS := TR0R If TNAME = 10 then R-BUS := TR1R If TNAME = 11 then T-BUS := TR2R

*True only if RA:RD are being used as part of the stack. RA:RD often are used by the microprogram as scratch pad registers when not used otherwise.

Table 5-2. R-Bus Field Code Definitions (Continued)

Label and Name	Field Code	Description
SP0 (Scratch Pad 0)	1101	The 16-bit content of the SP0 Register is loaded into the R-Bus Register.
SP1 (Scratch Pad 1)	1100	The 16-bit content of the SP1 Register is loaded into the R-Bus Register.
SR (Stack Register)	0001	The 3-bit content of the SR Register is loaded into R-Bus Register bits 0 thru 2. R-Bus Register bits 3 thru 15 become zeros.
UBUS	1110	The 16-bit U-Bus data word is loaded into the R-Bus Register. The U-Bus data is established by the preceding microinstruction.
X (Index)	0110	The 16-bit content of the X Register is loaded into the R-Bus Register.
XC (X Conditional)	0111	The XC R-Bus field code is used with indexed memory addressing. If the index bit of the current instruction (CIR bit 4) is zero, the R-Bus Register is loaded with zeros. Otherwise, the R-Bus Register is loaded with the 16-bit content of the X Register.
Z (Stack Limit)	0010	The 16-bit content of the Z Register is loaded into the R-Bus Register.

Table 5-3. S-Bus Field Code Definitions

Label and Name	Field Code	Description
CC (Condition Code)	10111	The CC S-Bus field code is used to retrieve the condition code (CC) portion of the status word for use with certain conditional branch instructions. When executed, bits 6 and 7 of the status word are loaded into bits 8 and 9 of the S-Bus Register and, if both of these bits are zeros, S-Bus Register bit 7 becomes a one. All other S-Bus Register bits become zeros.
CIR (Current Instruction Register)	00000	The 16-bit output of the CIR is loaded into the S-Bus Register.
CPX1	00100	CPX1, a collection of 16 special signals, is loaded into the S-Bus Register. Refer to para 2-127.
CPX2	00110	CPX2, a collection of 16 special signals, is loaded into the S-Bus Register. Refer to para 2-128.
CTRH (Counter High)	01101	The 6-bit content of the CNTR Register is loaded into bits 4 thru 9 of the S-Bus Register. All other S-Bus Register bits become zeros.
CTRL (Counter Low)	01100	The 6-bit content of the CNTR Register is loaded into bits 10 thru 15 of the S-Bus Register. All other S-Bus Register bits become zeros.
DB (Data Base)	10101	The 16-bit content of the DB Register is loaded into the S-Bus Register.
DL (Data Limit)	11100	The 16-bit content of the DL Register is loaded into the S-Bus Register.

Table 5-3. S-Bus Field Code Definitions (Continued)

Label and Name	Field Code	Description
IOA (I/O Address)	01001	The 8-bit content of the IDN Register is loaded into bits 8 thru 15 of the S-Bus Register. Bits 0 thru 7 of the S-Bus Register become zeros.
IOD (I/O Data)	01010	The 16-bit content of the Direct Input Data (DID/MUXMA) Register in the IOP is loaded into the S-Bus Register.
MOD (Module Number)	00101	The MOD S-Bus field code provides the CPU with two pieces of information. When executed, the 4-bit content of the IMN Register is loaded into bits 4 thru 7 of the S-Bus Register. Also, bit 13 of the S-Bus Register becomes a 1, indicating MOD 1. These bit are used to fetch the correct Q1 and Z1 entries in the CST. All other bits of the S-Bus Register become zeros.
NOP (No Operation)	11111	No operation.
OPND (Operand)	10110	The 16-bit content of the OPND Register is loaded into the S-Bus Register. An attempt to execute an OPND while an MCU operand directed operation is in progress results in a CPU freeze until the MCU operation is complete.
P (Program Counter)	10000	The 16-bit content of the P Register is loaded into the S-Bus Register.
PADD (Program Base)	00010	The 16-bit output of the Pre-Adder is loaded into the S-Bus Register.
PB (Program Base)	11110	The 16-bit content of the PB Register is loaded into the S-Bus Register.

Table 5-3. S-Bus Field Code Definitions (Continued)

Label and Name	Field Code	Description
PCLK (Process Clock)	01011	The 16-bit content of the PCLK Reg-register is placed in the S-Bus Register.
Q (Stack Marker Pointer)	10001	The 16-bit content of the Q Regis- is loaded into the S-Bus Register.
QDWN (Stack Marker Pointer Down)	01000	The QDWN S-Bus field code is used to put the content of the lowest valid TOS register in the S-Bus Register. Refer to para 5-6.
RA	11011	The register named RA by the Namer Register is placed in the S-Bus Register.* Refer to para 2-96 and 5-2.
RB	11010	The register named RB by the Namer Register is placed in the S-Bus Register.* Refer to para 2-96 and 5-2.
RBR (Read Bank Register)	00011	Read Bank Register onto S-Bus (12: 15). The S-Bus bits 0 - 11 are zeroed. The Bank Register to be read is specified in the MCU field. Execution of the Special field is inhibited.
RC	11001	The register named RC by the Namer Register is placed in the S-Bus Register.* Refer to para 2-96 and 5-2.
RD	11000	The register named RD by the Namer Register is placed in the S-Bus Register.* Refer to para 2-96 and 5-2.
SBUS	01111	The SBUS code causes the S-Bus Register content to remain unchanged.

*True only if RA:RD are being used as part of the Stack. RA:RD are often used by the microprogram as scratch pad registers when not used otherwise.

Table 5-3. S-Bus Field Code Definitions (Continued)

Label and Name	Field Code	Description
SM (Stack Memory)	10011	The 16-bit content of the SM Register is loaded into the S-Bus Register.
SP1 (Scratch Pad 1)	00001	The 16-bit content of the SP1 Register is loaded into the S-Bus Register.
SP3 (Scratch Pad 3)	10101	The 16-bit content of the SP3 Register is loaded into the S-Bus Register.
STA (STATUS)	10100	The 16-bit status word is loaded into the S-Bus Register.
SWCH	00111	The 16-bit content of the Switch Register is loaded into the S-Bus Register.
UBUS	01110	The 16-bit U-Bus data word is loaded into the S-Bus Register. The U-Bus data is established by the preceding microinstruction.

Table 5-4. Function Field Code Definitions

Label and Name	Field Code	Description
ADD	11111	The content of the R-Bus Register is added to the content of the S-Bus Register and the result is placed on the T-Bus.
ADDO (Add-Enable Over-flow)	11011	The content of the R-Bus Register is added to the content of the S-Bus Register and the result is placed on the T-Bus.
AND	00111	The content of the R-Bus Register is logically "anded" with the content of the S-Bus Register and the result is placed on the T-Bus.
BNDT (Bounds Test)	01101	The Function field code BNDT is used to perform a bounds test of an address. Execution of this code results in the content of the R-Bus Register minus the content of the S-Bus Register being placed on the T-Bus. If RRZ, RLZ, LRZ, or LLZ is specified, then BNDT does a "CAD" instead of a "SUB". The R- and S-Bus fields are coded so that this result is a negative number (CARRY=0) if a bounds violation occurs. If the CPU is not operating in the priviledged mode (STATUS(0) = 0), and a bounds violation occurs, a microjump to ROM address 0003 is executed. If no violation has occurred (CARRY = 1) or the CPU is operating in the priviledged mode (STATUS (0) = 1), the next microinstruction will be executed in the usual manner.
CAD (Complement and Add)	01110	The content of the R-Bus Register is added to the one's complement of the content of the S-Bus Register and the result is placed on the T-Bus. If the S-Bus Register contains all zeros, CAD results in the

Table 5-4. Function Field Code Definitions (Continued)

Label and Name	Field Code	Description
CAD (Cont)		R-Bus Register contents minus 1 on the T-Bus.
CADO (Complement and Add-Enable Over-flow)	01010	The content of the R-Bus Register is added to the one´s complement of the content of the S-Bus Register and the result is placed on the T-Bus. Carry and overflow are modified in the Status Register and the condition code is set to CCA on the T-Bus data.
CAND (Complement-And)	00101	The R-Bus Register content is logically "anded" with the complement of the S-Bus Register content and the result is placed on the T-Bus.
CRS (Circular Shift)	11010	The R-Bus Register content is added to the S-Bus Register content and the result is placed on the T-Bus. The T-Bus is then circular shifted one place right or left as specified in the Shift field (SR1 orSL1) and placed on the U-Bus.
CTSD (Controlled Shift Double)	10111	The Function field code CTSD adds the contents of the R-Bus Register and the S-Bus Register, puts the result on the T-Bus, and performs a double word shift of the T-Bus and a scratch pad left or right as specified by the Shift field code (SR1 or SL1). The type of shift is determined by the content of the CIR as follows: If CIR(7) = 1 then circular shift If CIR(7:8) = 01 then logical shift If CIR(7:8) = 00 then arithmetic shift The most significant word is on the T-Bus. If a left shift is specified, SP1 Register contains the least significant word. If a right shift is specified, SP3 Register contains the least significant

Table 5-4. Function Field Code Definitions (Continued)

Label and Name	Field Code	Description
CTSD (Cont)		word. Regardless of the direction of the shift, both the SP1 and SP3 Registers are shifted left and right respectively.
CTSS (Controlled T-Bus Shift Single)	11100	The R-Bus Register content is added to the S-Bus Register content and the result is placed on the T-Bus. The T-Bus is then shifted left or right as specified by the Shift field code (SR1 or SL1). The type of shift is determined by the content of the CIR as follows: If CIR(7) = 1 then circular shift If CIR(7:8) = 01 then logical shift If CIR(7:8) = 00 then arithmetic shift
DCAD (Decimal Add)	11110	The contents of the R- and S-Bus Registers are added and the result is placed into the Decimal Correct-or Adder. The Decimal Corrector Adder output is placed on the U-Bus.
DVSB (Divide-Subtract)	01000	The Function field code DVSB performs the subtract, shift, and test necessary to execute a divide algorithm. The R- and S-Bus fields of the microinstruction are coded so that initially the 16-bit divisor is in the S-Bus Register and the most significant 16-bits of the dividend are in the R-Bus Register. The least significant 16-bits of dividend are in the SP1 Register. Both divisor and dividend must be positive numbers upon execution of the DVSB code and Flag 2 (F2) must be 0 (cleared). An SL1 code in the Shift field of the microinstruction directs the left shift on the T-Bus. The following algorithm is then executed repeatedly to perform the complete divide.

Table 5-4. Function Field Code Definitions (Continued)

Label and Name	Field Code	Description

DVSB (Cont)		TBUS := RBUS - SBUS
		 UBUS(0:14) := TBUS(1:15)
		 If ALU carry or F2=1 then:
		
		     BEGIN
		
		       RREG(0:14) := UBUS(0:14)
		       RREG(15) := SP1(0)
		       SP1(0:14) := SP1(1:15)
		       SP1(15) := 1
		       F2 := TBUS(0)
		
		     END
		
		   else:
		
		     BEGIN
		
		       RREG(0:14) := RREG(1:15)
		       RREG(15) := SP1(0)
		       SP1(0:14) := SP1(1:15)
		       SP1(15) := 0
		       F2 := RREG(0)
		
		     END
		
		 For example, after 17 executions of the above algorithm, a 16-bit quotient is contained in the SP1-Register and the remainder times 2 is contained in the R-Bus Register. When the remainder is unloaded from the R-Bus Register, it is shifted right one place (divided by 2).

INC (Incremented Add)	11101	The R-Bus Register content is added to the S-Bus Register content plus 1. The result is placed on the T-Bus.
INCO (Incremented Add-Enable Overflow)	11001	The R-Bus Register content is added to the S-Bus Register content plus 1, the result placed on the T-Bus, and the carry and overflow is modified in the Status Register. The condition code is set to CCA on the T-Bus Data.

Table 5-4. Function Field Code Definitions (Continued)

Label and Name	Field Code	Description
IOR (Inclusive OR)	10110	The content of the R-Bus Register is logically inclusively "ored" with the content of the S-Bus Register and the result is placed on the T-Bus.
JMP (Jump)	01100	The JMP Function field code directs a micro-jump to the ROM address (jump target) specified by bits 20 thru 31 of the ROM Output Register if the Skip field condition is met (a condition must be specified). The R-Bus, Shift, and Special Field Decoders are disabled and the U-Bus and T-Bus become the S-Bus Register content.
JSB (Jump to Subroutine)	00100	The JSB Function field code directs a micro-subroutine jump to the ROM address specified by bits 20 thru 31 of the ROM Output Register if the Skip field code condition is met. If the condition is met and the JSB is executed, the Save Register is loaded with the address of the line following the JSB and is used as a return address at the subroutine end (see Function field code RSB). During execution of the JSB, the R-Bus, Shift, and Special Field Decoders are disabled and the T-Bus and U-Bus become the S-Bus Register contents.
MPAD (Multiply-Add)	11000	The Function field code MPAD performs the add, shift, and test necessary to execute a multiply algorithm. The R-Bus field of the microinstruction is coded so that initially the 16-bit multiplicand is in the R-Bus Register. The S-Bus field code is UBUS which is initially all zeros. The SP3-Register contains the 16-bit multiplier. Both multiplier and multiplicand

Table 5-4. Function Field Code Definitions (Continued)

Label and Name	Field Code	Description

MPAD (Cont)		must be positive numbers upon execution of the MPAD code. An SR1 code in the Shift field directs the right shift of the T-Bus. The following algorithm is executed repeatedly to perform a complete multiply.
		
		  T-BUS := R-REG plus S-REG
		  U-BUS(1:15) := T-BUS(0:14)
		  U-BUS(0) := ALU carry
		  If SP3(15) = 1 then:
		
		    BEGIN
		
		      S-REG := U-BUS
		      SP3(1:15) := SP3(0:14)
		      SP3(0) := T-BUS(15)
		
		    END
		
		  else:
		
		    BEGIN
		
		      S-REG(1:15) := S-REG(0:14)
		      SP3(1:15) := SP3(0:14)
		      SP3(0) := S-REG(15)
		
		    END
		
		After 16-executions of the above algorithm, the result is a 32-bit word with the most significant 16 bits in the S-Bus Register and the least significant 16 bits in the SP3 Register.

PNLR (Panel Read)	10000	The PNLR Function field code allows the auxiliary control panel to select and display a CPU register. This code appears in the microprogram during execution of HALT and PAUSE routines. When PNLR is executed, the ROM Output Register (ROR1) and R- and S-Bus fields are disabled. The maintenance panel

Table 5-4. Function Field Code Definitions (Continued)

Label and Name	Field Code	Description
PNLR (Cont)		interface supplies these field codes which put the content of the selected register in the associated register (R- or S-Bus). The T-Bus and U-Bus become the R-Bus Register content plus the S-Bus Register content (one of which will be zeros). The auxiliary control panel completes this operation by displaying the U-Bus as the selected register.
PNLS (Panel Store)	10001	The PNLS Function field code allows the auxiliary control panel to load a CPU register with the content of one of its switch registers. This code is part of the halt mode interrupt micro-routine for servicing a maintenance panel interrupt. When PNLS is executed, the ROM Output Register (ROR2) Store field is disabled and the maintenance panel interface card supplies the Store field code respective of the selected CPU register. A SWCH S-Bus field code causes the S-Bus Register to be loaded with the content of the selected auxiliary control panel switch register. The T-Bus and U-Bus become the R-Bus Register content (zeros) plus the S-Bus Register content and, at the end of the cycle, the selected register is loaded with the U-Bus data.
QASL	00000	The QASL Function field code causes a four register arithmetic shift left of the U-Bus, SP3 and SP1 Registers, and the R-Bus Register containing the most, next most, next least, and least significant words respectively. Shift Left One code (SL1) is required in the Shift field. The sign bit is preserved.

Table 5-4. Function Field Code Definitions (Continued)

Label and Name	Field Code	Description
QASL (Cont)		T-BUS := S-REG U-BUS(0) := T-BUS(0) U-BUS(1:14) := T-BUS(2:15) U-BUS(15) := SP3(0) SP3(0:14) := SP3(1:15) SP3(15) := SP1(0) SP1(0:14) := SP1(1:15) SP1(15) := R-REG(0) R-REG(0:14) := R-REG(1:15) R-REG(15) := 0
QASR	00001	The QASR function field code causes a four register arithmetic shift right of the U-Bus, SP3 and SP1 Registers, and the S-Bus Register containing the most, next most, next least, and least significant words respectively. Shift Right One code (SR1) is required in the Shift field. The sign bit is propagated. T-BUS := R-REG U-BUS(0:1) := T-BUS(0) U-BUS(2:15) := T-BUS(1:14) SP3(0) := T-BUS(15) SP3(1:15) := SP3(0:14) SP1(0) := SP3(15) SP1(1:15) := SP1(0:14) S-REG(0) := SP1(15) S-REG(1:15) := S-REG(0:14)
REPC (Repeat Until Condition)	10100	The REPC Function field code causes the next microinstruction to be executed repeatedly until the Skip field condition of that microinstruction is met. During execution, the T-Bus becomes the R-Bus Register content plus the S-Bus Register content. The REPC code is decoded from ROR2 and, at that time disables the RAR increment function and ROR1 load function and sets the Repeat FF. The RAR then contains the address of the microinstruction following the one to be repeated and ROR1 contains the microinstruction to be repeated. The next cycle

Table 5-4. Function Field Code Definitions (Continued)

Label and Name	Field Code	Description
REPC (Cont)		loads ROR2 and executes the micro-instruction to be repeated. As long as the Repeat FF remains set, the content of ROR1 and ROR2 does not change and is executed each cycle. When the Skip field condition is met, the Repeat FF is cleared, the pipeline is filled correctly, and the next microinstruction is fetched in the usual manner.
REPN (Repeat N Times)	10101	The REPN Function field code operates in the same manner as the REPC code previously described. The difference is that REPN loads a Repeat Counter Register with the contents of the microinstruction Skip field. Bits 1 thru 5 of the counter become ROR2 bits 5 thru 9; bit 0 of the counter becomes a 1. The counter content is then the two's complement of the number of repeats to be performed. To utilize the counter, the repeated microinstruction contains a Special field code INCTR (Increment Counter) and a Skip field condition CTRM (Counter Maximum).
ROM	10011	The Function field code ROM loads the R-Bus Register with a 16-bit constant obtained from the microinstruction. The ROM code is decoded from ROR1, loading the the R-Bus Register with bits 16 thru 31 of ROR1. The T-Bus then becomes the R-Bus Register content plus the S-Bus Register content. The R-Bus, Shift, Special, and Skip Field Decoders are disabled by the ROM code.
ROMI (ROM Inclusive)	10010	The Function field code ROMI loads the R-Bus Register with a 16-bit constant obtained from the microinstruction. The ROMI code is decoded from ROR1, loading the R-Bus

Table 5-4. Function Field Code Definitions (Continued)

Label and Name	Field Code	Description
ROMI (Cont)		Register with ROR1 bits 16 thru 31. The T-Bus then becomes the R-Bus Register content inclusive "ored" with the S-Bus Register content. The R-Bus, Shift, Special, and Skip Field Decoders are disabled by the ROMI code.
ROMN (ROM And)	00011	The Function field code ROMN loads the R-Bus Register with a 16-bit constant obtained from the microinstruction. The ROMN code is decoded from ROR1, loading the R-Bus Register with ROR1 bits 16 thru 31. The T-Bus becomes the R-Bus Register content logically "anded" with the S-Bus Register content. The R-Bus, Shift, Special, and Skip Field Decoders are disabled by the ROMN code.
ROMX (ROM Exclusive)	00010	The Function field code ROMX loads the R-Bus Register with a 16-bit constant obtained from the microinstruction. The ROMX code is decoded from ROR1, loading the R-Bus Register with ROR1 bits 16 thru 31. The T-Bus becomes the R-Bus Register content exclusive "ored" with the S-Bus Register content. The R-Bus, Special, Shift, and Skip Field Decoders are disabled by the ROMX code.
SUB (Subtract)	01111	The content of the S-Bus Register is subtracted from the content of the R-Bus Register and the result is placed on the T-Bus.
SUBO (Subtract-Enable Overflow)	01011	The content of the S-Bus Register is subtracted from the content of the R-Bus Register and the result is placed on the T-Bus. Carry and overflow are modified in the Status Register and condition code CCA is set on the T-Bus data.

Table 5-4. Function Field Code Definitions (Continued)

Label and Name	Field Code	Description
UBNT (Unconditional Bounds Test)	01001	The Function field code UBNT is used to perform an unconditional bounds test of an address. Execution of this code results in the content of the R-Bus Register minus the content of the S-Bus Register being placed on the T-Bus. If RRZ, RLZ, LRZ, LLZ is specified, then UBNT does a "CAD" instead of a "SUB". The R- and S-Bus fields are coded so that this result is a negative number (CARRY = 0) if a bounds violation occurs. The response to a bounds violation is a micro-jump to ROM address 0003. If no violation occurs (CARRY = 1), the next micro-instruction is executed in the us- manner.
XOR (Exclusive OR)	00110	The content of the R-Bus Register is exclusive "ored" with che content of the S-Bus Register and the result is placed on the T-Bus.

Table 5-5. Shift Field Code Definitions

Label and Name	Field Code	Description
(blank)	111	No shift, the T-Bus word is placed directly on the U-Bus.
LLZ (Left to Left and Zero)	001	The Shift field code LLZ places the left byte of the T-Bus in the left byte of the U-Bus and places zeros in the right byte of the U-Bus.
LRZ (Left to Right and Zero)	000	The Shift field code LRZ places the left byte of the T-Bus in the right byte of the U-Bus and places zeros in the left byte of the U-Bus.
RLZ (Right to Left and Zero)	101	The Shift field code RLZ places the right byte of the T-Bus in the left byte of the U-Bus and places zeros in the right byte of the U-Bus.
SWAB (Swap Bytes)	110	The Shift field code SWAB places the left byte of the T-Bus in the right byte of the U-Bus and the right byte of the T-Bus in the left byte of the U-Bus.
RRZ (Right to Right and Zero)	100	The Shift field code RRZ places the right byte of the T-Bus in the right byte of the U-Bus and places zeros in the left byte of the U-Bus
SL1 (Shift Left 1)	010	The Shift field code SL1 shifts the T-Bus one place left onto the U-Bus. Refer to the Function field code descriptions for the action taken when used with Function field codes CRS, CTSD, CTSS, DVSB, and TASL.
SR1 (Shift Right 1)	011	The Shift field code SR1 shifts the T-Bus logically one place right onto the U-Bus. Refer to the Function field code descriptions for the action taken when used with Function field codes CRS, CTSD, CTSS, MPAD, and TASR.

Table 5-6. Store Field Code Definitions

Label and Name	Field Code	Description
NOP (No Operation)	11111	No Operation.
BSP0 (Bus to Scratch Pad 0)	00101	The Store field code BSP0 stores the U-Bus into ACOR or DCOR, depending on the MCU field option selected and into SP0. Disables the Special field and enables the MCU options, one of which must be used.
BSP1 (Bus to Scratch Pad 1)	00100	Same as BSP0 except SP1 is used.
BUS	00111	Same as BSP0 except none of the scratch pad registers are used.
CTRH (Counter High)	01111	The Store field code CTRH stores U-Bus bits 4 thru 9 in CNTR Register bits 0 thru 5.
CTRL (Counter Low)	01110	The Store field code CTRL stores U-Bus bits 10 thru 15 in CNTR Register bits 0 thru 5.
DB (Data Base)	10011	The Store field code DB stores the 16-bit U-Bus word in the DB Register.
DL (Data Limit)	11100	The Store field code DL stores the 16-bit U-Bus word in the DL Register.
IOA (I/O Address)	00001	The Store field code IOA sends the command on the U-Bus (bits 5 thru 7) to the device whose address is on the U-Bus in bits 8 thru 15. U-Bus bit 0 = 1 sends a service-out signal to the device.
IOD (I/O Data)	00010	The Store field code IOD stores the 16-bit word currently on the U-Bus in the Direct Output Data (DOD) Register.

Table 5-6. Store Field Code Definitions (Continued)

Label and Name	Field Code	Description
MREG (Memory Register)	00011	The Store field code MREG is used to store data in an address that lies in a TOS register (i.e., S > E > SM where S = SR + SM). Prior to executing MREG, the value E minus S is placed in the SP1 Register. During execution, TNAME becomes the sum of NAME and SP1(14:15) and the TOS registers are loaded as follows: If TNAME = 00 then TR0 := U-BUS If TNAME = 01 then TR1 := U-BUS If TNAME = 10 then TR2 := U-BUS If TNAME = 11 then TR3 := U-BUS Due to the pipeline effect, a TOS register referenced in the R- or S-Bus field of the following microinstruction assumes the above described TNAME.
P (Program Count)	10000	The Store field code P stores the 16-bit U-Bus word in the P Register.
PB (Program Base)	11110	The Store field code PB stores the 16-bit U-Bus word in the PB Register.
PCLK (Process Clock)	00000	The Process Clock, PCLK, is placed in the S-Bus Register.
PL (Program Limit)	01001	The Store field code PL stores the 16-bit U-Bus word in the PL Register.
PUSH	01000	The Store field code PUSH effectively moves all stack elements down one location and loads the U-Bus word on the TOS. Refer to paragraph 5-3.
Q (Stack Marker Pointer)	10001	The Store field code Q stores the 16-bit U-Bus word in the Q Register.

Table 5-6. Store Field Code Definitions (Continued)

Label and Name	Field Code	Description
QUP (Stack Marker Pointer UP)	10001	The Store field code QUP effectively inserts the U-Bus word into the stack at location SM plus one. Refer to para 5-5.
RA	11011	The Store field code RA stores the U-Bus word in the register named RA by the Namer Register.* Refer to para 2-96 and 5-2.
RAR (ROM Address Register)	10111	The Store field code RAR stores bits 0 thru 15 of the U-Bus in ROM Address Register bits 0 thru 15. The intent of this code is to force the processor to a new microprogram address specified by the U-Bus word. Execution of the RAR code requires three microcycles. The first cycle loads the ROM Address Register and the next two cycles are NOPs allowing the ROM Output Registers (ROR1 and ROR2) to be loaded with the new microinstruction.
RB	11010	The Store field code RB stores the U-Bus word in the register named RB by the Namer Register.* Refer to para 2-96 and 5-2.
RC	11001	The Store field code RC stores the U-Bus word in the register named RC by the Namer Register.* Refer to para 2-96 and 5-2.
RD	11000	The Store field code RD stores the U-Bus word in the register named RD by the Namer Register.* Refer to para 2-96 and 5-2.
SBR (Stack Bank Register)	00110	The Store field code SBR stores U-Bus bits 12:15 in the Bank Register specified in the MCU field code.

*True only if RA:RD are being used as part of the stack, RA:RD often are used by the microprogram as scratch pad registers when not used otherwise.

Table 5-6. Store Field Code Definitions (Continued)

Label and Name	Field Code	Description
SBR (Cont)		Execution of the Special field is inhibited.
SM (Stack Memory Pointer)	10010	The Store field code SM stores the U-Bus word in the SM Register.
SP0 (Scratch Pad 0)	01101	The Store field code SP0 stores the U-Bus word in the SP0 Register.
SP1 (Scratch Pad 1)	01100	The Store field code SP1 stores the U-Bus word in the SP1 Register.
SP2 (Scratch Pad 2)	11101	The Store field code SP2 stores the U-Bus word in the SP2 Register.
SP3 (Scratch Pad 3)	10101	The Store field code SP3 stores the U-Bus word in the SP3 Register.
STA (Status)	10100	The Store field code STA stores the U-Bus word in the Status Register.
X (Index)	10110	The Store field code X stores the U-Bus word in the X Register.
Z (Stack Limit Pointer)	01010	The Store field code Z stores the U-Bus word in the Z Register.

Table 5-7. Special Field Code Definitions

Label and Name	Field Code	Description
(blank)	11111	No Special field operation.
CCA (Condition Code A)	11110	The Special field code CCA sets the condition code bits to CCL (01) if the T-Bus word is less than zero (T(0) = 1), CCE (10) if the T-Bus word is equal to zero (Signal T = 0 is true), or CCG (00) if the T-Bus word is greater than zero (T(0) = 0 and signal T = 0 is false).
CCB (Condition Code B)	00000	The Special field code CCB sets the condition code to CCL (01) if bits 8:15 of the U-Bus form a special ASCII character, CCE (10) if an alphabetic ASCII character, or CCG (00) if a numeric ASCII character.
CCE (Condition Code E)	11101	The Special field code CCE sets the condition code bits to CCE (10).
CCG (Condition Code G)	11100	The Special field code CCG sets the condition code bits to CCG (00).
CCL (Condition Code L)	11011	The Special field code CCL sets the condition code bits to CCL (01).
CCPX (Clear CPX1)	00001	Clears the Interrupt Status Register bits as specified by the true bits on the U-Bus. U-Bus Bit 0 Halt 1 Run 2 System Halt 3 (Unused) Bits 4 (MSB) through 7 (LSB) code the following functions: Octal Code 0 NOP 1 Clear BNDV 2 Clear Illegal Address

Table 5-7. Special Field Code Definitions (Continued)

Label and Name	Field Code	Description
CCPX (Cont)		3 Clear CPU Timer 4 Clear System Parity Error 5 Clear Address Parity Error 6 Clear Data Parity Error 7 Clear Module Interrupt 10 Clear External Interrupt 11 Power Fail Turn-Off Interrupt 16 Reverse System Parity 17 Reverse MCUD Parity 8 Diagnostic NIRTOCIR 9 (Unused) 10 Diagnostic Set CPX1 (Bits 1:18) 11 Clear ICS Flag 12 Clear DISP Flag 13 (Unused) 14 Diagnostic Freeze 15 Clear Panel FF's
CCRY (Clear Carry)	10101	The Special field code CCRY clears carry in the Status Register.
CCZ (Condition Code Zero)	11010	The Special field code CCZ sets the condition code bits to CCE (10) if the T-Bus word is equal to zero (signal T = 0 true) or CCG (00) if the T-Bus word is not equal to zero (signal T = 0 false).

Table 5-7. Special Field Code Definitions (Continued)

Label and Name	Field Code	Description
CF1 (Clear Flag 1)	10010	The Special field code CF1 clears CPU Flag 1 FF.
CF2 (Clear Flag 2)	10001	The Special field code CF2 clears CPU Flag 2 FF.
CF3 (Clear Flag 3)	00111	The Special field code CF3 clears CPU Flag 3 FF.
CLIB (Clear Indirect Bit)	01110	At the end of the cycle, CLIB sets the Indirect Bit FF which masks the indirect line until a NEXT or JLUI option in the Skip field is encountered.
CLO (Clear Overflow)	11001	The Special field code CLO clears the status word overflow bit.
CLSR (Clear SR)	00010	The Special field code CLSR clears the SR Register. This is an asynchronous reset. No other SR operation is allowed during that time.
CTF (Set Carry to Flag 1)	00110	The Special field code CTF stores the ALU carry bit in the Flag 1 FF.
DCSR (Decrement SR)	01001	The Special field code DCSR decrements the content of the SR Register by a count of one.
FHB (Flag to High Bit)	01101	The Special field code FHB transfers the content of the Flag 1 FF to bit 0 of the U-Bus.
HBF (High Bit to Flag 1)	01100	The Special field code HBF transfers the content of U-Bus bit 0 to the Flag 1 FF.
INCN (Increment Name)	01010	The Special field code INCN increments the content of the Name Register by a count of one.
INCT (Increment Counter)	01011	The Special field code INCT increments the content of the Counter Register by a count of one.

Table 5-7. Special Field Code Definitions (Continued)

Label and Name	Field Code	Description
INSR (Increment SR)	01000	The Special field code INSR increments the content of the SR Register by a count of one.
LBF (Low Bit to Flag 2)	01111	The Special field code LBF transfers the content of U-Bus bit 15 to the Flag 2 FF.
NOP	10111	No Operation
POP	10111	The Special field code POP moves the stack elements up one location such that the second element of the stack (S minus one) becomes the top element (S), etc. The previous top of stack element is lost. When executed, this is accomplished by decrementing the SR Register and incrementing the Namer Register. Refer to para 5-4.
POPA (Pop setting CCA)	10110	The Special field code POPA functions the same as Special field code POP with the addition that the condition code is set to CCL (01) if the T-Bus word is less than zero, to CCE (10) if the T-Bus word is equal to zero, or to CCG (00) if the T-Bus word is greater than zero
SCRY (Set Carry Bit)	10100	The Special field code SCRY sets Carry in the Status Register.
SDFG (Set Dispatcher Flag)	00101	The Special field code SDFG sets Dispatcher Flag (bit 12 of Interrupt Status Register CPX1).
SF1 (Set Flag 1)	10011	The Special field code SF1 sets CPU Flag 1 FF.
SF2 (Set Flag 2)	10000	The Special field code SF2 sets CPU Flag 2 FF.
SF3 (Set Flag 3)	00011	The Special field code SF3 sets CPU Flag 3 FF.

Table 5-7. Special Field Code Definitions (Continued)

Label and Name	Field Code	Description
SIFG (Set Interrupt Flag)	00100	The Special field code SIFG sets the Interrupt Flag (bit 11 of Interrupt Status Register CPX1).
SOV (Set Overflow)	11000	The Special field code SOV sets the status word overflow bit.

Table 5-8. MCU Option Field Code Definitions

Label and Name	Field Code	Description
ABS (Absolute)	00000	The MCU option code ABS specifies the ABS Register which may be read into S-Bus (12:15) with "RBR" or stored from U-Bus (12:15) with "SBR". This bank register is used as a scratch pad bank register by the microcode.
CMD (Command)	00010	The MCU option code CMD enables the bus options (BUS, BSP0, and BSP1) in the Store field to store the U-Bus into the address CPU output register, ACOR, and to initiate a low-request command. When selected, the ACOR is outputed to the MCU-Bus and the command and module number ("TO" lines) are obtained from the TO and MOP Registers.
CRL (Control)	00001	The MCU option code CRL enables the Store field bus options (BUS, BSP0, and BSP1) to load the TO and MOP Registers from the U-Bus. MOP(0:1) := U-BUS(10:11) Command TO(2:4) := U-BUS(13:15) Address The MOP Register then contains a user defined command for the module whose address is contained in the TO Register. The CPU freezes until any pending MCU requests are completed.
DATA	10001	The MCU option code DATA enables Store field bus options (as in CRL) to store the U-Bus into DCOR and to initiate a high-request command.
DB (DB-Relative)	10000	The MCU option code DB functions the same as ABS except that it specifies the DB-Bank Register used with DB-relative addressing.

Table 5-8. MCU Option Field Code Definitions (Continued)

Label and Name	Field Code	Description
DPOP (Data-POP)	10010	The DPOP MCU option code functions the same as DATA and also pops the stack.
NIR (Next Instruction Register)	01001	The MCU option code NIR enables the Store field bus options (as in CRL) to store the U-Bus into DCOR and to initiate a high-request command. On the following select cycle, DCOR is read into the MCU-Bus and stored in the CPU NIR Register.
OPND (Operand Register)	11001	Same as NIR except that the MCU-Bus is stored in the CPU OPND Register.
PB (PB-Relative)	01000	Same as ABS, except that it specifies the PB-Bank Register used in PB-relative addressing.
RND (Returned Data)	10100	The MCU option code RND enables the Store field bus options (as in CRL) to store the U-Bus in ACOR and to initiate a low-request command. The DB-Bank Register generates the module number used to initiate a data fetch from memory. The returned data is loaded into the NIR.
RNP	10111	Same as RND, except that the PB-Bank Register generates the module number.
RNS	11100	Same as RND, except that the Stack-Bank Register generates the module number.
ROA	00111	Same as RND, except that the ABS-Bank Register generates the module number and the returned data is loaded into the OPND Register.
ROD	10111	Same as RND, except that the DB-Bank Register generates the module number and the returned data is loaded into the OPND Register.

5-8. MCU Option Field Code Definitions (Continued)

Label and Name	Field Code	Description
ROND	10011	Same as RND, except that the returned data is stored in both the NIR and the OPND Register.
RONP	01011	Same as RNP, except that the returned data is stored in both the NIR and the OPND Register.
RONS	11011	Same as RNS, except that the returned data is stored in both the NIR and the OPND Register.
ROP	01111	Same as RNP, except that the re-returned data is stored in the OPND Register.
ROS	11111	Same as RNS, except that the returned data is stored in the OPND Register.
S	11000	Same as ABS, except that the Stack-Bank Register is specified and is used with DB-, Q-, or S-relative addressing.
WRA	00110	The MCU option code WRA enables the Store field bus options (as in CRL) to store the U-Bus in ACOR and to initiate a low-request command. The ABS-Bank Register generates the module number used to initiate a data store into memory. On the select cycle, the addressed memory module interprets the MCU-Bus data as an address and goes busy. The module stays busy until it receives the data to be stored (normally sent on the following cycle with a microcode BUS DATA instruction) and completes the write cycle, or until its timer runs down.
WRD	10110	Same as WRA, except that the DB-Bank Register generates the module number.

Table 5-8. MCU Option Field Code Definitions (Continued)

Label and Name	Field Code	Description
WRS	11110	Same as WRA, except that the Stack-Bank Register generates the module number.

Table 5-9. Skip Field Code Definitions

Label and Name	Field Code	Description
BIT6	00101	The Skip field code BIT6 sets the NOP2 FF if bit 6 of the U-Bus word is a logic 1.
BIT8	00110	The Skip field code BIT8 sets the NOP2 FF if bit 8 of the U-Bus word is a logic 1.
CRRY (Carry)	01000	The Skip field code CRRY sets the NOP2 FF if the ALU carry out is a logic 1.
CTRM (Counter Max)	11011	The Skip field code CTRM sets the NOP2 FF if the counter contains all ones.
EVEN	00010	The Skip field code EVEN sets the NOP2 FF if the U-Bus word is an even number (U-Bus bit 15 is a logic 0).
F1 (Flag 1)	01100	The Skip field code F1 sets the NOP2 FF if Flag 1 FF is set.
F2 (Flag 2)	01110	The Skip field code F2 sets the NOP2 FF if Flag 2 FF is set.
F3 (Flag 3)	11100	The Skip field code F3 sets the NOP2 FF if Flag 3 FF is set.
INDR (Indirect)	10100	The Skip field code INDR sets the NOP2 FF if the Indirect Bit FF is set and the indirect signal is a logic 1.
JLUI	11001	The Skip field code JLUI causes a microjump to the ROM address specified by the LUT providing the indirect condition (Skip field INDR code) is not met. If the indirect condition is met, the microjump is not executed.
NCRY	01001	The Skip field code NCRY sets the

Table 5-9. Skip Field Code Definitions (Continued)

Label and Name	Field Code	Description
NCRY (Cont)		NOP2 FF if the carry out from the ALU is zero.
NEG	01011	The Skip field code NEG sets the NOP2 FF if the U-Bus word is a negative number (U-Bus bit 0 is a logic 1).
NEXT	11101	Terminates current instruction and initiates the sequence necessary to begin execution of the next instruction. If stackop A has just been executed and stackop B is not a NOP, then the hardware executes stackop B. Otherwise, the action shown in the timing figure below takes place (a,b,c,d,e,f are equal length CPU clock cycles).

```
       a              b         c        d            e            f

...Mem. Sel.                          NEXT         NOP2         Execute
   cycle                                                        1st line
   DATA->NIR     NIR->LUT    ...   BUSL,RWP     P+1->P          of mic-
                                   Issue LOREQ  Select cycle    rocode
                                   LUT->        RANK1->         of new
                                     VBUS->       RANK2         instr.
                                     ROM->      (If memory
                                     RANK1       reference,
                                   NIR->CIR      force PADD,
                                                 BASE to R,
                                                 S-BUS Reg's.)
```

Time periods a, b, c (if present), and d occur in the currently executing instruction. Time periods a and b must occur before d for maximum execution speed. Otherwise, a CPU freeze will occur at d. Time periods a and b result from the next instruction prefetch of the

Table 5-9. Skip Field Code Definitions (Continued)

Label and Name	Field Code	Description
NEXT (Cont)		current instruction. Time period c may or may not be present depending on the length of the instruction. Time period d is the last line of the current instruction. It initiates a next instruction prefetch, transfers NIR to CIR, and applies the address on the VBUS (normally using the LUT output) to the ROM input. The ROM word at this address is stored in RANK1. In addition, the NOP2 FF is set. Time period e is used to increment the P Register, transfer RANK1 to RANK2, and, if the new instruction is a memory-reference type, load the R- and S-Bus Registers with the Preadder output and the proper base register. This is also the select cycle for the next instruction prefetch if there is no MCU conflict. During time period f, the first line of the new instruction is executed. The above is the normal sequence of operation of NEXT. This sequence is modified in the event an interrupt is pending or the microcode line is "...DATA NEXT". NEXT also clears F1, F2, F3, CNTR, Subroutine Flag FF, and the ABS-Bank Register.
NF1 (Not Flag 1)	01101	The Skip field code NF1 sets the NOP2 FF if Flag 1 FF is cleared.
NF2 (Not Flag 2)	01111	The Skip field code NF2 sets the NOP2 FF if Flag 2 FF is cleared.
NOFL (Not Overflow)	00111	The Skip field code NOFL sets the NOP2 FF if the ALU overflow bit is not a logic 1. Causes conditional jump and JSB to be two-cycle instructions.

Table 5-9. Skip Field Code Definitions (Continued)

Label and Name	Field Code	Description
NOP	11111	No operation.
NPRV (Not Privileged)	10110	The Skip field code NPRV sets the NOP2 FF if the privileged mode bit (status word bit 0) is zero.
NSME (Not Same)	00100	The Skip field code NSME sets the NOP2 FF if all bits of the T-Bus are not the same.
NZRO (Not Zero)	00001	The Skip field code NZRO sets the NOP2 FF if the T-Bus word is not equal to zero.
ODD	00011	The Skip field code ODD sets the NOP2 FF if the U-Bus word is an odd number (U-Bus bit 15 is a logic 1).
POS (Positive)	01010	The Skip field code POS sets the NOP2 FF if the U-Bus word is a positive number (U-Bus bit 0 is a logic 0).
RSB (Return from Subroutine)	11000	The Skip field code RSB causes a microjump to the ROM address contained in the Save Register.
SR4 (SR=4)	10010	The Skip field code SR4 sets the NOP2 FF if the SR Register content is equal to four.
SRL2 (SR<2)	10101	The Skip field code SRL2 sets the NOP2 FF if the SR Register content is less than two.
SRL3 (SR<3)	10111	The Skip field code SRL3 sets the NOP2 FF if the SR Register content is less than three.
SRN4 (SR Not 4)	10011	The Skip field code SRN4 sets the NOP2 FF if the SR Register content is not four.
SRNZ (SR Not Zero)	10001	The Skip field code SRNZ sets the NOP2 FF if the SR Register content is not zero.

Table 5-9. Skip Field Code Definitions (Continued)

Label and Name	Field Code	Description
SRZ (SR Zero)	10000	The Skip field code SRZ sets the NOP2 FF if the SR Register content is equal to zero.
TEST	11010	The Skip field code TEST sets the NOP2 FF if any enabled interrupt is pending.
UNC (Unconditional)	11110	The Skip field code UNC and/or unconditional JMP´s set the NOP2 FF.
ZERO	00000	The Skip field code ZERO sets the NOP2 FF if the T-Bus word is equal to zero.

NOTES

NOTES

This section contains principles of operation and servicing information for the computer system's Module Control Unit (MCU) and Main Memory.

6-1. MCU OPERATIONS

As previously discussed in paragraph 2-15, each computer module gains access to the CTL Bus through its MCU. For any given module, the MCU may be located on a single dedicated PCA, distributed on multiple PCA's, or located on a small part of a PCA. Whatever its physical configuration may be however, each MCU performs the same function of interfacing its associated module with all other modules connected to the CTL Bus. Although the following discussion of MCU operations is specifically for the Central Processor Module MCU and Main Memory's MCU logic circuit, it is representative of any of the other MCU logic circuits. Since the purpose of the MCU is to control CTL-Bus transmissions, its operations will be discussed dynamically by following the sequence of logical operations involved for each of the different types of CTL-Bus transmissions between the CPU and memory. (For information concerning the MCU's operations with the I/O Processor, refer to Section VII.)

6-2. Fetch Next Instruction Operations

The operations involved in order to fetch an instruction from memory consist of three major steps.

a. The CPU transmits the address of an instruction word to memory and tells memory what to do with that address.

b. Memory receives the address, reads the contents of the addressed location, and transmits the contents back to the CPU.

c. The CPU receives the instruction word and loads it into the NIR.

6-3. CPU ADDRESS TRANSMIT. When a NEXT instruction is decoded from the ROM Skip field, a NEXT signal loads the contents of the P Register (address of instruction to be fetched) into the ACOR Register. (See figure 2-20.) NEXT also transfers the contents of the NIR into the CIR. The CPU executes the CIR contents. The objective is to refill NIR while the CIR instruction is being executed so as to implement the CPU instruction look-ahead feature. Assuming that the transmission may proceed, NEXT sets the CPU Low Request (LREQ) flip-flop (figure 6-1) in the MCU. (The difference between low request and high request is that low request always checks to see if the destination module is ready to

Figure 6-1. MCU Simplified Logic Diagram

receive a transmission because a memory operation is being initiated, whereas high request assumes that the destination module is expecting the transmission of data to complete a memory write operation.) By this time, the MCU Encoder has encoded the appropriate memory opcode (MOP), which is now in the MOP Register. The memory opcode is a two-bit code which tells memory what to do when it receives bus data. There are three possible memory opcodes: No Operation (NOP), Write (W), and Read (R). NEXT locks the code in the MOP Register and sets the Next In Process (NIP) flip-flop. Setting NIP opens the next instruction register so that all CTL-Bus transmissions are gated to NIR until NIP is reset. NEXT also locks the TO Register which now contains the destination module number.

The LREQ signal reads the contents of the TO Register into the Ready Comparator which checks the Ready (RDY) line for the intended destination to see if that module is ready to receive. If not, nothing further happens until the the RDY line is true. The output of the Ready Comparator, through a set of changeable jumpers, pulls the Enable (ENB) line low for this module number. Since no module can transmit unless all ENB lines of higher priority modules are high, pulling the ENB line low disables all lower priority modules. Provided that no higher priority module has pulled its ENB line low to this module (through a second set of jumpers), and provided the I/O Processor is not requesting the bus, the output of the Ready Comparator now sets the CPU Select (SEL) flip-flop. The SEL signal reads out the ACOR contents to the CTL Bus as well as TO and FROM module numbers and the memory opcode. SEL also pulls the destination module's RDY line low for one cycle so that other modules will not assume that the memory module is ready before memory has a chance to pull the RDY line low itself on the next cycle.

6-4. MEMORY RECEIVE AND TRANSMIT. The memory module's TO Comparator (figure 6-2) identifies the code on the TO lines as its own module number and sets the Ready flip-flop and Address Latch flip-flop which locks the address word from the bus into the Address Register and the FROM address into the From Register. The Ready signal also keeps the module's RDY line pulled low (the CPU had pulled it low temporarily in the preceding cycle) and, together with the decoded memory opcode, begins the read memory cycle. The addressed memory location is read into the Read Data Register. Meanwhile, on the next clock edge, the MCU begins the process of requesting access to the bus by setting the Enable flip-flop. (Since memory transmits only to modules that are expecting the transmission, only high requests are used.) The Enable signal pulls its enable (ENB) line low to lower priority modules and, provided no high priority module has pulled low on its ENB to this module, sets the Data Out flip-flop on the next clock edge. The memory location contents are in the Read Data Register and the Data Out signal reads the contents out to the CTL Bus. The Data Out signal also reads out the wired FROM code and TO code (which is simply the saved FROM code, since transmission is back to the CPU).

Figure 6-2. Memory Module Simplified Logic Diagram

6-5. CPU RECEIVE. The MCU's TO Comparator (figure 6-1) identi-
fies the code on the TO lines as its own module number, and gives
a true output. Also, the FROM Comparator identifies the trans-
mission as the one it is waiting for by comparing the saved TO
Register contents with the FROM lines of the bus; it therefore
also gives a true output.   If the FROM code is not the expected
one, it is loaded into the MOD Register and a module interrupt is
generated to the CPU.   The two true outputs together reset the
NIP flip-flop. The NIR, which up until now has been loading all
bus transmissions into itself, is now inhibited from further
loading because it now contains the expected next instruction.

6-6. Fetch An Operand Operations

The operations for fetching an operand from memory are very sim-
ilar to the operations for fetching an instruction.   The main
differences are that the initiating signals are different and the
receiving register is the Operand (OPND) Register rather than the
NIR. Therefore, the following discussion primarily only gives
the overall flow of information.   Refer back to paragraphs 6-2
through 6-5 for additional details.

6-7. CPU ADDRESS TRANSMIT. The process of sending an address to
memory begins when a signal from the ROM Store field loads the
U-Bus contents into the ACOR Register on the CPU and sets the
LREQ flip-flop on the MCU. (See figures 2-20 and 6-1.) The MCU
Operation Encoder gives a memory opcode to the MOP Register and
sets the Operand in Process (OPINP) flip-flop.   The LREQ signal
causes the Ready Comparator to check if the destination module is
ready and, if so, enters the priority. When priority allows (ENB
present), the Select flip-flop is set, causing the address stored
in ACOR to be read out to the CTL Bus.

6-8. MEMORY RECEIVE AND TRANSMIT. The memory module (figure 6-
2), after recognizing its TO code and setting the Ready flip-
flop, locks the address from the bus into the Address Register.
The Ready signal, together with the decoded memory opcode, ini-
tiates the reading of the addressed location into the Data Regis-
ter. Meanwhile, the Enable flip-flop is set and priority is
established.   When the module has priority to use the bus on the
next clock cycle, the Data Out flip-flop is set causing the oper-
and, now in the Read Data Register, to be read out to the CTL
Bus.   The saved FROM code is used to identify the destination
(TO) as the CPU.

6-9. CPU RECEIVE. The TO and FROM Comparators together cause
the OPINP flip-flop to reset, thus locking the operand from the
bus into the OPND Register.

Note

If the CPU is frozen awaiting the operand, the operand
in addition to being loaded into the OPND Register, is
also loaded into the CPU S-Bus Register, thus saving one
clock of instruction execution time.

6-10. Store An Operand Operations

The operations for storing an operand in memory involves much the same logic operations that were discussed in the preceding fetch transmissions. The main difference is that instead of a CPU to memory transmission and then a memory to CPU transmission, there are two consecutive transmissions from CPU to memory. The first transmission is the address and the second is the operand. The following paragraphs are again condensed to illustrate only the overall flow of information.

6-11. CPU ADDRESS TRANSMIT. A signal from the ROM Store field loads the U-Bus contents into the ACOR Register on the CPU and sets the LREQ (low request) flip-flop on the MCU. (See figures 2-20 and 6-1.) The MCU Operation Encoder gives a memory opcode to the MOP Register. In this case, the opcode is Write rather than Read as in the previous cases. (Neither the NIP nor OPIND flip-flops are set.) After checking to see if the destination module is ready and the Enable (ENB) signals are present, the LREQ signal sets the Select flip-flop which causes the address to be read out to the CTL Bus.

6-12. MEMORY RECEIVE. The memory module (figure 6-2) after recognizing its TO code and setting the Ready flip-flop, locks the address from the CTL Bus into the Address Register. The FROM, MOP, and Address Registers remain locked and the RDY line goes low so that no other module can send a new address to this memory module.

6-13. CPU DATA TRANSMIT. Meanwhile, the CPU has put the operand on the U-Bus, and a DATA signal from the ROM Store field loads it onto the CPU DCOR (figure 2-20). The DATA signal also sets the High Request (HREQ) flip-flop on the MCU (figure 6-1). Destination readiness does not need to be checked, however, since memory is expecting a data transmission from this module. After priority checks, the HREQ signal sets the CPU Select flip-flop which reads out the operand to the CTL Bus. (The memory opcode is NOP, since memory is already holding the appropriate opcode.)

6-14. MEMORY RECEIVE. In the memory module the TO Comparator recognizes its TO code and the FROM Comparator verifies transmission from the correct module. The true ouputs from both of these comparators cause the operand from the bus to be loaded into the Write Data Register and cause the memory timing to start the memory write cycle. This causes the operand to be stored into the addressed location.

6-15. Command A Module

The machine instruction set includes a Command (CMD) instruction that permits privileged mode programs to issue commands directly to a module (assuming the module is equipped to handle such commands). When programmed, the CMD instruction takes a 16-bit word from the TOS and sends it to a module whose module number (and

two-bit opcode) are given in another word in the stack. (Refer to Section IV for the CMD instruction definition.)

WARNING

> The normal checks and limitations that apply to the standard users in MPE are bypassed in privileged mode. It is possible for a privileged mode program to destroy file integrity, including the MPE operating system software itself. Hewlett-Packard cannot be responsible for system integrity when programs written by users operate in the privileged mode.

The ROM MCU field codes of Control (CRL) and Command (CMD), in that order, are used to effect the execution of the CPU instruction CMD. When the hardware decodes a CRL in the MCU field, it gates bits 13 through 15 from the U-Bus, through the TO MUX lines in the MCU, and these bits are clocked into the TO Register (figure 6-1). U-Bus bits 10 and 11 are clocked into the MOP Register. A following line of microcode then executes a CMD MCU function which gates that line's U-Bus data into ACOR and issues a Low Request (LREQ).

The MCU logic performs the normal sequence of checking for bus priority. Then the contents of the TO Register are gated to the TO lines of the CTL Bus, the contents of the FROM jumpers in the MOD Register are gated to the FROM lines on the CTL Bus, and the contents of the MOP Register are gated to the MOP lines of the CTL Bus. The contents of ACOR (figure 2-20) are placed on the MCUD lines of the CTL Bus. What affect the MOP and MCUD lines have on the addressed module depends on the design of the module.

When a module needs to communicate with the CPU, it cannot pass a data word to the MCUD lines because the MCU is not expecting the communication and will not gate the MCUD lines back. The CTL Bus, however, since it does not use a handshake signal sequence, is monitoring the bus at all times and the CPU is able to detect that the module is trying to communicate. The hardware generates a module interrupt and sets CPX1 (bit 7).

When the module has priority to use the CTL Bus, it places the CPU module number on the TO lines, CPU address on the FROM lines, and a value on the MOP lines. The MCU logic recognizes its own address, but because it is not expecting a value from the calling module, the FROM lines value is sent to bit positions 5, 6, and 7 of the MOD Register and the MOP value is sent to bit positions 2 and 3. (See figure 6-1.) The microcode will fetch these values

from the MOD Register and pass them to the interrupt handling CPU instruction. To command the TO ME - FROM ME lines, the CPU instruction XEQ takes the 16-bit TOS value and executes it as an instruction. In order to do this, the value must be placed in the NIR before it can be gated to the CIR to be executed. Because the CTL Bus is the only data path into NIR, the value is placed on the FROM lines of the CTL Bus with an address of TO the CPU (in essence, TO ME). The FROM lines will have the CPU's port number (FROM ME).

The TO ME - FROM ME action is caused by a code of NIR in the MCU field of a microinstruction. When the MCU detects the NIR code, the High Request (CPU HREQ) flip-flop and the Next-In-Process (NIP) flip-flop (figure 6-1) are set and the value placed on the U-Bus is clocked into DCOR (figure 2-20). The hardware places a NOP code in the MOP Register and clocks the FROM jumper value through TO MUX into the TO Register. The MCU, when granted priority to use the CTL Bus, gates the contents of the TO, MOP, and DCOR Registers onto the CTL Bus. Because the MCU for the CPU is monitoring the CTL Bus at all times, the TO Comparator recognizes that the CPU is being addressed and that the CPU is expecting the communication (determined by the fact that the values on the FROM lines match the contents of the TO Register). The MCU, therefore, will accept the value on the MCUD lines and, because the NIP flip-flop has been set, the value is clocked into NIR. The same sequence as described for TO ME - FROM ME will occur if an Operand (OPND) code is decoded in the MCU field of a microinstruction. The only difference is that the MCU's OPINP flip-flop (figure 6-1) is set instead of the NIP flip-flop and the value will be gated into the OPND Register instead of the NIR.

6-16. MCU SERVICING INFORMATION

The Central Processor Module MCU PCA is a nonrepairable PCA and must be replaced if found defective. No repair procedures are required. However, the MCU PCA does contain jumpers that must be properly configured. The jumper configurations are discussed in paragraphs 6-17 through 6-21 and identified in figure 6-3. (Servicing information for other computer module MCU circuits is provided with the discussions of the individual modules.)

6-17. ENABLE. The ENABLE signal is used to establish priority for accessing the CTL Bus. The insertion of jumpers W1 through W4 establish the priority of the MCU PCA. Jumper W1 must be installed.

6-18. READY. The READY signal is used to signify that a module is ready to communicate with memory. The insertion of jumpers W5 (READY 4) through W7 (READY 6) determine which READY line is assigned to the MCU PCA. Jumper W6 (READY 5) must be installed.

6-19. CPU NUMBER. The system is designed to have only one CPU which must be designated as CPU number 1. Ensure that jumper W10 is installed and that jumper W11 has been removed.
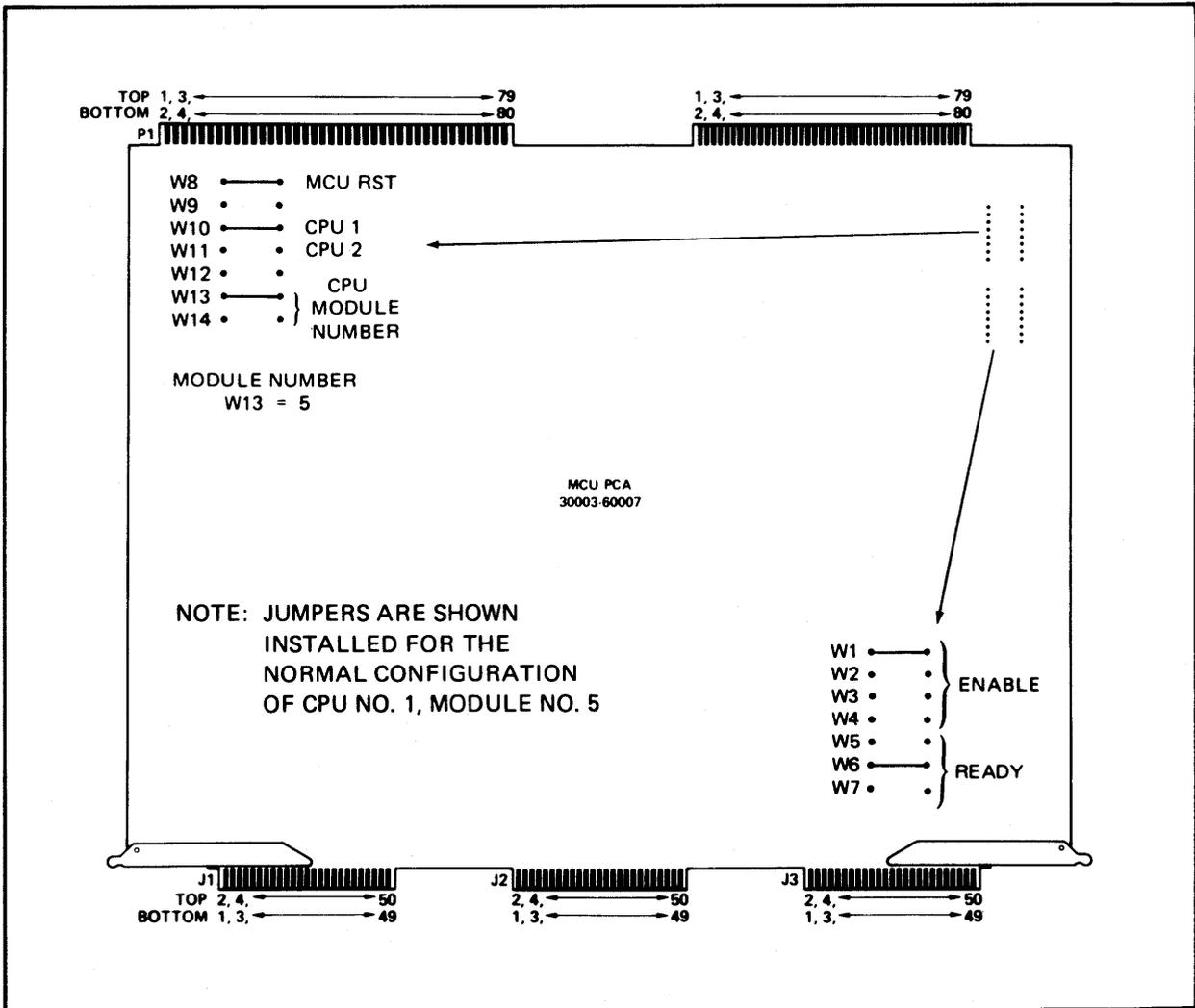
Figure 6-3. MCU PCA Jumper Locations

6-20. CPU MODULE NUMBER. The CPU connected to the CTL Bus is as-signed a module number between 4 and 6 by insertion of jumpers W13 and W14. The module number 5 must be assigned to the CPU by installing jumper W13.

6-21. MCU RESET. Jumper W8 is always installed.

6-22. MAIN MEMORY

As previously discussed in paragraph 2-11, Main Memory is an ex-pandable memory module that consists of at least three PCA's as shown in figure 6-4.

6-23. Memory PCA Interfacing

6-24. CTL BUS. As shown in figure 6-4, the CTL Bus provides for the transfer of data between the memory module and the other mod-
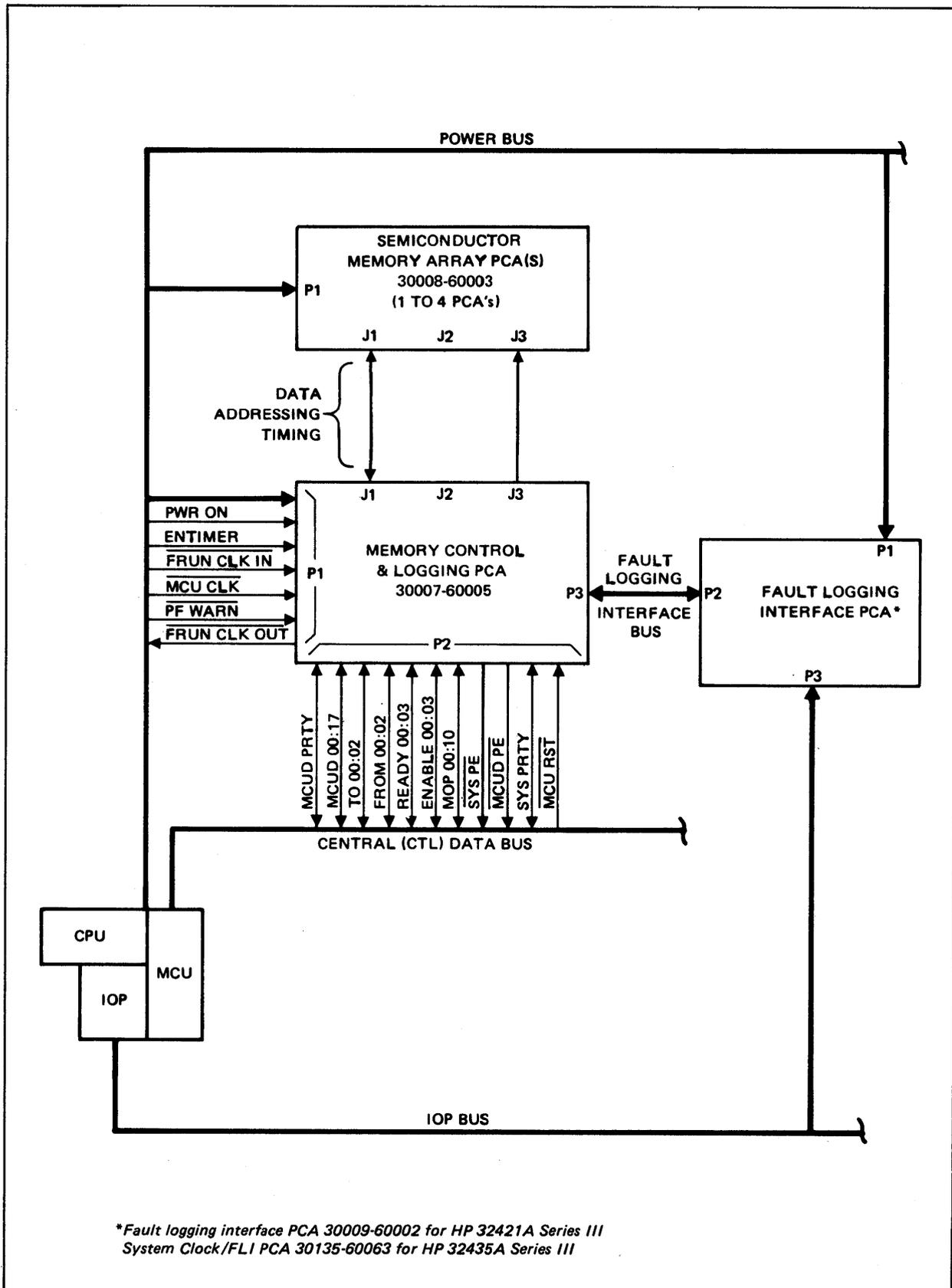
Figure 6-4. Memory Module Interface Diagram

ules on the bus via the MCUD (00-15) lines. Data parity is pro-
vided by the MCUD PRTY line. Addressing information used to
select a specific address of a word in memory is transferred via
the data lines. Control information is provided for the follow-
ing functions:

a.   To select a module, TO (00-02).

b.   To designate the requesting module, FROM (00-02).

c.   To indicate whether the module is ready or not, READY
     (00-03).

d.   To resolve priority considerations, ENABLE (00-03).

e.   To select a specific memory operation, MOP (00-10).

f.   To indicate a parity error on transfer of control information
     and data parity error during a write cycle, NSYS PE.

g.   To indicate parity for the control lines, SYSPRTY.

h.   The "not" master reset (NMCU RST) pulse initializes the MCL
     PCA during initial power turn-on and during power failure.
     Control can be reset during a refresh cycle without loss of
     stored data.

i.   To indicate an address parity error, NMCUD PE.

6-25. IOP BUS. The FLI PCA interfaces with the IOP Bus to exe-
cute direct I/O commands. The FLI PCA interrogates the MCL fault
logging array and stores the contents into the FLI I/O logging
array. Commands on the IOP Bus then cause a read of the I/O log-
ging array and transfer the contents into a disc file for future
analysis.

6-26. FAULT LOGGING INTERFACE BUS. The Fault Logging Interface
Bus is a flat cable connected between P3 on the MCL PCA and P2 on
the FLI PCA. All communication between these two PCA's occurs on
this bus. When a memory is configured above 512K words, this bus
extends to P3 of the upper 512K MCL PCA. A single FLI PCA can
communicate independently over the bus to the selected (upper or
lower 512K) MCL PCA.

6-27. POWER BUS. The CPU applies an ENTIMER signal to the MCL
PCA via the Power Bus to enable a timer during every write cycle.
If the data is not received within 2.87 msec during a write oper-
ation, the timer resets the control circuits to prevent memory
from waiting for data. No loss of memory data occurs during an
incomplete write cycle. The system clock (NMCUCLK) is a 175-nsec
square wave from the CPU that provides timing to the MCU cir-
cuits. The NMCUCLK can be halted and pulsed for maintenance pur-
poses, NMCUCLK can be disconnected and replaced with an external
timing signal. Refresh clock timing is automatically selected
from an oscillator internal to the MCL PCA or from the NFRUNCLK

signal input. NFRUNCLK is in sync with NMCUCLK, but cannot be halted with the single-cycle switch. Power failures are sensed in the power supplies (Sections IX and X) to initiate the NPF WARN (power fail warning) to the CPU and memory. The MCL PCA guarantees 3.0 msec for the CPU to execute its power fail routine and store the necessary information into memory. After this 3.0-msec interval, clock timing to memory is disabled to prevent any further read or write operations until power is restored.

6-28. Memory PCA Descriptions

Brief descriptions of the memory PCA's are contained in paragraphs 6-29 through 6-31. The various memory PCA configurations are discussed in paragraph 2-11.

6-29. SMA PCA. The SMA PCA contains the data and check bit storage array for the 128K-word dynamic semiconductor memory. The address/data receivers and drivers that interface with the MCL PCA are also contained on this PCA. All communication with the SMA PCA is governed by the MCL PCA. SMA PCA operations are discussed in paragraphs 6-32 through 6-36.

The individual semiconductor memory chip is a 16K by 1 storage device (16 thousand one bit words). On the SMA PCA, the chips are physically arranged in eight rows with 22 chips on a line as shown in figure 6-5. Since a maximum of eight SMA PCA's can be contained in memory, each SMA PCA contains Switch S1 that identifies the portion of memory represented by that SMA PCA. Refer to paragraph 6-48 for the proper setting of S1.

6-30. MCL PCA. The MCL PCA contains the read/write control circuits, MCU logic, refresh circuits, address and data registers, MCU logic, refresh circuits, error logging array, and error correction logic for the memory module. This PCA also contains the check bit parity generators and checkers. Since the MCL PCA can support up to four SMA PCA's, it must be configured to its associated memory module size. Refer to paragraph 6-49 for the various switch settings. MCL PCA operations are discussed in paragraphs 6-32 through 6-36.

6-31. FLI PCA. The FLI PCA (part no. 30009-60002 for HP 32421A Series III and part no. 30135-60063 for HP 32435A Series III) contains the control circuits and I/O logging array for interrogating the MCL PCA's error logging array. Refer to paragraph 6-50 for FLI PCA switch settings. FLI PCA operations are discussed in paragraphs 6-32 through 6-36.

6-32. Memory Operations

Memory has the following operating modes and specifications:

a. WRITE; 700 nsec cycle time (minimum)
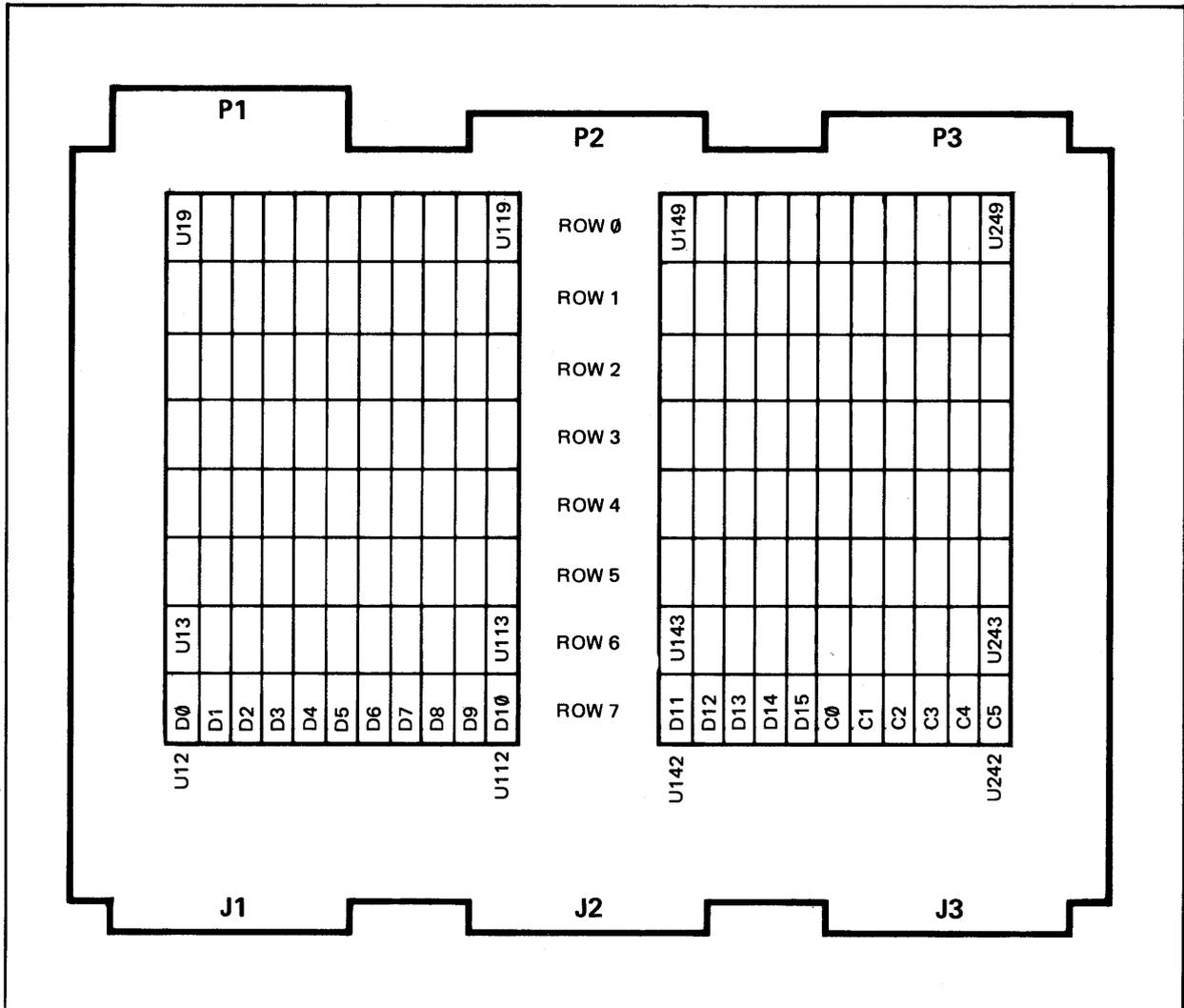
Figure 6-5. SMA PCA Chip Arrangement

b.   READ; 350 nsec access, 700 nsec cycle time

c.   NO OPERATION (NOP); 700 nsec cycle time

These operations plus fault correction and error logging are discussed briefly in the following paragraphs.

6-33. READ. A read operation (MOP 10) outputs 17 data bits and a parity bit from the addressed location to the requesting module via the CTL Bus. Refer to paragraphs 6-2 through 6-14.

6-34. WRITE. A write operation (MOP 01) loads 17 data bits and a check bit into a given address location. Two transmissions to the memory module are required to initiate a write operation. Refer to paragraphs 6-2 through 6-14.

6-35. NOP. A NOP (No Operation) memory operation (MOP 00) is similar to a read operation. However, no data is transferred to the originating module. The NOP can occur during an address phase when there is a system address parity error. During a write operation, the MOP code sent with the data is a NOP.

6-36. FAULT CORRECTION AND ERROR LOGGING. During a read operation, the error handling circuits detect, log, and correct all single-bit data errors. These circuits also detect double-bit errors and force bad CTL Bus data parity to alert the receiving module. Error checking and correction takes place during the normal memory cycle. An error logging scheme uniquely reports all single-bit errors (or groups of double-bit error pairs) so that problem chips on the SMA PCA's can be replaced during scheduled maintenance. Refer to paragraphs 6-38 through 6-42.

6-37. Memory Servicing Information

Since the fault correction and error logging features of memory are useful maintenance tools, they are discussed briefly in paragraphs 6-38 through 6-47 on a how-to-use basis when maintaining, troubleshooting, or repairing Main Memory. For a more detailed discussion of these features, refer to the Stand-Alone Memory Diagnostic D430B, part no. 30000-90004. In addition, the repair philosophy for each of the memory module PCA's are discussed in paragraphs 6-48 through 6-50.

Note

Throughout the following fault correction and error logging discussion, the term "FLI PCA" pertains to the Fault Logging Interface PCA, part no. 30009-60002 for the HP 32421A Series III and to the System Clock/FLI PCA, part no. 30135-60063 for the HP 32435A Series III.

6-38. FAULT CORRECTION. The fault correction logic on the MCL PCA generates a six-bit check field for each 16-bit memory word and stores this field along with the data on the SMA PCA. The construction of the check field is shown in figure 6-6. It should be noted that check bits C5, C3, C1, and C0 are generated to provide even parity for the eight data bits they check and that check bits C4 and C2 are generated to provide odd parity for the eight bits they check. Each check bit is generated over a different eight bits of the data word. The check bits are generated on each data word for each write operation to memory. The hardware checks parity over the same data word when the location is read back from memory. The read from memory includes the check bits processed on a write operation. If the resulting check of the data with the check bits is zero, no error has occurred in the write-read sequence. If the resulting check of the data word with the check bits is non-zero, the parity checker outputs H01 through H05 which are decoded as shown in figure 6-7 and any single data bit error is corrected by complementing that bit.

Figure 6-6. Error Correction Codes

		DATA BITS																CHECK BITS						PARITY
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	C0	C1	C2	C3	C4	C5	
*5BITS	LSB	X	X	X	X	X	X	X	X														X	EVEN
		X	X	X	X					X	X	X	X									X		ODD
		X				X	X							X	X	X					X			EVEN
	MSB		X			X		X		X		X		X	X		X			X				ODD
				X			X		X				X	X	X	X	X		X					EVEN
					X			X	X		X		X		X	X	X	X						EVEN
	5B*	07	13	23	03	15	25	11	21	16	06	32	22	34	14	24	30	00	20	10	04	02	01	

*5B = 5-bit octal error logging address H01 — H05

207

ERROR LOGGING ARRAY ADDRESS STRUCTURE

BD0	BD1	ROW0	ROW1	ROW2	H01	H02	H03	H04	H05

OCTAL
SMA ADDRESS
(0-3)

OCTAL
ROW ADDRESS
(0-7)

5-BIT OCTAL ADDRESS OF
DATA BITS 0-15 AND
CHECK BITS C0 - C5

**5-BIT OCTAL ADDRESS TO DATA OR CHECK BIT
CONVERSION TABLE**

H01 – H05	BIT		H01 – H05	BIT
00	C0		20	C1
01	C5		21	D7
02	C4		22	D11
03	D3		23	D2
04	C3		24	D14
05	Note 1		25	D5
06	D9		26	
07	D0		27	
10	C2		30	D15
11	D6		31	
12	Note 2		32	D10
13	D1		33	
14	D13		34	D12
15	D4		35	
16	D8		36	
17			37	

Note 1. Forced double error write.
Note 2. Missing SMA.

Figure 6-7. Decode of H01 through H05

Check bit C0 is instrumental in detecting double-bit errors. Double-bit errors are logged, but a unique address is not generated for each double-bit error pair and errors are not corrected. Instead, bad parity is generated on the CTL Bus requesting the receiving module to flag the error. The MCL PCA includes logic circuits to detect double errors and to force bad CTL Bus parity. The MCL PCA also includes logic circuits to log an error in the Error Logging Array (ELA) during read operations.

For diagnostic purposes, the SMA PCA(s) can be programmed through the FLI PCA to disable write error correction. The check bits write control logic can be disabled during a write operation to allow data bits to be changed without modifying the check bits. Error correction is always enabled during read operations. All detected errors are recorded regardless whether they are single bit or multiple bit. Each single-bit detected error is logged according to chip location in the ELA. The ELA uses static a 1K RAM located on the MCL PCA. Five bits of memory address and five bits of error code determine the error logging address in which a "1" is stored in the ELA.

The 1K RAM is organized as a 1024 x 1 bit array requiring 10 bits of address. The five most significant bits of ELA address correspond to the five most significant bits of memory address; two bits define one of four SMA PCA's and three bits define one of eight rows of chips on the particular SMA PCA. The five least significant bits define one of 22 chips in a particular row and correspond to the error code H01 through H05. Refer to figures 6-5 through 6-7. The ELA can be read under program control by means of the FLI PCA. Any ELA location containing a "1" signifies that an error was detected. The location of the faulty chip can be found by using the 5-bit octal address portion of the ELA address and then looking up the bit in the conversion table of figure 6-7.

6-39. MEMORY ERROR LOGGING FACILITY. The memory error logging facility permits the system's user to examine the error history of Main Memory. The facility consists of the following elements.

a. Error correcting memory and the FLI PCA

b. Memory error logging system process (MEMLOGP)

c. Memory error log analysis program (MEMLOGAN)

d. Memory error logging internal update program (MEMTIMER)

Memory error logging is in no way connected to or related to standard system logging. Both function independently. None of the operator interfaces to system logging have an effect on memory logging. Memory error logging will always be invoked if error correcting memory is present in the system. MEMLOGP is a system process that runs under MPE. Once initiated, MEMLOGP automatically and periodically interrogates the FLI PCA to obtain the latest error information. MEMLOGP is activated during the

initialization phase of MPE (coolstart, warmstart, coldstart, re-
load, update). MEMLOGP cannot be activated at any other time.
If MEMLOGP cannot be initiated successfully during initializa-
tion, another attempt cannot be made until the system is brought
up again. Initially, MEMLOGP attempts to obtain the status of
the FLI PCA (DRT 2). If the PCA does not respond, error correct-
ing memory is absent from the system and MEMLOGP immediately ter-
minates. If this occurs, no message relating to error logging
will appear at the system console. If MEMLOGP initiation is suc-
cessful, the following message appears at the system console:

     ST/<TIME>/MEMORY ERROR LOGGING INITIATED

The message occurs after the DATE/TIME messages and before the
WELCOME message. MEMLOGP attempts to open the system memory er-
ror disc file MEMLOG (MEMLOG.PUB.SYS). If no such file exists on
the system, a new file will be created. If MEMLOG already ex-
ists, the file will be opened without altering the information
contained in the file. The file remains open as long as the sys-
tem is up. During those periods when MEMLOGP is accessing the
file, it will lock and unlock the file as necessary to ensure no
other access to it. The log analysis program (MEMLOGAN) similar-
ly locks and unlocks the file when it is accessing it. If an
operational error is encountered by MEMLOGP, the process will
display an error message and terminate. The message displayed
is:

     ST/<TIME>/MEMORY LOGGING ERROR#<ERRNUM>.LOGGING STOPPED

The range and definitions for <ERRNUM> are:

         1-10 Internal MEMLOGP errors.
            1 FLOCK error on MEMLOG file.
            2 FUNLOCK error on MEMLOG file.
            3 TIO error. Error logging hardware not ready.
            4 CIO error during copy operation from logging array.
            5 RIO error during scan of logging array. (Errors 6-10
              reserved for future use.)
       20-500 File system errors involving MEMLOG file. All file
              errors encountered by MEMLOGP are fatal to the process
              and cause it to terminate. Refer to the MPE Intrinsics
              Reference Manual, part no. 30000-90010 for definitions
              of the file system error numbers.

Once the MEMLOG file has been opened, MEMLOGP periodically inter-
rogates the error logging array in the MCL PCA via the FLI PCA.
If errors have occurred, the MEMLOG file is updated. The error
logging array is interrogated via the FLI PCA when MEMLOG is
first activated and thereafter once approximately every hour. If
one or more errors have occurred in the hour interval since the
last log update, MEMLOGP will perform the following operations:

a. Read the appropriate MEMLOG from disc.

b. Scan the error logging array for errors.

c. Update the error counter in the   MEMLOG record for each loca-
    tion where an error occurred.

d. Reset the error logging array to a no-error condition.

e. Write the updated MEMLOG record to disc.

MEMLOGAN (MEMLOGAN.PUB.SYS) is the utility that reads and inter-
prets the error information logged and kept in the MEMLOG file.
Because of the security placed on the MEMLOG file by the system,
MEMLOGAN can successfully run from an account other than PUB.SYS
if:

a. The RELEASE command was entered for MEMLOG by the system
    manager, or

b. The log on account has system manager capability.

MEMLOGAN will read MEMLOG and output its contents in a meaningful
manner. The formal designator of the output device is "OUT". To
direct output to the line printer, the following file equation
must be entered:

     :FILE OUT;DEV=LP

There is no user dialogue with MEMLOGAN. However, certain MEMLOG
file handling operations are available through the PARM parameter
of the RUN command. The following PARM values are recognized by
MEMLOGAN:

PARM=0 Causes the current contents of MEMLOG to be printed on the
       output device. The contents of the file will not be
       changed. This is the default PARM value.

PARM=1 Causes the current contents of MEMLOG to be printed on
       the output device after which the file is reset to a
       no-error state. All previously logged errors are deleted
       from the log file.

                              Note

             When a system with error correct-
             ing memory is initialized for the
             first time or the memory size is
             changed to cross the 512K bound-
             ary, MEMLOGAN should be run with
             PARM=1 as soon as the system is up
             and running. This will ensure a
             clean MEMLOG file and that subse-
             quent error counts are valid.

PARM=2 Causes the current contents of MEMLOG to be printed on the
output device after which the file is deleted from the
system. (This is the only way to remove the MEMLOG file
from the system and normally only the system manager would
use this PARM value.)

MEMTIMER (MEMTIMER.PUB.SYS) is the utility program which allows
the user to modify the interval of time between successive memory
log updates. The normal default interval is 60 minutes. This
interval provides the average installation with an adequate log
of the memory system. For other reasons, it may be desirable to
modify the interval to allow increased monitoring of the memory
performance. It is the function of MEMTIMER to modify the time
interval. MEMTIMER alters the current timing interval to a new
value and terminates the current interval. This causes MEMLOGP
to update the memory log file immediately and periodically
thereafter according to the new interval specified. A new timing
interval is specified through the PARM parameter of the RUN com-
mand. There is no user dialogue with MEMTIMER. The PARM value
is given as a positive integer greater than zero which represents
the number of seconds between log file updates. To begin memory
logging at 10 second intervals, the following RUN command would
be entered:

    :RUN MEMTIMER;PARM=10

To return logging to the default interval (60 minutes), the fol-
lowing RUN command would be entered:

    :RUN MEMTIMER;PARM=3600

Three error conditions are detected by MEMTIMER. If the PARM
parameter of the RUN command is equal to or less than zero, then
MEMTIMER will terminate after printing the following message:

    ** INVALID PARM (DELAY) VALUE **

The current time interval remains unchanged.

If MEMLOGP has been terminated, then MEMTIMER will terminate it-
self after printing:

    ** MEMORY LOGGING PROCESS NOT ACTIVE **

In this case there is no timing interval update.

If MEMLOGP is currently updating the memory log file, the follow-
ing message will appear:

    ** MEMLOGP TIMER ENTRY NOT FOUND **

In this case, the timing interval will be updated. MEMTIMER
should be run again to ensure that MEMLOGP will recognize the
updated interval immediately. Note that the default timing in-
terval will become the current timing interval each time the sys-

tem is brought up. Therefore, if a non-default timing interval
is desired, MEMTIMER must be run after each initialization of the
system.

6-40. Output. MEMLOGAN output will vary according to whether
the MEMLOG file is null or updated and, if updated, whether er-
rors have occurred. If the MEMLOG file is null (after running
MEMLOGAN with PARM=1) MEMLOGAN will terminate after displaying
the message:

    NO ENTRIES IN MEMLOG FILE

If the MEMLOG file is not empty, MEMLOGAN will print the date and
time of the first and last log updates. If errors have been
logged, the date and time of the first and last error logged will
also be printed. If no errors have been logged, MEMLOGAN will
terminate after displaying the message:

    LOGGING STARTED -DATE:
    LAST LOG UPDATE -DATE:
    NO ERRORS LOGGED

If errors have been logged, MEMLOGAN will continue by printing a
tabular interpretation of the information in the MEMLOG file.
The format of the printout is shown in figure 6-8. A typical
printout is shown in figure 6-9.

6-41. Errors. If an operational error is encountered by MEMLOGAN,
the program will print the appropriate error information and then
terminate. A non-file system error causes the following message
to be displayed:

    *MEMLOGAN ERROR:<ERRNUM>:

Where

<ERRNUM> 1= FLOCK ERROR ON MEMLOG FILE
         2= FUNLOCK ERROR ON MEMLOG FILE

The occurrence of a file system error will cause an error tomb-
stone to be displayed followed by either *OUT FILE ERROR* or,
LOG FILE ERROR.

6-42. Obtaining Memory Errors Copy. Use the following commands
to obtain a line printer copy of the log file:

    :HELLO FIELD.SUPPORT,HP32230
    :FILE OUT;DEV=LP
    :RUN MEMLOGAN.PUB.SYS

6-43. FLI PCA PROGRAMMING. The FLI PCA only accepts direct I/O
commands. These commands cause the FLI PCA to copy the contents
of the MCL PCA's Error Logging Array into the FLI PCA'S I/O Log-
ging Array (CIO-Read Copy), determine if an error log (1) was
transferred (TIO-Read Copy Error), search the I/O Logging Array

ADDRESS			ERROR TYPE			ERROR	
CONTROLLER	BOARD	ROW	TYPE	BIT	CHIP	COUNT	
controller	board	row	type	bit	chip	cnt	

where:

controller   The memory controller where the error occurred, shown as CONTROLLER A or CONTROLLER B.

board   The memory module board on which the error occurred, indicated by a digit from 0 through 3.

row   The row designation on the board in which the failing chip is located, indicated as a digit from 0 through 7.

type   Type of error detected, as follows:

CHECK	Check bit error.
DATA	Data bit error.
MULTIPLE BIT ERROR	Error in more than one bit.
FORCED D.E.W.	Forced Double Error Write. Indicated data parity error on the data transmitted to memory.
MISSING ARRAY BOARD	Non-responding array board.

bit   If *type* = CHECK, bit refers to the failing check bit — C0 through C5.

       If *type* = DATA, bit refers to the failing data bit — 0 through 15.

chip   Chip on which error occurred, in format:

      U*n*

    The variables n is a digit indicating the chip number.

cnt   The number of logging intervals during which this error was detected at least once.

NOTE

This value does not represent the number of times that an error was actually detected.

Figure 6-8. MEMLOGAN Table

I	ADDRESS				I	ERROR TYPE			I	ERROR	I
I CONTROLLER	I BOARD	I ROW	I	TYPE		BIT	CHIP		I	COUNT	I
I CONTROLLER A	I 0	I 1	I	CHECK		0	U198		I	2	I
I	I	I 6	I	DATA		9	U103		I	13	I
I	I 1	I 0	I	DATA		0	U19		I	4	I
I	I	I 0	I	MULTIPLE	BIT		ERROR		I	1	I
I	I	I 0	I	DATA		11	U149		I	3	I
I	I 2	I 7	I	DATA		14	U172		I	2	I
I	I 3	I 3	I	CHECK		5	U246		I	3	I
I CONTROLLER B	I 2	I 0	I	CHECK		2	U219		I	1	I
I	I	I 6	I	DATA		11	U143		I	3	I
I	I 3	I 2	I	DATA		15	U187		I	4	I

Figure 6-9. Typical MEMLOGAN Printout

for the presence of a "1" (CIO-Read Scan) and send the address of where the "1" was stored to the IOP Bus (RIO-Address). Direct commands clear the I/O Logging Array (WIO-Load Block) and then transfer these zeros into the Error Logging Array (WIO-Write Copy). The Error Logging Array would then be cleared and able to start a new fault logging sequence. Interrogating and writing into the Error Logging Array can only occur during refresh time. The following paragraphs explain in detail the control word formats.

6-44. TIO Command. Figure 6-10 shows the word format for a TIO instruction. This instruction can be executed any time.

Bit 0,1,   These bits have the standard I/O significance.
and 2

Bit 3   Read Scan. This bit is set when a Read Scan is in process and remains set until the completion or termination of the scan.

Bit 4   DISWEC. This bit is set if the Disable Error Correction flip-flop (DISWEC) is set.

Bit 5   L/NU 512K. If bit 5 = 1, then the upper 512K Error Logging Array has been selected. If bit 5 = 0, then the lower 512K Error Logging Array has been selected.

Bit 7   Read Copy Error. This bit is set if there are one or more errors (1's stored in Error Logging Array) during a Read Copy. This bit is cleared by any CIO or WIO Instruction.

TIO

```
        ┌─┬─┬─┬─┬─┬─┬─┬─┬─┬─┬──┬──┬──┬──┬──┬──┐
        │0│1│2│3│4│5│6│7│8│9│10│11│12│13│14│15│
        └─┴─┴─┴─┴─┴─┴─┴─┴─┴─┴──┴──┴──┴──┴──┴──┘
SIO OK ──┘ │ │ │ │ │ │ │
R/W OK ────┘ │ │ │ │ │ │
INT. PENDING ┘ │ │ │ │ │
READ SCAN ─────┘ │ │ │ │
DISWEC ──────────┘ │ │ │
L/Ū512K ───────────┘ │ │
=0 ──────────────────┘ │
READ COPY ERROR ───────┘
                                    ZEROS
```
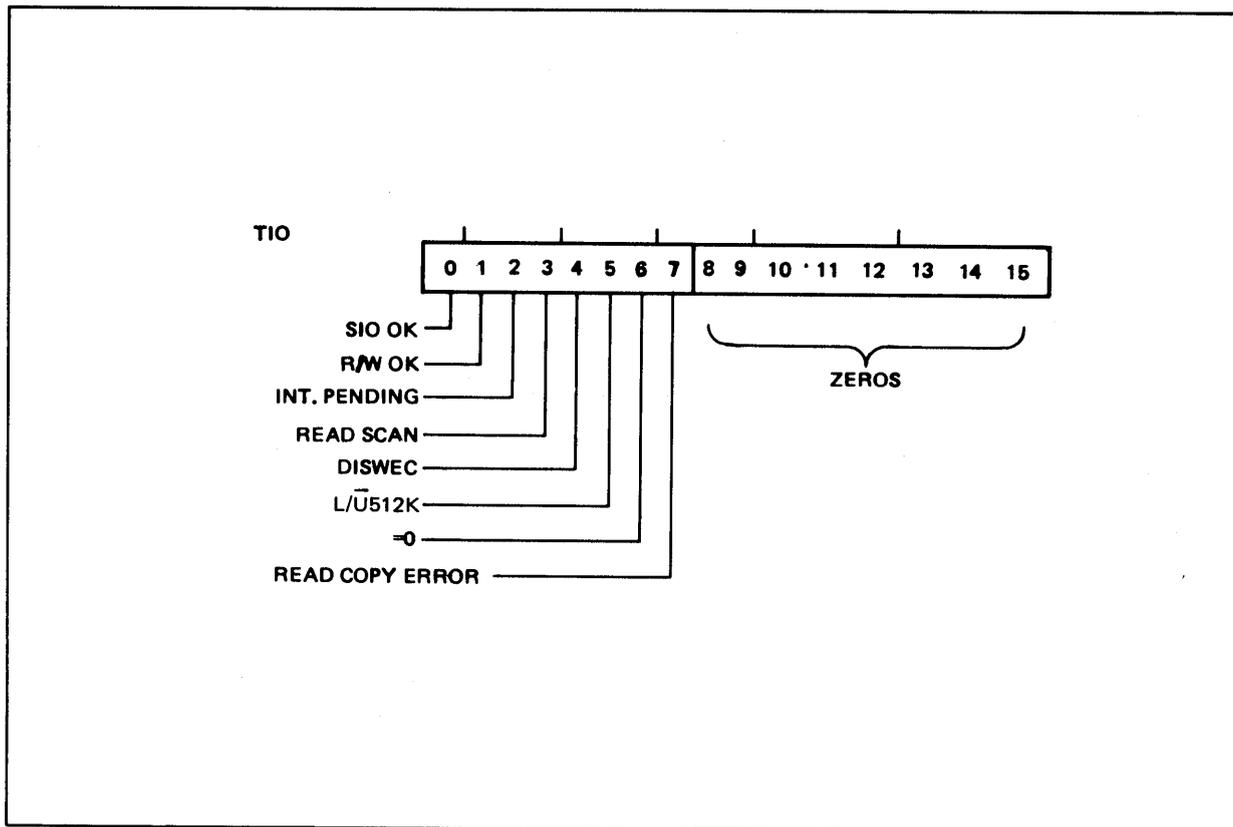
Figure 6-10. TIO Word Format

6-45. CIO Command. CIO instructions may be executed only if TIO bit 1 = 1 (R/W OK) except for a CIO master clear which can be executed at any time. Figure 6-11 shows the word format of the CIO instruction.

Bit 0   Master Clear. When set, causes the FLI PCA to reset and inhibits simultaneous Read Copy, Read Scan and/or DISWEC CIO instructions, or aborts any of these operations.

Bit 2   Read Copy. Causes the contents of the lower or upper 512K Error Logging Array to be transferred to the I/O Logging Array starting at the address location specified by bits 6 through 15 and ending at %1777. The Read Copy is completed when the Address Counter rolls over to %0000. During a transfer, TIO bit 1 = 0 (R/W not OK) and goes to a 1 within 50 usec after completion of a copy. A Read Copy will be terminated by the occurrence of a PFW, PON, I/O Reset or CIO Master Clear. Read Copy will inhibit a simultaneous Read Scan and is aborted by a Master Clear.
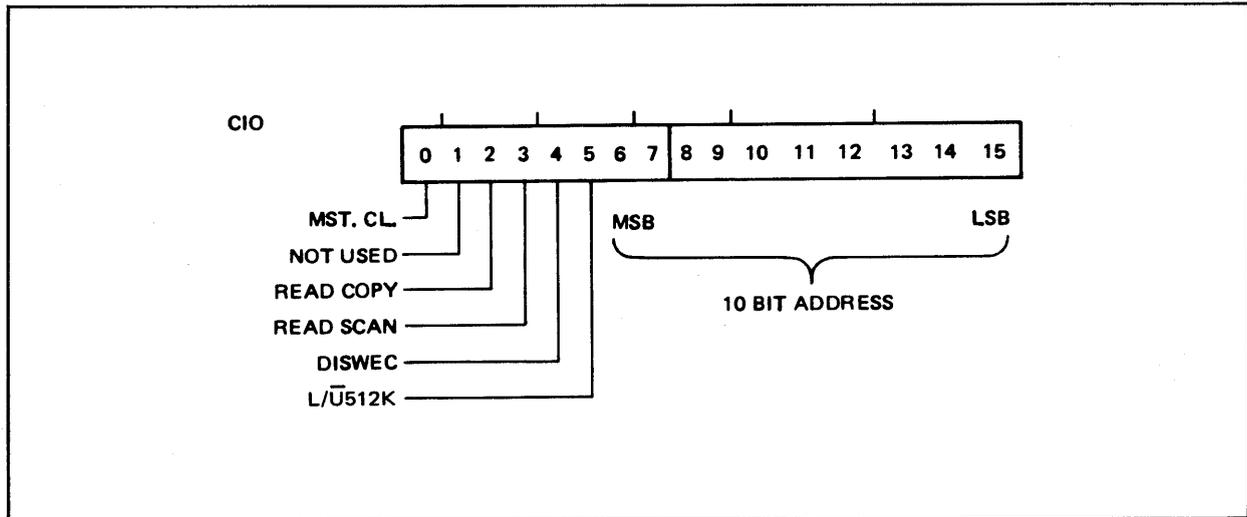
Figure 6-11. CIO Word Format

Bit 3   Read Scan. The I/O Logging Array contents are inter-rogated starting at the address specified by bits 6 through 15 and finishing at address %1777. The scan is completed when the Address Counter rolls over to %0000. The scan will halt at any I/O Logging Array location containing an error. An RIO instruction may be executed to retrieve the address of that error lo-cation. After the RIO completion, the scan will re-sume. During a scan, TIO bit 1 = 0 (R/W not OK) and goes to a 1 two usec after a halt on error or a scan completion. A read scan can be aborted by a Master Clear.

Bit 4   DISWEC. The Disable Write Error Correction flip-flop is set on the FLI PCA resulting in a memory operation with the error correction logic disabled. During a write operation, the data bits can be changed without modifying check bits C8 through C5. Note that error correction is disabled on all MCL PCA´s. Bit 4 can be cleared by I/O Reset, PFW, PON, completion of a Read Copy or Write Copy (WIO instruction), CIO Master Clear, or setting bit 4 to a 0.

Bit 5   L/NU 512K. If bit 5 = 1, then the upper 512K Error Logging Array is selected. If bit 5 = 0, then the lower 512K Error Logging Array is selected. The upper or lower 512K Error Logging Array is selected after one CPU clock cycle.

Bits 6-15   Address. The 10 bit address (1024 locations) is the starting address for a Read Copy of the Error Logging Array to the I/O Logging Array or a Read Scan of the I/O Logging Array. The address is loaded on all CIO instructions and is not affected by Master Clear.

6-25

6-46. WIO Command. WIO instructions may be executed only if TIO bit 1 = 1 (R/W OK). Figure 6-12 is the word format for a WIO instruction.
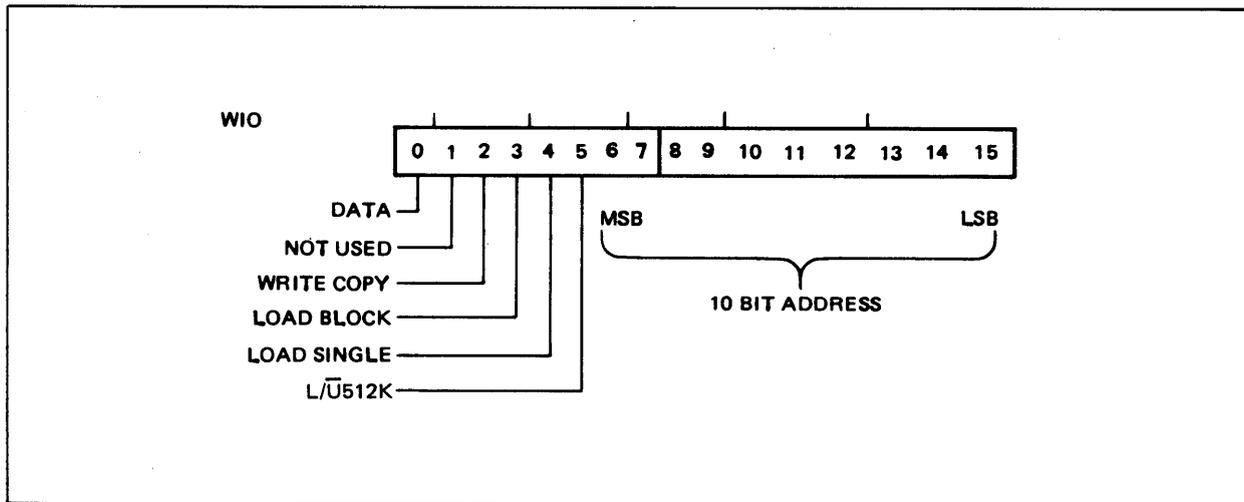
Figure 6-12. WIO Word Format

Bit 0   Data. This bit is used as the data in a WIO Load Block and Load Single instruction.

Bit 2   Write Copy. The contents of the I/O Logging Array are transferred into the upper 128K Error Logging Array or the lower 512K Error Logging Array starting at the address specified by bits 6 through 15, and ending at address %1777. At the completion of the transfer, the Address Counter is set to %0000. During a transfer, TIO bit 1 = 0 (R/W not OK) and goes to a 1 at 50 usec after a transfer. A CIO Master Clear will abort a Write Copy.

Bit 3   Load Block. The data in bit 0 (Data) is transferred to the I/O Logging Array in all locations starting at the location specified by bits 6 through 15 and ending at %1777. Upon completion of the transfer the Address Counter is set to %0000. During the transfer, TIO bit 1 = 0 and goes to a 1 two usec after a transfer. A Load Block will override simultaneous Write Copy and/or Load Single.

Bit 4   Load Single. The data in bit 0 (Data) is transferred to the I/O Logging Array at the address location specified by bits 6 through 15. The Address Counter is not changed after the completion of the transfer.

Bit 5   L/NU 512K. If bit 5 = 1, then the upper 512K Error Logging Array is selected and, if bit 5 = 0, then the lower 512K Error Logging Array is selected during a Write Copy execution. The upper or lower 512K Error Logging Array is selected after one CPU clock cycle.

Bits 6-15   Address. The 10 bit address (1024) locations is the
             starting address for a Write Copy of the I/O Logging
             Array to the Error Logging Array, a Load Block of the
             I/O Logging Array, or the address for a Load Single of
             the I/O Logging Array.

6-47. RIO Command.   Figure 6-13 shows the word format for the
RIO instruction.
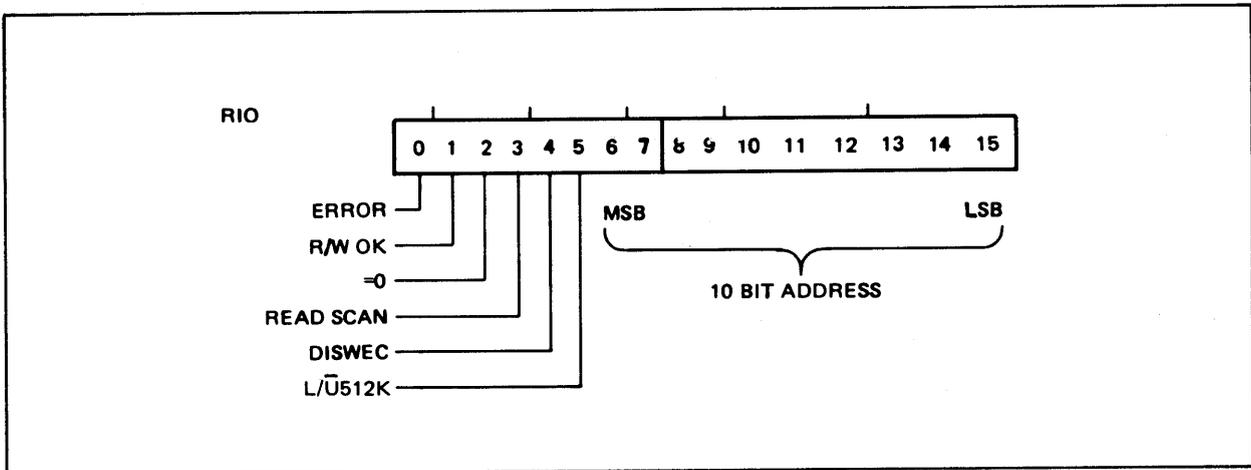
Figure 6-13. RIO Word Format

Bit 0   This bit is set if there was an error logged in the

             I/O Logging Array during a Read Scan and the address
             of that error is specified by bits 6 through 15.

Bit 1   R/W OK. This bit contains the same information as R/W
             OK on a TIO instruction.

Bit 3   Read Scan. This bit is set when a Read Scan is in
             process and remains set until the completion or ter-
             mination of a scan.

Bit 4   DISWEC. This bit is set if the Disable Error Correc-
             tion flip-flop is set.

Bit 5   L/NU 512K. If bit 5 = 1, the upper 512K Error Logging
             Array has been selected and, if bit 5 = 0, the lower
             512K Error Logging Array has been selected.

Bit 6-15   Address. This is the I/O Logging Array address of a
             logged error during a Read Scan.

6-48. SMA PCA SERVICING. Although the SMA PCA can be repaired to
the component level, repair procedures should be attempted only
by persons specificaly trained for such tasks. The replaceable
components are identified in figure 6-5. (For additional informa-
tion, refer to the HP 3000 Series III Computer System Engineering
Diagrams Set, part no. 30000-90173.) Each SMA PCA in the memory

module contains Switch S1 that must be set to reflect which des-
ignated PCA number it is in the memory module (e.g., first 128K
of memory, second 128K of memory, etc.). The SMA PCA's Switch S1
is identified in figure 6-14. As a convention only, S1 is set to
3 on the PCA installed closest to the MCL PCA, 2 on the next
closest PCA, 1 on the next PCA, and 0 on the SMA PCA installed
furthest from the MCL PCA.
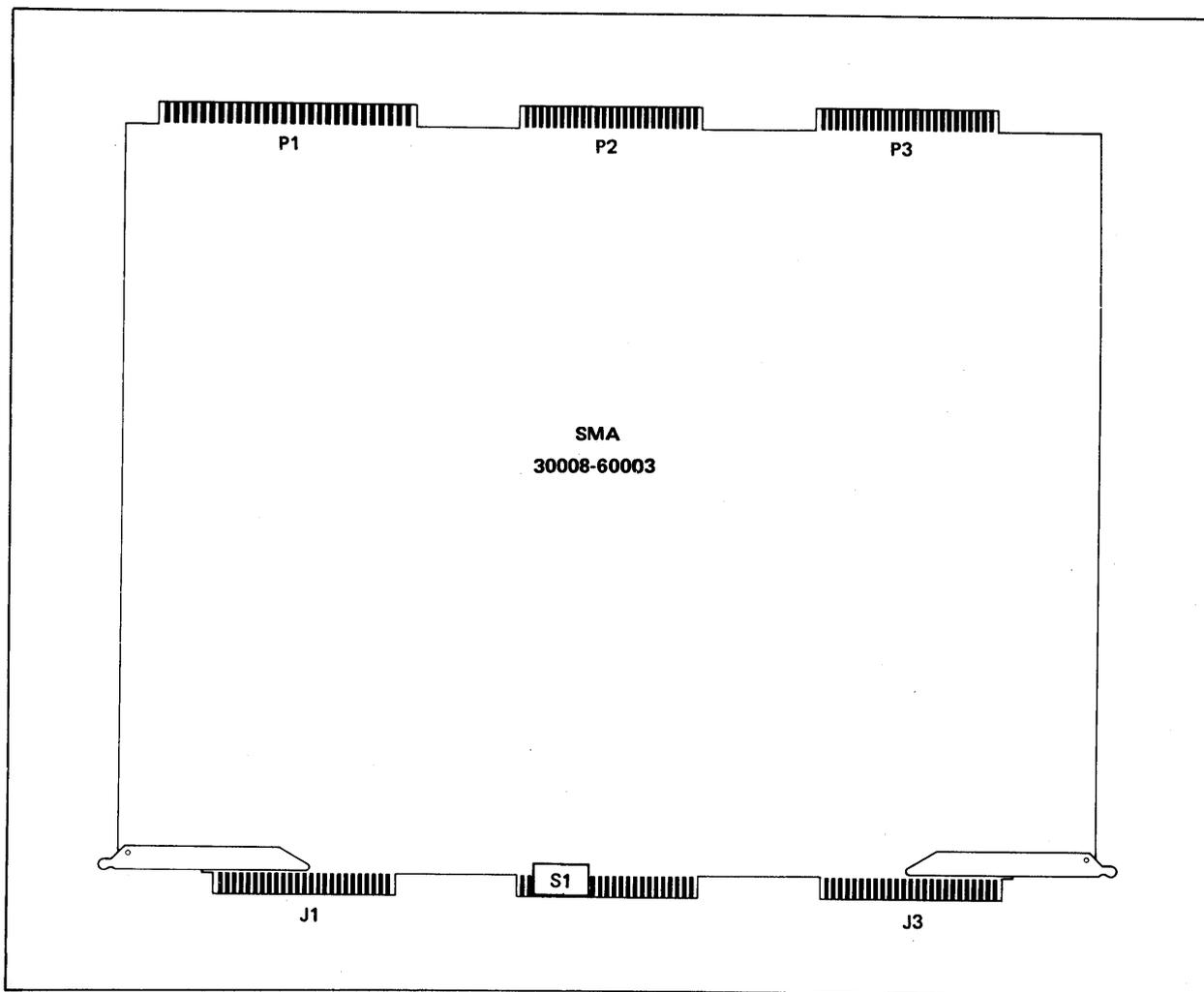
204

Figure 6-14. SMA PCA Switch Location

6-49. MCL PCA SERVICING. The MCL PCA is a nonrepairable PCA and
must be replaced if found defective. However, the MCL PCA con-
tains two switches, one of which (S2) must be set to reflect the
memory module configuration. Both switches are identified in
figure 6-15. In systems with over 512K words of memory, set
Switch S2 to position A on one MCL PCA and to position B on the
other MCL PCA. As a convention only, Switch S2 is set to posi-
tion A on the MCL PCA installed for lower memory (banks 0-7) and
and to position B on the MCL PCA installed for upper memory
(banks 8-15). Switch S1 is the ELA Manual Clear Switch and can
be used to clear the MCL PCA's Error Logging Array.
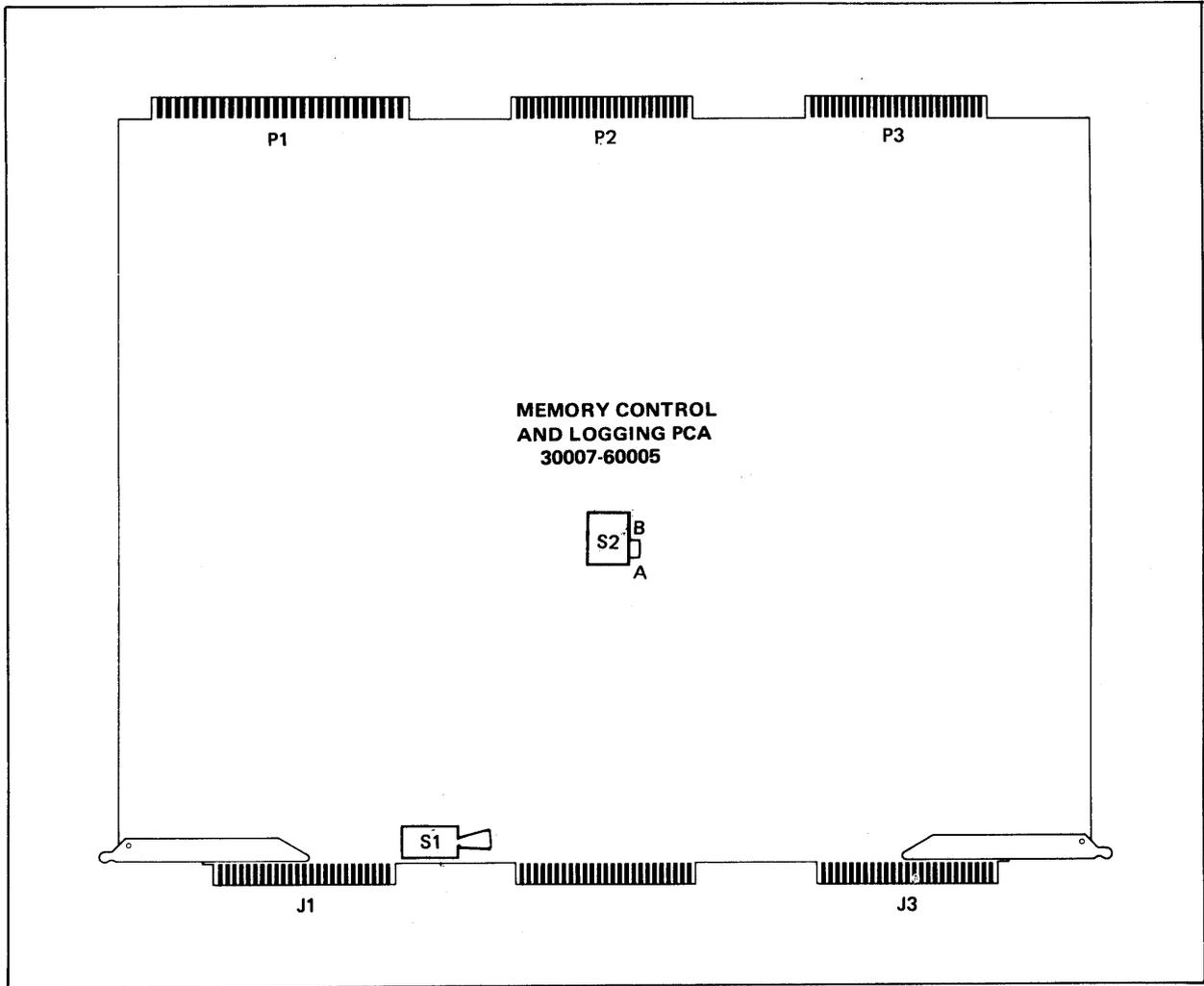
Figure 6-15.   MCL PCA Switch Locations

6-50. FLI PCA SERVICING. The FLI PCA´s are non-repairable PCA´s
(part no. 30009-60002 for HP 32421A Series III and part no.
30135-60063 for HP 32435A Series III) and must be replaced if
found defective.   Both PCA´s are preconfigured at the factory
with a DRT number of 2 and their jumpers should not be changed.
No repair or servicing procedures are required.

NOTES

This section contains an overview of the computer's I/O system which includes discussions on file system operation, I/O system operation, I/O instructions, and I/O system hardware. In addition, this section contains principles of operation and servicing information for the computer system's Input/Output Processor (IOP), Multiplexer Channel, Port Controller, and Selector Channel.

7-1. INTRODUCTION

The general purpose of any computer system is to input, process, and output information. Under MPE, this information may be created and used by the operating system itself, by compilers or other systems, by user programs, or by users themselves. To handle all information in a uniform, efficient way, MPE treats it as groups of data called files. Specifically, a file is a collection of information or data identified by a name recognized by MPE. MPE uses media such as discs, cards, and tapes for storing the information. On any of these media, a file may contain MPE commands, system or user programs, or data; alone or in any combination. Within a file, all information is organized into units of related data called logical records that for most applications are similar in form, purpose, and content. The records in the file can be arranged in almost any order; alphabetically, numerically, chronologically, by subject matter, etc. The logical record is the smallest grouping of data that MPE can address directly; you specify its length when you create the file. Individual subsystems and user programs, however, also can recognize fields for data items within each record. In addition, programs can also recognize and manipulate individual words, eight-bit bytes, and bits within a byte.

Data is transferred to and from files in units called blocks. These are the basic units that are physically transferred between Main Memory and the peripheral device on which the file resides. On disc and magnetic tape files, a block consists of one or more logical records; on files of other media, a block normally is equivalent to one logical record (unless you request input/output under the multi-record mode). To summarize the interrelation of files, logical records, and blocks: a file is a collection of records treated as a unit and recognized by a name; a logical record is a collection of fields treated as a unit, residing in a file; and a block is a group of one or more logical records transmitted to or from a file by an input/output operation. The purpose of the I/O system, then, is to perform actual physical input/output operations for the file system of the MPE operating system. The user normally does not interact directly with the I/O system; only indirectly via the file system as shown in figure 7-1. Normally, all I/O operations are invisible to the user. However, as shown in figure 7-1, privileged users may access the I/O system directly.
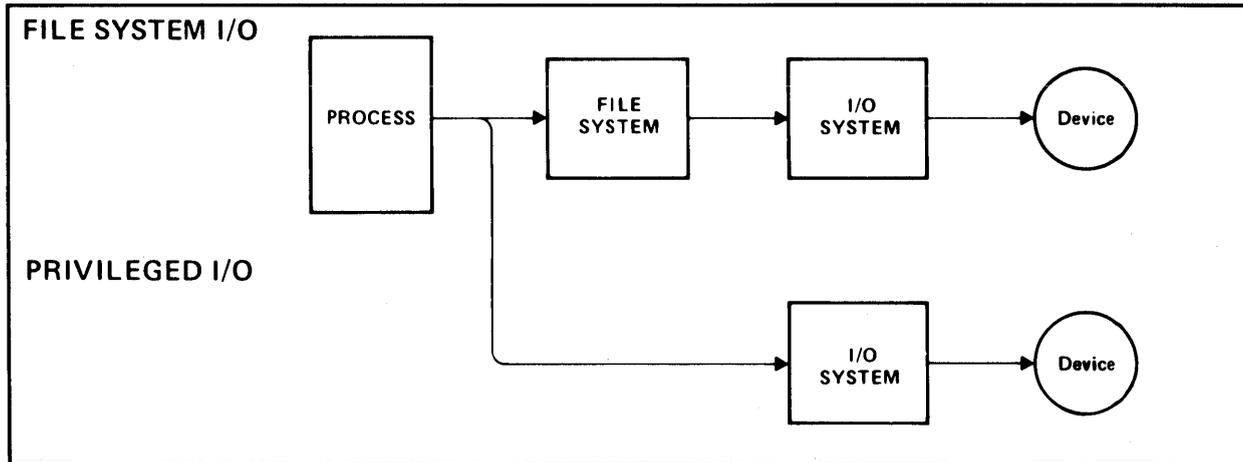
Figure 7-1. Basic I/O Access Methods

7-2. FILE SYSTEM OPERATION

Figure 7-2 illustrates the function of the I/O system in the overall handling of files. The I/O system, as shown, is part software and part hardware. Several peripheral devices are shown connected to the I/O system, each of which has some capability for handling files; entering files, storing files, or both. Of particular interest in this discussion are the files stored on disc. (Several physical disc units might be used.) Each disc file is broken up into one or more extents. (Disc extents are composed of a number of blocks.) When the file system causes the I/O system to transfer data to or from the disc, it does so one block at a time. As noted previously, the blocks are further subdivided into records and then into individual words. When the file system processes user file requests, it does so on the basis of records.

The memory management routine is also shown in figure 7-2 (dotted line) since it frequently makes its own requests to the I/O system. Memory management calls the I/O system in order to bring code and data segments into Main Memory where they can be accessed by user processes. In a typical operation, a user process might request the file system to read a file using the FREAD intrinsic (1). (Refer to the MPE Intrinsics Reference Manual for a discussion of the FREAD intrinsic.) The file system reads the stack associated with the user process (3). Note that in this example, no input/output has taken place. This is because the named record is already present in a buffer (BUFFER 0) in Main Memory.

Assume another case in which the requested record is not present. In this case, the file system makes a request to the I/O system (A) to read the block containing the particular record. The I/O system accordingly reads this block from the disc (B) and loads it into one of the buffers (BUFFER 1) allocated to the named file (C). (When you open a file, you specify how many buffers should
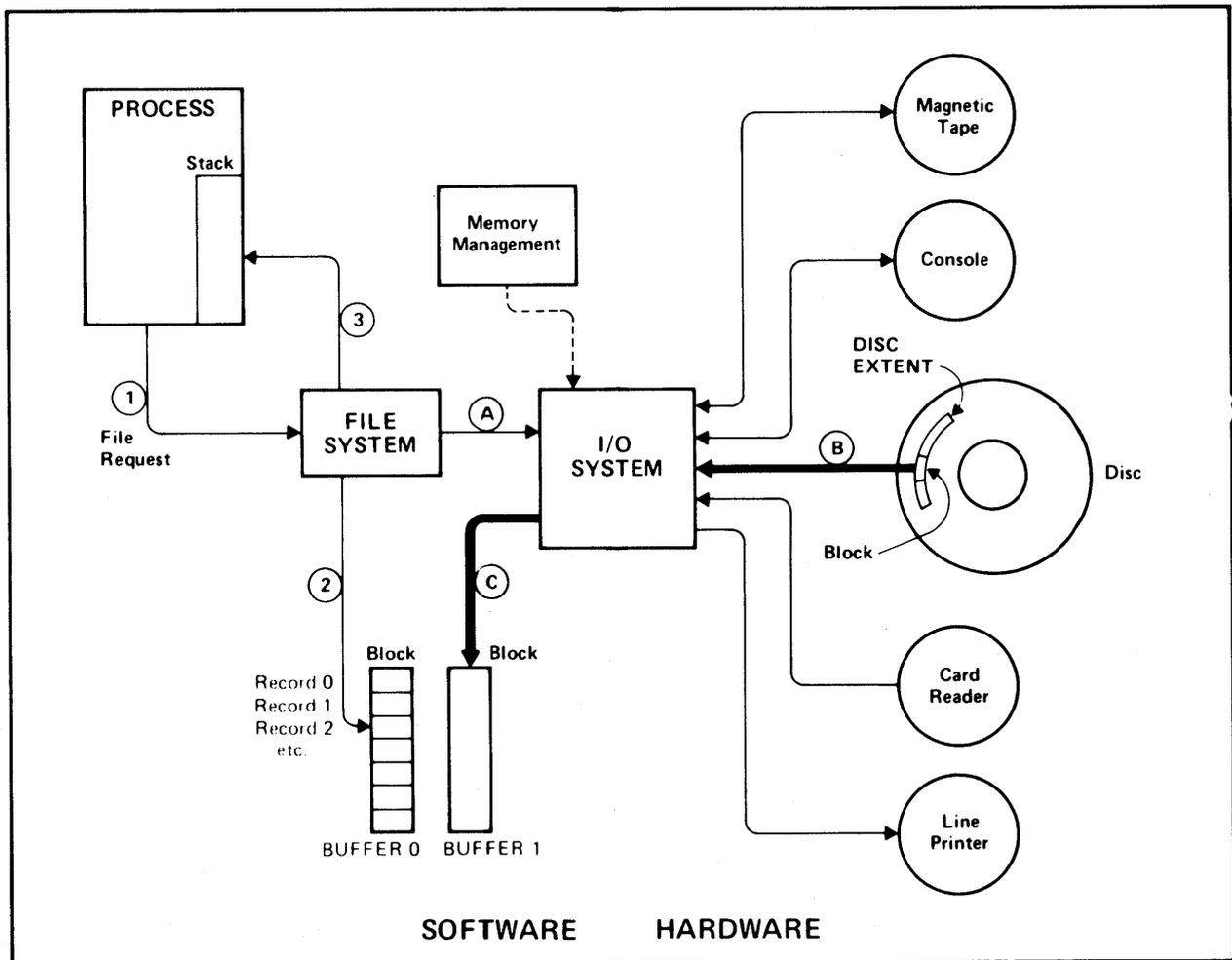
Figure 7-2. File System Basic Operation

be allocated for that file. However, you cannot access the buffers directly; only by naming records within files.) The file system can now complete the request by reading the requested record to the stack. Note that in none of the preceding operations did the user process specify a device. An actual I/O operation may or may not have occurred and the user is completely unaware of such an occurrence. The operating system, however, allows devices to be specified either by class name or by a specific logical device number. This permits, for example, inputting or outputting files via a specific terminal, card reader, or line printer.

7-3. DEFINITION OF TERMS

As shown in figure 7-3, a Device Controller in the I/O system is the hardware I/O linkage between the CPU and I/O device. It typically consists of one or more logic cards. Depending on particular controllers, the Device Controller may drive only one peripheral (such as a card reader) or may be capable of driving several peripherals (such as disc units). Figure 7-3 illustrates
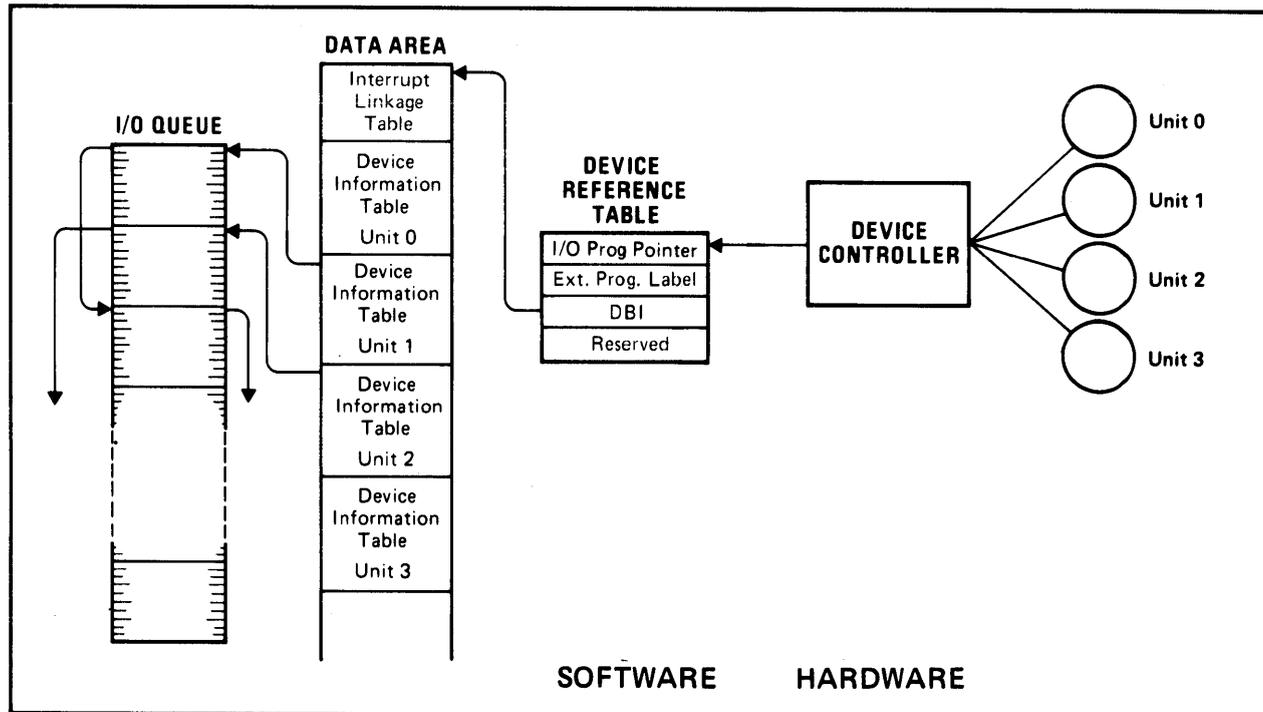
Figure 7-3.   I/O System Fundamental Elements

some of the important elements of the I/O system. (This figure is
by no means complete, but rather is intended to define the chain
of linkages that are basic to the I/O system.)   For each Device
Controller there is a four-word entry in the Device Reference
Table (DRT).   The third word in the four-word table entry con-
tains a pointer to a data area uniquely associated with that
table entry.   The data area consists of an Interrupt Linkage
Table (ILT), one or more Device Information Tables (DIT) (depend-
ing on how many units the Device Controller is driving), and an
I/O program area.   Along with various other information, the
Driver Linkage Table (DLT) contains Code Segment Table (CST) and
Segment Transfer Table (STT) values for defining the location of
the driver routines associated with that particular Device Con-
troller.   The DIT contains information relevant to one physical
I/O device and is configured differently for each type of device.
In each case, however, the third word of this table points to an
entry in the I/O Queue (IOQ) when a request is being made.

The IOQ is a single table (only one per system) containing a
fixed number of entries having a fixed number of words per entry.
If there are no I/O requests pending in the system, none of the
DIT entries will be pointing to the IOQ.   In this case, all en-
tries of the IOQ are unused, and the second word of each entry
points to the first word of the next entry. Thus, all unused
entries are linked together. Assume that the file system makes a
request to use Unit 1 of the Device Controller shown in figure
7-3.   The I/O system will unlink the first free entry in the IOQ
and fill it with information pertaining to the request (including
buffer address and logical device number).   Assume that the next

request is for Unit 2 (uses the next available entry), followed by a second request for Unit 1. This second request for Unit 1 causes the first word of the initial request to point to the next unused entry, which is then filled with information pertaining to the second request. Therefore, eventually the IOQ will contain a queue of requests for Unit 1, a separate queue for Unit 2, and so on, plus a linked list of free entries.

Next, an I/O driver is executed to initiate the request. An I/O program will then be run on a device, using the request parameters given in the IOQ. When the request is completed, the IOQ entry is returned to the free list. Note that the IOQ only establishes the priority of requests for each device on a first-in first-out basis. Questions of priority in executing I/O drivers are resolved by the Dispatcher. (Refer to the MPE General Information Manual for a description of the Dispatcher.) Once several Device Controllers are running I/O programs, priority conflicts are resolved by hardware service priority.

The DRT (figure 7-4) consists of a number of four-word entries corresponding to the number of Device Controllers present in the system. The DRT is located in fixed memory locations beginning at octal address 14. (Locations 0 through 13 are allocated to other purposes; refer to table 2-4.) The upper limit for the table is location 777 which limits the maximum number of four-word entries to 125 (decimal). Because each DRT entry is always four words in length, it is convenient for the hardware to map device numbers to DRT addresses simply by multiplying by four. (Leftshift device number two binary places.) Thus the entry for device number 3 begins at octal location 14 (i.e., %(3x4) = %14). Because the DRT begins at location 14, device number 3 is the lowest device number. (Devices 0, 1, and 2 do not exist.)

Note

The device number associated with a particular DRT entry defines a Device Controller, and not necessarily an actual physical device. Also remember that some controllers identified by one device number are capable of driving several physical devices. Individual identification of physical devices is made by logical device numbers. The logical device number is the value used by the file system in requesting I/O, and the I/O system software performs the logical to physical device number translation.

FORMAT

| I/O Program Pointer |
| Ext. Prog. Label |
| DBI Address |
| (Reserved) |
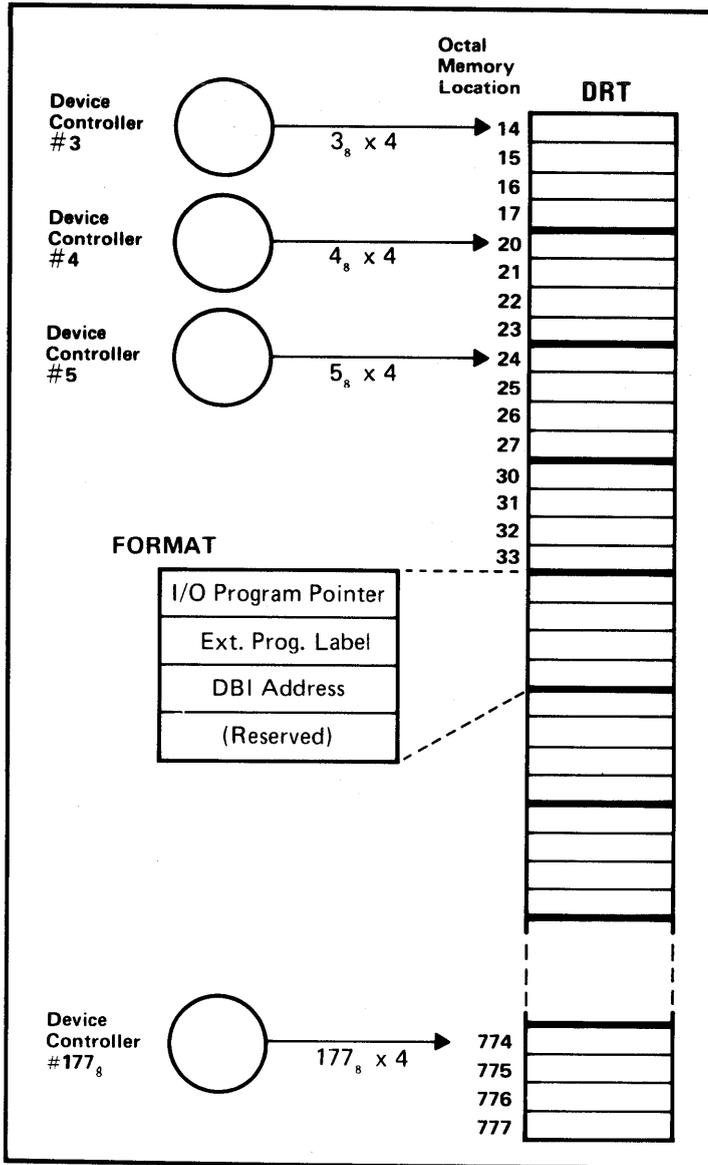
Figure 7-4.   Device Reference Table

7-4. I/O INSTRUCTIONS

There are ten I/O instructions in the system's instruction set, six of which are defined in paragraph 4-14 of this manual. (All of the I/O instructions are fully defined in the HP 3000 Series II/III Computer System Machine Instruction Set, part no. 30000-90022.) The distinction to note here is that the SIO instruction is used in conjunction with an I/O program and that the other instructions are not.   That is, an SIO instruction commands a Device Controller to begin executing its associated I/O program which effects a block transfer of data between an I/O device and memory.   This is termed an SIO transfer mode. The other I/O instructions transfer only one word per instruction between the device and TOS in the CPU. This is termed a direct transfer mode and is used primarily with terminal devices.

7-5. GENERAL I/O OPERATION

An overview of the I/O system's operations for I/O transfers is illustrated in figure 7-5. (It should be noted that figure 7-5 does not apply to direct I/O devices.) To provide a complete sequence of operations, it is assumed that the file request results in a need for physical I/O to be performed. (As previously discussed in paragraph 7-2, this is not always the results.) The sequence of operations is as follows:

a. An executing user process generates a file request (1, figure 7-5) to the file system.

b. The file system tests the validity of the request and calls the Attach I/O (ATTACHIO) procedure (2). This is the entry point to the I/O system.

c. Attach I/O inserts the request parameters (3) in the I/O Queue for the requested device.

d. When all earlier requests for the device have been completed (4), the I/O Monitor Procedure begins execution for this request.

e. The I/O Monitor ensures that the data buffer for the file is present in memory. It then issues a procedure call (PCAL) to the initiator section (5) of the device driver, passing the request parameters to that routine.

f. The initiator section assembles the I/O program (using the request parameters), issues an SIO instruction to the Device Controller, and exits back to the I/O Monitor. The SIO instruction initializes the DRT to point at the starting location (6) of the I/O program.

g. The I/O program issues commands (7) to the Multiplexer or Selector Channel.

h. The Multiplexer Channel or Selector Channel enables (8) the Device Controller.

i. The Device Controller, on receiving a read or write command from the I/O program, transfers a block of data (9) to or from the data buffer. The length of the block is specified by the I/O command.

j. On completion of the data transfer, the I/O program commands the Device Controller to request an interrupt (10) via the Multiplexer or Selector Channel. The I/O program then ends.

k. The Device Controller causes a CPU interrupt to an interrupt routine (11) which tells the I/O Monitor that an interrupt has occurred.

Figure 7-5. I/O System Overview

Note

There are several interrupt routines for
external interrupts. For example, one is
the General Interrupt Processor (GIP) for
all types of devices except terminals, and
another is the Terminal Interrupt Proces-
sor (TIP) for terminals.

l.   The interrupt routine (or the last routine to use the Inter-
     rupt Control Stack) exits (12) to the Interrupted User Pro-
     cess. (Refer to Section VIII.) It also may activate the
     related I/O process if necessary.

m.   When the I/O Monitor Procedure is executed again, it recog-
     nizes that an interrupt has occurred and accordingly calls
     the completion section (13) of the device driver.

n.   The completion section checks the results of the transfer.
     If necessary, it may initiate additional transfers by telling
     the I/O Monitor Process to call the initiator section again.
     Otherwise, it updates the I/O Queue with information regard-
     ing results of the original request (14). The file system
     may then check these results (15).

o.   When the user process is dispatched again, a return is made
     to a point following the file request (16), depending on
     whether blocked or unblocked I/O was specified. (Refer to
     paragraph 7-9.)

7-6. DIRECT I/O OPERATION

The operations for direct I/O transfer mode involve considerably
more software overhead than the operations for the SIO transfer
mode. This is due to the varied nature of the terminal devices
that use direct I/O. The following sequence descriptions present
only a broad generalization of direct I/O terminal operations.
The sequences given should not be construed as representing any
particular device or even a typical device. It is assumed that
the log-on sequence has been accomplished.

Figures 7-6 and 7-7 illustrate the handling of data via direct
I/O terminal devices. Figure 7-6 illustrates input (read) oper-
ations and figure 7-7 illustrates output (write) operations.

In comparison with figure 7-5, note that there is no I/O program
in the data area. Instead, the interrupt routine performs the
functions of an I/O program. The interrupt routine, in this
case, is part of the device driver. It should also be noted that
direct write uses no completion section, and that no Multiplexer
Channel or Selector Channel is involved.
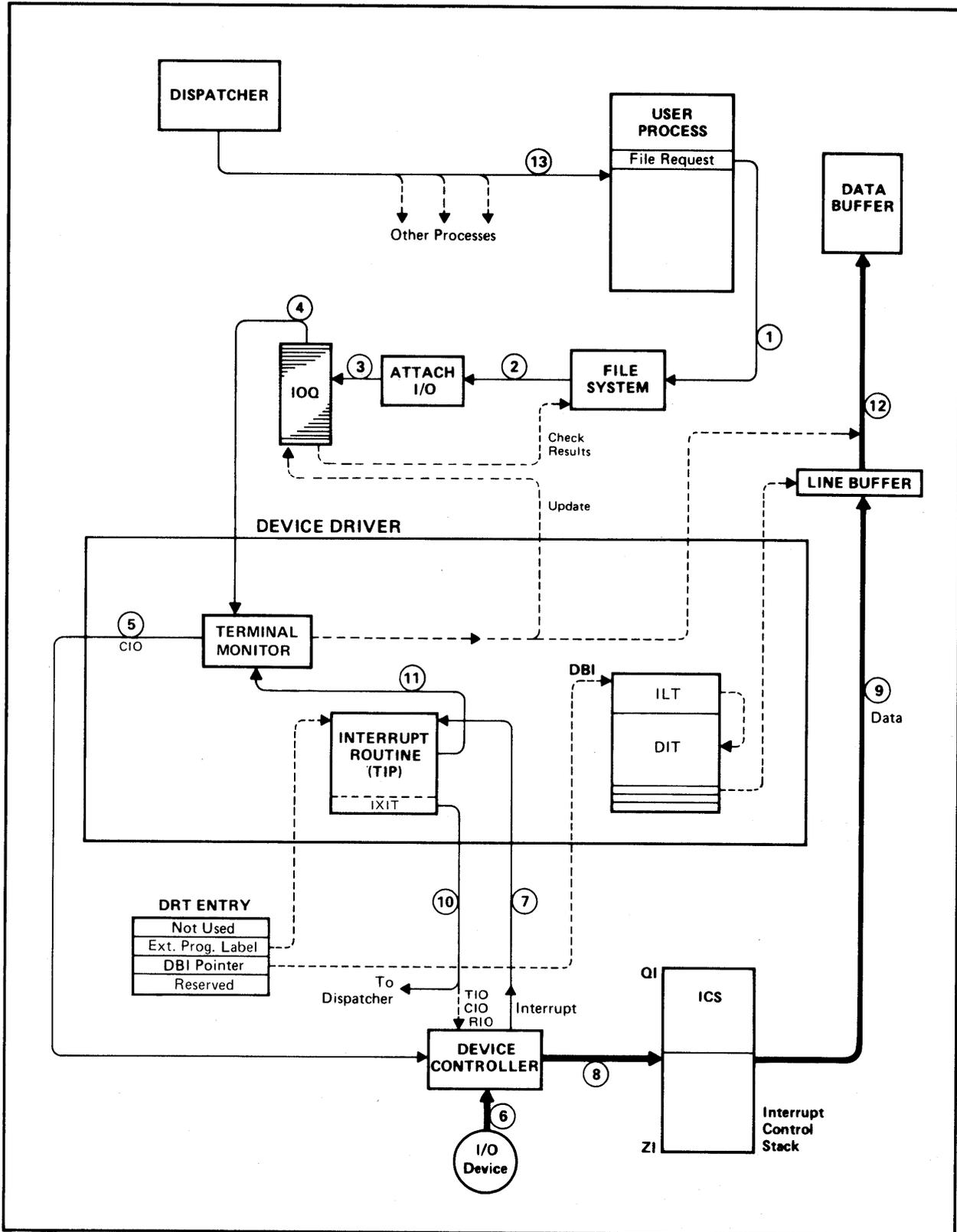
Figure 7-6.   Direct Read For Terminal Devices
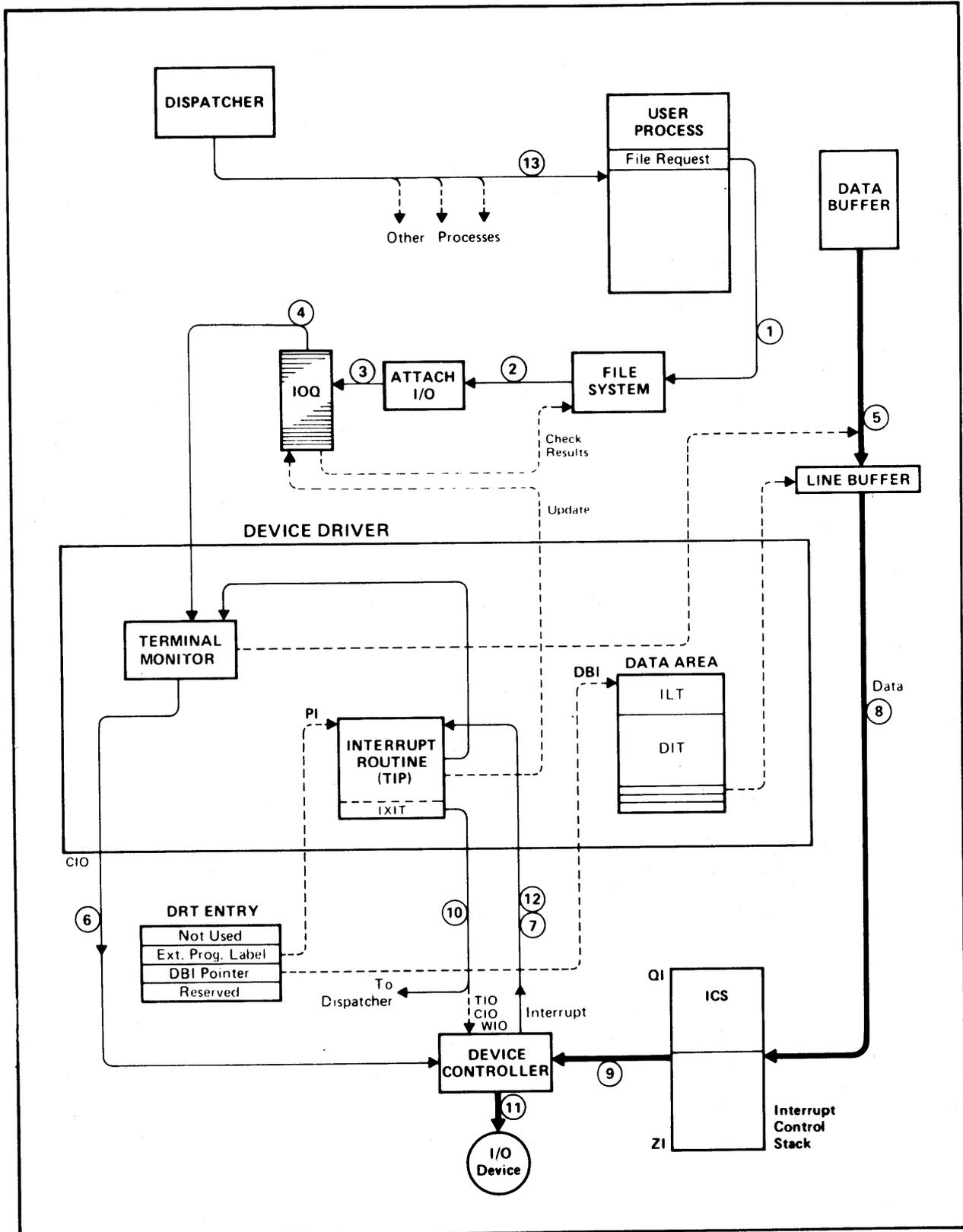
Figure 7-7.   Direct Write For Terminal Devices

One element not previously present is the terminal buffer. The terminal buffer consists of a linked list of buffers, which are pointed to by an address word in the Device Information Table (DIT) for a particular terminal. A sufficient number of these buffers are used to accommodate the line or record length of the associated device. Data is transferred between the terminal buffer and the data buffer on a record basis. Thus, the terminal buffer reads characters from the device until the complete line (or record) is read, then transfers the complete line (or record) to the data buffer. This scheme conserves Main Memory space by allowing the data buffer to be absent on disc while the comparatively slow terminal device is transferring individual characters.

7-7. Direct Read

The sequence of operations for direct read, illustrated in figure 7-6, is as follows. (It is assumed that the file request requires a physical read from the terminal.)

a. The executing user process generates a file request (1, figure 7-6) to the file system.

b. The file system tests the validity of the request and calls the Attach I/O procedure (2).

c. Attach I/O inserts the request parameters (3) in the I/O Queue for the requested device. Unlike SIO which uses a first-in/first-out queue for the requests, terminal requests are analyzed for relative importance and are then inserted into an appropriate place in the queue. (For example, factors such as whether the request is from the system console are considered.)

d. When all higher priority requests have been completed, the TERM procedure (4) begins execution for this request.

e. The Terminal Monitor issues a CIO instruction (5) directly to the Device Controller, causing TIP to initiate the read operation.

f. The Device Controller enables the device to transmit a character (6). When a key is pressed, the device returns the the character to the controller.

g. On receipt of the character, the Device Controller causes a CPU interrupt (7) to the interrupt routine for terminals (TIP).

h. TIP issues an RIO instruction to the Device Controller. This causes the character (8) to be loaded onto the ICS and also causes a command to be issued to the device to transmit the next character. TIP now checks the character on the ICS to determine if it is a data character or a control character.

i. If the character on the ICS is a data character, it is trans-
    ferred (9) by TIP to the terminal buffer. If it is a control
    character, TIP performs the appropriate control function.

j. TIP exits (10) to the Dispatcher and the sequence repeats
    back to step g until the entire record has been read.

k. When a CR (Carriage Return) control character is detected,
    TIP sets a flag in the DIT to signify that the record is
    complete, then causes the Terminal Monitor to be executed
    (11).

l. The Terminal Monitor transfers the content of the terminal
    buffer to the user stack (12) and the transmission log in the
    IOQ is updated.

m. The Terminal Monitor releases the line buffer and control is
    returned to the user process (13). To read another record,
    the file system must make another I/O request to Attach I/O.

7-8. Direct Write

The sequence of operations for direct write is illustrated in
figure 7-7. The sequence of operations is as follows:

a. The executing user process generates a file request (1, fig-
    ure 7-7) in the file system.

b. The file system tests the validity of the request and calls
    the Attach I/O intrinsic (2).

c. Attach I/O inserts the request parameters (3) in the I/O
    Queue for the requested device.

d. When all higher priority requests have been completed, the
    Terminal Monitor begins execution for this request (4).

e. The Terminal Monitor transfers the data (5) from the data
    buffer to the line buffer.

f. A CIO instruction (6) is issued to the Device Controller to
    initiate the write operation.

g. The Device Conroller causes the CPU to interrupt to TIP (7).

h. TIP transfers a character (8) to the ICS.

i. TIP executes a WIO instruction, transferring the character
    (9) from the ICS to the Device Controller.

j. TIP then exits to the Dispatcher (10), and hardware takes
    control.

k. The Device Controller writes the character (11) out to the
    device.

1. On completion of the write, the Device Controller generates another interrupt (12) to TIP. The sequence repeats back to step h until all characters in the record have been written out to the terminal.

m. The Dispatcher then returns control (13) to the user process.

7-9. BLOCKED/UNBLOCKED I/O

At the conclusion of all three of the preceding operating sequences (general I/O, direct read, and direct write), control is returned to the user process. While the I/O operation was in progress, the user process may have been suspended to await I/O completion (blocked I/O), or may have continued to execute while periodically checking for I/O completion (unblocked I/O). The choice of blocked or unblocked I/O is made in the call to ATTACHIO. (The file system nearly always uses unblocked I/O.) Paragraphs 7-10 and 7-11 contain a description of the characteristics of blocked and unblocked I/O.

7-10. Blocked I/O

As shown in figure 7-8, the user process goes into an I/O wait state as soon as the I/O request is given. The user process remains in the wait state while the I/O operations proceed. The request is entered into the I/O Queue and is ultimately processed via the hardware I/O system. At the end of the I/O operation, the results of the transfer are entered into the IOQ. Control is then returned to the user process. During terminal writes, the operation is considered completed when the data has been transferred to the terminal buffers. The user process now continues to execute from the point following the I/O request.

7-11. Unblocked I/O

During unblocked I/O (figure 7-8), privileged capability is assumed. The process must also specify the action to be taken on completion of I/O; either no action or reactivate the process if in an I/O wait state. This specification (like the blocked/unblocked I/O choice) is made in the call to ATTACHIO. After calling ATTACHIO, the process may then continue to execute and may generate other unblocked I/O requests. It is the responsibility of the process to synchronize all unblocked requests and to check for I/O completion. The process also has the capability to put itself into the I/O wait state and to change the I/O completion action for any unblocked request at any time. Obviously, however, the process should not specify no action for all unblocked requests and then go into the I/O wait state since there is no way to recover from this situation. At least one request must reactivate the process.

While the process continues to execute, ATTACHIO enters the request into the I/O Queue and hardware processing of the request begins. At the end of the I/O operation, the results of the transfer are entered into the IOQ. Then the completion action
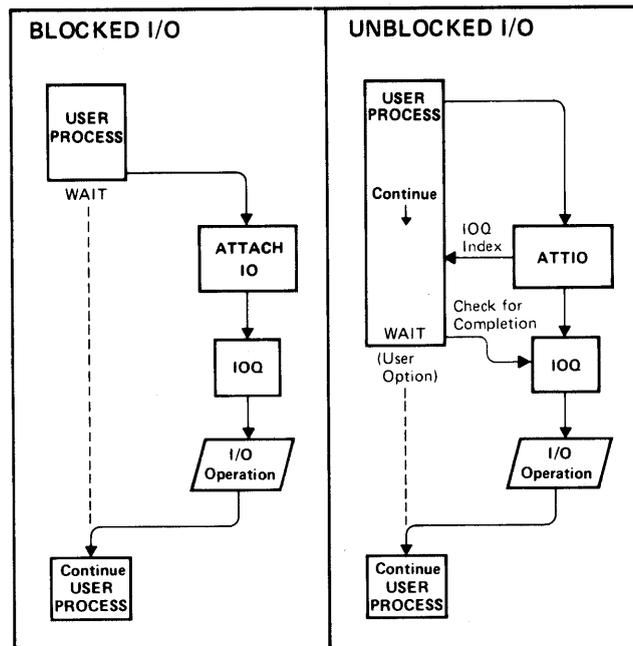
Figure 7-8. Blocked and Unblocked I/O

bit is examined. If the awaken process is specified, the process will be reactivated if it has put itself into the I/O wait state. If no action is specified, presumably the process has continued to execute without any wait, or will be reactivted by some other process. In any case, the process checks for I/O completion.

7-12. I/O HARDWARE ELEMENTS

The I/O hardware elements are responsible for a large portion of the execution of an I/O request. When software passes control to hardware, the I/O hardware elements assume full control from that point while the software performs other functions. The computer's I/O hardware elements are the I/O Processor (IOP), part of the Module Control Unit, the Multiplexer Channel, and Selector Channel. The following paragraphs contain detailed discussions for each of these elements.

7-13. I/O Processor

In addition to interrupting the CPU on behalf of the Device Controllers (Section VIII), the IOP performs three specific functions relating to the three different I/O transfer modes shown in figure 7-9. For direct I/O transfers, the IOP executes the direct I/O instructions (RIO, WIO, TIO, CIO, SIN, and SMSK) and transfers data, device status, and control information between the CPU and a Device Controller. For programmed I/O transfers via a Multiplexer Channel, the IOP transfers I/O program words between Main Memory and the Multiplexer Channel and transfers data between Main Memory and the Device Controller. For programmed I/O transfers via a Selector Channel, the IOP only transfers initialization information to the Device Controller and is
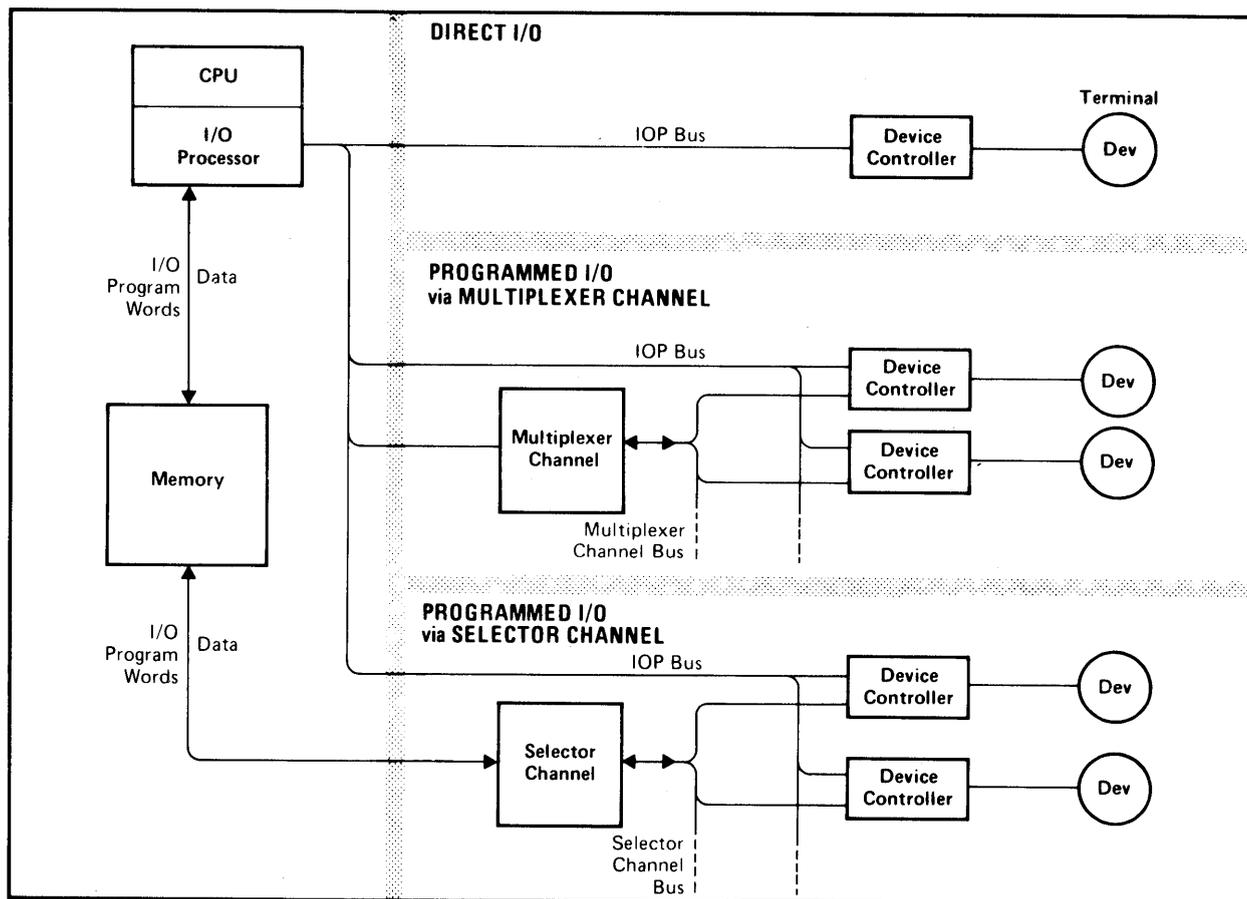
Figure 7-9. I/O Hardware Elements

not involved in any part of the I/O program execution. A simpli-
fied logic diagram of the IOP is shown in figure 7-10 and discus-
sed in paragraphs 7-14 through 7-19. For additional information
concerning IOP operations in conjunction with the Module Control
Unit (MCU), Multiplexer Channel, Selector Channel, and system in-
terrupt system, refer to paragraphs 7-20, 7-33, 7-42, and Section
VIII, respectively.

7-14. I/O COMMAND. The I/O instruction information is combined
by the CPU into a single word, placed on the U-Bus, and sent
through the IOA logic to the IOP Bus. (See figure 7-10.) The in-
struction from the code segment has been translated into a three-
bit command (IOCMD). The command can now be read out onto the
IOCMD lines of the IOP Bus. The device number has been obtained
from the stack, and can now be read out on the device numbers
(DEVNO) lines of the IOP Bus. The Service Out (SO) bit tells the
addressed device, via the IOP control, to accept and respond to
the accompanying information. (The Device Controller must return
a Service In (SI) handshake signal.)
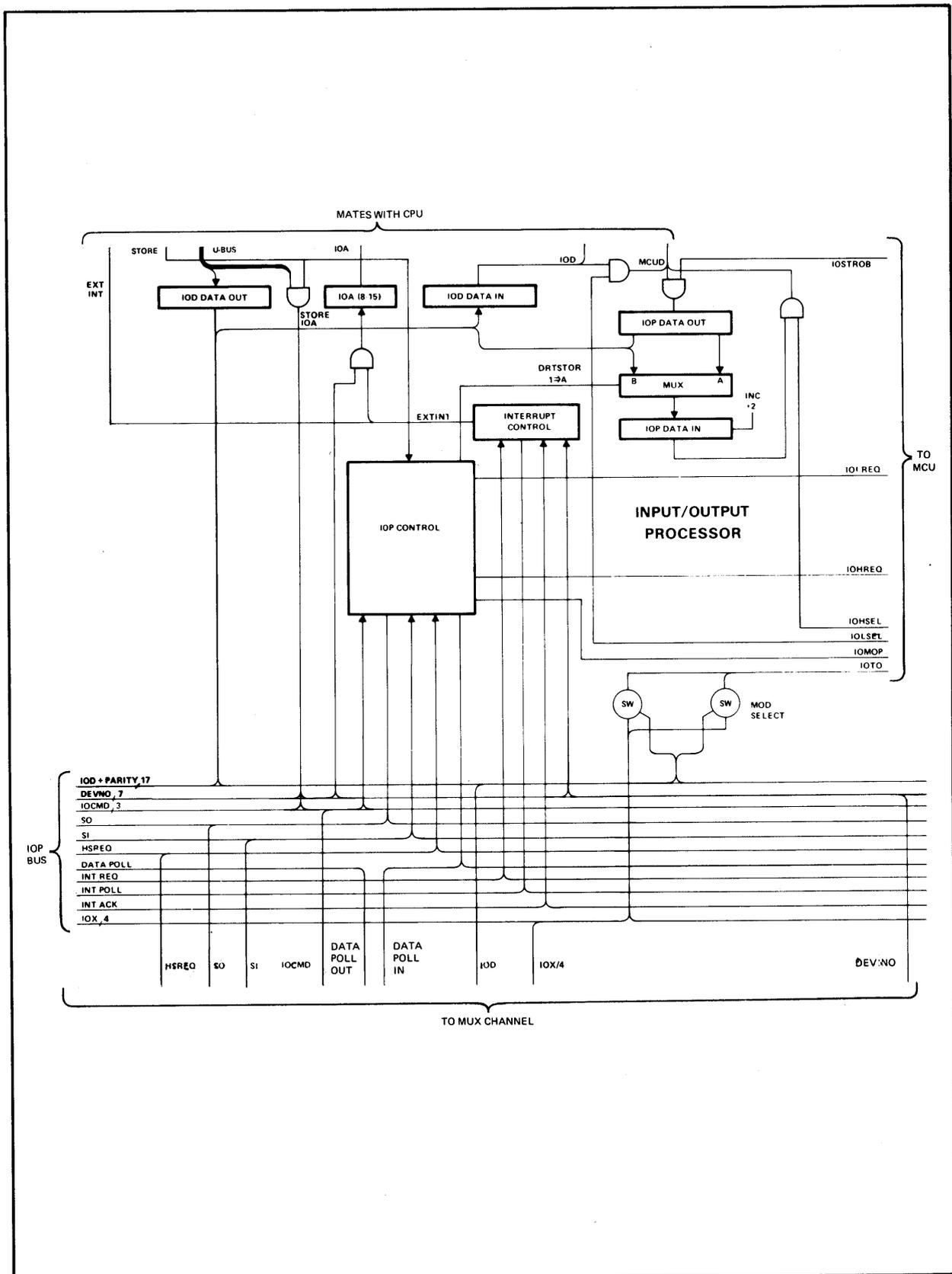
Figure 7-10. IOP Simplified Logic Diagram

7-15. IOP CONTROL. The IOP control block represents sequencing logic for transfers between the device and memory, and between the device and the CPU. Each of the lines shown entering or leaving this block are discussed with their associated transfer sequences.

7-16. INTERRUPT CONTROL. The interrupt control logic accepts an Interrupt Request (INTREQ) from the Device Controllers on the IOP Bus, interrogates the Device Controllers with INTPOLL to find the highest-priority request, and, when Interrupt Acknowledge (INTACK) is received, loads the device address into the IOA Register.An External Interrupt (EXTINT) signal is issued to the CPU.

7-17. INT DEVNO. The I/O Address (IOA) Register holds the device number of the interrupting device so that, upon command, the CPU can read the contents onto the S-Bus for interrupt processing.

7-18. DATE OUTPUT REGISTERS. There are two data output registers, the IOP Data Out Register for memory data received from the CTL Bus, and the IOD Data Out Register for direct data received via the U-Bus from the CPU. Signals from IOP control can either read the contents out onto the IOP Bus or transfer the contents into MUX for restoring a DRT entry.

7-19. DATA INPUT REGISTERS. There are two input registers. The IOP Data In Register is used for sending data to memory via the CTL Bus. This register is loaded from either the IOP Bus or, for DRT entry restoring, from the IOP Data Out Register. When doing a DRT store, the IOP Data In Register is incremented by two before the transfer is made. The second input register, IOD Data In, may be used either as a direct data input register or as a memory address register. It is loaded from the IOP Bus. During direct I/O execution, the register contents are read onto the CPU S-Bus. When addressing memory, the register contents are read out to the CTL Bus.

7-20. Module Control Unit

As previously discussed in Sections II and VI, the Module Control Unit (figure 6-1) contains MCUs for both the CPU and the IOP. The MCUs operate basically in parallel, but not independently. Since both MCUs share the same access to the CTL Bus, and also share the same module number, it is necessary to resolve priority when both the IOP and CPU simultaneously attempt to use the bus. Priority is resolved so that IOP requests take precedence over CPU requests except that a CPU high request takes precedence over an IOP low request. This exception means that the CPU is in the middle of a memory write operation, having sent an address to memory, and the high request is an attempt to follow up by sending the data. CPU low request represents the beginning of a transfer (attempt to send an address) and any IOP request will have priority over the CPU low request.

An IOP Request (IOP REQ) signal (figure 6-1) is generated when either a low request (IOLREQ) signal or a high request (IOHREQ) is about to set one of the select flip-flops (IOLSEL or IOHSEL). The IOP REQ signal inhibits the setting of the CPU Select flip-flop. However, a CPU HREQ signal will inhibit IOLRQ from generating the IOP signal. When data is returned from memory the FROM comparator compares the data with the contents of the TO Register to check that the transmission is from the same memory module to which the address was sent. The TO Comparator also checks that the transmission is to this module. Together, the outputs of the two comparators generate an I/O Strobe (IOSTRB) signal which locks the IOP Data Out Register (see figure 7-10), because it now contains the correct information from the CTL Bus. The IOSTRB also tells the IOP that the data is ready for output via the IOP Bus. The MCU Ready comparator checks to see if a destination module is ready or that an I/O low request signal can set the I/O Low Select (IOLSEL) flip-flop. Setting the IOLSEL flip-flop causes the contents of the IOP Data In Register, FROM, TO, and MOP signals to be read out onto the CTL Bus for transmission to Main Memory.

7-21. Multiplexer Channel

The Multiplexer Channel (figure 7-9) acts as a switch to enable one of its associated Device Controllers to transfer one word of data to or from memory via the IOP and then to allow another controller, based on priority, to perform its transfer. At all times, the Multiplexer Channel contains the current I/O program doubleword (paragraph 7-28) for each of the possible 16 Device Controllers. To accomplish this, the Multiplexer Channel has a 16-location, solid-state memory to contain the 16 I/O program words, and is responsible for fetching the next I/O progam doubleword when necessary. A more detailed discussion of Multiplexer Channel operations is contained in paragraph 7-33.

7-22. Selector Channel

The Selector Channel (figure 7-9) acts as a switch, but in a manner different from a Multiplexer Channel. The Multiplexer Channel switches between Device Controllers on demand, based on hardware priority, wheras the Selector Channel maintains the connection for one Device Controller until it has completed the I/O program. Therefore, only one I/O program is current at a given time for one channel. Another major difference, as shown in figure 7-9, is that the Selector Channel accesses memory directly for data and I/O program word transfers, rather than indirectly through the IOP. These features permit a very high speed data transfer rate. A more detailed IOP discussion of Selector Channel operations is contained in paragraph 7-47.

7-23. I/O SYSTEM FUNCTIONAL OPERATION

The following paragraphs contain discussions of I/O priorities, a summary of data routes, a comparison of basic transfer modes, and detailed discussions of the I/O hardware operations.

7-24. I/O Priorities

There are two types of priority in the I/O system; interrupt priority and service priority. The ability of a device to interrupt the CPU is based on a priority structure that is separate and distinct from the priority structure that handles service requests.

The interrupt poll determines the priorities of all I/O interrupts. The interrupt poll originates in the IOP (figure 7-11) and is wired in series through every Device Controller in the system. The proximity to the IOP on this line determines the interrupt priority of each controller. The desired wiring sequence is dependent on system configuration. Physically, the interrupt poll is a twisted-wire pair (signal and ground) connected into and out of each unit at INT POLL IN and INT POLL OUT connector pins. Functionally, the interrupt poll is an IOP response to a received Interrupt Request (INT REQ line in the IOP Bus). The poll propagates through each non-requesting unit and stops at the first requesting unit. The unit then returns INT ACK (Interrupt Acknowledge) and its device number to the IOP. The IOP then generates an interrupt signal to the CPU. When the CPU is ready to process the interrupt, it uses the device number saved in the IOP (Interrupt DEVNO Register) to refer to the device.

Service priority, unlike the series-linked structure of interrupt priority, is determined in two levels. For Multiplexer Channel devices, the first level determines the priority among two or more Multiplexer Channels. The second level determines the priority of each Device Controller associated with that Multiplexer Channel. Figure 7-11 shows only the first-level determination of priority among Multiplexer Channels by means of a data poll; the remaining priority determination is by logic not shown. The data poll operates very much like the interrupt poll. That is, when the IOP receives a Service Request, it sends out a data poll. The first requesting Multiplexer Channel encountered by the poll stops propagation of the poll and proceeds to specify the kind of service required. Therefore, since priority is determined by proximity to the IOP, the poll is wired through each Multiplexer Channel in the desired priority sequence. The second-level priority determination for Multiplexer Channel devices is by a service request number. Since each Multiplexer Channel can handle 16 Device Controllers, there are 16 service request numbers (0 through 15). Each Device Controller associated with a given Multiplexer Channel is uniquely wired by a jumper to connect to one of these 16 numbers to give the Device Controller a specific priority level. (Service request number 0 is the highest priority and 15 is the lowest priority.)
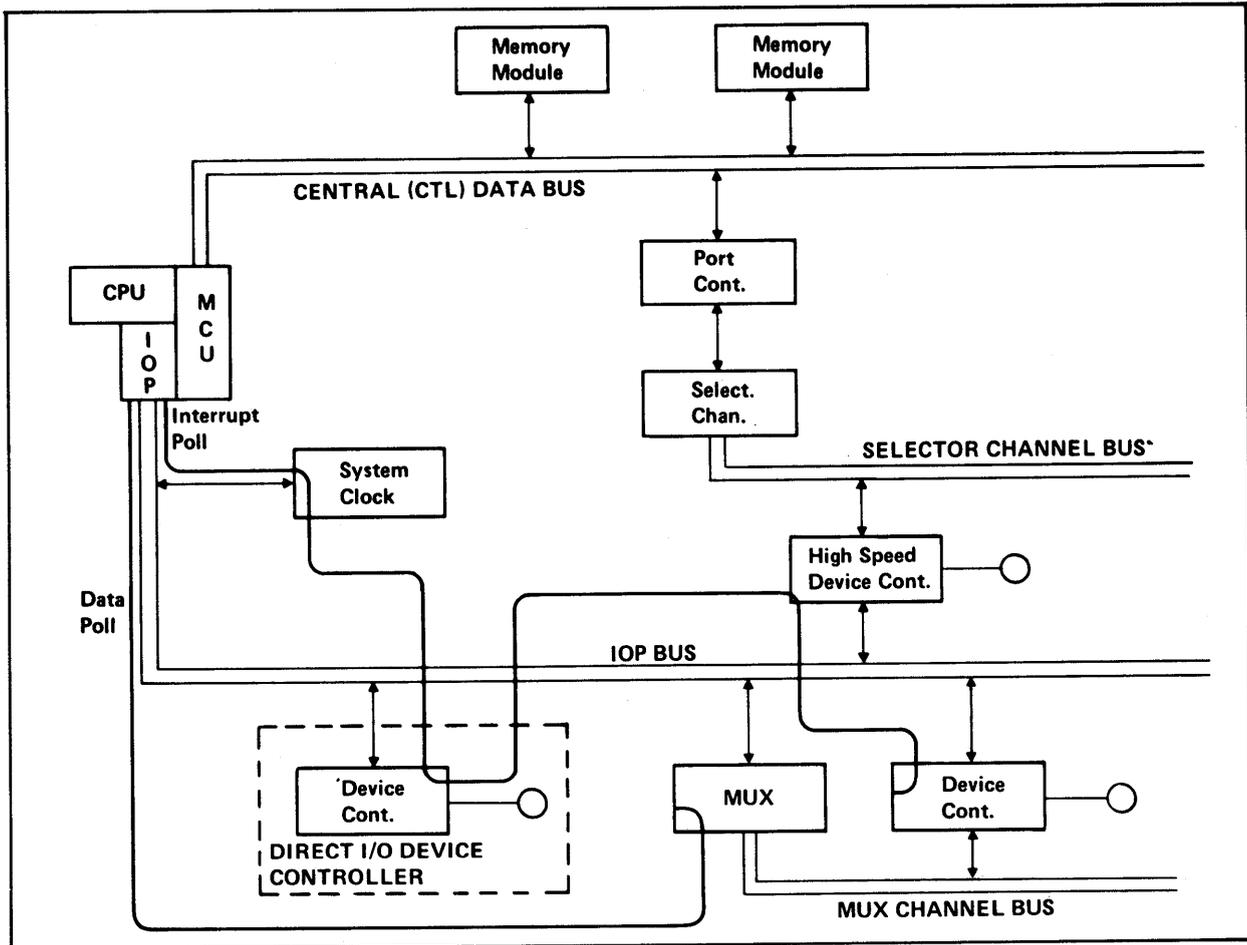
Figure 7-11. Interrupt Poll and Data Poll

Note

The service request number has no asociation with the devuce number. It is simply a convenient way by which a Multiplexer Channel can communicate with and assign priorities to its set of Device Controllers.

For high-speed Device Controllers, the Port Controller determines the first level of priority. Selector Channel 1 has highest priority and Selector Channel 4 has lowest priority. Although three Selector Channel ports are available (port 2 cannot be used), two Selector Channels is the maximum system configuration. The second-level determination is a simple preemptive process. The first device to be given an SIO instruction on a particular channel will have exclusive use of that channel until its I/O program is finished. No further SIO instructions for devices connected to that channel can be honored until that time.
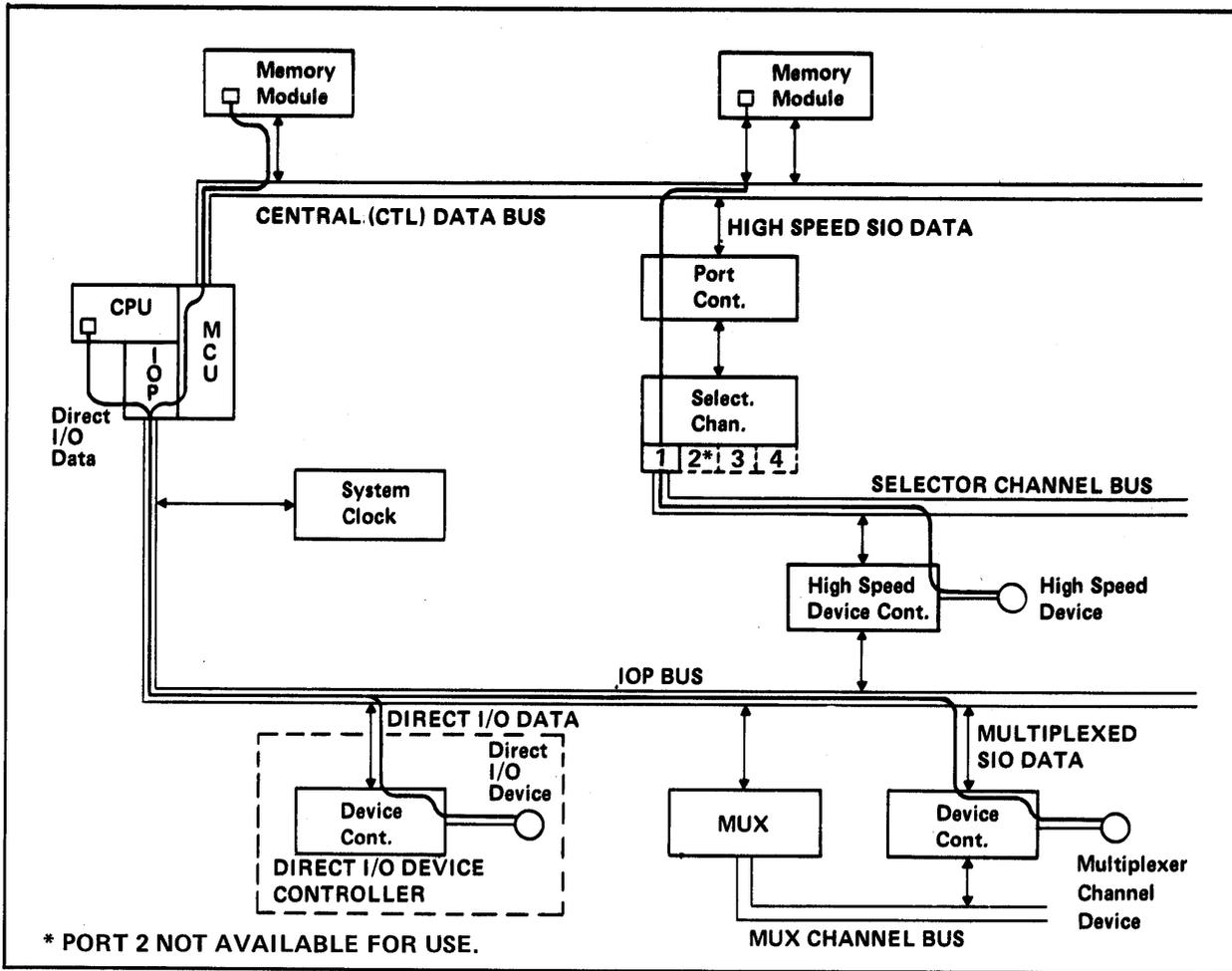
Figure 7-12. I/O Data Routes

7-25. I/O Data Routes

Data transfer routes for both low- and high-speed devices and direct I/O are shown in figure 7-12. At least one of each type of unit (two low-speed Device Controllers, one high-speed Device Controller, one Multiplexer Channel, two memory modules, etc.) is shown. The routes shown in figure 7-12 are the normal input/output data routes.

For direct I/O instructions to a Multiplexer Channel device, information is transferred to or from the TOS in the CPU via the IOP and IOP Bus. The information could be device status (for TIO or rejected SIO, RIO, or WIO), control information (CIO), or data (RIO or WIO). For SIO operation, data is transferred to and from memory via the CTL Bus, IOP, and IOP Bus.

For direct I/O instructions to a Selector Channel device, the data route is the same as for a Multiplexer Channel device; to or from the TOS in the CPU via the IOP and IOP Bus. For SIO operation, data is transferred to and from memory via the CTL Bus, Port Controller, Selector Channel, and Selector Channel Bus.

7-26. I/O Transfer Modes

There are three basic modes of data transfer; direct I/O and two SIO type transfers. Direct I/O operation consists of the transfer of a single word (per CPU instruction) between the CPU and a Device Controller. The Multiplexer and Selector Channels are not involved. Direct I/O operations are discussed in paragraph 7-27.

During the two SIO type transfers, the CPU gives the I/O system a command to "start I/O" for a particular device and the I/O system proceeds to execute an I/O progam for that device. The program, which resides in memory, controls the input and output of data. Specifically, the two SIO transfer modes are moderate-speed transfers via the Multiplexer Channel and high-speed transfers via the Selector Channel. Figure 7-12 illustrates the difference in data routes for these two modes. However, the significant difference is in the sequencing of transfers for multiple Device Controllers. Paragraphs 7-31 and 7-30 describe the difference between Multiplexer Channel and Selector Channel transfers.

7-27. DIRECT I/O. During direct I/O operations, the CPU transfers information directly to and from a Device Controller without involving memory, the Multiplexer Channel, or Selector Channel. (See figure 7-12.) For each I/O instruction, one word is transferred either to or from the CPU TOS. The CPU has four direct I/O instructions; Test I/O (TIO), Control I/O (CIO), Read I/O (RIO), and Write I/O (WIO).

Note

Some Device Controllers cannot accept all direct I/O commands (see the specific subsystem manual). However, all Device Controllers will accept a TIO or CIO using bits 0 and/or 1. Bit 0, the standard control bit, causes a master clear of the subsystem. Bit 1 causes the subsystem interrupt logic to reset.

The TIO instruction obtains the contents of the Device Controller's Status Register and pushes it onto the TOS. When the CPU encounters a TIO instruction, its TIO microprogram sends a command word to the IOP Control circuit (figure 7-10) in the IOP. The IOP then issues a Service Out (SO) and a TIO command on the IOCMD lines via the IOP Bus to the device addressed by the Device Number (DEVNO) code. The addressed device is therefore enabled to accept and decode the command, and accordingly, reads the contents of the Status Register onto the IOD lines. SI is also issued which causes the IOP to load the status word into the IOD Data In Register and informs the CPU that the word is present. The CPU then issues a Read signal which reads the contents of the IOD Data In Register to the S-Bus. From the S-Bus, the status word is placed on the U-Bus and pushed onto the stack.

The RIO instruction begins by performing a TIO to the Device Controller as previously discussed to check the Read/Write OK status bit (bit 1). If status is acceptable, the same sequence is repeated except that the command is RIO and data is transferred from the Data In Buffer rather than the Status Register.

The CIO instruction obtains a control word from the TOS register (RA) and sends it to the Device Controller's Control Register. When the CPU encounters a CIO instruction, its CIO microprogram loads the RA contents into the IOD Data Out Register and then issues a command word to the IOP. The command word causes a CIO IOCMD to be issued to the Device Controller addressed by the DEVNO code along with SO. Simultaneously, the contents of the IOD Data Register are read out onto the IOD lines. When the Device Controller decodes the IOCMD, it loads the word on the IOD lines into its Control Register and returns SI to the IOP. When the IOP receives SI, the IOP returns a signal to the CPU, indicating completion of the instruction.

The WIO instruction begins by performing a TIO to the controller to check the Read/Write OK status bit. If status is acceptable, the remaining operations for the Write I/O instruction are the same as for CIO except that the information sent is a data word, the IOCMD is WIO instead of CIO, and the information is loaded into the Device Controller's Data Out Buffer instead of its Control Register.

7-28. PROGRAMMED I/O. When a driver issues an SIO instruction to a requested Device Controller, the I/O hardware begins to execute the I/O program independently of the CPU. The CPU is then free to continue processing in parallel with the I/O operations. Paragraphs 7-29 and 7-30 define the elements of an I/O program and describe the hardware actions occurring after the SIO instruction is issued.

7-29. I/O Program Word. The format of an I/O program word is illustrated in figure 4-14. Two computer words are used to accommodate the 32-bit word length. The first word is designated as the I/O Command Word, or IOCW, and the second word is designated as the I/O Address Word, or IOAW. Data chaining occurs for WRITE and READ orders if bit 0 of the IOCW is a "1". This bit may be a "1" for a WRITE order followed by a WRITE or for a READ order followed by a READ. This will permit the hardware to treat the counts of each order as a continuous chained count, without reinitializing for each order. The DC bit should be "0" for all other orders. The count field of the IOCW contains the least significant 12 bits of a two's complement negative count value for WRITE and READ orders. The count is a word count, independent of the particular recording format (bytes, words, or records). For a CONTROL order, these 12 bits are used for control information in addition to the 16 control bits in the IOAW (a total of 28 bits). Complete definitions of the I/O orders are contained in paragraph 4-16, Instruction Commentary number 8.

7-30. Typical I/O Program Operation. Figure 7-13 illustrates the sequence of operations occurring as the result of an SIO instruction. The sequence is as follows:

a. The SIO instruction (decoded by the CPU) fetches the device number given at S-K (1, figure 7-13) in the stack (2), and puts the TOS into the first word of its DRT entry as the I/O program pointer.

b. SIO then sends the device number (3) to the IOP Control Register and sends an SIO command (4) to the IOP.

c. The IOP issues the SIO command (5) to the Device Controller and execution by the hardware begins. The CPU is now free to continue execution elsewhere.

d. On demand from the Multiplexer Channel, the IOP obtains the program pointer (6) from the DRT. (The Selector Channel obtains the program pointer directly, not via the IOP.) As illustrated previously in figure 7-4, the address is obtained by multiplying the device number by four. The program pointer is the first word of the four-word DRT entry.

e. The program pointer points to the first doubleword of the I/O program (7). The pointer is updated to point at each I/O program double word as the program progresses. (The Selector Channel, to minimize memory fetches, copies the pointer value into a register and updates the pointer internally; the Multiplexer Channel updates the pointer directly in the DRT.)

f. The sample I/O program operates as follows. The first doubleword (7) contains a CONTROL order which enables the hardware I/O subsystem for this device. The second doubleword contains a READ order which causes the subsystem to read 4096 words (or 8192 bytes) into the data buffer (8) whose starting location is given in the IOAW word. Since the data chaining bit is on, the next (third) doubleword is also a READ order which specifies the remaining count required to fulfill the I/O request. (Additional READ orders could be given for larger requests.) The IOAW will either specify an additional buffer area contiguous to the first 4096-word buffer if desired or, in another part of memory.

g. When the transfer is complete, the final doubleword contains an END order, which obtains the result of the transfer (device status) and loads it into the IOAW; the END order then tells the controller to generate an interrupt to inform the software that the transfer is complete.

h. At the completion of an I/O program, the Selector Channel returns the current program pointer value to the DRT. The Multiplexer Channel does not take any special action since it updates the DRT after each order fetch.
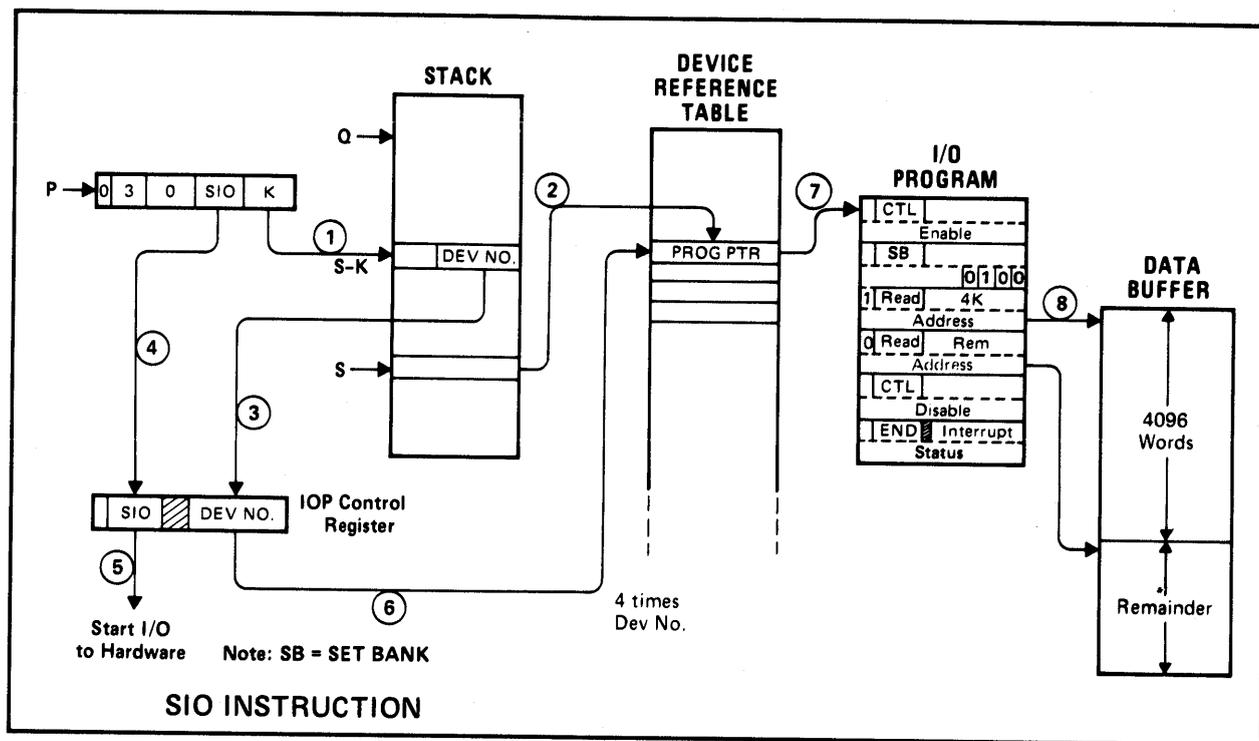
Figure 7-13. I/O Program Operation

7-31. Multiplexer Channel Transfers. A multiplexer transfers data from many sources on an apparently simultaneous basis. Therefore, it is the function of the Multiplexer Channel to perform one discrete operation for one Device Controller (such as to transfer one word to or from memory), and then check to see which Device Controller has highest priority for the next discrete operation. The Multiplexer Channel includes a 16-location solid-state memory as shown in figure 7-14. Each location in this memory corresponds to one of the 16 Device Controllers connected to the Multiplexer Channel Bus. Each location contains the information required to execute the next operation for that device. Typically, this would be the current I/O program word. When a particular Device Controller is selected for service, the stored word is read out to a set of registers and the Multiplexer Channel executes the indicated operation. Then, the information is updated for the next anticipated operation and is stored back in the memory location.

The overall Multiplexer Channel operating sequence is as follows. Each time a Device Controller requires a new I/O program, it causes the Multiplexer Channel to fetch an address from the DRT (1, figure 7-14) and load it into its solid-state memory location. (Some other operation for another device could be interleaved after each of these steps.) Then (2), the I/O program doubleword is fetched and loaded into the same memory location. This I/O program word is then read out (3), control signals are issued to the Device Controller (4), and the updated operation information is stored back into the memory location (5). If the Device Controller was commanded to transfer data, it issues a
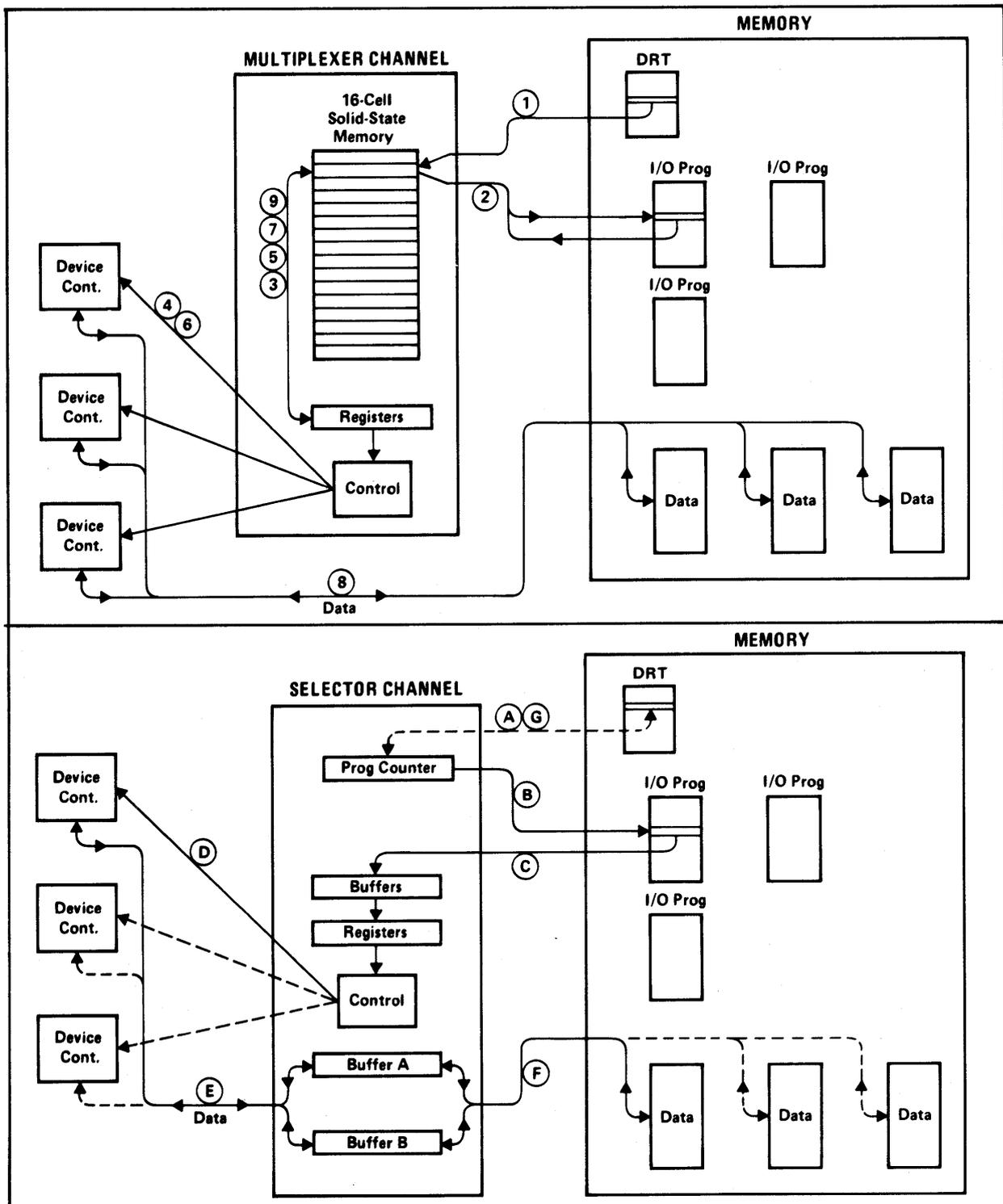
Figure 7-14. Multiplexer and Selector Channel Comparisons

service request when it is ready (6), causing another readout of the stored information (7) and a transfer of data (8). Updated operation information is restored (9). Steps (6) through (9) are repeated for each word transferred. (For a more detailed discussion, refer to paragraph 7-33.)

7-32. Selector Channel Transfers. A Selector Channel transfers data from many sources in a data block manner. That is, it locks onto one Device Controller until the I/O program for that device is completed. Then, a check is made to see which Device Controller has highest priority for the next block transfer. Since only one I/O program will be in progress as long as a particular device is selected, the Selector Channel is designed to facilitate very high speed transfers. The Selector Channel uses double-buffering for both data and I/O program words. (See figure 7-14.) For data, this permits device/channel transfers to overlap channel/memory transfers. For I/O program words, this permits the next program word to be fetched from memory while the current word is active. Both of these features contribute to the speed capablity. In addition, the necessity to repeatedly fetch a DRT entry for the address of the current I/O program word (as is done by the Multiplexer Channel) is eliminated by including a Program Counter in the Selector Channel. The Program Counter is loaded with the initial address contained in the DRT, but is thereafter incremented (or altered for jumps) internally in the Selector Channel. To provide software compatibility with Multiplexer Channel transfers, the final value of the Program Counter is automatically restored in the DRT at the end of the program. Software cannot distinguish whether the transfer occurred by way of the Multiplexer Channel or the Selector Channel.

The overall Selector Channel operating sequence is as follows. When the Device Controller is commanded by the CPU to "start I/O", it causes the Selector Channel to fetch the starting address of the I/O program from the DRT (A). This address is used to fetch an I/O program doubleword (B) and load it into either the active control registers or, during order prefetch, into the buffers (C). The Program Counter is incremented after each fetch. Control signals are issued to the Device Controller (D), and (E), if the command is a Read, the Device Controller reads data into buffer A (or buffer B if A is full). If the command is a Write, the Device Controller writes data from buffer A (or buffer B if A is empty). Meanwhile (F), the Selector Channel attempts to keep both buffers full for output or both empty for input, by transmissions to or from memory. At the end of the block transfer, the next I/O program word is fetched. Repeat back to step (B). At the end of the I/O program, the Selector Channel stores its Program Counter contents into the DRT (G). (For a more detailed discussion, refer to paragraph 7-47.)

7-33. Multiplexer Channel Operations

A detailed discussion of Multiplexer Channel operations is contained in paragraphs 7-34 through 7-46.

7-34. INITIALIZE. When the CPU encounters an SIO instruction, the CPU outputs a command word to the IOP Control Register. (See figure 7-10.) The IOP relays this information to the Device Controller (figure 7-15) via the IOP Bus. The DEVNO on the IOP Bus is compared with the internally wired device number. A true result, together with the SO signal from the IOP, enables the IOCMD
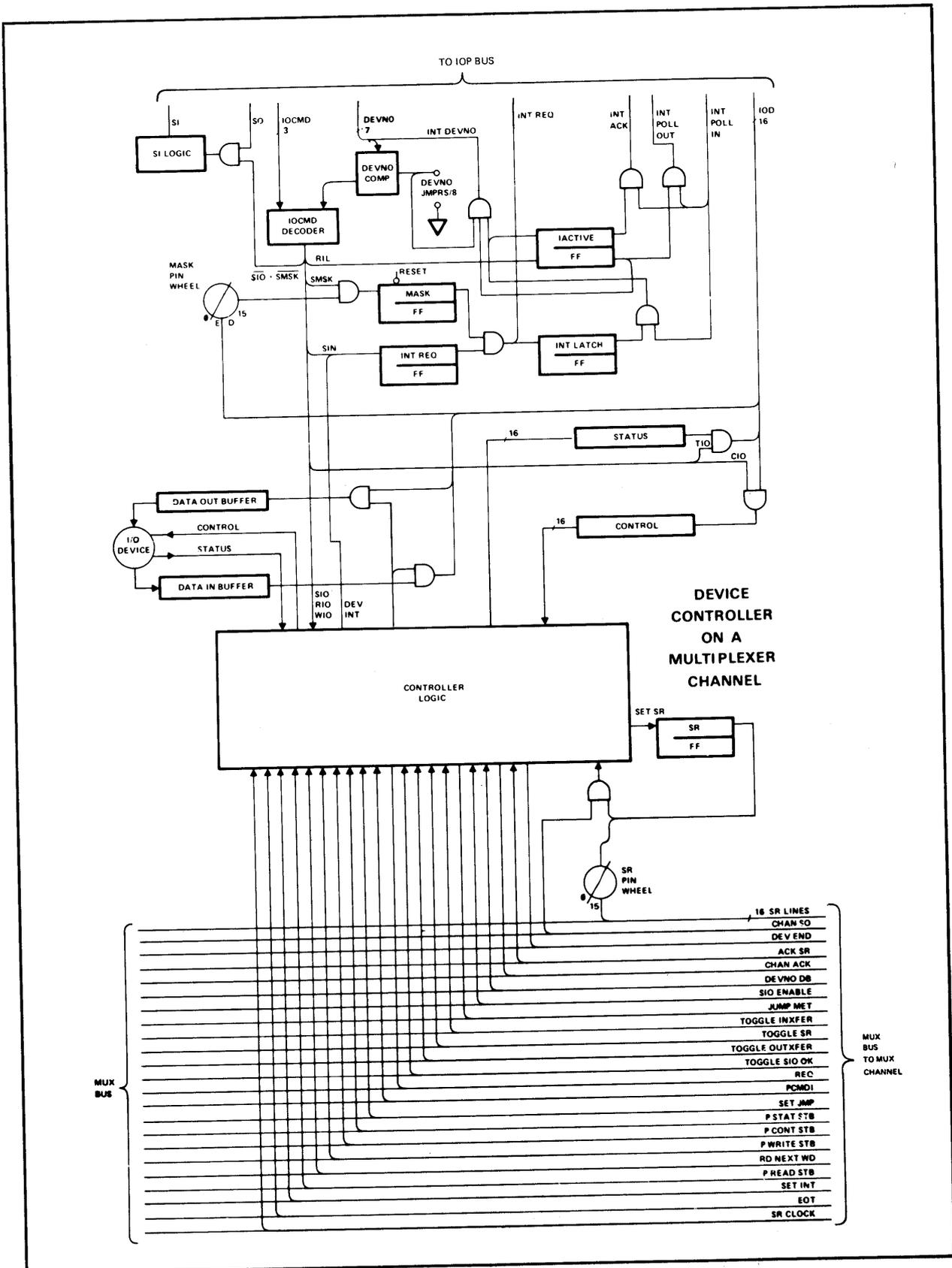
Figure 7-15. Multiplexer Channel and Device Controller
Simplified Logic Diagram

to be decoded. The IOCMD in this case is SIO which, when decoded, sets the Service Request (SR) flip-flop. The Service Request (SR) along with a Request (REQ) signal is sent Multiplexer Channel Bus to the Multiplexer Channel. (See figure 7-16.) The SR and REQ cause the Multiplexer Channel, instead of the Device Controller, to return SI and force a DRT FEtch to be the first operation performed for the Device Controller on the next Service Request from the Device Controller. An SIO to a Device Controller temporarily inhibits service requests from all other Device Controllers. Therefore, the only Device Dontroller requesting is the one receiving the SIO command. The Priority Encoder/Select Decoder then issues a 4-bit binary code which corresponds to the SR line number. The binary code is used as a RAM address to enable one of the 16 locations in the solid-state multiplexer memory. The solid-state memory contains separate RAM for each of the IOCW and IOAW parts of the I/O program doubleword, and one to specify the state (or next operation). In this case, a DRT fetch and an Auxiliary RAM containing the I/O order. The IOCW is contained in the Order RAM (16 bits), the IOAW is contained in the Address RAM (16 bits) and the state is contained in the State RAM (4 bits). Each of the addressable locations therefore contains 36-bits.

For the initialize operation, the State RAM location for the requesting device is forced to the condition required for a DRT fetch. Once this is done, the Multiplexer Channel returns a DI signal to the IOP, which in turn, causes the IOP to free the CPU to execute other instructions. The Auxiliary RAM uses bits 12 through 15 of the Set Bank I/O order on the IOD lines to send IOX (B12 and B13) to the CPU Mod Select switches (see figure 7-10) and to send IOX (B14 and B15) to memory (see figure 6-2) as part of the 18-bit memory address. The IOP Mod Select Switches (figure 7-10) supply an IOTO signal to the MCU where it is gated to memory (figure 6-2). The Multiplexer Channel will transmit a bank number of 0 unless actually moving data for a Read or Write order pair. In the following description, unless otherwise specified, the bank number will be considered to be zero.

7-35. DRT FETCH. The Service Request received at the Multiplexer Channel from the Device Controller (figure 7-16) causes the Transfer/Control Logic to send a Multiplexer Channel Service Request (HSREQ) to the IOP and also sets the SR latch. Any of the 16 SR inputs can set this latch and generate an HSREQ signal. However, only the highest priority requests will be honored by the Priority Encoder. The IOP, when it receives an HSREQ, issues a DATA POLL to all Multiplexer Channels. The highest priority Multiplexer Channel stops the propagation of the poll (since SR Latch is set), and its transfer logic is enabled. First, the contents of the address RAM location are loaded into the State, Address, Auxiliary, and Order Registers. The state bits tell the transfer logic to send out a command to the Device Controller via the Multiplexer Channel Bus along with the Service Request number signal (which is returned on the same line used for Service Request) and SO. This command tells the Device Controller to read out its device number to the IOP Bus.
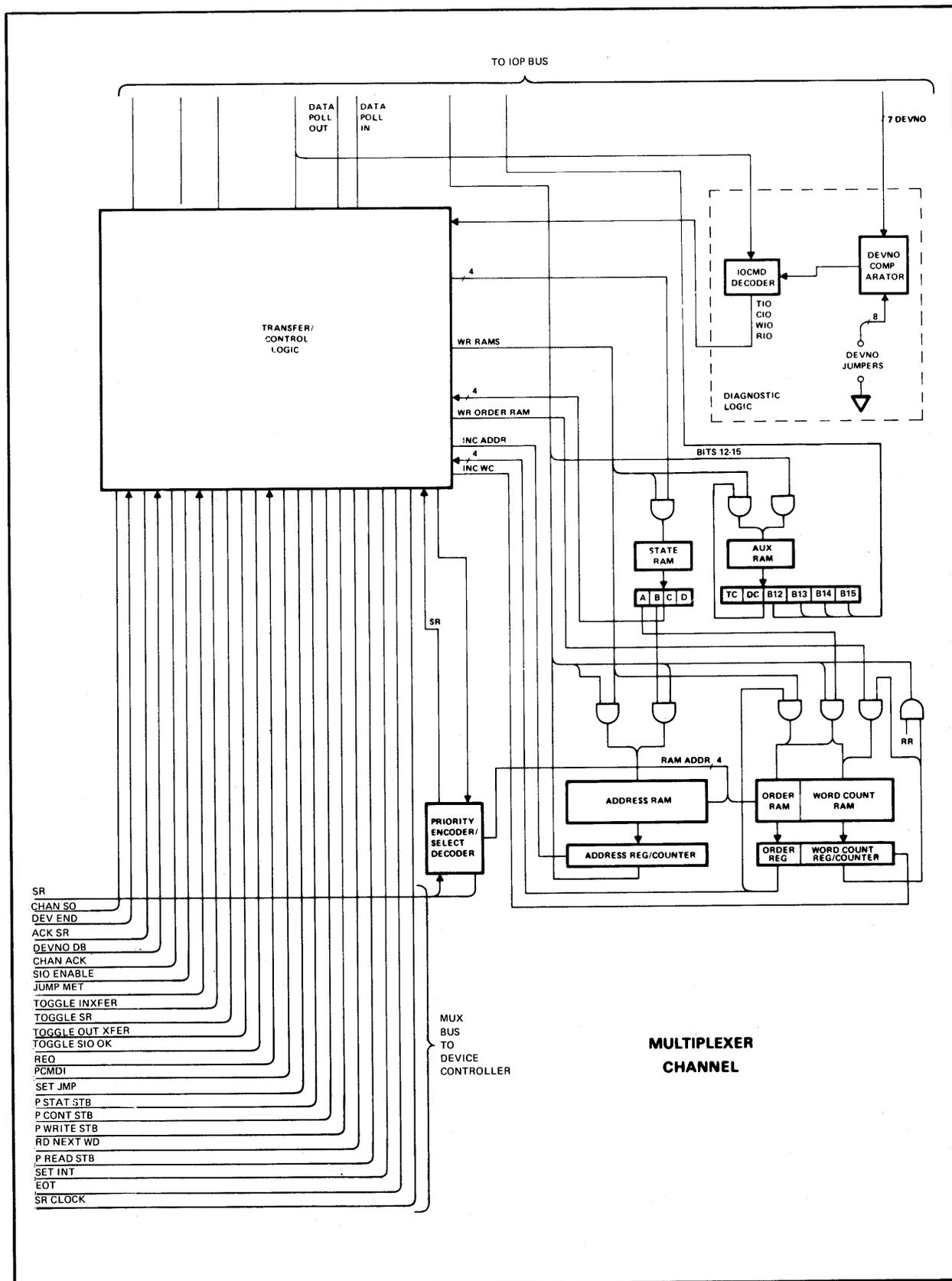
Figure 7-16.   Multiplexer Channel Simplified Logic Diagram

The Device Controller, for a DRT fetch, reads out its device number (DEVNO) onto the IOD lines. Instead of being read onto the eight least significant lines of the bus (8 through 15), the number is read onto lines 6 through 13, which is left-shifted by two bits. This effectively multiplies the number value by four, thus automatically providing the correct address for that device's DRT entry. (Remember that each device uses four locations in the DRT.) Simultaneously, the Multiplexer Channel is returning an SI response to the IOP along with an IOCMD which tells the IOP to accept the address existing on the IOD lines and that a DRT fetch from that address is required.

The IOP then proceeds to fetch the DRT entry. (See figures 6-1 and 7-10.) The IOP issues an IOLRQ to its MCU with an appropriate MOP to read memory. When select occurs, the address is transmitted to memory. When memory returns the DRT entry contents, I/O Strobe (IOSTRB) loads the word into the IOP Data Out Register. The IOP Data Out contents are then read out onto the IOD lines and SO is issued. Upon receipt of SO, the Multiplexer Channel loads the DRT word into the Address RAM, restores the Order Register contents into the Order RAM, and sets the State RAM to the condition required for an I/O program word fetch. Meanwhile, the IOP transfers its copy of the DRT word from the Data Output Register to the Data Input Register, increments it by two, and sends it back to the DRT in memory. (This is an anticipatory move, as the Address RAM presently contains the desired address for the next operation; the incremented address in the DRT will not be used until the next DRT fetch.)

7-36. I/O PROGRAM WORD TRANSFERS. Each I/O program word consists of two words in Bank 0 of Main Memory; the IOCW and the IOAW. Therefore, two memory transfers are required. The first transfer is to fetch the IOCW. Depending on the order that the IOCW contains, the second transfer may be either a fetch or a store.

7-37. IOCW Fetch. The SR flip-flop in the Device Controller is still set from the previous procedure (DRT Fetch, paragraph 7-4), so HSREQ is still present at the IOP. The IOP therefore issues a new DATA POLL. The SR Latch in the Multiplexer Channel, which had reset on the trailing edge of the previous SO, has become set again since the SR input was still present at the next clock. Thus DATA POLL is stopped from further propagation, and the Transfer/Control Logic is enabled again.

The contents of the Address RAM location are loaded into the State, Address, and Order Registers. The state specifies an IOCW fetch, so the transfer logic reads out the contents of the Address Register and issues SI and the IOCMD "transfer from memory" to the IOP. The address now on the IOD lines is the word previously fetched from the DRT, indicating the address of the I/O program word. The IOP issues an IOLRQ to the MCU. When priority allows, the MCU transmits the address to memory. When memory returns the IOCW, IOSTRB loads this word into the IOP Data Out Register in the IOP. The IOP then reads the word out to the IOD lines and issues SO.

Upon receipt of SO, the Multiplexer Channel loads the IOCW into the Order RAM. If the order is Control, the Multiplexer Channel issues a command through the Multiplexer Channel Bus so that the Device Controller will also load the IOCW into its Control Register. The contents of the Address Register/Counter is incremented by one and restored in the Address RAM. The next state, fetch or store IOAW, is stored in the State RAM.

The next operation, transfer of the IOAW, begins the same way for each of the orders. That is, SR to the Multiplexer Channel causes a HSREQ to be sent to the IOP. The IOP returns a DATA POLL which enables the Multiplexer Channel to load the addressed RAM location into the State, Address, and Order Registers. Action after this point varies depending on the order that the IOCW contains.

7-38. IOAW Fetch. The Read, Write, Jump, Control, and Interrupt orders each cause an IOAW fetch. However, the action taken upon receipt of the IOAW is different in each case. The IOAW fetch begins by reading the contents of the Address Register (incremented on the trailing edge of DATA POLL in the IOCW fetch procedure) to the IOD lines. The Multiplexer Channel also issues SI and the IOCMD "transfer from memory" to the IOP. The IOP issues IOLRQ with MOP to its MCU to request a memory read. When memory returns the contents of the address location, IOSTRB loads it into the IOP Data Out Register. The IOP then reads the contents of the IOP Data Out Register to the IOD lines and issues SO. For Read, Write, Interrupt, and Jump orders, the Multiplexer Channel will store the word (IOAW) into the Address RAM. For a Control order, the Multiplexer Channel issues a command via the Multiplexer Channel Bus to tell the Device Controller to load the word into its Control Register. For an Interrupt order, the fetched information is loaded into the Address RAM but is disregarded.

For Read, Write, and Conditional Jump, a command is sent to the Device Controller to specify conditions for the next action. For Read, the in-transfer condition is set. For Write, the out-transfer condition is set. For Conditional Jump, the Device Controller is given the choice of setting or not setting the "jump met" condition. If "jump met" is true in the next DRT fetch sequence (or if an unconditional jump was given), a store operation (instead of fetch) will occur. That is, the Multiplexer Channel will cause the contents of the Address Register to be sent to the IOP which will increment the value by two before storing in the DRT. (The Address RAM already contains the correct jump address, so a DRT fetch is not necessary.)

7-39. IOAW Store. The Sense, End, and Return Residue orders each cause an IOAW store operation. This operation begins as the Multiplexer Channel reads the incremented contents of the Address Register out to the IOD lines and issues SI with a "transfer-to-memory" IOCMD. The IOP loads this address into its Memory Address Register (MAR) and issues IOLRQ to its MCU with a Clear/Write MOP. The insuing CTL Bus transmission prepares memory for receiving data. Simultaneously, the IOP has issued SO to the

Multiplexer Channel to ask for data. Depending on the current
order, the Multiplexer Channel either gates the Order Register
contents out to the IOD lines (Return Residue order) or issues a
command to the Device Controller, telling it to read out its
Status Register contents (Sense or End orders). When either ac-
tion occurs, SI is returned to the IOP which causes the IOP to
load the IOD information into its Memory Data Input Register.
The IOP then proceeds to transmit this information to memory by
issuing IOHRQ to its MCU. When the transmission occurs, the ap-
propriate information will be stored into the IOAW location of
the I/O program doubleword.

7-40. Next Operation. At this point (after the IOAW fetch or
store), the I/O program word transfer is complete. In addition,
all orders except Read and Write (i.e., Control, Set Bank, Sense,
Return Residue, End, Jump, and Interrupt) are fully executed.
The next operation for any of these orders (except End, which
terminates the program) is to return to the DRT fetch operation.
For Read or Write, however, a data transfer is indicated.

7-41. DATA TRANSFERS. Data transfers are very similar to the I/O
program word transfers previously described, in that the basic
operation is to fetch or store information using a memory address
that has been put in the Address RAM by a previous operation.
For I/O program word transfers, the previous operation was the
DRT fetch; for data transfers, the previous operation is the I/O
program word transfer. The main difference is that the data
transfer is device-initiated. That is, when a device is ready
for a transfer, it informs its Device Controller which then is-
sues an SR to the Multiplexer Channel. Another difference is
that the word count and memory address contained in the Order and
Address Registers must be incremented during each word transfer.
Each data transfer consists of two distinct steps; the transfer
of an address to memory and the transfer of data to or from that
address. The first step (address to memory) is the same for ei-
ther output or input.

7-42. Address Transfer. When the device sets the Device Control-
ler's SR flip-flop, the SR signal to the Multiplexer Channel gen-
erates an HSREQ signal to the IOP. The IOP returns DATA POLL
which enables the Multiplexer Channel to begin its transfer.
First, the addressed RAM location is read out to the State, Ad-
dress, Auxiliary, and Order Registers. Then the Address Register
contents are read out to the IOD lines and the Auxiliary Register
to the IOX lines. Also, SI and an appropriate IOCMD ("transfer
to memory" or "transfer from memory") are sent to the IOP. The
IOP loads the address and issues IOLRQ to its MCU with a Read/
Restore or a Clear/Write MOP. When priority allows, the MCU will
transmit the address to memory. Simultaneously, the Multiplexer
Channel resets the Device Controller's SR flip-flop via the Mul-
tiplexer Channel Bus and increments the Address and Order
Registers.

7-43. Output Transfer. When memory returns a data word, IOSTRB loads the word into the IOP Data Out Register. The IOP then reads the contents of this register out to the IOD lines and issues SO. Upon receipt of SO, the Multiplexer Channel issues a command to the Device Controller via the Multiplexer Channel Bus telling the Device Controller to load the word on the bus into its Data Out Buffer. The Device Controller returns SI to the IOP and proceeds to output the word to the device. Simultaneously, the Multiplexer Channel restores the contents of the State, Address, and Order Registers into the RAM location, and the output data transfer is complete. Some other operation for another device could be interleaved here. Otherwise, the entire data transfer procedure repeats.

7-44. Input Transfer. As the input data transfer procedure begins, memory is expecting the data. The procedure begins when the IOP sends SO to the Multiplexer Channel to ask for data. Upon receipt of SO, the Multiplexer Channel issues a command to the Device Controller via the Multiplexer Channel Bus, telling the Device Controller to read the contents of its Data In Buffer out to the IOD lines. When the Device Controller does this, it also sends an SI response which causes the IOP to load the data into its Memory Data Input Register. The IOP then issues IOHRQ to its MCU with a Write MOP, causing a data transmission to memory via the MCU Bus. Simultaneously, the Multiplexer Channel restores the contents of the State, Address, Auxiliary, and Order Registers into the RAM location, and the input data transfer is complete. Some other operation for another device could be interleaved here. Otherwise, the entire data transfer procedure repeats.

7-45. End Of Transfer By Word Count. If the word count rolls over while incrementing (during the address transfer sequence), then in the data transfer sequence the Multiplexer Channel will issue a command which will reset the in-transfer or out-transfer condition in the Device Controller. Also, an End-of-Transfer (EOT) signal accompanies the last command from the Multiplexer Channel to read or write. The Device Controller logic will therefore not transfer any more data to or from the device. It will, however, issue one more SR. In the Multiplexer Channel, the transfer logic sets the next state to DRT fetch, when restoring the RAMS at the end of the final data transfer. When the Multiplexer Channel receives the SR from the Device Controller and when priority conditions are satisfied, a new DRT fetch procedure will begin. This advances the I/O program to the next IOCW.

7-46. End Of Transfer By Device. On termination of a transfer by a device, the Device Controller issues an SR to the Multiplexer Channel. The Multiplexer Channel responds with CHAN SO. The Device Controller returns a Device End signal that causes the Multiplexer Channel to initiate a DRT fetch, thus advancing the I/O program to the next IOCW.

7-47. Selector Channel and Port Controller Operations

A Selector Channel operates only one I/O program and transfers blocks of data for only one device at a time. Data is passed back and forth from memory, through the CTL Bus, to the Port Controller, Selector Channel, and through the Selector Channel Bus to the Device Controller and the operating device. (See figure 7-12.) A detailed discussion of Selector Channel and Port Controller operations is contained in paragraphs 7-48 through 7-60. Since there may be two Selector Channels operating in the system, the Port Controller is discussed first to explain how each Selector Channel gains access to the CTL Bus.

7-48. PORT CONTROLLER. The Port Controller (figure 7-17) provides three ports (ports 1, 3, and 4) to the CTL Bus for I/O programs and data transfers between Selector Channels and Main Memory. Figure 7-17 illustrates the logic for only one port; logic for the remaining ports is identical to the one shown. The Port Controller Bus contains five sets of signal lines; one set for each of four ports (port 2 is not available for use) and one set to data lines which is shared by all four Selector Channels. (It should be noted that although three ports are available for Selector Channels, only two Selector Channels can be installed in the system simultaneously.)

The Port Controller is assigned a module number 4 which gives the Port Controller a transmission priority higher than the CPU or IOP as discussed in Section II. A Selector Channel requiring transfer of a word to or from memory, presents the Port Controller with a request for a Write or a Read operation along with the memory module number (0,1,2, or 3) to which the address will be sent. A Write operation consists of a Low Request (LREQ) for an address transfer followed by a Low Select (LSEL) of that address from the Selector Channel to memory via the CTL Bus; then a High Request (HREQ) for a data transfer followed by a High Select (HSEL) of that data to memory, via the CTL Bus. A Read operation consists of a LREQ for an address transfer followed by a LSEL of the address to the bus and memory. Then, a wait for a return transfer of data to the Port Controller from the module to which the address was sent. This return transfer of data is indicated to the Selector Channel by the STRB (Strobe) signal. While one section of the Port Controller is waiting, another section could be intructing another part of memory to fetch or store a data word for another Selector Channel. Priority is resolved among the three ports in the Port Controller on the following basis: Low requests with the desired destination module ready are granted first to Selector Channel 1, next to Selector Channel 3, and last to Selector Channel 4. A High Request for any Selector Channel takes precedence over all Low Requests.

The Write sequence is as follows: A Write on the request lines to the Port Controller sets the LREQ flip-flop and sets the MOP flip-flop to the Write state. The TO lines from the Selector Channel are clocked into the TO Register and the content is then compared with the Ready (RDY) line for that module. When the
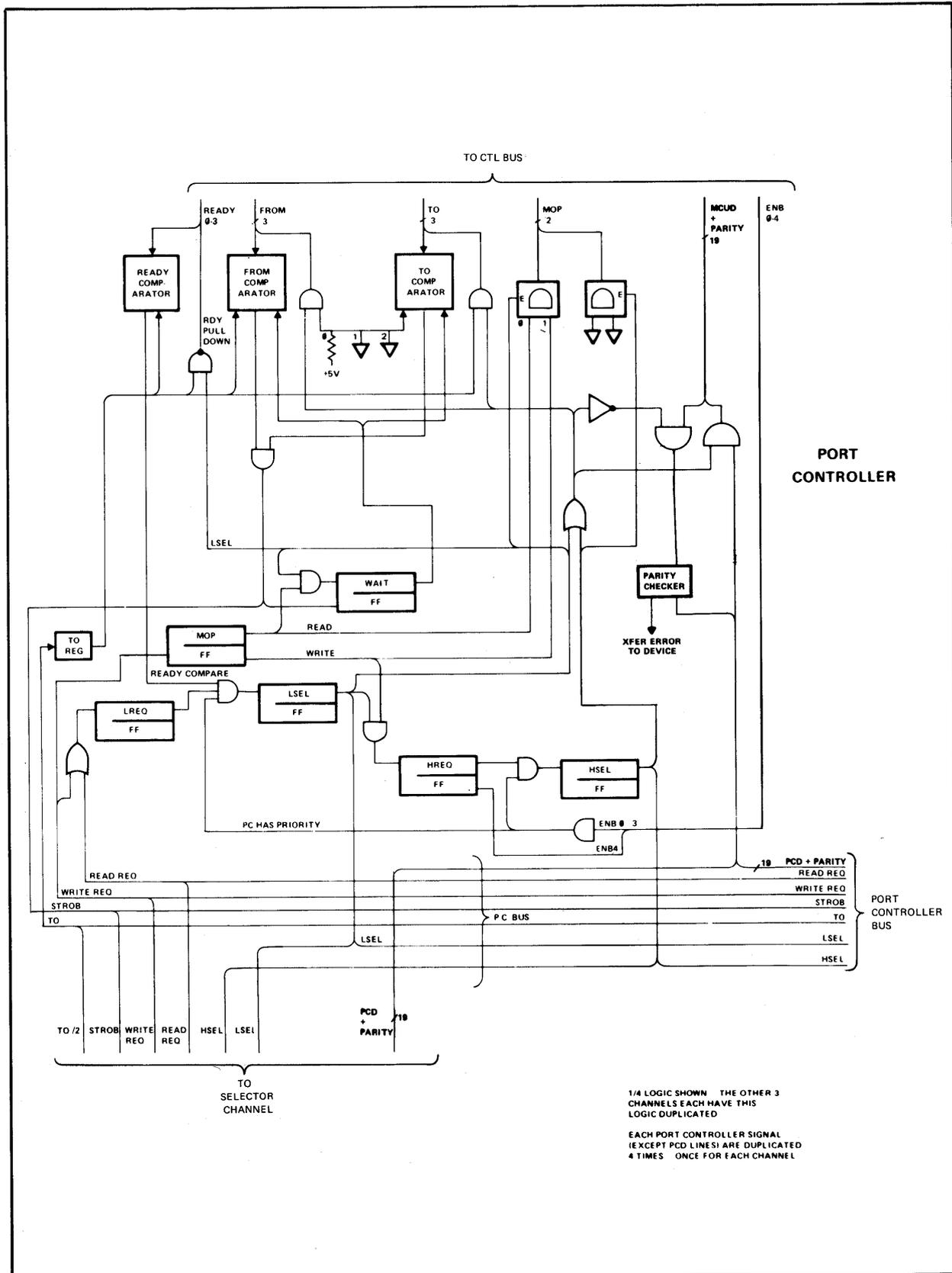
Figure 7-17. Port Controller Simplified Logic Diagram

destination is ready, the ENB is present, the Port Controller has priority, and the LSEL and HREQ flip-flops are set. LSEL gates the address from the Selector Channel to the CTL Bus along with TO, FROM, and MOP. LSEL also pulls the destination's RDY line low. Then, when ENB is present, the HSEL flip-flop is set. HSEL gates data form the Selector Channel to the CTL Bus along with TO, FROM, and MOP.

The Read sequence is as follows: A Read on the request lines to the Port Controller sets the LREQ flip-flop and sets the MOP flip-flop to the Read state. The TO lines from the Selector Channel are clocked into the TO Register and the content is compared with the RDY line for that module. When the destination is ready, the ENB is present, the Port Controller has priority, and the LSEL flip-flop is set. LSEL gates the address from the Selector Channel to the CTL Bus along with TO, FROM, and MOP. LSEL also sets the Wait flip-fop. Then, when returning data is present on the bus and the TO and FROM comparisons match, a STRB signal is sent to the Selector Channel to tell it to accept the data on the Port Controller Data (PCD) lines.

7-49. INITIATOR SEQUENCE. The following paragraphs describe how the Selector Channel's program counter is initialized as the first step in executing an I/O program for one device. The Selector Channel Bus originates at the Selector Channel and is routed to all Device Controllers controlled by this Selector Channel. The Selector Channel Bus is similar to the Multiplexer Channel Bus in purpose, but differs in that it uses 16 lines for transfer of control, status, and data words between the Device Controller and Selector Channel; the corresponding Multiplexer Channel Bus lines are used as service request lines for up to 16 devices.

The initiator sequence begins when the CPU encounters an SIO instruction. The CPU, under control of its SIO microprogam, outputs a command word to the IOP Control Register. (See figure 7-12.) This initial command is a TIO to see if there is already an I/O program active on the channel. The IOP issues the TIO with SO and DEVNO on the IOP Bus. The Device Controller compares DEVNO with its internal wired device number and a true comparison, with SO, causes the Device Controller to return SI to the IOP with a 16-bit status word on the IOP Bus. (See figure 7-18.) The CPU microprogram obtains this status word from the IOP and checks to see that bit 0, the SIO OK bit, is true. This bit will be true if the device is ready and the Selector Channel is inactive. Assuming that the SIO OK bit is true, the CPU microprogram outputs an SIO command to the IOP Control Register and the IOP issues the SIO command to the Device Controller. The DEVNO on the bus is again compared with the internally wired device number (figure 7-16) and the true result, with SO, enables the I/O Command (IOCMD) to be decoded. The IOCMD is now SIO which, when decoded, issues a Request (REQ) signal to the Selector Channel control logic. The channel then returns SI to the IOP as an acknowledgement response. From now on (except for processing an interrupt), the IOP is not involved. The data gating logic routes all data transmissions to the DATA lines of the Selector
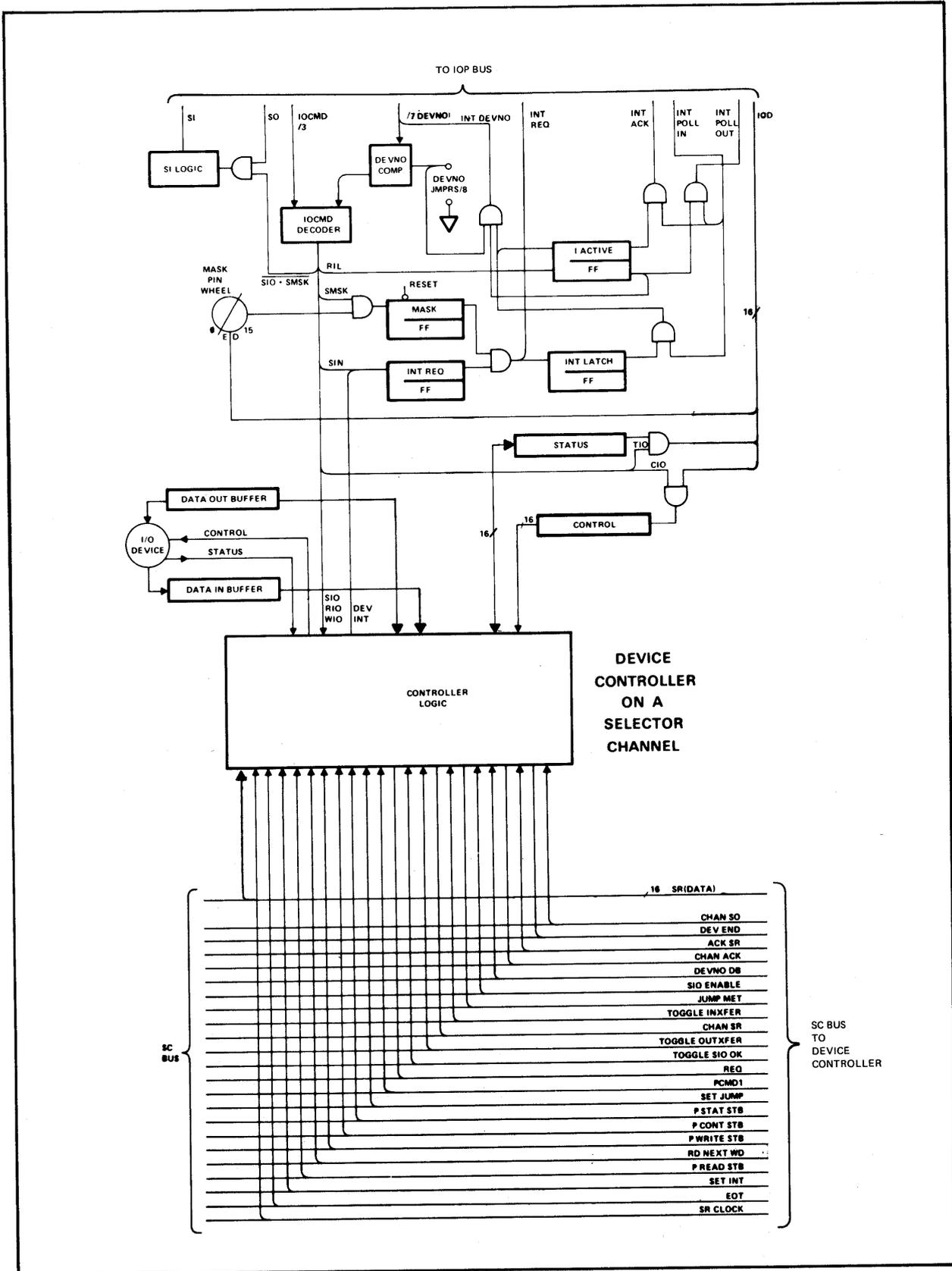
Figure 7-18. Selector Channel and Device Controller Simplified
              Logic Diagram

Channel Bus rather than to the IOD lines of the IOP Bus.

When the Selector Channel (figure 7-19) receives a Request (REQ) from the Device Controller, it sets the control logic to "active". The Selector Channel then issues the Device Number Data Base (DEVNO DB) to the Device Controller. The Device Controller gates the DEVNO, left shifted by two, onto the SR (Data) lines of the Selector Channel Bus. The DRTE address is then loaded into the DEVNO DBV Register. The Selector Channel is now exclusively reserved for that device. Only this Device Controller will respond to Channel Service Out (CHANSO) from the Selector Channel. The Selector Channel now reads the device number from the DEVNO DB Register and requests a memory transfer by issuing a Read to the Port Controller (figure 7-17). The Port Controller checks if memory is ready and, when Enable (ENB) is present, sets the LSEL flip-flop. The LSEL signal is returned to the Selector Channel (figure 7-19), where it reads the DRTE address onto the PCD lines on the PC Bus. LSEL also reads out the TO, FROM, and MOP codes in the Port Controller, thus effecting an address transmission to memory.

When memory returns the DRT contents, the Port Controller issues STRB to the Selector Channel. Since the Selector Channel control logic is expecting a DRT word, it loads the bus data into the I/O Program Counter. The contents of the I/O Program Counter will hereafter be used to address the individual locations of the I/O program and no further DRT fetches are necessary. Program execution will occur as a result of fetch and execute sequences.

7-50. FETCH SEQUENCE. Fetching an I/O program doubleword requires two memory fetches. Unlike the Multiplexer Channel which examines the IOCW to determine what to do about the IOAW (fetch it, store into it, or gate it out to the device controller) the two memory fetches always occur. The different operations for the various types of I/O orders are accomplished in the execute sequence. The fetch sequence begins with the Selector Channel reading out the contents of the I/O Program Counter and requesting a memory read. When the Port Controller has obtained transmit priority, it returns LSEL, transmitting the I/O Program Counter contents to memory as an address. (The Counter is incremented immediately.)

When memory returns the IOCW from the addressed location, the Port Controller issues STRB to the Selector Channel. The Selector Channel contorl logic, which is expecting the IOCW, loads the word into the IOCW Active Register. Then the I/O Program Counter is again read out with another memory transfer request. The Port Controller transmits this address to memory and the I/O Program Counter is again incremented. Then, when memory returns the IOAW from the addressed location, the Selector Channel loads the word into the IOAW Active Register. At this point the fetch sequence is complete.
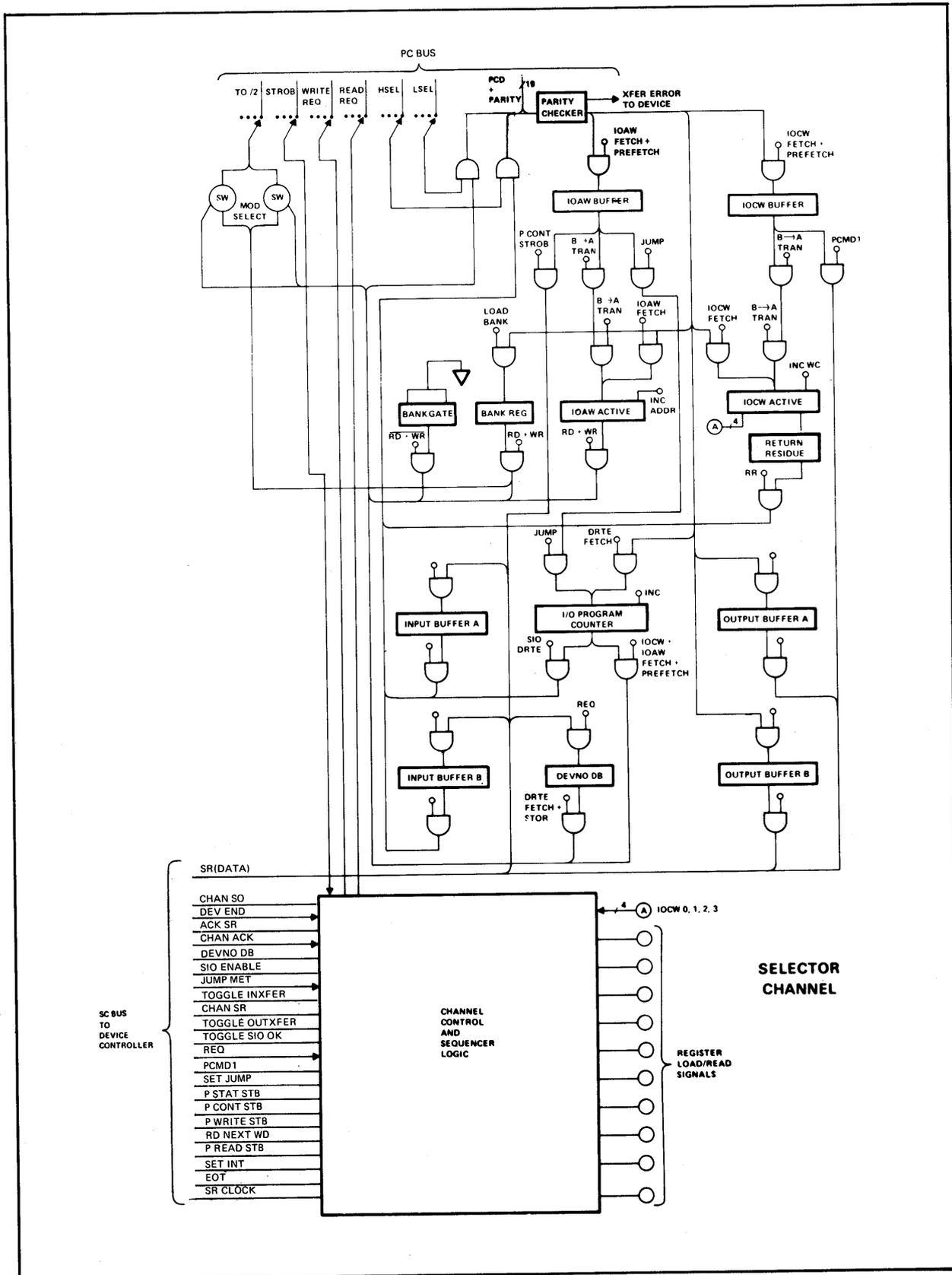
Figure 7-19. Selector Channel Simplified Logic Diagram

The Selector Channel control logic can now examine the order. If the order specified in the IOCW is Read or Write and, if data chaining is also specified, a pre-fetch sequence is enabled. This operation is the same as the fetch sequence described in the preceding two paragraphs except that the returned data is loaded into the IOCW Buffer and IOAW Buffer instead of the IOCW and IOAW Active Registers. An additional condition for the pre-fetch sequence is that data transfer take precedence; i.e., pre-fetch will occur only when both Input Buffers A and B are empty (for Read) or both Output Buffers A and B are full (for Write). Then, when the Read or Write order finishes, due either to word count rollover or to a device end condition (see Read and Write execute sequences), the IOCW/IOAW Buffers are read into the IOCW/IOAW Active Registers. The data transfer can thus continue uninterrupted. If the new IOCW specifies further data chaining, another pre-fetch is initiated to refill the buffers.

7-51. EXECUTE SEQUENCES. The following paragraphs contain separate descriptions of the execute sequence for each of the nine I/O orders. In each case except End, which terminates the I/O program, operation returns to the fetch sequence following completion in order to fetch the next I/O program word.

7-52. Sense. The Selector Channel issues a P STATUS STB signal to the Device Controller, with CHANSO, via the Selector Channel Bus. The Device Controller accordingly reads the contents of its Status Register onto the channel DATA lines and returns CHAN ACK (Channel Acknowledge). On receipt of CHAN ACK, the Selector Channel loads the status information into one of the two input buffers and prepares for a memory transfer. First the contents of the I/O Program Counter are decremented by one. This is necessary because the status word must be stored in the IOAW location for the current order, whereas the fetch sequence has incremented the I/O Program Counter to point at the next word. Once this is done, the contents of the I/O Program Counter and the input buffer containing the status word are read out to the channel PCD gates (but not gated out yet). The bank number becomes the TO address. A number is either loaded into the Bank Register or Bank 0 is picked up at the Bank Gate (figure 7-19), gated through the MOD select swiches, and sent as the TO address to the Port Controller via the TO lines of the Port Controller Bus. A Write request to the Port Controller requests a transmission to memory and, when the Port Controller returns LSEL, the address from the I/O Program Counter is sent to memory and the Counter is incremented. An HSEL from the Port Controller (which follows immediately unless ENB has been preempted by a higher-priority module) then reads out the status word to the PCD lines and sends it to memory. This stores Status in the IOAW location.

7-53. Interrupt. The Selector Channel control logic issues a P SET INT signal to the Device Controller, with CHANSO, via the Selector Channel Bus. The Device Controller returns CHAN ACK and sets its Interrupt Request flip-flop. Provided the Mask flip-flop is set, the Device Controller issues INT REQ to the IOP via the IOP Bus. When the IOP returns INT POLL, the device number is

sent to the IOP along with INT ACK. On receipt of INT ACK, the IOP generates an Interrupt signal to the CPU.

7-54. Jump. The Jump order may be specified to be either conditional or unconditional. It is the function of an uncondi- tioned jump or a successful conditional jump to transfer the con- tents of the IOAW Buffer (the jump address) to the I/O Program Counter. (The IOAW Buffer and IOAW Active Register contain iden- tical contents at this time.) In the case of a conditional jump order, the Selector Channel issues a Set Jump command to the De- vice Controller, with CHANSO, via the Selector Channel Bus. The Device Controller returns a true or false Jump Met signal. If the jump is not met, operation returns to the fetch sequence. If the jump is met for an Unconditional Jump order, the channel con- trol logic gates the contents of the IOAW Active Register into the I/O Program Counter. Thus, subsequent orders will be fetched and executed from a new I/O program area.

7-55. Control. The Control order routes both the IOCW and the IOAW to the Device Controller. The Selector Channel first reads out the contents of the IOCW Active Register to the channel DATA lines and issues a PCMDl (Programmed Command One) signal, with CHANSO, for the Device Controller to load the data word. The Device Controller accordingly loads the word into its Control Register and then issues a request (CHAN SR) back to the Selector Channel to send the second word. The Selector Channel reads out the contents of the IOAW Active Register to the DATA lines and issues a second command (P CONT STB), with CHANSO, for the Device Controller to load this new word. When the Device Controller has loaded the new word and is ready for the next order, it returns the appropriate response (another CHAN SR) signal to the Selector Channel.

7-56. Set Bank. When requesting a memory Read or Write (for data words only), an IOAW word goes into the Selector Channel (figure 7-19) on the PCD lines and the four least significant bits are loaded into the Bank Register by the Set Bank order. Two bits from the Bank Register (TOl-1 and TOl-2) are gated through the MOD select switches and Port Controller to become the memory module TO signal on the CTL Bus. The remaining two bits from the Bank Register (PB14 and PB15) are applied back through the Port Controller via the PC Bus to become part of the memory module 18-bit address (see figures 6-2 and 7-17 through 7-19) on the CTL Bus.

7-57. Read. The Read order causes a block of data to be trans- ferred from the device to memory. The block size in words is specified in two's complement form by the word count (IOCW bits 4 through 15) and the absolute starting address in memory is speci- fied by the IOAW. While the block transfer is in progress, there are two separate, simultaneous operations taking place; the device-to-channel transfer and the channel-to-memory transfer. To begin the Read execute sequence, the Selector Channel issues CHANSO to the Device Controller. (See figure 7-18 and 7-19.) When the controller returns CHAN ACK, the Selector Channel issues the

initial Read Next Word (RD NXT WD) with CHANSO still asserted. When CHANSO is removed, both the Selector Channel and the controller are set to the in-transfer condition to enable data transfers.

After the device has read a word and the controller is ready to transfer it to the channel, it sends Channel Service Request (CHAN SR) to the channel. The channel issues P READ STB and CHANSO, causing the Device Controller to read its Data In Buffer onto the channel Data lines and to return CHAN ACK. On receiving CHAN ACK, the Selector Channel loads the data into either Input Buffer A or Input Buffer B (depending on which is empty), increments the word count in the IOCW Active Register, and re-issues RD NEXT WD. The above transfer sequence repeats for each data word until the Device Controller asserts DEV END to terminate the block, or until the word count rolls over. In either case, the channel sends End of Transfer (EOT) to the controller and, if not data chaining, clears the in-transfer condition. A CHAN SR from the controller is required to resume program execution.

Meanwhile, the Selector Channel attempts to keep both Input Buffers empty by transmitting their contents to memory. The control logic for the A and B Buffers ensures tha data is transmitted to memory in the same sequence as received from the device. To accomplish a memory transfer, the Selector Channel enables the IOAW Active Register for use as a memory address, enables Input Buffer A or B for use as a data word, and sends a Write Request and a mapped TO code to the Port Controller. When the Port Controller (figure 7-17) returns LSEL, the IOAW is gated onto the bus as an address to memory and the IOAW is incremented to point to the next data location. When the Port Controller returns HSEL, the Input Buffer is gated onto the bus to be stored in the addressed memory location. The preceding operation in this paragraph repeats until the Read order completes, via a DEV END or word count rollover, and all input data has been sent to memory.

If the data chaining bit in the IOCW Active Register is true, the next order pair will have been prefetched when possible during the block data transfer. When the Read order completes, the prefetched order pair will be transferred from the IOCW/IOAW buffers to the Active Registers without the need for a normal fetch sequence. Data input can thus continue for the next block with minimum interruption. If the data chaining bit is not set, the read termination will be followed by a normal fetch sequence.

7-58. Return Residue. The function of the Return Residue order is to send the current contents of the Residue Register (which reflects the results of the most recent Read or Write order) to the IOAW location of the current I/O program word. The Device Controller is not involved. To begin the procedure, the channel control logic decrements the I/O Program Counter (for the same reason described in the preceding paragraph). The contents of the I/O Program Counter and the Residue Register are then read out to the PCD gates, while a Write Request and a mapped TO code are issued to the Port Controller. When the Port Controller re-

turns LSEL, the address from the I/O Program Counter is sent to memory. When HREQ sets the HSEL flip-flop, the word count from the Residue Register is sent to memory. This stores the residue in the IOAW location.

7-59. Write. The Write order causes a block of data to be transferred form memory to the device. The block size in words is specified in two's complement form by the word count (IOCW bits 4 through 15) and the absolute starting address of the block in memory is specified by the IOAW. While the block transfer in progress, there are two separate, simultaneous operations taking place; the memory-to-channel transfer and the channel-to-device transfer. To begin the Write execute sequence, the Selector Channel issues CHANSO to the controller and, when the controller returns CHAN ACK, both the Selector Channel and the controller are set to the out-transfer condition to enable data transfers.

Meanwhile, the Selector Channel proceeds with a memory fetch and will attempt to keep both Output Buffers full. The control logic for the A and B Output Buffers ensures that data is transmitted to the device in the same sequence as it was fetched from memory. To accomplish a memory fetch, the Selector Channel enables the IOAW Active Register for use as a memory address and sends a Read Request and the Bank Register as a TO address to the Port Controller. When the port returns LSEL, the IOAW is gated onto the bus as an address to memory and the IOAW is incremented to point to the next data location. When the port returns STRB, the data on the bus from memory is loaded into an empty Output Buffer. The preceding operation in this paragraph repeats until the Write order completes by either a DEV END or word count rollover.

When the controller is ready to accept a data word from the channel, it sends CHAN SR. The channel issues CHANSO and P WRITE STB and gates Output Buffer A or B onto the channel Data lines. The controller returns CHAN ACK causing the channel to remove P WRITE STB, increment the word count, and remove CHANSO in that order. The Device Controller uses the removal of P WRITE STB to latch the data word from the channel Data lines. The previous transfer sequence in this paragraph repeats for each data word sent to the Device Controller until the Device Controller asserts DEV END to prematurely terminate the block or until the word count rolls over. In either case, the Selector Channel sends EOT (End of Transfer) to the controller and, if not data chaining, clears the out-transfer condition. To resume program execution, a new CHAN SR from the controller is required by the Selector Channel.

If the data chaining bit is true, (IOCW bit 0) the next order pair will have been prefetched when possible during the block transfer. When the Write order completes, the pre-fetched order pair will be transferred from the IOCW/IOAW buffers to the Active Registers without the need for a normal fetch sequence. Data output to the controller can thus continue for the next block with minimum interruption. If the data chaining bit is not set, termination of the Write order will be followed by a normal fetch sequence.

7-60. End. The execute sequence for the End order begins by duplicating the operations of a Sense order, obtaining the controller's status word and storing it in the IOAW location in the I/O program. Additionally, if IOCW bit 4 is true, a P SET INT signal is also issued to the controller. (Refer to Interrupt order description.) Then the channel proceeds to store the contents of its I/O Program Counter into the device's DRT location. As previously discussed, this is to maintain compatibility with I/O programs run via a Multiplexer Channel. The Selector Channel enables its DEVNO DB Register, enables the I/O Program Counter for use as data, and sends a Write Request and a TO=0 to the Port Controller. When the port returns LSEL, the shifted device number is gated out as the DRT address and, when the port returns HSEL, the I/O Program Counter content is gated out to the bus as data. This completes all operations for the I/O program. The channel control logic resets to the inactive condition, thus allowing another program for the same or another device to be initiated via that channel.

7-61. I/O SYSTEM SERVICING INFORMATION

The following paragraphs contain servicing information for the IOP, Multiplexer Channel, Port Controller, and Selector Channel PCA's.

7-62. IOP PCA Servicing

The IOP PCA is a nonrepairable PCA and must be replaced if found defective. No repair procedures are required. However, the IOP PCA does contain a jumper and three switches (figure 7-20) that must be properly configured as discussed in paragraphs 7-63 through 7-65.

7-63. ENABLE/DISABLE. Jumper W1 (figure 7-20) is used to enable or disable the IOP PCA. Installation of this jumper disables the IOP PCA.

7-64. MEMORY SIZE. Switch S3 (figure 7-20) is a 6-position switch used to select memory word size. The switch positions and corresponding memory word sizes are shown in figure 7-20.

7-65. MEMORY INTERLEAVING. Two switches, S1 and S2 (figure 7-20), are used for memory interleaving. At present, S1 and S2 must be configured for non-interleaving in accordance with table 2-8.

7-66. Selector Channel Maintenance Board PCA

The Selector Channel Maintenance Board PCA was designed to aid in servicing the Selector Channel and Multiplexer Channel. Under software control, this PCA can exercise all Selector Channel data paths and control circuitry. All I/O program orders can be executed and device dependent sequences such as conditional jump,

```
              TOP 1, 3,                    79        1, 3,              79
           BOTTOM 2, 4,                    80        2, 4,              80
           P1

                              W1




                            IOP PCA




                                                          S3




                                              S1    S2
              J1              J2              J3
           TOP 2, 4,         50    2, 4,         50    2, 4,         50
        BOTTOM 1, 3,         49    1, 3,         49    1, 3,         49


            MEMORY SIZE

            (WORDS)
               S3

            1 - 128K
            2 - 256K
            3 - 384K
            4 - 512K
            5 - 768K
            6 - 1024K
```
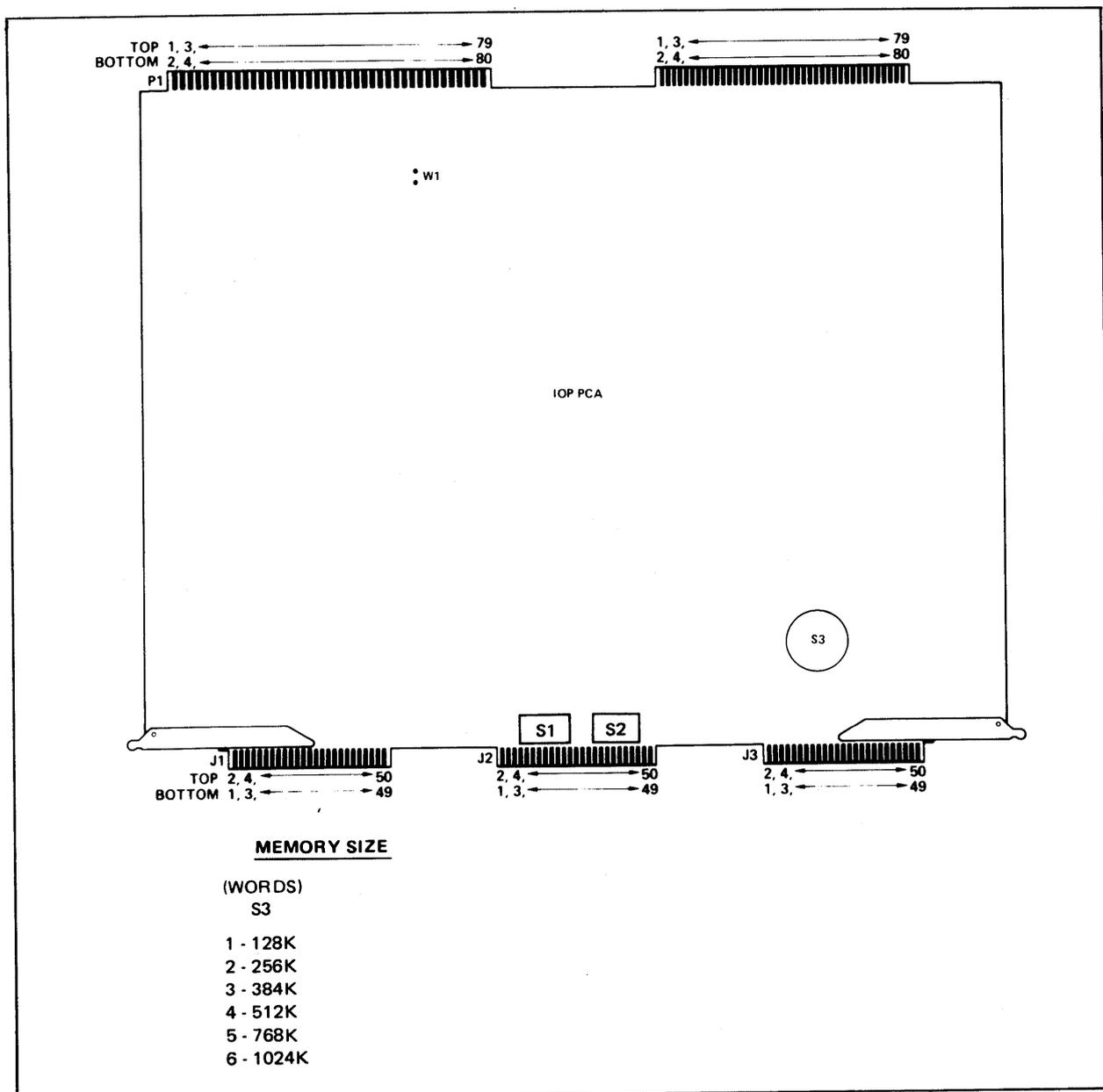
202   Figure 7-20.   IOP PCA Jumper and Switch Locations

device end, and clear interface can be exercised selectively. Also, device timeout conditions can be simulated causing a timeout error in the Selector Channel. Complete information on how to install and use the Selector Channel Maintenance Board is contained in the HP 3000 Series III CE Handbook, part no. 30000-90172.

7-67. Multiplexer Channel PCA Servicing

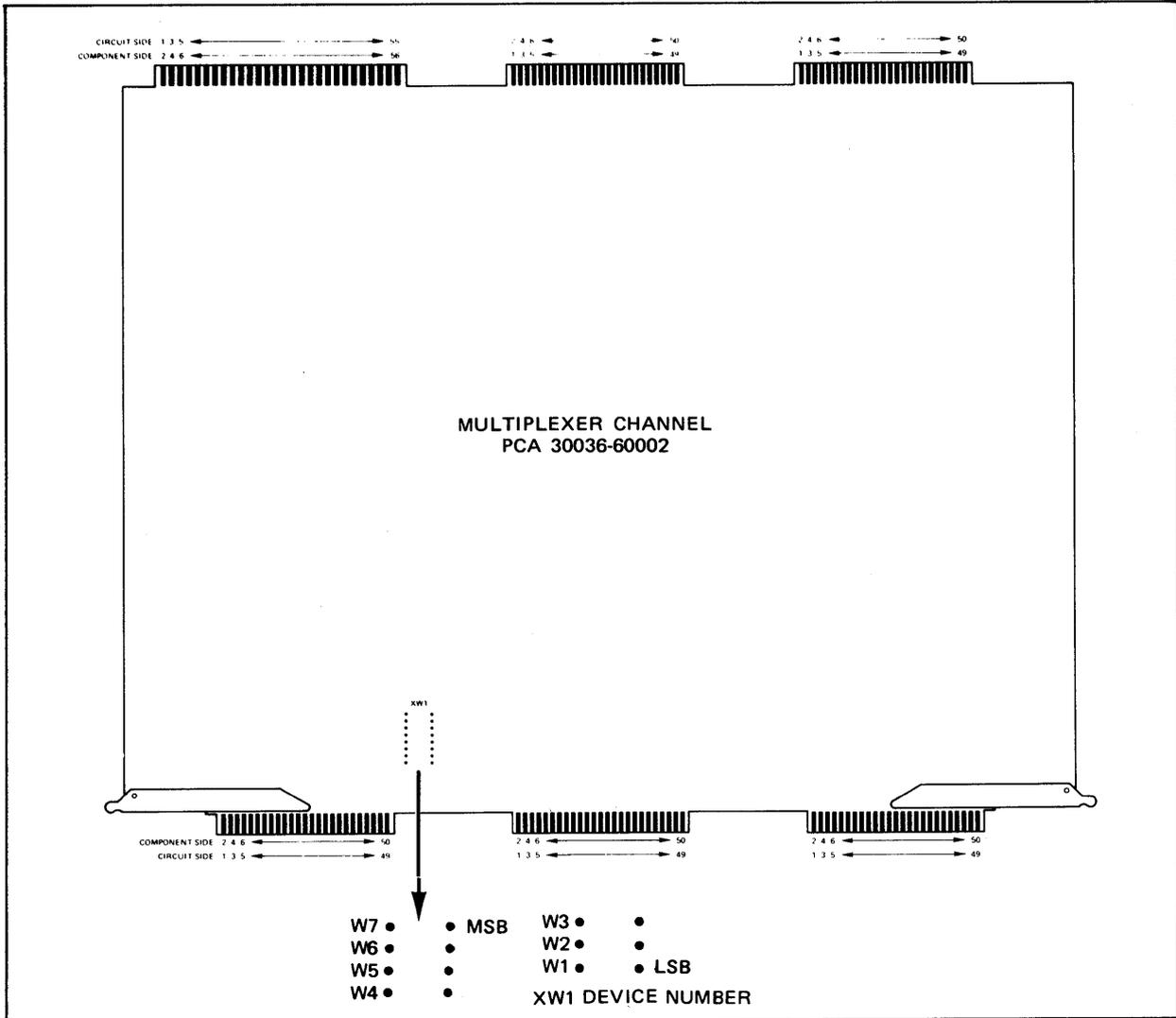The Multiplexer Channel PCA is a nonrepairable PCA and must be replaced if found defective. No repair procedures are required. However, the Multiplexer Channel PCA does contain jumpers that must be properly configured. The configuration of jumpers W1

through W7 (figure 7-21) determine the device number and, therefore, the DRT address associated with the Multiplexer Channel PCA. A logic "1" is represented by the absence of a jumper and, conversely, a logic "0" is represented when a jumper is installed. The PCA's device number is normally %77 (no jumpers).

7-68. Port Controller PCA Servicing

The Port Controller PCA is a nonrepairable PCA and must be replaced if found defective. No repair or servicing procedures are required.

7-69. Selector Channel Servicing

The Selector Channel consists of three PCA's; a Selector Channel Register PCA, a Selector Channel Control PCA, and a Selector Channel Sequencer PCA. Servicing information for the three PCA's are contained in paragraphs 7-70 through 7-75.

7-70. SELECTOR CHANNEL REGISTER PCA. The Selector Channel Register PCA is a nonrepairable PCA and must be replaced if found defective. However, the PCA does contain jumpers and switches (figure 7-22) that must be properly configured as discussed in paragraphs 7-71 through 7-73.

7-71. Port Controller Channel Number. Jumper connectors XW1 through XW4 (figure 7-22) are used for the selection of one of the three available channel ports from the Port Controller. Channel selection is made by installing a jumper in the jumper connector corresponding to the desired channel.

7-72. Memory Size. Switch S3 (figure 7-22) is a 6-position switch used to select memory word size. The switch positions and corresponding memory word sizes are shown in figure 7-22.

7-73. Memory Interleaving. Two switches, S2 and S3 (figure 7-22), are used for memory interleaving and, at present, must be configured for non-interleaving. (Refer to table 2-8.)

7-74. SELECTOR CHANNEL CONTROL PCA. The Selector Channel Control PCA is a nonrepairable PCA that must be replaced if found defective. No repair procedures are required. However, the PCA does contain the Error Logging Register, a test switch and indicators used for troubleshooting. Refer to the HP 3000 Series III CE Handbook, part no. 30000-90172 for complete information on how to use this PCA as a troubleshooting aid.

7-75. SELECTOR CHANNEL SEQUENCER PCA. The Selector Channel Sequencer PCA is a nonrepairable PCA and must be replaced if found defective. No repair or servicing procedures are required.

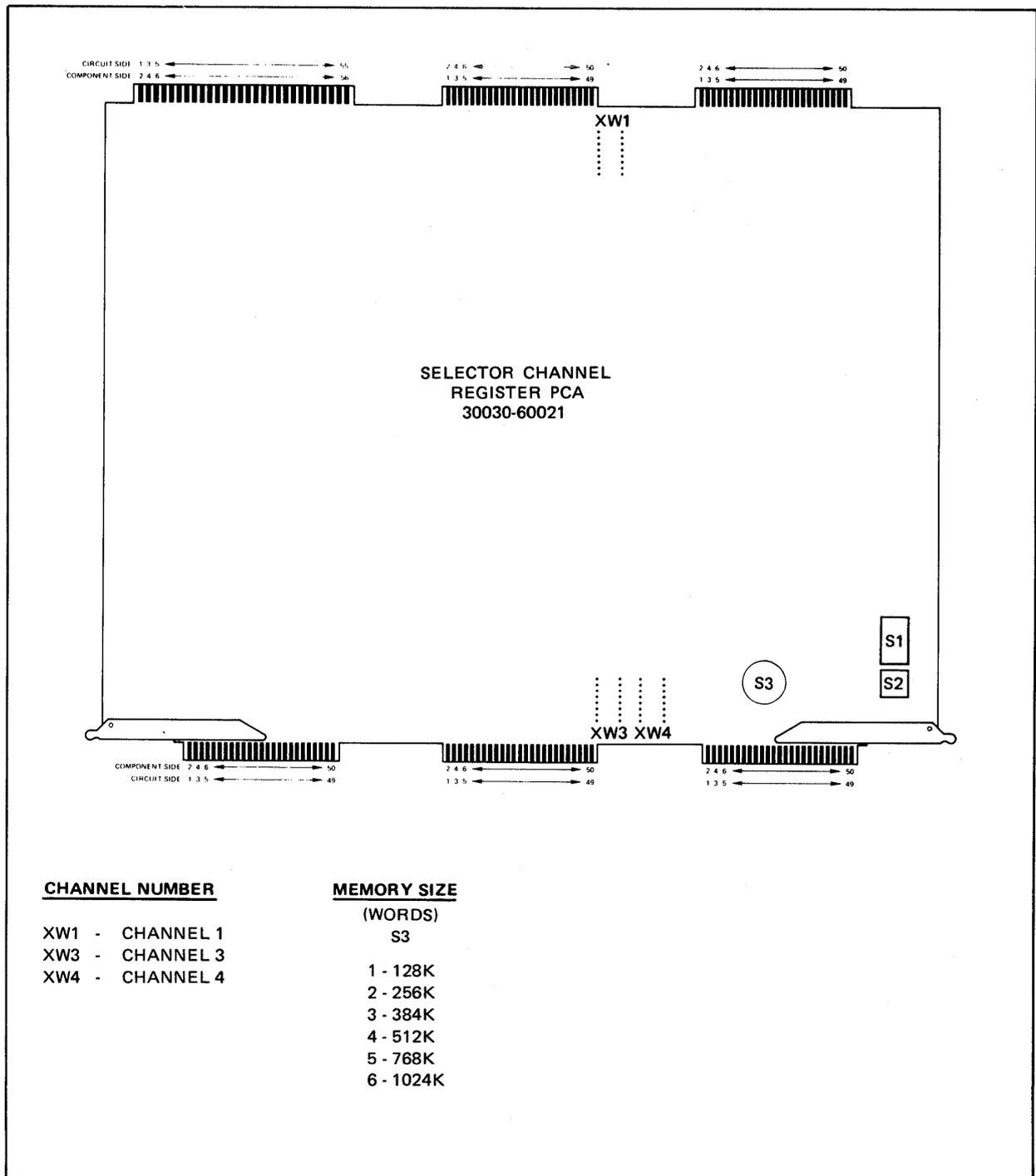Figure 7-21. Multiplexer Channel PCA Jumper Locations

CIRCUIT SIDE 1 3 5 ◄——————————————→ 55   2 4 6 ◄— ——→ 50   2 4 6 ◄——————————→ 50
COMPONENT SIDE 2 4 6 ◄——————————————→ 56   1 3 5 ◄——————→ 49   1 3 5 ◄——————————→ 49

XW1

SELECTOR CHANNEL
REGISTER PCA
30030-60021

S1

S3   S2

XW3 XW4

COMPONENT SIDE 2 4 6 ◄——————————→ 50   2 4 6 ◄——————→ 50   2 4 6 ◄——————→ 50
CIRCUIT SIDE 1 3 5 ◄—— ————→ 49   1 3 5 ◄——————→ 49   1 3 5 ◄——————→ 49

CHANNEL NUMBER	**MEMORY SIZE**
	(WORDS)
XW1 - CHANNEL 1	S3
XW3 - CHANNEL 3	1 - 128K
XW4 - CHANNEL 4	2 - 256K
	3 - 384K
	4 - 512K
	5 - 768K
	6 - 1024K

Figure 7-22. Selector Channel Register PCA Jumper and Switch
Locations

NOTES

NOTES

This section contains principles of operation and servicing information for the computer´s interrupt system.

8-1. INTRODUCTION

The computer´s interrupt system provides up to 125 external levels. When interrupts occur, the microprogrammed interrupt handler identifies each interrupt and grants control to the highest priority interrupt. Current operational status is retained by the microprogram which then sets up the interrupt processing environment and transfers control to the interrupt routine.

Interrupt routines operate on a common stack called the Interrupt Control Stack (ICS) which is known to both software and hardware. This feature permits nesting of interrupt routines in the case of multiple interrupts, thus allowing higher priority devices to interrupt lower priority devices.

The interrupt system also provides for 17 internal interrupts (user errors, system violations, hardware faults, and power fail/ restart) plus seven traps for arithmetic errors and illegal use of instructions.

8-2. INTERRUPT SYSTEM OVERVIEW

The interrupt system´s interrupt routines are called and exited in a manner resembling the way that procedures are called and exited. An interrupt is therefore an implicit PCAL instruction (vs. an explicit PCAL instruction). (Refer to Section IV.) Also, code and data domains are kept separate. The primary differences are that the calling operations are performed by a microprogrammed Interrupt Handler rather than the PCAL instruction and, in some cases, the IXIT (Interrupt Exit) instruction is used for exiting the interrupt code instead of EXIT. Internal interrupt procedures are contained in code segment number 1. Interrupt procedures for I/O devices may be in any code segment other than segment number 1. Table 8-1 lists the internal interrupts and traps with their corresponding entry numbers in the Segment Transfer Table (STT) of the internal interrupt code segment. The parameter is a value that is derived by the Interrupt Handler and which passes relevant information about the interrupt to the interrupt routine. The Device Reference Table (DRT) contains a label for each entry, pointing to the interrupt procedure for each device. Bit 8 of the CPX1 Register indicates an external interrupt. The parameter value for an external interrupt is the device number.

Table 8-1.   Interrupt Types

Ext Prog. Label (%)	STT No. (%)	Interrupt Type	Parameter*	Executing Stack**
100401	1	Bounds Violation		
101001	2	Illegal Memory Address		
101401	3	Non-Responding Module		
102001	4	System Parity Error		ICS
102401	5	Address Parity Error		ICS
103001	6	Data Parity Error		ICS
103401	7	Module Interrupt	Module No.	ICS
104001	10	(Unused)		
104401	11	Power Fail		ICS
105001	12	(Unused)		
105401	13	(Unused)		
106001	14	(Unused)		
106401	15	(Unused)		
107001	16	(Unused)		
107401	17	(Unused)		
110001	20	Unimplemented Instruction		
110401	21	STT Violation		
111001	22	CST Violation		
111401	23	DST Violation		
112001	24	Stack Underflow		
112401	25	Privileged Mode Violation		
113001	26	(Unused)		
113401	27	(Unused)		
114001	30	Stack Overflow		ICS
114401	31	User Traps		
		a. Integer Overflow	%000001	
		b. Floating-Point Over	%000002	
		c. Floating-Point Under	%000003	
		d. Integer Divide by 0	%000004	
		e. Floating-Point Divide by 0	%000005	
		f. Ext. Prec. Floating-Point Overflow	%000010	
		g. Ext. Prec. Floating-Point Underflow	%000011	
		h. Ext. Prec. Floating-Point Divide by 0	%000012	
		i. Decimal Overflow	%000013	
		j. Invalid ASCII Digit	%000014	
		k. Invalid Dec. Digit	%000015	
		l. Invalid Source Word Count	%000016	
		m. Result Word Count	%000017	
		n. Decimal Divide by 0	%000020	

Table 8-1. Interrupt Types (Continued)

Ext Prog. Label (%)	STT No. (%)	Interrupt Type	Parameter*	Executing Stack**
115001	32	(Unused)		
115401	33	(Unused)		
116001	34	(Unused)		
116401	35	(Unused)		
117001	36	(Unused)		
117401	37	Absent Code Segment		
		a. On PCAL	P-Label	
		b. On EXIT	N	
		c. On IXIT	0	
120001	40	Trace		
		a. On PCAL	P-Label	
		b. On EXIT	N	
		c. On IXIT	0	
120401	41	STT Entry Uncallable	P-Label	
121001	42	Absent Data Segment	DST No.	
121401	43	Power On		ICS
122001	44	Cold Load		ICS
		a. System I/O (SIO)	0	
		b. Direct I/O (DIO)	Label	

*Unless noted, the parameter is the External Program Label.
**Unless noted, Interrupts are serviced on the User Stack.

All User Traps (STT No. %31) are enabled by the User Traps bit in the Status Register.

8-3. INTERRUPT CONTROL STACK

The Interrupt Control Stack (ICS) is a single stack, unique to the CPU, which is used in common by all external interrupts and some of the internal interrupts (ICS type). When only minimal data is to be handled by an interrupt routine, the data is processed on the ICS. Otherwise, the separate data area defined in the DRT must be used for data. The use of a common stack also permits efficient nesting of interrupt routines by using stack markers. The ICS has a permanent stack marker, set up by the operating system which is used to enter the Dispatcher. Figure 8-1 illustrates the format of the Dispatcher marker on the ICS.
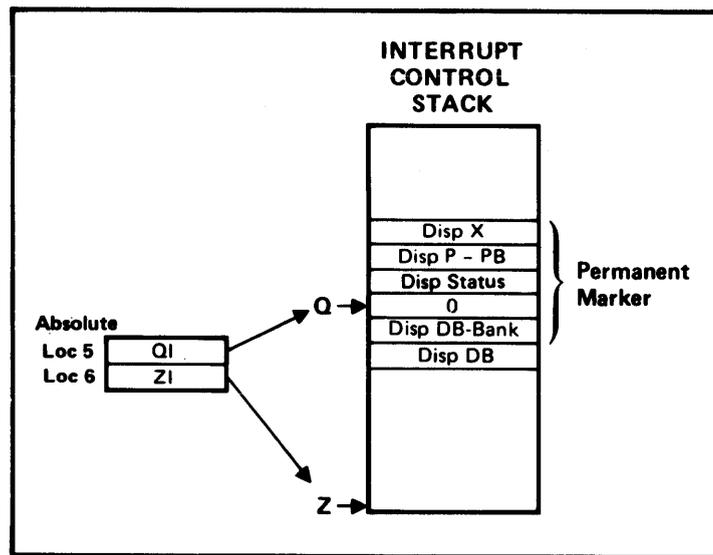
Figure 8-1.   ICS Dispatcher Marker

It should be noted that unlike the standard four-word stack
marker, the Dispatcher marker contains six words. As will be
explained later, all markers created because of an ICS-type
interrupt include two words to save the current value of DB and
DB-Bank. This information must be saved since all external inter-
rupts automatically alter DB (to the DBI value from word 2 of the
DRT) and DB-Bank. Additionally, DB may be changed when process-
ing an ICStype internal interrupt. The Delta Q location of the
Dispatcher marker always contains a "0" word in bits 1-15 (Bit   0
can be set as described later). The value is zero because there
is no previous marker on the ICS. The Dispatcher Flag is set
whenever the Dispatcher is entered and remains set while the Dis-
patcher is executing. It is cleared when the Dispatcher com-
pletes its execution, or is interrupted.

The segment-number field of the status word (Disp Status) in the
Dispatcher marker permanently points to the CST entry for the
Dispatcher, and the P-PB (Disp P-PB) word permanently points to
the starting point in the Dispatcher code segment. The DISP (Dis-
patch) instruction uses these values for transferring control to
the Dispatcher.   The locations preceding the Dispatcher marker
comprise the ICS global area which contains operating system in-
formation. It should be noted that since ICS-type interrupts use
a six-word marker, the parameter is found in location Q+3, rather
than the usual Q+1 location.   A hardware ICS Flag is set in the
CPX1 Register whenever a switch is made to the ICS from any other
stack.   The ICS Flag remains set until the ICS is no longer the
current stack.

Figure 8-1 also shows the delimiting of the ICS by QI and XI
("interrupt" Q and Z). These values are given fixed memory loca-
tions 5 and 6. The QI value points to the Delta-Q location of
the Dispatcher marker on the ICS. The ZI value points to the ICS
stack limit.

8-4. INTERRUPT TYPES

Interrupts may be divided into three basic types; external interrupts which are signals from the I/O system, and two types of internal interrupts which typically are unexpected conditions resulting from program execution. The three interrupt types are external interrupts (from I/O devices), ICS-type internal interrupts (using the Interrupt Control Stack), and non-ICS internal interrupts. A comparison of the overall operations of all three interrupt types is illustrated in figure 8-2. Note that operations proceed mostly left-to-right. For example, external interrupts begin by triggering some actions in hadware, then the interrupt processing environment is set up in software. The Dispatcher is the part of the operating system which schedules the execution of processes.
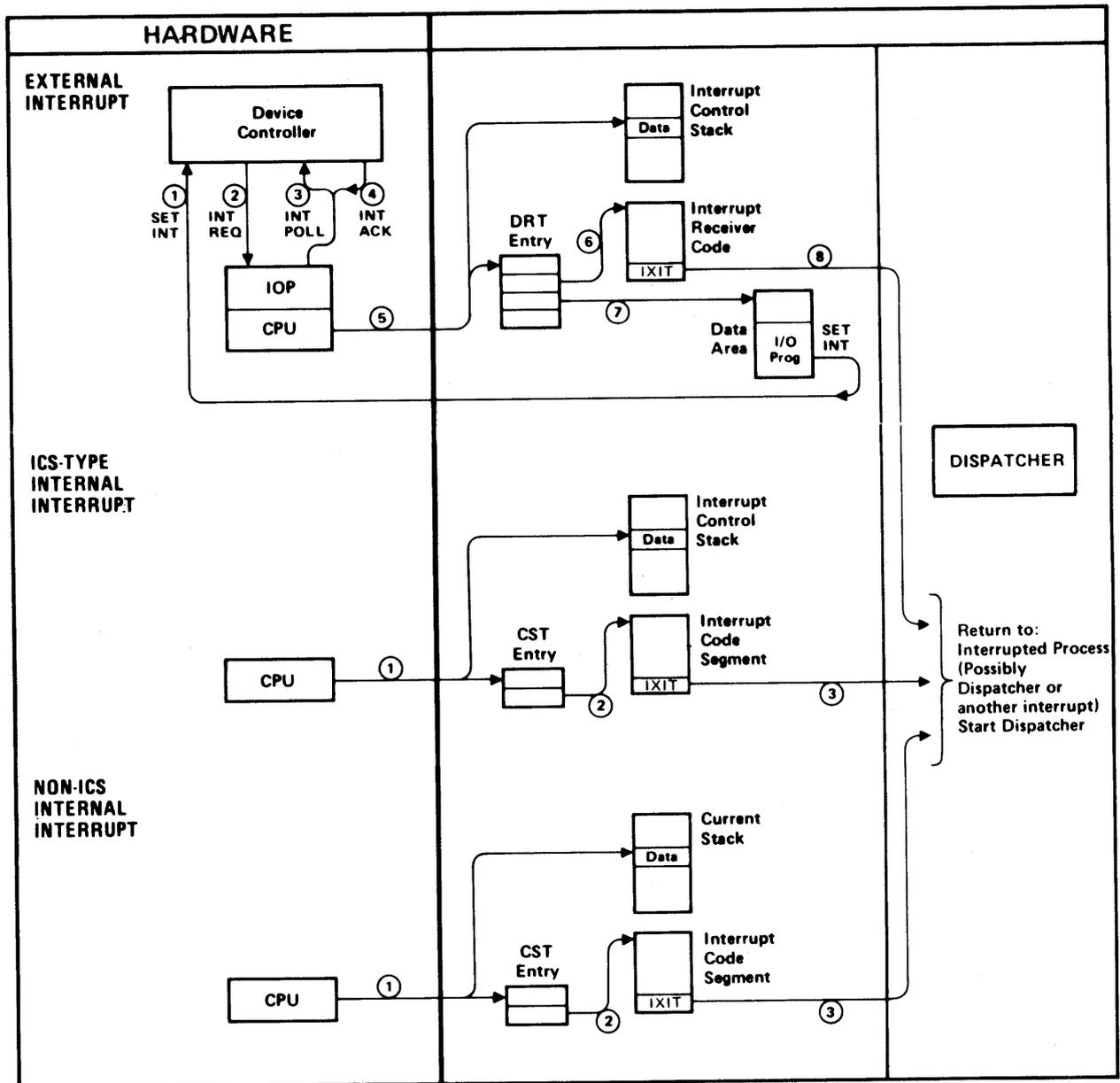
Figure 8-2. Interrupt System Overview

Note

It is assumed in this discussion that only one interrupt is being processed. As will be shown later, interrupt routines can be interrupted by other interrupts.

All external interrupt routines are entered with the external interrupt system enabled. All internal interrupt routines are entered with the external interrupt system disabled. The following paragraphs individually describe each of the three interrupt types. Only a brief introductory description is given at this point. Detailed operating sequences are discussed later in this section.

8-5. External Interrupts

External interrupts interface external events to software processes. Referring to figure 8-2 (top example), the overall operation is as follows:

a. The device controller sets the Interrupt flip-flop (1) by one of the following:

   (1) Set Interrupt (SIN) software instruction executed by the CPU telling the device controller to interrupt.

   (2) Set Interrupt (SET INT) command decoded by the device controller.

   (3) End with Interrupt (END,I) command decoded by the device controller.

   (4) The device controller detects an interruptable condition.

b. The setting of the Interrupt Request flip-flop causes the device controller to issue an Interrupt Request (INT REQ) signal to the IOP (2).

c. The IOP issues a poll (INT POLL) to activate the highest-priority request (there may be more than one request).

d. The device controller sets the Interrupt Active flip-flop, resets the Interrupt Request flip-flop, and sends the device number to the IOP.

e. The IOP examines the device number and, if the device number does not equal zero;

   (1) Puts the device number into an IOP register.

(2) Passes an interrupt signal to the CPU.

(3) Turns on the external interrupt flag (Bit 8 in the CPX1 Register).

(4) Drops the INT POLL signal.

If the device number equals zero, the IOP disregards the interrupt signal and drops the INT POLL signal.

f. The IOP requests the CPU to set up the interrupt environment (5). The initial steps are to set up the data segment registers to point at the Interrupt Control Stack (after saving the user's environment on his own stack) and to fetch the device's DRT entry.

g. The external program label in the second word of the DRT entry (6) is used to get the CST entry for the interrupt receiver code which, in turn, is used to set the PB-Bank, PB, and PL Registers. The starting address for the interrupt receiver code is obtained from the STT entry pointed to by the external program label and is loaded into the P Register, thus transferring control to the interrupt receiver code.

h. The information in the data area for this device (pointed to by the third word of the DRT) (7) is updated by the interrupt receiver. This information will tell the I/O monitor process that the initiator section of the device driver has done its word and that the completion section should be called.

i. The interrupt receiver code IXITS (8) normally returns control back to the interrupted process (which may be another interrupt or the Dispatcher). The interrupt receiver may also request a new dispatch by executing a DISP instruction. When an IXIT is executed by external interrupt code, a reset command is sent to the appropriate device.

8-6. ICS Internal Interrupts

ICS-type internal interrupts operate on the ICS and the interrupt code for each separate interrupt is permanently allocated in code segment 1. (Refer to table 8-1.) Referring to the second example in figure 8-2, the overall operation is as follows:

a. A condition such as power failure, stack overflow, or module interrupt causes the CPU to switch to the ICS (1) after saving the user's environment on his own stack by creating an External Program Label which points to a Segment Transfer Table entry in the internal interrupt segment (CST entry 1).

b. The PB and PL Registers are set up based on CST entry 1.

c. The Status Register is set to Privileged Mode, Segment 1 with all other bits cleared (%1000001).

d. The P Register is set from the local label, reached via the STT entry in the External Program Label, thus transferring control to the internal interrupt code segment (2).

8-7. Non-ICS Internal Interrupts

The non-ICS type interrupts operate on the current user's stack. Referring to figure 8-2, the overall operation is as follows:

a. A special condition is detected and causes the CPU to save the user's environment on his own stack and to fetch the CST entry 1 (1) by creating an External Program Label to the code for processing the interrupt.

b. The PB and PL Registers are set up based on CST entry 1.

c. The Status Register is set to Privileged Mode, Segment 1 with all other bits cleared (%100001).

d. The P Register is set from the local label, reached via the STT entry in the External Program Label, thus transferring control to the internal interrupt code segment (2).

8-8. EXTERNAL INTERRUPT PROCESSING

Prior to discussing the sequence of operations for external interrupts, there are two important factors that must be considered; interrupt priorities and interrupt program pointers. Servicing of external interrupts is accomplished in descending order of priority (i.e., the highest priority is serviced first). A higher priority interrupt can always interrupt the processing of a lower priority.

8-9. Interrupt Priorities

The interrupt priority of a device is completely independent of the device number. It is determined by the device's logical proximity to the IOP on the interrupt poll line. The interrupt poll is wired at system configuration time from one device controller to another using twisted-pair, clip-on wires. The routing of the interrupt poll is determined by the desired interrupt priorities of the device controllers and is completely independent of other parameters. Each device controller therefore, has a distinct priority level in relation to all other controllers. The maximum number of controllers and, hence interrupt levels, is 125.

8-10. Interrupt Program Pointer

The Device Reference Table (DRT) was defined in Section VII. As previously discussed, the second word of each DRT entry contains the interrupt program pointer. This is an external program label pointing to the start of the interrupt routine associated with a particular device controller. It should be noted that several controllers could point to the same routine.

8-11. Sequence of Operations

Figures 8-3 and 8-4 illustrate the sequence of operations for processing external interrupts. Basically, this discussion covers that portion of the overall I/O operation that establishes the interrupt processing environment on receipt of an external interrupt. In previous figures, this corresponds to steps (11) and (12) in figure 7-5, and to steps (5), (6), and (7) in figure 8-2.

Figure 8-3 illustrates how control is transferred from the point of interrupt in a user's code segment to the start of the interrupt receiver code. Also shown is the transfer of the data domain from the current user's stack to the interrupt control stack. Figure 8-4 illustrates how a second interrupt is handled and how exit is made from the interrupt routines. The following paragraphs describe the sequence of operations, step by step. It should be noted that all operations are under control of the hardware-implemented Interrupt Handler until control is transferred to the interrupt receiver code in software. Initially, it is assumed that the current process is operating at point P in some user's code when the CPU recognizes an external interrupt. The CPU thereupon passes control to the Interrupt Handler.

a. The first action of the Interrupt Handler is to push into memory any TOS elements of the current user's data that are in CPU registers (1, figure 8-3). This takes a maximum of four memory cycles if all four registers are full. Next, a normal four-word stack marker is pushed onto the user's stack followed by the value of the user's DB-Bank and the absolute value of DB that is currently in use. (DB may not necessarily point to a location within the user's stack, such as if a system intrinsic using a split stack had been called at the time of the interrupt.) This action preserves most of the user's environment; the current value of S will be preserved later in step f.

b. The S-Bank Register (2) is set to 0. (The ICS is always in Bank 0.)

c. The Interrupt Handler now goes to location 5 and loads the QI value into the Q Register (3). This points at the Delta Q location of the permanent Dispatcher marker. (As explained previously, this location contains a value of 0.)

d. The contents of location 6 are fetched and the value of ZI is loaded into the Z Register (4). This establishes the stack limit for the ICS. (The ICS Flag in the CPX1 Register is also set.)

e. The DL Register (5) is set to the limit value of %177777.

f. The user's value of S relative to Stack DB (at QI-4) is calculated and stored in QI-6 (6).
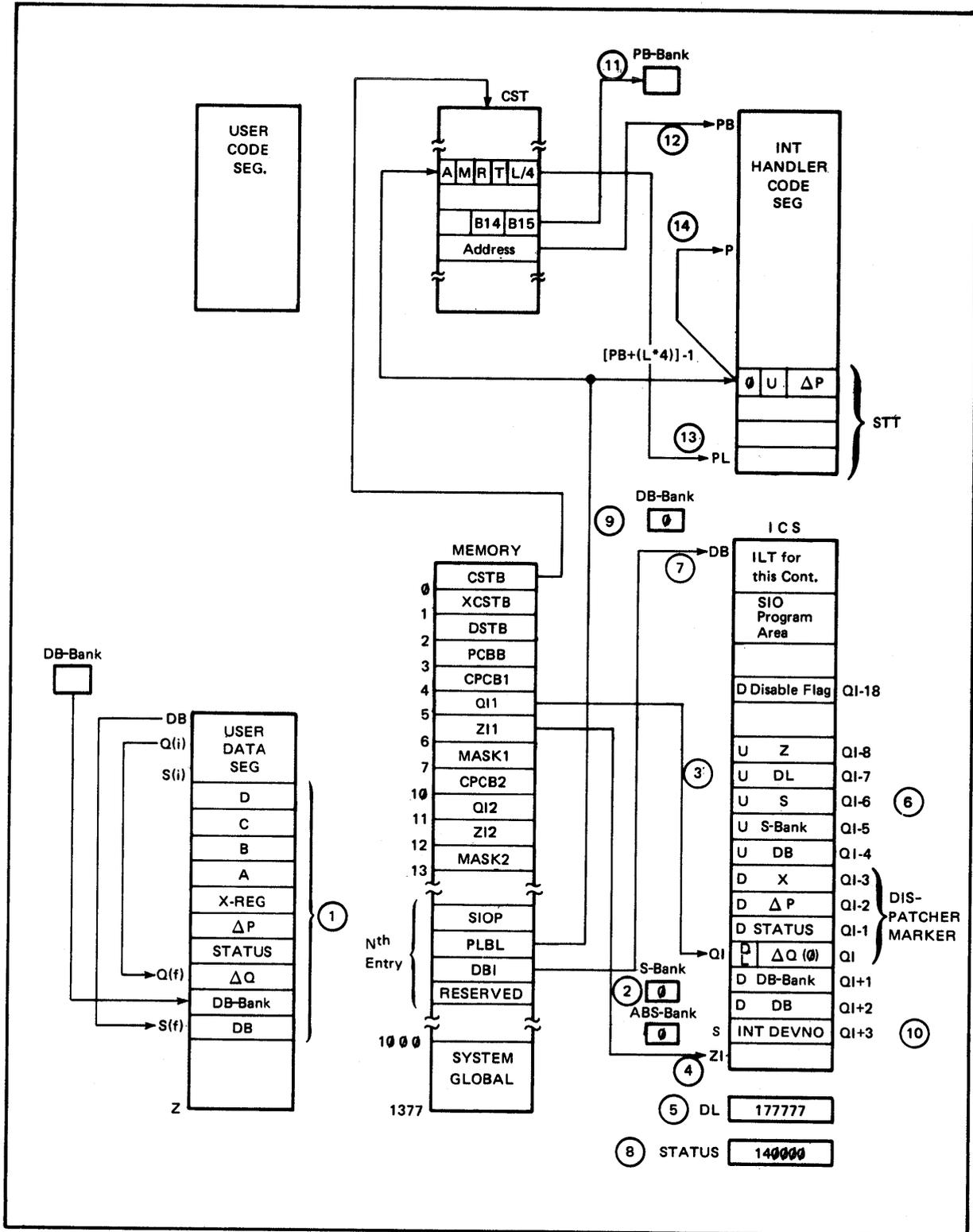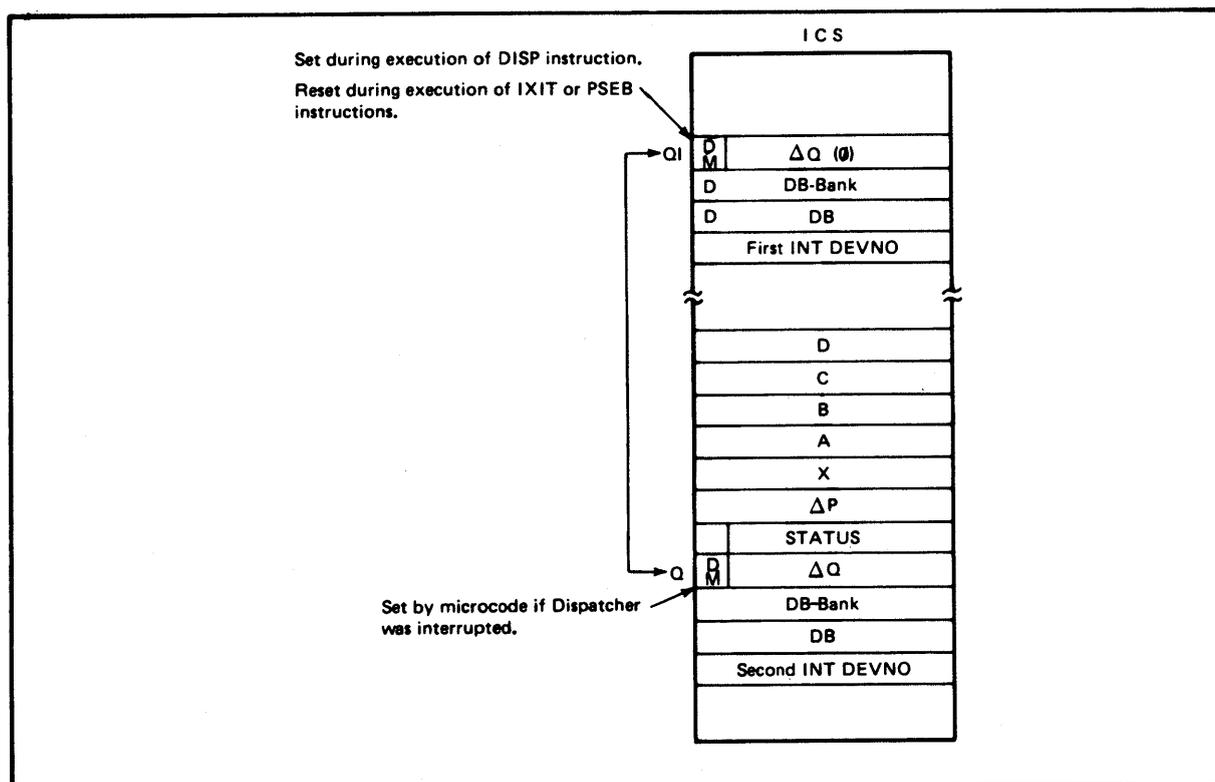
Figure 8-3. First Level External Interrupt

Figure 8-4.   Second Level Interrupt or Dispatcher Interrupted

g.   The CPU obtains the device number from the Interrupt Address Register in the IOP and calculates the address of the DRT entry.   DB is set to the DBI value in the third word of the DRT entry (7).

h.   The Status Register (8) is set to privileged mode, external interrupts enabled (%140000).

i.   The DB-Bank Register (9) is set to 0.

j.   The S Register is set to point at location Q+3 (10) and the device number of the interrupting device is stored into this location.   At this point, the ICS is fully delimited by register values and is ready for handling interrupt data.

k.   The external program label for the interrupt receiver code is fetched from the second word of the DRT entry.   The CST entry is obtained from the segment number in the external program label.   Then, the PB-Bank Register (11) is set based on the CST entry.

l.   The PB Register (12) is set based on the CST entry.

m.   The PL Register (13) is set based on the CST entry.

8-11

n. The starting address of the interrupt receiver code is ob-
    tained from the STT entry pointed to by the external program
    label in the DRT entry. The interrupt receiver code segment
    number is placed in the Status Register. The P Register (14)
    is set to this value and the CPU fetches the instruction at P
    and begins executing the interrupt receiver code.

The following steps relate to figure 8-4 and list the actions oc-
curring if a second interrupt of higher priority is received
while processing the first interrupt. Assuming a still higher
priority, another interrupt could interrupt the second routine in
the same manner as described below. This example shows how sev-
eral levels of interrupts can be nested on the ICS. Since the
ICS is common to all external interrupts, no further switching of
environments is necessary for additional interrupts. This reduc-
es the interrupt response time. If, however, the second inter-
rupt did not occur before completing the processing of the first
interrupt, the sequence of operations would skip from this point
(step a) to step g. The CPU recognizes a second interrupt while
executing the interrupt receiver code for the first interrupt.
The CPU, therefore, again passes control to the Interrupt Hand-
ler. The sequence continues as follows:

a. The Interrupt Handler pushes into memory any TOS elements
    that are in CPU registers, and pushes the usual six-word mar-
    ker onto the ICS. The fifth and sixth words are the values
    that are currently in the DB-Bank and DB Registers respec-
    tively at the time of the interrupt.

b. The Q Register is updated to point at the Delta-Q word of the
    new marker. The Delta-Q value is the number of locations
    back to the Delta-Q word of the previous marker.

                            Note

        Unlike the first interrupt, subsequent in-
        terrupts do not store S into Q-6 at this
        point since such action would overlay one
        of the variables associated with the user
        who was first interrupted.

c. The CPU obtains the device number from the Interrupt Address
    Register in the IOP and calculates the address of the DRT
    entry. DB is set to the DBI value in the third word of the
    DRT entry.

d. The S Register is set to point at location Q+3 and the device
    number is stored into this location. At this point, the ICS
    is fully delimited by register values and is ready for hand-
    ling interrupt data.

e. The external program label for the interrupt receiver code is fetched from the second word of the DRT entry. The starting address of the interrupt receiver code is obtained from the STT entry pointed to by the external program label. The P Register is set to this value and the CPU fetches the instruction at P and begins executing the interrupt receiver code.

f. Assuming there are no other higher priority interrupts, the interrupt routine for the second device runs to completion and then exits using the IXIT instruction. The exit, as usual, is made via the stack marker. The return address is obtained from the stack marker, the Q Register is restored back to the previous setting (using the Delta-Q value from the stack marker), pointing to the Delta-Q word of the Dispatcher marker. The S Register is moved back to the location just preceding the second stack marker. One of the actions of the IXIT instruction is to issue a Reset Interrupt command to the interrupting device controller which clears the interrupt active condition and unblocks the interrupt poll line to lower priority devices.

g. The interrupt receiver code for the first interrupt now runs to completion and an exit is made, usually back to the user process. Again, the IXIT instruction issues a Reset Interrupt command to the device controller. This completes the sequence of operations.

If an external interrupt should occur while the Dispatcher is executing, the interrupt is treated in a slightly different way. If the CPU recognizes an interrupt while the Dispatcher Flag is set, the sequence effectively repeats steps b through g above with the added actions that, in step d, bit 0 of Delta Q is set to 1 (indicating a Dispatcher interrupt) and the Dispatcher Flag is cleared.

8-12. INTERNAL INTERRUPT PROCESSING

As listed in table 8-1, there are 35 internal interrupts including 14 user traps. These 35 interrupts are processed by the segment whose CST entry number is 1. Each interrupt has an entry in the Segment Transfer Table (STT) which points to the start of the code to process the interrupt. The user-related traps all share the same STT entry and the parameter value determines the processing to be performed. When internal interrupts are being processed, all external interrupts are disabled. Internal interrupts therefore have higher priority. Among internal interrupts, however, there is no priority structure (except in the case of simultaneous interrupts); any internal interrupt may interrupt the processing of any other. If multiple interrupts occur simultaneously, they stack their markers in the following order and are therefore, serviced in the reverse order; integer overflow, system parity error, memory address parity error, data parity error, non-responding module, bounds violation, illegal address, module interrupt, external interrupt, and power fail. In all cases, the

Interrupt Handler loads a parameter onto the stack. The parameter (listed in table 8-1) passes information regarding the interrupt from the hardware to the interrupt processing software. In some cases, the parameter is simply an interrupt identification number; in other cases, the parameter gives specific information, such as a program label, to the interrupt routine.

8-13. General Descriptions

8-14. BOUNDS VIOLATION. A bounds violation trap is caused by attempting to address locations outside of a specified program domain or data domain. (Refer to paragraph 2-65.)

8-15. ILLEGAL MEMORY ADDRESS. A memory address interrupt is caused by attempting to access a word of memory that does not physically exist on the system.

8-16. NON-RESPONDING MODULE. A non-responding module interrupt occurs when the CPU requests information from some other module and that information is not received in a reasonable length of time (a preset time on the order of 4.6 milliseconds).

8-17. SYSTEM PARITY ERROR. A parity error is detected on the 8-bit system information (TO, FROM, COMMAND) transmitted by the CPU to memory, or by memory to the CPU. This error will also be generated in the case where the CPU is waiting for data and a Memory-to-IOP transmission takes place with bad parity. In this case a transfer error is also sent to the requesting device. Note that the converse is also true (i.e., if the IOP is waiting for data and the CPU receives a transmission with bad system parity, a transfer error is sent to the requesting device). The above is the result of the CPU and IOP sharing the same module number. A system parity error also results if any module sends data with bad parity (not addresses) to memory.

8-18. ADDRESS PARITY ERROR. A parity error is detected by the memory on the 16-bit address word sent to it from any module. Upon detection of the error, the memory sends an appropriate error signal back to the CPU and prevents the word addressed from being altered.

8-19. DATA PARITY ERROR. A parity error is detected by the CPU on the 16-bit data word sent to it from the memory. When a parity error is detected on a memory transmission, the appropriate bit is set in CPX1 (the CPU status word for RUN-mode interrupts) and the instruction runs to completion (with the exception of certain interruptable instructions such as the group of move instructions). The result of the instruction is normally meaningless. If the parity error is due to a CPU read cycle (outgoing system or address information or incoming data), it is possible that the received data will be used by the CPU as an address for a following write cycle. In this case it would be possible to store erroneous data at some location. However, since bounds checking is done on the address, the worst that can happen is the destruction of a memory location in the current user's stack (as-

suming user mode; if in privileged mode, a system crash could occur). With single-bit error correcting memory, the probability of having a data parity error is very small.

8-20. MODULE INTERRUPT. A module interrupt occurs when a CPU receives a transmission from a system module (hardware) from which it is not expecting a transmission. The offending module number (FROM code) is passed to the interrupt routine as a parameter. The interrupt routine may then attempt to identify the source of the error and take appropriate action. The interrupt is disabled (when bit 1 of the Status Register is 0). This interrupt can also be used as a flag between the CPU and another module for information swapping.

8-21. POWER FAIL. This routine saves the software status in a format suitable for automatic restart, making use of the finite time between the detection of a power failure and the loss of usable power (approximately 10 milliseconds).

8-22. UNIMPLEMENTED INSTRUCTION. This system trap occurs when the CPU detects a bit pattern in the Current Instruction Register which is not a valid instruction. This trap cannot be disabled by the User Traps Enable/Disable bit in the Status Register.

8-23. STT VIOLATION. The STT Violation trap cannot be disabled. The conditions that can cause this trap are as follows:

a. The STT in an external program label is greater that the STT length (pointed to by PL) in the referenced segment. This error can occur while attempting to set up a new segment.

b. In the LLBL instruction, if the label which is fetched from PL-N is an internal label and N is greater than 127 (%177), the trap is invoked. (This would require too large an STT number when creating the external label.)

c. In PCAL, when setting up a new segment, if the STT number in the external program label points to an external program label in the new segment, the trap is invoked.

d. If (PL-N) in an SCAL instruction is an external label, the trap is invoked.

8-24. CST VIOLATION. This trap is caused by an attempt to transfer to segment 0 or a segment number referenced through an external program label that is greater than the CST length.

8-25. DST VIOLATION. The DST segment number referenced by the MFDS, instruction is greater than the number of entries contained in DSTL (the first word of the DST).

8-26. STACK UNDERFLOW. The process being exited is non-privileged and SM is less than DB. This might result from deleting too much information from the stack or from using the SETR or SUBS instructions incorrectly.

8-27. PRIVILEGED MODE VIOLATION. This trap is caused by an attempt to execute a privileged instruction in user mode (that is, when bit 0 of the Status Register is 0). This violation also occurs in EXIT if an attempt is made to exit from user to privileged mode or if exiting from user mode and the external interrupts bit in the status word has been altered.

8-28. STACK OVERFLOW. A stack overflow results from attempting to stack more data than can be contained on the current stack (SM greater than Z). The system makes the decision whether to abort the process or to expand the stack.

8-29. INTEGER OVERFLOW. An integer overflow occurs when the result of an integer operation (ADD, SUB, etc.) is outside the allowable range of integers which is -32768 to +32767.

8-30. FLOATING-POINT OVERFLOW. This trap occurs when the magnitude of the result of a two-word floating-point operation is larger than the largest representable floating-point number which is 1.157921×10^{77}.

8-31. FLOATING-POINT UNDERFLOW. This trap occurs when the magnitude of the result of a two-word floating-point operation is less than the smallest representable positive number which is 8.63617×10^{-78}, and is not equal to zero.

Note

Floating-point overflow and underflow can best be understood by referring to the chart below showing the range of valid numbers.

```
 |_____    |    |    |_____    |
 |----------------    |    |    |-------------------    |
 |                    |    |    |                       |
 A                    B    0    C                       D
```
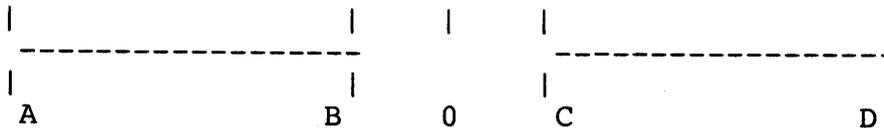
where

A = -1.157921×10^{77}

B = -8.63617×10^{-78}

C = 8.63617×10^{-78}

D = 1.157921×10^{77}

A number is valid if it is between A and B, C and D, or equal to 0.

8-32. INTEGER DIVIDE BY ZERO. This trap occurs when the divisor in a DIV, DIVI, DIVL, or LDIV instruction is equal to zero.

8-33. FLOATING-POINT DIVIDE BY ZERO. This trap occurs when the divisor in an FDIV instruction is equal to 0.

8-34. EXTENDED PRECISION FLOATING-POINT OVERFLOW. This trap occurs when the magnitude of the result of an extended precision floating-point operation exceeds the largest representable extended precision value which is $1.157920892373162 \times 10^{77}$.

8-35. EXTENDED PRECISION FLOATING-POINT UNDERFLOW. This trap occurs when the magnitude of an extended precision floating-point operation is less than the smallest representable positive extended precision value which is $8.636168555094445 \times 10^{-78}$ and is not zero.

8-36. EXTENDED PRECISION FLOATING-POINT DIVIDE BY ZERO. This trap occurs when the divisor in an extended precision divide operation is zero.

8-37. DECIMAL OVERFLOW. This trap occurs when a packed decimal result has too many significant digits for the specified storage size. Except for the NSLD instruction and MPYD with actual result greater than 28 digits, when this occurs the low order digits of the result are stored; surplus high order digits are discarded.

8-38. INVALID ASCII DIGIT. This trap occurs when a decimal arithmetic instruction encounters an invalid ASCII digit.

8-39. INVALID DECIMAL DIGIT. This trap occurs when a decimal arithmetic instruction encounters an invalid packed decimal digit.

8-40. INVALID WORD COUNT. This trap occurs when a word count for a decimal instruction is less than zero or greater than six.

8-41. RESULT WORD COUNT OVERFLOW. This trap occurs when a digit count for a decimal instruction < 0 or > 28.

8-42. DECIMAL DIVIDE BY ZERO. This trap occurs when an attempt is made to divide a decimal number by zero.

8-43. ABSENT CODE SEGMENT. The absence bit in the CST entry for the referenced segment is set to 1. This check is performed in PCAL, EXIT, IXIT, DISP, and the firmware Interrupt Handler. If during PCAL, the program label is passed as the parameter to the Interrupt Handler; if in EXIT, the number of words to be deleted from the stack is passed; and if in IXIT, a zero is passed.

8-44. TRACE. Non-local PCAL and external interrupts check the trace bit in the CST entry for the referenced segment. EXIT and IXIT check bit two of the return address in the marker stacked by PCAL or the external interrupt (this bit is set by the trace rou-

tine software if it is desired to trace exits). In either case, if the bit tested is 1, the trace routine in entered. For PCAL's and external interrupts, another marker is stacked first which is used by the EXIT from the trace routine. For EXIT and IXIT, no marker is stacked; hence, bit 0 of the return address of the last marker stacked (prior to EXITing from trace) must be cleared by software in the trace routine. Otherwise, an infinite trace loop could occur. Tracing segment 1 results in a system halt. Tracing external interrupts or the Dispatcher requires special software in the trace routine due to the differences in EXIT and IXIT.

8-45. STT ENTRY UNCALLABLE. The uncallable bit in a local label (or in PL if the STT number is 0) is set to 1. This label is referenced by a PCAL from another segment. This trap does not stack a new marker.

8-46. ABSENT DATA SEGMENT. The absence bit in the DST entry for the referenced segment is set to 1.

8-47. POWER ON. The Power On routine is entered either by an internal power turn-on or by an automatic restart following a power failure when automatic restart is enabled by a panel-switch. (The computer will halt on restoration of power if automatic restart is disabled.) Assuming that automatic restart is enabled, the Power On routine will set up the software environment and pass control to the operating system.

8-48. COLD LOAD. Pressing the LOAD switch while simultaneously pressing the ENABLE switch causes the CPU to start its cold-load microprogram which begins by reading the operator-set switches on the panel. The switches will have been set to indicate the cold load device number and an 8-bit control byte. The microprogram generates an eight-word I/O program beginning at the DRT entry locations for the specified device and then issues an SIO instruction to that device and goes into a waiting loop to wait for an external interrupt from that device. Meanwhile the IOP causes the device controller to begin executing the eight-word I/O program. This program reads in a 32-word bootstrap loader (a larger program) which in turn reads in still larger blocks (e.g., 128 words) which eventually accomplish the loading of all required fixed memory locations. This includes overlaying the previously used DRT locations with normal DRT entries. Finally, the I/O program causes the device controller to generate the external interrupt that the CPU has been waiting for, and ends. The CPU then proceeds to initialize the registers for execution of code segment 1, with the ICS as the data domain. The Status Register is set to 100001 (octal) to indicate privileged mode, and code segment 1. Then the CPU halts. When RUN is pressed, the cold-load routine in segment 1 will execute, setting up the operating conditions for the operating system (software tables, linkages, etc.). Once this is complete, the system is in full operation.

8-49. Sequence For ICS-Type Interrupts

Figure 8-5 illustrates the sequence of operations for processing ICS-type internal interrupts. The figure shows how control is transferred from the point of interrupt in the user's code to the start of the interrupt code segment and how the data domain is switched from the user's stack to the Interrupt Control Stack.

The initial assumption is that the current process is executing at the point P in the user's code when an interrupt condition occurs. The CPU then passes control to the Interrupt Handler. The sequence of operations is as follows:

a. The first action of the Interrupt Handler is to push into memory any TOS elements of the current user's data that are in CPU registers (1, figure 8-5). This takes a maximum of four memory cycles if all four registers are full. Next, a normal four-word stack marker is pushed onto the user's stack followed by the value of the user's DB-Bank and the absolute value of DB that is currently in use. (DB may not necessarily point to a location within the user's stack, such as if a system intrinsic using a split stack had been called at the time of the interrupt.) This action preserves most of the user's environment; the current value of S will be preserved later in step f.

b. The S-Bank Register (2) is set to 0. (The ICS is always in Bank 0.)

c. The Interrupt Handler now goes to location 5 and loads the QI value (3) into the Q Register. This points at the Delta-Q location of the permanent Dispatcher marker. (As explained previously, this location contains a value of 0.)

d. The contents of location 6 is fetched and the value of ZI (4) is loaded into the Z Register. This establishes the stack limit for the ICS. (The ICS Flag in the CPX1 Register is also set.)

e. The DL Register (5) is set to the limit value of %177777.

f. The user's value of S relative to stack DB (at QI-4) is calculated and stored in QI-6 (6). (Up to this point the operation has been identical to the sequence of operations for external interrupts, described earlier.)

g. An external program label (7) is created which points to segment 1, and whose STT number is a function of the type of interrupt. (Refer to table 8-1.)

h. S is now set to Q+3 and a parameter is pushed onto the ICS at that location (8). Most ICS-type internal interrupts pass the external program label however. For example, a Module Interrupt passes the module number.
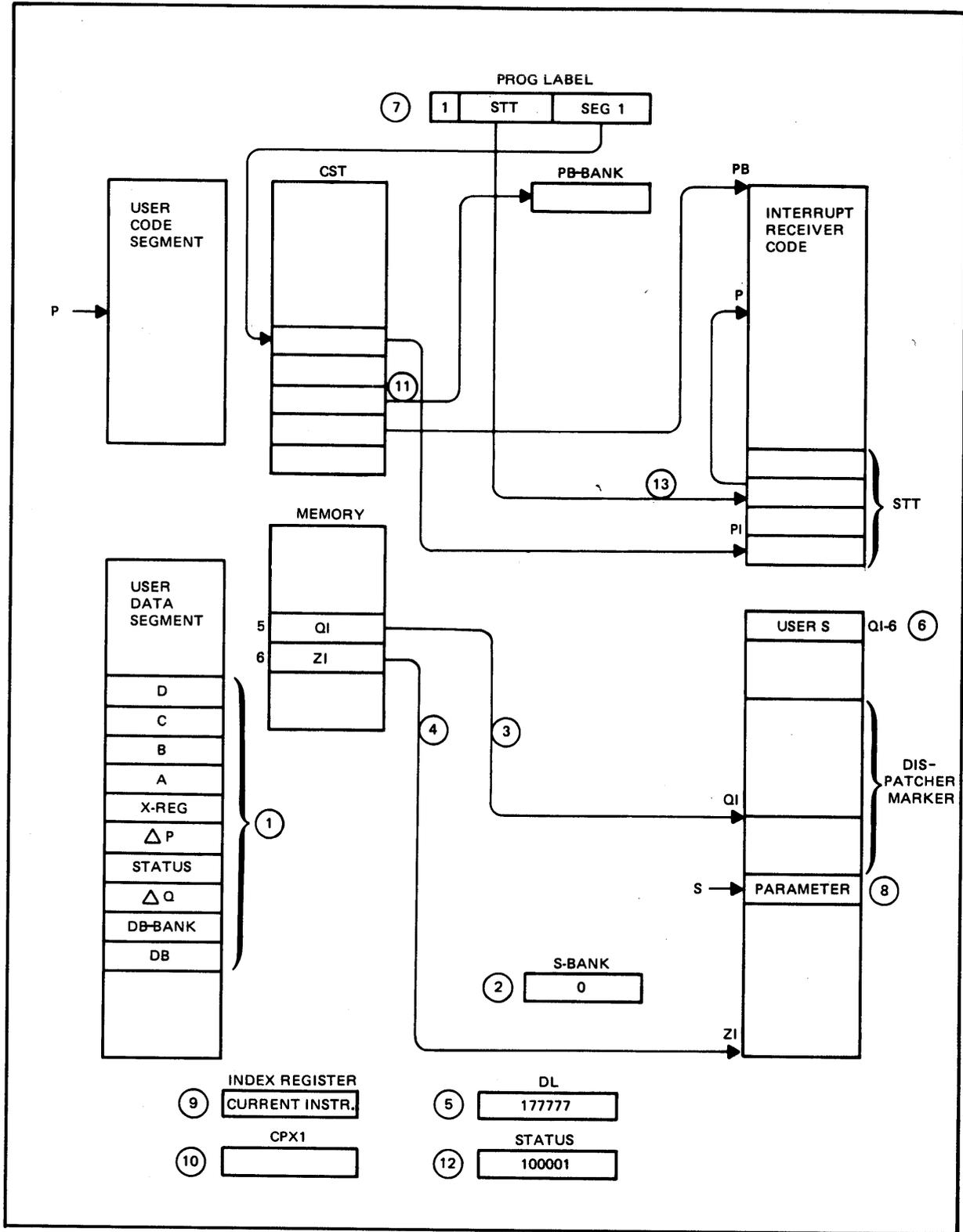
Figure 8-5. ICS-Type Internal Interrupt

i. The current instruction (9) is placed in the Index Register.

j. The interrupt condition is cleared (10).

k. The PB-Bank, PB, and PL Registers are set up based on CST entry number 1 (11).

l. The Status Register (12) is set to privileged mode, segment 1 with all other bits cleared (%100001).

m. The starting address of the interrupt receiver code is obtained from the STT entry (13) pointed to by the external program label. The P Register is set to this value and the CPU fetches the instruction at P and begins executing the interrupt receiver code.

Additional ICS-type internal interrupts could occur before exiting from the interrupt code segment and they would be stacked on the ICS in a manner similar to that shown in figure 8-4. If there are any external interrupts, either suspended on the ICS or waiting for priority, they will be processed after all internal interrupts have been processed. (However, external interrupts can interrupt internal interrupt routines if the software re-enables the external interrupt system.) After all internal and external interrupts using the ICS have been processed, an exit back to the interrupted user will occur or the Dispatcher may be entered.

8-50. Sequence For Non-ICS Type Interrupts

Figure 8-6 illustrates the processing of non-ICS type internal interrupts. As shown in the figure, the ICS is not used and the interrupt code segment will operate on the user's stack. Assume that the user is executing at point P when an interrupt condition occurs. The CPU passes control to the Interrupt Handler and the sequence is as follows:

a. Any TOS elements that are in CPU registers (1, figure 8-6) are pushed into memory.

b. A normal four-word stack marker is pushed onto the user's stack (2).

c. The parameter (3) is pushed onto the stack. (Refer to table 8-1.)

d. The current instruction (4) is placed in the Index Register.

e. The Interrupt Handler generates an external program label (5) to the interrupt receiver code in segment number 1.

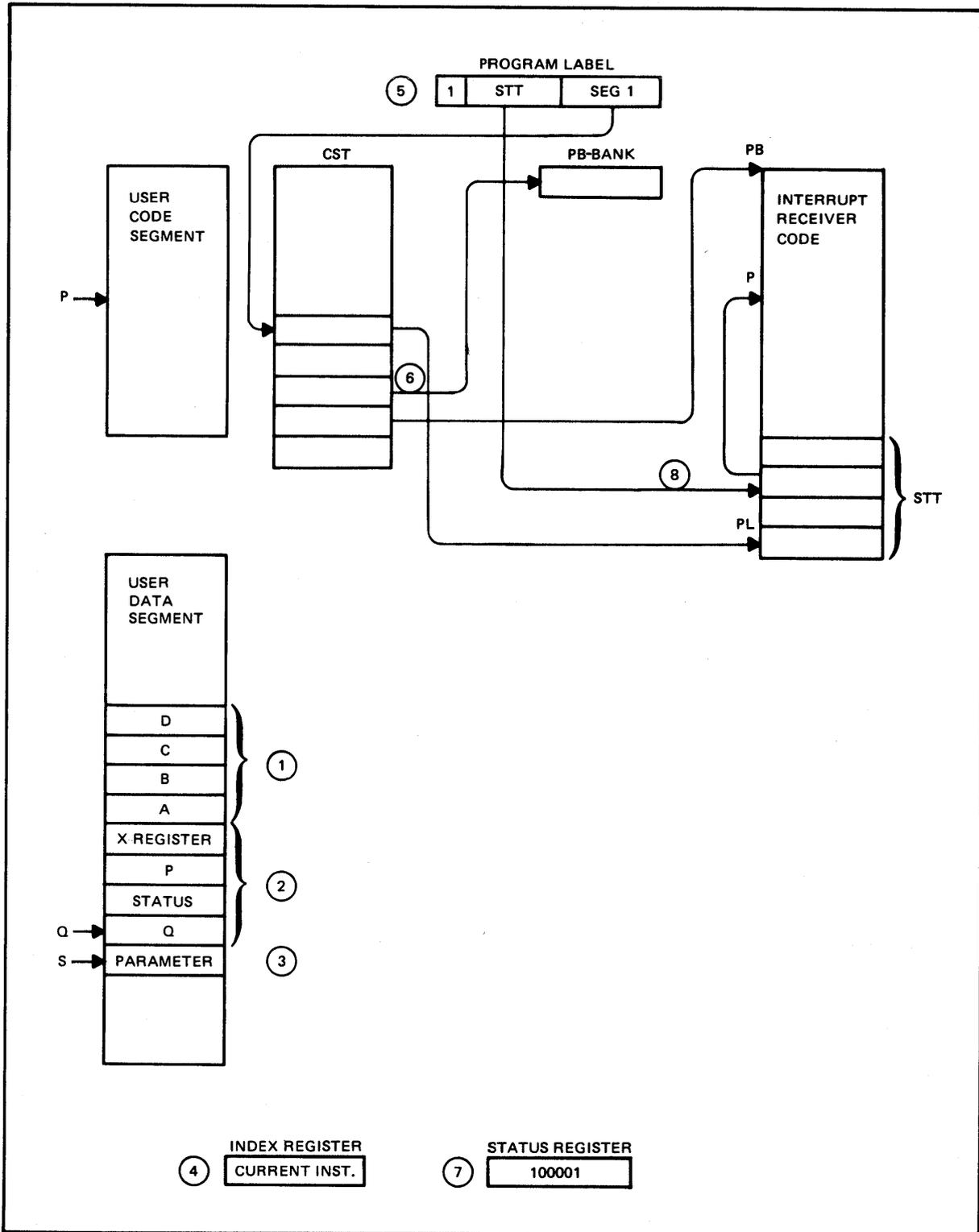f. The PB-Bank, PB, and PL Registers are set based on the CST entry (6).

Figure 8-6. Non-ICS Type Internal Interrupts

g. The Status Register (7) is set to privileged mode, segment 1 with all other bits cleared (%100001).

h. P is set from the local label, reached via the STT entry in

the external program label (8), thus transferring control to the interrupt receiver code.

8-51. INTERRUPT HANDLER

The Interrupt Handler is a microprogram (actually a set of micro-programs) permanently stored within a read-only memory in the CPU. The CPU periodically checks for the existence of a waiting interrupt condition which is stored in one of several bit posi-tions in a dedicated CPU register (CPX1 or CPX2), and then trans-fers control to the Interrupt Handler. The purpose of the Interrupt Handler is to save the interrupted environment and transfer control to the interrupt routine in software. The sus-pended environment is saved in a format that is ready to resume execution. The descriptions that follow are essentially a sum-mary of the preceding portion of this section. Figure 8-7 illustrates the operations performed by the Interrupt Handler. Generally, the sequence begins with the START block at the top left corner and ends with the NEXT CPU INSTRUCTION block at the bottom right corner.

8-52. DISP Instruction

The DISP instruction calls the Dispatcher which is a system pro-cess whose primary function is determining which active process will use the CPU and then transfer control to that process. The Dispatcher can be called from a user program if in privileged mode. For example, the last instruction of a user process is a PCAL to a system process called TERMINATE´ which, among other things, cleans up the CST, DST, and PCB entries for the user pro-cess. TERMINATE´ then issues a DISP instruction. Some system error handling routines such as trap handlers may use it to call the Dispatcher after aborting the user program. The DISP in-struction can be executed by an Interrupt Handler after servicing all pending interrupts from a multiple device controller. In this case, the Dispatcher is not actually called, but instead a condition code of CCG is set and bit 0 of QI is set to instruct the IXIT instruction what to do. The next CPU instruction after the DISP instruction would then be executed and the Interrupt Handler would execute the IXIT instruction. The IXIT instruction then uses bit 0 of QI to determine which path to take. All pro-grams which use the DISP instruction must be prepared to handle the condition of the Dispatcher being pseudo disabled. (Refer to paragraph 4-16, 7.)

8-53. Pseudo Enabling/Disabling The Dispatcher

The PSDB (Pseudo Disable) and PSEB (Pseudo Enable) instructions are used to pseudo disable and enable the Dispatcher. (Refer to paragraph 4-16, 7.) The two instructions must be executed in pairs; for each disable, there must be a corresponding enable within the same process. The Dispatcher can be locked several levels deep with PSDB instructions, but must have one PSEB to unlock each level. A count is maintained in QI-18 for the number of disables which have not been unlocked. These instructions are

used to prevent a dispatch during critical sections of code and to avoid unnecessarily restarting the Dispatcher. If the DISP instruction is executed and the Dispatcher is disabled (QI-18 is non-zero), then bit 0 of QI is set to 1 and the next CPU instruction is fetched. This bit is reset either by IXIT or PSEB when QI-18 becomes zero. If QI-18 is already zero at the start of a PSEB instruction, a system halt will occur.

8-54. IXIT Instruction

Figure 4-13 is a simplified flowchart of the IXIT instruction. IXIT operates in either of two ways. The first is by the Dispatcher to transfer to a process being launched (1, figure 4-13). The second (2) through (6), is to exit from ICS type interrupt routines. If the interrupt service routine is not in segment number 1, it is assumed to be an external interrupt routine and a Reset Interrupt is sent to the device whose device number is in Q+3. If bit 0 of Q is zero and if Q=QI, the return is to the interrupted process (2). Otherwise the return is to a lower priority interrupt which was interrupted (3). If bit 0 of Q is 1 and bit 0 of QI is zero, the return is to the Dispatcher which was interrupted (4). If both bit 0 of Q and bit 0 of QI are 1, a DISP instruction has been executed and the request to start the Dispatcher is still pending. If QI-18 is zero, the Dispatcher is not disabled, QI is cleared, and a transfer is made to the Dispatcher's entry point (5) or (6). It does not matter whether a process (Q=QI) or the Dispatcher (Q not equal to QI) was interrupted. If QI-18 is non-zero, the Dispatcher is disabled and the DISP request cannot be carried out at this time. Instead, IXIT returns to the interrupted Dispatcher (Q not equal to QI (4a)), or to the interrupted process (Q = QI (2a)). The Start Dispatcher request is still pending, (bit 0 of QI is 1).

8-55. INTERRUPT SYSTEM SERVICING INFORMATION

Except for checking the interrupt poll line for proper installation, no repair procedures are required for the interrupt system. As previously discussed, the interrupt priority of a divice is determined by the device's logical proximity to the IOP on a jumpered interrupt poll line. The interrupt poll line is wired during system configuration from the IOP to whatever device is assigned first priority and then from device to device according to assigned priority. The interrupt poll line terminates at the device of lowest priority.

The interrupt poll line for any system starts at connector pins 79 (INTPOLL) and 80 (GND) of connector 10P1 of the CPU/IOP backplane. The interrupt poll line consists of a twisted pair of wires; one blue wire and one white wire. This twisted pair is terminated at each end with a two-pin spring-clip connector that clips onto pairs of vertically-aligned connector pins. At the CPU/IOP backplane, the twisted pair must be installed with the white wire connected to the top pin of the two vertically-aligned pins. At the device controller interface PCA's, the twisted pair must be installed with the white wire connected to the bottom pin

of the two vertically-aligned pins.

The interrupt poll line carries the INTPOLL signal from the IOP to connector P1 of the device interface PCA with the next highest priority. It enters this PCA on the fifth vertically-aligned pair of connector pins from the left; pins 48 (INTPOLL IN) and 47 (GND). The signal is exited from each PCA on the seventh vertically-aligned pair of connector pins from the left; pins 44 (INTPOLL OUT) and 43 (GND).
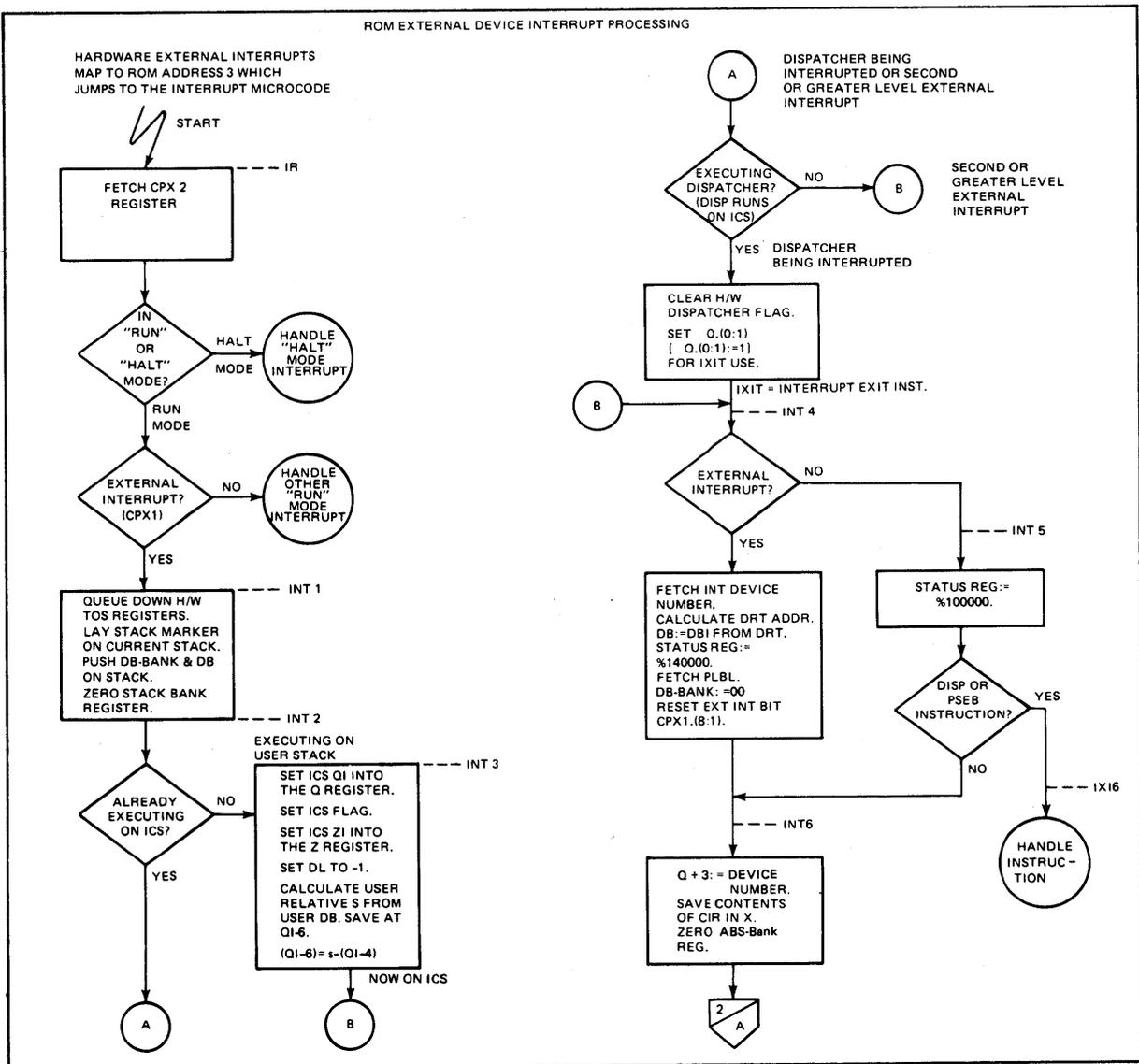

Figure 8-7. Interrupt Handler Flowchart (Sheet 1 of 2)
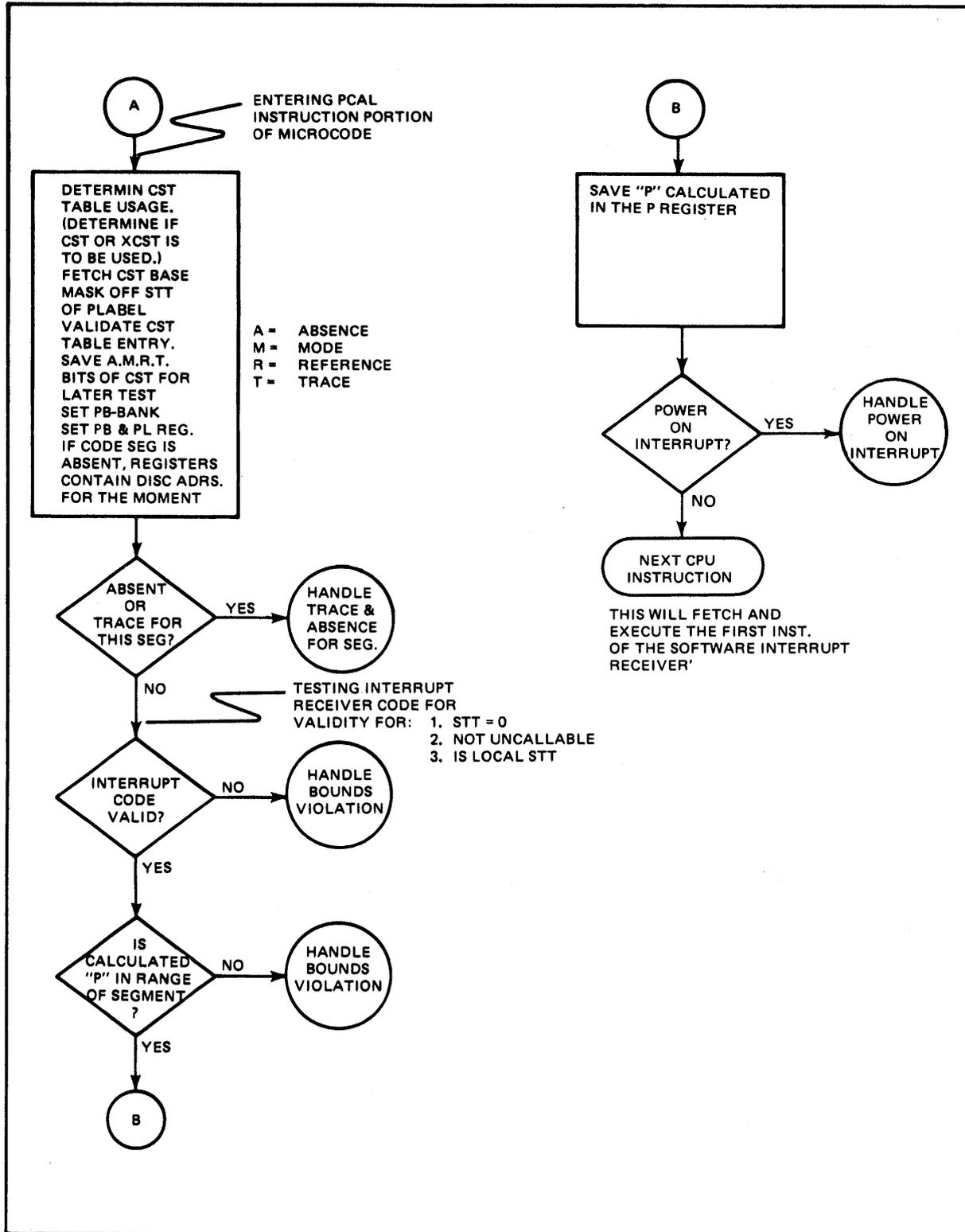
Figure 8-7. Interrupt Handler Flowchart (Sheet 2 of 2)

NOTES

NOTES

This section contains principles of operation and servicing information for the HP 32421A Series III Computer System power supplies. The HP 32435A Series III power supplies are discussed in Section X of this manual.

9-1. INTRODUCTION

The HP 32421A Series III Computer System has four power supplies; two HP 30310A Power Supplies, one HP 30311A Power Supply, and one HP 30312A Power Supply.

9-2. HP 30310A OPERATION

The HP 30310A Power Supply provides +20V, +15V, +5V, -20V, -15V, and -5V regulated dc supply voltages by converting a 208/240volt, single-phase, 50- or 60-Hz power source. The ac input to the power supply is controlled by the front panel POWER switch shown in figure 9-1. The dc output voltges are controlled by the Dc Control Panel SYSTEM switch also shown in figure 9-1. The output voltages provide status signals for protection of the software stored in CPU memory. The computer system hardware is protected by circuits that sense overvoltage, overcurrent, or overtemperature conditions. Circuits within the power supply are protected by various overvoltage and overtemperature circuits, current limit circuits, and fuses. The power supply is designed to immediately turn the power off when an overvoltage condition occurs. For undervoltage or overtemperature conditions, a short delay is generated before the power is turned off. This delay permits the CPU to store data which, in turn, makes power on and restart much easier.

A block diagram of the power supply is shown in figure 9-2 and discussed in paragraphs 9-3 through 9-11. (Complete specifications and detailed theory of operation for the power supply are contained in the HP 3000 Series II/III Computer Systems Service Manual, part no. 30000-90018.)

9-3. Primary Power Circuit

The ac line voltage enters the power supply through a connector and passes through a 5-ampere fuse and radio frequency interference (RFI) filter to the POWER switch. The POWER switch is located on the front panel of the power supply. With the POWER switch in the ON position, line voltage is applied to the preregulator, cooling fan, and a step-down transformer. The fan and transformer circuit are protected by a 1-ampere fuse.
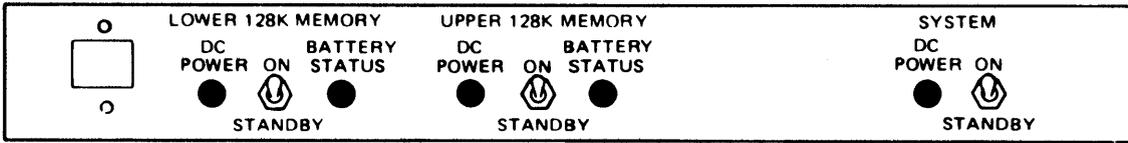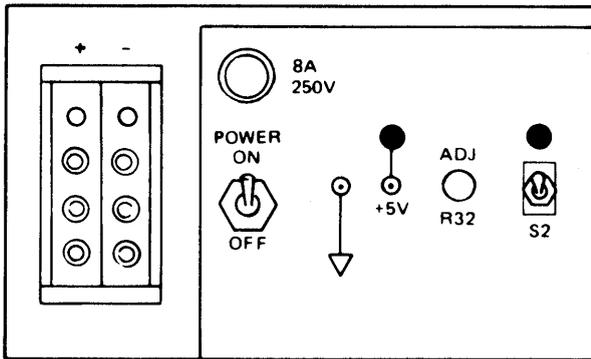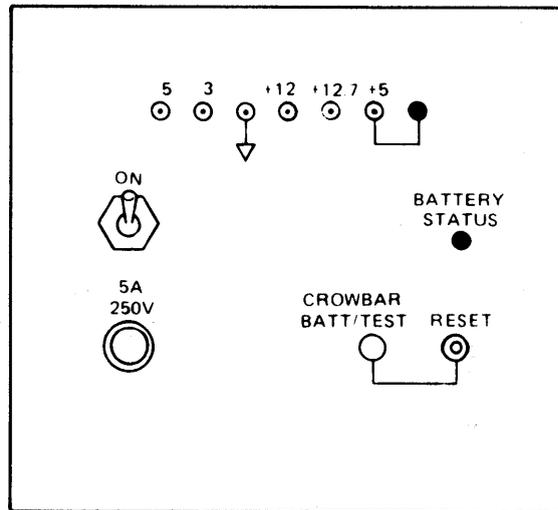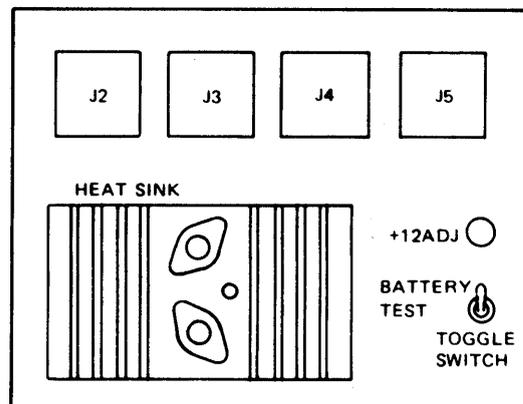
Figure 9-1. Power Controls and Indicators
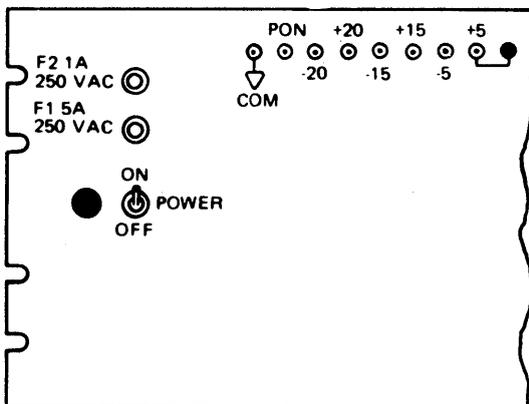
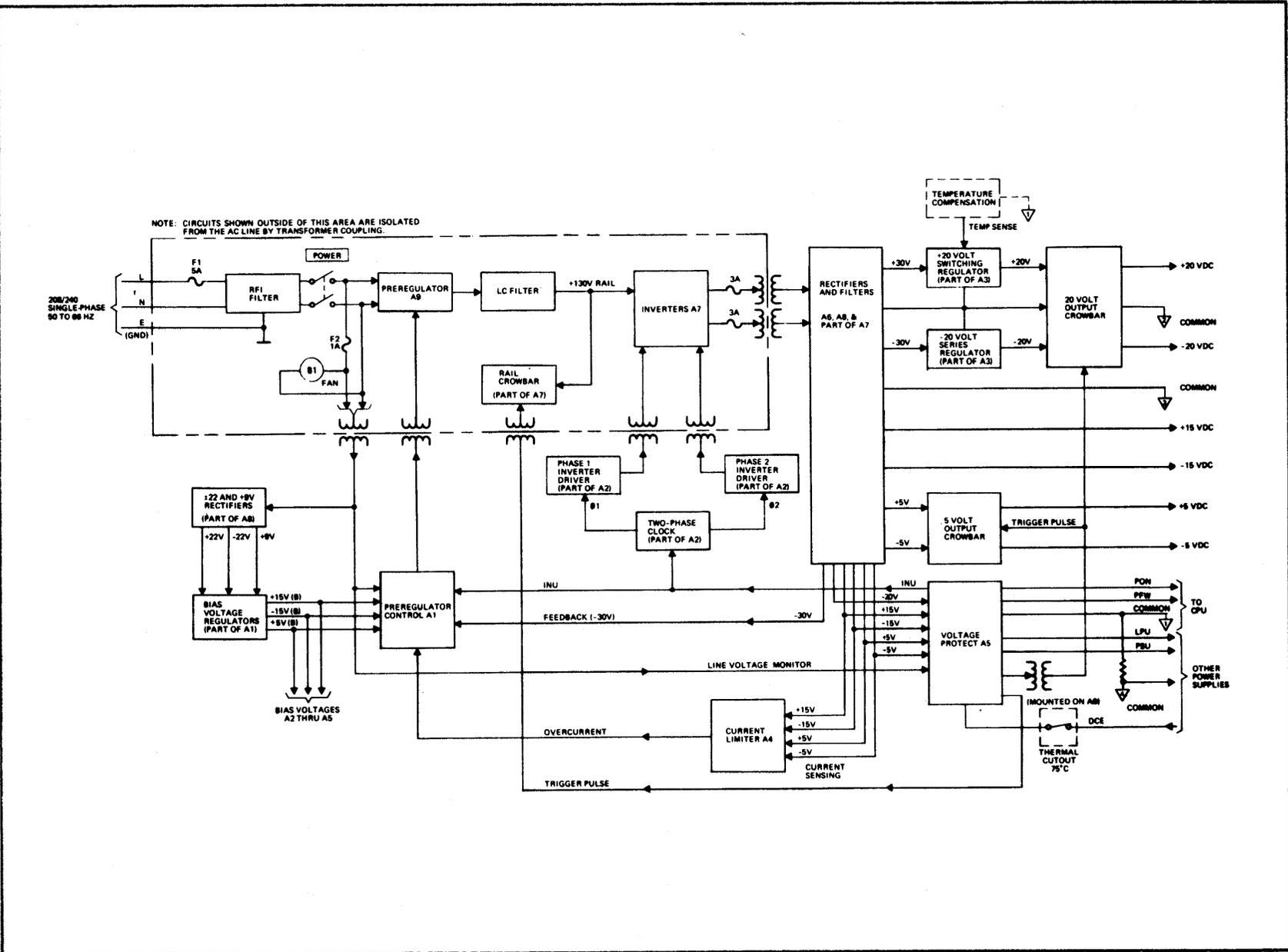Figure 9-2. HP 30310A Power Supply Block Diagram

Figure 9-2. HP 30310A Power Supply Block Diagram

9-4. Preregulator A9

The preregulator contains a silicon controlled rectifier (SCR) bridge that converts the ac line voltage to a unidirectional, pulsed voltage. The pulsating voltage is filtered to become the regulated +130-volt rail voltage. The rail voltage is applied to the inverters and is used as a basis for all dc output voltages. The SCR bridge is controlled by the preregulator control circuit through an isolating transformer.

9-5. Preregulator Control A1

The stepped-down line voltage is applied to full-wave rectifiers which supply unregulated +22, -22, and +9 volts dc for the bias voltage regulators. The regulators provide internal power supply bias voltages. The preregulator control circuit uses the ac line frequency, -30 volt dc feedback, and status of the current limiter and voltage protect circuits. The preregulator control circuit supplies trigger pulses to the preregulator that determine the "on" time of the SCR bridge circuit to maintain proper control of the +130-volt dc rail.

9-6. Inverter A7

The inverter circuits convert the 130-volt dc output of the preregulator to a square-wave ac voltage that is transformer-coupled to the rectifiers. The transformer coupling provides isolation for stages following the inverter. The 800-Hz operating frequency of the inverter is determined by the inverter driver. There are two inverter circuits within the inverter that operate 90 degrees out of phase with each other. Each circuit is fused for 3.0 amperes.

9-7. Inverter Driver A2

The inverter driver generates an 800-Hz, two-phase clock which is timed to develop Phase 1 and Phase 2 drive signals. The two inverter driver circuits are transformer-coupled to the two inverter circuits.

9-8. Full-Wave Rectifiers and Filters

The transformer-coupled inverter output is rectified and filtered to provide dc outputs of +15, +5, -5 , and -15 volts dc. Additionally, +30 and -30 volts dc are supplied to the +20 and -20 volt regulators. An independent -30 volt output is fed back to the preregulator control circuit to maintain output voltage regulation.

9-9. 20-Volt Regulators

The 20-volt regulators consist of a +20 and a -20 volt regulator. The +20-volt regulator is a switching regulator which converts the +30 volt rectifier output to a regulated +20 volt dc output. The -20-volt regulator is a series regulator which converts the

-30 volt rectifier output to a regulated -20 volt dc output. The +20 and -20 volts dc are used by a power supply for the semiconductor memory. An analog signal from the memory power supply is used by the +20-volt regulator to control the output voltage. The -20V-regulator is designed to track the +20-volt regulator so that the two outputs are equal and opposite in polarity.

9-10. Current Limiter A4

The current limiter circuits monitor the individual dc voltage drops across the output filter chokes for the +15, +5, -5 and -15 volt outputs. Any excessive current drawn from these outputs results in the immediate generation of an Overcurrent signal. The Overcurrent signal is used by the preregulator control circuit to limit the preregulator output voltage to protect the power supply.

9-11. Voltage Protection and Control A5

The voltage protection and control circuits contain overvoltage sensing circuits to protect the computer system hardware and overvoltage sensing circuits to protect system software. The overvoltage comparator circuits monitor all dc output voltages, with the exception of the +20 volt output. When an overvoltage condition is sensed, an overvoltage latch is set. Transformer-coupled crowbar trigger pulses are generated which crowbar the +130V RAIL, +20, -20, +5, and -5 volt dc outputs. Also, when the latch is set, the Inverter Up (INU) goes low and disables the inverters and preregulator. Circuits also monitor the internal thermal cutout switch (overtemperature sense) and the external Dc Enable (DCE) signal. If the internal power supply temperature exceeds its fixed limit or an external DCE signal is removed, the Power Fail Warning (PFW) signal goes low after a 70-ms delay. After another delay of 12 ms, the INU signal goes low, turning the preregulator and inverter off. Since the output voltages go down, the Power Supply On (PON) signal goes low. The SYSTEM switch controls the power supply outputs with the DCE signal. This signal appears as an overtemperature condition to the power supply and initiates the overtemperature sequence.

The undervoltage comparator circuits monitor all output voltages with the exception of the +20-volt output. If any of these dc output voltages drop below specified limits, the PON signal goes low. An excessive current overload causes the output dc voltages to drop. The undervoltage sequence is initiated due to this condition. Circuits also monitor the input ac line voltage. If it drops below a preset limit, the PFW signal goes low and, after a minimum delay of 5 ms, the PON signal also goes low. When the dc output voltages and ac line voltage are above the specified low limits and the thermal switch is closed, a high PON signal is provided for the system within 0.6 second.

The undervoltage sensing circuits also provide power supply control signals for use when multiple supplies are "control paralleled" in a computer system. These signals are Power Supply Up (PSU), Line Power Up (LPU), and control common $\nabla\!\!\!4\!\!\!/$. PSU indicates that the dc output voltages are above specified limits. LPU indicates that the AC line voltage is above a specified limit, the thermal switch is closed and DCE enabled. When the PSU, LPU, DCE, and control common $\nabla\!\!\!4\!\!\!/$ signals of multiple supplies are wired in parallel, any supply can provide the PON and PFW signals to the system, and all multiple supply outputs can be controlled by a single DCE signal.

9-12. HP 30310A SERVICING INFORMATION

The HP 30310A Power Supply is a nonrepairable unit hinge-mounted in the cabinets as shown in figures 1-2 and 1-3. The power supply swings out on hinges for servicing and is removable from the hinges for replacement.

WARNING

The HP 30310A Power Supply weighs 50 pounds (22.7 kilograms). Two persons are required to remove the unit from its mounting hinges.

If the power supply is found defective, it must be replaced. Except for replacing open fuses, no repair procedures are required. However, preventive maintenance procedures must be performed on the power supply at scheduled intervals to prevent or minimize equipment deterioration. Preventive maintenance, adjustment, and troubleshooting procedures for the power supply are contained in paragraphs 9-13 through 9-18.

WARNING

Death or serious injury may occur if the following precautions are not observed.

While the input power is connected, use caution when working inside the power supply. Many exposed conductors carry low dc voltages which are capable of supplying heavy currents if short-circuited, resulting in high heat and the possibility of painful burns. Use caution when manipulating metal tools or probes. Wrist watches, metal necklaces, bracelets, or rings must not be worn. Avoid dropping tools,

screws, or other metal objects onto con-
ductors. Remove power and recover dropped
objects at once; if forgotten, damage
could result later.

Ac powerline voltage and 130 volts dc are
exposed when covers are removed. Exercise
extreme caution when working in the power
supply with covers removed, and never work
under this condition unless another person
is nearby and within sight. Also, remem-
ber that the test equipment is floating
with respect to earth ground, so the cases
can be at the same line voltage as the
point being measured in the power supply.
Thus, test equipment must be temporarily
enclosed and marked dangerous to alert all
personnel of unsafe conditions.

If feasible, before performing any work
inside the power supply, unplug the ac
power cable and wait three minutes for
filter capacitors to discharge. To pre-
vent explosion resulting from internal
heating, always be sure to replace filter
capacitors properly with respect to
polarity.

The highest ac voltage in the power supply is the ac line voltage
(250 volts rms, 350 volts peak). The highest dc voltage in the
power supply is 130 volts. The ac line voltage is exposed at the
input circuits of the power supply and at filter choke L1. The
130 volts dc is exposed at the preregulator filter circuit and
the inverter assembly A7. Additionally, ac voltages of 240 volts
peak are exposed at transformers T1 and T2.

If the test equipment has a metal case, the negative test lead
should not be connected to the case. Also, the negative lead
should not be connected to digital voltmeters that have floating
(guarded) inputs, to multimeters, nor to power supply chassis.
Instead, the test equipment chassis should be connected to the
computer system cabinet earth ground through the test equipment
power cable. Consequently, the oscilloscope case will "float"
with respect to earth ground. All test equipment should be plug-
ged into the ac power convenience outlets provided in the comput-
er system bay cabinets.

All measurements and connections to the power supply must be ref-
erenced to the appropriate common circuit. There are four such
circuits within the HP 30310A Power Supply: common $\boxed{1}$, common $\boxed{2}$,
common $\boxed{3}$, and common $\boxed{4}$. In subsequent paragraphs, all test and
adjustment procedures specify the particular common circuit to be
used.

```
CAUTION
```

Do not connect test equipment to power supply
chassis ground. All common circuits within
the power supply "float" with respect to
chassis ground. Damage to test equipment or
power supply components, and erroneous mea-
surements may result if this caution is
ignored.

9-13. Preventive Maintenance

The following preventive maintenance procedures are performed at
monthly or semi-monthly intervals. The frequency depends upon
the physical conditions prevailing at a particular site. Routine
maintenance once per month is adequate for most power supplies
that operate 24 hours per day, seven days per week. The interval
can be reduced in accordance with the amount of time the power
supply is turned off. The power supply is not removed from the
computer to perform preventive maintenance. Perform the preven-
tive maintenance procedures as follows:

a. Remove dust.

b. Check PCA's for proper seating.

c. Check cooling fan operation.

d. Check the dc operating voltages at the power supply front
    panel.

e. Check the ac voltages for ripple at the PCA cage backplane.

To gain access to the power supplies, open the rear door of the
CPU cabinet bay. The power supplies are hinge-mounted and may be
swung-out by removing the screws that attach the left side of the
front panel to the CPU cabinet bay. In this position, the power
supply top and bottom covers can be removed for maintenance and
test.

If required, use a vacuum cleaner to remove dust and other light
debris from the power supply. Loosen encrusted dust with a soft-
bristled brush, paying particular attention to heat dissipating
areas. With the top cover of the power supply removed, check all
PCA's for proper seating. Adjust where necessary. Set the power
switch to ON and check the cooling fan for proper operation.
Ensure that no objects interfere with fan blade rotation.

Before making voltage checks, the voltmeter must be allowed time
to warm up as prescribed by the manufacturer of the instrument.
Also, the computer must run, with any type of program, for at
least 15 minutes before making the voltage measurement. Perform

the voltage checks as follows:

a. Stop the computer program.

b. Measure the six dc voltages listed in table 9-1. These voltages are available for cursory measurement only at test jacks mounted on the power supply front panel. (See figure 9-1.)

c. Set the oscilloscope for checking ac voltage. On the PCA cage backplane, check each of the six voltages listed in table 9-1 for ripple. For each voltage, the indicated ripple should be less than that listed. If any voltage is not within specified limits, make the necessary adjustments as described in paragraphs 9-14 through 9-17.

Table 9-1. HP 30310A Dc Output Voltages

Voltage Test Point	Minimum Reading	Maximum Reading	Ripple Voltage Tolerance
+20	Refer to para 9-16.		
+15	+14.7	+16.5	0.4 volt peak-to-peak
+5	Set at +5.17		0.3 volt peak-to-peak
-5	-4.5	-5.3	0.3 volt peak-to-peak
-15	-14.7	-16.5	0.4 volt peak-to-peak
-20	Refer to para 9-16.		

9-14. HP 30310A Adjustments

Three adjustments should be made to the power supply after it is installed in the computer. These adjustments are accessible through the top cover of the power supply.

9-15. PREREGULATOR ADJUSTMENT. The +15, +5, -15 volt supply outputs are controlled by preregulator adjustment resistor A1R1 (+5, +15V ADJ) on the preregulator control PCA. If one or more of these voltages are not within tolerance when the voltage check is made, the preregulator should be adjusted as follows:

9-9

a. Set the power supply POWER switch to the ON position.

b. Connect the voltmeter between the center conductor of the +5 VDC connector and COM on the PCA cage backplane. While observing the voltmeter, adjust the +5, +15V ADJ resistor (A1R1) until the +5 volt output is 5.17 volts as specified in table 9-1.

c. Using the same COM as a return, connect the voltmeter, in turn, to the +15V, -5V, and -15V test jacks and verify that each output voltge is within the limits specified in table 9-1.

d. Set the power supply POWER switch to OFF and disconnect voltmeter.

9-16. 20-VOLT ADJUSTMENT. The +20 and -20 volt supply outputs are adjusted by setting resistor A3R2 (+20V ADJ) fully clockwise. These voltages are used by the HP 30311A Power Supply which regulates the voltages by applying an analog signal to the HP 30310A Power Supply's TEMP SENSE input terminal. The value is approximately 17.3 volts when the battery is fully charged and stabilized at room temperature.

9-17. VOLTAGE PROTECT PCA ADJUSTMENT. The purpose of this procedure is to check and, if necessary, adjust the +4.22 volt reference supply and line voltage monitor circuits on the A5 voltage protect PCA. Check and adjust the reference voltage and line voltage monitor circuits as follows:

a. Plug power supply ac line cord into autotransformer.

b. Set POWER switch to ON position. Increase input voltage to 208 volts ac.

c. Connect a voltmeter to terminal E1(+) and E2(-,COMMON 1) on A5. Adjust A5R2 until voltmeter indicates +4.22 (+/-0.01) volts dc.

d. Connect multimeter between TB3, pin 5(+) and TB3, pin 6(-, COMMON \\$\boxed{1}$/). Set multimeter controls to +10 VOLTS DC. This monitors the PON signal.

e. Connect digital voltmeter between TB3, pin 7(+) and TB3, pin 6(-, COMMON \\$\boxed{1}$/). This monitors the PFW signal.

f. Both PON and PFW should be at a high level (+4 volts dc, minimum).

g. Slowly reduce the ac input voltage to 160 volts while watching the PFW voltage. The PFW voltage should drop to approximately zero volts when the ac input voltage is between 170 and 160 volts. If the PFW voltage fails to drop to zero as the ac voltage is reduced below 170 volts, adjust A4R1.

Note

Each time PFW goes low, the circuit must be
reset by increasing the ac input voltage to
208 volts.

h. Set the ac input voltage to 160 volts. PFW and PON voltages
    should be low. Increase ac input voltage to 180 volts. PFW
    should go high, followed by PON after a 1-second maximum
    delay.

i. Reduce ac input voltage to zero. Set POWER switch to OFF
    position.

j. Unplug ac line cord from autotransformer. Plug into ac power
    receptacle.

9-18. HP 30310A Troubleshooting

Troubleshooting in the field is limited to visual checkout, volt-
age checks, alignment, and power supply replacement if necessary.
Proceed as follows:

a. Open rear door of cabinet and observe that the POWER switch
    is set to ON and that the indicator light is lighted. If the
    switch is ON but the indicator is not lighted, the indicator
    is defective or ac input power is not available to the power
    supply. Check the fuse.

b. With the power on, observe that the +5-volt red indicator is
    lighted. If it is not lighted, check that the DCE signal at
    terminal of TB3 is low and that the indicator is good.

c. Use the procedure in paragraph 9-13 to check the output volt-
    ages of the power supply at the test points furnished on the
    front panel. If the voltages are not correct, follow the
    appropriate adjustment procedure in paragraphs 9-14 through
    9-17.

d. Replace the power supply if it remains inoperative or does
    not respond to alignment.

9-19. HP 30311A OPERATION

The HP 30311A Power Supply (figure 9-1) provides the semiconduc-
tor memory with backup battery power during the absence of ac
input power. The volatile nature of dynamic MOS semiconductor
memory requires this backup power to prevent data loss. The pow-
er supply receives its input power from the HP 30310A Power Sup-
ply and, in turn, provides backup battery power for main memory.
The HP 30311A is a self-contained modular unit. It occupies one-
half of a standard 19-inch rack mounting as shown in figures 1-2
and 1-3. If only one power supply is mounted in the system, a
filler panel is supplied. The filler panel extends the full

width of the cabinet and has a 4 by 2-5/8 inch opening used for accessing power supply controls. When an HP 30312A Power Supply is mounted next to this power supply, a half panel covers only the HP 30311A Power Supply. The power supply provides +12.7V, +12.0V, +5.0V, -3.0V, and -5.0V for the semiconductor memory. The Semiconductor Memory Array PCA uses all the dc voltages except -3.0V. The Semiconductor Memory Control PCA uses only the +5.0V output.

The power supply normally operates from the +20V, -20V, and +15V outputs of the HP 30310A Power Supply as shown in figure 9-3. When ac input power to the HP 30310A Power Supply is interrupted, a 7-cell, lead-acid battery pack in the HP 30311A Power Supply furnishes the input power and maintains the voltages to the memory. The amount of time the battery continues to furnish power depends on the condition of the battery and the size of memory requiring power. Normally, the battery will furnish power for 40 to 90 minutes before it discharges to a level that activates the undervoltage circuits and removes all power from memory. The power supply can be operated without the battery. However, no power will be available to the memory system if ac power is removed from the HP 30310A Power Supply. When operating in this mode, the battery status lights are disabled. A battery mode switch on the 30311-60003 Control PCA must be set to operate without a battery.

The +20-volt output of the HP 30310A Power Supply provides the charging power for the battery pack and the input power for the +12.7V, +12.0V, and +5.0V regulators. (See figure 9-3.) The TEMP SENSE input to the HP 30310A controls the output of the +20V line. The +20V output also supplies input power to the +5V regulator and the control circuits.

Outputs of -3V and -5V are derived from a source voltage obtained from the +5V switching regulator. Load current from the -3V line adds to the -5V load current so that a single current limit on the -5V line protects both outputs. The maximum combined current for the -3V and -5V loads may be divided between the loads in any combination.

Over/under voltage detectors can disable the PSU (Power Supply Up) system dc power line, and shut down the power supply by sensing the +12.0V, +5V, and -5V outputs for undervoltage. The -3V output is sensed for undervoltage to disable the +12.0V and +12.7V outputs, thus protecting the memory array chips from lack of either -3V or -5V substrate bias. Overvoltage conditions sensed on the +5V and +12.0V outputs cause the power supply to crowbar to protect the TTL elements and memory array chips.

Controls and indicators mounted on the front panel of the HP 30311A Power Supply and on the DC Control Panel are provided to monitor and control power supply status and operation. In addition, test points on the power supply front panel are provided as an aid in troubleshooting. An internal three-position slide-switch permits power supply operation with a battery or without a
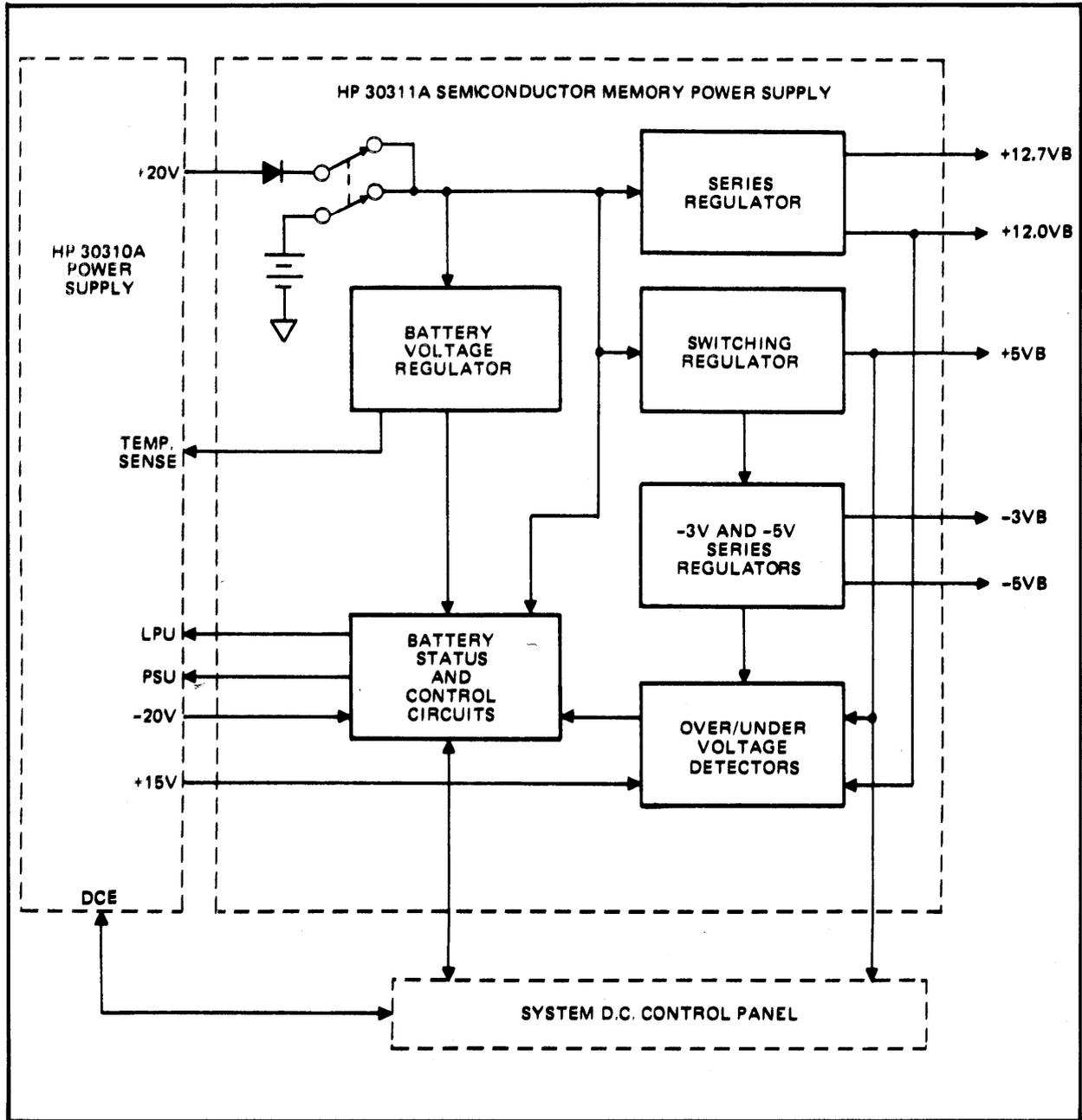
Figure 9-3. HP 30311A Power Supply Block Diagram

battery, and in the calibrate position allows setting of the bat-
tery float voltage level. Figure 9-1 illustrates the locations
of the controls and indicators on the power supply panels and on
the Dc Control Panel. Table 9-2 lists the function of each con-
trol and indicator. The BATTERY STATUS indicator is duplicated
on the power supply, on the Dc Control Panel, and on the top of
the cabinet door of the main bay. The power supply +5 indicator
is also duplicated on the Dc Control Panel as the SYSTEM DC POWER
indicator. (Complete specifications and detailed theory of opera-
tion for the power supply are contained in the HP 3000 Series
II/III Computer Systems Service Manual, part no. 30000-90018.)

Table 9-2. Hp 30311A Power Supply Controls and Indicators

Control or Indicator	Function
Power ON/OFF Toggle Switch	In the ON position, connects +20V from the HP 30310A Power Supply to the HP 30311A to maintain the charge on the battery and to develop required memory voltages. In the OFF position, disables the HP 30311A.
BATTERY TEST Momentary Toggle Switch (mounted on rear of unit)	Places power supply in a battery discharge mode for test purposes. (Simulates a power failure condition.)
RESET Pushbutton Switch	Resets the battery discharge mode, returning the power supply to normal operation.
+5 LED Indicator	When lit, indicates that +5V is being produced by the HP 30311A Power Supply
CROWBAR/BATT TEST LED Indicator	Used in conjunction with the BATTERY STATUS indicator to determine if the crowbar circuit has fired and shut down the power supply.
BATTERY STATUS LED Indicator	Indicates the battery conditions as follows: a. Remains continually lit for a fully-charged battery. b. Flashes at a 2-Hz rate when the battery is discharging. c. Flashes at a 0.5-Hz rate when the battery is charging. d. Remains off if battery is low or not present.

9-20. HP 30311A SERVICING INFORMATION

Preventive maintenance, adjustment, and replacement procedures for the power supply are contained in paragraphs 9-21 through 9-32. No high voltage points exist within the power supply. However, the supply is capable of supplying low voltage at moderate current levels. Use caution when manipulating metal tools or probes near exposed conductors and terminals. Extra care should be exercised to prevent the possibility of shorting the output lines of the battery pack.

CAUTION

When cables are connected to or disconnected from the power supply, the corresponding UPPER or LOWER MEMORY DC POWER switch on the System Dc Control Panel must be placed in STANDBY and the corresponding HP 30311A power-er switch must be in the off (down) position to prevent equipment damage.

9-21. Preventive Maintenance

Preventive maintenance is performed at monthly or bi-monthly intervals depending on the physical environment prevailing at the site. Preventive maintenance consists of measuring the dc voltages at the test jacks on the power supply front panel and performing the battery test procedure.

9-22. VOLTAGE CHECKS. Measure the five voltages listed in table 9-3 using a digital voltmeter. Allow the recommended warm-up period for the voltmeter before taking any measurements. If any voltage is out of tolerance, make the necessary adjustments as described in paragraphs 9-24 through 9-27.

Table 9-3. Dc Output Voltages

Voltage Test Jack	Indication
+5B	+5.1 +/- 0.1V
+12.7B	+12B + 0.7(+/-0.2)V
+12B	+12.0 +/- 0.1V
-3B	-3.0 +/- 0.25V
-5B	-5.0 +/- 0.2V

Note

If all voltages are out of tolerance, perform the +5.00-volt internal reference adjustment first. (Refer to paragraph 9-27.) Also, the +12.7 test jack voltage is equal to whatever voltge is measured at the +12 test jack plus 0.7 +/- 0.2 volts.

9-23. BATTERY TEST. The battery test certifies that the backup capability of the power supply is functioning normally. The system must be halted before performing this test. Proceed as follows:

a. On rear panel of the power supply, momentarily press the BATTERY TEST toggle switch. The CROWBAR/BATT TEST indicator should light and the BATTERY STATUS indicator should flash at a 2-Hz rate. Allow the battery to discharge for three to five minutes.

b. Return the power supply to normal operation by pressing the RESET pushbuton on power supply front panel.

c. The BATTERY STATUS indicator flashes at a 0.5-Hz rate until the battery is fully charged. Then, the indicator remains continually lighted.

9-24. HP 30311A Adjustments

The following adjustments should be performed after replacing a power supply or after replacing a circuit board or battery pack within the power supply.

9-25. BATTERY (FLOAT) VOLTAGE ADJUSTMENT. Float voltage must be adjusted whenever a battery pack is replaced.

> ┌──────────────┐
> │   CAUTION   │
> └──────────────┘

> The replacement battery pack must be at a stable known ambient temperature before the adjustment is performed. For a change in ambient temperature, the settling time for the pack is four to six hours. Failure to observe this precaution may considerably degrade the backup time and/or shorten the life of the battery pack.

Adjust the float voltage as follows:

a. Power down the system by placing the corresponding UPPER or LOWER MEMORY DC POWER switch on the System Dc Control Panel to STANDBY and place the corresponding HP 30311A Power Supply power switch off (down).

b. Remove the power supply from the cabinet and place on a suitable support.

c. Remove the top cover from the power supply.

d.   Connect a digital voltmeter between the +16.45V test point on the control board (figure 9-4) and a common ground point on the edge of the control board.

Note

Allow the digital voltmeter to warm-up before taking any measurements.

e.   Place control board switch S1 in position 1 (calibrate).

f.   Power up the system and allow five minutes for power supply circuits to stabilize.

g.   Refer to table 9-4 and determine the float voltage setting.

h.   Adjust potentiometer R35 on the control board (figure 9-4) for the voltmeter indication determined in the previous step. This assumes that the battery is almost fully charged. If it is not, the voltmeter indication will be low and gradually increase as the battery charges. A stable indication must exist before R35 can be satisfactorily adjusted.

i.   Set switch S1 to position 2 (normal).

j.   Power down the system and disconnect the digital voltmeter.

k.   Replace power supply top cover.
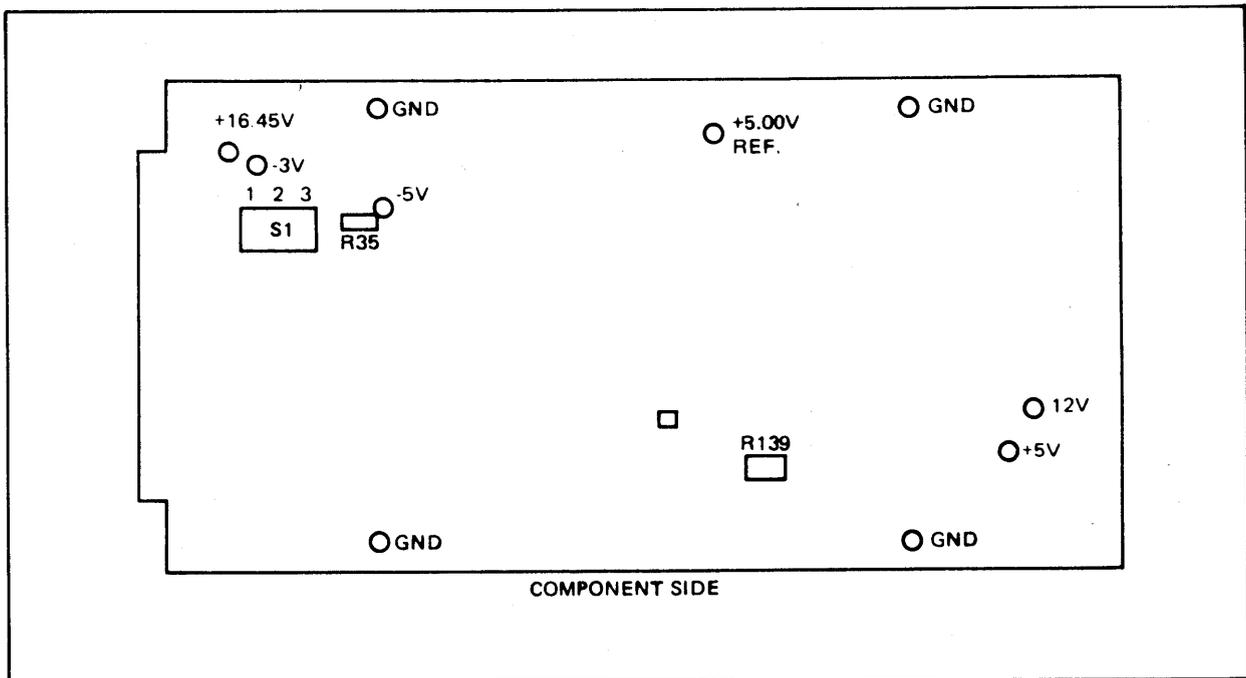
l.   Install the power supply into the cabinet.

Figure 9-4.   Control Board Adjustment Locations

Table 9-4. Float Voltage Versus Temperature*

Temperature (0 degree C)	Float Voltage	Temperature (0 degree C)	Float Voltage
0	17.10	28	16.26
1	17.07	29	16.23
2	17.04	30	16.20
3	17.01	31	16.17
4	16.98	32	16.14
5	16.95	33	16.11
6	16.92	34	16.08
7	16.89	35	16.05
8	16.86	36	16.02
9	16.83	37	16.09
10	16.80	38	15.96
11	16.77	39	15.93
12	16.74	40	15.90
13	16.71	41	15.87
14	16.68	42	15.84
15	16.65	43	15.81
16	16.62	44	15.78
17	16.59	45	15.75
18	16.56	46	15.72
19	16.53	47	15.69
20	16.50	48	15.66
21	16.47	49	15.63
22	16.44	50	15.60
23	16.41	51	15.57
24	16.38	52	15.54
25	16.35	53	15.51
26	16.29	54	15.48
27	16.29	55	15.45

*For seven-cell lead-acid battery

9-26. +12 VOLT ADJUSTMENT. The +12-volt output of the power sup-
ply is adjusted as follows:

a. Connect a digital voltmeter between the +12 and (common) test
    jacks on the power supply front panel.

Note

Allow the digital voltmeter to warm-up before
taking any measurements.

b. Adjust the +12ADJ control located on the rear panel of the power supply for a +12.0 (+/-0.1) V indication on the digital voltmeter.

9-27. +5.00 VOLT INTERNAL REFERENCE ADJUSTMENT. The +5.00-volt internal reference is adjusted as follows:

a. Power down the system by placing the corresponding UPPER or LOWER MEMORY DC POWER switch on the Dc Control Panel to STANDBY and place the corresponding HP 30311A Power Supply power switch off (down).

b. Remove the power supply from the cabinet and place on a suitable support.

c. Remove the top cover from the power supply.

d. Connect a digital voltmeter between the +5.00 volt internal reference test point on the control board (figure 9-4) and a common ground point on the control board.

Note

Allow the digital voltmenter to warm-up for the recommended warm-up period before taking any measurements.

e. Power up the system and allow five minutes for power supply circuits to stabilize. Measure the +5.00-volt internal reference voltage (+5.00 +/-0.005V).

f. If necessary, adjust R139 on the control board for a +5.00 +/-0.005V indication on the digital voltmeter.

Note

If an adjustment was necessary, the BATTERY (FLOAT) VOLTAGE adjustment procedure must also be performed.

g. Power down the system and disconnect the digital voltmeter.

h. Replace power supply top cover.

i. Install the power supply into the cabinet.

9-28. Replacement Procedures

9-29. POWER SUPPLY REPLACEMENT. The power supply is replaced by performing the following steps:

a. Power down the system by placing the corresponding UPPER or LOWER MEMORY DC POWER switch on the System DC Control Panel to STANDBY and place the corresponding HP 30311A Power Supply power switch off (down).

b. Disconnect plugs P2, P3, and P4 on the rear panel of the power supply.

c. Remove four screws on the front of the cabinet securing the power supply to the cabinet frame and the two screws on the right rear of the power supply. Remove the power supply by sliding it forward.

d. The power supply is installed using the reverse order of the above procedure.

9-30. BATTERY PACK REPLACEMENT. To replace a battery pack in the power supply, perform the following steps:

a. Remove the power supply from the cabinet by performing steps a through c of paragraph 9-29.

b. Remove the top cover from the power supply.

c. Disconnect jack J7 coming from the battery pack.

d. Remove the battery pack cover plate.

e. Remove the battery pack by lifting it out.

Install a new battery pack by using the reverse order of the above procedure. The battery float voltage adjustment which is described in paragraph 9-25 must be performed anytime a battery pack is replaced.

9-31. CONTROL PCA REPLACEMENT. The control board is replaced by performing the following steps:

a. Remove the power supply from the cabinet by performing steps a through c of paragraph 9-29.

b. Remove two screws holding the control board to the metal spacers on the top right side of the frame.

c. Disconnect plug P2 on the control board and disconnect the control board from the motherboard. Lift out the control board.

Note

The +5.00V internal reference voltage must be checked and the battery float voltage must be adjusted after replacing a control board.

The control board is reassembled by using the reverse order of the above procedure.

9-32. MOTHERBOARD PCA REPLACEMENT. To replace the motherboard with its associated heat sink, perform the following steps:

a. Power down the system by placing the corresponding UPPER or LOWER MEMORY DC POWER switch on the Dc Control Panel to STANDBY and place the corresponding HP 30311A Power Supply switch off (down).

b. Disconnect plugs P2, P3, P4, and P5 on the rear panel of the power supply.

c. Remove the power supply from the cabinet.

d. Remove top and bottom covers from the power supply.

e. Free the motherboard by removing four screws, two from the top and two from the bottom edges of the rear panel frame.

f. Disconnect plug P6 on the motherboard.

g. Slide the two boards out of the rear of the power supply frame.

h. Remove the control board from J1 and the motherboard.

Note

Steps i and j should be performed when the motherboard and heat sink are to be replaced.

i. Remove four screws holding the motherboard to the rear panel.

j. Remove four screws securing the heat sink to the rear panel.

The motherboard and heat sink are replaced by using the reverse order of the above procedure. Power supply output voltages should be checked after replacing the motherboard.

9-33. HP 30312A OPERATION

The HP 30312A Power Supply (figure 9-1) provides up to 100 amperes of 5-volt power for operation of the interface PCA´s in the I/O area of the computer. The power supply is an HP 62605M Power Supply modified by adding a power switch, fuse, ac receptacle, and a threshold adjustment circuit. The threshold adjustment circuit is adjusted when the system is configured to provide overcurrent shutdown of the HP 62605M Power Supply when the drain on the power supply exceeds a predetermined amount by ten amperes.

The following discussion covers only the items and circuits added to the HP 62605M Power Supply. The HP 62605M Power Supply is described in Modular Power Supplies, L and M Series, Models 62605L, 62605M, and 62615M Manual, part no. 5950-1756. The Model 62605M is equipped with Option 106 which enables the power supply to operate on 187 to 250 volts ac input power. The HP 30312A Power Supply provides 5-volt power over a current range nominally

15 to 100 amperes. It augments the approximate 55-ampere, 5-volt output of the HP 30310A Power Supply which powers the CPU/IOP, the HP 30311A Power Supply, and portions of the computer I/O area.

Figure 9-5 illustrates the major functional areas of the HP 30312A Power Supply, Interface Board, part no. 30312-60002. The model 62605M Power Supply is also shown to illustrate the relationship between the power supply and the interface board. The interface logic is required to properly sense and control the 62605M Power Supply with the existing HP 30310A Power Supply system dc control lines (DCE and PSU). Input power of 208/230 VAC is applied to the power supply and interface board by POWER switch S3 through the eight-ampere fuse. An internal power supply provides the operating potentials for the circuits including the reference voltage (adjusted to 4.75 volts by potentiometer R52) used by several functional circuits on the interface board.

9-34. Overcurrent Protection

The operating current configurator monitors the +5V load on the 62605M Power Supply and is activated whenever the load increases 10 amperes above the normal load. When switch S2 is pressed, ADJ potentiometer R32 is adjusted at the normal steady state current level. When S2 is released, the threshold is automatically incremented to sense a current 10 amperes above the nominal value set with R32. Transient overcurrents exceeding the 10-ampere margin cause the over current indicator (located above switch S2) to light for the transient time interval. The two-second timer prevents the power supply from shutting down when overloads are less than two seconds long. When such overloads last longer than two seconds, the two-second timer sets the latch (causing the overcurrent indicator to light continually) and shuts down the 62605M Power Supply output by pulling the Al terminal control voltage below its turn on value. The Cal/Test switch S1 and Cal Adj potentiometer R26 are used at the factory to calibrate the operating current configurator to the resistance of the high current wires connected to the power supply. These controls are normally not used in the field and should be left as they are.

9-35. Undervoltage Protection

The undervoltage detector, connected to the +S (+sense) terminal of the power supply, sets the Power Supply Up (PSU) signal low (1V) to the system whenever the output voltage at the load drops below 4.61V. The PSU signal is restored to the open circuit condition whenever the output voltage at the load rises above 4.81V.

9-36. Power Failures

An ac power failure sensed by the HP 30310A Power Supply causes a Power Fail Warning (PFW) signal to be sent to the CPU. Therefore, the HP 30312A does not require a power failure detector. Maximum utilization of this detection capability can only be obtained when the HP 30310A and HP 30312A are supplied from the same ac power source.

VOLTAGE SENSE LEADS

62605M
POWER
SUPPLY

5V @ 100A

+

−

4 − NO. 12 CONDUCTORS

4 − NO. 12 CONDUCTORS

NOTE 1

LOAD

NOTE 1

−5

+5

A1

AC   ACC   GRD

POWER

ON

OFF

AC

ACC

GRD

8A

208/230
VAC
±10%

SYSTEM DCE

TURN-OFF
DELAY
(0.15 SEC)

NOTE 2

TTL DCE

TTL INPUT
LEVEL ON/OFF
CONTROL

+15V   −19V

INTERNAL
POWER
SUPPLY

+4.75V
PRECISION
REFERENCE

R51
REF
ADJ

OPERATING
CURRENT
CONFIGURATOR

R26
CAL
ADJ

ADJ
R32

S1
CAL/TEST

UNDER
VOLTAGE
DETECTOR

+5V
INDICATOR

2 SEC
TIMER

LATCH

S2

10 AMP
INC/DEC

OVER CURRENT
INDICATOR

PSU

NOTES:
1. THIS CONNECTION PHYSICALLY LOCATED
   AT THE END OF THE NO. 12 AWG CONDUCTORS
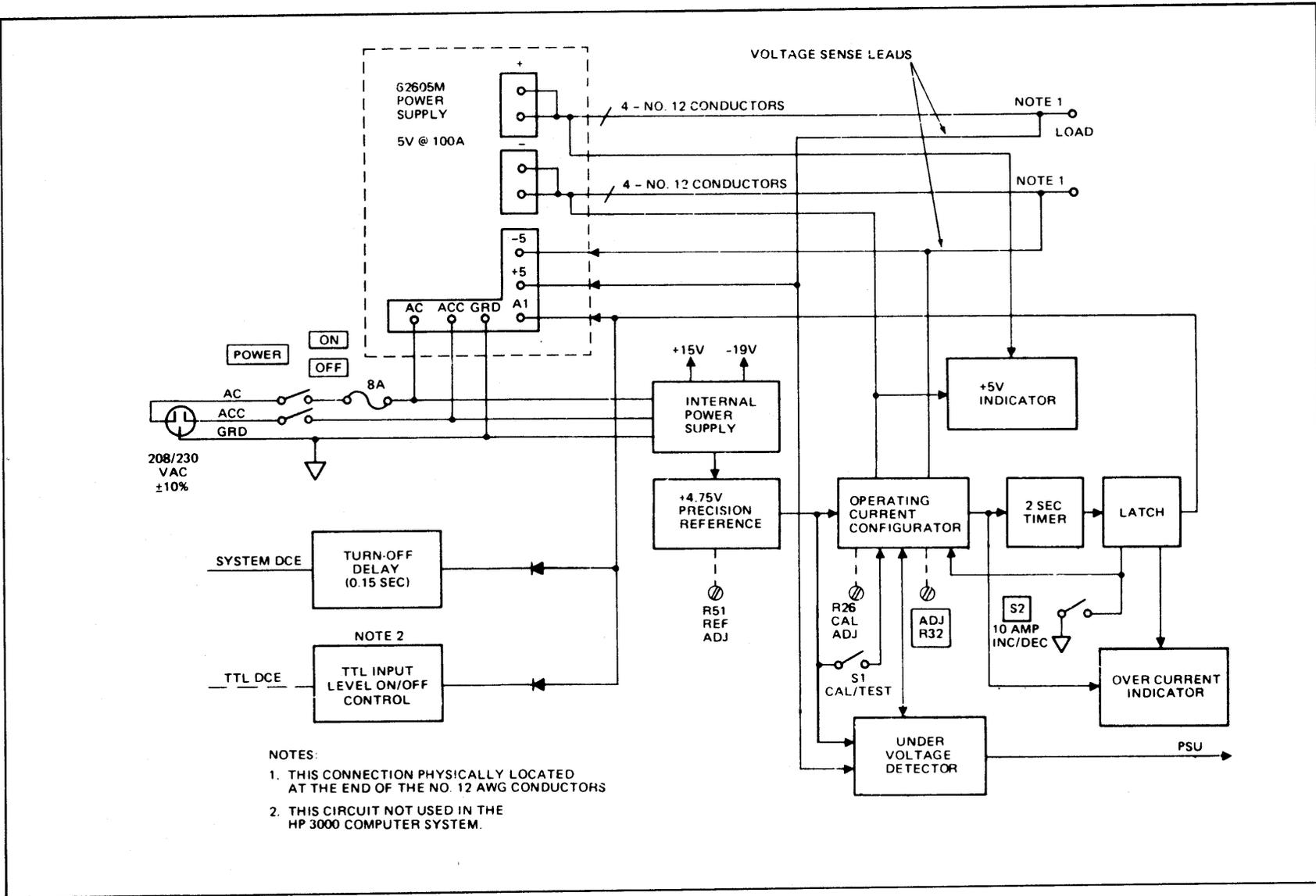2. THIS CIRCUIT NOT USED IN THE
   HP 3000 COMPUTER SYSTEM.

Figure 9-5. HP 30312A Power Supply Block Diagram

9-37. Dc Enable

The DCE signal is the same control line connected to the HP 30310A DCE terminal. It turns on (DCE at gnd) and turns off (DCE approximately +10V) the +5V output of the HP 30312A. The interface board receives the DCE signal and generates the appropriate signal to the A1 control terminal on the 62605M Power Supply. The interface board circuits delay the turnoff of the +5V line after release of the DCE signal by about 0.15 second to allow ample time for the HP 30310A Power Supply to generate its PFW signal for the CPU. This delay guarantees the +5V output will be present for at least 1.0 msec after the CPU receives the PFW signal.

9-38. HP 30312A SERVICING INFORMATION

WARNING

> While the input power is connected, use caution when working inside the power supply. Many exposed conductors carry low dc voltages which are capable of supplying heavy currents if shortcircuited, resulting in high heat and the possibility of painful burns. Use caution when manipulating metal tools or probes. A wrist watch, or a metal necklace, bracelet, or ring must not be worn. Avoid dropping tools, screws, or other metal objects onto conductors. Remove power and recover dropped objects at once; if forgotten, damage could result later. Ac power line voltage is exposed when covers are removed. Exercise extreme caution when working in the power supply with covers removed, and never work under this condition unless another person is nearby and within sight.

The HP 30312A Power Supply is a nonrepairable unit and, if found defective, it must be replaced. Except for replacing an open fuse or the interface board (30312-60002), no repair procedures are required. If the computer system is down because the 5-volt output of this power supply is absent, disconnect the wire at terminal A1 of the HP 62605M Power Supply. If power is restored, the interface board is defective. If power is not restored, the HP 62605M unit is defective.

NOTES

NOTES

This section contains servicing information for the HP 32435A Series III power supplies.

10-1. INTRODUCTION

The HP 32435A Series III Computer System has a maximum of five power supplies; four power supplies in the CPU Bay and, if the optional I/O Bay is installed, one power supply in the I/O Bay. The HP 62605M-P31/P41, HP 63312F-P02, HP 63312F-P09, and HP 61315D-P07 Power Supplies are located in the CPU Bay as shown in figures 1-4 and 1-5. A second HP 62605M-P31/P41 is located in the I/O Bay of 2-Bay models as shown in figure 1-5.

10-2. POWER SUPPLY TROUBLESHOOTING

The HP 32435A Series III Computer System power supplies are all controlled and monitored by the Power Supply Control and Display Assembly located at the lower front of the CPU Bay as shown in figures 1-4 and 1-5. The Power Supply Control and Display Assembly (figure 10-1) contains twelve test points (including a common ground) and associated LED indicators that monitor all the system DC voltages provided by the power supplies. The test points are connected directly to the backplane voltages through a 1K resistor and can be used to measure any system DC voltage without entering the rear of the equipment bay. The test point DC voltages, associated power supplies, and purposes are listed in table 10-1. It should be noted that the LED indicators located below each test point do not "follow" their respective power supplies and that their status (lighted or not lighted) has no meaning except during a power supply failure. (Refer to the following text and table 10-1 for additional information.)

In addition to the DC VOLTAGES test points and indicators, the Power Supply Control and Display Assembly also contains three DC STATUS indicators and two DC POWER switches located as shown in figure 10-1. The functions of these indicators and switches are listed in table 10-2.

If the DC POWER LOGIC switch is set to DISABLE, a power failure is simulated so that the contents of memory will not be destroyed and first the PFW indicator and then the PON indicator will no longer be lighted. In addition, the DC VOLTAGES indicators will be reset to their lighted status.

When the DC POWER switches are set to ENABLE, the PFW indicator will light immediately if the system's input AC power is within specifications. After approximately one or two seconds, the PON indicator will light if all the system's DC voltages are present.

If the PON indicator does not light, one or more of the DC volt-
ages has failed and the DC VOLTAGES indicator(s) that is lighted
indicates the DC voltage that has failed. (Use table 10-1 to de-
termine the power supply source.) If this occurs, all power sup-
plies except memory backup are shut down and their supply volt-
ages are not available at the DC VOLTAGES test points.

If one or more of the DC voltages fail during normal system opera-
tion, the PON indicator will no longer be lighted and the faulty
DC voltage(s) will be indicated by the lighted DC VOLTAGES indi-
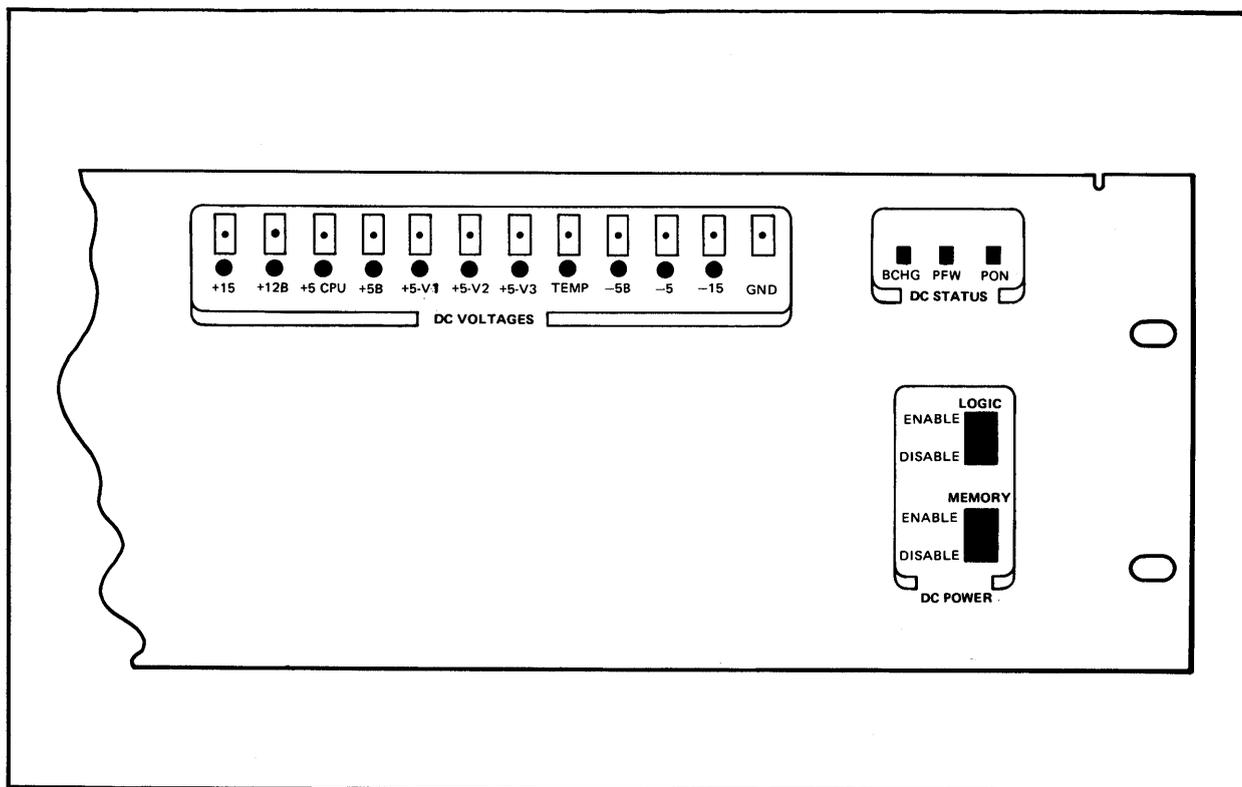cator(s). The PFW indicator will remain lighted.

Figure 10-1. Power Supply Control and Display Assembly

Table 10-1. DC Power Supply Specifications

| LED and Test Point | Voltage Range | | Power Supply Source | Supplies DC Voltage To |
	Minimum	Maximum		
+15	+14.5V	+16.5V	HP 63312F-P02	Card Cage 3-6
+12B	+11.9V	+12.1V	HP 61315D-P07	Card Cage 2-6
+5 CPU	+5.17V	+5.17V	HP 62605M-P31	Card Cage 1-2
+5B	+5.0V	+5.25V	HP 61315D-P07	Card Cage 2
+5-V1	+5.0V	+5.25V	HP 63312F-P02	Card Cage 3
+5-V2*	+5.0V	+5.25V	HP 63312F-P09	Card Cage 5
+5-V3*	+5.0V	+5.25V	HP 62605M-P31	Card Cage 5-6
TEMP	N/A	N/A	See Note	See Note
-5B	-5.0V	-5.25V	HP 61315D-P07	Card Cage 2-6
-5	-4.5V	-5.5V	HP 63312F-P09	Card Cage 3-6
-15	-14.5V	-15.5V	HP 63312F-P02	Card Cage 3-6

Note: The sense line for TEMP goes through a thermal switch in
the equipment cabinet fan panel to +5-V2 in Card Cage
No. 4. High temperatures open the thermal switch which
simulates a power failure. If +5-V2 fails, the +5-V2
and TEMP LED's may both be lighted. However, the TEMP
LED may remain lighted with the +5-V2 LED not lighted.

*On 1-Bay Models +5-V3 is internally connected to +5-V2
and, if +5-V2 fails, one or both +5-V2 and +5-V3 LED's
will be lighted.

10-3. POWER SUPPLY ADJUSTMENTS

All HP 32435A Series III Computer System power supplies have vol-
tage and current limit adjustments located at the rear of each
power supply. The HP 63312F-P02 and HP 62605M-P31 power supplies
are accessed by opening the rear door of the equipment bay(s).
The HP 63312F-P09 and HP 61315D-P07 power supplies are accessed
by opening the rear door of the equipment bay and then removing
the fan panel located at the bottom of the bay. The current lim-
it adjustments should never be attempted and voltage adjustments
should only be performed when a particular voltage is out of tol-
erance.

Table 10-2.   DC STATUS/POWER Indicators and Switches

Indicator or Switch	Function
DC STATUS BCHG Indicator	When fully lighted, indicates that the system's battery is fully charged. When slowly blinking, indicates that the battery is charging. When rapidly blinking, indicates that the battery is discharging. When not lighted, indicates that the battery is completely discharged or that the battery is not installed in the system.
DC STATUS PFW Indicator	When lighted, indicates that the system's input AC power is within specifications. When not lighted, indicates that the AC power has dropped below specifications and that DC power from the power supplies may begin to decay.
DC STATUS PON Indicator	When lighted, indicates that AC power is applied to the power supplies and that the power supplies are operating properly. When not lighted, indicates that one or more power supplies are not operating properly or that the DC voltages from the power supplies has begun to decay due to loss of input AC power.
DC POWER LOGIC Switch	When set to DISABLE, removes DC power from all PCA's except memory to permit removal and replacement of the I/O and CPU PCA's without destroying the contents of memory. When set to ENABLE, applies DC power to all PCA's except memory PCA's.
DC POWER MEMORY Switch	When set to DISABLE, removes all DC power from all PCA's. When set to ENABLE, applies DC power to all PCA's provided that the LOGIC switch is also set to ENABLE.

```
┌─────────────┐
│   CAUTION   │
└─────────────┘
```

Never attempt any current limit adjustments in the
field. These limits are correctly set at the fac-
tory prior to the system's shipment and must never
be changed. Failure to observe this caution may
result in serious damage to the computer system.

The power supply output voltages should be checked from the Power
Supply Control and Display Assembly DC VOLTAGES test points after
the system is installed and during each scheduled preventive
maintenance interval. If any measured voltage is not within the
limits specified in table 10-1, adjust the associated power sup-
ply to bring its output voltage within the specified limits. It
should be noted that the +15- and -15-volt outputs of the HP
63312F-P02 power supply are controlled by the same adjustment.
If either voltage is adjusted, then the other voltage must be
checked and possibly readjusted.

During any power supply adjustment, perform the adjustment slowly
to avoid an overvoltage condition that will shutdown the system.
If an overvoltage condition does occur, rotate the voltage ad-
justment counterclockwise a small amount, reset the system, and
slowly rotate the adjustment clockwise until the output voltage
is within its specified limits.

10-4. REPAIR AND REPLACEMENT

The HP 32435A Series III Computer System power supplies are non-
repairable units and, if found defective, must be replaced. Ex-
cept for replacing open fuses, no repair procedures are required.
All the power supplies are removed from the front of the equip-
ment bay(s) after first removing and labeling all wires attached
to the rear of the supplies.

```
┌─────────────┐
│   CAUTION   │
└─────────────┘
```

Replacement power supplies may be shipped with a
strap on the terminal board for 120V operation.
This strap must be removed for 220V operation or
the power supply will be damaged when power is
first applied to the computer system.

Early models of the HP 32435A Series III Computer System contain
one or two HP 62605M-P31 power supplies; later models contain the
HP 62605M-P41 power supply. In countries where VDE Certification
is required, the HP 62605M-P31 can only be replaced by another HP

62605M-P31 or an HP 62605M-P41. Elsewhere, the HP 62605M-P31 can be replaced by another HP 62605M-P31, an HP 62605M-P41, or by an HP 30312A (Section IX). In order to replace an HP 62605M-P31 with an HP 30312A, the Overcurrent Adjust PCA Module and associated sheet metal atttached to the rear of the HP 30312A must first be removed.

NOTES

This section contains information for installing the HP 3000 Series III Computer Systems and ensuring that the systems are operating in accordance with factory specifications. It should be noted that before a system can be installed, a site must be prepared in accordance with the information contained in the applicable Computer System Site Preparation Manual and Computer System Site Planning Workbook. (For HP 32421A Series III Computer Systems, refer to the HP 3000 Computer System Site Preparation Manual, part no. 30000-90082 and the HP 3000 Computer System Site Planning Workbook, part no. 30000-90086. For HP 32435A Series III Computer Systems, refer to the HP 3000 Computer System Site Preparation Manual, part no. 30000-90145 and the HP 3000 Computer System Site Planning Workbook, part no. 30000-90146.) It should also be noted that site preparation is the responsibility of the customer and not Hewlett-Packard's unless it has been negotiated as a separate contract. For additional installation information, refer to the applicable HP 3000 Computer System Installation Manual (part no. 30000-90081 for the HP 32421A Series III Computer Systems and part no. 30000-90147 for the HP 32435A Series III Computer Systems).

This section is divided into two parts. Part One (paragraphs 11-1 through 11-10) contains information strictly for the HP 32421A Series III Computer System. Part Two (paragraphs 11-11 through 11-17) contains information strictly for the HP 32435A Series III Computer System.

PART ONE
HP 32421A SERIES III COMPUTER SYSTEM INSTALLATION

11-1. EQUIPMENT BAY INSTALLATION

WARNING

Do not attempt to slide or swing out any equipment from the bays or cabinets until they are completely installed, the Anti-Tip Base Extensions extended, and cabinet feet lowered. Failure to comply may result in serious injury or death and severe damage to equipment.

Before performing the installation instructions contained in paragraphs 11-2 through 11-4, mechanically join the equipment bays in accordance with the instructions contained in the HP 3000 Computer System Installation Manual, part no. 30000-90081. Then, looking at the equipment bays from the rear, note that the Power Control Module (PCM) is at the bottom of Bay No. 1, the Power Control Unit (PCU) is at the bottom of Bay No. 2, and that a Power Distribution Unit (PDU) is in each equipment bay except for Bay No. 1. (Refer to figures 1-2 and 1-3.) It should be noted that special order systems may consist of more than three equipment bays.

11-2. Power Distribution Unit

Perform the following instructions for each PDU in the system.

a. Ensure that no system ac power cables are connected to any ac power source and that the MAIN SYSTEM POWER circuit breaker on the PCM, the EXTENDED SYSTEM POWER circuit breaker on the PCU, and all ac power switches on power supplies are OFF.

b. Remove the access plate from each PDU.

c. Connect or check straps (0360-1571) between TB1 terminals in each PCU according to the cabinet in which the PCU is located as listed in table 11-1 and shown in figure 11-1. The PDU must be connected to the proper phase according to where the cabinet is located in the system.

d. Connect the five-wire harnessed cable between the PCM in the CPU Bay (Bay No. 1) and the PDU in the I/O Bay, according to table 11-2.

e. Connect PDU's in adjacent bays (if any) through the five-wire harnessed cable(s) furnished, to join TB1 terminals 1 to 1, 3 to 3, 5 to 5, 12 to 12, and Earth Bus Bar to Earth Bus Bar. (See figure 11-1.)

f. Check the ac power service strip connections and phase strapping to TB1 of the PDU in the I/O bay as shown in figure 11-1 and tables 11-1 and 11-3. Check the ac power service strips in the other bays (if applicable) in the same manner.

g. Leave the access plates for each PDU off and proceed to paragraph 11-3.

Table 11-1. PDU Strap Connections at TB1

Ac Input Voltage					
120/208, 60 Hz				230, 50 Hz	
Bay Number (Counted From Rear Left)	PDU TB1 Strap Connections	Phase Connected		Bay Number (Counted From Rear Left)	PDU TB1 Strap Connections
		115V	208V		
2	2-3 4-5 6-7 8-9 9-10	* B	* B,C	2,3,4	1-2 5-6 9-10
3,6,9	1-2 4-5 7-8 8-9 10-11	C	Not Used	NA	NA
4,7,10	1-2 3-4 6-7 8-9 9-10	A	Not Used	NA	NA
5,8,11	2-3 4-5 6-7 8-9 10-11	B	Not Used	NA	NA

*In 2-Bay Model Systems, the 220V service strip is not used in cabinet 2. In 3-Bay Model Systems, the 115V service strip is not used in cabinet 2.

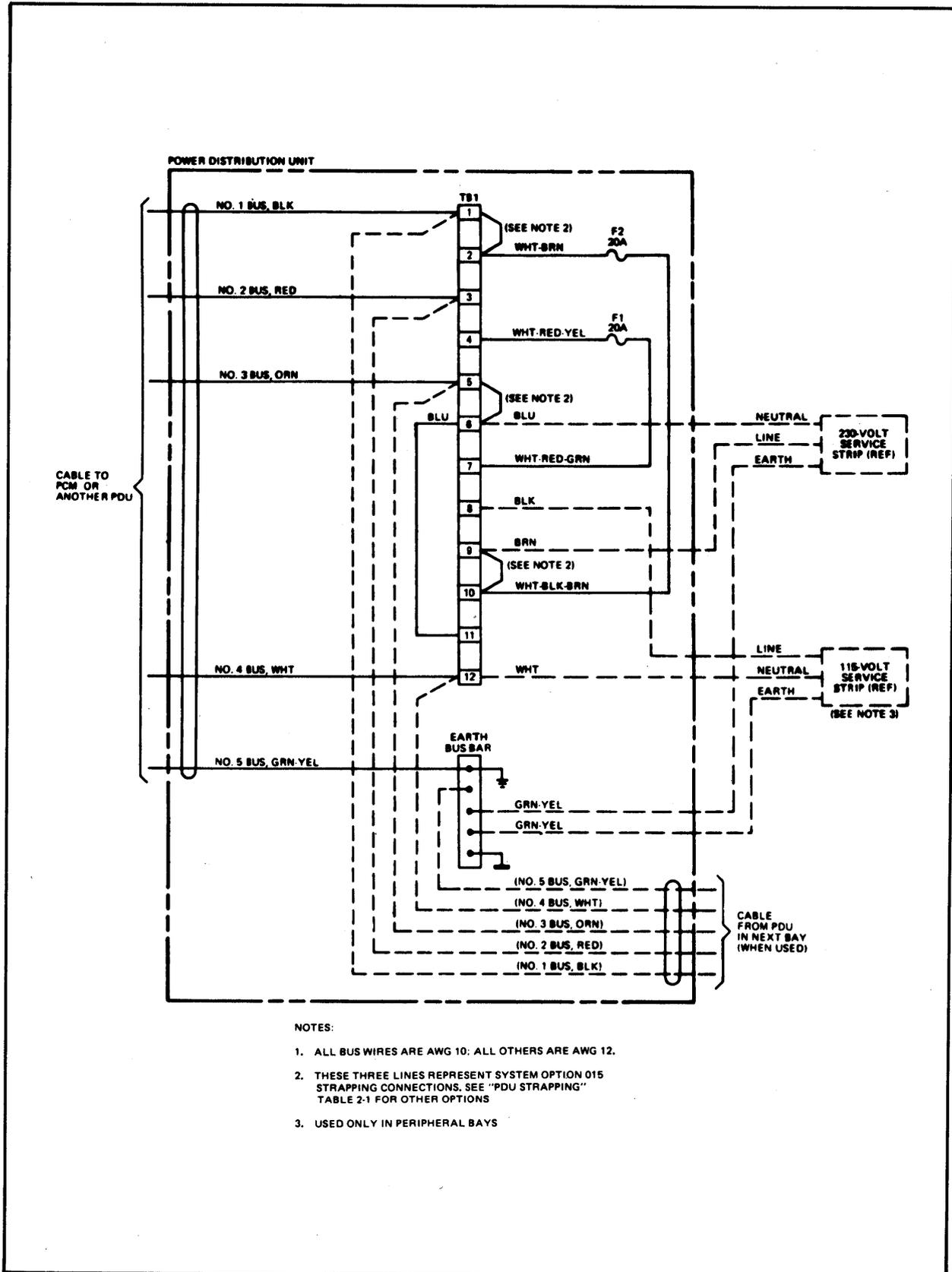NOTE: The 208V service strip in Bay No. 1 is connected to phases A and B in the PCM.

Figure 11-1.   PDU Schematic Diagram

Table 11-2. PDU to PCM Connections

Wire Color	PDU at TB1	PCM at TB3
Black	1	6
Red	3	7
Orange	5	8
White	12	9
Green-Yellow	Earth Bus	Earth Bus

Table 11-3. PDU Ac Service Strip Wiring

Service Strip Identity	Wire Color	PCU Terminal
115V	Black	TB1 Terminal 8
	White	TB1 Terminal 12
	Green-Yellow	Earth Bus-Bar
230V (CCE-22)	Brown	TB1 Terminal 9
	Blue	TB1 Terminal 6
	Green-Yellow	Earth Bus-Bar

11-3. Power Control Unit

If the system does not contain a PCU, proceed to paragraph 11-4. Otherwise, perform the following instructions.

a. Remove the large and small panels (seen from the rear of the cabinets) from the PCU. Save all hardware.

b. Verify that jumpers have been installed correctly on TB1 as listed within figure 11-2.

c. Have an electrician perform this step. Connect primary ac power cable wires to the PCU line filter terminals as shown in figure 11-3. Then, replace the panels removed in step a.

d. Remove all ac power plugs from all ac power service strips in the equipment bays.

e. Use the PCU-to-PCU/PCM Interconnecting Cable furnished with each PCU to connect J1 of the PCU to J2 of the PCM in the CPU Bay (para 11-4). Figure 11-4 shows the wiring of the cable.

f. Leave the access plates off and proceed to paragraph 11-4.

Figure 11-2. PCU Schematic Diagram

HP 30330A POWER CONTROL UNIT (PCU)

CB1 40A (SEE NOTE 7)

TB1 (SEE NOTE 2)

LINE FILTER FL1

(SEE NOTE 1)

EARTH

(SEE NOTE 5)

P1 J1

P2 J2

(SEE NOTE 3)

P2

FROM J2 OF PCM OR PREVIOUS PCU (SEE NOTE 4)

EARTH BUS BAR

TO G-LINE OF SERVICE STRIPS (LOCAL) AND TO PDU IN NEXT BAY

TO EQUIPMENT DC RETURN

F1 F2 F3

F4 F5 F6

F7, 6A

J3

	120-208V 3 PH 60 HZ	230V 1 PH 50 HZ
A	PHASE A	PHASE A
B	PHASE B	PHASE A
C	PHASE C	PHASE A
D	OPEN	NEUTRAL
E	PHASE C (6A)	OPEN
F	NEUTRAL	OPEN
G	EARTH	EARTH

DISC DRIVE POWER

115 VOLT CONVENIENCE RECEPTACLE

EARTH BUS

J4

	120-208V 3 PH 60 HZ	230V 1 PH 50 HZ
A	PHASE A	PHASE A
B	PHASE B	PHASE A
C	PHASE C	PHASE A
D	OPEN	NEUTRAL
E	PHASE C (6A)	OPEN
F	NEUTRAL	OPEN
G	EARTH	EARTH

DISC DRIVE POWER

115 VOLT CONVENIENCE RECEPTACLE

EARTH BUS

TB2

NOTES

1. INPUT TERMINALS ON FL1 ARE NUMBERED, AND TABLE A IS GIVEN, FOR REFERENCE ONLY. POWER INPUT CONNECTIONS ARE TO BE MADE IN ACCORDANCE WITH INSTRUCTIONS IN THE CABINET MANUAL.

2. TERMINAL BLOCK STRAPPING IS NOT SHOWN. STRAPPING CONNECTIONS ARE PER TABLE B.

3. TERMINATING PLUG P2 OBTAINED FROM J2 OF PCM OR NEIGHBORING PCU.

4. P2 OF INTERCONNECTING CABLE MATES WITH J2 OF PCM OR NEIGHBORING PCU.

5. WIRES A, C, S, AND T ARE PART OF CB1. WIRES A AND T ARE SPLICED.

6. FUSES F1 THRU F6 ARE RATED 20A.

7. CB1 IS RATED 40A BY MANUFACTURER. IT IS DE-RATED TO 30A FOR THIS APPLICATION.

TABLE A. POWER INPUT CONNECTIONS

TERMINAL	120/208V, 3 PH, 60 HZ	230V 1 PH 50 HZ
1	PHASE A	PHASE A
2	PHASE B	NO CONN
3	PHASE C	NEUTRAL
4	NEUTRAL	NO CONN
5	EARTH	EARTH

TABLE B. STRAPPING CONNECTIONS

TERMINAL BLOCK	120/208V, 3 PH, 60 HZ	230V 1 PH 50 HZ
TB1	1 2	2 3
TB1	4 5	3 4
TB1	6 7	7 8
TB1	9 10	8 9

A. 120/208V, 3ph, 60-Hz

B. 230V, 1-ph, 50-Hz (System Option 015)

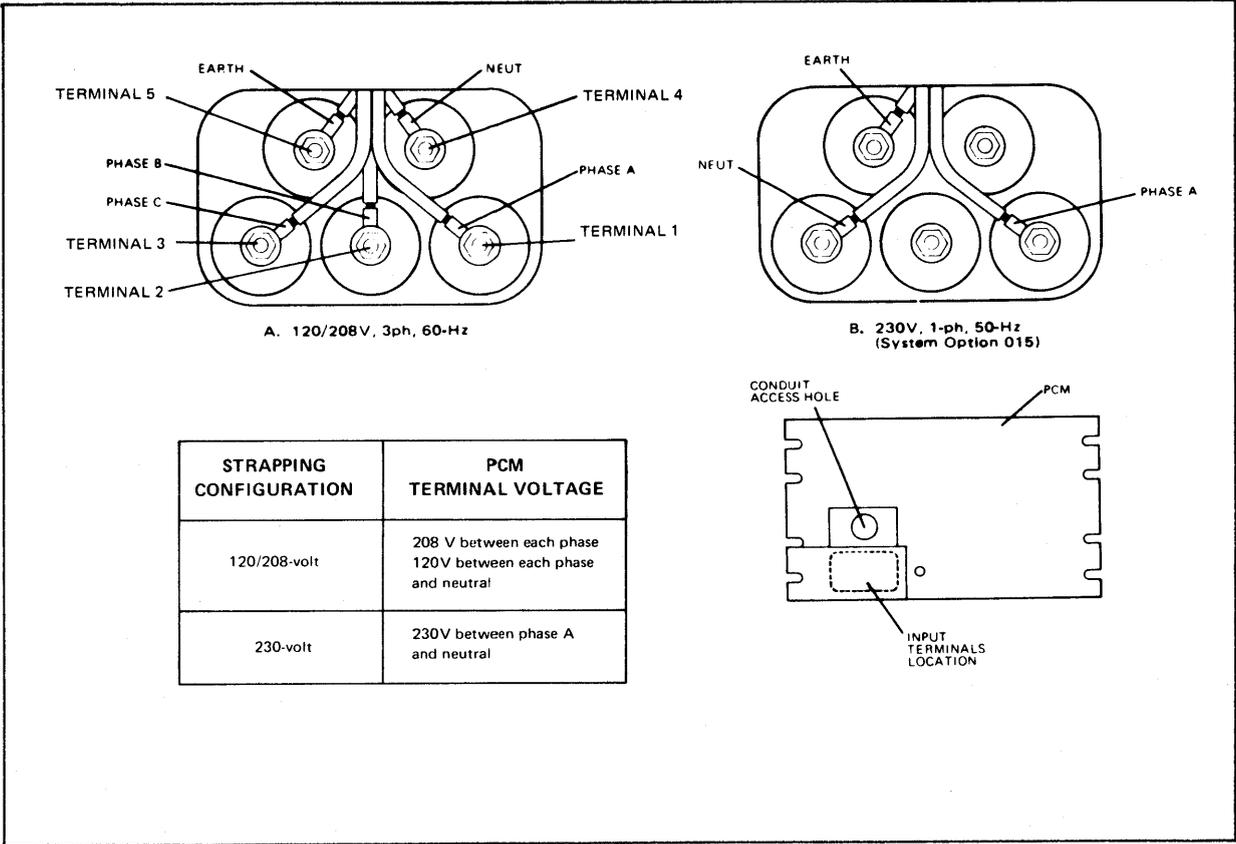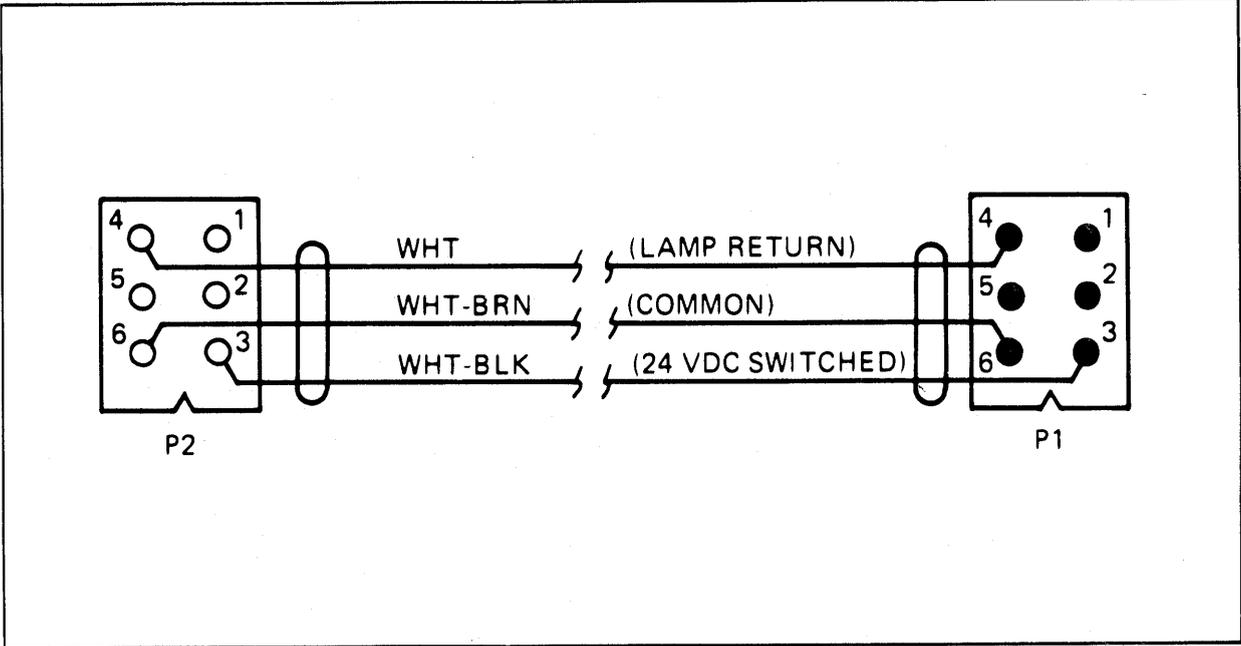| STRAPPING CONFIGURATION	PCM TERMINAL VOLTAGE
120/208-volt	208 V between each phase 120V between each phase and neutral
230-volt	230V between phase A and neutral

Figure 11-3. PCU/PCM Line Filter Connections

Figure 11-4. PCU To PCU/PCM Interconnecting Cable

11-4. Power Control Module

Perform the following instructions for the PCM.

a. Remove the large and small panels (seen from the rear of the CPU Bay) from the PCM. Save all hardware.

b. Verify that jumpers have been correctly installed on TB1 and TB2 as listed within figure 11-5.

c. Have an electrician perform this step. Connect primary ac power cable wires to the PCM line filter terminals as shown within figure 11-3.

d. Remove all ac power plugs from all ac power service strips in the equipment bays.

e. Have an electrician perform this step. Be sure the ac power at the Computer Mainframe Power Panel and PCM MAIN SYSTEM POWER and PCU EXTENDED SYSTEM POWER circuit breakers are turned OFF. Then, connect the other end of the primary ac power cable wired in step c to the isolation transformer. Then, turn ON the Computer Mainframe Power Panel main line.

f. Use a suitable voltmeter to check voltages at each of the PCM line filter terminals for correct value as listed within figure 11-3.

g. Use a voltmeter to check the voltage between ground and neutral. Voltage should not exceed 1 VAC (rms). Use an oscilloscope to check the ripple content of the waveform between ground and neutral (must not exceed 25 mv p-p).

h. Check that the EMERGENCY OFF pushbutton at the top front of the CPU Bay is lighted. If it is not, press it once. If it fails to light, an error exists in that circuit. Do not proceed further until the error has been corrected and the pushbutton will light.

i. After confirming that the EMERGENCY OFF lamp lights, set the PCM MAIN SYSTEM POWER circuit breaker ON. Then, check the ac power voltages at the ac power service strips throughout the cabinets as shown in figure 11-1. If the voltages are incorrect, set the PCM MAIN SYSTEM POWER and Computer Mainframe Power Panel circuit breakers OFF and check the ac power service strip within the PDU against table 11-3. If any errors are found, correct them.

j. Set the PCM MAIN SYSTEM POWER circuit breaker to OFF and replace the access covers on all PDU's.

k. Connect all fans, card cages, and other devices in the cabinets into the appropriate ac power service strips and set the PCM MAIN SYSTEM POWER circuit breaker to ON. Check that all fans in the system operate.

l. Press and release the EMERGENCY OFF pushbutton and check for three results:

   (1) The pushbutton lamp goes out.

   (2) All fans in the card cages, cabinets, and other devices stop.

   (3) All circuit breaker handles on the PCM and, if present, all PCU's, move to the OFF (down) position.

m. Set the circuit breaker in the Computer Mainframe Power Panel to OFF. Set the circuit breaker in the Computer Peripheral Equipment Power Panel to OFF.

n. If step l is completed successfully, continue this procedure. Otherwise, go no further until the fault is corrected.

o. Continue installation by connecting the interrupt poll, data poll, MCU clock signal distribution, and flat cables as described in paragraphs 11-5 through 11-6. Then, continue by returning to step p.

p. Do not turn ON the Computer Mainframe Power Panel or the Computer Peripheral Equipment Power Panel ac power until instructed to do so in paragraph 11-8.

q. Leave all equipment bay rear doors off.

11-5. Bus Cable Connections

If the system was shipped with the equipment bays separated, three flat cables for the IOP bus, Multiplexer Channel bus, and Power Bus must be connected between the equipment bays. Step-by-step instructions for installing the bus cables are contained in the HP 3000 Computer System Installation Manual, part no. 30000-90081.

11-6. Interrupt Poll, Data Poll, and MCU Clock Connections

If the system was shipped with the equiment bays separated, the interrupt poll cable (one white and one blue twisted-pair cable), Multiplexer Channel data poll cable (one white and one orange twisted-pair cable), and MCU clock cable (gray coaxial cable) must be connected between the equipment bays. Step-by-step instructions including cabling diagrams for installing these cables are contained in the HP 3000 Computer System Installation Manual, part no. 30000-90081.
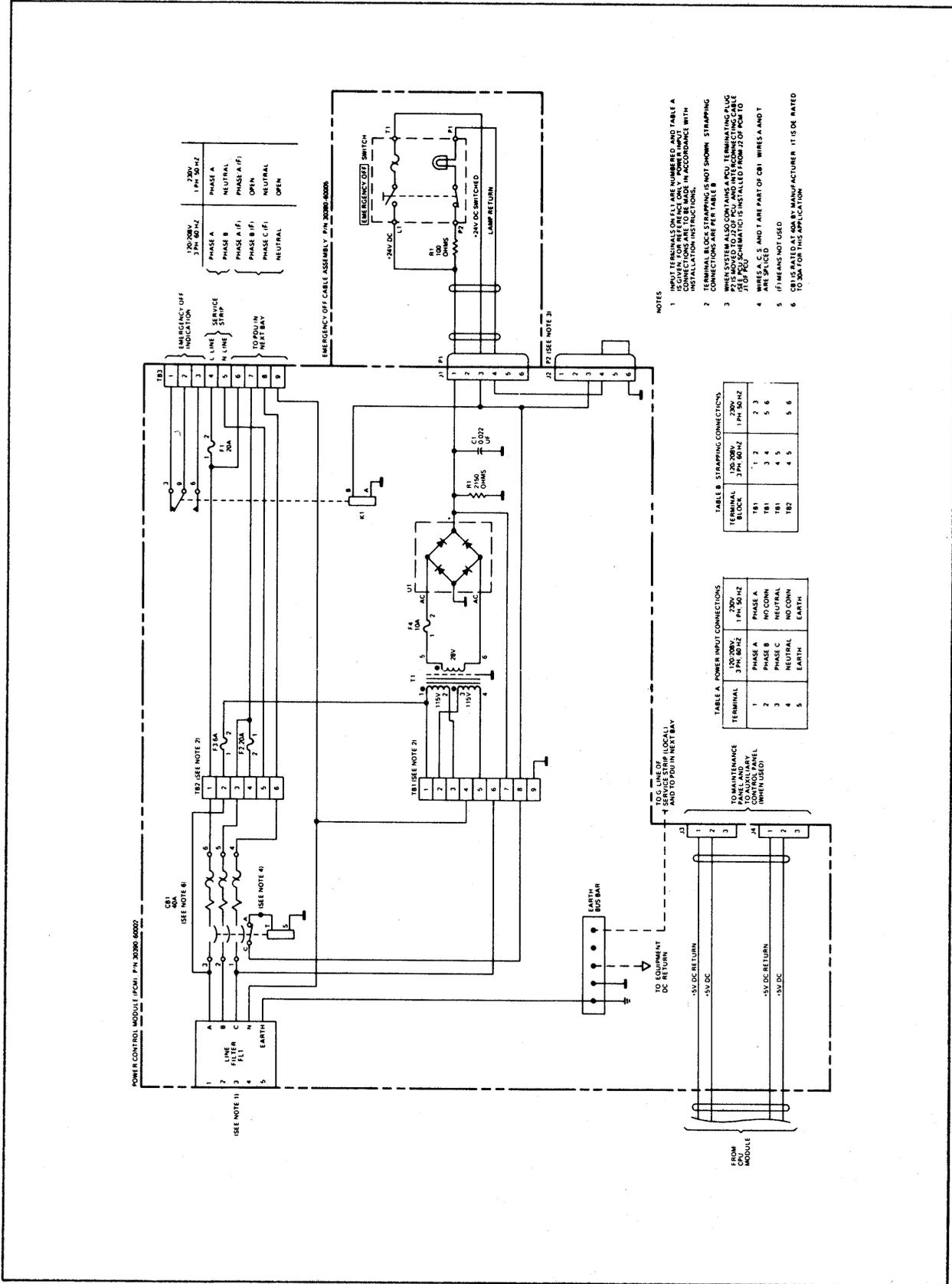
Figure 11-5. PCM Schematic Diagram

11-7. PERIPHERAL DEVICE INSTALLATION

Installation instructions for peripheral devices that are part of a standard computer system are contained in the HP 3000 Computer System Installation Manual, part no. 30000-90081. Installation instructions for non-standard peripheral devices are contained in separate instruction manuals specifically for each device.

11-8. NEW INSTALLATION TURN-ON

To turn-on a newly installed system for the first time, perform the following steps.

a. Set all power switches OFF on each HP 30310A, HP 30311A, and HP 30312A Power Supply in the system. All HP 30310A Power Supplies are at the rear of an equipment bay; all others are behind a front door of an equipment bay.

b. Set the Computer Main Frame Power Panel circuit breaker to ON.

c. At the rear of an equipment bay, set the PCM MAIN SYSTEM POWER and CPU EXTENDED SYSTEM POWER circuit breakers to ON. Then, check for the following conditions:

    (1) All fans in card cages and equipment bays or cabinets are blowing.

    (2) At the upper-right corner of the CPU Bay, the EMERGENCY OFF pushbutton is lighted.

<div align="center">Note</div>

> If the EMERGENCY OFF pushbutton is not lighted or any fan is not blowing, correct the error before going any further in this procedure. Refer to paragraphs 11-2 through 11-4.

d. Behind the front door of the CPU Bay, set Dc Control Panel toggle switches to ON in the following order. (See figure 9-1.)
    (1) LOWER MEMORY DC POWER

    (2) UPPER MEMORY DC POWER

    (3) SYSTEM DC POWER.

e. On the inside of the front door at the top, set two toggle switches; PANEL to ENBL (enable) and PF/ARS (Power Fail/ Automatic Restart) to ENBL (enable).

f. At the rear of equipment bays, set the power switch to ON on each HP 30310A Power Supply.

g. Behind the front door of equipment bays, set the power switch to ON on each HP 30312A Power Supply; then, do so on each HP 30311A Power Supply.

h. Check the BATTERY STATUS lamps for the following indications:

   (1) The battery is charging; the lamp blinks at about a 0.5-Hz rate.

   (2) The battery indicated is not needed, is absent, or an error condition exists; the lamp is not lighted.

   (3) After the system has been ON for a while, the battery should become fully charged; the lamp remains lighted.

i. Close the front door of equipment Bay No.1 and check the panel at the top front. Two BATTERY STATUS lamps indicate the same conditions as those behind the door, and the EMERGENCY OFF pushbutton is lighted.

11-9. SYSTEM VOLTAGE ADJUSTMENTS

Note

The voltage measurements made in the following instructions must be made in reference to the appropriate common circuit. The common circuits specified in the following instructions are labeled COM and the common symbol. For all measurements, use an HP 3439A Digital Voltmeter with an HP 3441A Range Selector (or their equivalents).

Perform the system voltage checks and adjustments as follows:

a. Allow at least 15 minutes warm-up time, preferably with the CPU in the RUN state. (The RUN lamp is lighted.)

b. Check the control panel at the top front of the full length door on the CPU Bay. If the RUN lamp is lighted, press the top of the RUN/HALT switch to turn-off the RUN lamp.

c. Behind the front door of equipment bays, look for the small panel of an HP 30312A Power Supply (figure 9-1). For each HP 30312A Power Supply in the system, perform these steps:

   (1) Set the ADJ R32 potentiometer fully clockwise.

   (2) Press and hold toggle switch S2 and turn ADJ R32 counterclockwise until the LED in the upper-right corner of

the HP 30312A panel lights to indicate "overcurrent".

(3) Release toggle switch S2 and check that the overcurrent LED goes out and a 5V LED lights.

d. Open the rear door of equiment Bay No. 1, remove the screw(s) at the left edge of the HP 30310A Power Supply (figure 9-1) and swing that Power Supply out of the bay.

e. At the top of the HP 30310A Power Supply, turn R12 fully clockwise.

f. Connect the digital voltmeter between the +5VDC connector (BNC type) center conductor and a COM terminal lug on the back plane of the CPU card cage.

Note

Do not connect the digital voltmeter common lead to chassis ground. Failure to comply will produce erroneous and possible destructive results.

g. Use a small tip screwdriver to reach through the top of the HP 30310A Power Supply to adjust A1R1 until the digital voltmeter displays +5.17 volts.

h. Leave the digital voltmeter common lead where it is, but move input lead to the +15, -5, and -15 test jacks on the HP 30310A rear panel. The voltmeter should display a value between 14.2 and 16.7 for either the +15 or -15 test jacks, and between 4.4 and 5.7 for the -5 test jack.

i. Move the voltmeter input lead to the +20 test jack on the HP 30310A panel. Then, use a small tip screwdriver to reach through the top of the HP 30310A Power Supply to set A3R2 fully clockwise and leave at this setting. The voltmeter should display a value between 16.8 and 18.0 volts.

j. Move the voltmeter input lead to the -20 test jack. The same value (but opposite polarity) noted in step i should be displayed.

k. Repeat steps d through j for each HP 30310A Power Supply in the system.

l. Move the voltmeter to the front of the equipment bays.

m. Connect the voltmeter common or return lead to the common symbol jack on the front panel of the HP 30311A Power Supply in the CPU Bay (figure 9-1).

n. Connect the input lead of the voltmeter to each of the test
    jacks on the HP 30311A Power Supply front panel and compare
    the value displayed against table 11-4.

o. If all the values displayed are out of tolerance, proceed no
    further until the +5.00V Internal Reference Adjustment pro-
    cedure described in the HP 3000 Series II/III Computer System
    Service Manual part no. 30000-90018 has been performed on
    your HP 30311A Power Supply.

p. Repeat steps m through o for each HP 30311A Power Supply in
    the system.

Table 11-4. HP 30311A Test Jack Voltages

Test Jack	Value Required
+5B	+5.1; +/- 0.1
+12B	+12.0; +/- 0.1
+12.7B	value of + 12 test jack +0.7(+/- 0.2)
-3B	-3.0; +/- 0.25
-5B	-5.0; +/- 0.2

11-10. SYSTEM VERIFICATION

To verify that the system operates correctly, turn on all periph-
eral devices and connect them on-line. Then, use the QA Verifi-
cation Tape to exercise the system's System Verification Process.
Also run the appropriate diagnostics in accordance with the in-
structions contained in the HP 3000 Series II/III Computer System
Service Manual, part no. 30000-90018. Next, perform the MPE/3000
Cold-Start Procedure in accordance with the instructions con-
tained in the HP 3000 Computer System Operator's Guide, part no.
32000-90013.

PART TWO
HP 32435A SERIES III COMPUTER SYSTEM INSTALLATION

WARNING

Prior to performing any installation procedures, ensure that the three Power Control Module circuit breakers (located at bottom rear of the CPU Bay) are set to their OFF positions and that the main power cable is not connected to the power source. Failure to comply may result in serious injury or death and severe damage to equipment.

11-11. EQUIPMENT BAY INSTALLATION

The system equipment bay(s) are completely assembled and tested at the factory prior to shipment. If the system to be installed is a 2-bay model (Option 200), the two equipment bays must first be mechanically and electrically connected in accordance with the instructions contained in the HP 3000 Computer System Installation Manual, part no. 30000-90147. Once this is accomplished or if the system to be installed is a 1-bay model, simply position the bay(s) as desired, extend and lock the anti-tip base extension legs in place, and lower the equipment bay feet located at the end of the extension legs and at the rear of the bay(s).

11-12. Isolation Transformer Strapping

The computer system is shipped from the factory with its isolation transformer pre-strapped for 208 VAC to 60-Hz sites and for 230 VAC to 50-Hz sites. However, to ensure proper system operation, check and, if necessary, restrap the isolation transformer as follows:

a. Using a voltmeter, measure the actual steady-state voltage at the AC receptacle that has been installed to provide power to the computer system.

b. Refer to table 11-5 and check that the measured steady-state voltage is within the limits specified for the rated voltage the customer contracted for during his computer site preparation. If the voltage is within the limits specified in table 11-5, proceed to step c. If the voltage is not within the limits specified for the contracted rated voltage, the computer site has not been properly prepared. Do not proceed with the system installation until the site has been properly wired in accordance with the instructions contained in the HP 3000 Computer System Site Preparation Manual, part no. 30000-90145.

c. As viewed from the front, remove the CPU equipment bay right-side panel.

Table 11-5. Primary Power Voltage Tolerances

Rated Voltage	Acceptable Steady-State Voltage	
	From	To
200	180.0	208.0
208	187.2	216.32
220	198.0	228.8
230	207.0	239.2
240	216.0	249.6

Note: In general, peripherals not receiving power via the CPU bay Power Control Module require a steady-state line voltage from 103.5V to 119.6V for 60-Hz sites and from 207.0V to 239.2V for 50-Hz sites. However, various power options are available for most HP peripherals. Regardless of the peripheral power option ordered, the available steady-state voltage must be within the limits of +4/-10 percent of the rated voltage.

d. Remove the isolation transformer primary side cover plate shown in figure 11-6.

WARNING

Ensure that all Power Control Module circuit breakers are set their OFF positions and that the main power cable is not connected to the power source. Failure to comply may result in serious injury or death.

e. Using figure 11-6 as a guide, check that the isolation transformer is correctly strapped for the primary power rated voltage available at the site. If necessary, restrap the isolation transformer for the available RATED voltage (not the actual STEADY-STATE voltage measured in step a).

The isolation transformer must always be strapped for the available RATED voltage and not for the actual STEADY-STATE voltage measured in step a. For example, if the site RATED voltage is 208 VAC and the measured STEADY-STATE voltage is 216 VAC, strap the isolation transformer for the RATED voltage of 208 VAC even though the STEADY-STATE voltage of 216 VAC seems to indicate strapping for 220 VAC. The computer site must be wired to provide a RATED primary power source
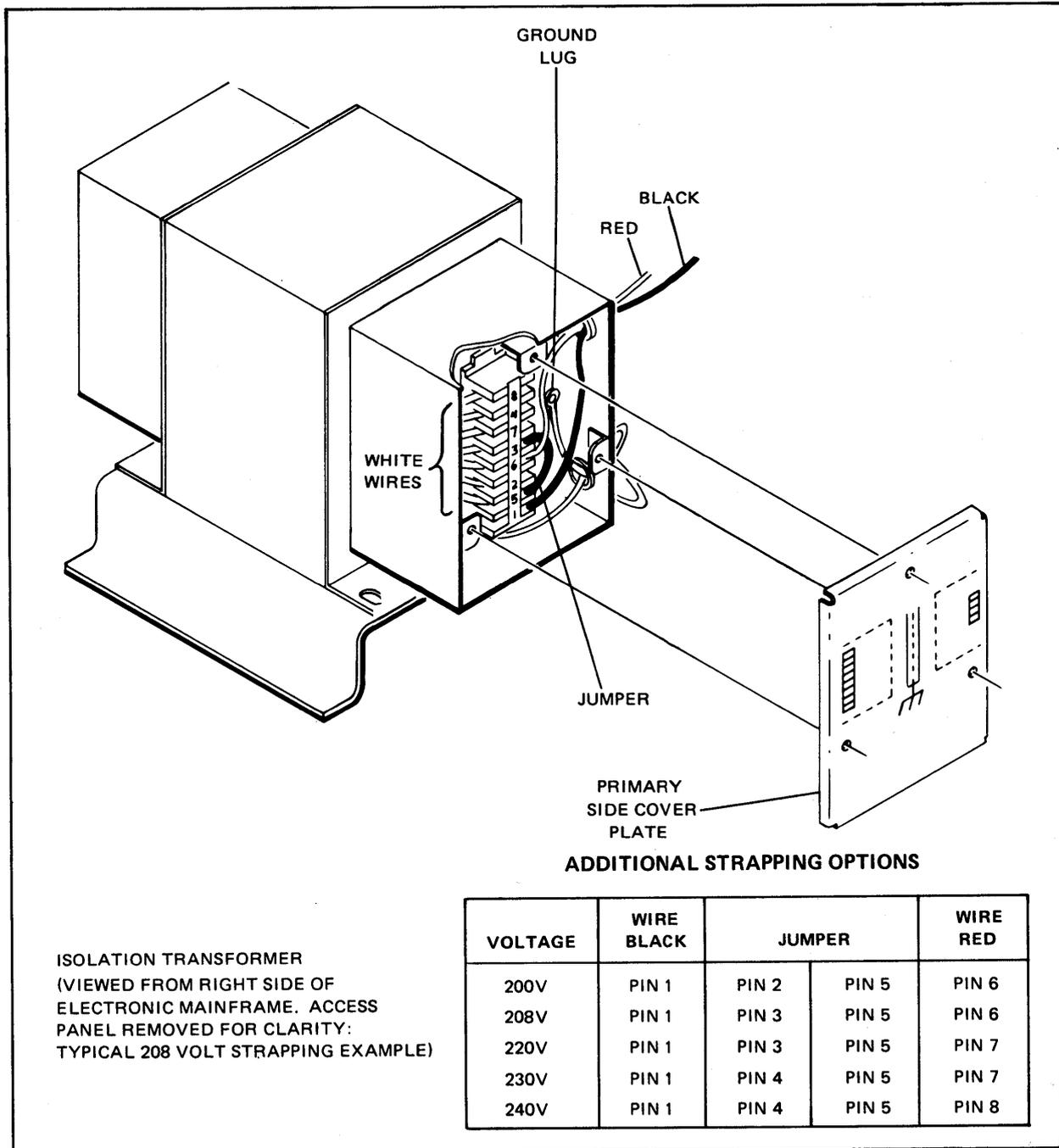
GROUND
LUG

BLACK

RED

WHITE
WIRES

JUMPER

PRIMARY
SIDE COVER
PLATE

ADDITIONAL STRAPPING OPTIONS

ISOLATION TRANSFORMER
(VIEWED FROM RIGHT SIDE OF
ELECTRONIC MAINFRAME. ACCESS
PANEL REMOVED FOR CLARITY:
TYPICAL 208 VOLT STRAPPING EXAMPLE)

VOLTAGE	WIRE BLACK	JUMPER		WIRE RED
200V	PIN 1	PIN 2	PIN 5	PIN 6
208V	PIN 1	PIN 3	PIN 5	PIN 6
220V	PIN 1	PIN 3	PIN 5	PIN 7
230V	PIN 1	PIN 4	PIN 5	PIN 7
240V	PIN 1	PIN 4	PIN 5	PIN 8

Figure 11-6.   Isolation Transformer Strapping Otions

that meets the voltage requirements of table 11-5 and the
isolation transformer must always be strapped for the   RATED
voltage.

f. Ensure that all terminal block connections are tight and that
no loose strands of wire are protruding from the terminal
block.

g. Ensure that infinite resistance (open circuit) exists between all terminal block connections and the ground lug shown in figure 11-6.

h. If necessary, reconfigure the Power Control Module input VAC rating plates located under the main circuit breaker to reflect the current isolation transformer strapping.

i. Replace the isolation transformer primary side cover plate and the equipment bay right-side panel.

11-13. Cable Connections

Step-by-step equipment bay(s) cable connection instructions are contained in the HP 3000 Computer System Installation Manual, part no. 30000-90147.

11-14. PERIPHERAL DEVICE INSTALLATION

Installation instructions for peripheral devices that are part of a standard computer system are contained in the HP 3000 Computer System Installation Manual, part no. 30000-90147. Installation instructions for all other peripheral devices must be obtained from the individual instruction manuals supplied with the devices.

11-15. NEW INSTALLATION TURN-ON

Do not attempt to turn-on a newly installed system until the isolation transformer has been properly strapped, all cable connections completed, all peripheral devices installed, and the I/O system properly configured in accordance with the instructions contained in the HP 3000 Computer System Installation Manual, part no. 30000-90147. After this has been accomplished, perform the following steps to turn-on the system for the first time.

a. Set the Power Control and Display Panel DC POWER LOGIC and MEMORY switches to DISABLE. (The Power Control and Display Panel is located at the bottom front of the CPU equipment bay.)

b. Ensure that the Power Control Module MAIN POWER, INTERNAL POWER, and SWITCHED 120V OUTLETS circuit breakers are still set to their OFF positions and connect the main power cable to the customer-furnished AC power source.

c. Set the MAIN POWER circuit breaker to its ON position and then set the INTERNAL POWER circuit breaker to its ON position. (The SWITCHED 120V OUTLETS circuit breaker can be set to its ON position at this time if desired although it is not necessary until power is required for the two associated AC power receptacles located on the Power Control Module. (The two standard AC receptacles are for the system mag tape unit and disc drive.)

d. Set the Power Control and Display Panel DC POWER LOGIC and MEMORY switches to ENABLE.

e. Check that all Power Control and Display Panel DC VOLTAGES and DC STATUS indicators are lighted. (The BCHG indicator may be slowly flashing to indicate that the battery pack is being charged. When the batteries are fully charged, the BCHG indicator will stop flashing and remain lighted.) If all the indicators are not lighted, refer to Section X to determine the problem.

f. Open the front door of the CPU equipment bay and, at the top, set the PANEL and PF/ARS (Power Fail/Automatic Restart) toggle switches to ENBL.

11-16. SYSTEM VOLTAGE CHECKS

Prior to performing any system verification procedures, allow the system to warm-up for approximately 15 minutes and then check that all the Power Control and Display Panel DC VOLTAGES are within the tolerances specified in table 11-6. Use the Power Control and Display Panel voltage and common ground test points to measure the system voltages. If any measured voltage is not within the limits specified in table 11-6, adjust the appropriate power supply in accordance with the instructions contained in Section X.

Table 11-6. System DC Voltage Tolerances

Test Point DC Voltage	Voltage Tolerance	
	Min	Max
+15	+14.5	+16.5
+12B	+11.9	+12.1
+5 CPU	+5.17	+5.17
+5B	+5.0	+5.25
+5 V1	+5.0	+5.25
+5 V2	+5.0	+5.25
+5 V3	+5.0	+5.25
-5B	-5.0	-5.25
-5	-4.5	-5.5
-15	-14.5	-15.5

11-17. SYSTEM VERIFICATION

To verify that the system operates correctly, proceed as follows:

a. Run the system microdiagnostics in accordance with the instructions contained in the HP 3000 Computer System Installation Manual, part no. 30000-90147.

b. Cold load SLEUTH from the magnetic tape unit and run the I/O Configuration Test using the SLEUTH CONF (configure) command in accordance with the instructions contained in Stand-Alone SLEUTH Diagnostic Manual, part no. 03000-90123.

c. Run all applicable stand-alone diagnostics in accordance with the instructions contained in the individual diagnostic manuals supplied with the system.

NOTES