RCA
Information
Systems

SPECTRA 70

70/46

Processor
Reference Manual

70-46-601
March 1968

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

First Printing: April 1967
Reissued: March 1968

CONTENTS

Page

INTRODUCTION

SYSTEM STRUCTURE

INSTRUCTION FORMATS

ADDRESSING

PROGRAM INTERRUPT

INPUT/OUTPUT OPERATION

CONTENTS (Cont'd)

Page

INPUT/OUTPUT OPERATION (Cont'd)

MULTI-PROCESSOR INSTALLATION

PRIVILEGED INSTRUCTIONS

CONTENTS (Cont'd)

Page

PRIVILEGED INSTRUCTIONS (Cont'd)

PROCESSOR STATE CONTROL INSTRUCTIONS

FIXED-POINT INSTRUCTIONS

CONTENTS (Cont'd)

Page

FIXED-POINT INSTRUCTIONS (Cont'd)

DECIMAL ARITHMETIC INSTRUCTIONS

LOGICAL INSTRUCTIONS

CONTENTS (Cont'd)

CONTENTS (Cont'd)

Page

LIST OF TABLES

LIST OF ILLUSTRATIONS

INSTRUCTION INDEX

The index marks at the right edge of this page line up with similar index marks in the text. By merely examining the page edges, the reader can quickly locate a category of instructions.

Appendix A summarizes the instruction set for the 70/46 Processor, including timing, formats and condition codes.

INTRODUCTION

RCA MODEL 70/46 PROCESSOR

◆ The 70/46 Processor incorporates features which increase the efficiency of the system for time-sharing use and for conventional batch processing. This is accomplished by using main and subsidiary memory to create a virtual memory of two million bytes. The virtual memory consists of blocks of either 4,096 or 2,048 bytes which are called pages. An address translation feature translates the addresses of the virtual memory pages into actual addresses as assigned in working memory by the operating system. The translated actual addresses are then stored in a translation memory which is used to implement the virtual memory.

The 70/46 Processor is a halfword-organized, variable-format processor consisting of main memory, nonaddressable main memory, scratch-pad memory, translation memory, read-only memory, program control and arithmetic unit, input/output control, and a program interval timer. The 70/46 provides multiprogramming with multiaccess time-sharing capabilities.

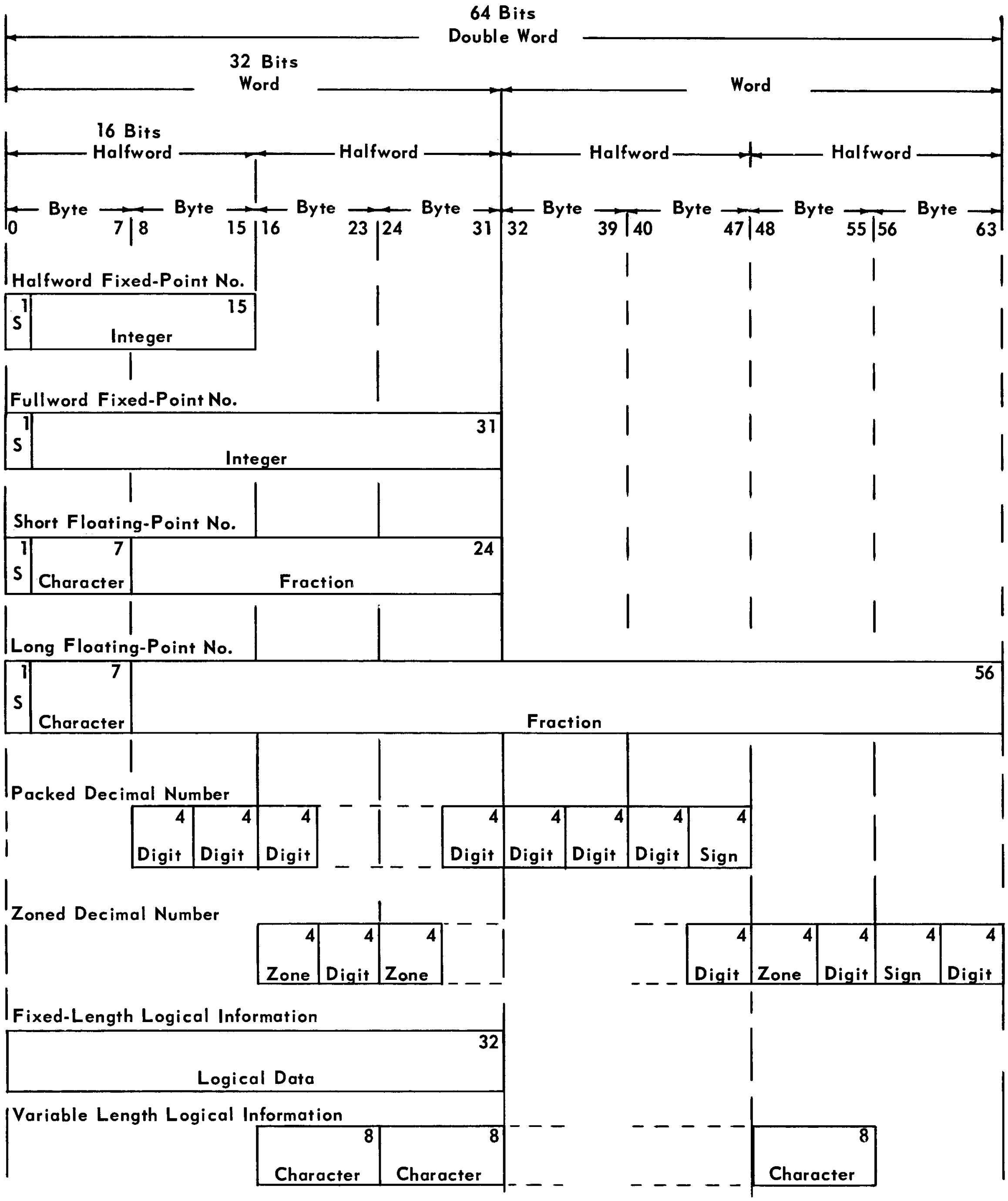
User programs may run interactively at remote terminals or sequentially under the automatic control of a job stream monitor where the presence of the user is not required. The 70/46 also features an efficient technique for the handling of I/O data transfer through the reduction in processing interference during I/O selector channel operations, and an increase in the I/O transfer rate capability.

The Time Sharing Operating System, which is used with the 70/46 Processor, consists of a set of control routines, language processors, and service routines which enable the complete system to provide efficient batch processing concurrently with time-sharing operations from remote terminals.

Compatibility

◆ All instructions, character codes, interrupt facilities, formats, and programming features are functionally the same as corresponding features on the 70/35, 70/45, and 70/55 Processors. Programs can be interchanged between processors provided that:

1. Systems features are equivalent (Emulator features are *not* provided).
2. Programs are written to be independent of strict timing considerations.
3. Programs are restricted to specified functions and do not use unspecified characteristics peculiar to the hardware of either processor.
4. Program interrupts does not occur where an instruction is terminated with unpredictable results.
5. Programs are written subject to all specified compatibility restrictions.



NOTE: Numbers in upper right corners of blocks indicate number of bits used.

Figure 1. Data Formats

ORGANIZATION OF DATA

◆ The following definitions describe the various levels of data organization for the 70/46 Processor:

Bit

◆ A bit is a single binary digit having the value of either zero or one.

Byte

◆ A byte consists of eight information bits. It represents two decimal digits, one alphabetic character, or one special symbol.

Halfword

◆ A halfword consists of two consecutive bytes beginning on a main memory location that is a multiple of two.

Word

◆ A word consists of four consecutive bytes beginning on a main memory location that is a multiple of four.

Doubleword

◆ A doubleword consists of eight consecutive bytes beginning on a main memory location that is a multiple of eight.

Item/Field

◆ An item/field consists of any number of bytes that specify a particular unit of information (numeric field, alphabetic name, street address, stock number, etc.).

Record

◆ A record consists of one or more related items.

DATA FORMATS

◆ The basic unit of information in the 70/46 Processor is a byte, which is the smallest addressable unit. A byte consists of eight information bits. The parity bit ensures the accuracy of all bytes accessed by the processor. Odd parity is used in the 70/46 Processor.

The internal code representation in the 70/46 is either the Extended Binary-Coded-Decimal Interchange Code (EBCDIC) or the USA Standard Code for Information Interchange (USASCII) as specified by program. (See Appendices D and E.)

There are eight distinct formats for data in main memory (see figure 1). Further explanation of each format appears in the instruction sections of this manual.

NUMBERING SYSTEM

◆ Since binary addresses are cumbersome to work with, the hexadecimal numbering system has been adopted to represent characters and addresses in the 70/46 Processor. The hexadecimal system has a base of 16. The first ten marks are represented by decimal numbers zero (0) through nine (9); the last six marks are represented by the letters A through F.

The basic hexadecimal marking system and its binary and decimal equivalent are specified in table 1. (See Appendix H.)

Table 1. Basic Hexadecimal Marking System

Hexadecimal (Base 16)	Binary (Base 2)	Decimal (Base 10)	Hexadecimal (Base 16)	Binary (Base 2)	Decimal (Base 10)
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	A	1010	10
3	0011	3	B	1011	11
4	0100	4	C	1100	12
5	0101	5	D	1101	13
6	0110	6	E	1110	14
7	0111	7	F	1111	15

SYSTEM STRUCTURE

MAIN MEMORY

◆ The main memory of the RCA 70/46 Processor is the central storage for both data to be processed and the controlling instructions. Main memory consists of planes of magnetic cores, with each core representing one binary digit. The smallest addressable unit of information in main memory is one byte (eight bits). The first 128 locations of main memory are reserved for processor use and must not be used by the program.

The basic cycle time of the 70/46 Processor is the time required to access and transfer a halfword from main memory to the memory register and regenerate the information in main memory. The memory cycle time is 1.44 microseconds, and memory is available in a 262 KB module.

NON-ADDRESSABLE MAIN MEMORY

◆ A non-addressable main memory, is in addition to main memory and cannot be addressed by programming. It contains the subchannel registers that control the operation of input/output devices on the multiplexor channel. A set of three 32-bit registers services each device on the multiplexor channel; 256 subchannel register sets and devices can be connected to the multiplexor channel.

SCRATCH-PAD MEMORY

◆ The scratch-pad memory is a micromagnetic storage device consisting of 128 four-byte words, the cycle time of which is 300 nanoseconds. Each word is scratch-pad memory is uniquely addressed.

The following registers are contained in scratch-pad memory. (See also Appendix I.):

1. *Processor Utility Registers* — All locations designated as processor utility registers are used by the processor for program control and cannot be used by the program.
2. *General Registers* — These locations are the general registers for each processor state. These registers are used by the program for base addressing, for indexing, or for storing operands.

   Note: The 70/46 Processor has four processor states that pertain to system and program interrupts.
3. *Interrupt Mask Registers* — An Interrupt Mask register for each processor state permits or inhibits 32 interrupt conditions.
4. *Interrupt Status Registers* — An Interrupt Status register for each processor state stores interrupt identification information and operational control information. This register contains indications of the last state interrupted, the protection key, the decimal mode (USASCII or EBCDIC), the privileged mode bit, and the supervisor call identification.

SCRATCH-PAD MEMORY (Cont'd)

5. *Program Counter* — A Program Counter for each processor state contains the main memory address of the next instruction to be executed, the condition code, the instruction length code, and the program mask.

6. *Input/Output Channel Registers* — A set of four registers for each selector channel controls input/output operation. A set of four registers for the multiplexor channel controls initiation and termination of input/output operations on the multiplexor channel.

7. *Floating-Point Registers* — Four floating-point registers (each is two words long) are used in floating-point arithmetic.

8. *Interrupt Flag Register* — One Interrupt Flag register is provided. When an interrupt condition occurs, a bit associated with this condition is set in the Interrupt Flag register.

TRANSLATION MEMORY

◆ The Translation Memory is a magnetic storage device consisting of 512 halfwords (1,024 bytes), the cycle time of which is 300 nanoseconds. Each halfword (two bytes) in the translation memory is uniquely addressed and contains a translation table element which is used in translating virtual addresses to actual addresses (see figure 2).

The translation table which is maintained in the translation memory is loaded and stored from and to main memory by special EO (Elementary Operation) routines. It is addressed during each main memory addressing cycle when translation is required. Address translation does not require additional instruction time from that required by the basic 70/45 timing; however, staticizing time for the SS-Format Load Multiple and Execute instructions is increased when operating in 70/46 Mode.

Each element of the table consists of 17 bits (16 data bits plus 1 parity bit).

	P	W	G	U	S	E	M	XXX	REAL PAGE	H
Bits		**0**	**1**	**2**	**3**	**4**	**5**	**6 8**	**9 14**	**15**

P = *Parity bit.*

W = *Written Into Bit:* indicates, when set, that the page addressed in memory by this translation word has been written into. This bit indicates, when reset, that the page has not been written into. This bit is set and reset by the processor.

G = *Accessed Bit:* indicates, when set, that the page addressed in memory by this translation word has been accessed (read, or written into). This bit indicates, when reset, that the page has not been accessed. This bit is set and reset by the processor. Attempted but unsuccessful access to a page does not set this bit.

U = *Utilization Bit:* indicates, when set, that the addressed translation word can be utilized. This bit indicates, when reset, that the addressed translation word cannot be utilized and a Paging Queue Program Interrupt condition occurs. This bit is set and reset by the program.

TRANSLATION MEMORY (Cont'd)

S = *State Bit:* indicates, when set, that the addressed page is nonprivileged. When this bit is reset, it indicates that the addressed page is privileged. When this bit is reset and the nonprivileged bit in the ISR is set, a Paging Error Program Interrupt condition occurs. This bit is set and reset by the program.

E = *Executable Bit:* indicates, when set, that the page addressed in memory by this translation word can be read as an operand or instruction, but cannot be written into. If a program attempts to write into a page with this bit set in the translation word, a Paging Error Program Interrupt condition occurs. This bit indicates, when reset, that the page addressed in memory can be executed, read or written into. This bit is set and reset by the program.

M = *Page Control Bit:* indicates, when set, that a 2,048-byte page is referenced. This bit indicates, when reset, that a 4,096-byte page is referenced. If the high-order bit of the displacement field is set and M is set, a Paging Error Program Interrupt condition occurs. This bit is set and reset by the program.

H = *Page Address Bit:* indicates, when set, and M is set, the high-order address (2,048 bytes of a 4,096 byte page). This bit indicates, when reset and M is set, the low-order address (2,048 bytes of a 4,096 byte page). This bit is ignored if M is reset. This bit is set and reset by the program.

XXX bits are for future expansion and must be zeros (program restriction).

Notes

◆ 1. The G condition is provided as a program flag to indicate *written into* and/or *accessed,* respectively. A first time Read or Write to a page would cause the G bit to be set.

2. This translation memory is provided in addition to the 128-word memory used in scratch pad.

3. Addresses used in I/O servicing and I/O data transfer are direct and do not go through translation.

READ-ONLY MEMORY

◆ Three banks of Read-Only Memory (ROM) are standard on the Model 70/46 Processor. Each ROM bank consists of 2,048 56-bit words (each containing one micro-instruction of 53-bit [plus 3 parity bits] length). In addition each ROM contains a 12-bit address register and a 54-bit memory register.

The wired-in microprogram logic contained in the first read-only memory bank controls the elementary operations when in the 70/45 or 70/46 Mode. The effective cycle time of the ROM banks is 480 nanoseconds with a 56-bit access.

Although the Read-Only Memory is a standard feature in the 70/46, it is not accessible by programming and the programmer need not be familiar with the detailed method of operation of the ROM.

PROGRAM CONTROL AND ARITHMETIC UNIT

◆ The program control and arithmetic unit in the Model 70/46 Processor interprets and executes the instructions stored in main memory. Registers and indicators monitor the sequence of operations, perform automatic accuracy checks, and communicate with the RCA standard interface in the control of input/output devices.

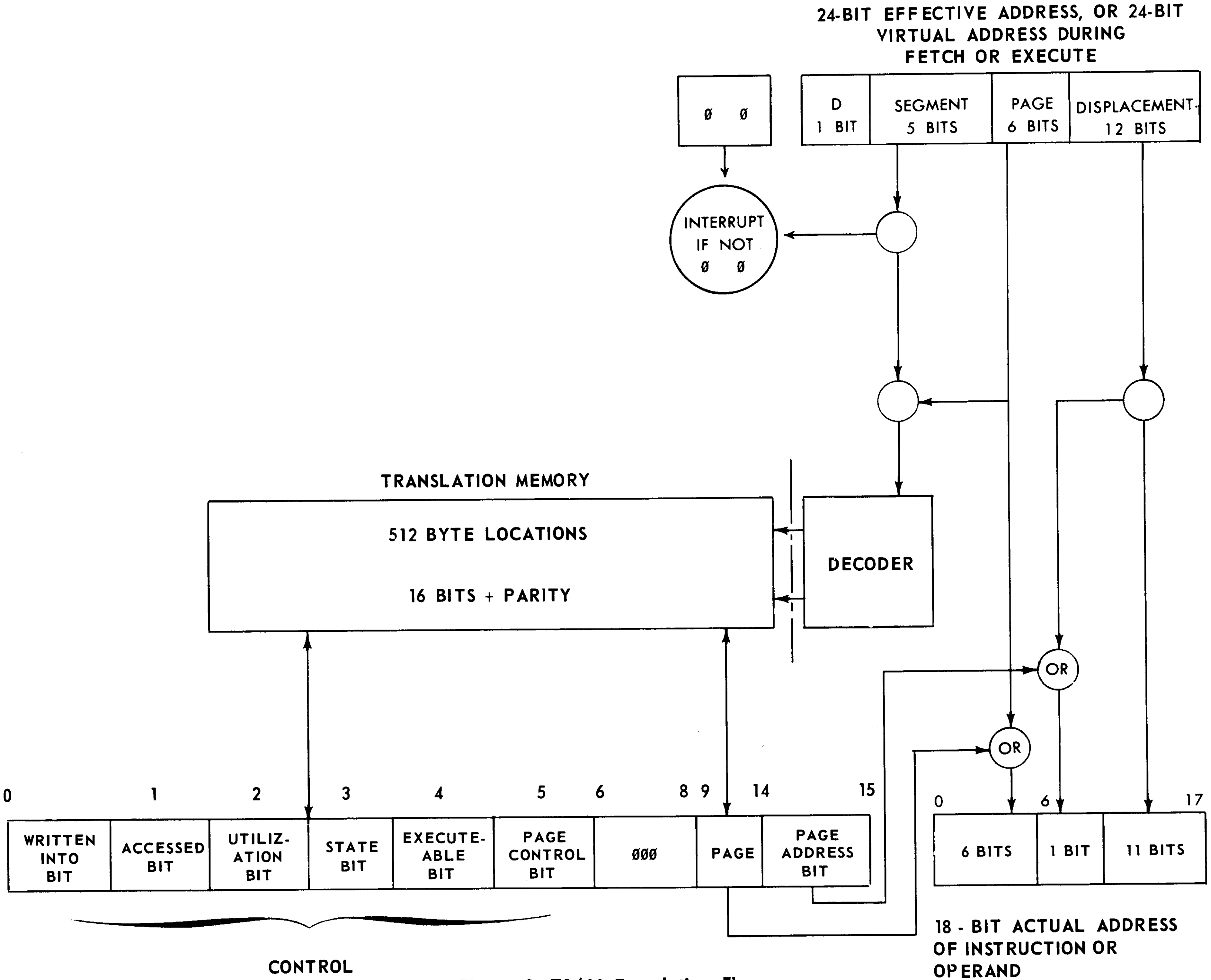


Figure 2. 70/46 Translation Flow

INPUT/OUTPUT CONTROL

◆ The RCA 70/46 Processor communicates with all input/output devices through the RCA standard interface.

The 70/46 Processor can have up to four selector channels (optional). Each selector channel contains two standard interface trunks. Each standard interface trunk controls one device subsystem (from 1 to 16 devices). All selector channels can operate simultaneously.

In addition to the selector channels, a multiplexor channel is standard equipment on the 70/46 Processor.

The multiplexor channel on the 70/46 contains eight standard interface trunks. Each trunk controls one device subsystem. All trunks on the multiplexor channel can operate simultaneously. Also, the multiplexor channel and all selector channels can operate simultaneously.

INTERVAL TIMER

◆ The 70/46 has a variable 16-bit Interval Timer which can be set and read by the program. Upon being set to a nonzero value, the least significant bit position of the timer is decremented by one every 100 microseconds until its count becomes zero. Further decrementing is suppressed and the Interval Timer (Flag Position 12) interrupt is effected, subject to the corresponding mask. The Interval Timer runs when set to a nonzero value; otherwise it does not run. The decrement of the count occurs such that the total elapsed time is never less than the count set in the Interval Timer. The maximum possible time interval is not greater than 100 microseconds more than the loaded count. This timer is not available to 70/35, 70/45, and 70/55 programs.

The Interval Timer is an independent unit capable of being read and loaded by Special Functions. Decrementation occurs simultaneously with processing and causes no interference to either processing (except for program interrupt upon lapse of count) or I/O servicing. When the processor is halted, the Interval Timer decrementing is stopped. General reset causes the Interval Timer to be reset to zero.

Note: Use of the Interval Timer and Diagnostic Snapshot by programs may not occur together because the Counter register is common to both. If the Diagnose function is initiated while the Interval Timer is running, the shared counter is cleared to zero without occurrence of the Interval Timer interrupt and the Diagnose function assumes control of the counter. If the function being diagnosed is the Load Interval Timer, the actual loading of the counter is inhibited but the E/O Flow is diagnosed.

INSTRUCTION FORMATS

◆ The five basic instruction formats express, in general terms, the operation to be performed as follows:

RR = register-to-register
RX = register-to-indexed main memory
RS = register-to-main memory
SI = main memory and immediate operand operation
SS = main memory to main memory

The instruction subfields are defined as follows:

R_1, R_2, R_3 — four-bit general register designation used for an operand
X_2 — four-bit general register designation used for indexing
B_1, B_2 — four-bit general register designation used for base addressing
D_1, D_2 — 12-bit displacement
I_2 — eight-bit immediate operand
L_1, L_2 — four-bit operand length specification
L — eight-bit operand length specification
M — eight-bit mask

Before executing the Load Multiple, Store Multiple, and the SS format instructions, an address look-up is performed to insure that all pages referenced can be utilized. The time required for this address look-up is in addition to regular staticizing time. If T = 1 (70/46 Mode), the additional time is required. If T = 0 (70/45 Mode), no additional time is required.

RR FORMAT

◆ The contents of the general register specified by R_1 is the first operand. The contents of the general register specified by R_2 is the second operand. In floating-point operations, R_1 designates the address of the floating-point register that contains the first operand. R_2 designates the floating-point register that contains the second operand. The first and second operands can be the same and are designated by identical R_1 and R_2 addresses.

Op Code	R_1	R_2
0 7	8 11	12 15

RX FORMAT

◆ The contents of the general register specified by R_1 is the first operand. To obtain the address of the second operand, the contents of the general registers specified by X_2 and B_2 are added to the D_2 field. In floating-point operations, R_1 designates the floating-point register that contains the first operand.

Op Code	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

RS FORMAT

◆ The RS format is used by shift instructions, branching instructions, and load/store multiple instructions.

Op Code	R_1	R_3	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Shift Instructions

◆ The contents of the general register specified by R_1 is the first operand. The contents of the general register specified by B_2 are added to the D_2 field. The sum specifies the number of bits of shifting to be done by the shift operation. The R_3 field is ignored.

Branching Instructions

◆ The contents of the general register specified by R_1 is the first operand. The contents of the general register specified by B_2 are added to the D_2 field to obtain the branch address. The contents of the general register specified by R_3 is the third operand.

Load/Store Multiple Instructions

◆ The R_1 and R_3 fields specify the general register boundaries. The contents of the general register specified by B_2 are added to the D_2 field to obtain the main memory address of the second operand.

SI FORMAT

◆ The contents of the general register specified by B_1 are added to the contents of the D_1 field to obtain the address of the first operand. The second operand is the immediate eight-bit byte in the I_2 field of instruction.

Op Code	I_2	B_1	D_1
0 7	8 15	16 19	20 31

SS FORMAT

◆ The contents of the general register specified by B_1 are added to the contents of the D_1 field to obtain the address of the leftmost byte of the first operand. The L_1 field specifies the number of additional bytes in the operand that are to the right of the first operand address. To obtain the second operand address, the contents of the general register specified by B_2 are added to the contents of the D_2 field. The L_2 field specifies the number of additional bytes in the operand that are to the right of the second operand address. The L field specifies the number of additional bytes that are to the right of the first and the second operand address.

Op Code	L		B_1	D_1	B_2	D_2
	L_1	L_2				
0 7	8 11	12 15	16 19	20 31	32 35	36 47

Notes

◆ 1. A zero appearing in the X_2, B_1 or B_2 fields indicates an absence of the corresponding address or shift-amount component. An instruction can specify the same general register both for address modification and for operand location.

2. Address modification is completed before the execution of an operation.

3. The results replace the first operand (except in Store Character instruction, where the result replaces the second operand).

4. A variable-length result is never stored outside the field specified by the address and length.

5. The contents of all registers and main memory locations not specified by an instruction remain unchanged except for the Edit and Mark instruction and the Translate and Test instructions. These instructions automatically use certain general registers as given in table 2.

SS FORMAT (Cont'd)

Table 2. Use of General Registers

Processor State*	Edit and Mark	Translate and Test
P_1	GR 1	GR 1 and 2
P_2	GR 1	GR 1 and 2
P_3	GR 13	GR 13 and 14
P_4	GR 9	GR 9 and 10

* Processor States are discussed on page 16.

ADDRESSING

◆ Locations in main memory are consecutively numbered starting with zero. In forming an address, the base address (B_1 B_2) and the index (X_2) are treated as unsigned 24-bit positive binary numbers. The displacement (D_1 D_2) is treated as a 12-bit positive binary number. The three are added together as absolute binary numbers and overflow is ignored. The results of these additions yields an 18-bit effective address.

Any address that is within the effective address, but specifies memory not available in the particular installation, causes an interrupt to occur. Any address that is outside the effective address as shown above is ignored. However, to maintain program compatibility on all processors, all addressing should assume a 24-bit effective address. Negative indexing may be achieved by address wrap-around since overflow bits over the 24-bit address are ignored.

Paging and Segmentation

◆ Paging and segmentation are used to allocate more memory space to the computer program than is actually available in the processor. The 70/46 System uses special equipment features and programming to provide this virtual memory capability.

The 70/46 main memory is divided into blocks of equal size called pages. A 70/46 program can consist of many of these pages but, during any one execution stage, only those pages required for that execution stage are in main memory. The pages that are not required are maintained in subsidiary storage. The 70/46 programming relocates program pages dynamically within main memory so that programs are executable in different main memory locations. The 70/46 basic page size is 4,096 bytes. At the discretion of the program, a 2,048-byte page size can also be used. This shorter page length makes it possible to pack main memory more tightly as well as reducing the transfer time between subsidiary storage and main memory for short routines that do no require a full 4,096 byte page of storage space. Use of the 2,048 byte page, however, reduces the available virtual memory space by half since the addressing scheme provides for only 512 pages regardless of whether they are 4,096 bytes or 2,048 bytes.

The 70/46 provides a grouping of the virtual memory pages into segments. Segments are independent, logical entities composed of 64 pages. In the 70/46 only eight of the 32 potential virtual segments are implemented, each segment consisting of 64 pages. If all pages of all eight segments are 4,096 byte pages, a total virtual memory of two million bytes is available. If all pages of all segments are 2,048 byte pages, a total virtual memory of one million bytes is available. Because address incrementation wraps around on 262,144 bytes (equivalent of a segment), no equipment means are provided to sequence a program from one segment to another.

MEMORY ADDRESS TRANSLATION

◆ The following two modes are defined for control of memory address translation:

1. *70/46 Mode:* causes all non-I/O memory addresses (instruction sequence and operands) to be translated if the D-bit within each address is zero.

MEMORY ADDRESS TRANSLATION (Cont'd)

2. *70/45 Mode (non-translate):* Memory is addressed directly using the addresses generated during staticizing. The 70/45 mode is used by all programs other than 70/46 programs.

In the 70/46 mode each memory address, except I/O instruction execution and servicing, may be translated. The addresses (called virtual addresses) of instructions, taken from the next instruction address field of the P-counter, and data operands may be translated via a table look-up to obtain actual memory addresses. Virtual addresses consist of 24 bits using the following format:

1 Bit	5 Bits	6 Bits	12 Bits
D	Segment	Page	Displacement

The 24-bit virtual address is the address in the P-Counter (NIA Field), or, the operand address after all required address arithmetic has been performed.

The page and displacement compose the 18-bit address field and are generated by the 18-bit address arithmetic. The segment is an additional subfield carried in the 24-bit NIA field within the P-counter or supplied by 5 bits within the 24-bit low-order address portion of a base register.

Note: These bits positions are ignored in index registers (RX Format). The D bit is used to specify direct addresses (untranslated) if the Privileged Mode is established (N = 0).

In the 70/46 Mode, with the Page Control Bit set (2,048 byte pages), the 11 low-order bits of the displacement field are used, untranslated, in actual memory addresses. The high-order bit must be zero. When the Page Control Bit is reset (4,096 byte pages), the 12-bits of the displacement field are used, untranslated, in actual memory addresses. The 11-bits of the page and segment are used to address 8 virtual segments, each with 64 virtual pages. The six page-bits and three low-order segment bits are combined to yield a nine-bit Translation Table address. The two unused segment bits are reserved for future expansion and must be zeros in order to avoid a PD error interrupt.

Notes

◆ 1. The page size of the 70/46 is 2,048 bytes or 4,096 bytes, depending on whether the Page Control Bit (M) in the translation word is set (1) or reset (0), respectively. Pages are independent and need not occupy contiguous physical memory space. The low-order halves of 4,096-byte virtual pages are occupied by 2,048-byte virtual pages; if they do not, (i.e., M = 1 and the high-order Displacement Field bit is set), a Paging Error Program Interrupt Condition occurs.

2. Only base registers and P-counters supply segments; these bits are ignored within index registers in forming effective operand addresses. If no base register is specified (that is B = 0), then the D bit is interpreted as 0 and segment 0 is addressed.

3. The contents of a translation table element are accessed and linked together with the displacement to compose the actual memory address. A control field is provided with each table element.

Notes
(Cont'd)

4. The SS MOVE instructions in the 70/45 (to maintain compatibility with the System/360) are implemented such that if the source and destination fields are adjacent (overlap each other), the first byte of the source field will be extended into the destination field, resulting in a symbol fill. Similarly, in SS logical instructions with adjacent fields, each byte operation uses the result of the preceding logical operation as an operand. Since the implementation of these instructions in the 70/46 are more complex, the purpose of this note (with its supporting tables) is to define this implementation in further detail.

   The step-by-step operations for overlapped and non-overlapped fields in the move instructions are detailed in table 2A. Similar results are obtained for the SS Logicals except that instead of extending a source character when fields overlap, the result of each operation is extended.

   The conditions which determine whether a MOVE in the 70/46 will result in an actual move or in a fill are detailed on the chart in table 2B. The same rules apply to SS Logical instructions in as much as an actual move is equivalent to a valid logical result and a fill is equivalent to extending the preceding result as an operand. The chart may generally be summarized in narrative form as follows:

   a. Two addresses are virtually non-adjacent and they do not translate into the same page: A move field results.

   b. Two addresses are virtually adjacent (page and displacement) and they are:

      (1) in the same segment: a symbol fill results.

      (2) in different segments and actual addresses are adjacent: a symbol fill results.

      (3) in different segments and actual addresses are not adjacent: a field move results.

   c. If one address is direct (untranslated) and the other address is a virtual address, the result is a field move. An exception occurs when the virtual and direct addresses are adjacent and the translated virtual address is adjacent to the direct address, in which case the result is a symbol fill.

Notes
(Cont'd)

Table 2A. Analysis of 70/46 Move Instruction Results

	1st Address				2nd Address				Translation Result	Instruction Result
	D	S	P	Dis	D	S	P	Dis		
1.	0	A	B	C+1	0	A	B	C		Fill
2.	0	A	B	C+1	0	A	D	C	(B) ≠ (D) C+1 ≠ 0	Move
3.	0	A	B+1	000	0	A	B	FFF	(B) ≠ (B+1)	Fill
4.	0	A	B+1	000	0	A	B	FFF	(B) = (B+1)	Fill
5.	0	A	B	C+1	0	A	D	C	(B) = (D)	Move
6.	1		B	C+1	0		B	C	(B) = B	Fill
7.	1		B	C+1	0		D	C	(D) ≠ B	Move
8.	1		B	C+1	0		B	C	(B) ≠ B	Move
9.	1		B	C+1	0		D	C	(D) = B	Move
10.	0	A	B	C+1	0	E	B	C	(AB) ≠ (EB)	Move
11.	0	A	B	C+1	0	E	B	C	(AB) = (EB)	Fill
12.	0	A	B	C+1	0	E	D	C	(AB) = (ED)	Move
13.	0	A	B	C+1	0	E	D	C	(AB) ≠ (ED)	Move

Legend:

D = Direct Address Bit
S = Segment
P = Page

Dis = Displacement
(X) = X Translated

Table 2B. Analysis of Overlapped and Non-Overlapped Fields of 70/46 Move Instruction

Overlapped Fields

(Move four byte field starting at Memory Location 1 into destination field starting at Memory Location 2.)

	Memory Locations				
	1	2	3	4	5
Before	A	B	C	D	E
1st Operation	A	A	C	D	E
2nd Operation	A	A	A	D	E
3rd Operation	A	A	A	A	E
4th Operation	A	A	A	A	A
After	A	A	A	A	A

Non-Overlapped Fields

(Move four byte field starting at Memory Location 1 into destination field starting at Memory Location 5.)

	Memory Locations							
	1	2	3	4	5	6	7	8
Before	A	B	C	D	Z	Y	X	W
1st Operation	A	B	C	D	A	Y	X	W
2nd Operation	A	B	C	D	A	B	X	W
3rd Operation	A	B	C	D	A	B	C	W
4th Operation	A	B	C	D	A	B	C	D
After	A	B	C	D	A	B	C	D

PROGRAM INTERRUPT

INTRODUCTION

◆ Program interrupts occur as a result of errors in data or instruction specifications, input/output operations, external signals, equipment malfunctions or arithmetic errors. The instruction being executed at the time of the interrupt can be completed, suppressed, or terminated depending on the cause of the interrupt.

An interrupt can be inhibited or permitted in any state through programming. If an interrupt occurs and is permitted, conditions existing in the interrupted state are automatically stored. Control is then passed to the Interrupt Control State P_3 or Machine Condition State P_4, depending on the cause of the interrupt. (See Processor States below.) The priority of the interrupt is established and an analysis is made to determine the proper linkage to the Interrupt Response State P_2 so that the interrupt may be processed. After interrupt processing is completed, control is returned to the state which was last interrupted, and normal processing is resumed.

If several interrupts occur at the same time, the one having the highest priority is processed. The remaining interrupts are processed in turn, depending on their priority.

PROCESSOR STATES

◆ The RCA 70/46 Processor has four processor states that provide control of system and program interrupts. Programs can be executed in any one of the states, because each state is completely independent and has its own set of registers. The processor states and their functions are as follows:

Processing State P_1

◆ The Processing State P_1 interprets and executes the user's program. This processing state is the problem-oriented state.

Interrupt Response State P_2

◆ The Interrupt Response State P_2 performs specific program tasks as dictated by the Interrupt Control State P_3.

Interrupt Control State P_3

◆ The Interrupt Control State P_3 is automatically entered when an interrupt that is other than one caused by a machine check or power failure is recognized. In this state, programming is responsible for performing a detailed analysis of the cause of the interrupt and establishing its priority. After these functions are performed, linkage is provided to the related interrupt processing routine in the Interrupt Response State P_2.

Machine Condition State P_4

◆ The Machine Condition State P_4 is entered whenever a machine check, scratch pad memory parity error (if applicable), or power failure occurs. In this state, programming analyzes the cause of a machine interrupt and establishes its priority. Control is then transferred to the Interrupt Response State P_2, so that an indication of the cause of interrupt can be given to the operator.

PROCESSOR STATE REGISTERS

◆ Registers are provided in scratch-pad memory, for each processor state as given in table 3.

Table 3. Processor State Registers

Register	State			
	P_1	P_2	P_3	P_4
Program Counter	1	1	1	1
General Registers	16	16	6	5
Floating-Point Registers	4	*	*	*
Interrupt Status Register	1	1	1	1
Interrupt Mask Register	1	1	1	1

* Floating-point instructions executed in any of the processor states use the floating-point registers assigned to P_1.

Because each processor state has its own general registers, Interrupt Status Register and Interrupt Mask Register, storing and reloading these registers is not necessary during interrupt processing.

Program Counter

◆ The Program Counter (P-counter) is a 32-bit register that is located in scratch-pad memory. A separate P-counter is provided for each of the four processor states.

The format of the P-counter is as follows:

ILC	CC	Program Mask	Next Instruction Address
0 1	2 3	4 7	8 31

Bit Positions 0 and 1 contain the instruction length code. When an interrupt occurs and is taken, or a Program Control instruction is executed, the length of the last instruction executed in the terminated state, before the interrupt condition occurred, is stored in bit positions 0 and 1 as given in table 4. The instruction length code is always generated from the operation code of the instruction.

Table 4. Instruction Length Codes

ILC	Length in Bytes
01	Two-byte instruction.
10	Four-byte instruction.
11	Six-byte instruction.

Notes:

1. If the interrupt condition is an operation code trap, the length of the instruction causing the interrupt is generated from the operation code and is stored in bit positions 0 and 1 as given in table 6.

2. The instruction length code is unpredictable if the interrupt was caused by one of the following:
   Power Failure
   Machine Check
   Address Error (only if the address error was caused
   by an invalid instruction address)

Program Counter *(Cont'd)*

Bit Positions 2 and 3 contain the condition code. When an interrupt occurs or a Program Control instruction is executed, the condition code is moved from a machine register, where it is maintained for instruction execution, and stored in this field of the P counter of the state being terminated. The condition code in this field of the P counter of the state being initiated is moved into a machine register where it is maintained for possible future use.

Bit Positions 4 through 7 contain the program mask. When an interrupt occurs or a Program Control instruction is executed, the program mask is moved from the machine register, where it is maintained for instruction execution, and stored in bits 4 through 7 of the P counter of the state being terminated. The program mask in this field of the P counter of the state being initiated is moved into the machine register where it is maintained for possible future use.

Bit Positions 8 through 31 contain the next instruction address. This field stores the address of the next instruction in main memory to be staticized by the appropriate processor state. Each time an instruction is staticized, the P counter is updated to the next instruction. This field is left intact whenever an interrupt requires switching to a new processor state.

General Registers

◆ A separate set of general registers is assigned to each processor state. Each general register is 32 bits long. Sixteen general registers are assigned to P_1 and P_2, six general registers are assigned to P_3, and five general registers are assigned to P_4. These registers serve as operands, base address registers, or index registers.

Floating-Point Registers

◆ Four floating-point registers are provided. Each floating-point register is 64 bits long (double length). These registers are used only in floating-point arithmetic. The floating-point registers can be used by any of the processor states.

Interrupt Status Registers

◆ The Interrupt Status register is a 32-bit register. A separate register is provided for each of the four processor states.

The format of each Interrupt Status register is as follows:

ISI	000	PI	KEY	A	T	B	N	00000000	Call (R_1 R_2)
0 2	3 5	6 7	8 11	12	13	14	15	16 23	24 31

Bit Positions 0 through 2 contain the interrupt state identifier. When an interrupt occurs, the number of the processor state being interrupted is stored in this field of the processor state being initiated as given in table 5.

Table 5. Interrupt State Identifier Codes

ISI	Definition
000	P_4 was interrupted.
001	P_3 was interrupted.
010	P_2 was interrupted.
011	P_1 was interrupted.

Interrupt Status Registers *(Cont'd)*

Bit Positions 3 through 5 are not used and must be zeros.

Bit Positions 6 and 7 contain the program indicators. When an interrupt occurs due to a parity error in Main Memory or Scratch Pad Memory, the program indicators are stored in this field in P_4 as given in table 6.

Table 6. Program Indicator Codes

Program Indicators	Definition
00	Neither error has occurred.
01	Scratch Pad Memory parity error has occurred.
10	Main Memory parity error has occurred.
11	Scratch Pad Memory parity error and Main Memory parity error have occurred.

Bit Positions 8 through 11 contain the memory protection key. This field is set by the program to indicate the desired protection key. When an interrupt occurs or a Program Control instruction is executed, the memory protection key is extracted from this field of the processor state being initiated and placed in a machine register where it performs the memory protect function. The four-bit key provides a possible 15 keys ranging from $(1)_{16}$ to $(F)_{16}$. Each 2,048-byte block of main memory has its individual machine register for the protection key. When the key related to the current processor state and the key related to the main memory block are equal, or either is zero, the main memory block accepts a data store. Conversely, if the keys do not match, and neither is zero, an address error (protection) interrupt occurs.

Notes:

1. If the memory protect feature is not installed, this field must be zero.
2. Keys are effective on actual (after translation) addresses.

Bit Position 12 designates the internal decimal code. When an interrupt occurs or a Program Control instruction is executed, the decimal code (either USASCII or EBCDIC) for the processor state being initiated is established by the setting of this bit. If the bit is 1, USASCII Code is established; if the bit is 0, EBCDIC is established.

Note: The setting of this Decimal Code does not affect any automatic translation of data read into or written from the processor. The Decimal Code is used to determine what zone configuration (USASCII or EBCDIC) is to be established internally when executing the decimal arithmetic instruction set, the Edit instruction, and the Edit and Mark instruction.

Bit Position 13 is defined as the 70/45-46 Mode Control bit (T-bit) for the 70/46 processor. It specifies whether translation is allowed.

T = 1: *70/46 Mode:* Direct Addressing or Translate Addressing is specified by the setting of the D-bit.

T = 0: *70/45 Mode:* Direct Addressing only, the setting of the D-bit is ignored.

Notes:

1. General reset resets the T-bit to zero.

Interrupt Status Registers *(Cont'd)*

2. When T = 1, the D-bit within the virtual addresses (supplied by the NIA field of the P counter or the effective operand addresses after staticizing — except I/O instructions) specify either address translation or direct addressing.

   D = 1: Direct addressing.

   D = 0: Translate addressing.

Bit Position 14, the B-bit, is controlled by the hardware via the Function Call instruction to control which ROM bank is used. It has no meaning to the software.

Bit Position 15 is the non-privileged mode bit. This field is set by the program to indicate the privileged status of the processor state being initiated. If N = 0, the initiated processor state runs in the privileged mode, allowing execution of the privileged instructions; if N = 1, the processor state runs in the non-privileged mode, inhibiting the execution of the privileged instructions.

Bit Positions 16 through 23 are not used and must be zeros.

Bit Positions 24 through 31 is the call field. This field is set during the execution of a Supervisor Call instruction. The R_1 and R_2 field of this instruction provide a code which is placed into the call field of the Interrupt Status register of the processor state in which the Supervisor Call instruction is issued. This code provides linkage to the program required to accomplish the purpose of the Supervisor Call instruction.

Interrupt Mask Registers

◆ The Interrupt Mask register is a 32-bit register. A separate register is provided for each of the four processor states. Each bit in the Interrupt Mask register is associated with an interrupt condition. A 0 bit in any bit position in this register inhibits the associated interrupt condition; a 1 bit in any bit position in this register permits the associated interrupt condition.

Important:

1. The Power Failure and Machine Check interrupts must be inhibited in the Machine Condition State P_4. The mask bits in the Interrupt Mask register for these interrupt conditions must always be zero. This is a program restriction.
2. All interrupts except the Machine Check interrupt must be inhibited in the Interrupt Control State P_3. The mask bit in the Interrupt Mask register for this interrupt condition must always be zero. This is a programming restriction.

Program Mask Registers

◆ In addition to the Interrupt Mask register, a Program Mask register is also provided for each state. The Program Mask register is not contained in main memory or scratch-pad memory. It is a separate machine register which is set by the non-privileged instruction, Set Program Mask, and it applies to the following interrupt conditions:

Significance error.
Exponent underflow.
Decimal overflow.
Fixed-point overflow.

Program Mask Registers *(Cont'd)*

The program mask bit settings have priority over the bit settings in the Interrupt Mask register for the above four program interrupts. A 0 bit in any bit position in this register cancels the interrupt condition if it occurs. A 1 bit in any bit position in this register indicates that the Interrupt Mask register is to be examined. If an interrupt condition occurs and is inhibited by the Interrupt Mask register, it remains pending until it is serviced (permitted).

Register Addressing

◆ Register addressing in each of the processor states is given in table 7.

Table 7. Register Addressing in the Processor States

Register Number	Processor States			
	P_1	P_2	P_3	P_4
0	GR	GR	IMR, P_1 State	Processor Utility
1	GR	GR	ISR, P_1 State	Processor Utility
2	GR	GR	P counter, P_1 State	Processor Utility
3	GR	GR	Interrupt Flag Register	Processor Utility
4	GR	GR	IMR, P_2 State	Processor Utility
5	GR	GR	ISR, P_2 State	Processor Utility
6	GR	GR	P counter, P_2 State	Processor Utility
7	GR	GR	GR	Processor Utility
8	GR	GR	IMR, P_3 State	GR
9	GR	GR	ISR, P_3 State	GR
10	GR	GR	P counter, P_3 State	GR
11	GR	GR	GR	GR
12	GR	GR	GR	IMR, P_4 State
13	GR	GR	GR	ISR, P_4 State
14	GR	GR	GR	P counter, P_4 State
15	GR	GR	GR/Weight	GR/Weight

GR = General Register
IMR = Interrupt Mask Register
ISR = Interrupt Status Register

Notes:

1. The P counter, Interrupt Status register, and Interrupt Mask register for processor state P_1, P_2 and P_3 can be addressed by register notation (R_1, R_2 or R_3 field of an instruction) in processor state P_3 only. The P counter, ISR and IMR for processor state P_4 can be addressed by register notaticn in processor state P_4 only. Because the P counter, the ISR's and the IMR's are contained in scratchpad memory, they can be addressed in any of the processor states by using the Load Scratch Pad instruction and the Store Scratch Pad instruction. However, these instructions are privileged instructions and the processor state in which they are executed must be running in the privileged mode. (Bit position 15 of the appropriate Interrupt Status register must be set to zero.)
2. Floating-Point registers may be addressed by floating point instructions only, and are addressed as 0, 2, 4 and 6 in all processor states.

Interrupt Flag Register

◆ The Interrupt Flag register is a 32-bit register. There is only one Interrupt Flag register. When an interrupt condition occurs, a bit associated with the specific interrupt is set in the Interrupt Flag register. If the corresponding bit in the Interrupt Mask register for the current state is set, an interrupt occurs.

Interrupt Flag Register *(Cont'd)*

Note: If the interrupt condition is one of the four program interrupts, the corresponding bit in the Program Mask register must also be set to cause an interrupt.

The Interrupt Flag register is scanned on a priority basis and the highest priority interrupts are serviced first. Each interrupt condition is assigned a specific weight which is put into the rightmost eight bits of General register No. 15 of the initiated state (P_3 or P_4). This weight can be used by the program to enter the proper interrupt routine.

Note: The rightmost two bytes of General register No. 15 in P_3 or P_4 are cleared and reloaded each time an interrupt occurs.

Table 8 lists the priority, the Interrupt Flag register position, the program state initiated, and the weight of each of the interrupt conditions.

Table 8. Interrupt Conditions and Priority

Priority	Interrupt Condition	Flag *Bit	State Initiated	Weight
1	Power Failure	2^{0}	P_4	0
2	Machine Check	2^{1}	P_4	4
3	External Signal No. 1	2^{2}	P_3	8
4	External Signal No. 2	2^{3}	P_3	12
5	External Signal No. 3	2^{4}	P_3	16
6	External Signal No. 4	2^{5}	P_3	20
7	External Signal No. 5	2^{6}	P_3	24
8	External Signal No. 6	2^{7}	P_3	28
9	Interval Timer	2^{8}	P_3	32
10	Selecter Channel No. 1	2^{9}	P_3	36
11	Selector Channel No. 2	2^{10}	P_3	40
12	Selector Channel No. 3	2^{11}	P_3	44
13	Selector Channel No. 4	2^{12}	P_3	48
	Not used			
	Not used			
16	Multiplexor Channel	2^{15}	P_3	60
17	Elapsed Time Clock	2^{16}	P_3	64
18	Console Interrupt Request	2^{17}	P_3	68
19	Paging Error	2^{18}	P_3	72
20	Paging Queue	2^{19}	P_3	76
21	Supervisor Call Instruction	2^{20}	P_3	80
22	Privileged Operation	2^{21}	P_3	84
23	Op-Code Trap	2^{22}	P_3	88
24	Address Error (Protect, Addressing, Specification)	2^{23}	P_3	92
25	Data Error	2^{24}	P_3	96
26	Exponent Overflow	2^{25}	P_3	100
27	Divide Error	2^{26}	P_3	104
28	Significant Error**	2^{27}	P_3	108
29	Exponent Underflow**	2^{28}	P_3	112
30	Decimal Overflow**	2^{29}	P_3	116
31	Fixed Point Overflow**	2^{30}	P_3	120
32	Test Mode	2^{31}	P_3	124

* 2^{0} = The rightmost bit in the Interrupt Flag register.

** *Note:* These interrupt conditions can be masked by two separate masks. The first, the program mask, is a four-bit, non-privileged, program settable mask, that

Interrupt Flag Register *(Cont'd)*

can be used to cancel the interrupt condition when it occurs. The second mask is composed of bits 2^{30} through 2^{27} of the 32-bit Interrupt Mask register associated with the state in which the processor is operating. If the Program Mask prohibits the interrupt it is cancelled. If the Program Mask permits the interrupt, the Interrupt Mask register is scanned. Like all the other interrupt conditions, the masks of the 32-bit Interrupt Mask register leave these four interrupt conditions pending if the associated mask bits are zeros.

INTERRUPT CONDITIONS

◆ A description of the individual interrupt conditions is given in table 9. More detailed information concerning the interrupt conditions is given in the instruction descriptions. Some interrupt conditions arise from input/output channel operations, and these conditions are further discussed in the Input/Output Operational Control section.

Notes:

1. When an interrupt condition occurs, the current instruction can be suppressed or it can be terminated. When an instruction is suppressed, the condition code setting that existed before the instruction was attempted remains unchanged. Data in main memory and the general registers specified by the instruction also remain unchanged. When an instruction is terminated, the condition code setting and data in the general registers and/or main memory are unpredictable.
2. When operating with T = 0, program interrupt functions described in the 70/35, 70/45, 70/55 Processor Reference Manual apply. When operating with T = 1, program interrupt functions of table 9 apply.

INTERRUPT MECHANIZATION

◆ There are two ways of causing a change of processor state. They are:

1. *Automatic Interrupt:* effected when any interrupt condition described in table 9 occurs, and is permitted.
2. *Program Controlled Interrupt:* effected when a Program Control instruction is executed.

Whenever the processor state is changed, either by automatic interrupt or by the execution of a Program Control instruction, some machine conditions must be stored in the P counter and the Interrupt Status register of the terminated state for possible use when the state is initiated again. In addition, certain machine conditions associated with the state being initiated must be extracted from the P counter and the Interrupt Status register of the new state.

All the storing and extracting required when processor status are changed is accomplished by hardware.

Automatic Interrupt

◆ When an automatic interrupt condition occurs, the following events occur: (See figure 3.)

Block 1

◆ A check is made to see if the interrupt condition is one of the following four:

Significance Error

Exponent Underflow

Decimal Overflow

Fixed-Point Overflow

Table 9. Interrupt Conditions

Priority No.	Condition	Flag Bit	Explanation
1	Power Failure	2^0	A power failure interrupt occurs when there is a power failure in the processor or main memory caused by a line failure or by pressing the MASTER pushbutton indicator on the 70/97 Console. Any instruction being executed at the time of interrupt is terminated. It is a program restriction that the mask bit in processor state P_4 for this interrupt condition must always be zero when this interrupt occurs. This permits the program to operate in processor state P_4 for the purpose of closing down the machine during a one-millisecond interval between power failure and actual power loss to the system.
2	Machine Check	2^1	The machine check interrupt occurs when a machine fault or malfunction is detected. Any instruction being executed at the time of interrupt is terminated. It is a program restriction that the mask bit in processor state P_4 for this interrupt condition must always be zero when this interrupt occurs. The following conditions can cause a machine check interrupt to occur: *Scratch-Pad Memory Parity Error*—This error can occur when data is read from the Scratch-Pad Memory. *Main Memory or Non-Addressable Main Memory Parity Error*—If a main memory parity error occurs during an I/O data transfer, this interrupt condition does not occur. A channel interrupt occurs and the program is notified of the condition via the channel status byte.
3 4 5 6 7 8	External Signal No. 1 External Signal No. 2 External Signal No. 3 External Signal No. 4 External Signal No. 5 External Signal No. 6	2^2 2^3 2^4 2^5 2^6 2^7	The external signal interrupt occurs when a signal is received on an external line (1-6) associated with the Direct Control option. Any instruction being executed at the time of interrupt goes to completion.
9	Interval Timer	2^8	Lapse of Interval Timer. This interrupt occurs at completion of an instruction.
10 11 12 13 16	Selector Channel No. 1 Selector Channel No. 2 Selector Channel No. 3 Selector Channel No. 4 Multiplexor Channel	2^9 2^{10} 2^{11} 2^{12} 2^{15}	This interrupt provides the means by which the processor can receive and act upon signals from input/output devices connected to a Selector Channel (1-6) or the Multiplexor Channel. This interrupt can occur as a result of the termination (normal or abnormal) of an input/output operation or at the request of an input/output device. It can also occur as the result of a program controlled interrupt. Any instruction being executed at the time of interrupt goes to completion. (Selector Channels are optional.)
17	Elapsed Time Clock	2^{16}	This interrupt occurs when the Elapsed Time Clock counts downward from positive to negative, indicating that its maximum range has been reached. Any instruction being executed at the time of interrupt goes to completion. (The Elapsed Time Clock is an option.)

Table 9. Interrupt Conditions (Cont'd)

Priority No.	Condition	Flag Bit	Explanation
18	Console Interrupt Request	2^{17}	This interrupt is controlled by the Console Interrupt key on the operator's console. Any instruction being executed at the time of interrupt goes to completion.
19	Paging Error	2^{18}	The conditions under which this interrupt occurs are when memory is addressed in the translation mode (T = 1) and any of the following occurs: 1. When the Non-Privileged Mode is set (N = 1), address translation is occurring (D = 0) and the State Control Bit S in the Translation Table is reset (S = 0). 2. When a non-existent translation table element is addressed (i.e., the two unused bits of the segment field of a virtual address are not zero) and address translation is occurring (D = 0). 3. When a 2,048-byte page error occurs; i.e., direct address bit is reset (D = 0), the page control bit in the Translation word is set (M = 1) and the high-order bit of the Displacement field of the address is set (1). 4. When the Non-Privileged mode is set (N = 1) and the direct address bit in the virtual address is set (D = 1). 5. An operation to write into a location within a page which cannot be written to (i.e., Control Bit E = 1 and D = 0). *Note:* These interrupts cause termination of the instruction with unpredictable results.
20	Paging Queue	2^{19}	Translation Table Location contains an element with control bit U = 0 (i.e., page cannot be used) and memory is addressed via address translation (D = 0 and T = 1). This interrupt causes the instruction to be suppressed. *Note:* This interrupt may occur on instruction staticizing or operand fetching. The interrupt occurs such that the instruction is suppressed. If the interrupt occurs on the first halfword of the instruction being staticized, the NIA field of the object P counter is altered to address the second halfword, and the ILC field is set to 01) by the equipment. If the interrupt occurs on subsequent halfwords, the NIA field is adjusted to the next halfword and the ILC incremented by one for each halfword automatically by the equipment. This enables the object instruction to be re-staticized and executed after the applicable pages have been called into memory. A Special Function is provided to facilitate backing-up of the P counter with the use of the ILC and identification of these pages which may not be utilized.
21	Supervisor Call	2^{20}	This interrupt results from the execution of the Supervisor Call instruction. The P counter and the Interrupt Status register of the interrupted state are updated normally. The rightmost eight bits of the Interrupt Status register of the state in which the instruction is executed receives the R_1, R_2 field of the Supervisor Call instruction.

Table 9. Interrupt Conditions (Cont'd)

Priority No.	Condition	Flag Bit	Explanation
22	Privileged Operation	2^{21}	This interrupt occurs when a privileged instruction is attempted and the current processor state is in non-privileged mode. (Bit position 15 of the Interrupt Status register is set.) The instruction is suppressed. The privileged instructions in the 70/46 Processor are: Diagnose Start Device Test Device Halt Device Check Channel Program Control Load Scratch Pad Store Scratch Pad Idle Set Storage Key } Insert Storage Key } If the Memory Protect option is installed. Write Direct } Read Direct } If the Direct Control option is installed.
23	Operation Code Trap	2^{22}	This interrupt occurs when an operation code that is either not assigned or not available on the particular processor is attempted. No operation is performed. The instruction length code field of the P counter of the terminated state tells how far to rollback the P-count to reach its value prior to interrupt as follows: ILC — Length in Bytes 01 — Two-bytes back 10 — Four-bytes back 11 — Six-bytes back *Note:* The ILC is always generated from the operation code of the instruction.
24	Address Error (Cont'd)	2^{23}	Three conditions cause an address error interrupt to occur. They are: address error, specification error, and protection error. *Addressing* — An address error (addressing) interrupt occurs when: 1. An address specifies any part of data, an instruction or control word outside the available main memory for the particular installation. The instruction operation is terminated for an invalid data address, and the results of the instruction are unpredictable. The instruction operation is suppressed for an invalid instruction address. 2. An Execute instruction specifies another Execute instruction to be performed. The operation is suppressed. 3. The first operand address field of an instruction designates an odd register address for a pair of general registers that contain a double word operand. The operation is suppressed.

Table 9. Interrupt Conditions (Cont'd)

Priority No.	Condition	Flag Bit	Explanation
24	Address Error (Cont'd)	2^{23}	4. A floating-point instruction addresses a floating-point register other than 0, 2, 4, 6. The operation is suppressed. *Specification* — An address error (specification) interrupt occurs when: 1. A data, instruction, or control word address does not specify a doubleword, word, halfword, or byte boundary as required by the particular instruction concerned. The operation is suppressed. 2. The multiplier or divisor in decimal arithmetic exceeds 15 digits and sign. The operation is suppressed. 3. The first operand field is not longer than the second operand field in decimal division or multiplication. The operation is suppressed. 4. Bit positions 28 through 31 of the second operand of a Set Storage Key or Insert Storage Key instruction are not zero. The operation is suppressed. 5. The Memory Protect option is not installed and the protection key in the Interrupt Status register is not zero. The operation is suppressed. 6. The Program Control instruction specifies an instruction address which is not on a halfword boundary. The operation is suppressed. *Protection* — An address error (protection) interrupt occurs when the storage key and the protection key of the result main memory location do not match, and neither is zero. The operation is suppressed if the first main memory location specified that the instruction is in a protected area. The operation is terminated with unpredictable results if the instruction is in progress when the protected area is addressed. (This interrupt can only occur if the Memory Protect option is installed.) *Notes:* 1. If an address error type interrupt occurs during an input/output operation (after initiation), an address error interrupt does not occur. Instead, a channel interrupt occurs for the appropriate channel. 2. It is a program restriction that the mask bit in processor state P_3 for this interrupt condition must always be zero when this interrupt occurs.
25	Data Error	2^{24}	This interrupt occurs when any of the following conditions occur: 1. The sign or digit codes of operands in decimal arithmetic, editing, or Convert To Binary instructions are incorrect. 2. Fields overlap incorrectly in decimal arithmetic. 3. A decimal multiplicand has too many high-order, significant digits. The operation is terminated (suppressed if the operation is a Convert To Binary instruction) upon detection of any of the above.

Table 9. Interrupt Conditions (Cont'd)

Priority No.	Condition	Flag Bit	Explanation
26	Exponent Overflow	2^{25}	The exponent overflow interrupt occurs when the result exponent of floating-point addition, subtraction, multiplication, or division is greater than 127. The operation is terminated.
27	Divide Error	2^{26}	The divide error interrupt occurs when any of the following occur: 1. A quotient exceeds the general register size in fixed-point division, including division by zero. The division is suppressed. 2. The result of a Convert To Binary instruction exceeds one word. The conversion is completed by ignoring information which is outside the general register size. 3. A quotient exceeds the specified data field size in decimal divide. The division is suppressed. 4. Floating-point division is attempted with a divisor whose mantissa is zero. The operation is suppressed.
28	Significance Error	2^{27}	This interrupt occurs when the result mantissa of a floating-point add or subtract instruction is zero. If the interrupt is permitted (by the program mask and the interrupt mask) the operation is completed, the exponent is unaltered, and the interrupt is taken. If the interrupt is inhibited by the program mask, the interrupt condition is cancelled and the operation is completed by setting the result to true zero (zero sign, zero exponent and zero mantissa). If the interrupt is permitted by the program mask but inhibited by the interrupt mask, the interrupt remains pending and the operation is completed by setting the result to true zero (zero sign, zero exponent and zero mantissa).
29	Exponent Underflow	2^{28}	This interrupt occurs when the result exponent of a floating-point addition, subtraction, multiplication, or division is less than zero. The operation is completed by making the result true zero (zero sign, zero exponent, and zero mantissa). If the interrupt is inhibited by the program mask, the interrupt condition is cancelled. If the interrupt is permitted by the program mask, but inhibited by the interrupt mask, the interrupt remains pending.
30	Decimal Overflow	2^{29}	This interrupt occurs when the result field is too small to contain the result of a decimal operation. The operation is completed by ignoring the overflow data. If the interrupt is inhibited by the program mask, the interrupt condition is cancelled. If the interrupt is permitted by the program mask, but inhibited by the interrupt mask, the interrupt remains pending.

Table 9. Interrupt Conditions (Cont'd)

Priority No.	Condition	Flag Bit	Explanation
31	Fixed-Point Overflow	2^{30}	This interrupt occurs when a high-order carry occurs or high-order significant bits are lost in fixed-point addition, subtraction, shifting, or sign control operations. The operation is completed by ignoring the overflow data. If the interrupt is inhibited by the program mask, the interrupt condition is cancelled. If the interrupt is permitted by the program mask, but inhibited by the interrupt mask, the interrupt remains pending.
32	Test Mode	2^{31}	This interrupt provides program control over the processor during program testing. The program test interrupt flag is set by the Program Control instruction. When the interrupt flag bit and the related interrupt mask bit in the state to be initiated are both set, an interrupt occurs after the first instruction that is executed in the initiated processor state.

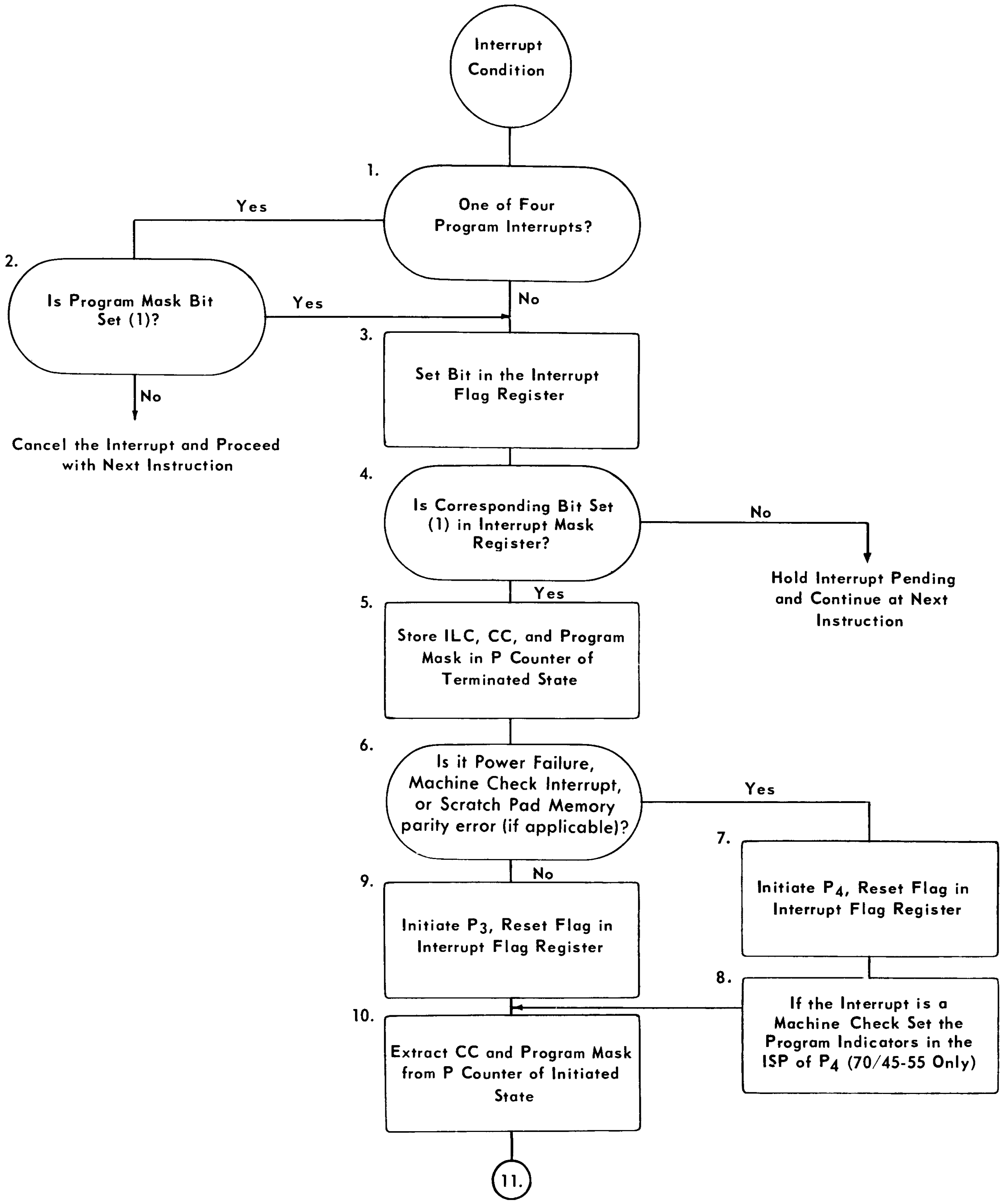

Interrupt Condition
1.
One of Four Program Interrupts?
Yes
2.
Is Program Mask Bit Set (1)?
Yes
No
Cancel the Interrupt and Proceed with Next Instruction
No
3.
Set Bit in the Interrupt Flag Register
4.
Is Corresponding Bit Set (1) in Interrupt Mask Register?
No
Hold Interrupt Pending and Continue at Next Instruction
Yes
5.
Store ILC, CC, and Program Mask in P Counter of Terminated State
6.
Is it Power Failure, Machine Check Interrupt, or Scratch Pad Memory parity error (if applicable)?
Yes
7.
Initiate P4, Reset Flag in Interrupt Flag Register
No
9.
Initiate P3, Reset Flag in Interrupt Flag Register
8.
If the Interrupt is a Machine Check Set the Program Indicators in the ISP of P4 (70/45-55 Only)
10.
Extract CC and Program Mask from P Counter of Initiated State
11.

Figure 3. Functional Logic of Automatic Interrupt

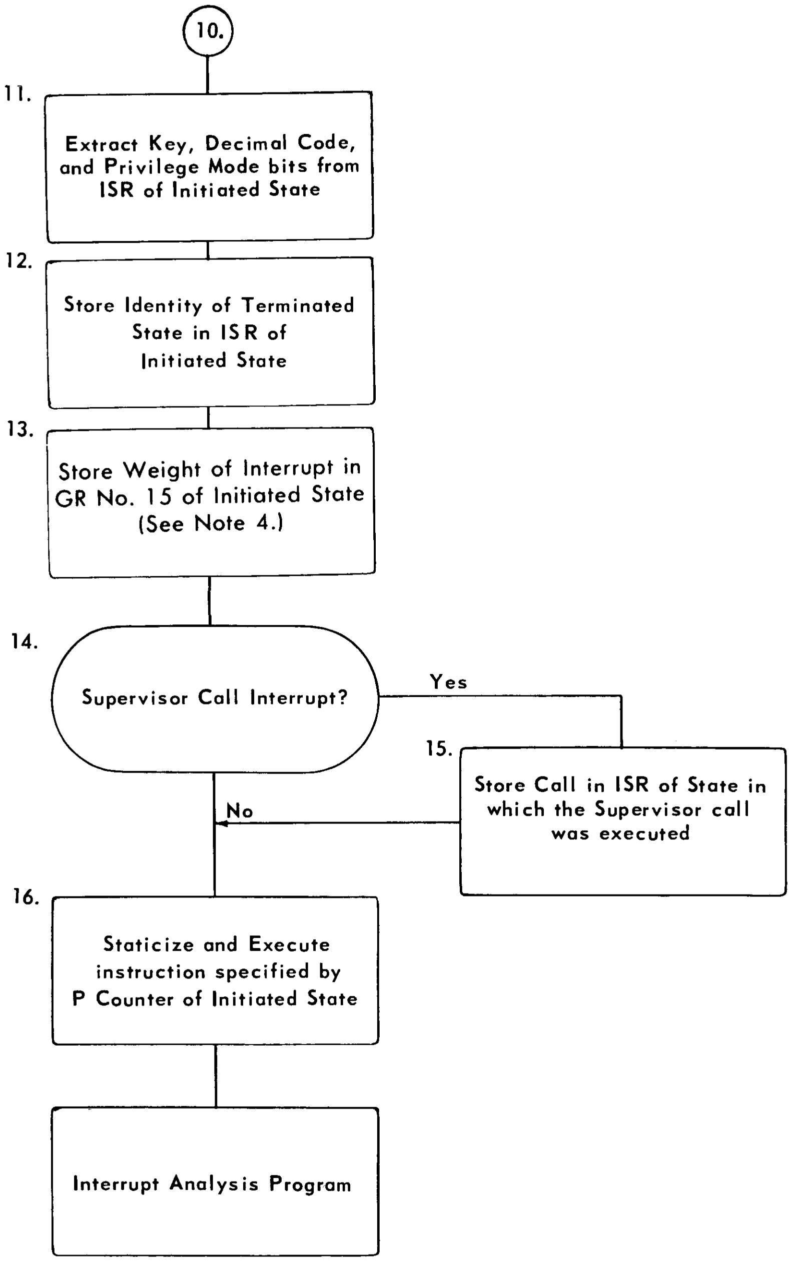

10.
11.
Extract Key, Decimal Code, and Privilege Mode bits from ISR of Initiated State
12.
Store Identity of Terminated State in ISR of Initiated State
13.
Store Weight of Interrupt in GR No. 15 of Initiated State (See Note 4.)
14.
Supervisor Call Interrupt?
Yes
15.
Store Call in ISR of State in which the Supervisor call was executed
No
16.
Staticize and Execute instruction specified by P Counter of Initiated State
Interrupt Analysis Program

Figure 3. Functional Logic of Automatic Interrupt (Cont'd)

Block 2 ◆ If the interrupt condition is one of the above, the program mask (machine register) for the current program state is checked to see if the interrupt is permitted. If the program mask indicates that the interrupt is inhibited (mask = 0), the interrupt condition is cancelled and the next instruction in the current processor state is executed.

Block 3 ◆ If the interrupt condition is not one of the four program interrupts, or is one of the four program interrupts but the program mask indicates that the interrupt is to be permitted (mask = 1), the specific bit associated with the interrupt condition is set in the Interrupt Flag register.

Block 4 ◆ The bit in the Interrupt Flag register is compared with the corresponding bit in the Interrupt Mask register for the current state. If the bit in the Interrupt Mask register is reset (0), the interrupt condition remains pending and the next instruction in the current processor state is executed. The interrupt remains pending until the mask is changed to a permit status and the interrupt is serviced.

Block 5 ◆ If the bit in the Interrupt Mask register is set, the interrupt is taken and information (ILC, CC, program mask) is stored in the P counter of the state being terminated.

Blocks 6 and 7 ◆ If the interrupt condition is a power failure, a machine check, or Scratch Pad Memory parity (if applicable), the Machine Condition State P_4 is initiated. The flag in the Interrupt Flag register is reset.

Block 8 ◆ If the interrupt is a Machine Check, the Program Indicators are stored in the Interrupt Status register of P_4.

Block 9 ◆ If the interrupt condition is not a power failure or machine check, the Interrupt Control State P_3 is initiated. The flag in the Interrupt Flag register is reset.

Block 10 ◆ The condition code setting and the program mask are extracted from the P counter of the initiated state and stored in the appropriate hardware registers.

Block 11 ◆ The memory protection key, the decimal code and the privileged mode bits are extracted from the Interrupt Status register of the initiated state and stored in the appropriate registers.

Block 12 ◆ The state being terminated is identified to the state being initiated by setting an interrupted state identifier code in the Interrupt Status register of the initiated state.

Block 13 ◆ The weight of the condition causing the interrupt is stored in general register No. 15 of the initiated state (P_3 or P_4).

Blocks 14 and 15 ◆ If the interrupt condition is a Supervisor Call, the R_1 and R_2 fields of the Supervisor Call instruction are stored in the rightmost eight-bits of the Interrupt Status register of the state in which the instruction is executed.

Block 16 ◆ The instruction at the address specified in the P counter of the initiated state is staticized and executed.

Program Controlled Interrupt

◆ The Program Control instruction transfers the program from one processor state to another. This instruction is a privileged operation and can be executed only if the state in which the processor is operating is in the privileged mode (bit position 15 of the Interrupt Status register = 0). When a Program Control instruction is executed, the following events occur. (See figure 4.)

Block 1 ◆ The address (B_1/D_1) specified in the Program Control instruction is stored in the P counter of the terminated state. The length of the last instruction executed in the terminated state, the condition code setting, and the program mask are stored in the P counter of the terminated state.

Block 2 ◆ A check is made to see if the program test bit in the Program Control instruction is set.

Block 3 ◆ If the program test bit is not set, the Interrupt Mask register for the state to be initiated by the Program Control instruction is compared to the Interrupt Flag register. If an interrupt condition has occurred, the events described under automatic interrupt take place (see figure 3, block 3).

Important: If an interrupt is outstanding in the state to be initiated by the Program Control instruction, the number of the *initiated state* specified by the Program Control instruction is stored in the interrupt status identifier field of the Interrupt Status register of the initiated state (P_3 or P_4).

Block 4 ◆ If an interrupt condition is not outstanding in the state to be initiated by the Program Control, instruction control is transferred to the state specified by the Program Control instruction (directly or indirectly — See Program Control instruction).

Block 5 ◆ The condition code setting and the program mask are extracted from the P counter of the initiated state and stored in the appropriate machine registers.

Block 6 ◆ The memory protection key, the decimal code and the privileged mode bits are extracted from the Interrupt Status register of the initiated state and stored in the appropriate machine registers.

Block 7 ◆ The instruction at the address specified in the P counter of the initiated state is staticized and executed.

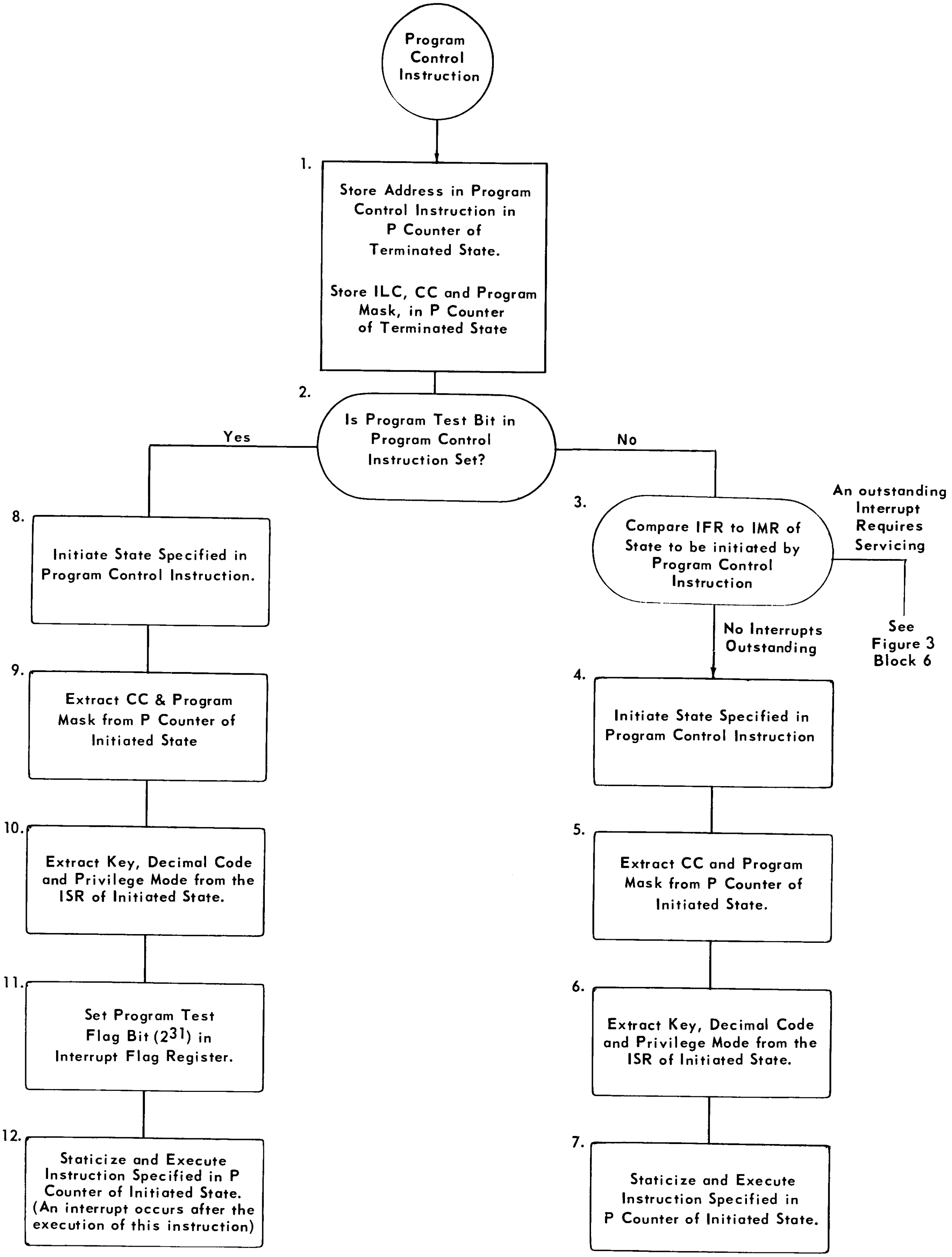

Program Control Instruction
1.
Store Address in Program Control Instruction in P Counter of Terminated State.
Store ILC, CC and Program Mask, in P Counter of Terminated State
2.
Is Program Test Bit in Program Control Instruction Set?
Yes
No
3.
Compare IFR to IMR of State to be initiated by Program Control Instruction
An outstanding Interrupt Requires Servicing
See Figure 3 Block 6
No Interrupts Outstanding
4.
Initiate State Specified in Program Control Instruction
5.
Extract CC and Program Mask from P Counter of Initiated State.
6.
Extract Key, Decimal Code and Privilege Mode from the ISR of Initiated State.
7.
Staticize and Execute Instruction Specified in P Counter of Initiated State.
8.
Initiate State Specified in Program Control Instruction.
9.
Extract CC & Program Mask from P Counter of Initiated State
10.
Extract Key, Decimal Code and Privilege Mode from the ISR of Initiated State.
11.
Set Program Test Flag Bit (2^31) in Interrupt Flag Register.
12.
Staticize and Execute Instruction Specified in P Counter of Initiated State. (An interrupt occurs after the execution of this instruction)

Figure 4. Functional Logic of Program Control Instruction

Block 8 ◆ If the program test bit is set, control is transferred to the state specified by the Program Control instruction (directly or indirectly — see Program Control instruction).

Block 9 ◆ The condition code setting and the program mask are extracted from the P counter of the initiated state and stored in the appropriate registers.

Block 10 ◆ The memory protection key, the decimal code, and the privileged mode bits are extracted from the Interrupt Status register of the initiated state and stored in the appropriate registers.

Block 11 ◆ The program test flag bit (2^{31}) in the Interrupt Flag register is set.

Block 12 ◆ The instruction at the address specified in the P counter of the initiated state is staticized and executed.

Notes:

1. When a Program Control instruction has the program test bit set, the first instruction of the initiated state is always executed before any interrupt is taken.
2. If the initiated state permits the program test interrupt (via the Interrupt Mask register), a program test interrupt occurs after the first instruction in the initiated state is executed.
3. An interrupt condition can occur while executing the first instruction of the initiated state. If it does, and is permitted, it is serviced before the program test interrupt.

General Notes for Program Interrupt:

1. The decimal mode in the 70/46 Processor is either USASCII or EBCDIC as specified by bit 12 in the Interrupt Status register. When an automatic interrupt occurs or a Program Control instruction is executed, the decimal mode is not stored in the Interrupt Status register of the terminated state. The mode of the state being initiated is determined by the mode bit in its own Interrupt Status register.

   Consequently, to change mode, the mode bit of the Interrupt Status register associated with the appropriate state must be altered by the program, and that state must be initiated either by an interrupt condition or a Program Control instruction. This is the method available to the program for changing the mode.

2. The interrupt flags are scanned to determine whether or not an interrupt shall occur if the Interrupt Mask register associated with the current state or the Interrupt Flag register is written into by the program.

3. Changing the protection key, decimal mode, or privileged mode fields in the Interrupt Status register does not change the protection key, machine mode, or privileged mode bits of the associated processor state. To change the status of the processor, the state concerned must be initiated by an interrupt condition or a Program Control instruction.

4. The condition of General register 15 of states P_3 and P_4, at time of interrupt, and loading of the weight is as follows: The low-order 16 bits are cleared. The least significant 7 bits are loaded with the weight. The next most significant 9 bits are zeros. The high-order 16 bits are not cleared, but are shifted one bit to the left.

INPUT/OUTPUT OPERATION

INTRODUCTION

◆ The RCA Model 70/46 Processor can control a variety of input/output devices. All the input/output devices function independently of normal processor operation. This simultaneous operation is achieved by processor input/output channels that control input/output operations. The control electronics of each peripheral device is connected to an input/output channel via the RCA Standard Interface. This interface permits all peripheral equipment (with the exception of remote communications and random access devices) to be attached to any channel in the 70/46 Processor. Remote communication devices must be connected to the multiplexor channel. Random access devices must be connected to a selector channel.

After an input/output operation is initiated by the program, data is transferred, byte-by-byte, between the processor and the peripheral device. This data transfer over the standard interface is controlled by the applicable input/output channel, freeing the processor to continue the program. Each of the channels on the 70/46 Processor can interrupt normal process or operations.

INPUT/OUTPUT CHANNELS

◆ The 70/46 Processor has two types of input/output channels, selector channels and a multiplexor channel.

Selector Channels

◆ Up to four selector channels (optional) can be attached to a 70/46 Processor; each selector channel can address up to 256 peripheral devices.

Provision is made for up to four high-speed selector channels, as options on the 70/46 Processor. The high-speed selector channels reduce main memory interference due to input/output data transfers, such that the maximum aggregate system interference rate is 1000KB per second; however, the maximum data rate on any one selector channel is 465KB.

The programming characteristics of the multiplexor and optional selector channels are identical to the 70/45.

On the 70/46 Processor, each selector channel has two standard interface trunks; each standard interface trunk can be connected to the control electronics of an input/output device. A device control electronics controls one device (i.e., card reader, printer), or a number of devices (i.e., tape controller: up to 16 tape stations).

Only one device can operate on a selector channel at one time. However, all selector channels can operate simultaneously with, and independently of, normal processor operation.

The multiplexor channel operates simultaneously with selector channels and independently of normal processor operation.

Control and operation of each input/output device connected to the multiplexor channel is done through a set of subchannel registers contained in non-addressable main memory.

Selector Channels *(Cont'd)*

In addition to the subchannel registers, four 32-bit registers, called multiplexor registers, are provided in scratch-pad memory. These registers are used for subchannel initiation and termination. Upon servicing a termination interrupt of a device connected to the multiplexor channel, the information which pertains to the completed operation is transferred from the non-addressable main memory to the scratch-pad memory.

The multiplexor registers in scratch-pad memory are called:

Channel Address Register (CAR)
Channel Command Register-II (CCR-II)
Channel Command Register-I (CCR-I)
Assembly/Status Register
Channel Block Address Register

Each selector channel is controlled and operated via four 32-bit registers. These registers are located in scratch-pad memory and are called:

Channel Address Register (CAR)
Channel Command Register-II (CCR-II)
Channel Command Register-I (CCR-I)
Assembly/Status Register
Channel Block Address Register

All the information that is required to control selector channel operation is contained in these registers. Data is transferred between the selector channel and the peripheral device one byte at a time.

Multiplexor Channel

◆ The multiplexor channel is standard on the 70/46 Processor, and can address up to 256 devices.

The multiplexor channel has eight standard interface trunks each of which can be connected to a device control electronics. This permits the multiplexor channel to operate devices on all eight trunks simultaneously. The limit as to the number of input/output devices that can be connected is determined by the device control electronics.

Although the multiplexor channel can handle slow-speed devices on a time-sharing basis, it can accommodate fast devices through a burst mode. Burst mode operation is specified by the program, and causes a transfer of data to occur between a specific device and main memory without time-sharing the multiplexor channel with other input/output devices. If a program is to specify burst mode, a program check is made that other devices on the multiplexor channel have completed operation. This ensures that data is not lost.

Data is transferred between the multiplexor channel and each peripheral device one byte at a time.

Note: When a burst mode operation is executed the subchannel registers are not utilized. The input/output operation is similar to a selector channel operation and is controlled entirely by the multiplexor registers in scratch-pad memory.

INPUT/OUTPUT OPERATIONAL CONTROL

Programming Considerations Prior to Input/Output Initiation

◆ All input/output operations are executed by the selected channel and are independent of normal processor operation. Prior to initiation of an input/output operation, the program must supply information concerning the operation. The program must store information in main memory, such as the type of operation (read, write, etc.), the data area address in main memory at which to begin the operation, and the number of bytes to be transferred by the channel. This information is called the Channel Command Word (CCW).

After the channel command word is stored in main memory, the address of this CCW must be stored in a standard main memory location. This standard location is called the Channel Address Word (CAW) and is main memory locations 72 through 75.

When in 70/46 Mode (T = 1) the address of the Channel Control Block (CCB) must also be stored in a standard main memory location. This standard location is called the Channel Block Address (CBA) and is main memory locations 76 through 79. This address gives the software the address of the Device Status Code.

Once the Channel Address Word, Channel Block Address, and the Channel Command Word have been assembled, the input/output operation can be initiated.

Input/Output Initiation

◆ All input/output operations are initiated by executing a Start Device instruction or by manually pressing the LOAD pushbutton/indicator on the Model 70/97 Console. Execution of the Start Device instruction causes the information contained in the Channel Address Word (CAW), Channel Block Address (CBA) and the Channel Command Word (CCW) to be transferred to the input/output channel registers in scratch-pad memory for the specified selector channel. If the specified channel is the multiplexor channel, this information is transferred to the subchannel registers in non-addressable main memory for the specified device. Once this has been accomplished, the Start Device instruction terminates and the input/output operation has been initiated. Completion of the input/output operation is under control of the channel, and normal processor operation can proceed.

Channel Servicing

Servicing a Data Transfer

◆ When an input/output operation has been initiated and the input/output device control electronics is ready to send or receive a data byte, the channel asks the processor for a service request. When the processor permits the service request, a data transfer occurs. This service permits the transfer of a data byte between main memory and the input/output device to occur. It also updates the information in the input/output channel registers or the subchannel registers (multiplexor) to prepare for the next data byte.

End and Chaining Servicing

◆ When an input/output operation has been completed, the channel asks the processor for another service request. This service request is required so that the channel can (1) tell the device control electronics to set a channel interrupt condition, or (2) check the current command to see if chaining is specified, and if it is to initiate the next command.

Interrupt Servicing

◆ If an input/output operation has been completed and chaining has not been specified, the input/output device control electronics causes the appropriate channel interrupt flag to be set in the Interrupt Flag register. If the Interrupt Mask register for the current processor state permits the interrupt, it is taken. At this time the channel asks the processor for another service request. This service request is required so that the channel can transfer information concerning the status of the device and the channel to the input/output channel registers in scratch-pad memory. If the interrupt is caused by a device on the multiplexor channel, the appropriate subchannel registers are transferred from non-addressable main memory to scratch-pad memory.

Because all input/output servicing requires that the channel utilize main memory, scratch-pad memory and nonaddressable main memory (multiplexor devices), normal processor operation is *held-off* until the servicing has been completed. Servicing is time-shared with normal mode processing.

Servicing Priority

◆ Because input/output operations on all selector channels and the multiplexor channel proceed simultaneously, the processing is stopped if servicing is required and the input/output device is serviced. After a device is serviced, processing resumes, or another device is serviced.

Each selector channel and the multiplexor channel has a scanning priority. If servicing is required by devices on more than one channel, the channel with the highest priority is serviced first. The priority is as follows:

Selector Channel No. 1

Selector Channel No. 2

Selector Channel No. 3

Selector Channel No. 4

Multiplexor Channel

The devices on the multiplexor have a priority depending upon the standard interface trunk to which they are connected; the lower the standard interface trunk in the scanning sequence, the higher the priority.

Servicing of a device connected to the multiplexor channel may be temporarily interrupted by a selector channel service request. If this occurs, all selector channels requiring service are served before multiplexor channel servicing resumes.

The optimum connection of device control electronics to selector channels and the multiplexor channel depends on the requirements of each installation. However, a general rule is to connect the device control electronics which control devices with the highest data transfer requirements to the channels with the highest priority. The remaining device control electronics are connected in descending order of data transfer requirements to descending priority sequence of channels. Buffered devices should always have lowest priority.

Channel Address Word (CAW)

◆ The Channel Address Word (CAW) is used by the Start Device instruction (see Privileged Instructions section), and specifies the address of the first Channel Command Word (CCW) used to control the operation of the input/output device. If the Memory Protect option is installed, the memory protection key must also be stored in the CAW before a Start Device instruction is issued.

The CAW must be stored in main memory locations 72 through 75 before executing a Start Device instruction and has the following format:

Key	0000	Address of CCW
0 3	4 7	8 31

Bit Positions 0 through 3 contain the memory protection key. It is used to ensure that data is not being transferred to a protected memory area. If the Memory Protect option is not installed, these bits must be zero.

Bit Positions 4 through 7 are reserved for future expansion.

Bit Positions 8 through 31 contain the main memory address of the initial channel command word.

Channel Block Address (CBA)

◆ The Channel Block Address (CBA) is used by the Start Device instruction (see Privileged Instructions Section), and specifies that the Start Device Operation was initiated (CC = 0). In this event, the contents of the CBA are loaded into the selector channel registers as follows:

Selector No.	Scratch Pad Word Address
1	36
2	66
3	76
4	A6

If multiplexing, the contents of the CBA are loaded into sub-channel registers XX00 — XX03, where XX is the device number. The CBA must be stored in main memory locations 76 through 79 before executing a Start Device instruction and has the following format:

Reserved	Address of Channel Control Block
0 7	8 31

Bit Positions 8 through 31 contain the main memory address of the Channel Control Block.

Note: The CCW and the CBA addresses are 24 bits and direct (not translatable).

Channel Command Word (CCW)

◆ The Channel Command Word (CCW) supplies the information for controlling the operation of the input/output device. This information must be stored in main memory by the program before a Start Device instruction is issued. The CCW consists of two 32-bit words in main memory that must be aligned on a double word boundary. The CCW has the following format:

Channel Command Word (CCW) *(Cont'd)*

Command Code	Address of First Data Byte or Address of Next CCW if Command is a Transfer in Channel
0 7	8 31

Flags	Reserved for Future Expansion	Byte Count
32 36	37 47	48 63

Bit Positions 0 through 7 contain the command code, which specifies the operation to be performed by the I/O device. (See table 10.)

Table 10. Command Code Operations

Command Code								Operation
0	1	2	3	4	5	6	7	Bit Position
M	M	M	M	0	0	0	1	Sense
M	M	M	M	M/0	0	1	0	Read Reverse
M	M	M	M/B	M/0	0	1	1	Write
M	M	M	M/B	M/0	1	0	0	Write Erase
M	M	M	M/B	M/0	1	0	1	Read
M	M	M	M	0	1	1	1	Write Control
M	M	M	M	1	0	0	1	Transfer in Channel

Notes:

1. Any command code other than the ones shown in table 10 is illegal and must not be programmed. If this rule is violated, the resulting effect on the channel and device is unpredictable. If one of the legal commands is issued to a device which is not capable of accepting the operation (i.e., a Write command is issued to a card reader), the command, after being accepted, is terminated by the device control electronics. A channel interrupt occurs and the sense byte(s) indicate the illegal operation.
2. The bit position designated as "B" indicates that the specified device is connected to the multiplexor channel and the multiplexor is to be operated in the burst mode. If this position is a 1 bit, the multiplexor channel is *locked-on* to the selected device, and the servicing of other devices connected to the multiplexor channel is inhibited. A burst mode can only be initiated when it is specified in the first command of a chain. Subsequent commands, linked by chaining, cannot initiate a burst mode. However, if the first command in a chain specifies burst mode, all commands in the chain are executed under burst mode conditions.
3. Bit positions designated as M (modifier) indicate variations of the operation and are unique to the specific input/output device.

An explanation of the commands shown in table 10 is as follows:

Sense (0001) — Information is transferred from the specified input/output device control electronics to main memory. The information trans-

Channel Command Word (CCW) *(Cont'd)*

ferred indicates unusual conditions that occurred as a result of the last operation performed by the device. (The information received is defined in the individual input/output device reference manuals.) The address specified by the CCW is the leftmost main memory location of the input area.

Note: Parity is not checked on data transferred to main memory by this command.

Read Reverse (0010) — Information is transferred from the specified input/output device to main memory in descending order. The address specified by the CCW is the rightmost main memory location of the input area.

Write (0011) — Information is transferred from main memory to the specified input/output device. The address specified by the CCW is the leftmost main memory location of the output area.

Write Erase (0100) — Information is transferred from main memory to the specified input/output device control electronics. Data is not written to tape and the tape is erased in accordance with the byte count (applicable to magnetic tape only). The address specified by the CCW is the leftmost main memory location of the output area.

Read (0101) — Information is transferred from the specified input/output device to main memory in ascending order. The address specified by the CCW is the leftmost main memory location of the input area.

Write Control (0111) — Information is transferred from main memory to the specified input/output device control electronics. The device control electronics interprets this information as control information and initiates a function not involving a data transfer. The address specified by the CCW is the leftmost main memory location of the output area.

Transfer in Channel (1001) — This command provides chaining of CCW's that are not located in adjacent double word main memory. An actual branch to the address of the next CCW is taken. The branch address (specified in bits 8 through 31 of the channel command word) must specify a double word location. (Bits 29 through 31 must be zero.) This command cannot be the first command in a chain. A Transfer in Channel command may address another Transfer in Channel command.

Note: The flag bits are ignored if a Transfer in Channel command is specified. The flag bits of the preceding command remain effective.

Bit Positions 8 through 31 (see CCW format) contain the address of the first byte in main memory at which the input/output operation begins, or if the command is a transfer in channel, the main memory address of the next CCW to be executed. The address of the first byte of the next data segment can also be specified if data chaining.

Bit Positions 32 through 36 are the flag bits and have the following significance:

1. Bit position 32 is the Chain Data flag (CD). In addition to transferring data to and from a single main memory area, the 70/46 Processor can read into, or write from, many non-contiguous areas of main memory by executing one Start Device instruction. When data chaining is specified by setting this bit, a chain (series of channel command words in sequence) is used and each channel com-

Channel Command Word (CCW) *(Cont'd)*

mand word designates an area of main memory at which to continue the current operation. When one channel command word has a lapsed byte count, the next channel command word in sequence is automatically fetched. The current operation is continued at the main memory area specified by the new channel command word. The command code of the new CCW is ignored unless it specifies a Transfer in Channel. If any of the following channel status byte conditions occur, the chain is terminated (see Channel Status Byte for further definition):

Program Check

Protection Check

Channel Control Check

Channel Data Check (if the operation is a write)

Incorrect Length Condition

When data chaining, the chain data flag in the last channel command word must be reset. This causes the data chain to be terminated upon completion of the operation specified by this CCW.

2. Bit position 33 is the Chain Command flag (CC). The 70/46 Processor can perform several operations to an input/output device by executing one Start Device instruction. When command chaining is specified by setting this bit, a chain (series of channel command words in sequence) is used and each channel command word specifies the operation to be performed. When the operation specified by one channel command word is completed, the next channel command word in sequence is automatically fetched and the operation specified is initiated. If any of the following conditions occur, the chain is terminated:

   a. Channel status byte conditions (see channel status byte for further definition).

      Incorrect Length Condition and suppress length flag is zero.

      Program Check

      Protection Check

      Channel Control Check

   b. Standard device byte conditions (see standard device byte for further definition).

      Secondary Indicator is set

      Device Inoperable is set

      Device End is not set

   When command chaining, the chain command flag in the last channel command word must be reset. This causes the command chain to be terminated upon completion of the operation specified by this CCW.

3. Bit position 34 is the Suppress Length Indication flag (SLI). Incorrect length occurs in the 70/46 Processor when the number of bytes specified in the channel command word is not equal to the

Channel Command Word (CCW) *(Cont'd)*

number of bytes sought by, or sent from, the input/output device. (When a command or chain of commands terminates, the data byte count has not lapsed.) An example of an incorrect length condition is a tape read which terminates on a gap before the byte count has lapsed. If the SLI bit is set, the program does not receive an indication of an incorrect length upon termination of the input/output operation. If the SLI bit is reset, the program receives an indication of an incorrect length upon termination of the input/output operation. This indication is contained in the channel status byte.

Notes:

1. If the SLI bit is set and the chain data flag of the final CCW in a chain is reset, the incorrect length indication is suppressed, if it occurs.
2. If the chain data flag of a CCW is set and an incorrect length condition occurs, the program is notified of the condition regardless of the setting of the SLI flag.
3. Bit position 35 is the Skip flag (SKIP). In conjunction with data chaining, portions of a block of information can be suppressed during an input operation. If this bit is set, the transfer of data to main memory specified by this command is suppressed. This bit can be used only with Read, Read Reverse or Sense commands.
4. Bit position 36 is the Program Controlled Interrupt flag (PCI). During data and command chaining, the 70/46 Processor has the ability to notify the program of the progress of chaining through an interrupt when a channel command word is fetched. When this bit is set, a channel interrupt occurs when the channel command word is fetched from main memory and the first data byte has been transferred. This flag is ineffective if the channel is the multiplexor operating in burst mode.
5. Bit positions 37 through 47 are reserved for future expansion and must be set to all zeros by the program.
6. Bit positions 48 through 63 contain the count of the number of bytes to be transferred to or from main memory during the input/output operation (from 0 to 65,536 bytes). An initial count of zero specifies the maximum number of bytes to be transferred.

The staticizing of 70/46 I/O instructions is subject to translation similar to any other instructions, except for the execution of I/O instructions data servicing, and interrupts that utilize direct addresses only, and are not subject to dynamic translation.

1. Address of I/O instruction	— Translatable
2. I/O instructions staticized address	— Direct
3. Address of CAW	— Direct
4. Address of CCW	— Direct
5. Address of CCB	— Direct
6. Contents of CCW	— Direct

INPUT/OUTPUT CHANNEL REGISTERS

◆ The Channel Address Word (CAW), Channel Block Address (CBA), and the Channel Command Word(s) (CCW) are stored by the program in main memory. However, when an input/output operation is initiated, the information contained in the CAW, CBA, and the first CCW is transferred to the scratch-pad input/output channel registers for the channel specified by the Start Device instruction. (See table 11.) Because the access speed in scratch-pad memory is faster than main memory, faster servicing of input/output devices is possible. These registers also eliminate the need for the program to reset channel command words, because incrementing and decrementing addresses and byte count is done in scratch-pad memory. These registers allow the input/output operation to proceed under control of the specified channel, thereby permitting normal mode processing to continue.

Table 11. Input/Output Channel Registers

Register	Selector Channel	Multiplexor Channel	
	Scratch-Pad Memory	Scratch-Pad Memory	Non-Addressable Main Memory
Channel Address Register (CAR)	1 per selector channel	1 per multiplexor channel	1 per device
Channel Command Register-I (CCR-I)	1 per selector channel	1 per multiplexor channel	1 per device
Channel Command Register-II (CCR-II)	1 per selector channel	1 per multiplexor channel	1 per device
Assembly/Status Register	1 per selector channel	1 per multiplexor channel	None
CBA Register	1 per selector channel	1 per multiplexor channel	1 per device

The format for each of these four 32-bit registers is as follows:

Channel Address Register (CAR)

Device No.	Address of next CCW
0 7	8 31

Bit Positions 0 through 7 contain the device number specified in the input/output operation. This number is obtained from the B_1/D_1 Address in the Start Device instruction.

Bit Positions 8 through 31 contain the address of the next channel command word if chaining is specified. This information is obtained by incrementing the address of the first CCW by eight. The address of the first CCW is obtained from the Channel Address Word (CAW).

Channel Command Register-II (CCR-II)

Flags	000	Channel Status Byte	Byte Count
0 4	5 7	8 15	16 31

Bit Positions 0 through 4 contain the flags. The flags are obtained from the channel command word. The flag bits are defined as follows:

Bit 0 — Chain data flag (CD)

Bit 1 — Chain command flag (CC)

Bit 2 — Suppress length indicator flag (SLI)

Channel Command Register-II (CCR-II) *(Cont'd)*

Bit 3 — Skip flag (SKIP)

Bit 4 — Program controlled interrupt flag (PCI)

Bit Positions 5 through 7 are reserved for future use.

Bit Positions 8 through 15 contain the channel status byte. The bits of the channel status byte are generated as a result of the input/output operation and are defined as follows:

Bit 8 — Program Controlled Interrupt

Bit 9 — Incorrect Length

Bit 10 — Program Check

Bit 11 — Protection Check

Bit 12 — Channel Data Check

Bit 13 — Channel Control Check

Bit 14 — Reserved for use by the processor

Bit 15 — Termination Interrupt

(For a detailed description of the above see Channel Status Byte section, page 61.)

Bit Positions 16 through 31 contain the number of bytes of main memory to or from which data is transferred. This information is obtained from the Channel Command Word. The count can range from 0 bytes to 65,536 bytes. When the I/O is terminated, these bit positions contain the remaining byte count (if any).

Channel Command Register-I (CCR-I)

0000	Command Code	Data Address of First Byte or Location of new CCW if Command is Transfer in Channel
0 3	4 7	8 31

Bit Positions 0 through 3 are used by the processor. It should be noted that these bits are used in the channel command word as modifier bits. Once the command has been initiated and the entire 8-bit command code has been sent to the specified device control electronics, these bits are used by the processor. They no longer contain the modifier bits.

Bit Positions 4 through 7 contain the command code. This code is obtained from the channel command word. The commands are defined as follows:

Read (0101)

Write (0011)

Write Control (0111)

Sense (0001)

Read Reverse (0010)

Write Erase (0100)

Transfer in Channel (1001)

Bit Positions 8 through 31 contain the address of the initial byte in main memory at which the operation begins; or contains the branch address if the command is a Transfer in Channel. This information is obtained from the Channel Command Word.

Assembly/Status Register

Data Bytes	Standard Device Byte
0 — 23	24 — 31

Bit Positions 0 through 31 are for equipment use only.

When the device status is stored as a result of an input/output operation, bit positions 24 through 31 of the assembly/status register are used to store the standard device byte. The bits of the standard device byte supply status information pertaining to the device control electronics and the input/output device and are defined as follows:

Bit 24 — External Device Request Interrupt Pending
Bit 25 — Terminating Interrupt Pending
Bit 26 — Device Busy
Bit 27 — Control Busy (not applicable)
Bit 28 — Device End
Bit 29 — Second Indicator
Bit 30 — Device Inoperable
Bit 31 — Status Modifier

(For a detailed description of the above, see Standard Device Byte section, page 65.)

CBA Register

◆ Not modified by channel, equals CBA originally fetched from 76-79.

INPUT/OUTPUT INSTRUCTIONS

◆ There are four processor instructions which are concerned with input/output operations. They are Start Device, Halt Device, Check Channel and Test Device. These instructions are executed by the processor and the results, in the form of condition codes, are available upon instruction completion. It should be noted that the condition code settings indicate the results of the instruction and not the results of the input/output operation that the instruction may be initiating. The channel continues off-line to accomplish the input/output operation as specified by the instruction. However, during this time the processor continues executing subsequent instructions.

Start Device Instruction

◆ The Start Device instruction is a privileged operation and, therefore, can be executed only if the mode bit (bit position 15 of the Interrupt Status register for the current state) is set to zero. This instruction is executed in the normal mode. Continuation of program execution is delayed until the Start Device instruction has been terminated.

Upon execution of a Start Device instruction, the following events occur. (See figure 5.)

Block 1

◆ If the privileged mode bit (bit position 15 of the Interrupt Status register) for the current state is not set to zero, the privileged operation bit is set in the Interrupt Flag register and an interrupt occurs (if permitted).

Block 2

◆ If the specified channel is a selector channel that is not available on the system, the condition code is set to 3, the Start Device instruction is terminated and program control is transferred to the next instruction. The input/output operation is not initiated.

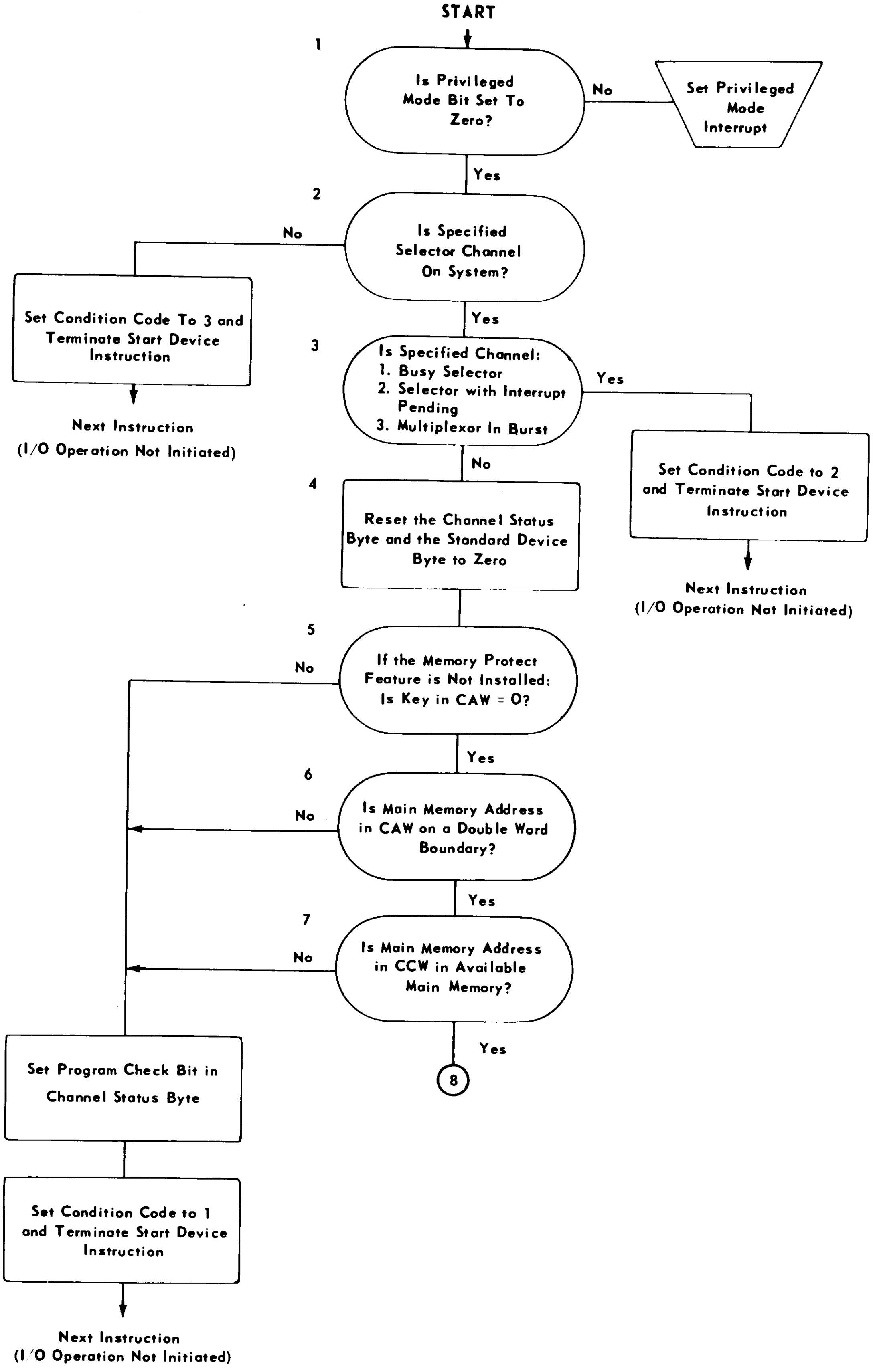

START
1
Is Privileged Mode Bit Set To Zero?
No
Set Privileged Mode Interrupt
Yes
2
Is Specified Selector Channel On System?
No
Set Condition Code To 3 and Terminate Start Device Instruction
Next Instruction
(I/O Operation Not Initiated)
Yes
3
Is Specified Channel:
1. Busy Selector
2. Selector with Interrupt Pending
3. Multiplexor In Burst
Yes
Set Condition Code to 2 and Terminate Start Device Instruction
Next Instruction
(I/O Operation Not Initiated)
No
4
Reset the Channel Status Byte and the Standard Device Byte to Zero
5
If the Memory Protect Feature is Not Installed: Is Key in CAW = 0?
No
Yes
6
Is Main Memory Address in CAW on a Double Word Boundary?
No
Yes
7
Is Main Memory Address in CCW in Available Main Memory?
No
Yes
8
Set Program Check Bit in Channel Status Byte
Set Condition Code to 1 and Terminate Start Device Instruction
Next Instruction
(I/O Operation Not Initiated)

Figure 5. Functional Logic of Start Device Instruction

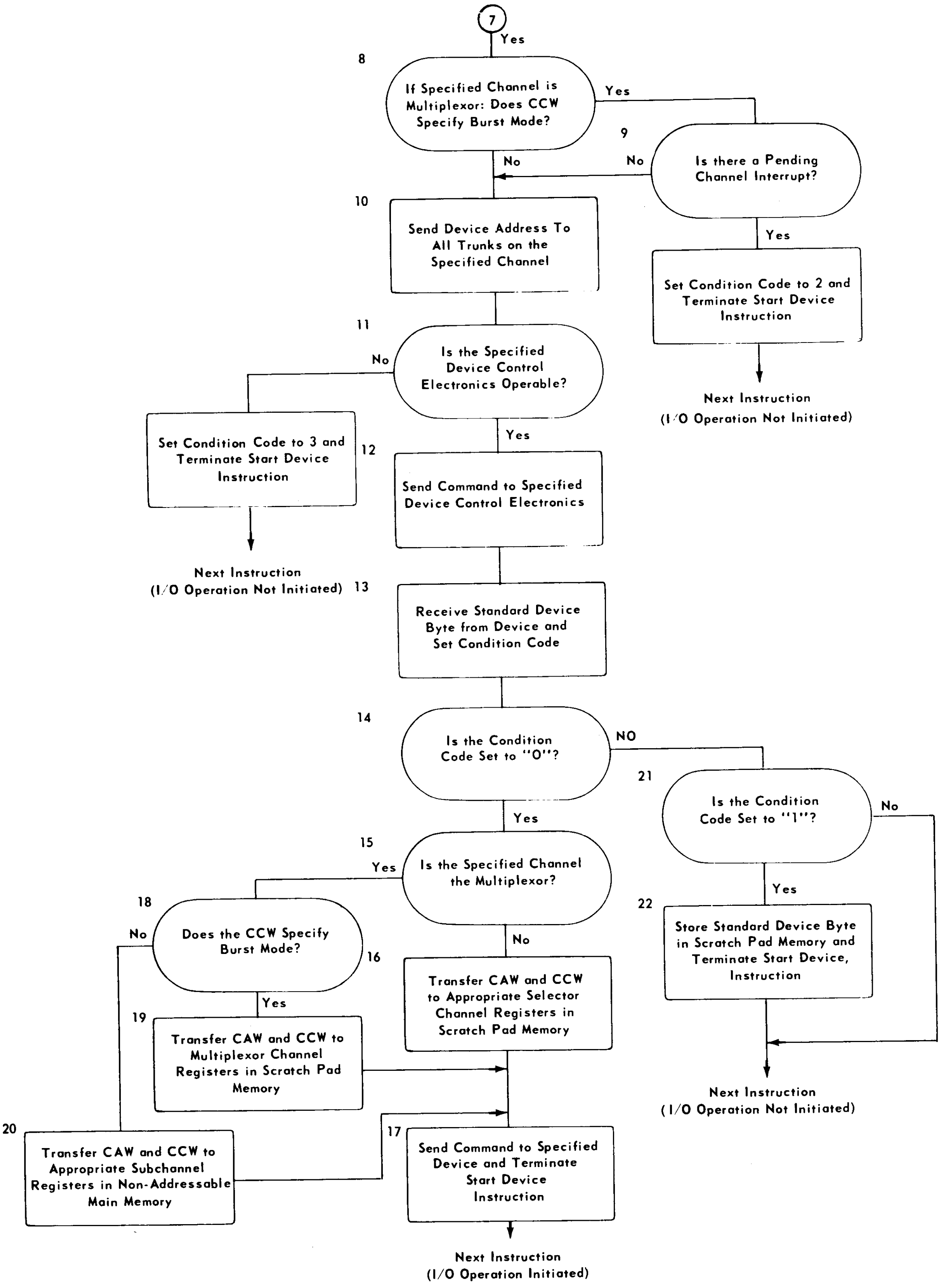

7
Yes
8
If Specified Channel is Multiplexor: Does CCW Specify Burst Mode?
Yes
No
9
No
Is there a Pending Channel Interrupt?
Yes
Set Condition Code to 2 and Terminate Start Device Instruction
Next Instruction
(I/O Operation Not Initiated)
10
Send Device Address To All Trunks on the Specified Channel
11
No
Is the Specified Device Control Electronics Operable?
Set Condition Code to 3 and Terminate Start Device Instruction
Next Instruction
(I/O Operation Not Initiated)
Yes
12
Send Command to Specified Device Control Electronics
13
Receive Standard Device Byte from Device and Set Condition Code
14
Is the Condition Code Set to "O"?
NO
21
Is the Condition Code Set to "1"?
No
Yes
22
Store Standard Device Byte in Scratch Pad Memory and Terminate Start Device, Instruction
Next Instruction
(I/O Operation Not Initiated)
Yes
15
Yes
Is the Specified Channel the Multiplexor?
18
No
Does the CCW Specify Burst Mode?
Yes
19
Transfer CAW and CCW to Multiplexor Channel Registers in Scratch Pad Memory
20
Transfer CAW and CCW to Appropriate Subchannel Registers in Non-Addressable Main Memory
No
16
Transfer CAW and CCW to Appropriate Selector Channel Registers in Scratch Pad Memory
17
Send Command to Specified Device and Terminate Start Device Instruction
Next Instruction
(I/O Operation Initiated)

Figure 5. Functional Logic of Start Device Instruction (Cont'd)

Block 3 ◆ If the specified channel is a selector channel that is busy or has an interrupt pending (termination or external device request) or if the specified channel is the multiplexor that is operating in the burst mode, the condition code is set to 2, the Start Device instruction is terminated and program control is transferred to the next instruction. The input/output operation is not initiated.

Block 4 ◆ The channel status byte and the standard device byte for the specified channel are reset to zeros in the appropriate channel registers.

Block 5 ◆ If the Memory Protect feature is not installed, the key in the Channel Address Word (CAW) is tested to see if it is equal to zeros. If it is not zeros, the program check bit in the channel status byte is set, the condition code is set to 1, the Start Device instruction is terminated, and program control is transferred to the next instruction. The input/output operation is not initiated.

Block 6 ◆ The main memory address in the Channel Address Word is tested to see if it is on a double word boundary. If it is not, the program check bit in the channel status byte is set, the condition code is set to 1, the Start Device instruction is terminated and program control is transferred to the next instruction. The input/output operation is not initiated.

Block 7 ◆ The main memory address in the Channel Command Word (CCW) is tested to see if it is within the available main memory for the system. If it is not, the program check bit in the channel status byte is set, the condition code is set to 1, the Start Device instruction is terminated and program control is transferred to the next instruction. The input/output operation is not initiated.

Block 8 ◆ If the specified channel is the multiplexor channel, the command code in the Channel Command Word is tested to see if a burst mode operation has been specified.

Block 9 ◆ If a burst mode operation has been specified, a test is made to see if there is a terminating interrupt pending on any of the trunks on the multiplexor. If a terminating interrupt is pending, the condition code is set to 2, the Start Device instruction is terminated and program control is transferred to the next instruction. The input/output operation is not initiated.

Block 10 ◆ The device address as specified in the Start Device instruction is sent to all trunks on the addressed channel.

Block 11 ◆ A test is made to see if the specified device control electronics is operable. The device control electronics has 50 microseconds to signal the processor that it is operable. If it does not, the condition code is set to 3, the Start Device instruction is terminated and program control is transferred to the next instruction. The input/output operation is not initiated.

Block 12 ◆ If the specified device control electronics is operable, the command code from the Channel Command Word is sent to the specified device control electronics.

Block 13 ◆ After receiving the command code, the device control electronics sends the standard device byte to the processor. This standard device byte is *not* stored in the channel registers in scratch-pad memory. It is used to set the proper condition code as follows:

Condition Code	**Definition**
3	Device control electronics is inoperable.
2	A termination interrupt pending condition exists in the device control electronics on the multiplexor channel.
2	The device control electronics is busy working with the specified device.
2	The device control electronics is busy working with a device other than the one specified.
1	An external device request interrupt pending condition exists in the device control electronics on the multiplexor channel.
1	The specified device is busy but the device control electronics is not busy (i.e., tape rewinding, off-line seek).
*1	The specified device is inoperable.
0	The specified device and control electronics is available.

* If the command is a Sense, the condition code is set to 0 permitting the operation to be initiated.

Block 14 ◆ A test is made to see if the condition code is set to 0 (input/output operation can be initiated).

Block 15 ◆ If the condition code is zero, a test is made to see if the specified channel is the multiplexor channel.

Block 16 ◆ If the specified channel is a selector channel, the Channel Address Word and Channel Block Address (if T = 1) are fetched from main memory locations 72 through 79 and stored in the appropriate channel address register. Using the main memory address specified in the CAW, the Channel Command Word is fetched from main memory and stored in the appropriate channel command registers.

Block 17 ◆ The command is sent to the specified device control electronics and the Start Device is terminated (with the condition code set to 0). The input/output operation is initiated and proceeds under control of the appropriate channel and registers in scratch-pad memory and non-addressable main memory (multiplexor devices). Normal program execution of the next instruction continues simultaneously with the input/output operation.

Block 18 ◆ If the specified channel is the multiplexor channel, the command code in the Channel Command Word is tested to see if a burst mode operation has been specified.

Block 19 ◆ If a burst mode operation has been specified, the Channel Address Word and Channel Block Address (if T = 1) are fetched from main memory locations 72 through 79 and stored in the channel address register for the multiplexor channel. Using the main memory address specified in the CAW, the Channel Command Word is fetched and stored in the channel command registers for the multiplexor channel.

Block 20 ◆ If a burst mode operation has not been specified, the Channel Address Word and the Channel Command Word are fetched from main memory and stored in the subchannel registers in non-addressable main memory for the device specified.

Block 21 ◆ If the condition code is not set to 0, a test is made to see if the condition code is set to 1.

Block 22 ◆ If the condition code is set to 1, the standard device byte is transferred to the channel registers for the channel specified, the Start Device instruction is terminated and program control is transferred to the next instruction. The input/output operation is not initiated.

Notes on Start Device Instruction

1. The channel status byte and the standard device byte are not stored if the condition codes are 0, 2, 3.
2. If the specified channel and device can be initiated (CC = 0) the contents of the Channel Address Word, Channel Block Address (if T = 1), and Channel Command Word are loaded into the appropriate channel registers and the command is sent to the device. The legality of the command is not determined at initiation time. If the device gets an illegal command, the operation is terminated and a channel interrupt occurs. The standard device byte (stored in the appropriate channel registers when the interrupt is taken) indicates a secondary indicator. A Sense command must be issued to bring the Sense byte(s) into main memory. The Sense byte(s) indicate the illegal operation.
3. If execution of this instruction causes the channel status byte or the standard device byte to be stored, the program must inhibit interrupts on this channel until the status byte has been analyzed or moved from the channel registers. If interrupts are permitted and one occurs the standard device byte and the channel status byte are destroyed.

Halt Device Instruction

◆ The Halt Device instruction is a privileged operation and can be executed only if the mode bit (bit position 15 of the Interrupt Status register) for the current state is set to 0. This instruction is executed in the normal mode. Continuation of program execution is delayed until termination is accepted by the device control electronics. When the device control electronics receives the termination, it causes a channel interrupt to occur. Both the channel number and the device number must be specified in the instruction. Because the Channel Address Word is not referred to by the Halt Device instruction, the Channel Address Word, Channel Block Address, and a Channel Command Word are not required.

Upon execution of a Halt Device instruction, the following events occur (see figure 6).

Block 1 ◆ If the privileged mode bit (bit position 15 of the Interrupt Status register) for the current state is not set to zero, the privileged operation bit is set in the Interrupt Flag register and an interrupt occurs (if permitted).

Block 2 ◆ If the specified channel is a selector channel which is not available on the system, the condition code is set to 3, the Halt Device instruction is terminated and program control is transferred to the next instruction.

Block 3 ◆ If the specified channel is a selector channel that is busy or if the specified channel is the multiplexor that is operating in the burst mode, the Chain Command (CC) flag in CCR-II is reset, the device control electronics is told to set an end condition, the condition code is set to 2, the Halt Device instruction is terminated, and program control is transferred to the next instruction.

Notes:

1. Setting an end condition causes the device to be halted on servicing the next data transfer (see Servicing a Data Transfer).
2. The Chain Command flag must be reset to suppress chaining during termination (see Chaining and End Servicing section, below).

Block 4 ◆ If the specified channel is not the multiplexor channel, the condition code is set to 0, the Halt Device instruction is terminated and program control is transferred to the next instruction.

Block 5 ◆ If the specified channel is the multiplexor channel, the channel status byte and the standard device byte are reset to zeros in the multiplexor channel registers.

Block 6 ◆ The device address as specified in the Start Device instruction is sent to all trunks on the multiplexor channel.

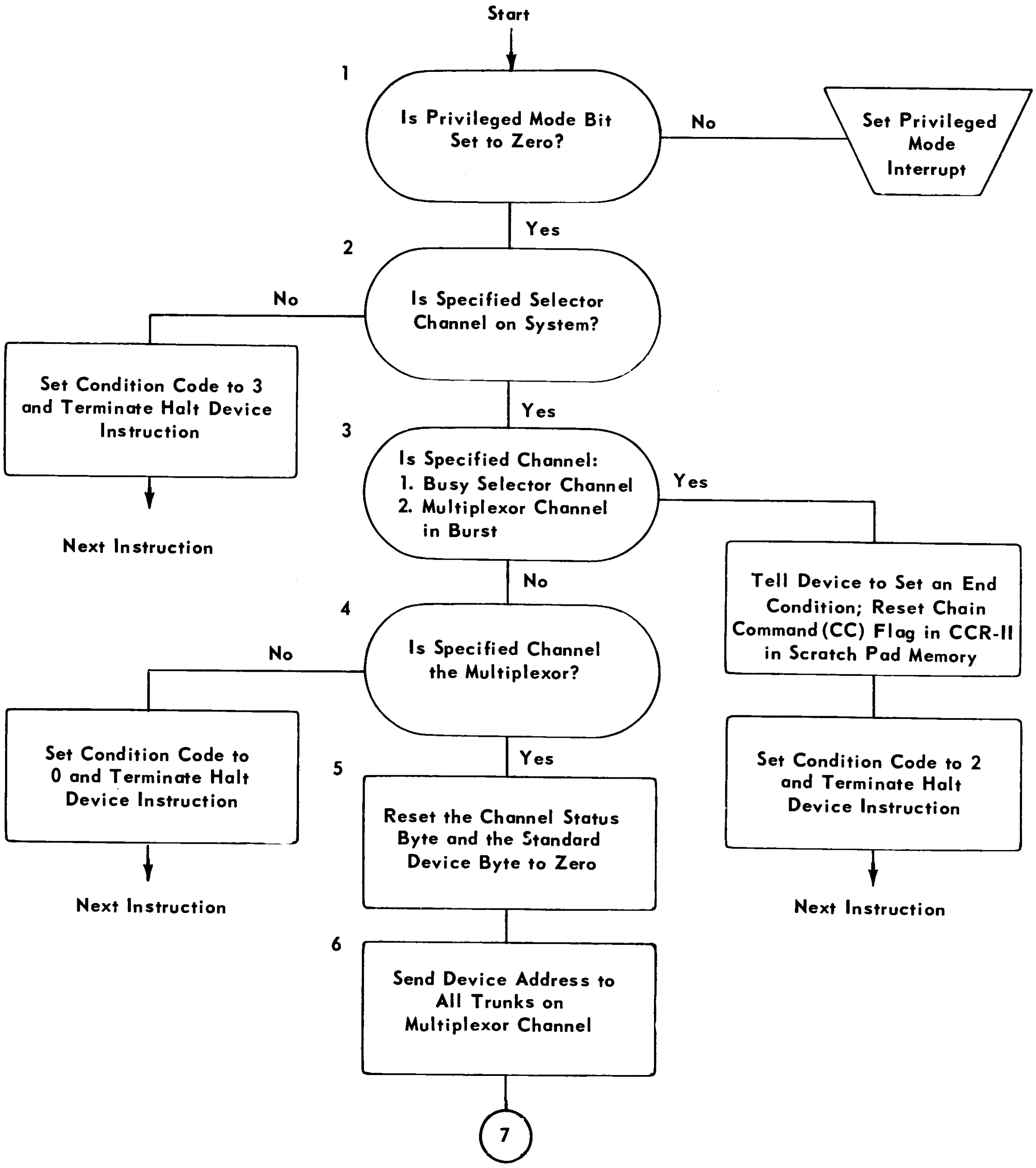
Block 7 ◆ A test is made to see if the specified device control electronics is operable. The device control electronics has 50 microseconds to signal the processor that it is operable. If it does not the condition code is set to 3, the Halt Device instruction is terminated and program control is transferred to the next instruction.

Block 8 ◆ If the specified device control electronics is operable, it sends the standard device byte to the processor. This standard device byte is *not* stored in the channel registers.

Block 9 ◆ If the condition code is not set to 1 (it is 0, 3) the Halt Device instruction is terminated and program control is transferred to the next instruction.

Notes on Halt Device instruction:

1. The channel status byte is not stored as a reult of this operation. However, the incorrect length bit in the channel status byte may be set.



Start
1
Is Privileged Mode Bit Set to Zero?
No
Set Privileged Mode Interrupt
Yes
2
Is Specified Selector Channel on System?
No
Set Condition Code to 3 and Terminate Halt Device Instruction
Next Instruction
Yes
3
Is Specified Channel:
1. Busy Selector Channel
2. Multiplexor Channel in Burst
Yes
Tell Device to Set an End Condition; Reset Chain Command (CC) Flag in CCR-II in Scratch Pad Memory
Set Condition Code to 2 and Terminate Halt Device Instruction
Next Instruction
No
4
Is Specified Channel the Multiplexor?
No
Set Condition Code to 0 and Terminate Halt Device Instruction
Next Instruction
Yes
5
Reset the Channel Status Byte and the Standard Device Byte to Zero
6
Send Device Address to All Trunks on Multiplexor Channel
7

Figure 6. Functional Logic of Halt Device Instruction

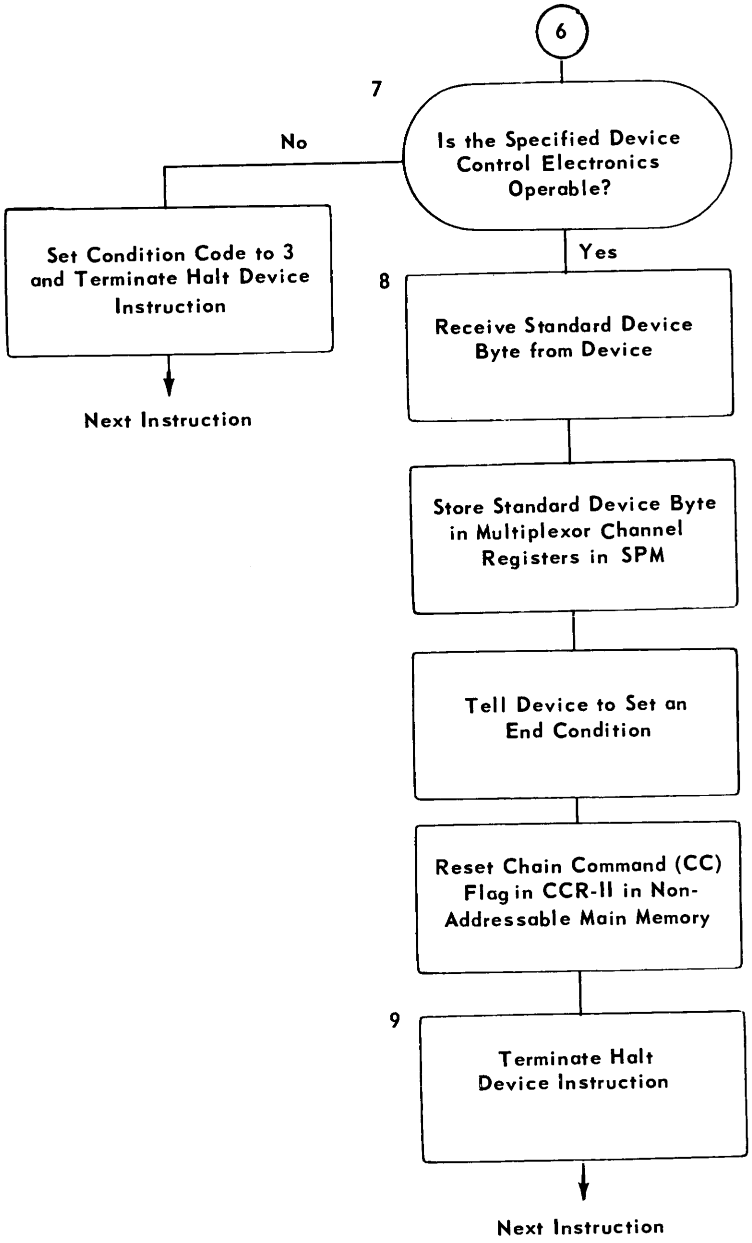

6
7
Is the Specified Device Control Electronics Operable?
No
Set Condition Code to 3 and Terminate Halt Device Instruction
Next Instruction
Yes
8
Receive Standard Device Byte from Device
Store Standard Device Byte in Multiplexor Channel Registers in SPM
Tell Device to Set an End Condition
Reset Chain Command (CC) Flag in CCR-II in Non-Addressable Main Memory
9
Terminate Halt Device Instruction
Next Instruction

Figure 6. Functional Logic of Halt Device Instruction (Cont'd)

Block 9 (Cont'd)

2. The standard device byte is not stored if the condition codes are 0, 2, 3.
3. If an interrupt pending (termination or external device request) condition exists on a specified selector channel, the condition code is set to zero.
4. The channel and device are terminated at the next data service request (see Servicing a Data Transfer).
5. The Channel Address Word (CAW), Channel Block Address (CBA), and Channel Command Word (CCW) are not used by this instruction.
6. If execution of this instruction causes the standard device byte to be stored in the multiplexor channel registers, the program must inhibit interrupts from the multiplexor channel until the standard device byte has been analyzed or moved from the channel registers. If interrupts are permitted and one occurs, the standard device byte is destroyed.

Test Device Instruction

◆ The status of an input/output device can be tested by executing a Test Device instruction. The Test Device instruction is a privileged operation and can be executed only if the mode bit (bit position 15 of the Interrupt Status register for the current state) is set to 0. This instruction is executed in the normal mode. Continuation of program execution is delayed until the instruction is terminated.

Both the channel number and the device number must be specified in the instruction. Because the Channel Address Word is not referred to by the Test Device instruction, the Channel Address Word, Channel Block Address, and a Channel Command Word are not required.

Upon execution of a Test Device instruction, the following events occur (see figure 7).

Block 1

◆ If the privileged mode bit (bit position 15 of the Interrupt Status register) for the current state is not set to 0, the privileged operation bit is set in the Interrupt Flag register and an interrupt occurs, if permitted.

Block 2

◆ If the specified channel is a selector channel that is not available on the system, the condition code is set to 3, the Test Device instruction is terminated and program control is transferred to the next instruction.

Block 3

◆ If the specified channel is a selector channel that is busy or has on interrupt pending (termination or external device request); or if the specified channel is the multiplexor that is operating in the burst mode, the condition code is set to 2, the Test Device instruction is terminated and program control is transferred to the next instruction.

Block 4

◆ The channel status byte and the standard device byte for the specified channel are reset to zeros in the appropriate channel registers.

Block 5

◆ The device address as specified in the Test Device instruction is sent to all trunks on the addressed channel.

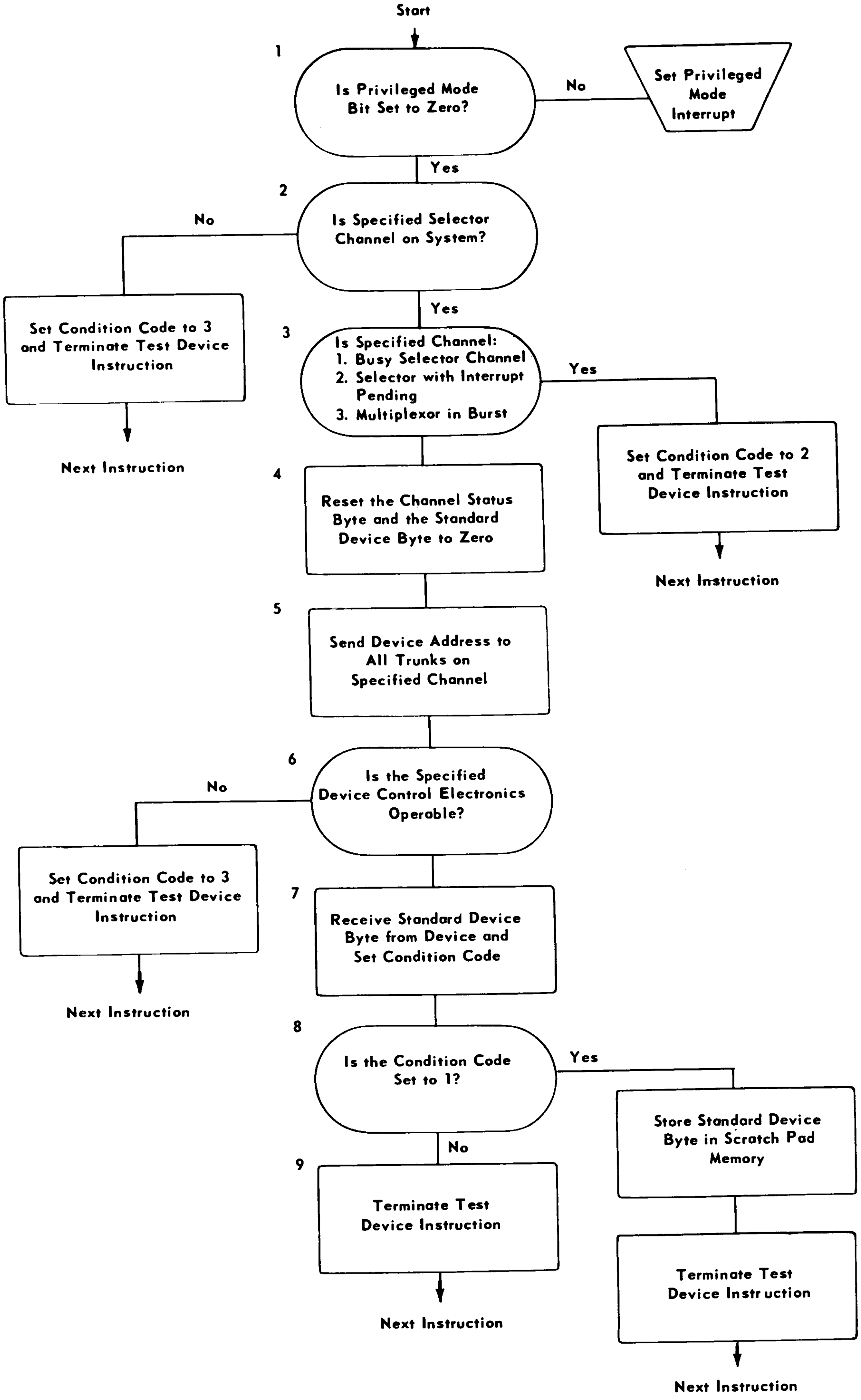

Start
1
Is Privileged Mode Bit Set to Zero?
No
Set Privileged Mode Interrupt
Yes
2
Is Specified Selector Channel on System?
No
Set Condition Code to 3 and Terminate Test Device Instruction
Next Instruction
Yes
3
Is Specified Channel:
1. Busy Selector Channel
2. Selector with Interrupt Pending
3. Multiplexor in Burst
Yes
Set Condition Code to 2 and Terminate Test Device Instruction
Next Instruction
4
Reset the Channel Status Byte and the Standard Device Byte to Zero
5
Send Device Address to All Trunks on Specified Channel
6
Is the Specified Device Control Electronics Operable?
No
Set Condition Code to 3 and Terminate Test Device Instruction
Next Instruction
7
Receive Standard Device Byte from Device and Set Condition Code
8
Is the Condition Code Set to 1?
Yes
Store Standard Device Byte in Scratch Pad Memory
Terminate Test Device Instr uction
Next Instruction
No
9
Terminate Test Device Instruction
Next Instruction

Figure 7. Functional Logic of Test Device Instruction

Block 6 ◆ A test is made to see if the specified device control electronics is operable. The device control electronics has 50 microseconds to signal the processor that it is operable. If it does not, the condition code is set to 3, the Test Device instruction is terminated and program control is transferred to the next instruction.

Block 7 ◆ If the specified device control electronics is operable, it sends the standard device byte to the processor. This standard device byte is *not* stored in the channel registers. It is used to set the proper condition code as follows:

Condition Code	Meaning
3	Device control electronics is inoperable.
2	A termination interrupt pending condition exists in the device control electronics on the multiplexor channel.
2	The device control electronics is busy working with the specified device.
2	The device control electronics is busy working with a device other than the one specified.
1	An external device request interrupt pending condition exists in the device control electronics on the multiplexor channel.
1	The specified device is busy but the device control electronics is not busy (i.e., tape rewinding, off-line seek).
1	The specified device is inoperable.
0	The specified device and control electronics is available.

Block 8 ◆ A test is made to see if the condition code is set to 1. If it is, the standard device byte is transferred to the channel registers for the channel specified, the Test Device instruction is terminated and program control is transferred to the next instruction.

Block 9 ◆ If the condition code is not set to 1, the Test Device instruction is terminated and control is transferred to the next instruction.

Notes on Test Device Instruction:

1. The channel status byte is not stored as a result of this operation.
2. The standard device byte is not stored if the condition codes are 0, 2, or 3.
3. The Channel Address Word (CAW), Channel Block Address (CBA) and Channel Command Word (CCW)) are not used by this instruction.
4. If execution of this instruction causes the standard device byte to be stored in the multiplexor channel registers, the program must inhibit interrupts from the multiplexor channel until the standard device byte has been analyzed or moved from the channel registers. If interrupts are permitted and one occurs, the standard device byte is destroyed.

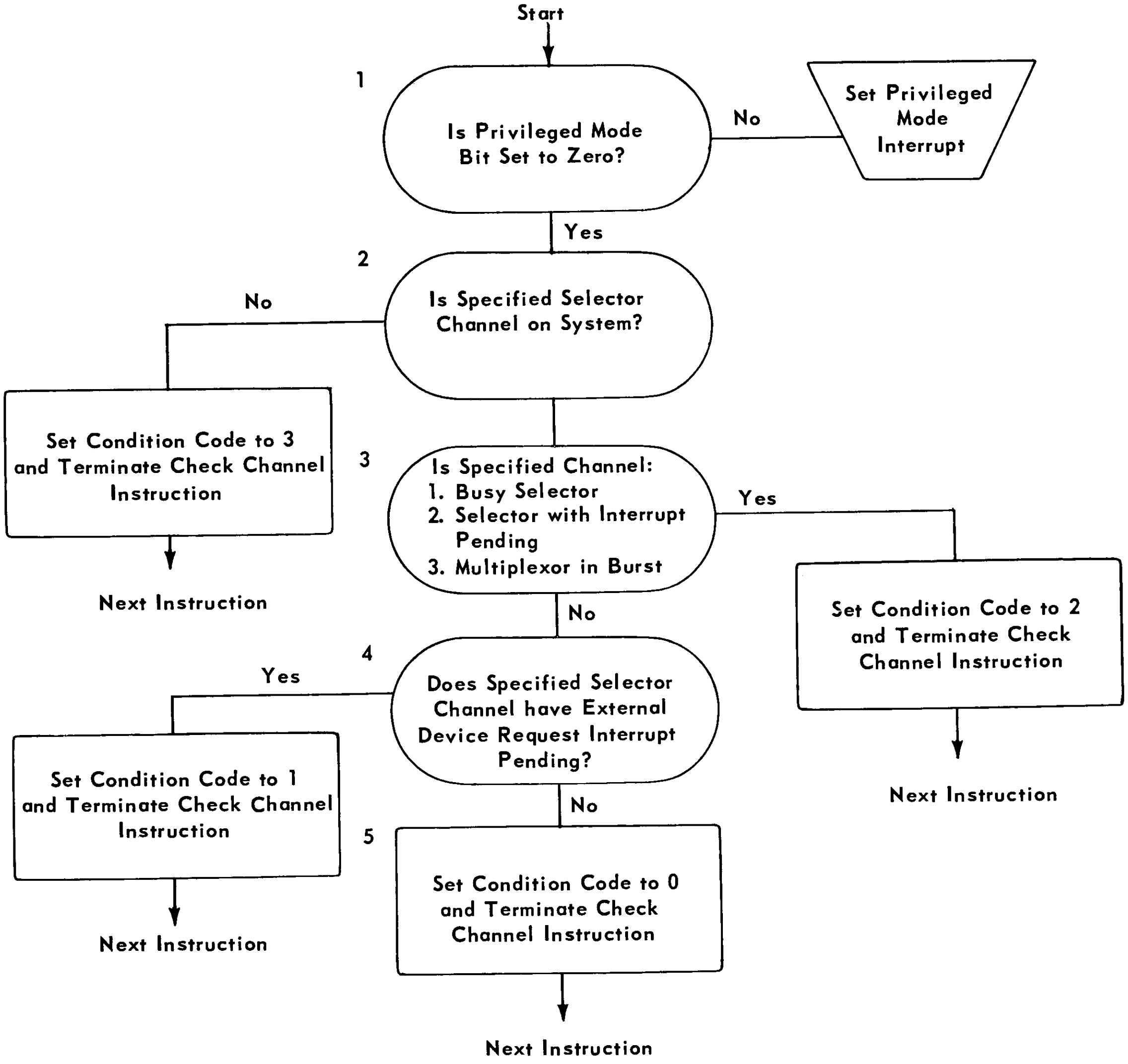

Start
1
Is Privileged Mode Bit Set to Zero?
No
Set Privileged Mode Interrupt
Yes
2
Is Specified Selector Channel on System?
No
Set Condition Code to 3 and Terminate Check Channel Instruction
Next Instruction
3
Is Specified Channel:
1. Busy Selector
2. Selector with Interrupt Pending
3. Multiplexor in Burst
Yes
Set Condition Code to 2 and Terminate Check Channel Instruction
Next Instruction
No
4
Does Specified Selector Channel have External Device Request Interrupt Pending?
Yes
Set Condition Code to 1 and Terminate Check Channel Instruction
Next Instruction
No
5
Set Condition Code to 0 and Terminate Check Channel Instruction
Next Instruction

Figure 8. Functional Logic of Check Channel Instruction

Check Channel Instruction

◆ The status of an input/output channel can be tested by executing a Check Channel instruction. The Check Channel instruction is a privileged operation and can only be executed if the mode bit (bit position 15 of the Interrupt Status register for the current state) is set to 0. This instruction is executed in the normal mode. Continuation of program execution is delayed until the instruction is terminated.

Only the channel number must be specified in the instruction. Because the Channel Address Word is not referred to by the Check Channel instruction, the Channel Address Word, and Channel Block Address (CBA) and a Channel Command Word are not required.

Upon execution of a Check Channel instruction, the following events occur (see figure 8).

Block 1

◆ If the privileged mode bit (bit position 15 of the Interrupt Status register) for the current state is not set to 0, the privileged operation bit is set in the Interrupt Flag register and interrupt occurs if permitted.

Block 2

◆ If the specified channel is a selector channel that is not available on the system, the condition code is set to 3, the Check Channel instruction is terminated and program control is transferred to the next instruction.

Block 3

◆ If the specified channel is a selector channel that is busy or has a termination interrupt pending; or if the specified channel is the multiplexor that is operating in the burst mode, the condition code is set to 2, the Check Channel instruction is terminated and program control is transferred to the next instruction.

Block 4

◆ If the specified channel is a selector channel that has an external device request interrupt pending, the condition code is set to 1, the Check Channel instruction is terminated and program control is transferred to the next instruction.

Block 5

◆ If the specified channel is a selector channel that is not busy and has no interrupts pending; or is the multiplexor channel that is not operating in the burst mode, the condition code is set to 0, the Check Channel instruction is terminated and program control is transferred to the next instruction.

Notes on Check Channel instruction:

1. The channel status byte and the standard device byte are never stored by this instruction.
2. The Channel Address Word (CAW), Channel Block Address (CBA) and the Channel Command Word (CCW) are not used by this instruction.

INPUT/OUTPUT STATUS INDICATORS

◆ Three levels of status information are available to the program to control input/output operation. The first pertains to the setting of the condition code when an input/output instruction is issued. The second level provides more detailed information by storing the channel status byte and the standard device byte in the appropriate input/output channel registers in scratch-pad memory. The third level of status information is generated

INPUT/OUTPUT STATUS INDICATORS (Cont'd)

by, and stored in, the device control electronics until a Sense command is issued. At that time the status information (Sense bytes) are transferred to main memory similar to a data transfer.

Condition Code

◆ The condition code is set by the input/output instructions and can be tested by the Branch On Condition instruction. It should be noted that the condition code settings indicate the result of the input/output instructions only. They do not indicate the results of the input/output operation. Condition Code settings for all input/output instructions are as follows:

Start Device Instruction Condition Code Settings

Condition Code	I/O Operation Initiated	Meaning
0	Yes	1. The device control electronics and the device specified are available. 2. The Start Device instruction specifies a Sense command to a device that is inoperable.
1	No	This condition code indicates that either the channel status byte or the standard device byte has been stored in the channel registers for the specified channel. The channel status byte is stored under the following conditions: 1. A parity error occurs while accessing the Channel Address Word, Channel Block Address, or a Channel Command Word. The channel control check bit in the channel status byte is set. 2. The Memory Protect feature is not installed and the key in the CAW is not zero. The program check bit in the channel status byte is set. 3. The main memory address specified in the CAW is not on a double word boundary. The program check bit in the channel status byte is set. 4. The main memory address in the CCW specifies an address outside the available memory for the system. The program check bit in the channel status byte is set. The standard device byte is stored under the following conditions: 1. The specified device control electronics on the multiplexor channel indicates that a device request interrupt pending condition is present. The external device request interrupt pending bit in the standard device byte is set. 2. The Start Device instruction specifies a command which is other than a Sense command and the addressed device is inoperable. The device inoperable bit in the standard device byte is set. 3. The specified device is busy but the device control electronics is not busy (i.e., tape rewinding, off-line seek to a random access device). The device busy bit and the device end bit in the standard device byte is set.

(Cont'd)

Condition Code
(Cont'd)

Start Device Instruction Condition Code Settings (Cont'd)

Condition Code	I/O Operation Initiated	Meaning
2	No	1. A selector channel is specified that is busy. 2. A selector channel is specified that has an interrupt pending (termination or external device request). 3. The multiplexor channel is specified and it is operating in burst mode. 4. The multiplexor channel is specified and the addressed device control electronics is busy with addressed or non-addressed device. 5. The multiplexor channel is specified and the addressed device control electronics has a termination interrupt pending. 6. A burst mode operation is directed to the multiplexor and there is a termination interrupt pending on one of the attached device control electronics.
3	No	1. A selector channel is specified that is not in the system. 2. The specified device control electronics is inoperable.

Halt Device Instruction Condition Code Settings

Condition Code	I/O Operation Initiated	Meaning
0	No	1. The device control electronics or the device specified on the multiplexor channel is not busy. No termination is required. 2. A selector channel or the multiplexor channel operating in burst mode is specified and it is not busy. No termination is required. 3. The multiplexor channel is specified and the addressed device control electronics has a termination interrupt pending. No termination is required.
1	No	This condition code indicates that the specified device is on the multiplexor channel and that the standard device byte has been stored in the channel registers for the multiplexor channel. The channel status byte is never stored. The standard device byte is stored under the following conditions: 1. The specified device indicates that a device request interrupt pending condition is present. The external device request interrupt pending bit in the standard device byte is set. 2. The specified device is busy but the device control electronics is not busy (i.e., tape rewinding). The device busy bit in the standard device byte is set. 3. The specified device is inoperable. The device inoperable bit in the standard device byte is set.
2	Yes	1. A selector channel is specified that is busy. 2. The multiplexor channel is specified and it is operating in the burst mode. 3. The multiplexor channel is specified and the addressed device control electronics and device are busy.
3	No	1. A selector channel is specified that it is not in the system. 2. The specified device control electronics is inoperable.

Condition Code *(Cont'd)*

Test Device Instruction Condition Code Settings

Condition Code	Meaning
0	The device control electronics and the device are available. *Note:* There may be pending interrupts on the multiplexor channel that would prohibit a burst mode operation being initiated.
1	This condition code indicates that the standard device byte has been stored in the channel registers for the specified channel. The channel status byte is never stored by this instruction. The standard device byte is stored under the following conditions: 1. The specified device control electronics on the multiplexor channel indicates that a device request interrupt pending condition is present. The external device request interrupt pending bit in the standard device byte is set. 2. The specified device is busy but the device control electronics is not busy (i.e., tape rewinding, off-line seek to a random access device). The device busy bit in the standard device byte is set. 3. The specified device is inoperable. The device inoperable bit in the standard device byte is set.
2	1. A selector channel is specified that is busy. 2. A selector channel is specified that has an interrupt pending (termination or external device request). 3. The multiplexor channel is specified and it is operating in burst mode. 4. The multiplexor channel is specified and the addressed device control electronics is busy with addressed or non-addressed device. 5. The multiplexor channel is specified and the addressed device control electronics has a termination interrupt pending.
3	1. A selector channel is specified which is not on the system. 2. The specified device control electronics is inoperable. 3. A device is specified that is not in the system.

Check Channel Instruction Condition Code Setting

Condition Code	Meaning
0	1. The specified selector channel is not busy and has no interrupts pending. 2. The specified multiplexor channel is not operating in the burst mode.
1	The specified selector channel has an external device request interrupt pending.
2	1. The specified selector channel is busy or has a terminating interrupt pending. 2. The specified multiplexor is operating in the burst mode.
3	A selector channel is specified that is not in the system.

Channel Status Byte

◆ The channel status byte is stored in Channel Command Register-II (bit positions 8 through 15) for the appropriate channel. It contains information concerning the status of the channel when a channel interrupt occurs, or at the completion of a Start, Halt or Test Device instruction if

Channel Status Byte *(Cont'd)*

the condition code indicates that Status is stored. The bit significance of the channel status byte is as follows:

Bit Position 8 is the program controlled interrupt bit. When set, this bit indicates that a Channel Command Word was accessed which had the program controlled interrupt flag bit set. A channel interrupt occurs for the appropriate channel while the input/output operation specified by the Channel Command Word is being executed.

Note: The program controlled channel interrupt occurs after the first data byte has been transferred.

Bit Position 9 is the incorrect length bit. When set, this bit indicates that when the input/output operation was terminated, the byte count specified in the channel command was not equal to the number of bytes received from, or sent to, the input/output device. The incorrect length indicator can be set only if the suppress length indicator flag bit in the channel command word is reset to 0.

The following conditions cause the incorrect length bit to be set:

1. Count High on Input (Read, Read Reverse, Sense). The main memory area specified by the Channel Command Word is not completely filled by transmission from the device. The final byte count in Channel Command Register-II is greater than zero.
2. Count High on Output (Write, Write Control). Data in the main memory area specified by the Channel Command Word is not completely transferred and the device terminated. The final byte count in Channel Command Register-II is greater than zero.

Notes:

1. If incorrect length occurs during command chaining and the Suppress Length Indicator flag bit of the current command is reset, the incorrect length bit is set.
2. If incorrect length occurs during the last command of a chain (the Chain Data flag bit is reset), and the Suppress Length Indicator flag of the command is set, the incorrect length bit is not set.

Bit Position 10 is the program check bit. When set, this bit indicates that a programming error was detected by the channel.

The following conditions cause the program check bit to be set:

1. Invalid Channel Command Word Address. The addressed Channel Command Word is not located on a double word boundary.
2. Invalid Channel Command Word Address. The addressed Channel Command Word is outside the available main memory for the particular installation.
3. Invalid Data Address. The main memory location specified by the data address in the Channel Command Word is outside the available main memory for the particular installation.

Channel Status Byte *(Cont'd)*

4. Invalid Key. The memory protection key in the Channel Address Word is not zero and the system does not have the Memory Protect option installed.

Notes:

1. If a program check error occurs during input/output initiation, the operation is suppressed and the program is notified of the error by the condition error setting.
2. If a program check error occurs while the input/output operation is in progress, the operation is terminated and a channel interrupt occurs for the specified channel.
3. If a program check error occurs during chaining (command or data), a channel interrupt occurs for the specified channel and chaining is suppressed.

Bit Position 11 is the protection check bit. When set, this bit indicates that the channel tried to store data in a protected main memory area. The operation is terminated and a channel interrupt occurs for the specified channel. If chaining (command or data) is in progress, it is suppressed.

Bit Position 12 is the channel data check bit. When set, this bit indicates that a parity error was detected in the channel, in main memory, non-addressable main memory or in scratch-pad memory. Characters with bad parity going into memory are replaced with the systems error byte (hexadecimal FF), and the input/output operation is completed. For parity error characters going to a device, (writing) the invalid character is transferred unchanged, the operation is terminated and a channel interrupt occurs for the specified channel. (The transfer of sense byte(s) to memory is not checked for parity.)

Bit Position 13 is the channel control check bit. When set, this bit indicates that a machine malfunction has occurred affecting the channel controls. Conditions which cause this bit to be set are parity error in the Channel Command Word, data address, or contents of the Channel Command Word. The operation is terminated and a channel interrupt occurs for the specified channel. If chaining (command or data) is in progress, it is suppressed.

Bit Position 14 is reserved for use by the processor.

Bit Position 15 is the termination interrupt bit. When set, this bit indicates that a termination interrupt has been effected.

Important: The channel status byte is reset *only* when an input/output operation is initiated.

Standard Device Byte

◆ The standard device byte is stored in scratch-pad memory in the Assembly/Status register (bit positions 24 through 31) for the appropriate channel. This byte indicates the status of a device after an input/output operation. It may also indicate a device request interrupt.

The standard device byte is automatically stored when:

1. An input/output interrupt is serviced (request or termination).

Standard Device Byte *(Cont'd)*

2. An input/output operation is attempted and the condition code indicates that status bits are stored (channel status byte, standard device byte).

The standard device byte is defined as follows:

Bit Position 24 is the external device request interrupt pending bit. When set, this bit indicates that a random access device, a data exchange control or a communications device requires servicing.

Bit Position 25 is the termination interrupt pending bit. When set, this bit indicates that a termination interrupt condition exists in an input/output device.

Bit Position 26 is the device busy bit. When set, this bit indicates that the specified device is busy and cannot accept another operation.

Bit Position 27 is the control busy bit. Not applicable.

Bit Position 28 is the device end bit. When set, this bit indicates that the specified device has terminated. Another operation can be accepted by the device if the device busy bit (26) is not set.

Bit Position 29 is the secondary indicator bit. When set, this bit indicates that the specified device has additional indicators to be tested. These indicators can be brought into main memory by using the Sense command.

Bit Position 30 is the device inoperable bit. When set, this bit indicates that the specified device is inoperable.

Bit Position 31 is the status modifier bit. This bit is used with Command chaining. When set, this bit indicates that the next Channel Command Word is skipped. This bit is set as a result of device termination.

Sense Bytes

◆ The sense byte, or bytes, are brought into main memory from an input/output device by using the Sense command. These bytes contain status information for the device referred to. The exact status information sent is defined in the Spectra 70 input/output reference manuals for the individual devices.

CHANNEL SERVICING

◆ The following sections explain in detail the three types of channel servicing which may be performed during input/output operations. They are: servicing a data transfer, end and chain servicing, and interrupt servicing.

Because channel servicing requires that the channel utilize main memory, scratch-pad memory and non-addressable main memory (multiplexor devices), normal mode processing is held off until the servicing has been completed. Consequently, channel servicing is time-shared with normal mode processing. Between service requests, normal mode processing is resumed, or another channel is permitted to service its device(s).

Channel servicing for a device on the multiplexor channel (multiplex mode) requires more time than channel servicing for a device on a selector channel. To balance the system's throughput rate, multiplexor channel servicing is segmented to permit selector channel servicing to break-in if any selector channels require servicing. After all selector(s) demanding service have been satisfied, multiplexor servicing is resumed. This tech-

CHANNEL SERVICING (Cont'd)

nique insures that the interference to selector channel servicing caused by the multiplexor channel is not greater than that of an additional selector channel.

Servicing a Data Transfer

◆ Once an input/output operation has been initiated, it proceeds under control of the appropriate channel and registers in scratch-pad memory and non-addressable main memory (multiplexor devices). When an input/output operation has been initiated and the input/output device is ready to send or receive a data byte, it asks the processor for a service request. When the processor honors this service request, servicing of a data transfer occurs.

Because servicing a data transfer requires that the channel utilize main memory, scratch-pad memory and non-addressable main memory (multiplexor devices), normal mode processing is held off until the servicing has been completed. Servicing of a data transfer is time-shared with normal mode processing. Between service requests, processing is resumed, or another channel is permitted to service its device(s).

If a burst mode operation has been initiated to the multiplexor channel, the channel operates similar to a selector and only one device is serviced. Service requests by devices other than the one operating in burst mode are ignored until the multiplexor channel is operating in the multiplexor mode. This occurs at the conclusion of the burst operation when the last data byte has been serviced (prior to interrupt).

Servicing of a data transfer causes the following events to occur (see figure 9).

Block 1

◆ If the service request comes from a device control electronics connected to the multiplexor channel which is operating in the multiplex mode, the processor gets the device address and fetches the appropriate subchannel registers in non-addressable main memory. These registers are placed in processor utility registers in scratch-pad memory. (They are *not* sent to the multiplexor channel registers in scratch-pad memory.) If the service request comes from a device control electronics connected to the multiplexor channel which is operating in the burst mode or from a device connected to a selector channel, the appropriate channel registers in scratch-pad memory are used to service the data transfer.

Block 2

◆ A test is made to. see if the Program Controlled Interrupt (PCI) flag is set. If it is, the channel interrupt bit is set in the Interrupt Flag register and an interrupt occurs, if permitted. The PCI flag is reset and the program control interrupt bit is set in the channel status byte.

Block 3

◆ A test is made to see if the device control electronics requesting service has indicated an end condition. An end condition is indicated when one of the following occurs:

1. The processor reaches a byte count lapse. If this occurs, the processor tells the device control electronics to indicate an end condition on the next data service request.

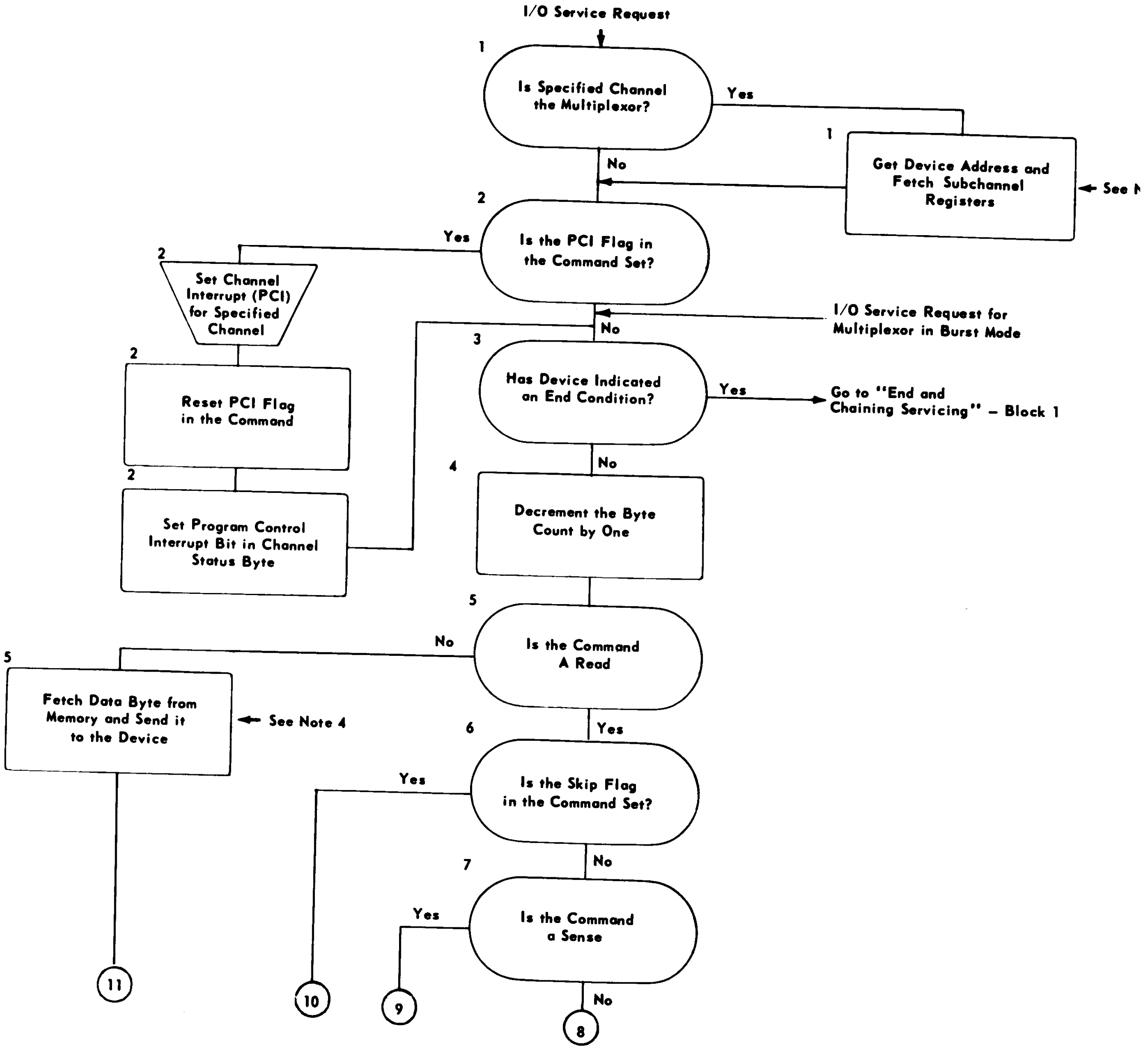

I/O Service Request
1
Is Specified Channel the Multiplexor?
Yes
1
Get Device Address and Fetch Subchannel Registers
See N
No
2
Is the PCI Flag in the Command Set?
Yes
2
Set Channel Interrupt (PCI) for Specified Channel
2
Reset PCI Flag in the Command
2
Set Program Control Interrupt Bit in Channel Status Byte
I/O Service Request for Multiplexor in Burst Mode
No
3
Has Device Indicated an End Condition?
Yes
Go to "End and Chaining Servicing" - Block 1
No
4
Decrement the Byte Count by One
5
Is the Command A Read
No
5
Fetch Data Byte from Memory and Send it to the Device
See Note 4
Yes
6
Is the Skip Flag in the Command Set?
Yes
No
7
Is the Command a Sense
Yes
No
11
10
9
8

Figure 9. Functional Logic of Servicing a Data Transfer

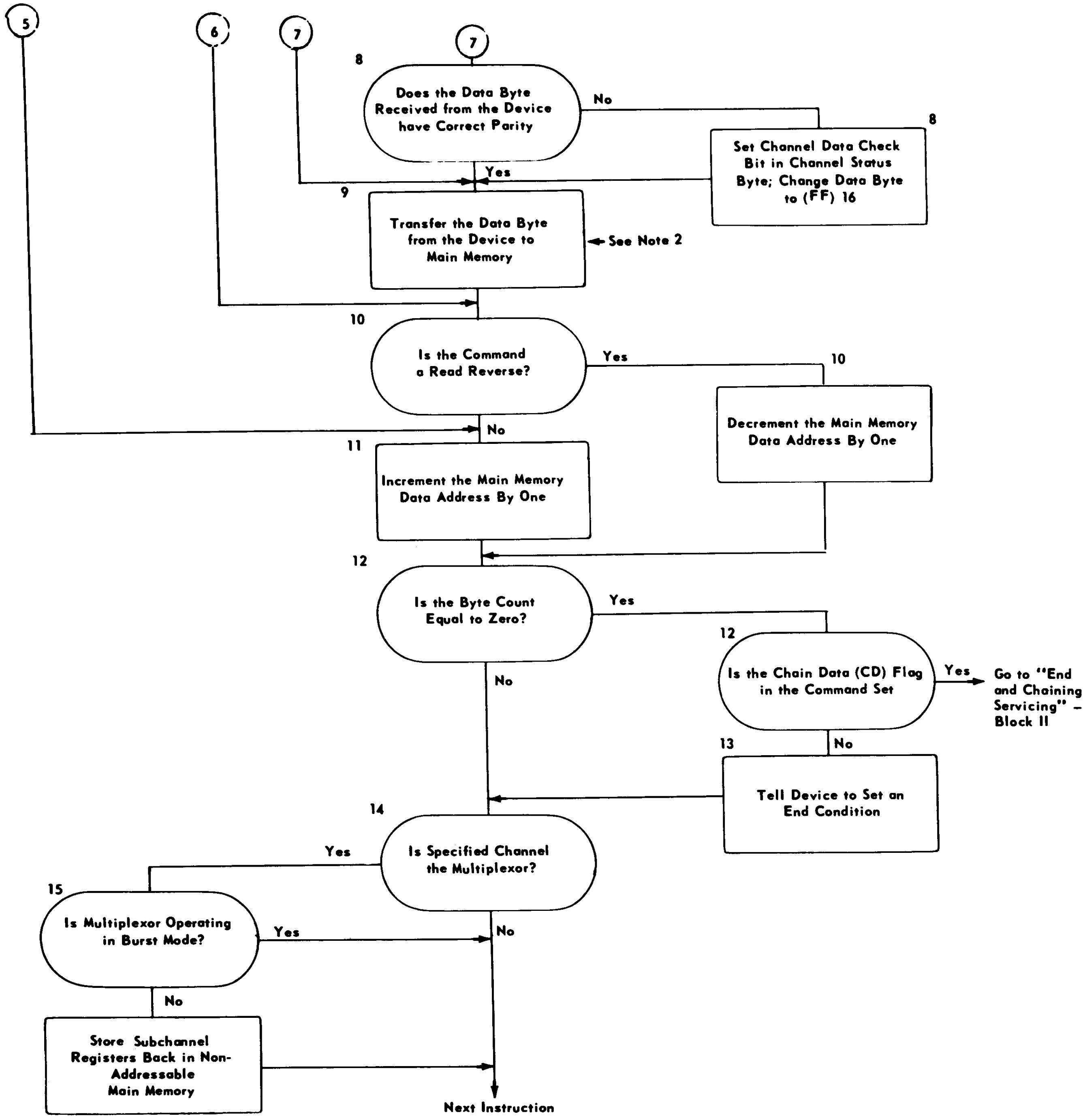

5
6
7
7
8
Does the Data Byte Received from the Device have Correct Parity
No
8
Set Channel Data Check Bit in Channel Status Byte; Change Data Byte to (FF) 16
Yes
9
Transfer the Data Byte from the Device to Main Memory
See Note 2
10
Is the Command a Read Reverse?
Yes
10
Decrement the Main Memory Data Address By One
No
11
Increment the Main Memory Data Address By One
12
Is the Byte Count Equal to Zero?
Yes
12
Is the Chain Data (CD) Flag in the Command Set
Yes
Go to "End and Chaining Servicing" – Block II
No
No
13
Tell Device to Set an End Condition
14
Is Specified Channel the Multiplexor?
Yes
15
Is Multiplexor Operating in Burst Mode?
Yes
No
No
Store Subchannel Registers Back in Non-Addressable Main Memory
Next Instruction

Figure 9. Functional Logic of Servicing a Data Transfer (Cont'd)

Servicing a Data Transfer *(Cont'd)*

2. The device has completed the input/output operation (i.e., a gap is sensed on tape). If this occurs, the device control electronics automatically indicates an end condition. (In this case the byte count is never zero.)

If an end condition has been indicated, the processor goes to End and Chaining Servicing (see figure 10, Block 1).

Note: Certain error conditions cause the processor to tell the device control electronics to indicate an end condition on the next data service request (see Notes 3, 4, 5, 6 on Servicing a Data Transfer).

Block 4 ◆ If the device control electronics has not indicated an end condition, the byte count is decremented by one.

Block 5 ◆ A test is made to see if the command is a read. A read command can be any one of the following:

Read Forward

Read Reverse

Sense

All other commands (except Transfer in Channel) are write commands. If the command is a write, the data byte is fetched from main memory and sent to the device. Control is then transferred to Block 11.

Block 6 ◆ If the command is a read, a test is made to see if the SKIP flag is set. If it is, transfer of the data byte to main memory is bypassed and control is transferred to Block 10.

Block 7 ◆ If the SKIP flag is not set, a test is made to see if the command is a Sense. If it is, parity checking of the data byte is bypassed and control is transferred to Block 9.

Block 8 ◆ If the command is not a Sense, a test is made to see if the data byte received from the device has correct parity. If it does not, the channel data check bit in the channel status byte is set and the data byte is converted to $(FF)_{16}$. The input/output operation continues.

Block 9 ◆ The data byte is transferred to the main memory address specified.

Block 10 ◆ A test is made to see if the command is a Read Reverse. If it is, the main memory address is decremented by one.

Block 11 ◆ If the command is not a Read Reverse, the main memory address is incremented by one.

Block 12 ◆ A test is made to see if the byte count has lapsed. If it has, a test is made to see if the Chain Data flag is set. If it is, the processor goes to End and Chaining Servicing (see figure 10, Block 11).

Block 13 ◆ If the Chain Data flag is not set, the processor tells the device control electronics to indicate an end condition on the next data service request.

Block 14 ◆ A test is made to see if the service request was honored for a device on the multiplexor channel. If it was not, program control continues with the next instruction or with the instruction that was interrupted due to the service request.

Block 15 ◆ If the service request was honored for a device on the multiplexor channel, a test is made to see if it is a burst mode operation. If it is not a burst mode operation, the sub-channel registers are sent back to non-addressable main memory. In either case, program control continues with the next instruction or with the instruction that was interrupted due to the service request.

Notes on Servicing a Data Transfer:

1. All input/output data service requests are honored depending on the channel's position in the priority sequence.
2. The following tests occur when a data byte is transferred to main memory:
   a. The main memory address to which the data byte is to be transferred is tested to see if it is in a memory protected area (Memory Protect feature must be installed). If it is, the protection check bit in the channel status byte is set (no data transfer occurs) and the device control electronics is told to set an end condition on the next data service request (see Block 13).

   b. The main memory address to which the data byte is to be transferred is tested to see if it is in available main memory for the system. If it is not, the program check bit in the channel status byte is set (no data transfer occurs) and the device control electronics is told to set an end condition on the next data service request (see Block 13).
3. The following tests occur when a data byte is transferred from main memory:
   a. The main memory address from which the data byte is to be transferred is tested to see if it is in available main memory for the system. If it is not, the program check bit in the channel status byte is set (no data transfer occurs) and the device control electronics is told to set an end condition on the next data service request (see Block 13).

   b. The data byte to be transferred is checked for correct parity. If parity is not correct, the data check bit in the channel status byte is set and the device control electronics is told to set an end condition on the next data service request (see Block 13).
4. If a main memory parity error occurs while fetching the subchannel registers, the channel control check bit in the channel status byte is set, and the device control electronics is told to set an end condition on the next data service request (see Block 13).
5. If a scratch-pad memory parity error occurs during the servicing of a data transfer, the channel control check bit in the channel status byte is set and the device control electronics is told to set an end condition on the next data service request (see Block 13).

End and Chaining Servicing

◆ End and chaining servicing is required when the input/output operation specified by the current command has been completed (normally or abnormally). Entry to this servicing always comes from "servicing a data transfer". The following conditions cause end and chaining servicing to take place:

1. A device control electronics has indicated an end condition. This end condition is recognized in Servicing a Data Transfer.
2. The byte count in the current command has lapsed and the Chain Data (CD) flag in this command is set. If this condition occurs, entry to End and Chaining Servicing occurs at a point which bypasses the normal end servicing with no chaining and the end servicing with command chaining.

For input/output operations that do not specify chaining, end servicing is used so that the processor can tell the appropriate device control electronics to set an interrupt condition. This interrupt condition is in turn reported to the processor and the appropriate flag in the Interrupt Flag register is set, at which time the interrupt is taken, if permitted.

For input/output operations that specify chaining (command or data), this servicing does one of the following:

1. If the current command specifies command chaining (the CC flag in the command is set) this service is used to fetch the next command in the chain and to send this new command to the input/output device.
2. If the current command specifies data chaining (the CD flag in the command is set) this service is used to fetch the next command in the chain so that the current operation can be continued.

End and Chaining Servicing causes the following events to occur (see figure 10).

Block 1

◆ Entry to this block occurs when the input/output device control electronics has indicated an end condition. This end condition is recognized during servicing a data transfer and is generated:

1. automatically by the device, or
2. by the device on command from the processor

The processor receives the standard device byte from the device control electronics. This standard device byte is used by the processor for testing purposes. It is *not* stored in the channel registers.

Block 2

◆ The standard device byte is tested to see if the device busy bit is set and the device end bit is reset. This condition normally arises in buffered devices (i.e., card punch, printer) when the buffer has been loaded and the completion of the operation is off-line (no more data has to be sent between the processor and the device control electronics). If this condition exists, the processor tells the device to set another end condition and ask for another service request when the device is no longer busy. Control is then transfrred to Block 14.

Block 3 ◆ If the device is not busy, a test is made to see if the Chain Command (CC) flag is set. If it is not, control is transferred to Block 8 which causes termination of the command to occur.

Block 4 ◆ If the Chain Command (CC) flag is set, a test is made to see if one of the following bits is set in the channel status byte:

Program Check bit

Protection Check bit

Data Check bit (This bit is checked only if the current operation is a write)

Channel Control Check bit

If any of the above bits are set (except the data check bit on a Read) control is transferred to Block 8 which causes termination of the command and suppression of command chaining to occur.

Block 5 ◆ If none of the bits tested in the channel byte are set, a test is made to see if the Chain Data (CD) flag is set. If the Chain Data flag is set, control is transferred to Block 8 which causes termination of the command and suppression of command chaining to occur.

Block 6 ◆ If the Chain Data (CD) flag is not set the standard device byte is tested to see that the following conditions are present:

Device is operable

Secondary indicator is not set

Device end is set

If any of the above conditions is not present, control is transferred to Block 8 which causes termination of the command and suppression of command chaining to occur.

Block 7 ◆ If all of the conditions tested in the standard device byte are present, a test is made to see if the byte count is *not* equal to zero and the Suppress Length Indicator (SLI) flag is equal to zero. If these conditions are present, the program desires an indication of incorrect length, and control is transferred to Block 8 which causes termination of the command and suppression of command chaining to occur.

Block 8 ◆ Entry to this block occurs under the following conditions:

a. A device control electronics has indicated and end condition, the device is not busy and the chain command flag bit is *not* set.

b. A device control electronics has indicated an end condition and the chain command flag is set. However, a condition is present which causes command chaining to be suppressed.

The processor tells the device control electronics to set a channel interrupt condition for the appropriate channel.

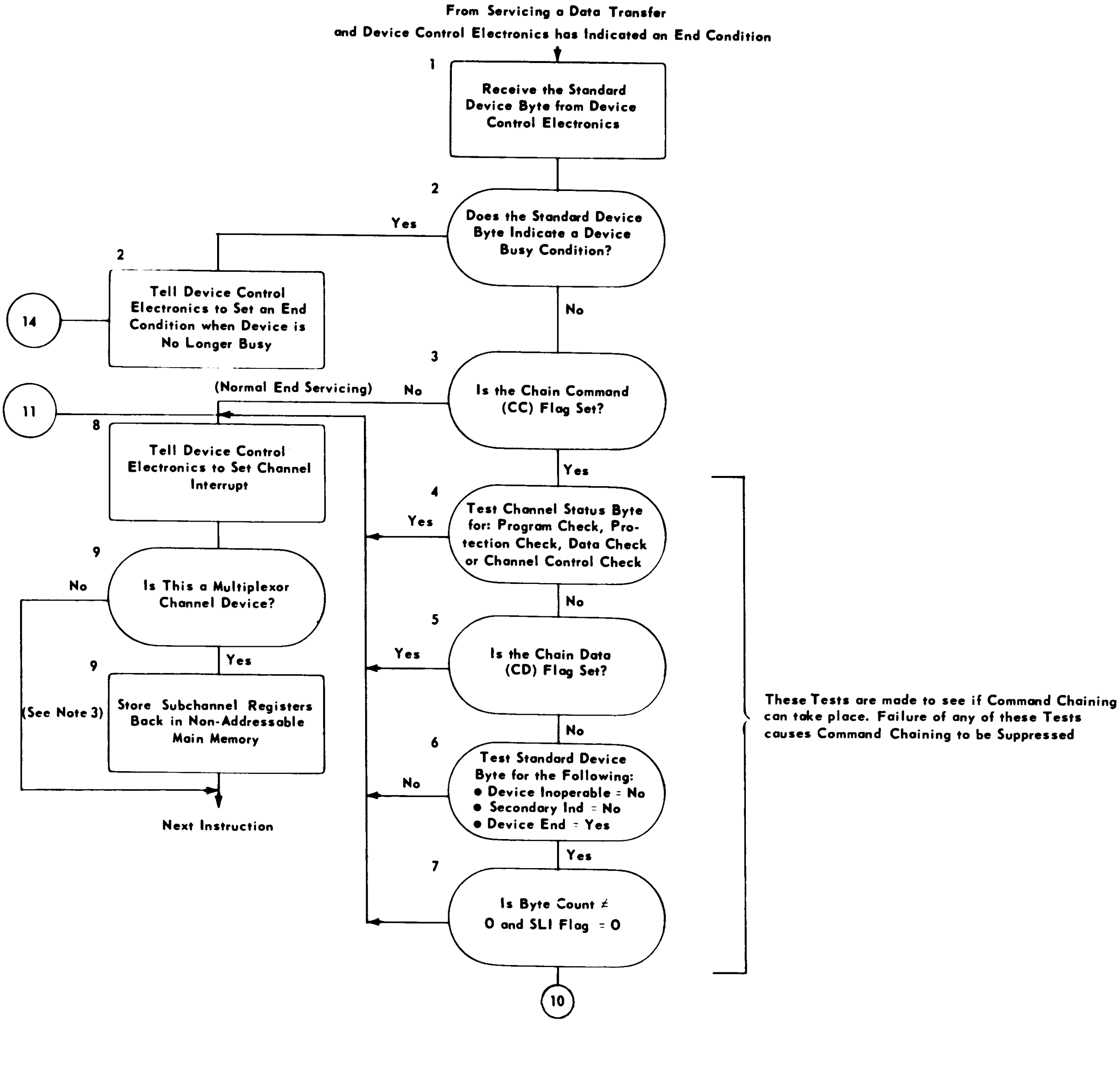

From Servicing a Data Transfer
and Device Control Electronics has Indicated an End Condition
1
Receive the Standard Device Byte from Device Control Electronics
2
Does the Standard Device Byte Indicate a Device Busy Condition?
Yes
2
Tell Device Control Electronics to Set an End Condition when Device is No Longer Busy
14
No
3
Is the Chain Command (CC) Flag Set?
(Normal End Servicing)
No
11
8
Tell Device Control Electronics to Set Channel Interrupt
Yes
4
Test Channel Status Byte for: Program Check, Protection Check, Data Check or Channel Control Check
Yes
9
Is This a Multiplexor Channel Device?
No
No
5
Is the Chain Data (CD) Flag Set?
Yes
Yes
9
Store Subchannel Registers Back in Non-Addressable Main Memory
(See Note 3)
No
6
Test Standard Device Byte for the Following:
• Device Inoperable = No
• Secondary Ind = No
• Device End = Yes
No
Next Instruction
Yes
7
Is Byte Count ≠ 0 and SLI Flag = 0
10
These Tests are made to see if Command Chaining can take place. Failure of any of these Tests causes Command Chaining to be Suppressed

Figure 10. Functional Logic of End and Chaining Servicing

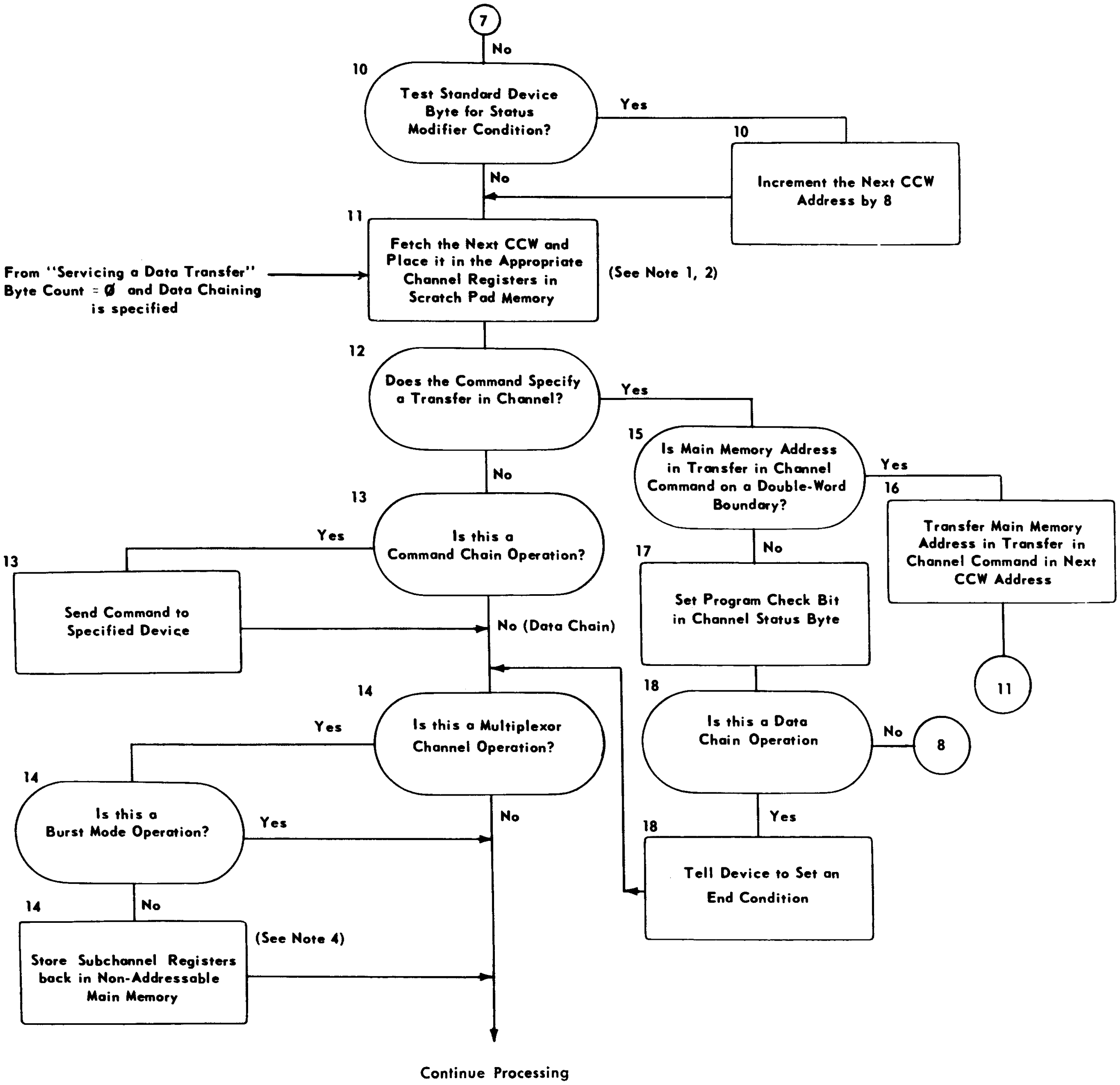

7
No
10
Test Standard Device Byte for Status Modifier Condition?
Yes
10
Increment the Next CCW Address by 8
No
11
From "Servicing a Data Transfer" Byte Count = Ø and Data Chaining is specified
Fetch the Next CCW and Place it in the Appropriate Channel Registers in Scratch Pad Memory
(See Note 1, 2)
12
Does the Command Specify a Transfer in Channel?
Yes
15
Is Main Memory Address in Transfer in Channel Command on a Double-Word Boundary?
Yes
16
Transfer Main Memory Address in Transfer in Channel Command in Next CCW Address
11
No
13
Is this a Command Chain Operation?
Yes
13
Send Command to Specified Device
No (Data Chain)
No
17
Set Program Check Bit in Channel Status Byte
18
Is this a Data Chain Operation
No
8
Yes
18
Tell Device to Set an End Condition
14
Is this a Multiplexor Channel Operation?
Yes
14
Is this a Burst Mode Operation?
Yes
No
14
No
Store Subchannel Registers back in Non-Addressable Main Memory
(See Note 4)
Continue Processing

Figure 10. Functional Logic of End and Chaining Servicing (Cont'd)

Block 9 ◆ A test is made to see if the device is on the multiplexor channel. If it is, the subchannel registers are sent back to non-addressable main memory. In either case, program control continues with the next instruction or with the instruction that was interrupted due to chaining and/or end servicing.

Note: If the operation that was terminated was a burst mode operation, the burst mode is completed at this point and other multiplex mode operations can be directed to devices on the multiplexor channel. The processor does *not* have to wait for the burst mode terminating interrupt to occur.

Block 10 ◆ Entry to this block occurs when command chaining is to take place. The standard device byte is tested to see if the status modifier bit is set. If it is, the next Channel Command Word (CCW) address is incremented by eight. (The next channel command word in sequence is skipped.)

Block 11 ◆ In addition to continuing command chaining processing, entry to this block occurs from Servicing a Data Transfer when the following conditions are present:

a. The byte count is equal to zero.

b. The Chain Data (CD) flag is set.

The next Channel Command Word (CCW) is fetched from main memory and placed in the appropriate channel registers. The next Channel Command Word address is incremented by eight.

Block 12 ◆ A test is made to see if the next command in sequence is a Transfer in Channel command.

Block 13 ◆ If the command is not a Transfer in Channel command, a test is made to see if this is a command chain or a data chain operation. If it is a command chain operation, the new command is sent to the specified device control electronics. (This is not required if this is a data chain operation.)

Block 14 ◆ A test is made to see if the chaining servicing has occurred for a device on the multiplexor channel. If it has, a test is made to see if it is a burst mode operation. If it is not a burst mode operation, the subchannel registers are sent back to non-addressable main memory. In all cases, program control continues with the next instruction, or with the instruction that was interrupted due to the chaining servicing.

Block 15 ◆ If the next command in sequence is a Transfer in Channel command, the main memory address specified by the Transfer in Channel command is tested to see if it is on a double word boundary.

Block 16 ◆ If the main memory address specified in the Transfer in Channel command is on a double word boundary, this address is placed in the next Channel Command Word address and control is transferred to Block 11 which fetches the CCW specified by the Transfer in Channel command.

Block 17 ◆ If the main memory address specified in the Transfer in Channel command is *not* on a double word boundary, the program check bit is set in the channel status byte.

Block 18

◆ A test is made to see if this is a data chain operation. If it is, the device is told to set an end condition on the next data service request and control is transferred to Block 14 to complete the end servicing. If this is a command chain operation (the device has already indicated an end condition) control is transferred to Block 8 where the device control electronics is told to set an interrupt condition.

Notes On End and Chaining Servicing:

1. The following test occurs when the next Channel Command Word is fetched:

   The main memory address specified is tested to see if it is in available main memory for the system. If it is not, the program check bit in the channel status byte is set; and, if data chaining, the device is told to set an end condition on the next data service request (see Block 2); if command chaining, the device control electronics is told to set a channel interrupt condition (see Block 8).

2. If a main memory parity error occurs when fetching the next Channel Command Word, the channel control check bit in the channel status byte is set; and, if data chaining, the device control electronics is told to set an end condition on the next data service request (see Block 2); if command chaining, the device control electronics is told to set a channel interrupt condition (see Block 8).

3. If a scratch-pad memory parity error occurs when storing the subchannel registers back in non-addressable main memory the channel control check bit in the channel status byte is set.

4. If a scratch-pad memory parity error occurs when storing the subchannel registers back in non-addressable main memory, the channel control check bit in the channel status byte is set; and, if data chaining, the device control electronics is told to set an end condition on the next service request (see Block 2); if command chaining, the device control electronics is told to set a channel interrupt condition (see Block 8).

Interrupt Servicing

◆ Interrupt servicing occurs when the appropriate flag in the Interrupt Flag register has been set, and the Interrupt Mask register for the current state permits the interrupt and it is taken. This service is required to:

1. Obtain the standard device byte from the device control electronics (if applicable) and store it in the appropriate input/output channel registers.

2. Fetch the appropriate subchannel registers from non-addressable main memory if the interrupt is due to a multiplexor channel device. The subchannel registers are stored in the multiplexor channel registers in scratch-pad memory.

There are three kinds of channel interrupts. They are as follows:

Programmed Control Interrupt—This interrupt occurs when a Channel Command Word is fetched and the program controlled interrupt flag bit is

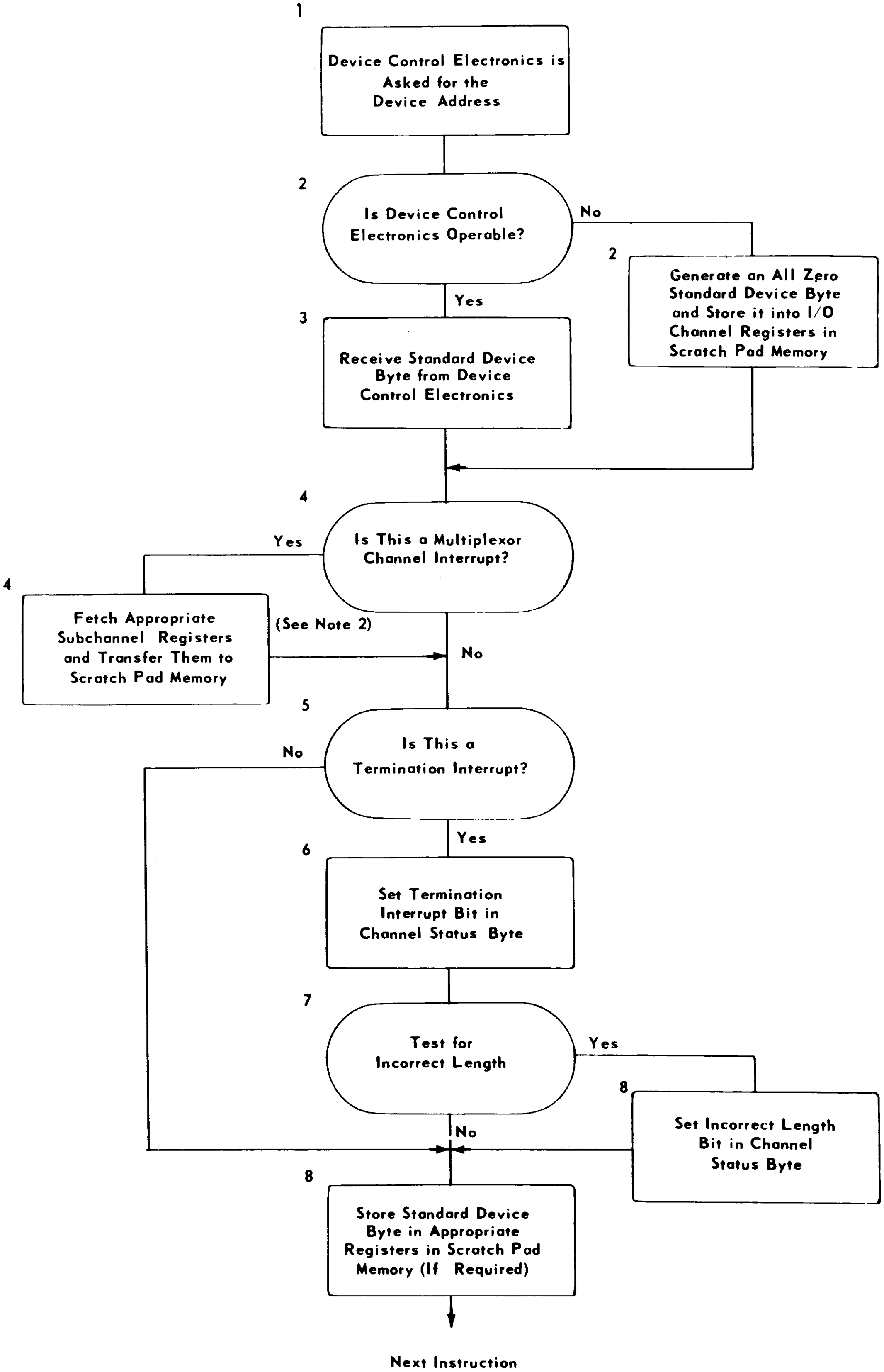

1
Device Control Electronics is Asked for the Device Address
2
Is Device Control Electronics Operable?
No
2
Generate an All Zero Standard Device Byte and Store it into I/O Channel Registers in Scratch Pad Memory
Yes
3
Receive Standard Device Byte from Device Control Electronics
4
Is This a Multiplexor Channel Interrupt?
Yes
4
Fetch Appropriate Subchannel Registers and Transfer Them to Scratch Pad Memory
(See Note 2)
No
5
Is This a Termination Interrupt?
No
Yes
6
Set Termination Interrupt Bit in Channel Status Byte
7
Test for Incorrect Length
Yes
8
Set Incorrect Length Bit in Channel Status Byte
No
8
Store Standard Device Byte in Appropriate Registers in Scratch Pad Memory (If Required)
Next Instruction

Figure 11. Functional Logic of Interrupt Servicing

Interrupt Servicing *(Cont'd)*

set. This interrupt condition has no effect upon the input/output operation specified by the Channel Command Word. The standard device byte and the subchannel registers are not stored.

Device Request Interrupt—This interrupt occurs as a result of a condition arising in an input/output device control electronics. It may occur independent of a processor initiated input/output operation. Examples of this type of interrupt are as follows:

1. A remote processor wishes to send data via a Data Exchange Control. The Data Exchange Control initiates the channel interrupt. (This interrupt occurs independent of a procesor initiated input/output operation.)
2. The processor initiates an off-line seek to a random access device. When the seek is complete, the random access device control electronics initiates a channel interrupt. (This interrupt occurs in conjunction with a processor initiated input/output operation.)

When an external device request interrupt occurs, the standard device byte and the subchannel registers (if a multiplexor device) are stored in the appropriate input/output channel registers.

Terminating Interrupt—This interrupt occurs when an input/output operation initiated by the processor has terminated. When this interrupt occurs, the standard device byte and the subchannel registers (if a multiplexor device) are stored in the appropriate input/output channel registers. This is the final servicing of the channel and device. At the completion of this servicing, the channel is free to accept another operation. The contents of the input/output channel registers must be utilized by the program before another operation is initiated. (When another operation is initiated, the contents of these registers are altered.) The following information is available in the input/output channel registers for interrogation by the program:

Channel status byte

Standard device byte

Byte count

Address of next CCW

Low-order 4 bits of the command code

Device number

Interrupt servicing causes the following events to occur (see figure 11).

Block 1

◆ The device control electronics is asked for the address of the device requiring interrupt servicing.

Block 2

◆ A test is made to see if the device control electronics is operable. The device control electronics has 50 microseconds to signal the processor that it is operable. If it does not, the processor generates a standard device byte of all zeros. Control is then transferred to Block 4.

Block 3

◆ If the device control electronics is operable, it sends the standard device byte to the processor.

Block 4 ◆ If the service request comes from a device control electronics connected to the multiplexor channel, the processor uses the device address to fetch the appropriate subchannel registers in non-addressable main memory. The subchannel registers are stored in the input/output channel registers in scratch-pad memory for the multiplexor channel.

Block 5 ◆ A test is made to see if this is a terminating interrupt. If it is not (it is a program controlled or a device request interrupt) control is transferred to Block 8.

Block 6 ◆ If the interrupt is a terminating interrupt, the termination interrupt bit in the channel status byte is set.

Block 7 ◆ A test is made to see if the byte count is *not* equal to zero and the Suppress Length Indicator (SLI) flag is equal to zero. If these conditions are present, the program desires an indication of incorrect length and the incorrect length bit in the channel status byte is set.

Block 8 ◆ The standard device byte is stored in the appropriate input/output channel registers and program control continues with the next instruction.

Notes on Interrupt Servicing:

1. The device address is always stored in the input/output channel registers in scratch-pad memory if the interrupt is due to a device connected to the multiplexor channel. If the interrupt is due to a device on a selector channel, the device address is stored *only* if it is a device request interrupt.
2. If a main memory parity error occurs when fetching the subchannel registers, the channel control check bit in the channel status byte is set.

MULTI-PROCESSOR INSTALLATION

INTRODUCTION

◆ Installations where more than one computer shares peripheral equipment or work loads require extra machine-program communications. To enable this rapid signaling between processors independent of input/output operations the Direct Control feature is provided.

To signal a receiving processor (or processors) a Write Direct instruction is used to effect an external interrupt in the receiving processor. To enable the receiving processor to honor this external interrupt and complete the transfer, a Read Direct instruction is used (refer to Privileged Instructions section). This Write Direct action of one processor to another is analogous to a Supervisor Call instruction and corresponding interrupt of a user's program to the Interrupt Control State (P_3).

Some typical cases for which this feature is used are:

Request use of a control file.

Notify that file access has been completed.

Notify back-up system that a processor machine failure has been detected.

Notify back-up system that a processor power failure has been detected.

Request assistance because of program overload.

Request for task assignments.

OPERATIONAL CHARACTERISTICS

◆ The 8-bit data byte transmitted from the out line of one processor to the in line of a second processor in a multi-processor installation by means of the Direct Control feature provides 256 code combinations. The code sets can be any required by the program including EBCDIC and USASCII with code interpretation being performed by the program.

When a transmitting processor issues a Write Direct instruction, an external interrupt is set in the receiving processor (specified by the I-Field of the Write Control instruction) in response to the signal. To service the interrupt, the receiving processor issues a Read Direct instruction to accept the control byte and then issues a Write Direct with an acknowledgement code to the transmitting processor. (Write Direct of an acknowledgement code does not require a return acknowledgement.) When an acknowledgement has been received from each of the receiving processors (if more than one connected), the transmitting processor may execute another transmission.

In the event of power failing in a processor, interrupt occurs to processor state P_4. In a multi-processor installation with the Direct Control feature, the failing processor issues a Write Direct instruction with a data byte of all zero bits to all processors it is connected to in the system.

Note: The Direct Control feature does not provide error checking on the data transmitted. When checking is required, it must be performed by program.

DIRECT CONTROL INTERFACE

◆ The Director Control interface connects from two to six processors into a multi-processor complex. Each of the processors can have up to six direct control trunks which contain the signal lines that transmit and receive the direct control information. These signal lines function as follows:

Static Out Lines

◆ The Static Out lines are logically identical (common) on all trunks (information on one trunk is identical to information of all other trunks). The state of these Static Out lines is established when a Write Direct instruction is executed and remains static until altered by a subsequent Write Direct instruction. Parity is not generated or checked on these lines. (See Write Direct instruction.)

Static In Lines

◆ The Static In lines provide the means for the receiving processor to receive 8-bit bytes of data from other transmitting processors via their Static Out lines. Each trunk may be uniquely sampled by a Read Direct instruction which specifies the desired trunk. (See Read Direct instruction.)

Signal Out Line

◆ The Signal Out line provides a signal to the other processors upon execution of a Write Direct instruction. The Direct Control Trunks (DCT) whose Signal Out lines are signaled is specified by the I-Field pattern of the instruction.

External Signal In Line

◆ The External Signal In line provides the means for receiving a signal from other processors via their Signal Out lines. The External Signal In line is logically connected to the external signal interrupt flag associated with each Direct Control Trunk (DCT) as indicated:

Trunk Signaled	*External Interrupt Flag*
DCT #1	1
DCT #2	2
DCT #3	3
DCT #4	4
DCT #5	5
DCT #6	6

Power Failure Line (PFND)

◆ The PFND line is logically identical on all Direct Control Trunks (DCT) in the complex. Its signal is normally up but is dropped upon detection of a power failure. The signal on this line remains down throughout the one millisecond of available program time remaining, and does not come up again until after power has been restored.

Power Failure Inhibit In Line (PFIR)

◆ The PFIR line provides the means for inhibiting a Read Direct instruction of the associated Static In lines when its signal is dropped. When the signal is dropped, all zeros are read by the receiving processor.

DUAL-PROCESSOR COMPLEX

◆ The following illustration is presented to demonstrate the manner in which two processors are interconnected. In this instance only one cable is required.

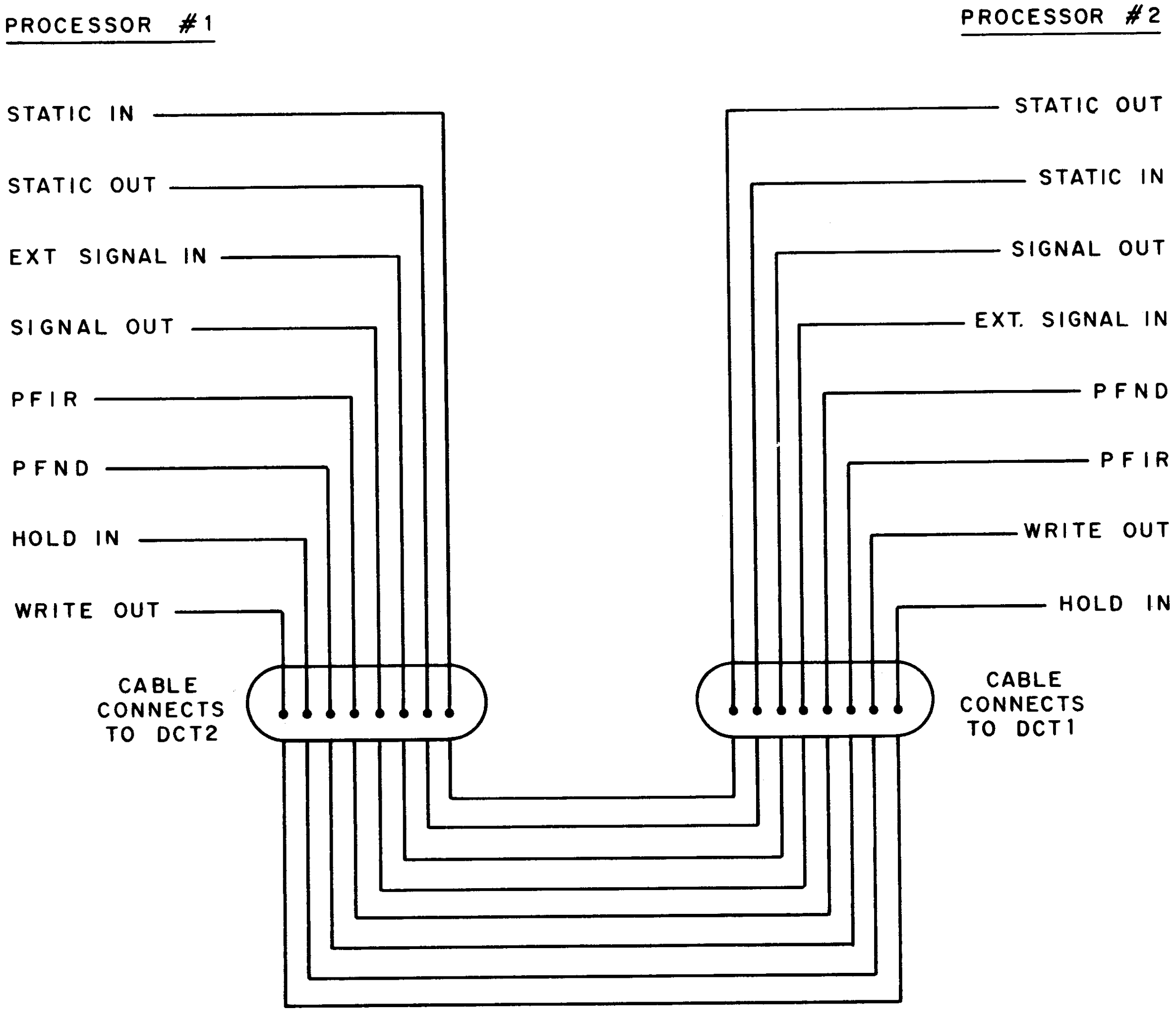


Figure 12. Dual-Processor Complex

MASTER/SATELLITE COMPLEX

♦ The Master/Satellite complex permits the master processor to communicate with its satellites and the satellites to communicate with the master processor. However, the satellites cannot communicate with each other. The following illustration demonstrates the manner in which the master processor interconnects with up to five satellite processors via the Direct Control Trunks (DCT).

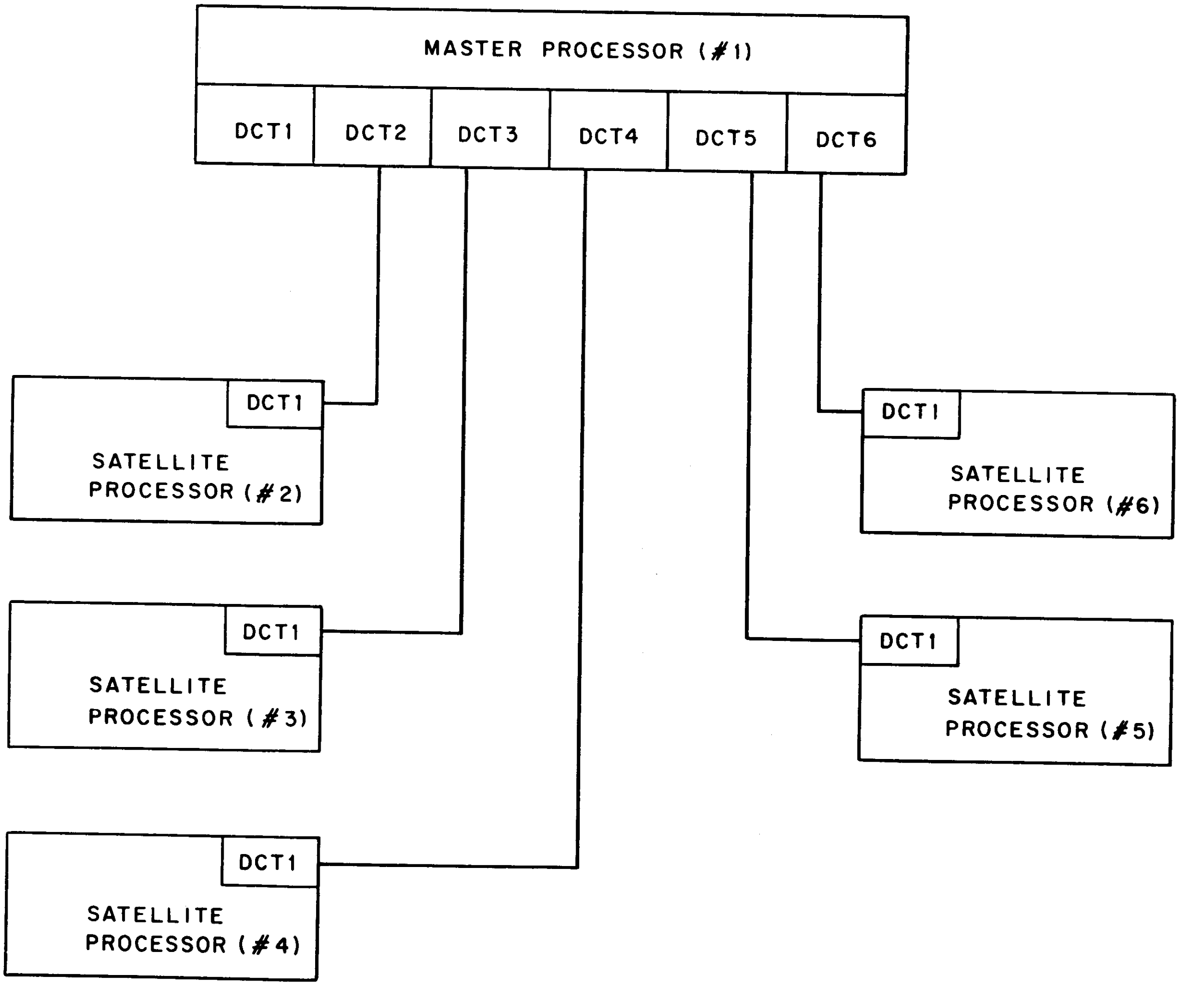


Figure 13. Master/Satellite Complex

MAXIMUM MULTI-PROCESSOR COMPLEX

◆ The following illustration demonstrates the manner in which six processors may be interconnected so that any two processors may communicate.

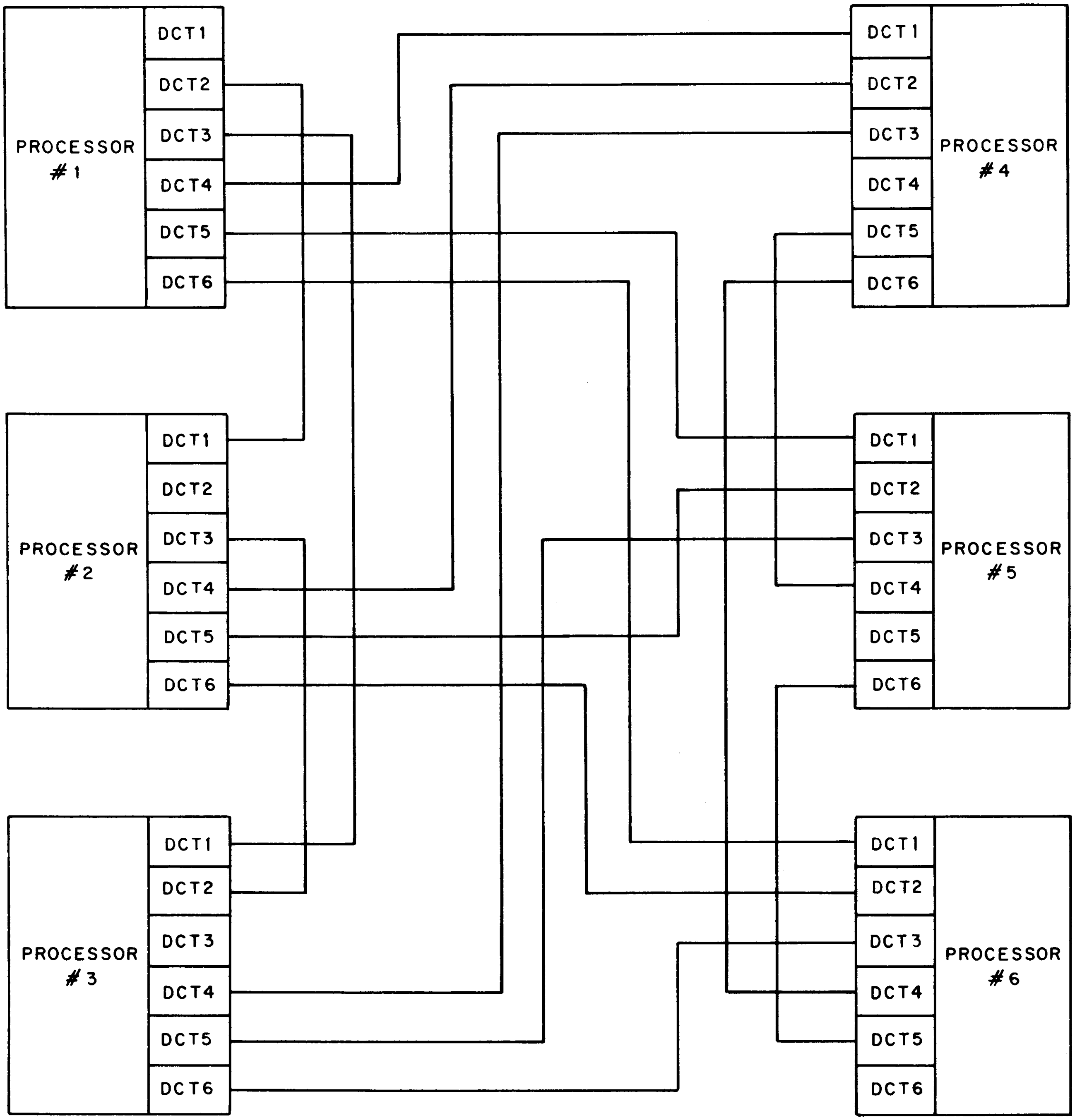


Figure 14. Maximum Multi-Processor Complex

OPERATIONAL PROCEDURES

◆ The following sections are furnished to illustrate typical operational procedures when using the Direct Control feature. They are presented for *clarification only* and are not meant to imply fixed and firm standards. For a detailed description of the actual programming procedures, reference should be made to the applicable reference manuals.

Transmission Procedure

◆ *User Program — (P_1)* The user program in Processing State (P_1) contains a Supervisor Call instruction with a Write Direct Interrupt Code. In addition, it contains the following parameters required when interrupt is effected to the operating system in processor state (P_3) :

Data Byte (8-bit code)

Signal Byte (specifies processor(s) to which
Write Direct is addressed)

Return Address (for return to normal processing)

Operating System — (P_3) The operating system accepts the Supervisor Call Interrupt and issues a Program Control instruction to (P_2). In addition, the locations of the user parameters are saved, the processor is set to the Privileged Mode and a change made from (P_3) to (P_2).

Supervisor Call Routine — (P_2) The Interrupt Weight is used to branch to the Supervisor Call routine where the Supervisor Call Interrupt Code is decoded and a branch is made to the required routine, in this case the Write Direct routine. The Write Direct routine then performs the following:

1. Checks to determine whether Write Direct instruction can be issued or must be stacked in queue.
2. Fetches the user parameters.
3. Sets Write Direct instruction I-Field to the Signal byte, the Address field to the Data byte, and the Return After Interrupt to the user Return Address in (P_1).
4. Executes Write Direct instruction.
5. If no acknowledgement is received, sets control in Acknowledge queue.
6. Set processor to non-privileged mode.
7. After interrupt, executes Program Control instruction and branch to user return address in (P1).

Response Procedure

◆ *Operating System — (P_3)* The operating system accepts the Direct Control Interrupt and issues a Program Control instruction to (P_2). In addition, the processor is set to privileged mode and a change made from (P_3) to (P_2).

Read Direct Routine — (P_2) The Interrupt Weight is used to branch to the Read Direct routine. The Read Direct routine then performs the following:

1. Issues a Read Direct instruction to read the Data Byte.
2. Saves the Data Byte and the External Interrupt number (which corresponds to the transmitting processor) for user Read Direct processing.

Response Procedure *(Cont'd)*

3. Issues a Program Control instruction to (P_1) and sets processor to non-privileged mode.

4. Changes from (P_2) to (P_1) and branches to user Read Direct routine.

User Read Direct Routine — (P_1) Using the External Interrupt number, the user Read Direct routine determines the transmitting processor number and decodes the Data Byte to determine the type of action required.

If the Power Failure code (all zeros) is received, the processor that is down is removed from the system configuration and a return to normal processing is effected.

For all other codes received, a Write Direct acknowledgement is issued as follows:

1. Supervisor Call is issued with a Write Direct Interrupt Code.

2. A Write Direct instruction with a Data Byte of an Acknowledge Code and a return address of the user Read Direct routine is executed.

When the return is accomplished, the function specified by the Data Byte initially read is performed, and at the end of the Read Direct processing a branch is made back to the (P_1) program.

PRIVILEGED INSTRUCTIONS

INTRODUCTION

◆ The instructions described in this section are called *privileged instructions* and can only be executed if the non-privileged mode bit (bit position 15 in the Interrupt Status register) for the current state is zero.

In addition to the standard privileged instruction set, inclusion of the memory protect and/or the direct control optional features cause additional privileged instructions to be added.

INSTRUCTION FORMATS

RR Format

Op Code	R_1	R_2
0 7	8 11	12 15

Description

◆ The RR format is used only by the Set Storage Key and the Insert Storage Key instructions. The contents of the general register specified by the R_1 field is the first operand. The general register specified by the R_2 field contains the second operand address.

SI Format

Op Code	I_2	B_1	D_1
0 7	8 15	16 19	20 31

Description

◆ The SI format is used by the Program Control, the Write Direct, the Read Direct instructions and all input/output instructions. The first address (B_1/D_1) specifies the main memory location of the first operand. The second operand is the immediate byte in the I_2 field.

SS Format

Op Code	L	B_1	D_1	B_2	D_2
0 7	8 15	16 19	20 31	32 35	36 47

Description

◆ The SS format is used by the Load Scratch Pad and the Store Scratch Pad instructions. The location of the first operand is specified by the first address (B_1/D_1), and the location of the second operand is specified by the second address (B_2/D_2). The L field is the number of *words* in addition to the addressed *word* that are to be transferred.

INTERRUPT ACTION

◆ The following interrupt conditions can occur as a result of a privileged instruction:

Address Error

Addressing

◆ An address error interrupt occurs when an address specifies a location outside the available main memory of the particular installation. The operation is terminated at the point of error. The result data and condition code, if produced, are unpredictable. If the address of an instruction is invalid, the operation is suppressed.

Specification

◆ An address error interrupt occurs when:

1. A Load Scratch Pad or Store Scratch Pad instruction specifies a first or second address which is not on a word boundary.
2. Bits 28 through 31 of the second operand of a Set Storage Key or Insert Storage Key instruction are not zero.
3. The memory protect feature is not installed and the protection key in the Interrupt Status register for the current program state is not zero.

In these error interrupt conditions, the operation is suppressed. The data in main memory and registers is unchanged.

Protection

◆ An address error interrupt occurs when the storage key and the protection key of the result location do not match. The operation is terminated. The result data is unpredictable. (This interrupt can occur only if the memory protect feature is installed.)

Privileged Operation

◆ A privileged operation interrupt occurs if execution of any privileged instruction is attempted and the non-privileged mode bit (bit position 15 in the Interrupt Status register) for the current state is 1. The operation is suppressed and the condition code, registers, and main memory are unaltered.

Operation Code Trap

◆ An operation code trap interrupt occurs under the following conditions:

1. The memory protect feature is not installed and an attempt to execute a Set Storage Key or Insert Storage Key instruction is made.
2. The direct control feature is not installed and an attempt to execute a Write Direct or Read Direct instruction is made.

Function Call (FC)

General Description

◆ This instruction is used to execute Elementary Operation (EO) routines contained in Read Only Memory (ROM). The I field is used to specify one of 128 possible EO routines. The address field specifies the address of the parameters used by the specified EO routine.

Format (SI)

9A	I_2	B_1	D_1
0 7	8 15	16 19	20 31

Condition Code

◆ Unchanged by this instruction, however, the routine called may modify the condition codes.

Interrupt Action

◆ Op Code Trap.

Power Failure.

Machine Check.

Privileged Operation.

Addressing.

Paging Error.

Paging Queue.

Others as defined by the specifications of the routine called.

Notes

◆ 1. The I field specifies one of 128 possible EO routines, called Special Functions. All 8-bit codes in which the 2^4 bit is zero are available for Special Functions. This instruction is only incorporated on the 70/46 Processor. The routine specified by the I field must be incorporated in the ROM. Otherwise, an Op Code Trap Interrupt condition occurs.

2. This instruction is available to 70/46 programs only. If it is executed in the 70/45 mode, an Op Code Trap Interrupt condition occurs.

3. If a location outside the available memory is addressed, an Addressing Interrupt condition occurs.

4. If this instruction is attempted under any of the following conditions, a Paging Error Interrupt condition occurs and the instruction is terminated with unpredictable results:

   a. A nonexistent Translation Table element is addressed (i.e., the two unused bits of a segment field of a virtual address are not zero).

   b. A 2,048-byte page is addressed in the high-order address half of a 4,096-byte page.

   c. A write operation into a location within a non-writable page is attempted.

Notes *(Cont'd)*

d. If this instruction is attempted under the following conditions the indicated interrupt results:

S-Bit	*N-Bit*	*D-Bit*	*Interrupt*
N/A	1	1	Paging Error
0	1	0	Paging Error
1	1	0	Privileged Operation

N/A — Not applicable.

5. If this instruction is attempted in a non-utilizable page, a Paging Queue Interrupt condition occurs and the instruction is suppressed.

Special Function #1 Load Translation Memory (LTM)

General Description

◆ The translation memory is loaded with blocks of halfwords from memory, where the blocks are specified by a Block Address Table which is also in memory. The location of the Block Address Table is addressed by the low-order three bytes of the general register specified by R1. The number of memory locations per block is given in the high-order byte of the general register specified by R1. The number of blocks to be loaded is specified by the low-order halfword of the general register specified by R2. The first location of the translation memory into which an entry is to be placed is specified by the high-order halfword contained in the general register specified by R2.

I Code

◆ C0.

Format

R_1	R_2
0 3	4 7

Condition Code

◆ Unchanged.

Interrupt Action

◆ Addressing.

Power Failure.

Machine Check.

Paging Error.

Paging Queue.

Notes

◆ 1. The high-order byte of the general register specified by R1 contains a count of 0–255 to specify 1–256 translation memory locations in each of the blocks.

2. The low-order three bytes of the general register specified by R1 containing the address of the Block Address Table may be either virtual or direct as indicated by the D bit.

3. Bit positions 7 through 15 in the general register specified by R2 contains the address of 0–511 of the first location of the Translation Table to be loaded. This may specify any location in the Translation Table. Bit positions 0 through 6 are not used and must be zeros. This is a program restriction only.

4. Bit positions 23 through 31 in the general register specified by R2 contains a count 0–511 specifying the number of words in the Block Address Table 1 to 512, respectively. Bit positions 16 through 22 are not used and must be zeros. This is a program restriction only.

Notes
(Cont'd)

5. The high-order byte (0–255) in each Block Address Table word specifies the number of halfwords (1 to 256) to be loaded from the block beginning at the address specified by the low-order three bytes. If this count is less than the count contained in the high-order byte of the general register specified by R1 (translation memory block size), the remaining Translation Table locations of the specified block are loaded with zeros. If this count is greater than the translation memory block size count, loading is terminated by the block size count reaching zero.

6. If an address of the Block Address Table specified by the general register designated by R1 is not on a full word boundary, an Addressing Interrupt condition occurs. The operation is suppressed with the operands unchanged.

7. If a location outside the available memory is addressed, an Addressing Error Interrupt condition occurs. The operation is terminated with unpredictable results.

8. When the translation memory entry is made from main memory, the word is copied except for the G-bit which is unaltered in main memory, and is reset to zero in the translation memory.

9. The format of the halfword in main memory from which the translation memory entry is copied is:

   WGUSEMXXXPPPPPPH

10. The format of the translation memory entry is specified under the Translation Memory description in this manual.

11. If the block address specified in Block Address Table entry is not on a halfword boundary, an Addressing Interrupt condition occurs. The operation is terminated with unpredictable results.

12. If this Special Function is attempted under any of the following conditions, a Paging Error Interrupt Condition occurs and the Special Function is terminated with unpredictable results:
    a. If either the address of the Block Address Table or the block address in a Block Address Table entry specifies a nonexistent translation memory element (i.e., the two unused bits of the segment field of a virtual address are not zeros).
    b. If either the address of the Block Address Table or the block address in a Block Address Table entry specifies a 2,048-byte page in the high-order address half of a 4,096-byte page.

13. If this Special Function is attempted with either a Block Address Table address or the block address of a Block Address Table entry specifying a non-utilizable page, a Paging Queue Interrupt condition occurs and the instruction is terminated with unpredictable results.

14. The contents of the Translation Table being loaded in the translation memory do not cause a Paging Queue condition or Paging Error Interrupt condition.

Special Function #2 Scan Translation Memory and Store (STMS)

General Description

◆ The Translation Memory is scanned for nonzero values of the G bit, and the table of halfwords thus found is stored into the corresponding halfwords of the block memory identified by the Block Address Table, etc., as defined for the Load Translation Memory Special Function (C0).

I Code

◆ C1.

Format

R_1	R_2
0 3	4 7

Condition Code

◆ Unchanged.

Interrupt Action

◆ Addressing.

Power Failure.

Machine Check.

Paging Error.

Paging Queue.

Notes

◆ 1. The high-order byte of the general register specified by R1 contains a count of 0–255 to specify 1–256 translation memory locations in each of the blocks.

2. The low-order three bytes of the general register specified by R1 containing the address of the Block Address Table may be either virtual or direct as indicated by the D bit of the address field.

3. Bit positions 7 through 15 in the general register specified by R2 contain the address 0–511 of the first translation memory location to be loaded. This may specify any location in the translation memory. Bit positions 0 through 6 are not used and must be zeros. This is a program restriction only.

4. Bit positions 23 through 31 in the general register specified by R2 contain a count 0–511 specifying the number of words in the Block Address Table 1 to 512, respectively. Bit positions 16 through 22 are not used and must be zeros. This is a program restriction only.

5. The high-order byte (0–255) in each Block Address Table word specifies the number of halfwords (1 to 256) to be loaded from the block specified by the low-order three bytes.

   If this count is less than the count contained in the high-order byte of the general register specified by R1 (translation memory block size), the remaining translation memory locations of the specified block are loaded with zeros. If this count is greater than the translation memory block size count, loading is terminated by the translation memory block size count reaching zero.

Notes *(Cont'd)*

6. If an address of the Block Address Table specified by the General Register designated by R1 is not on a full word boundary, an Addressing Interrupt condition occurs. The operation is suppressed with the operands unchanged.

7. If a location outside the available memory is addressed, an Addressing Error Interrupt condition occurs. The operation is terminated with unpredictable results.

8. The format of the halfword in main memory from which the translation entry is copied is:

   WGUSEMXXXPPPPPPH

9. The format of the Translation Table entry in the translation memory is specified under the Translation Memory description in this manual.

10. The format of the translation memory entry is specified under the Addressing description in this manual.

11. If the block address specified in Block Address Table entry is not on a halfword boundary, an Addressing Interrupt condition occurs. The operation is terminated with unpredictable results.

12. If this Special Function is attempted under any of the following conditions, a Paging Error Interrupt condition occurs and the Special Function is terminated with unpredictable results:

    a. If either the address of the Block Address Table or the block address in a Block Address Table entry specifies a nonexistent translation memory element (i.e., the two unused bits of the segment field of a virtual address are not zeros).

    b. If either the address of the Block Address Table or the block address in a Block Address Table entry specifies a 2,048-byte page in the high-order address half of a 4,096-byte page.

    c. If either the address of the Block Address Table or the block address in a Block Address Table entry specifies a page that is not writable.

13. If this Special Function is attempted with either a Block Address Table address or the block address of a Block Address Table entry specifying a nonutilizable page, a Paging Queue Interrupt condition occurs and the instruction is terminated with unpredictable results.

14. The contents of the Translation Table being loaded to the translation memory do not cause a Paging Queue or Paging Error Interrupt condition.

15. If the translation memory block size count is greater than the individual Block Address Table item count (N) that individual block scan and store is completed when the N-halfwords have been stored.

16. The contents of the translation memory being stored into memory do not cause a Paging Queue condition or Paging Error Interrupt condition.

Special Function #3 Store Translation Memory (STM)

General Description

◆ The 9-bit count contained in the lower half of the general register specified by R_2 specifies the number of Translation Table halfwords to be stored into memory (beginning with the memory address contained in the general register specified by R_1). The 9-bit Translation Table initial address is contained in the upper half of the general register specified by R_2.

I Code

◆ C4.

Format

R_1	R_2
0 3	4 7

Condition Code

◆ Unchanged.

Interrupt Action

◆ Addressing.

Power Failure.

Machine Check.

Paging Error.

Paging Queue.

Notes

◆ 1. The count, contained in bit positions 23 through 31, specifies 1–512 Translation Table halfwords with a count of 0–511, respectively. Bit positions 16 through 22 are not used and must be zeros. This is a program restriction only.

2. The initial memory address may be either virtual or direct.

3. If an address not on a halfword boundary is specified, an Address Interrupt condition occurs. The operation is suppressed with the operands unchanged.

4. If a location outside the available memory is addressed, an Addressing Error Interrupt condition occurs. The operation is terminated with unpredictable results.

5. The contents of the translation memory being stored into memory do not cause a Paging Queue condition or Paging Error Interrupt condition.

6. If this Special Function is attempted under any of the following conditions, a Paging Error Interrupt condition occurs and the operation is terminated with unpredictable results.

   a. If the main memory address specifies a nonexistent translation table element (i.e., the two unused bits of the segment field of a virtual address are not zeros).

   b. If the main memory address specifies a 2,048-byte page in the high-order address half of a 4,096-byte page.

   c. If the main memory address specifies a page that is not writable.

7. If this Special Function is attempted with a main memory address specifying a nonutilizable page, a Paging Queue Interrupt condition occurs and the operation is terminated with unpredictable results.

Special Function #4 Load Interval Timer (LIT)

General Description

◆ This Special Function loads the Interval Timer with a halfword. The address of the halfword to be loaded is indicated by the contents of the memory location addressed by the address field of the Function Call instruction. If the value loaded is a nonzero value, the timer begins to decrement by one. If the value loaded is zero, and the timer is not counting, the instruction has no effect. If this instruction is executed while the timer is running, the contents of the counter are replaced by the specified halfword. If the specified halfword is zero, the timer is reset to zero (shut-off) and no interrupt occurs for this condition.

I Code

◆ 02.

Format

Base	Displacement
0 3	4 15

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address Error.

Power Failure.

Machine Check.

Paging Error.

Paging Queue.

Notes

◆ 1. If either this Special Function or the timer halfword addresses are not on halfword boundaries, an Addressing Error Interrupt condition occurs.

2. If the counter is reset to zero after zero occurs and the interrupt flag has been set, the interrupt flag will not be cleared.

3. Use of the Interval Timer and Diagnostic Snapshot by programs may not occur together, since the counter register is common to both. If the Diagnose function is initiated while the Interval Timer is running, the shared counter is cleared to zero without occurrence of the Interval Timer Interrupt and the Diagnose function assumes control of the counter. If the function being diagnosed is the Load Interval Timer, the actual loading of the counter is inhibited but the E. O. Flow is Diagnosed.

Special Function #5 Store Interval Timer (SIT)

General Description

◆ This Special Function stores the current contents of the Interval Timer into a memory halfword. The address of the halfword to receive the contents of the Interval Timer is indicated by the contents of the memory location addressed by the address field of the Function Call instruction. This instruction has no effect upon the Interval Timer.

I Code

◆ 03.

Format

Base	Displacement
0 3	4 15

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address Error.

Power Failure.

Machine Check.

Paging Error.

Paging Queue.

Note

◆ 1. If either this Special Function or the timer halfword addresses are not on halfword boundaries, an Addressing Error Interrupt condition occurs.

2. Use of the Interval Timer and Diagnostic Snapshot by programs may not occur together, since the counter register is common to both. If the Diagnose function is initiated while the Interval Timer is running, the shared counter is cleared to zero without occurrence of the Interval Timer Interrupt and the Diagnose function assumes control of the counter. If the function being diagnosed is the Load Interval Timer, the actual loading of the counter is inhibited but the E. O. Flow is Diagnosed.

Special Function #6 Paging Queue and Paging Error Interrupt Service

General Description

◆ This Special Function determines all the segment and page addresses specified by the instruction whose Translation Table elements caused or might cause a Paging Queue condition or Paging Error Interrupt condition and adjusts the NIA field of the P counter in the suppressed program state. It interfaces with the program by stacking a list of addresses (page and segment) with applicable program indicators to identify the status of each. The address of the beginning of the stack (up to eight halfwords per stack) of the effective address list is indicated by the contents of the memory location addressed by the address field of the Function Call instruction.

I Code

◆ 01.

Format

Base	Displacement
0 3	4 15

(Address of the beginning of the up to eight halfword stack for the effective address list.)

Condition Code

◆ Unchanged.

Interrupt Action

◆ Power Failure.
Machine Check.

Notes

◆ 1. This Special Function can be used to analyze those interrupts that can occur during staticizing or execution. The NIA field of the object P counter and the ILC are set correctly to permit this Special Function to back-up the object P counter for reentry to the object instruction following completion of the page calling.

2. An index is provided in General Purpose Register 15 of the current state. This index is the number of virtual addresses in the stack minus one times 2; i.e., 0 is equivalent to 1 address and 14 is equivalent to 8 addresses. The addresses of the stack are either virtual or actual, depending on the status of the appropriate D bit settings, D = 0 and D = 1 respectively.

3. The size of the address stack is a function of the instruction type as follows:

Format	Number of Stack Addresses	Range of General Purpose Register 15
RR	1 (instruction)	0
RS	1–3 (up to 2 instruction and 1 operand)	0–4
*RX	1–3 (up to 2 instruction and 1 operand)	0–4
SI	1–3 (up to 2 instruction and 1 operand)	0–4
SS	1–6 (up to 2 instruction and 4 operand)	0–10

* If the instruction is an Execute, the number of stack addresses is 1 to 8 (up to 4 instruction and 4 operand) and the range of General Purpose Register 15 is 0–14, depending on the format of the object instruction.

Notes *(Cont'd)*

4. The Special Function must shift the effective address to provide the segment and page in the low-order position within each stack item. The format of an address in the stack is as follows:

Indicators	SEG	PAGE
0 6	7 9	10 15

The seven high-order bit positions, when set (1), indicate the specific interrupt condition(s) for the Paging Error Interrupt (Priority 19) and Paging Queue Interrupt (Priority 20) as follows:

Bit 0: Non-privileged mode was set (N = 1) and the control bit S was reset (S = 0).

Bit 1: Either one or both of the two unused bits of the segment field were not zero.

Bit 2: The Page Control Bit was set (M = 1) and the high-order bit of the Displacement field of the address was set (1).

Bit 3: Non-privileged mode was set (N = 1) and the direct address bit is set (D = 1).

Bit 4: Control bit E was set (E = 1) and a write operation was attempted to the page.

Bit 5: Translation table element has control bit U reset (U = 0) (i.e., page not utilizable).

Bit 6: Flags the stack address as a Direct Address, not subject to translation.

If multiple interrupt conditions of different kinds occur on the same page, the page address is listed once in the address stack and all applicable condition bits are set.

5. This Special Function is to be used for analysis of Paging Error condition or Paging Queue Interrupt condition on the normal instruction set of the 70/46 and is not usable for analysis of other Special Functions.

6. This Special Function provides a maximum of two instruction addresses for I/O instructions. It does not provide the addresses of the Channel Address Word or Channel Control Word for any of the Paging Error or Paging Queue Interrupt conditions.

7. The operand addresses provided for Paging Error condition or Paging Queue Interrupt condition on the Translate and the Translate and Test instructions are based on the assumption that the tables involved are maximum size (256 bytes).

In any case where the table is less than 256 bytes, a false indication of a Paging Error condition of Paging Queue condition may have occurred, and the ending table address provided by this Special Function may be incorrect.

Notes *(Cont'd)*

8. The operand addresses provided for Paging Error condition or Paging Queue Interrupt condition on the Edit and Edit and Mark instructions are based on the assumption that the number of source field bytes and the number of pattern field bytes are equal. In any case where the number of source field bytes is less than the number of pattern field bytes, a false indication of a Paging Error or Paging Queue may have occurred, and the ending source field address provided by this Special Function may not be correct.

9. When a Paging Error or Paging Queue Interrupt occurs, the Program Counter, Interrupt Status Register, and General Purpose Registers of the Interrupted State must not be altered before the special function is executed.

10. This Special Function should be used only if a Paging Queue condition or Paging Error Interrupt condition occurs. Otherwise, the results are unpredictable.

11. The ISI field of the current program states ISR is used to identify the Program State in which the interrupt occurred (Program State to be analyzed).

Load Scratch Pad (LSP)

General Description

◆ Operands from main memory, starting with the storage location specified by the second address (B_2/D_2), are loaded in the scratch-pad memory starting at the location specified by the first address (B_1/D_1).

Format (SS)

D8	L	B_1	D_1	B_2	D_2
0 7	8 15	16 19	20 31	32 35	36 47

Condition Code

◆ Unchanged except when the P counter in scratch-pad memory is loaded.

Interrupt Action

◆ Privileged operation.

Address error:

Addressing.

Specification.

Notes

◆ 1. The L field provides an eight-bit count specifying the number of scratch-pad memory locations to be loaded. An initial count of zero specifies one word to be loaded.

2. The first address specifies scratch-pad memory words 0 through 127 by the seven rightmost bits of the address. The bits to the left of the seven-bit address must be zero.

3. The second address must be on a word boundary. (This is a program restriction.)

4. The processor uses utility registers in scratch-pad memory to execute this instruction. If these registers are included in the range of this instruction results are unpredictable.

Store Scratch Pad (SSP)

General Description

◆ Operands from the scratch-pad memory, starting with the location specified by the first address (B_1/D_1), are stored in main memory locations, starting with the location specified by the second address (B_2/D_2).

Format (SS)

DO	L	B_1	D_1	B_2	D_2
0 7	8 15	16 19	20 31	32 35	36 47

Condition Code

◆ Unchanged.

Interrupt Action

◆ Privileged operation.

Address error:

Addressing.

Specification.

Protection.

Notes

◆ 1. The L field provides an eight-bit count specifying the number of scratch-pad memory locations to be stored. An initial count of zero specifies one word to be stored.

2. The first address specifies scratch-pad memory words 0 through 127 by the seven rightmost bits of the address. The bits to the left of the seven-bit address must be zero.

3. The second address must be on a word boundary. (This is a program restriction.)

Program Control (PC)

General Description

◆ This instruction specifies the termination of program execution in the current state, and the initiation of another state under control of the immediate byte in the I_2 field. The address computed from the B_1/D_1 address components of the instruction is stored in the P counter of the state being terminated (bit positions 8–31).

Format (SI)

82	I_2	B_1	D_1
0 7	8 15	16 19	20 31

Condition Code

◆ The condition code indicators of the state being terminated are preserved in the state's P counter. The condition code in the P counter of the initiated state is then used to set the condition code indicators.

Interrupt Action

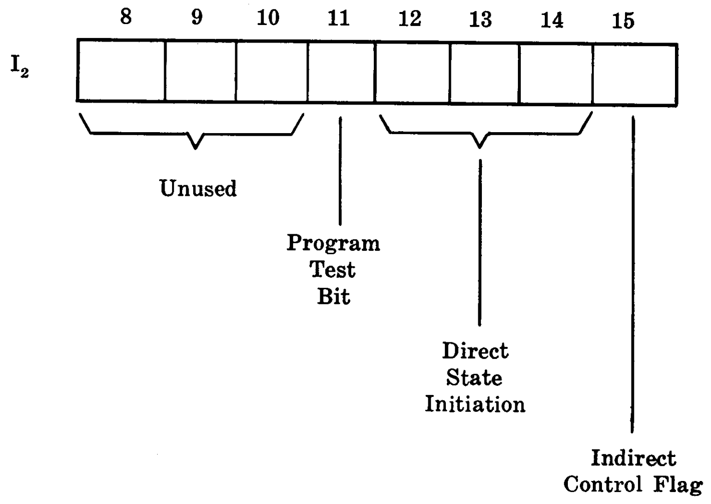
◆ Privileged operation.

Address error:

Addressing.

Note

◆ 1. The immediate byte in the I_2 field of the instruction is divided into four subfields as follows:



Bits 8 through 10 are unused. The three bit unused portion must be zero.

Bit 11 is the program test bit. If bit 11 = 1, the program test mode is initiated. The program test interrupt bit is set in the Interrupt Flag register of the initiated state.

The scan of the Interrupt Flag register in the initiated state is delayed until after the first instruction of the initiated state is executed, at which time the scan is made in normal priority.

If bit 11 = 0, the program test mode is not initiated.

Note *(Cont'd)*

Bits 12 through 14 are the direct state initiation bits. The three-bit direct state initiation codes that may be specified are as follows:

000 — Go to Machine Condition State P_4.
001 — Go to Interrupt Control State P_3.
010 — Go to Interrupt Response State P_2.
011 — Go to Processing State P_1.

Programming Note: The leftmost bit of the three-bit direct state initiation field must be zero. (This is a programming restriction.)

Bit 15 is the indirect control flag bit. If indirect state control is specified (bit 15 = 1), the three-bit direct state initiation field is ignored. The three-bit interrupted state identifier (ISI), which indicates the last state interrupted, specifies the state to be initiated. This information is contained in the Interrupt Status register of the state being terminated.

If bit 15 = 0, direct state initiation is used.

Idle (IDL)

General Description

◆ This instruction effects an idle mode within the processor by continuously branching back to itself.

Format (SI)

80	I_2	B_1	D_1
0 7	8 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Privileged operation.

Notes

◆ 1. When this instruction is operating with the I field zero, the Idle light of the console is on.

2. Any interrupt occurring while the idle mode is in effect is taken (if permitted via the Interrupt Mask register).

3. The B_1 and D_1 fields of this instruction must be zero.

4. For normal programming, the I field must be zero. For maintenance programming, bits within the I field, have the following meaning:

   Bit 15 = 1-set alarm inhibit.
   Bit 14 = 1-reset alarm inhibit.
   Bit 13 = 1-set inhibit simultaneity.
   Bit 12 = 1-reset-inhibit simultaneity.

Diagnose (DIG)

General Description

◆ The purpose of this privileged instruction is to store four additional bytes in the snapshot memory location table which will provide a means for facilitating maintenance techniques on the 70/46 Processor. It is provided for the RCA Customer Service and Engineering Representatives and cannot be used for a program debugging aid.

The mechanics of this instruction are implemented specifically for the 70/46 Processor.

Format (SI)

83	I_2	B_1	D_1
0 7	8 15	16 19	20 31

Note

◆ Use of the Interval Timer and Diagnostic Snapshot by programs may not occur together, since the counter register is common to both. If the Diagnose function is initiated while the Interval Timer is running, the shared counter is cleared to zero without occurrence of the Interval Timer Interrupt and the Diagnose function assumes control of the counter. If the function being diagnosed is the Load Interval Timer, the actual loading of the counter is inhibited but the E. O. Flow is Diagnosed.

Start Device (SDV)

General Description

◆ The contents of the general register specified by B_1 are added to the D_1 field. The resultant sum identifies the channel and device to which the instruction applies. These are specified by bit positions 21 through 31 of the sum. The I-field is not used and must be zeros.

The channel address word in main memory location 72 contains the protection key to be used and the address of the first channel command word. The channel command word designated by the channel address word specifies the operation to be performed, the main memory area to be used, and the action to be taken when the operation is completed. The condition code indicates the result of the instruction.

Format (SI)

9C	I_2	B_1	D_1
0 7	8 15	16 19	20 31

Condition Code

◆ 0 — input/output operation initiated and channel proceeding with execution.

1 — status bits stored in scratch-pad memory.

2 — busy or interrupt pending.

3 — inoperable.

(For a detailed description of the condition code settings, see Notes below.)

Interrupt Action

◆ Privileged operation.

Notes

◆ 1. The address portion of this instruction specifies the device and channel as follows:

Bit Positions			Channel Specified
21	22	23	
0	0	0	Multiplexor
0	0	1	Selector No. 1
0	1	0	Selector No. 2
0	1	1	Selector No. 3
1	0	0	Selector No. 4
1	1	0	Undesignated

Bit positions 24 through 31 specify one of 256 possible devices.

2. The standard device byte and the channel status byte stored by the previous input/output instruction in scratch-pad memory are destroyed if the condition code at the completion of the Start Device instruction is 0 or 1.

3. Status storage (channel status byte and standard device byte), if required, occurs before the Start Device instruction terminates.

4. Condition Code 0 is set under the following conditions:

Notes *(Cont'd)*

a. The device control electronics and the device specified are available.

b. The Start Device instruction specifies a Sense command to a device that is inoperable.

5. Condition Code 1 indicates that either the channel status byte or the standard device byte has been stored in the channel registers in scratch-pad memory for the specified channel.

The channel status byte is stored under the following conditions:

a. A parity error occurs while accessing the Channel Address Word (CAW), Channel Block Address (CBA), or a Channel Command Word (CCW). The channel control check bit in the channel status byte is set.

b. The Memory Protect feature is not installed and the key in the CAW is not zero. The program check bit in the channel status byte is set.

c. The main memory address specified in the CAW or CBA is not on a double word boundary. The program check bit in the channel status byte is set.

d. The main memory address in the CCW specifies an address outside the available memory for the system. The program check bit in the channel status byte is set.

The standard device byte is stored under the following conditions:

a. The specified device control electronics on the multiplexor channel indicates that a device request interrupt pending condition is present. The external device request interrupt pending bit in the standard device byte is set.

b. The Start Device instruction specifies a command which is other than a Sense command and the addressed device is inoperable. The device inoperable bit in the standard device byte is set.

c. The specified device is busy but the device control electronics is not busy (i.e., tape rewinding, off-line seek to a random access device end bit in the standard device byte are set.

6. Condition Code 2 is set under the following conditions:

a. A selector channel is specified that is busy.

b. A selector channel is specified that has an interrupt pending (termination or external device request).

c. The multiplexor channel is specified and it is operating in burst mode.

d. The multiplexor channel is specified and the addressed device control electronics is busy with addressed or non-addressed device.

e. The multiplexor channel is specified and the addressed device control electronics has a termination interrupt pending.

f. A burst mode operation is directed to the multiplexor and there is a termination interrupt pending on one of the attached device control electronics.

7. Condition Code 3 is set under the following conditions:

a. A selector channel is specified that is not in the system.

b. The specified device control electronics is inoperable.

Notes *(Cont'd)*

8. If the condition code is 1, 2 or 3 the input/output operation is not initiated.

9. Parity errors that occur while fetching the CAW, CBA, or CCW or that occur after the input/output operation has been initiated do not cause a machine check interrupt. A channel interrupt occurs and the program is notified of the error via the channel status byte.

10. If the first CCW is a Transfer in Channel command the Start Device instruction terminates and the condition code is set to 0. However, the specified device control electronics recognizes this command as an illegal operation and causes a channel interrupt to occur.

Halt Device (HDV)

General Description

◆ The contents of the general register specified by B_1 are added to the D_1 field, and the resultant sum identifies the channel to be halted. The channel is specified by bit positions 21 through 23 of the sum. If a multiplexor is specified, bit positions 24 through 31 of the sum identify the device to be halted. The I field is not used and must be zeros. Bufferred devices operating off-line, and independent of the channel/device control electronics, cannot be stopped by using this instruction. The condition code specifies the results of the instruction.

Format (SI)

9E	I_2	B_1	D_1
0 7	8 15	16 19	20 31

Condition Code

◆ 0 — not busy.

1 — standard device byte stored in scratch-pad memory.

2 — termination accepted.

3 — inoperable.

(For a detailed description of the condition code settings, see Notes below.)

Interrupt Action

◆ Privileged operation.

Notes

◆ 1. The address portion of this instruction specifies the device and channel as follows:

Bit Positions			Channel Specified
21	22	23	
0	0	0	Multiplexor
0	0	1	Selector No. 1
0	1	0	Selector No. 2
0	1	1	Selector No. 3
1	0	0	Selector No. 4
1	1	0	Undesignated

Bit positions 24 through 31 specify one of 256 possible devices.

2. If a device operating on a selector channel is to be halted, the device number does *not* have to be specified.

3. The channel address word in main memory location 72, the channel block address in main memory location 76, and the channel command word are *not* used by this instruction.

4. A termination interrupt occurs when any input/output operation is terminated. Status bits are stored in scratch-pad memory when the termination interrupt occurs.

5. All five flags in CCR-II are cleared if the Halt Device instruction is accepted. Therefore, upon termination, the incorrect length counter in the channel status byte is set if the count is not zero.

Notes *(Cont'd)*

6. A Halt Device instruction that specifies a multiplexor channel that is operating in the burst mode must specify a device that is operating in the burst mode.

7. Condition Code 0 is set under the following conditions:

   a. The device control electronics or the device specified on the multiplexor channel is not busy. No termination is required.

   b. A selector channel or the multiplexor channel operating in burst mode is specified and it is not busy. No termination is required.

   c. The multiplexor channel is specified and the addressed device control electronics has a termination interrupt pending. No termination is required.

8. Condition Code 1 indicates that the specified device is on the multiplexor channel and that the standard device byte has been stored in the channel registers in scratch-pad memory for the multiplexor channel. The channel status byte is never stored.

   The standard device byte is stored under the following conditions:

   a. The specified device indicates that a device request interrupt pending condition is present. The external device request interrupt pending bit in the standard device byte is set.

   b. The specified device is busy but the device control electronics is not busy (i.e., tape rewinding). The device busy bit in the standard device byte is set.

   c. The specified device is inoperable. The device inoperable bit in the standard device byte is set.

9. Condition Code 2 is set under the following conditions:

   a. A selector channel is specified that is busy.

   b. The multiplexor channel is specified and it is operating in the burst mode.

   c. The multiplexor channel is specified and the addressed device control electronics and device are busy.

10. Condition Code 3 is set under the following conditions:

    a. A selector channel is specified that it is not in the system.

    b. The specified device control electronics is inoperable.

11. Status storage (standard device byte), if required, occurs before the Halt Device instruction terminates.

Test Device (TDV)

General Description

◆ The contents of the general register specified by B_1 are added to the D_1 field. The resultant sum identifies the channel and device to which the instruction applies. These are specified by bit positions 21 through 31 of the sum. The I-field is not used and must be zeros. The condition code specifies the results of the instruction.

Format (SI)

9D	I_2	B_1	D_1
0 7	8 15	16 19	20 31

Condition Code

◆ 0 — available.

1 — standard device byte stored in scratch-pad memory.

2 — busy or interrupt pending.

3 — inoperable.

(For a detailed description of the condition code settings, see Notes below.)

Interrupt Action

◆ Privileged operation.

Notes

◆ 1. The address portion of this instruction specifies the device and channel as follows:

Bit Positions			Channel Specified
21	22	23	
0	0	0	Multiplexor
0	0	1	Selector No. 1
0	1	0	Selector No. 2
0	1	1	Selector No. 3
1	0	0	Selector No. 4
1	1	0	Undesignated

Bit positions 24 through 31 specify one of 256 possible devices.

2. The channel address word in main memory location 72, the channel block address in main memory 76, and the channel command word are not used by this instruction.

3. Status storage (standard device byte), if required, occurs before the Test Device instruction terminates.

4. Condition Code 0 is set if the device control electronics and the device are available.

   Note: There may be pending interrupts on the multiplexor channel that would prohibit a burst mode operation to be initiated.

5. Condition Code 1 indicates that the standard device byte has been stored in the channel registers in scratch-pad memory for the specified channel. The channel status byte is never stored by this instruction.

Notes *(Cont'd)*

The standard device byte is stored under the following conditions:

a. The specified device control electronics on the multiplexor channel indicates that a device request interrupt pending condition is present. The external device request interrupt pending bit in the standard device byte is set.

b. The specified device is busy but the device control electronics is not busy (i.e., tape rewinding, off-line seek to a random access device). The device busy bit in the standard device byte is set.

c. The specified device is inoperable. The device inoperable bit in the standard device byte is set.

6. Condition Code 2 is set under the following conditions:

   a. A selector channel is specified that is busy.

   b. A selector channel is specified that has an interrupt pending (termination or external device request.)

   c. The multiplexor channel is specified and it is operating in burst mode.

   d. The multiplexor channel is specified and the addressed device control electronics is busy with addressed or non-addressed device.

   e. The multiplexor channel is specified and the addressed device control electronics has a termination interrupt pending.

7. Condition Code 3 is set under the following conditions:

   a. A selector channel is specified which is not in the system.

   b. The specified device control electronics is inoperable.

   c. A device is specified that is not in the system.

Check Channel (CKC)

General Description

◆ The contents of the general register specified by B_1 are added to the D_1 field, and the resultant sum identifies the input/output channel to be tested. This is specified by bit positions 21 through 23 of the sum. Only the channel is tested.

Format (SI)

9F	I_2	B_1	D_1
0 7	8 15	16 19	20 31

Condition Code

◆ 0 — a. The specified selector channel is not busy and has no interrupts pending.

b. The specified multiplexor channel is not operating in the burst mode.

1 — The specified selector channel has an external device request interrupt pending.

2 — a. The specified selector channel is busy or has a terminating interrupt pending.

b. The specified multiplexor is operating in the burst mode.

3 — A selector channel is specified that is not in the system.

Interrupt Action

◆ Privileged operation.

Notes

◆ 1. The address portion of this instruction specifies the channel to be tested as follows:

Bit Positions			Channel Specified
21	22	23	
0	0	0	Multiplexor
0	0	1	Selector No. 1.
0	1	0	Selector No. 2.
0	1	1	Selector No. 3
1	0	0	Selector No. 4
1	1	0	Undesignated

2. The channel address word in main memory location 72, the channel block address in main memory 76, and the channel command word are not used by this instruction.
3. The device address (bit positions 24 through 31 of the sum) is not used by this instruction.
4. Status bits (channel status byte and standard device byte) are not stored in scratch-pad memory by this instruction.
5. Current operations proceeding in the specified channel are unaffected by this instruction.

Insert Storage Key (ISK)

General Description

◆ The storage key of the 2,048-byte main memory block, which is located at the address contained in the general register specified by the second address (R_2), is inserted in the general register specified by the first address (R_1).

Format (RR)

09	R_1	R_2
0 7	8 11	12 15

Condition Code

◆ Unchanged.

Interrupt Action

◆ Privileged operation.

Address error:

Addressing.

Specification.

Operation code trap (if the memory protect feature is not installed).

Notes

◆ 1. The general register specified by the second address (R_2) contains the location of the 2,048-byte main memory block in bits 8 through 20. Bits 0 through 7 and 21 through 27 are ignored. Bits 28 through 31 must be zero.

2. When the five-bit storage key is inserted into bits 24 through 28 of the general register specified by the first address, bits 0 through 23 are unaltered and bits 29 through 31 are made zero.

3. The address of the storage key for a specific 2,048-byte main memory block is specified in R_2 by a binary count as shown in the following examples:

Storage Key Address in R_2

IGNORED	0 0 0 0 0 0 0 0 0 0 0 0 0	IGNORED	0 0 0 0
0 7	8 20	21 27	28 31
	Address of Storage key for *first* 2,048 main memory block		Must be zeros

IGNORED	0 0 0 0 0 0 0 0 0 0 0 1 0	IGNORED	0 0 0 0
0 7	8 20	21 27	28 31
	Address of Storage key for *third* 2,048 main memory block		Must be zeros

IGNORED	0 0 0 0 0 0 0 0 0 1 0 0 1	IGNORED	0 0 0 0
0 7	8 20	21 27	28 31
	Address of Storage key for *tenth* 2,048 main memory block		Must be zeros

Set Storage Key (SSK)

General Description

◆ The storage key of a 2,048-byte main memory block located at the address contained in the general register specified by the second address (R_2) is set according to the value contained in the register specified by the first address (R_1).

Format (RR)

08	R_1	R_2
0 7	8 11	12 15

Condition Code

◆ Unchanged.

Interrupt Action

◆ Privileged operation.

Address error:

Addressing.

Specification.

Operation code trap (if the memory protect feature is not installed).

Notes

◆ 1. Bits 8 through 20 of the register specified by the second address (R_2) contain the location of the 2,048-byte main memory block where storage key is to be set. Bits 0 through 7 and 21 through 27 are ignored. Bits 28 through 31 must be zero.

2. Bits 24 through 28 of the general register specified by the first address (R_1) contain the five-bit storage key to be assigned. Bits 0 through 23 and 29 through 31 are ignored.

3. The address of the storage key for a specific 2,048-byte main memory block is specified in R_2 by a binary count (see examples under Insert Storage Key description).

Write Direct (WRD)

General Description

◆ The eight-bit byte specified by the first address (B_1/D_1) is accessed and transmitted to all units via the Static Out lines. The eight-bit I field specifies the Signal Out lines to be pulsed. The Static Out lines remain as specified until the next Write Direct instruction.

Format (SI)

84	I_2	B_1	D_1
0 7	8 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Privileged operation.

Address error:

Addressing.

Operation code trap (if Direct Control option is not installed).

Notes

1. Each trunk has only one Signal Out line and is pulsed according to the following pattern:

I-Field	*Trunk(s) Pulsed*
Bit 0 = 1	Six
Bit 1 = 1	Five
Bit 2 = 1	Four
Bit 3 = 1	Three
Bit 4 = 1	Two
Bit 5 = 1	One
Bit 6 = 0	Reserved (Must be zero)
Bit 7 = 0	Reserved (Must be zero)

More than one I-Field bit may be set to 1 providing pulses for sending over more than one direct control trunk. This permits sending the same byte to all processors connected to the transmitting processor.

2. A processor cannot Write Direct to itself. The I-Field bit associated with the transmitting processor must always be reset to zero. (This is a programming restriction.)

3. An I field of all zeros causes the byte specified by the first address to be placed on *all* trunks but does not cause an interrupt to occur in the other connected processors. This byte can be read by a Read Direct instruction.

Read Direct (RDD)

General Description

◆ The eight-bit I field specifies one of up to five possible sets of Direct Control trunks to be sampled. The sampled eight-bit byte is transferred to the main memory location specified by the first address (B_1/D_1) from the Static In lines.

Format (SI)

85	I_2	B_1	D_1
0 7	8 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Privileged operation.

Address error:

Addressing.

Protection.

Operation code trap (if Direct Control option is not installed).

Notes

◆ 1. Each of the six Direct Control trunks has a set of Direct In lines which are sampled according to the following pattern:

I-Field	*Trunk Sampled*
Bit 0 = 1	Six
Bit 1 = 1	Five
Bit 2 = 1	Four
Bit 3 = 1	Three
Bit 4 = 1	Two
Bit 5 = 1	One
Bit 6 = 0	Unused (Must be zero)
Bit 7 = 0	Unused (Must be zero)

The program must specify only one I-Field bit set to 1, otherwise results of the instruction are unpredictable.

2. A processor cannot Read Direct to itself. The I-Field bit associated with the receiving processor must always be reset to zero. (This is a programming restriction.)

3. This instruction may be prolonged by the presence of a HOLD signal. If so, timer updating may be skipped. However, I/O servicing will not be affected.

PROCESSOR STATE CONTROL INSTRUCTIONS

INTRODUCTION

◆ There are two control instructions that can be used in the *Processing State (P_1)*. These instructions are Supervisor Call, and Set Program Mask. These instructions can also be executed in any other state.

The Supervisor Call instruction enables the program to switch from any state to the *Interrupt Control State (P_3)*. Through this operation a program in any processor state can communicate with and initiate the *Interrupt Control State (P_3)* programs.

The Set Program Mask instruction permits the user to specify whether or not the program is to be interrupted for any of the following errors:

1. significance error.
2. exponent underflow.
3. decimal overflow.
4. fixed-point overflow.

The execution of the Set Program Mask instruction causes the condition code and program mask bits in the P counter of the state in which the system is operating to be set to the value specified by the instruction. This instruction always changes the condition code.

INSTRUCTION FORMAT

RR Format

Op Code	R_1	R_2
0 7	8 11	12 15

Description

◆ The RR format is used for the Supervisor Call and Set Program Mask instructions. For the Set Program Mask instruction, the R_2 field is ignored. The contents of the general register specified by the R_1 field form the first operand.

For the Supervisor Call instruction, the R_1 and R_2 fields are combined to become an immediate operand. This operand does not refer to any register, but is a value which is placed in the Interrupt Status Register (ISR) of the initiated state to provide communication with the software in this state.

CONDITION CODE UTILIZATION

◆ The condition code is changed by the Set Program Mask instruction. The condition code and program mask bits of the current P counter are replaced by the contents of the general register (bits 2-7) specified by the first address of the instruction.

INTERRUPT ACTION

◆ No error interrupts can occur as a result of using the instructions in this section. The Supervisor Call instruction causes an interrupt, but this interrupt is the desired result of its execution.

Supervisor Call (SVC)

General Description

◆ The R_1 and R_2 fields provide an interruption code and this code is placed into the rightmost byte of the Interrupt Status Register (ISR) of the program state in which this instruction is issued. The supervisor call interrupt flag bit (priority 21) is set in the Interrupt Flag register and a program interrupt may occur depending on the associated mask bit in the Interrupt Mask register of the current state.

Format (RR)

OA	R_1	R_2
0 7	8 11	12 15

Condition Code

◆ Unchanged.

Interrupt Action

◆ None.

Note

◆ If a higher priority interrupt is honored upon executing this instruction, the flag bit (priority 21) will be set and the Supervisor Call byte stored in the ISR so that when it is honored, the results are independent of any higher priority interrupts.

Set Program Mask (SPM)

General Description

◆ Bits 2-7 of the general register specified by the first address (R_1) establish new program masks and condition code setting for the current program state.

Format (RR)

04		R_1			
0	7	8	11	12	15

Condition Code

◆ The condition code is set according to bits 2 and 3 of the general register specified by R_1 as follows:

Condition Code Setting

2	3	Result
0	0	Set condition code 0 (zero).
0	1	Set condition code 1.
1	0	Set condition code 2.
1	1	Set condition code 3.

Program Mask

◆ The program mask is set according to bits 4-7 of the general register specified by R_1 as follows:

Program Mask Setting

Bit	Result
4	Fixed-point overflow.
5	Decimal overflow.
6	Exponent underflow.
7	Significance error.

Note

◆ The contents of the P-counter and the register specified by the first address are unaltered.

FIXED-POINT INSTRUCTIONS

INTRODUCTION

◆ Using fixed-point instructions, binary arithmetic is performed on operands used as addresses, index quantities, counts, and fixed-point data. Generally, the operands involved are 32 bits long and signed. One of the general registers always holds one operand. The other operand is in either main memory or in a general register. Negative quantities are in the two's-complement form.

This instruction set performs the following functions:

1. loading.
2. storing.
3. comparing.
4. shifting.
5. sign control.
6. radix conversion of fixed-point operands.
7. adding.
8. subtracting.
9. multiplying.
10. dividing.

The result of all sign control, compare, shift, add, and subtract operations is reflected in the condition code.

DATA FORMAT

◆ A fixed-length format of a one-bit sign followed by the integer field makes up fixed-point numbers. In one of the general registers, the number is a 31-bit integer field. The complete 32-bit register is occupied by the fixed-point quantity and sign. A 64-bit operand, with a 63-bit integer field, is used by some shift, multiply, and divide instructions. A pair of adjacent registers, addressed by the even address of the leftmost register, contains these longer operands. The sign-bit of the rightmost register becomes part of the integer field. The same register can be specified for both operands in register-to-register operations (except for the Divide instructions). In main memory, fixed-point operands are in either a 32-bit word or a 16-bit halfword. The integer fields are then either 31 bits or 15 bits. Radix conversion operations always use a 64-bit decimal field. Integral storage boundaries for these units of data must be observed. Halfword, full-word, or double-word operands are addressed with one, two, or three low-order address bits set to zero. Half-word operands are extended to full words when they are fetched from main memory and used as a full-word operand.

Halfword Fixed-Point Number

SIGN	15-bit Integer	
0	1	15

Full-word Fixed-Point Number

SIGN	31-bit Integer	
0	1	31

REPRESENTATION OF NUMBERS

◆ All fixed-point operands are treated as signed integers. True binary notation with a sign bit of zero is the representation of positive numbers. Two's-complement notation with a sign bit of one is the representation of negative numbers. To obtain the two's complement of a number, the value of each bit is changed and a one is added to the low-order bit.

REPRESENTATION OF NUMBERS (Cont'd)

This number representation can be regarded as the low-order part of an infinitely long representation of the number. A positive number has all zero bits, including the sign, to the left of the most significant bit of the number. A negative number has all one bits, including the sign, to the left of the most significant bit of the number. When an operand is to be extended with high-order bits, the extension is made by prefixing the operand with bits equal to the high-order bit of the operand.

A negative zero is not included in two's-complement notation. In the number range, the set of positive numbers is one less than the set of negative numbers. The *maximum* negative number is made up of an all-zero integer field with a one-bit sign. The maximum positive number consists of all 1's in the integer field with a zero-bit sign. The complement of the maximum negative number cannot be represented in the processor. For example, on a subtraction from zero that produces the complement of the maximum negative number, a fixed-point overflow exception is noted and the number remains unchanged. If the final result is within the representable range, then an overflow does not result (such as a subtraction from minus one). The representation of the product of two maximum negative numbers is a double-length positive number.

An overflow carries into the leftmost bit, which is the sign, and changes it. In algebraic shifting, however, the sign bit is unchanged even when significant bits in a shift left instruction are shifted out.

INSTRUCTION FORMATS

◆ The following three formats (RS, RX, RR) are used for fixed-point operations:

RS Format

Op Code	R_1	R_3	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Description

◆ An address is formed by adding the contents of the general register specified by B_2 to the displacement of field D_2. The address formed is that of the main memory location of the second operand in the Load and Store Multiple instructions. In the shift operations, the result formed designates the amount of shift. The R_1 and R_3 fields specify the general register boundaries for Load and for Store Multiple instructions. In shift operations, R_1 specifies the general register holding the first operand, and R_3 is ignored.

RX Format

Op Code	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Description

◆ An address is formed by adding the contents of general registers specified by the X_2 and B_2 fields to the displacement field D_2. This address specifies the main memory location of the second operand in the operation. The R_1 field designates the general register containing the first operand.

RR Format

Op Code	R_1	R_2
0 7	8 11	12 15

Description

◆ In this format, the R_1 field specifies the general register holding the first operand. The R_2 field specifies the general register holding the second operand. The same register can be specified for both operands.

Notes

◆ 1. A zero in an X_2 or B_2 field indicates there is no corresponding address component to enter in the forming of an address in either the RX or RS format.

2. Except for the instructions Store and Convert to Decimal, results of fixed-point operations replace the first operand.

3. Except for storing the result, the contents of general registers and main memory locations used in the operations are not changed.

4. It is possible to designate the same general register both for operand locations and for address modification. Address modification occurs prior to operation execution.

CONDITION CODE UTILIZATION

◆ The condition code indicates the results of fixed-point sign control, add, subtract, shift, and compare instructions. The code is not changed by any other fixed-point instruction. Decision making by branch on condition operations can be done after those instructions which set the code.

For most arithmetic instructions, the Condition Codes 0, 1, or 2 indicate respectively a zero, less than zero, or greater than zero result. Condition Code 3 is set for overflow result. In comparison instructions, the Condition Codes 0, 1, or 2 indicate that the first operand is equal to, less than, or greater than the second operand. In add and subtract logical instructions, the Condition Codes 2 and 3 indicate either a zero or non-zero result with a carry from the sign bit. The Condition Codes 0 and 1 indicate the same conditions with no carry out of the sign position. Instructions that cause the condition code to be set and the meaning of the setting are as follows:

Instruction	Condition Code Setting			
	0	1	2	3
Add Word	Zero	< Zero	> Zero	Overflow
Add Halfword	Zero	< Zero	> Zero	Overflow
Add Logical	Zero	Not Zero	Zero Carry	Carry
Compare Word	Equal	Low	High	——
Compare Halfword	Equal	Low	High	——
Load and Test	Zero	< Zero	> Zero	——
Load Complement	Zero	< Zero	> Zero	Overflow
Load Negative	Zero	< Zero	——	——
Load Positive	Zero	——	> Zero	Overflow
Shift Left Double	Zero	< Zero	> Zero	Overflow
Shift Left Single	Zero	< Zero	> Zero	Overflow
Shift Right Double	Zero	< Zero	> Zero	——
Shift Right Single	Zero	< Zero	> Zero	——
Subtract Word	Zero	< Zero	> Zero	Overflow
Subtract Halfword	Zero	< Zero	> Zero	Overflow
Subtract Logical	——	Not Zero	Zero Carry	Carry

INTERRUPT ACTION

◆ The following interrupt conditions can occur as a result of fixed-point instructions:

Address Error

Addressing

◆ An address error interrupt occurs when an address specifies a location outside the available main memory. The operation is terminated at the point of error. The result data and the condition code, if produced, are unpredictable.

Specification

◆ An address error interrupt occurs when an instruction specifies a:

1. Full-word operand that is not located on a 32-bit boundary.
2. Halfword operand that is not located on a 16-bit boundary.
3. Double-word operand that is not located on a 64-bit boundary.
4. Register with an odd-numbered address when using an even/odd pair containing a 64-bit operand.

The instruction is suppressed. The condition code, data in main memory, and registers remain unchanged.

Protection

◆ An address error interrupt occurs when the storage key and the protection key of the result location do not match. The operation is suppressed and the condition code and data in the registers and main memory are unaltered. The only exception is the Store Multiple instruction which is terminated. The amount of data stored is unpredictable. (This interrupt can only occur if the memory protect feature is installed.)

Data Error

◆ A data error interrupt occurs when an invalid digit or sign code of the decimal operand is encountered in the Convert to Binary instruction. The operation is suppressed and the condition code and data in the register and main memory are unaltered.

Fixed-Point Overflow

◆ A fixed-point overflow interrupt occurs when the results overflow in sign control, add, subtract or shift operations. The operation is completed by placing the truncated result in the register and setting Condition Code 3. Overflow bits are lost. If the fixed point program mask bit is reset, interrupt will not occur and the flag in the IFR will not be set.

Divide Error

◆ A divide error interrupt occurs when the quotient would exceed the register size in division, or the result of a Convert to Binary instruction exceeds 31 bits. The operation is suppressed and the data in the registers remains unaltered.

Load Word (LR) (L)

General Description

◆ The operand specified by the second address (R_2 or $X_2/B_2/D_2$) is loaded into the general register specified by the first address (R_1).

Format (RR)

(LR) 18	R_1	R_2
0 7	8 11	12 15

(RX)

(L) 58	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing (RX format).

Specification (RX format).

Note

◆ The operand in the register or main memory location specified by the second address remains unchanged.

Load Halfword (LH)

General Description

◆ The halfword operand in the main memory specified by the second address ($X_2/B_2/D_2$) is loaded into the general register specified by the first address (R_1).

Format (RX)

48	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing.

Specification.

Notes

◆ 1. When the halfword (second operand) is fetched from main memory, it is expanded to a full word by propagating the sign-bit value through the 16 high-order positions of the receiving register.

2. The operand specified by the second address is unaltered.

Load and Test (LTR)

General Description

◆ The operand in the register specified by the second address (R_2) is loaded into the general register specified by the first address (R_1). The condition code is determined by the magnitude and the sign of the loaded operand.

Format (RR)

12	R_1	R_2
0 7	8 11	12 15

Condition Code

◆ 0 — result is zero.

1 — result is less than zero.

2 — result is greater than zero.

3 — not used.

Interrupt Action

◆ None.

Notes

◆ 1. The same register can be specified for both R_1 and R_2. If this is done, the operation is equivalent to a test with no data movement.

2. The operand specified by the second address (R_2) is unaltered.

Load Complement (LCR)

General Description

◆ The two's complement of the operand in the register specified by the second address (R_2) is loaded into the general register specified by the first address (R_1). The condition code is determined by the magnitude and the sign of the loaded operand.

Format (RR)

13	R_1	R_2
0 7	8 11	12 15

Condition Code

◆ 0 — result is zero.

1 — result is less than zero.

2 — result is greater than zero.

3 — overflow.

Interrupt Action

◆ Fixed-point overflow.

Notes

◆ 1. Zero operands remain constant and unchanged under complementation.

2. A fixed-point overflow interrupt occurs when the maximum negative number is complemented.

3. The operand specified by the second address is unaltered.

Load Positive (LPR)

General Description

◆ The operand in the register specified by the second address (R_2) is made positive, if negative, and loaded into the general register specified by the first address (R_1). In loading the absolute value of the operand, negative numbers are complemented and positive numbers remain unaltered. The magnitude of the absolute value determines the condition code.

Format (RR)

10	R_1	R_2
0 7	8 11	12 15

Condition Code

◆ 0 — result is zero.

1 — not used.

2 — result greater than zero.

3 — overflow on complement.

Interrupt Action

◆ Fixed-point overflow.

Notes

◆ 1. A fixed-point overflow interrupt exists if a maximum negative number is complemented.

2. The operand specified by the second address is unaltered.

Load Negative (LNR)

General Description

◆ The two's complement of the operand in the register specified by the second address (R_2) is loaded into the general register specified by the first address (R_1). In loading the operand value, positive numbers are complemented and negative numbers remain unaltered. The magnitude of the loaded value determines the condition code setting.

Format (RR)

11	R_1	R_2
0 7	8 11	12 15

Condition Code

◆ 0 — result is zero.

1 — result is less than zero.

2 — not used.

3 — not used.

Interrupt Action

◆ None.

Notes

◆ 1. A zero operand is not altered and retains a positive sign.

2. The operand specified by the second address is unaltered.

Load Multiple (LM)

General Description

◆ The set of general registers, beginning with the register specified by the first address (R_1) and ending with the register specified by the third address (R_3), is loaded with operands from main memory. The second address (B_2/D_2) specifies the main memory location of the first word to be loaded. Loading of the general registers continues in the ascending order of their addresses beginning with the register specified by R_1. As many words as needed are fetched from the main memory location specified, continuing up to, and including, the register specified by R_3.

Format (RS)

98	R_1	R_3	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing.

Specification.

Notes

◆ 1. If R_1 and R_3 specify the same register, only one word is loaded.

2. If the register specified by R_3 is less than the register specified by R_1, wrap-around occurs from register 15 to 0.

3. The operands specified by the second address are unaltered.

Add Word (AR) (A)

General Description

♦ The operands specified by the first and second addresses (R_1 and R_2 or $X_2/B_2/D_2$) are added and the sum is placed in the general register specified by the first address (R_1). The magnitude and the sign of the sum determine the condition code setting.

Format (RR)

(AR) 1A	R_1	R_2
0 7	8 11	12 15

(RX)

(A) 5A	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

♦ 0 — sum is zero.

1 — sum is less than zero.

2 — sum is greater than zero.

3 — overflow.

Interrupt Action

♦ Fixed-point overflow.

Address error:

Addressing (RX format).

Specification (RX format).

Notes

♦ 1. All 32 bits of both operands participate in the addition. If the carries into and out of the sign bit disagree, an overflow exists. The overflow does not alter the sign bit created by the carries.

2. A negative overflow results in a positive sum and a positive overflow results in a negative sum with overflow bits being lost.

3. A zero result is always positive.

4. The operand specified by the second address is unaltered.

Add Halfword (AH)

General Description

◆ The halfword operand specified by the second address ($X_2/B_2/D_2$) is added to the operand specified by the first address (R_1) and the sum is placed into the register specified by the first address (R_1). The sign and the magnitude of the sum determine the condition code setting.

Format (RX)

4A	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ 0 — sum is zero.

1 — sum is less than zero.

2 — sum is greater than zero.

3 — overflow

Interrupt Action

◆ Fixed-point overflow.

◆ Address error:

Addressing.

Specification.

Notes

◆ 1. The halfword in main memory specified by the second address is expanded to full-word length prior to the addition by propagating the sign bit value through the high-order 16 positions. The addition is completed by adding all 32 bits of both operands.

2. An overflow exists if the high-order numeric result bit and the carry out of the sign-bit position disagree. The sign is not corrected after overflow occurs. A negative overflow results in a positive sum and a positive overflow results in a negative sum with the overflow bits being lost.

3. The operand specified by the second address is unaltered.

Add Logical (ALR) (AL)

General Description

◆ The operand specified by the second address (R_2 or $X_2/B_2/D_2$) is logically added (32-bit unsigned) to the operand specified by the first address (R_1). The sum is placed in the general register specified by the first address. The condition code is determined by the relation of the sum to a zero number and the occurrence of a carry out of the sign bit position. An overflow on such carries is not recognized and does not set an interrupt condition.

Format (RR)

(ALR) 1E	R_1	R_2
0 7	8 11	12 15

(RX)

(AL) 5E	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ 0 — sum is zero and no carry.
1 — sum is not zero and no carry.
2 — sum is zero with a carry.
3 — sum is not zero with a carry.

Interrupt Action

◆ Address error:
Addressing (RX format).
Specification (RX format).

Notes

◆ 1. All 32 bits of the operands participate in the logical addition.
2. The operand specified by the second address is unaltered.

Subtract Word (SR) (S)

General Description

◆ The operand specified by the second address (R_2 or $X_2/B_2/D_2$) is subtracted from the operand specified by the first address (R_1) and the difference is placed in the general register specified by the first address (R_1). The magnitude and the sign of the difference determine the condition code setting.

Format (RR)

(SR) 1B	R_1	R_2
0 7	8 11	12 15

(RX)

(S) 5B	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ 0 — difference is zero.

1 — difference is less than zero.

2 — difference is greater than zero.

3 — overflow.

Interrupt Action

◆ Fixed-point overflow.

Address error:

Addressing (RX format).

Specification (RX format).

Notes

◆ 1. The operation is accomplished by adding the one's complement of the second operand and a one in the low-order position of the first operand. The one's complement of a number is obtained by changing all the 1 bits to 0 bits and all the 0 bits to 1 bits. All 32 bits are involved in the operation. An overflow exists if the high-order numeric result bit and the carry out of the sign bit position disagree.

2. The difference between a maximum negative number and another maximum negative number is zero with no overflow.

3. When the same register is specified for R_1 and R_2, the operation is equivalent to clearing R_1 to zero.

4. The operand specified by the second address is unaltered.

Subtract Halfword (SH)

General Description

◆ The halfword operand specified by the second address ($X_2/B_2/D_2$) is expanded and subtracted from the operand specified by the first address (R_1). The difference is placed in the general register specified by R_1. The sign and the magnitude of the difference determine the condition code setting.

Format (RX)

4B	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ 0 — difference is zero.

1 — difference is less than zero.

2 — difference is greater than zero.

3 — overflow.

Interrupt Action

◆ Fixed-point overflow.

Address error:

Addressing.

Specification.

Notes

◆ 1. The halfword in main memory specified by the second address is expanded to full-word length by propagating the sign bit value through the 16 high-order positions.

2. The subtraction is completed by adding the one's complement of the second operand and a one in the low-order position of the first operand. All 32 bits are involved in the operation.

3. An overflow exists if the high-order numeric result bit and the carry out of the sign bit position disagree.

4. The operand specified by the second address is unaltered.

Subtract Logical (SLR) (SL)

General Description

◆ The operand specified by the second address (R_2 or $X_2/B_2/D_2$) is logically subtracted (32-bit unsigned) from the operand specified by the first address (R_1). The difference is placed in the general register specified by the first address. The condition code is determined by the relation of the sum to a zero number and the occurrence of a carry out of the sign bit position. An overflow on such carries is not recognized and does not set an interrupt condition.

Format (RR)

(SLR) 1F	R_1	R_2
0 7	8 11	12 15

(RX)

(SL) 5F	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ 0 — not used.

1 — difference is not zero and no carry.

2 — difference is zero with a carry.

3 — difference is not zero with a carry.

Interrupt Action

◆ Address error:

Addressing (RX format).

Specification (RX format).

Notes

◆ 1. Logical subtraction is accomplished by adding the one's complement of the second operand and a one in the low-order position of the first operand.

2. All 32 bits of the operands participate in the logical subtraction without change to the resulting sign bit.

3. The operand specified by the second address is unaltered.

Compare Word (CR) (C)

General Description

◆ The operand specified by the first address (R_1) is compared with the operand specified by the second address (R_2 or $X_2/B_2/D_2$). Both operands remain unaltered. The result of the comparison determines the condition code setting.

Format (RR)

(CR) 19	R_1	R_2
0 7	8 11	12 15

(RX)

(C) 59	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ 0 — operands are equal.

1 — the operand specified by the first address is low.

2 — the operand specified by the first address is high.

3 — not used.

Interrupt Action

◆ Address error:

Addressing (RX format).

Specification (RX format).

Note

◆ Both operands are considered as 32-bit signed integers and the comparison is algebraic.

Compare Halfword (CH)

General Description

◆ The operand specified by the first address (R_1) is compared with the halfword operand expanded to a full word, specified by the second address ($X_2/B_2/D_2$). Both operands remain unaltered. The result of the comparison determines the condition code setting.

Format (RX)

49	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ 0 — operands are equal.

1 — the operand specified by the first address is low.

2 — the operand specified by the first address is high.

3 — not used.

Interrupt Action

◆ Address error:

Addressing.

Specification.

Notes

◆ 1. The halfword in storage specified by the second address is expanded to full-word length by propagating the sign bit value through the 16 high-order positions.

2. Both operands are considered as 32-bit signed integers and the comparison is algebraic.

Multiply Word (MR) (M)

General Description

◆ The operand (multiplicand) specified by the first address (R_1) is multiplied by the operand (multiplier) specified by the second address (R_2 or $X_2/B_2/D_2$). The double-length product is loaded into the register specified by the first address (R_1), which must be an even number, and the next odd-numbered register.

Format (RR)

(MR) 1C	R_1	R_2
0 7	8 11	12 15

(RX)

(M) 5C	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing (RX format).

Specification.

Notes

◆ 1. The first address (R_1) must always refer to the even-numbered register of an even/odd pair. The multiplicand is taken from the odd-numbered register of the pair. The original contents of the even-numbered register, which is replaced by the product, is ignored. An overflow cannot occur.

2. Only when two maximum negative numbers are multiplied does the product exceed 62 significant bits. This product produces 63 significant bits.

3. In two's-complement notation, the sign bit is propagated right, up to the first significant product bit.

4. The sign of the product is determined algebraically. A zero result is always positive.

5. The least significant digit of the product goes into the odd-numbered register.

6. The operand specified by the second address (multiplier) is unaltered except when the first and second addresses specify the same (even numbered) register. In this case the multiplier is taken from the even register, the multiplicand is taken from the odd register and the product is placed into the even/odd pair.

Multiply Halfword (MH)

General Description

◆ The operand (multiplicand) specified by the first address (R_1) is multiplied by the halfword operand (multiplier) specified by the second address ($X_2/B_2/D_2$). The product of the operands replaces the contents of the register specified by the first address (R_1)

Format (RX)

4C	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing.

Specification.

Notes

◆ 1. The halfword operand in main memory is expanded to a full word before multiplication by propagating the sign bit value through the 16 high-order positions. Both operands are considered as 32-bit signed integers. The multiplicand is replaced by the low order 32 bits of the product. The product usually occupies 46 bits of significance except when both operands are maximum negative numbers and occupy 47 bits.

2. The bits to the left of the 32 low-order bits of the product are not tested for significance. No overflow indication is given. Since the bits to the left of the low-order 32 are ignored, the sign of the result may differ from the true sign of the product, if the product exceeds 32 bits.

3. The operand specified by the second address is unaltered.

4. A zero product is always positive.

Divide (DR) (D)

General Description

◆ The double-word operand (dividend) specified by the first address (R_1) is divided by the operand (divisor) specified by the second address (R_2 or $X_2/B_2/D_2$). The quotient and remainder replace the double-word operand in the registers specified by the first address (R_1). The register specified by the first address must be the even-numbered register of an even/odd pair.

Format (RR)

(DR) 1D	R_1	R_2
0 7	8 11	12 15

(RX)

(D) 5D	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing (RX format).

Specification.

Divide Error.

Notes

◆ 1. The dividend, a 64-bit signed integer, is replaced by a 32-bit signed quotient and a 32-bit signed remainder; the remainder is placed in the even-numbered register and the quotient is placed in the odd-numbered register. The divisor is a 32-bit signed integer and is unaltered.

2. A divide error interrupt occurs when the magnitude of the dividend to the divisor is such that the quotient cannot be expressed by a 32-bit signed integer. (The divisor must be greater in absolute value than the first word of the dividend.)

3. The sign of the quotient is determined algebraically except that a zero quotient as a zero remainder is always positive.

4. The remainder has the same sign as the dividend.

Convert to Binary (CVB)

General Description

◆ The radix of the double-word operand in main memory specified by the second address ($X_2/B_2/D_2$) is converted from decimal to binary notation and loaded into the general register specified by the first address (R_1). The operand in main memory is treated as a right-justified signed integer before and after the conversion.

Format (RX)

4F	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing.

Specification.

Data error.

Divide error.

Notes

◆ 1. The double-word operand in main memory (15 digits plus sign) must be in the packed decimal format. The operand is checked for valid sign and digit codes. The sign representation depends on the current decimal code (USASCII or EBCDIC).

2. The maximum decimal number that can be converted and still be contained in a 32-bit register is $(2{,}147{,}483{,}647)_{10}$ positive and $(2{,}147{,}483{,}648)_{10}$ negative. A larger decimal number causes a divide error interrupt.

3. Negative decimal zero is converted to positive binary zero.

4. The operand specified by the second address remains unaltered in main memory.

Convert to Decimal (CVD)

General Description

◆ The radix of the operand specified by the first address (R_1) is converted from binary to decimal notation and stored at the double-word main memory area specified by the second address ($X_2/B_2/D_2$). The operand is treated as a right-justified signed integer before and after the conversion.

Format (RX)

4E	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing.

Specification.

Protection.

Notes

◆ 1. The result is placed in the double-word main memory location in the packed decimal format of 15 digits plus sign.

2. The low-order four bits of the result are the sign which is generated according to the current decimal code, EBCDIC or USASCII.

3. The maximum binary number (32-bit signed integer) that can be converted is (2,147,483,647) positive and (2,147,483,648) negative. No overflow can occur.

Store Word (ST)

General Description

◆ The operand in the general register specified by the first address (R_1) is stored in the main memory location specified by the second address ($X_2/B_2/D_2$).

Format (RX)

50	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing.

Specification.

Protection.

Notes

◆ 1. The complete contents (32 bits) of the general register specified by the first address are placed unaltered in main memory.

2. The operand specified by the first address is unaltered.

Store Halfword (STH)

General Description

◆ The rightmost half (16 bits) of the operand in the general register specified by the first address (R_1) is stored unaltered in the halfword main memory location specified by the second address ($X_2/B_2/D_2$).

Format (RX)

40	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing.

Specification.

Protection.

Notes

◆ 1. The 16 high-order bits of the operand specified by the first address field are ignored by the operation.

2. The operand specified by the first address is unaltered.

Store Multiple (STM)

General Description

◆ The operands in the set of general registers, beginning with the register specified by the first address (R_1) and ending with the register specified by the third address (R_3), are stored in main memory locations starting with the location specified by the second address (B_2/D_2). The second address (B_2/D_2) refers to the main memory location where the first operand (word) is to be stored. Storing of the operands continues in the ascending order of the register number specified by R_1, up to and including R_3, storing as many words as indicated in the main memory locations that immediately follow the initial operand.

Format (RS)

90	R_1	R_3	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:
Addressing.
Specification.
Protection.

Notes

◆ 1. If the same register is specified for R_1 and R_3, only one word is stored.

2. If R_3 is less than R_1, the register addresses wrap around from 15 to 0. For instance, all registers can be stored by making R_3 one less than R_1.

3. The operands in the set of registers designated are unaltered.

Shift Left Single (SLA)

General Description

◆ The integer portion of the operand in the general register specified by the first address (R_1) is shifted left the number of positions specified by the second address (B_2/D_2). The second address is used as a count and not to address data. The low-order six bits of the second address constitute the count. The remaining bits are ignored.

Format (RS)

8B	R_1		B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ 0 — result is zero.
1 — result is less than zero.
2 — result is greater than zero.
3 — overflow.

Interrupt Action

◆ Fixed-point overflow.

Notes

◆ 1. All 31 bit positions of the integer are shifted. The sign is not altered. Zeros are inserted in the right-hand end of the operand for each shift.

2. If a bit is shifted out of the left-hand end that is not identical to the sign bit, a fixed-point overflow condition exists.

Shift Right Single (SRA)

General Description

◆ The integer portion of the operand in the general register specified by the first address (R_1) is shifted right the number of positions specified by the second address (B_2/D_2). The second address is used as a count and not to address data. The low-order six bits of the second address field constitute the count. The remaining bits are ignored.

Format (RS)

8A	R_1		B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ 0 — result is zero.

1 — result is less than zero.

2 — result is greater than zero.

3 — not used.

Interrupt Action

◆ None.

Notes

◆ 1. All 31 bit positions of the integer are shifted. The sign is not altered. The sign bit is propagated through the positions vacated in the left end of the operand. The bits shifted out to the right are lost.

2. Shifting to the right is equivalent to low-order truncation or division by powers of two.

3. Shifts greater than 31 cause all significant bits to be lost. A zero for positive numbers and a minus one for negative numbers is the result of such shifts.

4. Fixed-point positive numbers go towards zero; Fixed-point negative numbers go towards minus one.

Shift Left Double (SLDA)

General Description

◆ The integer portion of the double-word operand specified by the first address (R_1) and the first address plus one is shifted left the number of positions specified by the second address (B_2/D_2). The first address (R_1) specifies an even-numbered register of an even/odd pair that contains the 63-bit integer to be shifted. The second address is used as a count and not to address data. The low-order six bits of the second address field constitute the count. The remaining bits are ignored.

Format (RS)

8F	R_1		B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ 0 — result is zero.

1 — result is less than zero.

2 — result is greater than zero.

3 — overflow.

Interrupt Action

◆ Fixed-point overflow.

Address error:

Specification.

Notes

◆ 1. All 63 bit positions of the integer are shifted. The sign bit (position 0) in the even register is not altered. Zeros are inserted in the right-hand end of the double-word operand for each shift.

2. If a bit is shifted out of the left-hand end that is not identical to the sign bit, a fixed-point overflow condition exists.

Shift Right Double (SRDA)

General Description

◆ The integer portion of the double-word operand specified by the first address (R_1) and the first address plus one is shifted right the number of positions specified by the second address (B_2/D_2). The first address (R_1) specifies an even-numbered register of an even/odd pair that contains the 63-bit integer to be shifted. The second address is used as a count and not to address data. The low-order six bits of the second address constitute the count. The remaining bits are ignored.

Format (RS)

8E	R_1		B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ 0 — result is zero.

1 — result is less than zero.

2 — result is greater than zero.

3 — not used.

Interrupt Action

◆ Address error:

Specification.

Notes

◆ 1. All 63 bit positions of the integer are shifted. The sign bit in the leftmost position of the even-numbered register is not altered. This sign bit is propagated through the positions vacated in the left end of the double-word operand. The bits shifted out to the right are lost.

2. A shift count of zero provides a double-word sign and magnitude check.

DECIMAL ARITHMETIC INSTRUCTIONS

INTRODUCTION

◆ Decimal arithmetic is performed on data in packed format. In this format, two decimal digits are placed in one byte (four bits each). The operands may be variable in length, and must contain a sign in the rightmost four bits.

All decimal instructions are two-address, SS-type format. The instruction set includes addition, subtraction, comparison, multiplication, and division. Since data sent to, and from, external devices are usually in zoned (unpacked) format (one digit in one byte), there are also instructions for converting to, and from, packed and zoned format. All decimal arithmetic instructions are standard features of the 70/46 Processor.

DATA FORMATS

◆ The formats for decimal data in high-speed memory are:

Packed Format

Byte		Byte		Byte		Byte		Byte		Byte	
Digit	Digit	Digit	Digit	Digit	Digit	Digit	Digit	Digit	Digit	Digit	Sign

In packed format, one byte represents two decimal digits. The rightmost half-byte (4 bits) of a field represents the sign.

Zoned Format

Byte		Byte		Byte		Byte		Byte		Byte	
Zone	Digit	Zone	Digit	Zone	Digit	Zone	Digit	Zone	Digit	Sign	Digit

In zoned format, the low-order four bits of each eight-bit byte contain the decimal digit and the high-order four bits contain the zone. The high-order four bits of the rightmost byte of a field contain the sign of the field.

Description of Formats

◆ Decimal arithmetic instructions operate from right to left. The addresses specify the leftmost byte of the operand, and the length specifies the additional number of bytes that are to the right of the addressed byte. The fields specified by the addresses can be variable in length beginning at any byte in main memory and consisting of from 1 to 16 eight-bit bytes. Results of operations are always placed in the first operand field. The result never exceeds the limits set by the address and length of the first operand field. If a decimal arithmetic operation results in a carry outside the operand limits, a decimal overflow interrupt occurs. If the first operand is longer than the second, the second operand is extended with high-order zeros up to the length of the first operand during operation execution (in addition and subtraction only). This extension never changes main memory.

Because the code configurations of digits and sign are verified while arithmetic operations are performed, improper overlapping of fields is recognized as a data error. The arithmetic instruction set (except Pack, Unpack, Move with Offset) should not specify overlapping fields unless the rightmost byte of the fields coincide.

In the move-type instructions of this set (Pack, Unpack, Move with Offset), no checking is made for valid codes. Consequently, overlapping is permitted without any restrictions. (Although unusual results are possible, overlapping is dangerous.)

REPRESENTATION OF NUMBERS

◆ Decimal operands in packed format are four-bit, binary-coded, decimal digits packed two to a byte. The operands may be variable in length and must contain a sign in the rightmost four bits of the rightmost byte. The digit and sign codes are as follows:

Digit and Sign Codes

Digit	Code	Sign	Code
0	0000	+	1010
1	0001	−	1011
2	0010	+	1100
3	0011	−	1101
4	0100	+	1110
5	0101	+	1111
6	0110		
7	0111		
8	1000		
9	1001		

EBCDIC or USASCII sign or zone codes are generated for the decimal arithmetic results depending on the setting of the decimal code bit in the Interrupt Status Register. When the decimal code bit is set for EBCDIC, the following codes are generated:

Sign		Zone
Plus	Minus	
1100	1101	1111

When the decimal code bit is set for USASCII, the following codes are generated:

Sign		Zone
Plus	Minus	
1010	1011	0101

Note: The codes $(1110)_2$ and $(1111)_2$ are accepted as plus signs. However, if an arithmetic operation is performed on a field with these signs, the sign of the result will be in EBCDIC or USASCII, as shown above.

INSTRUCTION FORMAT

◆ Decimal arithmetic instructions use the two-address, SS format as follows:

SS Format

Op Code	L_1	L_2	B_1	D_1	B_2	D_2
0 7	8 11	12 15	16 19	20 31	32 35	36 47

Description

◆ The contents of the general register specified by B_1 are added to the contents of the displacement field (D_1) to obtain the main memory location of the leftmost byte of the first operand. The length (L_1) of the first address specifies the *number of bytes that are to the right* of the location obtained above, thus giving the processor the address of the rightmost byte of the first operand. The length of the operand can be from one to 16 bytes, since

Description (Cont'd)

L_1 can be from 0000 to 1111. The address and size of the second operand is obtained in the same way using B_2, D_2 and L_2.

Results of operations are always stored in the first operand field and never exceed the limits specified by the address and length. The second operand is not changed in an add-type instruction unless the second operand addresses the same rightmost byte as the first operand.

Note: A zero in the B_1 or B_2 field indicates that no general register is to be used.

CONDITION CODE UTILIZATION

◆ The condition code is set as a result of all add-type and comparison operations. No other decimal arithmetic instructions affect the condition code.

The condition code setting has a different meaning for the comparison operation result than for the add-type result. The results of the following decimal arithmetic instructions cause the indicated condition code settings:

Instruction	Condition Code Setting			
	0	1	2	3
Add Decimal	Zero	< Zero	> Zero	Overflow
Subtract Decimal	Zero	< Zero	> Zero	Overflow
Zero and Add	Zero	< Zero	> Zero	Overflow
Compare Decimal	Equal	Low	High	——

INTERRUPT ACTION

◆ The following interrupt conditions can occur as a result of a decimal arithmetic instruction.

Address Error

Addressing

◆ An address error interrupt exists when an address specifies a location outside the available main memory of the particular installation. The operation is terminated at the point of error. The result data and the condition code are unpredictable.

Specification

◆ An address error interrupt exists when a multiplier or divisor size exceeds 15 digits plus sign; or when the multiplier size or the divisor size is equal to, or greater than, the multiplicand or dividend size, respectively. The instruction is suppressed. The condition code, data in main memory, and registers remain unchanged.

Protection

◆ An address error interrupt exists when the protection key and the storage key of the result location do not match. The operation is terminated. The result data and condition code are unpredictable. (This interrupt can occur only if the memory protect feature is installed.)

Data Error

◆ A data error interrupt exists in decimal arithmetic when an invalid sign (not greater than nine) or digit code (not zero through nine) is detected in an operand, a multiplicand has insufficient high-order zeros, or there is incorrect overlapping of operands. The operation is terminated. The result data and the condition code setting are unpredictable.

Decimal Overflow

◆ A decimal overflow interrupt exists when the result field of an Add Decimal, Subtract Decimal, or Zero and Add instruction is too small to contain the overflow data. The operation is completed by ignoring the overflow data, and setting the condition code to 3. If the decimal overflow program mask bit is reset, interrupt will not occur and the flag in the IFR will not be set.

Divide Error

◆ A divide error interrupt occurs when the quotient is greater than the specified data field, including division by zero, or the dividend does not have one leading zero. Division is suppressed and the dividend and divisor remain unchanged in main memory.

Add Decimal (AP)

General Description

◆ The operand specified by the second address (B_2/D_2) is added algebraically to the operand specified by the first address (B_1/D_1). The result is stored in the field specified by the first address. The sign and the magnitude of the sum determine the condition code.

The operands can be variable in length up to 16 bytes and must be in packed format. If operands overlap, their rightmost byte location must coincide.

The addition of the two operands can cause decimal overflow. Two conditions which cause overflow are:

1. a carry out of the high-order position of the result.
2. a second operand that is larger than the first operand and significant result positions are lost.

Format (SS)

FA	L_1	L_2	B_1	D_1	B_2	D_2
0 7	8 11	12 15	16 19	20 31	32 35	36 47

Condition Code

◆ 0 — sum is zero.

1 — sum is less than zero.

2 — sum is greater than zero.

3 — overflow.

Interrupt Action

◆ Address error:

Addressing.

Protection.

Data error.

Decimal overflow.

Notes

◆ 1. High-order zeros are supplied for *either* operand during instruction execution.

2. All signs and digits are checked for validity.
3. The operand specified by the second address is unaltered.
4. Processing is from right to left.
5. A zero result is always positive except when high-order digits are lost because of overflow. In overflow, a zero result has the sign of the correct result.

Subtract Decimal (SP)

General Description

◆ The operand specified by the second address (B_2/D_2) is subtracted algebraically from the operand specified by the first address (B_1/D_1). The result is stored in the field specified by the first address. The sign and the magnitude of the difference determine the condition code.

The operands can be variable in length up to 16 bytes and must be in packed format. If operands overlap, their rightmost byte location must coincide.

The subtraction of two operands can cause decimal overflow.

Format (SS)

FB	L_1	L_2	B_1	D_1	B_2	D_2
0 7	8 11	12 15	16 19	20 31	32 35	36 47

Condition Code

◆ 0 — difference is zero.

1 — difference is less than zero.

2 — difference is greater than zero.

3— overflow.

Interrupt Action

◆ Address error:

Addressing.

Protection.

Data error.

Decimal overflow.

Notes

◆ 1. High-order zeros are supplied for *either* operand during instruction execution.

2. All signs and digits are checked for validity.

3. The operand specified by the second address is unaltered.

4. Processing is from right to left.

5. A zero difference is always positive except when high-order digits are lost because of overflow. In overflow, a zero result has the sign of the correct difference.

Zero and Add (ZAP)

General Description

◆ The operand specified by the second address (B_2/D_2) is loaded into the location specified by the first address (B_1/D_1). The operation is equivalent to an addition to zero and the result of the addition determines the condition code.

The operands may be variable in length up to 16 bytes and must be in packed format. High-order zeros are provided when necessary. Operands may overlap if their rightmost byte locations coincide, or if the rightmost byte of the first operand is to the right of the rightmost byte of the second operand.

A second operand that is longer than the first operand causes overflow.

Format (SS)

F8	L_1	L_2	B_1	D_1	B_2	D_2
0 7	8 11	12 15	16 19	20 31	32 35	36 47

Condition Code

◆ 0 — result is zero.

1 — result is less than zero.

2 — result is greater than zero.

3 — overflow.

Interrupt Action

◆ Address error:

Addressing.

Protection.

Data error.

Decimal overflow.

Notes

◆ 1. Only the second operand is checked for valid sign and digit codes.

2. The second operand is unaltered.

3. Processing is from right to left.

4. A zero result is positive except when high-order digits are lost because of overflow. In overflow, a zero result has the sign of the second operand.

Compare Decimal (CP)

General Description

◆ The operand specified by the first address (B_1/D_1) is algebraically compared with the operand specified by the second address (B_2/D_2). The results of the comparison determine the condition code.

The operands may be variable in length up to 16 bytes and must be in packed format. If the fields are unequal in length, the shorter is extended with high order zeros. If operands overlap, their rightmost byte location must be identical.

Overflow cannot occur as a result of this operation.

Format (SS)

F9	L_1	L_2	B_1	D_1	B_2	D_2
0 7	8 11	12 15	16 19	20 31	32 35	36 47

Condition Code

◆ 0 — the fields are numerically equal.

1 — the first operand is algebraically less than the second operand.

2 — the first operand is algebraically greater than the second operand.

Interrupt Action

◆ Address error:

Addressing.

Data error.

Notes

◆ 1. All signs and digits are checked for validity.

2. Both operands are unaltered.

3. Comparison is from right to left.

4. A positive zero compares equally to a negative zero.

Multiply Decimal (MP)

General Description

◆ The operand specified by the first address (multiplicand) is multiplied by the operand specified by the second address (multiplier). The product is stored in the location of the first operand, right-justified.

The operands may be variable in length and must be in packed format. Operands can overlap if their rightmost byte locations coincide.

The second operand (multiplier) must be shorter than the first operand (multiplicand) and must not exceed eight bytes in length (15 digits plus sign). Otherwise, an address error (specification) occurs.

The multiplicand must have high-order zero bytes equal to the number of bytes in the multiplier, field, or a data error occurs. The maximum product size is 31 digits.

Format (SS)

FC	L_1	L_2	B_1	D_1	B_2	D_2
0 7	8 11	12 15	16 19	20 31	32 35	36 47

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing.

Protection.

Specification.

Data error.

Notes

◆ 1. All signs and digits are checked for validity.

2. The second operand is unaltered unless operands overlap.

3. Overflow cannot occur.

4. The sign of the product is determined by the rules of algebra, even if one, or both, operands are zero; that is, minus zero is a possible result.

Divide Decimal (DP)

General Description

◆ The operand specified by the first address (the dividend) is divided by the operand specified by the second address (the divisor) and the result (quotient plus remainder) replaces the first operand. The quotient is placed leftmost in the first operand field. The remainder, which has a size equal to the divisor size, is placed rightmost in the first operand field.

The operands may be variable in length and must be in packed format. Overlapping is allowed if the rightmost byte locations are identical. The second operand (the divisor) must be shorter than the first operand (the dividend) and must not exceed eight bytes in length (15 digits plus sign). If either rule is not observed, an address error (specification) occurs.

The dividend must have at least one high-order zero. Otherwise, a data error occurs.

Together, the quotient and remainder occupy the entire dividend field after division. Therefore, the address of the quotient field is the address of the dividend field and its size in bytes is $L_1 - L_2$. The quotient and remainder are signed integers which are right-aligned in the first operand.

No overflow can occur. A quotient that is larger than the number of digits allowed causes a decimal divide error.

Format (SS)

FD	L_1	L_2	B_1	D_1	B_2	D_2
0 7	8 11	12 15	16 19	20 31	32 35	36 47

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing.

Protection.

Specification.

Data error.

Decimal divide error.

Notes

◆ 1. All signs and digits are checked for validity.

2. The second operand is unaltered.

3. The sign of the quotient is determined by the rules of algebra from dividend and divisor signs. The sign of the remainder has the same value as the dividend sign.

4. The first address plus $(L_1 - L_2)$ specifies the address of the remainder. The length of the remainder is specified by $L_2 + 1$.

Pack (PACK)

General Description

◆ The operand specified by the second address (B_2/D_2) is converted from zoned format to packed format and the result is placed in the location specified by the first address (B_1/D_1).

The operand specified by the second address must be in zoned format. The sign is obtained from the zone portion of the rightmost byte of the second operand and is placed in the rightmost four bits of the first operand (result field). All other zones are ignored. The four-bit numeric portions (stripping the four-bit zone) of each byte are then placed adjacent to the sign, and to each other, to fill the result field.

The result is extended with high-order zeros if the second operand field is shorter than the first. If the first operand field is not large enough to contain all the significant digits from the second operand field, the remaining digits are ignored. The operands may overlap.

Format (SS)

F2	L_1	L_2	B_1	D_1	B_2	D_2
0 7	8 11	12 15	16 19	20 31	32 35	36 47

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing.

Protection.

Notes

◆ 1. Signs and digits are not checked for validity.

2. The second operand is not changed except when the operands overlap.

3. Processing is from right to left, one byte at a time.

Unpack (UNPK)

General Description

◆ The operand specified by the second address (B_2/D_2) is converted from packed format to zoned format and the result is placed in the location specified by the first address (B_1/D_1).

Each of the eight-bit bytes of the packed, second-operand field represents two four-bit digits. Each of the four-bit digits is stored in a byte of the first operand field in the low-order four-bit positions. If the Decimal Code is EBCDIC, a zone code of 1111 is inserted into the high-order four bits of each byte. If the Decimal Code is USASCII, a zone code of 0101 is inserted. These zones are inserted in all but the zone portion of the rightmost byte, which receives the sign of the packed operand.

If the first operand is not large enough to receive the significant digits of the second operand, the remaining digits are ignored. The second-operand field is extended with zero digits before unpacking.

Format (SS)

F3	L_1	L_2	B_1	D_1	B_2	D_2
0 7	8 11	12 15	16 19	20 31	32 35	36 47

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing.

Protection.

Notes

◆ 1. Signs and digits are not checked for validity.

2. The second operand is not altered, except when operands overlap.

3. Processing is from right to left.

MOVE with OFFSET (MVO)

General Description

◆ The operand specified by the second address (B_2/D_2) is offset 4 bits to the left (a 1-digit left shift) and is placed to the left of, and adjacent to, the low-order four bits of the operand specified by the first address (B_1/D_1).

If the first operand is not large enough to receive all bytes of the second operand, the remaining bytes are ignored. If the second operand is shorter than the first operand, the second operand is extended with high-order zeros. The first and second operands may overlap.

Format (SS)

F1	L_1	L_2	B_1	D_1	B_2	D_2
0 7	8 11	12 15	16 19	20 31	32 35	36 47

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing.

Protection.

Notes

◆ 1. Signs and digits are not checked for validity.

2. The second operand is not changed except when operands overlap.

3. Processing is from right to left.

4. The initial low-order 4-bit digit of the operand specified by the first address is left unaltered.

LOGICAL INSTRUCTIONS

INTRODUCTION

◆ Logical instructions are used to manipulate data. The operands are usually treated as eight-bit bytes. Some logical operations require a single eight-bit byte specified as an operand; others may have variable-length operands composed of many eight-bit bytes. Some instructions operate on the zone portion only, or on the digit portion only, of the bytes of a variable-length operand. Some instructions have an operand that is part of the immediate instruction being executed. Finally, there is a group of instructions that provide for bit shifting.

Operands are in either main memory or general registers. Processing of data in main memory is from left-to-right starting at any byte location. Processing in general registers usually involves the entire contents of a general register, or in some cases, two general registers.

The Edit instruction is the only instruction which requires that the data be in packed decimal data. The Edit instruction converts packed decimal data into alphanumeric characters with editing under the control of a mask pattern.

The logical instruction set includes moving, comparing, bit testing, translating, editing, shifting, and bit connecting.

The condition code is set by all instructions except the moving, translating, and shifting instructions.

DATA FORMAT

◆ Data in general registers usually involves the entire 32 bits. There is no distinction made between sign and numeric bits. In some operations, only the least significant eight bits of the general register are involved, and in another case, the least significant 24 bits are involved. In addition, there are some shift operations in which an even/odd numbered pair of general registers is involved.

The storage data in memory-to-register operations resides in either a 32-bit word or an eight-bit byte. A word must be oriented on word boundaries (i.e., the address of the 32-bit word must have the two low-order bits zero).

The storage data in memory-to-memory operations have a variable length format and can have a field size of up to 256 bytes starting at any byte location. Processing is from left to right.

Instructions that specify an operand that is part of the immediate instruction being executed are restricted to a field size of one eight-bit byte.

The Translate and Test and the Edit and Mark instructions imply the use of General Register 1*. An address of 24 bits may be placed in this register during the execution of these instructions. The Translate and Test instruction also implies the use of General Register 2 where an insertion of an eight-bit function byte may be placed during the execution of the instruction.

Overlapping of fields in memory-to-memory operations may or may not affect the operands of the various instructions. The execution of some

* When these instructions are executed in P_3, General Registers 13 and 14 are used; in P_4, General Registers 9 and 10 are used.

DATA FORMAT (Cont'd)

logical instructions does not change the operands. Other instructions, such as Move, Edit, and Translate, replace one operand with new data, and this data is handled one eight-bit byte at a time. This procedure enables the user to determine the effect overlapping fields have on the execution of the instruction. Unpredictable results can occur while overlapping fields are being edited. Overlapping fields are valid for all other operations.

INSTRUCTION FORMATS

◆ The logical instructions use the following five instruction formats (RR, RX, RS, SI, SS):

RR Format

Op Code	R_1	R_2
0 7	8 11	12 15

Description

◆ In the RR format, the contents of the general register specified by R_1 are called the first operand. The contents of the general register specified by R_2 are called the second operand.

RX Format

Op Code	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Description

◆ In the RX format, the contents of the general register specified by R_1 are called the first operand. To obtain the address of the second operand, the contents of the general registers specified by X_2 and B_2 are added to the contents of the D_2 field.

RS Format

Op Code	R_1	R_3	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Description

◆ In the RS format, which is only used for shift instructions in this instruction set, the contents of the general register specified by R_1 are called the first operand. There is no actual storage address formed by adding the contents of the general register specified by B_2 and the contents of D_2. Instead, this sum specifies the number of bits to be shifted by the shift operations. The R_3 field is ignored in the shift operation.

SI Format

Op Code	I_2	B_1	D_1
0 7	8 15	16 19	20 31

Description

◆ In the SI format, the contents of the general register specified by B_1 are added to the contents of the D_1 field to obtain the address of the first operand. The second operand is the immediate eight-bit byte in the I_2 field of the instruction.

SS Format

Op Code	L	B_1	D_1	B_2	D_2
0 7	8 15	16 19	20 31	32 35	36 47

Description

◆ In the SS format, the contents of the general register specified by B_1 are added to the contents of the D_1 field to obtain the address of the leftmost byte of the first operand. The L field specifies the number of additional bytes in the operand that are to the right of the first operand. To obtain

SS Format (Cont'd)

the second operand address, the contents of the general register specified by B_2 are added to the contents of the D_2 field. The length of the second operand is the same as the length of the first.

The use of a zero in the X_2, B_1, or B_2 field of any instruction indicates that no register is to be used as a component of the instruction. Instructions may use a general register for both address modification and operand location. Addresses are always modified before an instruction is executed.

CONDITION CODE UTILIZATION

◆ The condition code is set as a result of using most of the logical instructions. The condition code setting has a different meaning when using different instructions and can be tested by subsequent branch on condition instructions for decision making. Altogether, there are six types of result meanings. The instructions which cause the condition code to be set and the meaning of the setting are as follows:

Instruction	Condition Code Setting			
	0	1	2	3
AND	Zero	Not Zero	——	——
Compare Logical	Equal	Low	High	——
Edit	Zero	< Zero	> Zero	——
Edit and Mark	Zero	< Zero	> Zero	——
Exclusive OR	Zero	Not Zero	——	——
OR	Zero	Not Zero	——	One
Test Under Mask	Zero	Mixed	——	——
Translate and Test	Zero	Incomplete	Complete	——
Test and Set	Zero	One	——	——

INTERRUPT ACTION

Address Error

◆ The following interrupt conditions can occur as a result of logical instructions:

Addressing

◆ An address error interrupt occurs when an address specifies a location outside the available memory. At the point of error the operation is terminated. The result data and condition code, if affected, are unpredictable.

Specification

◆ An address error interrupt occurs when a full-word operand is not located on a word boundary in a storage-to-register operation, or when an odd register is specified as the first register in an instruction which performs an operation on an even/odd pair of general registers. The operation is suppressed.

Protection

◆ An address error interrupt occurs when the storage key and the protection key of the result location do not match. The operation is suppressed and the condition code, registers, and main memory are unaltered. The variable-length memory-to-memory instructions are the only exception, in which case the operation is terminated and the result data and the condition code setting are unpredictable. (This interrupt can only occur if the memory protect feature is installed.)

Data Error

◆ A data error occurs if a digit code of the second operand in the Edit instruction or Edit and Mark instruction is invalid. The operation is terminated, and the result data and condition code setting are unpredictable.

Move (MVI) (MVC)

General Description

◆ To process the SS format Move instruction, the source field specified by the second address (B_2/D_2) is moved into the destination field specified by the first address (B_1/D_1). This format is used for a main memory-to-main memory move.

For the SI format Move instruction, the immediate byte in the I_2 field of the instruction being executed is stored in the main memory location specified by the first address (B_1/D_1).

Format (SI)

(MVI) 92	I_2	B_1	D_1
0 7	8 15	16 19	20 31

(SS)

(MVC) D2	L	B_1	D_1	B_2	D_2
0 7	8 15	16 19	20 31	32 35	36 47

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing.

Protection.

Notes

◆ 1. The bytes being moved are not inspected or changed.

2. Processing is from left to right and overlapping of fields is permitted.

3. The second operand is not altered, unless operands overlap in the SS format.

4. It is possible to propagate one byte through an entire field by having the first operand address specify one location to the right of the second operand address.

Move Numerics (MVN)

General Description

◆ The low-order four bits of each byte in the source operand specified by the second address (B_2/D_2) are placed into the low-order four bits of the corresponding byte of the destination operand specified by the first address (B_1/D_1).

Format (SS)

D1	L	B_1	D_1	B_2	D_2
0 7	8 15	16 19	20 31	32 35	36 47

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing.

Protection.

Notes

◆ 1. The numerics are not changed or checked for validity.

2. The operand specified by the second address is not altered, unless operands overlap.

3. Processing is from left to right.

4. The high-order four bits of the source and destination operand bytes are not altered.

5. The operand fields may overlap in any way and may be variable in length.

Move Zones (MVZ)

General Description

◆ The high-order four bits of each byte in the source operand specified by the second address (B_2/D_2) are placed into the high-order four bits of the corresponding byte of the destination operand specified by the first address (B_1/D_1).

Format (SS)

D3	L	B_1	D_1	B_2	D_2
0 7	8 15	16 19	20 31	32 35	36 47

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:
Addressing.
Protection.

Notes

◆ 1. The zones are not changed or checked for validity.

2. The operand specified by the second address is not altered, unless operands overlap.

3. Processing is from left to right.

4. The low-order four bits of the source and destination operand bytes are not altered.

5. The operand fields may overlap in any way and may be variable in length.

Test and Set (TS)

General Description

◆ This instruction is used to test and set a byte in memory and to set the condition code in accordance with the initial setting of the byte being tested. The I field of the instruction is not used (bits 8 through 15). The address field specifies the location of the byte being tested and set.

Format (SI)

93		B_1	D_1
0 7	8 15	16 19	20 31

Condition Code

◆ 0 — Leftmost bit of byte specified is zero.

1 — Leftmost bit of byte specified is one.

Interrupt Action

◆ Addressing.

Power Failure.

Machine check.

Notes

◆ 1. The leftmost bit (bit position 0) of the byte located at the first operand address is used to set the condition code, and the entire addressed byte is set to all ones.

2. The operation is terminated on any protection violation. The condition-code setting is unpredictable when a protection violation occurs.

3. This instruction, during execution, sequences two consecutive memory cycles, during which time I/O does not have access to memory. No other access to this location is permitted between the moment of fetching and the moment of storing all ones.

Compare Logical (CLR) (CL) (CLI) (CLC)

General Description

◆ The operand specified by the first address is logically compared with the operand specified by the second address (RR format: R_1 to R_2; RX format: R_1 to $X_2/B_2/D_2$; SI format: B_1/D_1 to I_2; SS format: B_1/D_1 to B_2/D_2). The result of the comparison determines the condition code. These instructions process all bits as part of an unsigned binary quantity. All codes are valid and the instruction is terminated on inequality or when the operand bytes have been exhausted.

Format (RR)

(CLR) 15	R_1	R_2
0 7	8 11	12 15

(RX)

(CL) 55	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

(SI)

(CLI) 95	I_2	B_1	D_1
0 7	8 15	16 19	20 31

(SS)

(CLC) D5	L	B_1	D_1	B_2	D_2
0 7	8 15	16 19	20 31	32 35	36 47

Condition Code

◆ 0 — the operands are equal.

1 — the first operand is less than the second operand.

2 — the first operand is greater than the second operand.

3 — not used.

Interrupt Action

◆ Address error:

Addressing (RX, SI, SS only).

Specification (RX only).

Notes

◆ 1. Both operands are unaltered.

2. In the SI format, the immediate byte in the I_2 field of the instruction being executed is the second operand.

3. Processing is from left to right and can extend to field lengths of 256 bytes.

4. The operation can be used for alphanumeric comparisons.

AND (NR) (N) (NI) (NC)

General Description

◆ These instructions perform a logical "AND" operation on two operands bit-by-bit according to the following rules:

Rules of Logical "AND" Operation

If Bit in First Operand is	And Bit in Second Operand is	Then Bit in Result is
0	0	0
0	1	0
1	0	0
1	1	1

The logical product of the operation is placed in the location specified by the first address (R_1 or B_1/D_1) and determines the condition code.

Format (RR)

(NR) 14	R_1	R_2
0 7	8 11	12 15

(RX)

(N) 54	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

(SI)

(NI) 94	I_2	B_1	D_1
0 7	8 15	16 19	20 31

(SS)

(NC) D4	L	B_1	D_1	B_2	D_2
0 7	8 15	16 19	20 31	32 35	36 47

Condition Code

◆ 0 — result is zero.
1 — result not zero.
2 — not used.
3 — not used.

Interrupt Action

◆ Address error:
Addressing (RX, SI, SS only).
Protection (SI, SS only).
Specification (RX only).

Notes

◆ 1. The second operand is unaltered, unless operands overlap in the SS format.

2. In the SI format, the immediate byte in the I_2 field of the instruction being executed is the second operand.

3. Processing is from left to right.

4. All operands and results are valid.

5. The "AND" instruction is also used to set a bit to zero.

OR
(OR) (O) (OI) (OC)

General Description

◆ This instruction performs a logical "OR" operation on two operands bit-by-bit according to the following rules:

Rules for Logical "OR" Operation

If Bit in First Operand is	And Bit in Second Operand is	Then Bit in Result is
0	0	0
0	1	1
1	0	1
1	1	1

The logical result of the operation is placed in the location specified by the first address (R_1 or B_1/D_1) and determines the condition code.

Format (RR)

(OR) 16	R_1	R_2
0 7	8 11	12 15

(RX)

(O) 56	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

(SI)

(OI) 96	I_2	B_1	D_1
0 7	8 15	16 19	20 31

(SS)

(OC) D6	L	B_1	D_1	B_2	D_2
0 7	8 15	16 19	20 31	32 35	36 47

Condition Code

◆ 0 — result is zero.

1 — result is not zero.

2 — not used.

3 — not used.

Interrupt Action

◆ Address error:

Addressing (RX, SI, SS only).

Protection (SI, SS only).

Specification (RX only).

Notes

◆ 1. The second operand is unaltered, unless operands overlap in the SS format.

2. In the SI format, the immediate byte in the I_2 field of the instruction being executed is the second operand.

3. Processing is from left to right.

4. All operands and results are valid.

5. The "OR" instruction is also used to set a bit to one.

Exclusive OR (XR) (X) (XI) (XC)

General Description

◆ These instructions perform an Exclusive "OR" operation on two operands bit-by-bit according to the following rules:

Rules for Exclusive "OR" Operation

If Bit of First Operand is	And Bit of Second Operand is	Then Bit in Result is
0	0	0
0	1	1
1	0	1
1	1	0

The modulo-two sum (binary addition without carries) of the operation is placed in the location specified by the first address (R_1 or B_1/D_1) and determines the condition codes.

Format (RR)

(XR) 17	R_1	R_2
0 7	8 11	12 15

(RX)

(X) 57	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

(SI)

(XI) 97	I_2	B_1	D_1
0 7	8 15	16 19	20 31

(SS)

(XC) D7	L	B_1	D_1	B_2	D_2
0 7	8 15	16 19	20 31	32 35	36 47

Condition Code

◆ 0 — result is zero.
1 — result is other than zero.
2 — not used.
3 — not used.

Interrupt Action

◆ Address error:

Addressing (RX, SI, SS only).
Protection (SI, SS only).
Specification (RX only).

Notes

◆ 1. The second operand is unaltered, unless operands overlap in the SS format.
2. In the SI format, the immediate byte in the I_2 field of the instruction being executed is the second operand.
3. Processing is from left to right.
4. All operands and results are valid.
5. These instructions may be used to complement a number (one's complement).

Test Under Mask (TM)

General Description

◆ The operand (byte) specified by the first address (B_1/D_1) is tested against the immediate I field (byte) as a mask. The result determines the condition code. The I field is used as an eight-bit mask and is made to correspond one-for-one with the bits of the byte in main memory that is specified by the first address.

A bit in the byte being examined is said to be selected when the corresponding mask bit is a one. When the mask bit is a zero, the bit in main memory is ignored.

Format (SI)

91	I_2	B_1	D_1
0 7	8 15	16 19	20 31

Condition Code

◆ 0 — selected bits all zero or mask is all zero.

1 — selected bits mixed zero and one.

2 — not used.

3 — selected bits all one's.

Interrupt Action

◆ Address error:

Addressing.

Note

◆ The operands are unaltered.

Insert Character (IC)

General Description

◆ The eight-bit byte specified by the second address ($X_2/B_2/D_2$) is loaded into the rightmost byte of the general register specified by the first address (R_1). The remaining bits of the register are unaltered.

Format (RX)

43	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing.

Note

◆ The operand specified by the second address is not altered or inspected.

Store Character (STC)

General Description

◆ The rightmost eight-bit byte of the general register specified by the first address (R_2) is stored into the main memory location specified by the second address ($X_2/B_2/D_2$).

Format (RX)

42	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing.

Protection.

Note

◆ The operand specified by the first address is not altered or inspected.

Load Address (LA)

General Description

◆ The final main memory address specified by the second operand ($X_2/B_2/D_2$) is loaded into the rightmost 24 bits of the general register specified by the first address (R_1). The leftmost eight bits of the register are set to zeros.

The contents of the registers specified by the X_2 and B_2 fields are added to the contents of the D_2 field of the instruction to obtain an address. This is the address that is loaded into the register specified by the first address. Any carry beyond the rightmost 24 bits is ignored.

Format (RX)

41	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ None.

Notes

◆ 1. All specified address arithmetic is computed before loading.

2. R_1, X_2 and B_2 may specify the same register; however R_1 only may specify register 0.

3. This instruction can be used to increment the low-order 24 bits of a general register (other than 0) by the contents of the D_2 field. The register to be incremented is specified by R_1, and either X_2 (with B_2 set to zero) or B_2 (with X_2 set to zero). Since R_1 and X_2 or B_2 must specify the same register, register zero cannot be incremented (a zero in the B_2 or X_2 field indicates that the corresponding address component is absent).

4. Main memory is not accessed by this instruction.

Translate (TR)

General Description

◆ The variable length operand specified by the first address (B_1/D_1) is translated, byte-for-byte, according to the byte translation table specified by the second address (B_2/D_2). The result replaces the bytes in the field specified by the first address.

The bytes of the first operand are termed the argument bytes. Bytes of the first operand are selected for translation from left-to-right, one byte at a time. Each argument byte is added to the second operand address, which is the starting location of a translation table. This sum, in turn, addresses a byte location within the table containing a function byte. The function byte at this location replaces the original argument byte of the first operand.

The operation terminates when the first operand bytes have been exhausted.

Format (SS)

DC	L	B_1	D_1	B_2	D_2
0 7	8 15	16 19	20 31	32 35	36 47

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing.

Protection.

Notes

◆ 1. The translation table is unaltered unless overlap occurs.

2. The field to be translated and the translation table are addressed by their leftmost byte.

3. The length of a table, in general, must be 256 bytes, unless the domain of argument bytes is limited to a specific subset by the program and data.

4. The L field specifies the length of the first operand minus one (binary 00000001 = 2 bytes).

Translate and Test (TRT)

General Description

◆ The variable length operand, which is specified by the first address (B_1/D_1), is used as the argument (byte-by-byte) to reference a list (functions) specified by the second address (B_2/D_2). The functions referenced are inspected for zero or non-zero. If a non-zero is encountered, the address of the argument byte is loaded into General Register 1 (General Register 13 in P_3; General Register 9 in P_4) and the function byte is loaded into the rightmost end of General Register 2 (General Register 14 in P_3; General Register 10 in P_4). Whenever zeros are encountered in the function list, the operation proceeds to the next byte. The first operand is unaltered.

The bytes of the first operand are termed the argument bytes. Processing of the first operand is from left-to-right, one byte at a time. Each argument byte is added to the second operand, which is the starting location of the translate table. This sum, in turn, addresses a byte location within the table, which is termed a function byte. Then, the function byte retrieved from the table is inspected for all zeros.

If the function byte is all zeros, the operation proceeds to the next argument byte and continues processing. If the function byte is not all zeros, the instruction inserts the address of the argument byte in the low-order 24 bits of General Register 1 (13 or 9) and inserts the retrieved non-zero function byte in the low-order eight-bits of General Register 2 (14 or 10). The high-order eight bits of General Register 1 (13 or 9) and high-order 24 bits of General Register 2 (14 or 10) are unaltered.

The operation terminates when a (non-zero) function byte is accessed or when the first operand field is exhausted.

Format (SS)

DD	L	B_1	D_1	B_2	D_2
0 7	8 15	16 19	20 31	32 35	36 47

Condition Code

◆ 0 — accessed function bytes all zeros.
1 — a non-zero function byte is encountered before the first operand field is exhausted.
2 — the last function byte is non-zero.
3 — not used.

Interrupt Action

◆ Address error:
Addressing.

Notes

◆ 1. The variable length field specified by the first address is unaltered.
2. If non-zero functions do not occur, General Registers 1 (13 or 9) and 2 (14 or 10) are unaltered.
3. The first operand and the translation table are addressed by their leftmost bytes.
4. The length of the table, in general, must be 256 bytes, unless the domain of argument bytes is limited to a specific subset by the program and data.
5. The L field specifies the length of the first operand minus one.
6. This instruction is useful for scanning input streams and locating delimiters for variable length records and fields.
7. In processor states P_1 and P_2, General Registers 1 and 2 are used. In processor state P_3, General Registers 13 and 14 are used. In processor state P_4, General Registers 9 and 10 are used.

Edit (ED)

General Description

◆ The variable length source field specified by the second address (B_2/D_2) is changed from packed format to zoned format with the results edited under the control of a mask pattern. The result of the operation replaces the mask pattern specified by the first address (B_1/D_1) and determines the condition code.

The L field applies to the mask pattern (first address field). The source digits are processed left-to-right, one byte at a time. The leftmost four bits of each byte are examined first and the rightmost four bits of each byte are held available for the next mask character that calls for digit examination. Immediately after the leftmost four bits have been examined, the rightmost four bits are checked for a sign code. When one of the sign codes is encountered, these bits are no longer treated as a digit. A new character is fetched from the mask pattern for the next digit to be examined.

Format (SS)

DE	L	B_1	D_1	B_2	D_2
0 7	8 15	16 19	20 31	32 35	36 47

Editing Rules

◆ Editing includes sign control, punctuation control, zero suppression or check protection, and also facilitates blanking of all-zero fields. In addition, multiple fields of digits can be edited in one operation, and numeric data can be combined with alphabetic and special characters.

Editing rules depend on the control code, significance, and the source digit, and are given as follows:

Editing Rules

Control Codes	Hexadecimal Code	Decimal Code	Function
Filler	Any	Any	*Replaces leading zeros.
Start Significance	21	33	Stops replacement of leading zeros. Also acts as a digit select code.
Digit Select	20	32	Specifies digit position in data (replaced by filler code if appears after a negative sign has been sensed).
Field Separator	22	34	Indicates editing of a new field is to begin (replaced by filler code).
Insertion Character	Any	Any	Inserted in the result.

* The most common filler characters are the blank and the asterisk.

1. Source digits are examined only when a digit select code $(20)_{16}$ or a start significance code $(21)_{16}$ is encountered in the mask pattern.
2. Significance is established either:
   a. upon encountering a non-zero digit in the source field.
   b. after encountering a start significance code $(21)_{16}$ within the mask pattern.

Editing Rules *(Cont'd)*

3. If significance has *not* been established, every control code or insertion character encountered in the mask pattern (including the start significance code) is replaced by the filler character.

4. If significance *has* been established, every digit select code $(20)_{16}$ or start significance code $(21)_{16}$ encountered in the mask pattern is replaced by a digit from the source field, which is expanded by attaching a zone.

5. If significance *has* been established, every insertion character (other than the digit select, start significance, or field separator codes) encountered within the mask pattern is left in place without alteration.

6. Significance is disestablished by:
   a. encountering a field separator code $(22)_{16}$ in the mask pattern.
   b. encountering a positive (plus) sign within the rightmost four bits of a source field byte.

7. A negative (minus) sign within the rightmost four bits of a source byte does *not* disestablish significance. Additional digit select codes encountered in the mask pattern are replaced by filler characters, but insertion characters are left in place without alteration.

8. Field separator codes $(22)_{16}$ are always replaced by the filler character.

   Note: The filler character is obtained from the mask pattern as part of the editing operation. The first character (leftmost byte) of the mask pattern is used as a filler character and is left unchanged in the result, except:

   a. when it is a digit select code.

   b. when it is a start significance code.

   In these codes, a source digit is examined and, when non-zero, inserted in the result field.

To facilitate blanking out all-zero result fields, or triggering negative field special processing, the condition code is used to indicate the sign and zero status of the last field edited. All digits examined are tested for zero, and the presence, or absence, of an all-zero source field is indicated in the condition code at the termination of the editing operation. Sign significance is also indicated by the condition code.

Condition Code

◆ 0 — indicates a zero source field regardless of whether or not significance is established.

1 — indicates non-zero result field with significance established to indicate less than zero.

2 — indicates non-zero result field with no significance established to indicate greater than zero.

3 — not used.

Note: The condition code setting reflects only the field following the last (rightmost) field separator code of the mask pattern for multiple-field-editing operations.

Interrupt Action

◆ Address error:

Addressing.

Protection.

Data error.

Notes

◆ 1. The leftmost four-bits of any source field byte must be a valid digit, otherwise a data error interrupt occurs.

2. The rightmost four-bits of any source field byte can be either a digit or a sign.

3. Multiple field editing is possible by using the field separator code within the mask pattern.

4. The zones of the expanded source digits can be either EBCDIC or USASCII, as specified by the mode code. When the mode code specifies EBCDIC, zone code 1111 is generated. When the mode code specifies USASCII, the zone code 0101 is generated.

5. The rightmost four bits of any source field byte can be a digit or sign as follows:

Codes	Definition
0000 → 1001	Digits
1010, 1100, 1110, 1111	Plus sign
1011, 1101	Minus sign

6. Overlapping of fields yields unpredictable results.

7. In testing for Paging Error or Paging Queue interrupt conditions, the hardware assumes that the number of bytes in the source field is equal to the number of bytes in the pattern field. If the assumed source field extends across two pages, of which the second page has any of the conditions causing Paging Error or Paging Queue interrupts, but the actual source field number of bytes is short enough to fit within the first page, a false Paging Error condition or Paging Queue Interrupt condition occurs.

Edit and Mark (EDMK)

General Description

◆ The variable length source field specified by the second address (B_2/D_2) is changed from packed format to zoned format and the results are edited under control of a mask pattern. The result of the operation replaces the mask pattern specified by the first address (B_1/D_1) and determines the condition code. In addition, the address of each first significant result digit is stored in General Register 1 (General Register 13 in P_3; General Register 9 in P_4).

The operation of this instruction is identical to the Edit instruction except for the additional function of inserting a byte address in General Register 1 (13 or 9). The destination address of the digit that establishes significance within the source field being edited is loaded into the rightmost 24 bits of General Register 1 (13 or 9). The leftmost eight bits are unaltered. The address is *not* loaded when significance is forced by recognition of the start significance code in the mask pattern.

The Edit and Mark instruction facilitates the insertion of floating currency symbols, sign indicators, relational operators, and other editing symbols ($, +, −, <, >, etc.). The address loaded into the register is one byte to the right of the address where such a symbol would be inserted. (The Branch on Count instruction, with zero in the R_2 field, can be used to reduce the loaded address by one.)

Because the address is *not* loaded when significance is forced by the start significance code, the address of the byte immediately to the right of the start significance code in the mask pattern field should be loaded in General Register 1 (13 or 9) before an Edit and Mark instruction is executed.

Format (SS)

DF	L	B_1	D_1	B_2	D_2
0 7	8 15	16 19	20 31	32 35	36 47

Condition Code

◆ 0 — indicates a zero source field whether or not significance is established.

1 — indicates non-zero result field with significance established to indicate less than zero.

2 — indicates non-zero result field with no significance established to indicate greater than zero.

3 — not used.

Interrupt Action

◆ Address error:

Addressing.

Protection.

Data error.

Notes

◆ 1. All notes of the Edit instruction are applicable to the Edit and Mark instruction.

2. The address of the byte is loaded each time significance is established and a non-zero character is inserted into the result field.

Notes *(Cont'd)*

3. The address is loaded into the rightmost 24 bits of General Register 1 (13 or 9). The leftmost eight bits are unaltered.
4. When a single instruction is used to edit multiple fields, the address of the first significant digit of each field is loaded into the register. However, only the address of the last field processed will be available upon completion of the instruction.
5. In processor states P_1 and P_2, General Register 1 is used. In processor state P_3, General Register 13 is used. In processor state P_4, General Register 9 is used.
6. In testing for Paging Error or Paging Queue interrupt conditions, the hardware assumes that the number of bytes in the source field is equal to the number of bytes in the pattern field. If the assumed source field extends across two pages, of which the second page has any of the conditions causing Paging Error or Paging Queue interrupts, but the actual source field number of bytes is short enough to fit within the first page, a false Paging Error condition or Paging Queue Interrupt condition occurs.

Shift Left Single Logical (SLL)

General Description

◆ The entire contents of the general register specified by the first address (R_1) are shifted left the number of bit positions specified by the second address (B_2/D_2). The R_3 field is ignored.

The second address does not refer to a main memory location. The low-order six bits of the second address are used as the count to specify the number of bits of shifting to be done. The remaining bits are ignored.

Format (RS)

89	R_1	R_3	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ None.

Notes

◆ 1. High-order bits of the register are shifted out and lost.

2. Zeros are placed into the right end of the register.

3. All 32 bits of the specified register are shifted.

Shift Right Single Logical (SRL)

General Description

◆ The entire contents of the general register specified by the first address (R_1) are shifted right by the number of bit positions specified by the second address (B_2/D_2). The R_3 field is ignored.

The second address does not refer to a main memory location. The low-order six bits of the second address are used as the count to specify the number of bits shifting to be done. The remaining bits are ignored.

Format (RS)

88	R_1	R_3	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ None.

Notes

◆ 1. Low-order bits of the register are shifted out and lost.

2. Zeros are placed into the left end of the register.

3. All 32 bits of the specified register are shifted; that is, the operation is unsigned.

Shift Left Double Logical (SLDL)

General Description

◆ The entire contents of the double-length operand (two general registers) — even/odd specified by the first address (R_1) are shifted left the number of bit positions specified by the second address (B_2/D_2). The R_3 field is ignored.

The second address does not refer to a main memory location. The low-order six bits of the second address are used as the count to specify the number of bits of shifting to be done. The remaining bits are ignored.

Format (RS)

8D	R_1	R_3	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:
Specification.

Notes

◆ 1. The first address must specify an even-numbered register.
2. All 64 bits of the double-length operand are shifted.
3. High-order bits are shifted out and lost.
4. Zeros are placed into the low-order end of the odd-numbered register.

Shift Right Double Logical (SRDL)

General Description

◆ The entire contents of the double-length operand (two general registers) — even/odd specified by the first address (R_1) are shifted right the number of bit positions specified by the second address (B_2/D_2). The R_3 field is ignored.

The second address does not refer to a main memory location. The low-order six bits of the second address are used as the count to specify the number of bits of shifting to be done. The remaining bits are ignored.

Format (RS)

8C	R_1	R_3	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interruption

◆ Address error:
Specification.

Notes

◆ 1. The first address must specify an even-numbered register.
2. All 64 bits of the double-length operand are shifted.
3. Low-order bits are shifted out and lost.
4. Zeros are placed into the high-order end of the even-numbered register.

BRANCHING INSTRUCTIONS

INTRODUCTION

◆ In normal processor operation, instructions are executed in sequential order according to the main memory locations in which they are stored. When branching is performed, a break in this normal sequential execution occurs. Branching instructions provide for referencing another subroutine or repeating a segment of coding or continuing to the next instruction in sequence. When branching occurs, the address specified in the branch instruction replaces the current address in the P counter. The branch address can be specified by an instruction address or it can be obtained from one of the general registers.

The actual branching execution is based on the setting of the condition code or on the contents of a general register as specified in the loop-closing operations.

In a branching operation, the current address in the updated P counter can be stored before the branch address is placed in the P counter. This stored address can be used for linking the new segment of instructions with the segment of instructions from which the branching occurred.

The Execute instruction is listed with the branch instructions, although only a temporary departure from sequential operation is entailed by use of this instruction. The branch address, in this instruction, specifies one instruction to be executed in the instruction sequence. The address in the P counter is not replaced by the branch address and only the instruction located at the address is executed before the sequence is continued based upon the updated P counter.

SEQUENTIAL EXECUTION

◆ Normally, the P counter instruction address specifies a main memory location from which the next instruction to be executed is fetched. This instruction address is updated in the P counter by the length, in bytes, of the instruction to be executed as indicated by the current P counter. The instruction currently indicated by the P counter is executed and the operation is repeated using the updated P counter to fetch the next instruction.

Instructions can occupy from one halfword (two bytes) up to three halfwords (six bytes). The high-order two bits of the operation code of each instruction designates its length as follows:

00 = halfword instruction (two bytes).
01, 10 = two-halfword instructions (four bytes).
11 = three-halfword instructions (six bytes).

INSTRUCTION FORMATS

◆ Branching instructions use the following three instruction formats:

RS Format

Op Code	R_1	R_3	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Description

◆ The contents of the general register specified by B_2 are added to the contents of the D_2 field to obtain the branch address (second operand). The R_1 field specifies the general register that contains the first operand. The R_3 field specifies the general register that contains the third operand.

RX Format

Op Code	R_1/M	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Description

◆ The contents of the general registers specified by X_2 and B_2 are added to the contents of the D_2 field to obtain the branch address (second operand). The R_1 field specifies the general register which contains the first operand. In a Branch on Condition instruction, the M field is a mask which specifies the condition codes to be tested.

RR Format

Op Code	R_1/M	R_2
0 7	8 11	12 15

Description

◆ The contents of the general register specified by the R_2 field are the branch address (second operand). The R_1 field specifies the general register that contains the first operand. The same register can be specified by R_1 and R_2. If R_2 is zero, no branching occurs. In a Branch on Condition instruction, the M field is a mask that specifies the condition codes to be tested.

Notes:

1. A zero in the X_2 or B_2 field indicates that the corresponding address component is absent.
2. The sequence of operations when using general registers is as follows:
   a. compute the address.
   b. store arithmetic or link information.
   c. replace the P counter with the branch address.

INTERRUPT ACTION

◆ Interrupts can occur as a result of an Execute instruction only. The interrupt conditions are as follows:

Address Error

Addressing

◆ An address error interrupt occurs when the branch address of an Execute instruction is outside the main memory for the particular installation, or if an Execute instruction is attempted to perform another Execute instruction. The operation is suppressed and the condition code, registers, and main memory are unaltered.

Specification

◆ An address error interrupt occurs if the branch address of an Execute instruction is not on a halfword boundary. The operation is suppressed and the condition code, registers, and main memory are unaltered.

Branch on Condition (BCR) (BC)

General Description

◆ If the condition code is set to any of the conditions specified by the four-bit mask field (M or M_1), the P counter is replaced by the branch address (R_2 or $X_2/B_2/D_2$). If the four-bit mask field (M or M_1) is not equivalent to the condition code settings, branching does not occur and the next instruction in sequence is executed. The branch is initiated whenever the condition code has a corresponding mask bit set.

Format (RR)

(BCR) 07	M_1	R_2
0 7	8 11	12 15

(RX)

(BC) 47	M	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ None.

Notes

◆ 1. The four-bit mask in M_1 corresponds, left-to-right, with the four condition codes:

Instruction Bit	Condition Code
8	0
9	1
10	2
11	3

2. If all mask bits are set ($M_1 = F_{16}$), an unconditional branch is effected.

3. When all mask bits are zero, or if R_2 in the RR format is zero, the instruction is a no-op.

4. When a branch occurs, the leftmost eight-bit portion of the 32-bit P counter (ILC, CC, and mask) is unpredictable. However, the actual condition code and program mask (hardware registers) are unaffected by branching.

5. The contents of the registers specified by the second address are unaltered.

Branch and Link (BALR) (BAL)

General Description

◆ The entire 32-bit contents of the P counter are loaded into the general register specified by R_1. Then, the program branches to the instruction address specified by the branch address (R_2 or $X_2/B_2/D_2$). The instruction length counter, the condition code, the program mask, and the updated instruction address are stored. However, when branching occurs, only the instruction address is replaced.

Format (RR)

(BALR) 05	R_1	R_2
0 7	8 11	12 15

(RX)

(BAL) 45	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ None.

Notes

◆ 1. The P counter is stored without branching in the RR format when the R_2 field is zero.

2. When a branch occurs, the leftmost eight-bit portion of the 32-bit P counter (ILC, CC, and mask) is unpredictable. However, the actual condition code and program mask (hardware registers) are unaffected by branching.

3. The contents of the register specified by the second address are unaltered.

4. The P counter is moved to a reserved area in memory; the branch then takes place as specified by the contents of R_2 or $X_2/B_2/D_2$. The P counter (from the reserved area) is then placed into R_1.

Branch on Count (BCTR) (BCT)

General Description

◆ The contents of the general register specified by the R_1 field are algebraically decremented by one. The contents of the register are examined, and if the contents are zero, no branching occurs. If the contents are not zero, the instruction address in the P counter is replaced by the branch address (R_2 or $X_2/B_2/D_2$) and branching occurs.

Format (RR)

(BCTR) 06	R_1	R_2
0 7	8 11	12 15

(RX)

(BCT) 46	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ None.

Notes

◆ 1. The subtraction executes as in fixed-point arithmetic with all 32 bits participating.

2. An initial count of zero in the R_1 field results in branching, because subtraction occurs before testing the contents of the register. If the value is zero, branching occurs and the result is minus one. To effect a *no branch,* the contents of the R_1 field must be 1.

3. The contents of the registers specified by the second address are unaltered.

4. When branching occurs, the leftmost eight-bit portion of the 32-bit P counter (ILC, CC, and mask) is unpredictable. However, the actual condition code and program mask (hardware registers) are unaffected by branching.

5. In the RR format, if the R_2 field is zero, counting is performed without branching.

6. If a negative number appears in R_1, an overflow condition occurs when this field is decremented. However, this overflow is ignored.

7. Overflow from a maximum negative number to a maximum positive number is ignored.

Branch on Index High (BXH)

General Description

◆ The operand specified by the third address (R_3) is added to the operand specified by the first address (R_1) and the sum is algebraically compared with the operand specified by the third address (R_3), if R_3 specifies an odd register. If R_3 specifies an even register, the sum is algebraically compared with $R_3 + 1$. If the sum is low or equal, branching does not occur and the next instruction is executed. If the sum is high, the instruction address in the P counter is replaced by the branch address (B_2/D_2) and branching occurs.

Format (RS)

86	R_1	R_3	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ None.

Notes

◆ 1. The sum replaces the operand specified by the first address (R_1) regardless of the comparison. The sum replaces (R_1) after the comparison has been made.

2. Overflow is not recognized.

3. The contents of the register specified by R_3 or $R_3 + 1$ are unaltered.

4. When a branch occurs, the leftmost eight-bit positions of the 32-bit P counter (ILC, CC, and mask) are unpredictable. However, the actual condition code and program mask (hardware registers) are unaffected by branching.

Branch on Index Low or Equal (BXLE)

General Description

◆ The operand specified by the third address (R_3) is added to the operand specified by the first address (R_1) and the sum is algebraically compared with the operand specified by the third address (R_3), if R_3 specifies an odd register. If R_3 specifies an even register, the sum is algebraically compared with $R_3 + 1$. If the sum is high, branching does not occur and the next instruction in sequence is executed. If the sum is low or equal, the instruction address in the P counter is replaced by the branch address (B_2/D_2) and branching occurs.

Format (RS)

87	R_1	R_3	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ None.

Notes

◆ 1. The sum replaces the operand specified by the first address (R_1) regardless of the comparison. The sum replaces (R_1) after the comparison has been made.

2. Overflow is not recognized.

3. The contents of the register specified by R_3 or $R_3 + 1$ are unaltered.

4. When a branch occurs, the leftmost eight-bit positions of the 32-bit P counter (ILC, CC, and mask) are unpredictable. However, the actual condition code and program mask (hardware registers) are unaffected by branching.

Execute (EX)

General Description

◆ The instruction in the location specified by the second address ($X_2/B_2/D_2$) is modified by the contents of the register specified by the first address (R_1). Then, the modified instruction is executed and control is returned to the instruction following the Execute instruction.

Format (RX)

44	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ May be set by the instruction being modified and executed.

Interrupt Action

◆ Address error:

Addressing.

Specification.

Notes

◆ 1. Bits 8–15 of the subject instruction are "OR"ed with bits 24–31 of the register specified by the first address (R_1).

2. If R_1 is zero, no modification takes place.

3. The ILC is set to two (length of the Execute) and the P counter is set to the address of the instruction following the Execute instruction.

4. The contents of R_1 and the subject instruction in main memory are unaltered.

5. Interrupts are inhibited until the subject instruction has been completed.

6. When the subject instruction is a successful branching instruction, the P counter is updated by the branch address.

FLOATING-POINT INSTRUCTIONS

INTRODUCTION

◆ Floating-point arithmetic instructions provide the capability to process operands of large magnitude with precise results.

A floating-point number is made up of three parts: a sign, an exponent and a mantissa. The sign portion applies to the mantissa. The exponent is a power to which the number 16 is raised. The mantissa is a hexadecimal number with an assumed radix point to the left of the high-order digit. The quantity that the floating-point number represents is obtained by multiplying the mantissa by the number 16 raised to the power represented by the exponent.

Four floating-point registers are provided, each of which is 64 bits long. These registers are numbered 0, 2, 4 and 6.

Included in this set are instructions for loading, adding, subtracting, comparing, multiplying, dividing, storing, and controlling signs of short and long operands.

Addition, subtraction, multiplication, and division produce normalized results. Addition and subtraction can also produce unnormalized results. Operands can be normalized, or unnormalized, in any floating-point operation.

Sign control, add, subtract, and compare operation results are indicated in the condition code settings.

DATA FORMATS

◆ Floating-point numbers are fixed in length and are either full-word *short* or double-word *long* in format.

The first bit in both formats is the sign of the mantissa. A 1 bit represents a minus sign and a 0 bit represents a plus sign. The next seven bits represent the exponent. The mantissa contains six hexadecimal digits (short floating-point number) or 14 (long floating-point number) hexadecimal digits.

The short format allows for faster processing and uses less storage. Because floating-point registers are 64 bits long, the rightmost 32 bits are ignored when dealing with short operands. When the short format is specified, all operands and the result are 32 bits long. When using the long format, which provides greater precision, all operands are 64 bits long and require the full register.

Short Floating-Point Number

1	7	24
S	Exponent	Mantissa
0	1 7	8 31

Long Floating-Point Number

1	7	56
S	Exponent	Mantissa
0	1 7	8 63

REPRESENTATION OF NUMBERS

◆ The mantissa is always represented in hexadecimal. An assumed radix point is always immediately to the left of the high-order digit of the mantissa.

The exponent, bits 1 through 7, indicates the power to which the number 16 must be raised. The range of the exponent is from −64 to +63 corresponding to the binary value of 0–127. The power is equal to the binary number minus 64, as shown in following table:

Exponent	Decimal Equivalent	Power
$(1\ 111\ 111)_2$	127 −64	= +63
$(1\ 000\ 111)_2$	71 −64	= +7
$(0\ 000\ 000)_2$	0 −64	= −64

Because the value $(64)_{10}$ represents the power zero, this technique is called excess 64 notation.

The sign of a result from addition, subtraction, multiplication, or division with a zero mantissa is positive. A zero sign, zero exponent, and zero mantissa in a floating-point number is called true zero.

NORMALIZATION

◆ A floating-point number with a mantissa containing a non-zero, high-order, hexadecimal digit is called a normalized number. An unnormalized number has one or more high-order hexadecimal zero digits in the mantissa. To change an unnormalized number into a normalized number, the mantissa is shifted to the left until the high-order digit is non-zero. Then, the exponent is decremented by the number of digits shifted.

Generally, normalization occurs when the intermediate arithmetic result is changed to the final result. However, in multiplication and division operations, normalization occurs before the arithmetic process.

Floating-point operations are performed with, or without, normalization. Most operations are performed in only one way; however, addition and subtraction may be performed either way as specified.

When normalization is not performed, high-order zeros in the result mantissa are not eliminated. Depending on the original operands, the result may, or may not, be normalized.

Initial operands in both normalized and unnormalized operations need not be in normalized form. Because normalization takes place on hexadecimal digits, the three high-order bits of a normalized mantissa can be zero.

INSTRUCTION FORMATS

◆ The following two instruction formats are used for floating-point operations:

RX Format

Op Code	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Description

◆ An address is formed by adding the contents of general registers X_2 and B_2 to the displacement field D_2. This address specifies a main memory location that contains the second operand in the operation. R_1 designates the floating-point register containing the first operand.

RR Format

Op Code	R_1	R_2
0 7	8 11	12 15

Description

◆ In this format, R_1 designates the address of the floating-point register holding the first operand. R_2 is the address of the floating-point register holding the second operand. The first and second operands can be the same and are designated by identical R_1 and R_2 addresses.

Notes:

1. Register addresses specified by the R_1 and R_2 fields must be 0, 2, 4, or 6 or an address error (specification) interrupt occurs.
2. A short operand must be located on a word boundary and a long operand must be on a double-word boundary; if not, an address error (specification) interrupt occurs.
3. Floating-point registers are used by floating-point instructions only.
4. A zero in an X_2 or B_2 field shows that there is no address component to enter in forming an address.
5. Except for the instructions Store (long) and Store (short), results of floating-point operations replace the first operand.
6. Except for the storing of the result, the contents of floating-point registers, general registers, and main memory locations used in the operations are not changed.
7. It is possible to designate the same general register to specify both operand locations and address generation. Addresses are generated before execution.

CONDITION CODE UTILIZATION

◆ The condition code reflects results of floating-point sign control, add, subtract, and compare instructions. The code is not changed by any other floating-point operation. Decision-making by branch on condition instructions can be done after those instructions that set the code.

For most arithmetic and load instructions, Condition Codes 0, 1, or 2 indicate respectively a zero, or less than, or greater than zero content, of the result. Condition Code 3 is set for overflow of the result in arithmetic instructions only. In comparison instructions, the Condition Codes 0, 1, or 2 show, respectively, that the first operand is either equal to, less than, or greater than the second operand.

Instructions that cause the condition code to be set and the meaning of the setting are as follows:

Instruction	Condition Code Setting			
	0	1	2	3
Add Normalized Short/Long	Zero	< Zero	> Zero	Overflow
Add Unnormalized Short/Long	Zero	< Zero	> Zero	Overflow
Compare Short/Long	Equal	Low	High	—
Load and Test Short/Long	Zero	< Zero	> Zero	—
Load Complement Short/Long	Zero	< Zero	> Zero	—
Load Negative Short/Long	Zero	< Zero	—	—
Load Positive Short/Long	Zero	—	> Zero	—
Subtract Normalized Short/Long	Zero	< Zero	> Zero	Overflow
Subtract Unnormalized Short/Long	Zero	< Zero	> Zero	Overflow

INTERRUPT ACTION

◆ The following interrupt conditions can occur as a result of a floating-point instruction.

Address Error

Addressing

◆ An address error interrupt occurs when an address in the RX instruction format specifies a location outside the available main memory. The operation is terminated at the point of error. The result data and the condition code (if affected) are unpredictable.

Specification

◆ An address error interrupt occurs if a short operand is not located on a word boundary or a long operand is not located on a double-word boundary. An address error interrupt also occurs if a floating-point register other than 0, 2, 4 or 6 is specified. The instruction is suppressed. The condition code, the data in main memory, and the registers remain unchanged. Address restrictions do not apply to the X_2, B_2 and D_2 components of the instruction.

Protection

◆ An address error interrupt occurs when the protection key and the storage key of the result location do not match. The operation is suppressed. The condition code, the data in main memory, and the registers remain unchanged. (This interrupt can only occur if the memory protect feature is installed.)

Significance Error

◆ A significance error interrupt occurs when the result mantissa of an add or subtract operation is zero. A program interrupt occurs if the significance error mask bit in the Interrupt Mask Register of the current state is set to 1. The operation is completed, the exponent is unaltered, and the interrupt is taken. If the significance error mask bit is zero, the interrupt is prohibited and the operation is completed by setting the result to true zero (zero sign, zero exponent, and zero mantissa). In either case, the condition code is set to zero.

Divide Error

◆ A divide error interrupt occurs if division by zero is attempted.

Exponent Overflow

◆ An exponent overflow interrupt occurs when the result exponent overflows and the mantissa is not zero. The operation is terminated and the result data is unpredictable. Addition and subtraction set the condition code to 3. Multiplication and division do not affect the condition code setting.

Exponent Underflow

◆ An exponent underflow interrupt occurs when the result exponent is less than zero and the result mantissa is not zero. The operation is completed by setting the result to true zero (zero sign, zero exponent, and zero mantissa). Addition and subtraction set the condition code to zero. Multiplication and division do not affect the condition code setting.

Load (LER) (LE) (LDR) (LD)

General Description

◆ The operand specified by the second address (R_2 or $X_2/B_2/D_2$) is loaded into the floating-point register specified by the first address (R_1).

Format (RR Short)

(LER) 38	R_1	R_2
0 7	8 11	12 15

(RX Short)

(LE) 78	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

(RR Long)

(LDR) 28	R_1	R_2
0 7	8 11	12 15

(RX Long)

(LD) 68	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing (RX format).

Specification.

Notes

◆ 1. The operand specified by the second address is unaltered.

2. Exponent overflow, underflow, or lost significance cannot occur.

3. The low-order half of the register specified by the first address is unaltered when short operands are used.

Load and Test (LTER) (LTDR)

General Description

◆ The operand in the floating-point register specified by the second address (R_2) is loaded into the floating-point register specified by the first address (R_1). The sign and magnitude of the loaded operand determine the condition code.

Format (RR Short)

(LTER) 32	R_1	R_2
0 7	8 11	12 15

(RR Long)

(LTDR) 22	R_1	R_2
0 7	8 11	12 15

Condition Code

◆ 0 — result mantissa is zero.

1 — result mantissa is less than zero.

2 — result mantissa is greater than zero.

3 — not used.

Interrupt Action

◆ Address error:

Specification.

Notes

◆ 1. If R_1 and R_2 are equal, the operation is equivalent to a test without data movement.

2. The operand specified by the second address is unaltered.

3. Short operands do not alter the low-order half of the register specified by the first address.

Load Complement (LCER) (LCDR)

General Description

◆ The operand in the floating-point register specified by the second address (R_2) is loaded into the floating-point register specified by the first address (R_1) and the sign is changed to the opposite value. The sign and magnitude of the loaded operand determine the condition code.

Format (RR Short)

(LCER) 33	R_1	R_2
0 7	8 11	12 15

(RR Long)

(LCDR) 23	R_1	R_2
0 7	8 11	12 15

Condition Code

◆ 0 — result mantissa is zero.

1 — result mantissa is less than zero.

2 — result mantissa is greater than zero.

3 — not used.

Interrupt Action

◆ Address error:

Specification.

Notes

◆ 1. The exponent and mantissa are unaltered.

2. Short operands do not alter the low-order half of the register specified by the first address.

Load Positive (LPER) (LPDR)

General Description

◆ The operand in the floating-point register specified by the second address (R_2) is loaded into the floating-point register specified by the first address (R_1) and the operand sign is made plus.

Format (RR Short)

(LPER) 30	R_1	R_2
0 7	8 11	12 15

(RR Long)

(LPDR) 20	R_1	R_2
0 7	8 11	12 15

Condition Code

◆ 0 — result mantissa is zero.

1 — not used.

2 — result mantissa is greater than zero.

3 — not used.

Interrupt Action

◆ Address error:

Specification.

Notes

◆ 1. The exponent and mantissa are unaltered.

2. Short operands do not alter the low-order half of the register specified by the first address.

Load Negative (LNER) (LNDR)

General Description

◆ The operand in the floating-point register specified by the second address (R_2) is loaded into the floating-point register specified by the first address (R_1) and the operand sign is made minus.

Format (RR Short)

(LNER) 31	R_1	R_2
0 7	8 11	12 15

(RR Long)

(LNDR) 21	R_1	R_2
0 7	8 11	12 15

Condition Code

◆ 0 — result mantissa is zero.

1 — result mantissa is less than zero.

2 — not used.

3 — not used.

Interrupt Action

◆ Address error:

Specification.

Notes

◆ 1. The exponent and mantissa are unaltered.

2. Short operands do not alter the low-order half of the register specified by the first address.

Add Normalized (AER) (AE) (ADR) (AD)

General Description

◆ The operand specified by the second address (R_2 or $X_2/B_2/D_2$) is added to the operand in the floating-point register specified by the first address (R_1). The *normalized* sum is loaded into the register specified by the first address. The sign and magnitude of the sum determine the condition code.

Format (RR Short)

(AER) 3A	R_1	R_2
0 7	8 11	12 15

(RX Short)

(AE) 7A	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

(RR Long)

(ADR) 2A	R_1	R_2
0 7	8 11	12 15

(RX Long)

(AD) 6A	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ 0 — result mantissa is zero.

1 — result mantissa is less than zero.

2 — result mantissa is greater than zero.

3 — result exponent overflows.

Interrupt Action

◆ Address error:

Addressing (RX format).

Specification.

Significance error.

Exponent overflow.

Exponent underflow.

Notes

◆ 1. To perform normalized addition, the computer must scale the two operands. Scaling consists of comparing the exponents of the two operands. If they do not agree, the mantissa with the smaller exponent operand is shifted right. Its exponent is increased by one for each digit right-shifted, until the two exponents agree. Then, the mantissas are added algebraically to form an intermediate sum. If an overflow carry occurs, the intermediate sum is right-shifted one digit and its exponent is increased by one. If this causes an overflow, an exponent overflow interrupt condition occurs.

For short operands, the intermediate sum consists of seven hexadecimal digits and a possible carry. The low-order digit is the guard digit which is retained from the mantissa which is shifted right. Only one guard digit participates in the mantissa addition. The guard digit is zero if no shift occurs.

Notes
(Cont'd)

For long operands, the intermediate sum consists of fourteen hexadecimal digits and a possible carry. No guard digit is retained.

2. After addition, the intermediate sum is left-shifted until all high-order zero hexadecimal digits have been eliminated. The vacated low-order digits are made zero and the exponent is decremented by one for each zero digit shifted. If no left-shift takes place, the intermediate sum is truncated to the proper mantissa length. If the exponent underflows (exceeds −64) during normalization, the floating-point number is made true zero and an exponent underflow interrupt occurs.
3. No normalization is performed when the intermediate sum is zero. The sum mantissa is unaltered and a significance error interrupt occurs. If a significance error interrupt is prohibited by the interrupt mask, the quantity is made true zero and a significance error interrupt does not occur.
4. Initial operands need not be in normalized form.
5. The sign of the sum is determined by the rules of algebra. A zero sum is always plus.
6. Short operands do not alter the low-order halves of the registers specified by the address fields.

Add Unnormalized (AUR) (AU) (AWR) (AW)

General Description

◆ The operand specified by the second address (R_2 or $X_2/B_2/D_2$) is added to the operand in the floating-point register specified by the first address (R_1). The *unnormalized* sum is loaded into the register specified by the first address. The sign and magnitude of the loaded sum determine the condition code.

Format (RR Short)

(AUR) 3E	R_1	R_2
0 7	8 11	12 15

(RX Short)

(AU) 7E	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

(RR Long)

(AWR) 2E	R_1	R_2
0 7	8 11	12 15

(RX Long)

(AW) 6E	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ 0 — result mantissa is zero.

1 — result mantissa is less than zero.

2 — result mantissa is greater than zero.

3 — result exponent overflows.

Interrupt Action

◆ Address error:

Addressing (RX format).

Specification.

Exponent overflow.

Significance.

Notes

◆ 1. The Add Unnormalized is similar to the Add Normalized, except that the sum is not normalized by this instruction and exponent underflow cannot occur.

Subtract Normalized (SER) (SE) (SDR) (SD)

General Description

◆ The operand specified by the second address (R_2 or $X_2/B_2/D_2$) is subtracted from the operand in the floating-point register specified by the first address (R_1). The *normalized* difference is loaded into the register specified by the first address. The sign and magnitude of the difference determine the condition code.

Format (RR Short)

(SER) 3B	R_1	R_2
0 7	8 11	12 15

(RX Short)

(SE) 7B	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

(RR Long)

(SDR) 2B	R_1	R_2
0 7	8 11	12 15

(RX Long)

(SD) 6B	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ 0 — result mantissa is zero.

1 — result mantissa is less than zero.

2 — result mantissa is greater than zero.

3 — result exponent overflows.

Interrupt Action

◆ Address error:

Addressing (RX format).

Specification.

Significance error.

Exponent overflow.

Exponent underflow.

Notes

◆ 1. The Subtract Normalized is the same as the Add Normalized, except that the sign of the second operand is changed to the opposite value before addition. A zero difference is always positive.

Subtract Unnormalized (SUR) (SU) (SWR) (SW)

General Description

◆ The operand specified by the second address (R_2 or $X_2/B_2/D_2$) is subtracted from the operand in the floating-point register specified by the first address (R_1). The *unnormalized* difference is loaded into the register specified by the first address. The sign and magnitude of the difference determine the condition code.

Format (RR Short)

(SUR) 3F	R_1	R_2
0 7	8 11	12 15

(RX Short)

(SU) 7F	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

(RR Long)

(SWR) 2F	R_1	R_2
0 7	8 11	12 15

(RX Long)

(SW) 6F	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ 0 — result mantissa is zero.

1 — result mantissa is less than zero.

2 — result mantissa is greater than zero.

3 — result exponent overflows.

Interrupt Action

◆ Address error:

Addressing (RX format).

Specification.

Significance error.

Exponent overflow.

Notes

◆ 1. Subtract Unnormalized differs from Subtract Normalized only in that the difference is not normalized before it is loaded into the result register.

2. Exponent underflow cannot occur.

Compare (CER) (CE) (CDR) (CD)

General Description

◆ The operand in the floating-point register specified by the first address (R_1) is algebraically compared to the operand specified by the second address (R_2 or $X_2/B_2/D_2$). The result determines the condition code.

Format (RR Short)

(CER) 39	R_1	R_2
0 7	8 11	12 15

(RX Short)

(CE) 79	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

(RR Long)

(CDR) 29	R_1	R_2
0 7	8 11	12 15

(RX Long)

(CD) 69	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ 0 — operands are equal.

1 — operand specified by the first address is less than the one specified by the second address.

2 — operand specified by the first address is greater than the one specified by the second address.

3 — not used.

Interrupt Action

◆ Address error:

Addressing (RX format).

Specification.

Notes

◆ 1. Comparison takes into account the sign, exponent, and mantissa of each number. Exponent inequality is not decisive for magnitude determination since the mantissas may have different numbers of leading zeros. The operands are scaled, as in Subtract Normalized, and if the mantissa of each operand is zero, the numbers are considered equal regardless of the sign and exponent.

2. Both operands are unaltered.

Halve (HER) (HDR)

General Description

◆ The operand in the floating-point register specified by the second address (R_2) is divided by two. The quotient is loaded into the floating-point register specified by the first address (R_1).

Format (RR Short)

(HER) 34	R_1	R_2
0 7	8 11	12 15

(RR Long)

(HDR) 24	R_1	R_2
0 7	8 11	12 15

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:
Specification.

Notes

◆ 1. The difference between the Halve instruction and a Divide instruction with a divisor of two, is that no normalization and no zero mantissa testing takes place. The sign and exponent are unaltered and the mantissa is shifted right one bit.

2. Short operands do not alter the low-order half of the result register.

Store (STE) (STD)

General Description

◆ The contents of the floating-point register specified by the first address (R_1) are stored in the main memory location specified by the second address ($X_2/B_2/D_2$).

Format (RX Short)

(STE) 70	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

(RX Long)

(STD) 60	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:
Addressing.
Specification.
Protection.

Notes

◆ 1. The first operand is unaltered.
2. Short operands do not alter the low-order half of the register specified by the second address.

Multiply (MER) (ME) (MDR) (MD)

General Description

◆ The operand in the floating-point register specified by the first address (R_1) is multiplied by the operand specified by the second address (R_2 or $X_2/B_2/D_3$). The *normalized* product is loaded into the register specified by the first address.

Format (RR Short)

(MER) 3C	R_1	R_2
0 7	8 11	12 15

(RX Short)

(ME) 7C	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

(RR Long)

(MDR) 2C	R_1	R_2
0 7	8 11	12 15

(RX Long)

(MD) 6C	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing (RX format).

Specification.

Exponent overflow.

Exponent underflow.

Notes

◆ 1. The exponents of the two operands are added, and the sum is reduced by 64 to form an intermediate exponent. The mantissas are normalized as described in the Add Normalize instruction, and multiplied to form an intermediate mantissa. The intermediate mantissa is then normalized (reducing its exponent by one for each digit left shifted) to form the final product.

2. The sign of the product is determined by the rules of algebra.

3. If the product mantissa is zero, the final product is made true zero.

4. If the final product exponent is greater than 127, an exponent overflow interrupt occurs.

5. If final product exponent is less than zero, an exponent underflow interrupt occurs.

6. For short operands, the low-order half of the register specified by the first address *is used* in the calculation of the intermediate mantissa. The product mantissa has the full 14 digits as in the long format and the two low-order digits are always zero.

Divide (DER) (DE) (DDR) (DD)

General Description

◆ The operand (dividend) in the floating-point register specified by the first address (R_1) is divided by the operand divisor specified by the second address (R_2 or $X_2/B_2/D_3$). The *normalized* quotient is stored in the register specified by the first address. The remainder is not retained.

Format (RR Short)

(DER) 3D	R_1	R_2
0 7	8 11	12 15

(RX Short)

(DE) 7D	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

(RR Long)

(DDR) 2D	R_1	R_2
0 7	8 11	12 15

(RX Long)

(DD) 6D	R_1	X_2	B_2	D_2
0 7	8 11	12 15	16 19	20 31

Condition Code

◆ Unchanged.

Interrupt Action

◆ Address error:

Addressing (RX format).

Specification.

Exponent overflow.

Exponent underflow.

Divide error.

Notes

◆ 1. The exponents of the two operands are subtracted and the difference is increased by 64 to form an intermediate exponent. The mantissas are normalized as described in the Subtract Normalize instruction, and divided to form the mantissa of the intermediate quotient. The intermediate exponent and mantissa are normalized to form a final quotient.

2. If the dividend (first operand) is zero, the quotient is made true zero.

3. If the divisor (second operand) is zero, a divide error interrupt occurs.

4. The sign of the quotient is determined by the rules of algebra.

5. If the final quotient exponent is less than zero, the final quotient is made true zero and an exponent underflow interrupt occurs.

6. If the final quotient exponent exceeds 127, an exponent overflow interrupt occurs.

7. For short operands, the low-order halves of the registers are unaltered.

OPTIONAL FEATURES

FEATURE 5001-46 MEMORY PROTECT

◆ Data in memory can be protected from destruction or intrusion by the erroneous storing or fetching of information during program execution through the optional Memory Protect feature. This feature provides store protection or store and fetch protection for memory blocks of 2,048 bytes each.

Operational Characteristics

◆ Memory protection is accomplished by a five-bit storage key associated with each block of 2,048 bytes of main memory. Whenever data is to be stored or accessed in main memory during the execution of an instruction, the five-bit protection key in the Interrupt Status register for the current program state is compared with the five-bit storage key. During a channel-to-memory data transfer, the protection key (as specified in the channel address word) is compared with the storage key. If the storage and protection keys are equal, or either one is zero, the storage or access of data is completed.

If the storage and protection keys do not match (neither is zero), the execution of an instruction that stores data into memory or accessor data is suppressed or terminated. An address error (protection) interrupt occurs, and the protected memory remains unaltered. If the storage and protection keys mismatch during a channel-to-memory data transfer, the data transfer is terminated and a channel termination interrupt occurs. The protected memory is unaltered and the indication of mismatch is stored in the input/output channel registers in scratch-pad memory for the specified channel.

The storage key can be changed by the privileged instruction Set Storage Key and can be inspected by the privileged instruction Insert Storage Key.

When the Memory Protect feature is not installed and the protection key is non zero, an address error (specification) interrupt occurs.

FEATURE 5002-46 ELAPSED TIME CLOCK

◆ The elapsed time clock is an optional feature available on the 70/46 Processor.

Operational Characteristics

◆ The elapsed time clock occupies a full word beginning at main memory location 80. The word is treated as a signed binary operand and follows the rules of fixed-point arithmetic.

The clock count is performed by decrementing bit positions 21 and 23 every 1/60th of a second (60 cycle processor) or by decrementing bit positions 21 and 22 every 1/50th of a second (50 cycle processor). In either case, the effect is equivalent to reducing the elapsed time clock by one in bit position 23 every 1/300th of a second (every 3.3 milliseconds). When the clock goes from positive to negative, an elapsed time clock interrupt occurs.

Normally, an updated elapsed time clock is available after the completion of each instruction execution. However, when input/output data transmission approaches the limit of main memory capability, or a Read Direct instruction time is excessive, elapsed time clock updating can be skipped.

Operational Characteristics *(Cont'd)*

When an elapsed time clock interrupt occurs, the clock may have been decremented several times before the interrupt takes effect, depending on the execution time of the current instruction.

FEATURE 5019-46 ELAPSED TIME CLOCK

◆ This feature provides an elapsed time clock with a greater resolution than the 5002-46 clock. The 5019-46 clock is decremented at a 1000-cycle rate.

Operational Characteristics

◆ The elapsed time clock count is performed by decrementing the elapsed time clock word at main memory location 80. The word is decremented by $4D_{16}$ every 1002 microseconds. When the clock count word changes from positive to negative, if the applicable program mask bit is set, a program interrupt occurs.

FEATURE 5003-46 DIRECT CONTROL

◆ The Direct Control feature enables one 70/46 processor program to directly signal the programs of from one to five other processors over an interface independent of the input/output channels. The processors directly connected by this feature may be remotely located up to 500 cable feet from the transmitting processor.

Operational Characteristics

◆ Two additional privileged instructions are provided with this option, Write Direct and Read Direct, which initiate the transfer of one byte of control information between processor memories, and which signal the opposite unit (by external interrupt) upon execution of an instruction.

This feature can also initiate initial program loading in a remote processor which is in a stopped state. In this case, the Load Unit Switches on the console of the processor being signaled specify the device from which the loading is to occur and the information byte is ignored.

FEATURE 5040 SELECTOR CHANNEL*

◆ This feature provides two enhanced selector channels for a system maximum of 12 I/O Trunks and the Console Trunk.

FEATURE 5041 SELECTOR CHANNEL*

◆ This features provides three enhanced selector channels for a system maximum of 14 I/O Trunks and the Console Trunk.

FEATURE 5042 SELECTOR CHANNEL*

◆ This feature provides four enhanced selector channels for a system maximum of 16 I/O Trunks and the Console Trunk.

* Only one feature (5040, 5041 or 5042) is permitted on a system.

APPENDICES

APPENDIX A — SUMMARY OF INSTRUCTIONS

Privileged Instructions

Instruction	Op$_{(16)}$	Mnemonic	Format	Interrupt Action	Condition Code	Timing (μsec) (Average and Includes Staticizing) 70/46
Check Channel	9F	CKC	SI	1. Privileged operation.	0 — I/O chan. avail. 1 — Interrupt pending in selector channel. 2 — Selector chan. busy or int. pending or multiplex chan. operating in burst mode. 3 — Inoperable.	Multiplexor = 5.52 Selector = 6.48
Diagnose	83	DIG	SI	1. Privileged operation.	Unaltered.	4.56
Halt Device	9E	HDV	SI	1. Privileged operation.	0 — Not busy. 1 — Standard device byte stored in scratch-pad memory. 2 — Termination accepted. 3 — Inoperable.	Multiplexor = 10.32 + CRT Burst = 5.52 + CRT Selector = 6.00 + CRT
Idle	80	IDL	SI	1. Privileged operation.	Unchanged.	6.00
Insert Storage Key	09	ISK	RR	1. Privileged operation. 2. Operation code trap (if feature not installed). 3. Address error.	Unchanged.	5.28
Load Scratch Pad	D8	LSP	SS	1. Privileged operation. 2. Address error.	Unchanged.	10.56 + 2.88R
Program Control	82	PC	SI	1. Privileged operation. 2. Address error.	CC of state being terminated is stored in P counter. CC of state being initiated used to set CC indicators.	7.44
Read Direct	85	RDD	SI	1. Privileged operation. 2. Operation code trap (if feature not installed). 3. Address error.	Unchanged.	To be supplied.
Set Storage Key	08	SSK	RR	1. Privileged operation. 2. Operation code trap (if feature not installed). 3. Address error.	Unchanged.	5.28

Start Device	9C	SDV	SI	1. Privileged operation.	0 — I/O operation started and channel proceeding. 1 — Status bits stored in scratch-pad. 2 — Busy or interrupt pending. 3 — Inoperable.	Multiplexor = 33.36 + CRT Selector = 27.60 + CRT
Store Scratch-Pad	D0	SSP	SS	1. Privileged operation. 2. Address error.	Unchanged.	11.52 + 2.88R
Test Device	9D	TDV	SI	1. Privileged operation.	0 — Available. 1 — Standard device byte stored in scratch-pad. 2 — Busy or interrupt pending. 3 — Inoperable.	Multiplexor = 8.40 + CRT Selector = 8.88 + CRT
Write Direct	84	WRD	SI	1. Privileged operation. 2. Operation code trap (if feature not installed). 3. Address error.	Unchanged.	To be supplied.
Function Call	9A	FC	SI	1. Privileged operation. 2. Operation code trap. 3. Power failure. 4. Machine check. 5. Addressing. 6. Paging error. 7. Paging queue 8. Others as defined.	Unchanged.	To be supplied.

Special Functions

Load Translation Memory	C0	LTM	SF	1. Addressing. 2. Power Failure. 3. Machine Check. 4. Paging Error. 5. Paging Queue.	Unchanged.	11.28 + (6.72 + 2.88S) N — 0.96 F
Scan Translation Memory and Store	C1	STMS	SF	1. Addressing. 2. Power Failure. 3. Machine Check. 4. Paging Error. 5. Paging Queue.	Unchanged.	11.28 + (6.24 + 2.88S) N + 0.96 (G-A)

Legend: A — number of locations skipped.
F — number of locations filled with zeros.
G — number of G-Bits set (1).

N — number of blocks to be loaded.
R — number of registers specified.
S — number of Halfwords in each Translation Memory Bank.
CRT — channel response time (two microseconds average).

SUMMARY OF INSTRUCTIONS (Cont'd)

Special Functions (Cont'd)

Instruction	Op(16)	Mnemonic	Format	Interrupt Action	Condition Code	Timing (μsec) (Average and Includes Staticizing) 70/46
Store Translation Memory	C4	STM	SF	1. Addressing. 2. Power Failure. 3. Machine Check. 4. Paging Error. 5. Paging Queue.	Unchanged.	12.24 + 2.88M
Load Interval Timer	02	LIT	SF	1. Addressing. 2. Power Failure. 3. Machine Check. 4. Paging Error. 5. Paging Queue.	Unchanged.	8.64
Store Interval Timer	03	SIT	SF	1. Addressing. 2. Power Failure. 3. Machine Check. 4. Paging Error. 5. Paging Queue.	Unchanged.	9.60
Paging Queue and Paging Error Interrupt Service	— 01	—	SF	1. Power Failure. 2. Machine Check.	Unchanged.	26 — RR Instruction or 1st Instruction Address Error. 48 — 2nd Instruction Address Error. 110 — Load Multiple/Store Multiple (no Instruction Address Error). 83 — Other RX/RS/SI Instruction (no Instruction Error Address Error). 75 — 3rd Instruction Address Error (LSP/SSP only). 130 — 3rd Instruction Address Error (other SS Instruction).

Processor State Control Instructions

Set Program Mask	04	SPM	RR	None.	CC set according to GR bits 2, 3 specified by R_1.	2.88
Supervisor Call	0A	SVC	RR	None.	Unchanged.	2.88

Fixed-Point Instructions

Add Halfword	4A	AH	RX	1. Fixed-Point overflow. 2. Address error.	0 — Sum is zero. 1 — Sum is less than zero. 2 — Sum is greater than zero. 3 — Overflow.	7.92
Add Logical	5E	AL	RX	1. Address error.	0 — Sum is zero & no carry. 1 — Sum is not zero & no carry. 2 — Sum is zero with carry. 3 — Sum is not zero with carry.	8.40
	1E	ALR	RR			4.80
Add Word	5A	A	RX	1. Fixed-Point overflow. 2. Address error.	0 — Sum is zero. 1 — Sum is less than zero. 2 — Sum is greater than zero. 3 — Overflow.	8.88
	1A	AR	RR			5.28
Compare Halfword	49	CH	RX	1. Address error.	0 — Operands equal. 1 — First operand low. 2 — First operand high. 3 — Not used.	7.44
Compare Word	59	C	RX	1. Address error.	0 — Operands equal. 1 — First operand low. 2 — First operand high. 3 — Not used.	8.40
	19	CR	RR			4.80
Convert to Binary	4F	CVB	RX	1. Address error. 2. Data error. 3. Divide error.	Unchanged.	91.20
Convert to Decimal	4E	CVD	RX	1. Address error.	Unchanged.	68.88 to 91.92
Divide	5D	D	RX	1. Address error. 2. Divide error.	Unchanged.	94.89
	1D	DR	RR			90.81
Load Complement	13	LCR	RR	1. Fixed-Point overflow.	0 — Result is zero. 1 — Result is less than zero. 2 — Result is greater than zero. 3 — Overflow.	5.28

Legend: M — number of locations stored.

SUMMARY OF INSTRUCTIONS (Cont'd)

Fixed-Point Instructions (Cont'd)

Instruction	Op(16)	Mnemonic	Format	Interrupt Action	Condition Code	Timing (µsec) (Average and Includes Staticizing) 70/46
Load Halfword	48	LH	RX	1. Address error.	Unchanged.	7.92
Load Multiple	98	LM	RS	1. Address error.	Unchanged.	9.60 + 2.88R
Load Negative	11	LNR	RR	None.	0 — Result is zero. 1 — Result is less than zero. 2 — Not used. 3 — Not used.	6.24
Load Positive	10	LPR	RR	1. Fixed-Point overflow.	0 — Result is zero. 1 — Not used. 2 — Result greater than zero. 3 — Overflow.	6.24
Load and Test	12	LTR	RR	None.	0 — Result is zero. 1 — Result is less than zero. 2 — Result is greater than zero. 3 — Not used.	5.28
Load Word	58	L	RX	1. Address error.	Unchanged.	8.88
	18	LR	RR			2.88
Multiply Halfword	4C	MH	RX	1. Address error.	Unchanged.	35.40
Multiply Word	5C	M	RX	1. Address error.	Unchanged.	65.64
	1C	MR	RR			62.52
Shift Left Double	8F	SLDA	RS	1. Fixed-Point overflow. 2. Address error.	0 — Result is zero. 1 — Result is less than zero. 2 — Result is greater than zero. 3 — Overflow.	Under 16 = 11.04 + 0.96 (N) 16 to 31 = 15.12 + 0.96 (N-16) 32 to 47 = 19.20 + 0.96 (N-32) 48 to 63 = 23.28 + 0.96 (N-48)
Shift Right Double	8E	SRDA	RS	1. Address error.	0 — Result is zero. 1 — Result is less than zero. 2 — Result is greater than zero. 3 — Not used.	Under 16 = 9.36 + 0.96 (N) 16 to 31 = 12.48 + 0.96 (N-16) 32 to 47 = 15.60 + 0.96 (N-32) 48 to 63 = 18.72 + 0.96 (N-48)

Shift Left Single	8B	SLA	RS	1. Fixed-Point overflow.	0 — Result is zero. 1 — Result is less than zero. 2 — Result is greater than zero. 3 — Overflow.	Under 16 = 10.08 + 0.48 (N) 16 to 31 = 13.20 + 0.48 (N-16) 32 to 47 = 16.32 + 0.48 (N-32) 48 to 63 = 19.44 + 0.48 (N-48)
Shift Right Single	8A	SRA	RS	None.	0 — Result is zero. 1 — Result is less than zero. 2 — Result is greater than zero. 3 — Not used.	Under 16 = 8.16 + 0.48 (N) 16 to 31 = 10.32 + 0.48 (N-16) 32 to 47 = 12.48 + 0.48 (N-32) 48 to 63 = 12.48 + 0.48 (N-48)
Store Halfword	40	STH	RX	1. Address error.	Unchanged.	5.04
Store Multiple	90	STM	RS	1. Address error.	Unchanged.	9.60 + 2.88R
Store Word	50	ST	RX	1. Address error.	Unchanged.	7.44
Subtract Halfword	4B	SH	RX	1. Fixed-Point overflow. 2. Address error.	0 — Diff. is zero. 1 — Diff. less than zero. 2 — Diff. greater than zero. 3 — Overflow.	7.92
Subtract Logical	5F	SL	RX	1. Address error.	0 — Not used. 1 — Diff. not zero; no carry. 2 — Diff. zero with carry. 3 — Diff. not zero with carry.	8.40
	1F	SLR	RR			4.80
Subtract Word	5B	S	RX	1. Fixed-Point overflow. 2. Address error.	0 — Diff. is zero. 1 — Diff. less than zero. 2 — Diff. greater than zero. 3 — Overflow.	8.88
	1B	SR	RR			5.28

Decimal Arithmetic Instructions

Add Decimal	FA	AP	SS	1. Address error. 2. Data error. 3. Decimal overflow.	0 — Sum is zero. 1 — Sum is less than zero. 2 — Sum is greater than zero. 3 — Overflow.	$15.84 + 1.8L_1 + 0.42L_2$ (Note 1)
Compare Decimal	F9	CP	SS	1. Address error. 2. Data error.	0 — Fields algeb. equal. 1 — 1st operand algeb. less than 2nd operand. 2 — 1st operand algeb. greater than 2nd operand.	$17.28 + 1.08L_1 + 0.42L_2$ (Note 1)

Legend: L_1 — number of bytes in first operand field.
L_2 — number of bytes in second operand field.

N — total number of bits shifted.
R — number of registers specified.

SUMMARY OF INSTRUCTIONS (Cont'd)

Decimal Arithmetic Instructions (Cont'd)

Instruction	Op$_{(16)}$	Mnemonic	Format	Interrupt Action	Condition Code	Timing (μsec) (Average and Includes Staticizing) 70/46
Divide Decimal	FD	DP	SS	1. Address error. 2. Data error. 3. Decimal divide error.	Unchanged.	$26.81 + 36.71L_1 - 35.14L_2 +$ $5.40L_2\ (L_1 - L_2)$
Move with Offset	F1	MVO	SS	1. Address error.	Unchanged.	$11.52 + 1.92L_1 + 0.96L_2$
Multiply Decimal	FC	MP	SS	1. Address error. 2. Data error.	Unchanged.	$28.97 + 16.96L_1 - 14.35L_2 +$ $2.34L_2\ (L_1 - L_2)$
Pack	F2	PACK	SS	1. Address error.	Unchanged.	$9.36 + 1.92L_1 + 0.96L_2$
Subtract Decimal	FB	SP	SS	1. Address error. 2. Data error. 3. Decimal overflow.	0 — Diff. is zero. 1 — Diff. is less than zero. 2 — Diff. is greater than zero. 3 — Overflow.	$15.84 + 1.80L_1 + 0.42L_2$ (Note 1)
Unpack	F3	UNPK	SS	1. Address error.	Unchanged.	$10.38 + 0.96L_1 + 0.90L_2$
Zero and Add	F8	ZAP	SS	1. Address error. 2. Data error. 3. Decimal overflow.	0 — Result is zero. 1 — Result is less than zero. 2 — Result is greater than zero. 3 — Overflow.	$15.96 + 1.08L_1 + 0.42L_2$ (Note 1)

Logical Instructions

And	54	N	RX	1. Address error.	0 — Result is zero. 1 — Result is not zero. 2 — Not used. 3 — Not used.	8.40
	D4	NC	SS			12.07+ 2.22L
	94	NI	SI			6.96
	14	NR	RR			5.28

Compare Logical	55	CL	RX	1. Address error.	0 — Operands equal. 1 — 1st operand less than 2nd operand. 2 — 1st operand greater than 2nd operand. 3 — Not used.	8.40
	D5	CLC	SS			$12.32 + 1.44B$ (Note 2)
	95	CLI	SI			6.0
	15	CLR	RR			4.8
Edit	DE	ED	SS	1. Address error. 2. Data error.	0 — Indicates zero source field whether or not signif. is established. 1 — Non-zero result field with signif. established to indicate less than zero. 2 — Non-zero result field with no signif. established to indicate greater than zero. 3 — Not used.	$13.44 + 3L_1 + 1.92L2_2 -$ $0.12F - 0.6K$
Edit and Mark	DF	EDMK	SS	1. Address error. 2. Data error.	0 — Indicates zero source field whether or not signif. is established. 1 — Non-zero result field with signif. established to indicate less than zero. 2 — Non-zero result field with no signif. established to indicate greater than zero. 3 —Not used.	$16.32 + 3L_1 + 1.92L_2 -$ $0.12F - 0.6K$
Exclusive Or	57	X	RX	1. Address error.	0 — Result is zero. 1 — Result is not zero. 2 — Not used. 3 — Not used.	8.40
	D7	XC	SS			$12.07 + 2.22L$
	97	XI	SI			6.96
	17	XR	RR			5.28
Insert Character	43	IC	RX	1. Address error.	Unchanged.	5.52
Load Address	41	LA	RX	None.	Unchanged.	7.92

Legend: B — total number of bytes processed. This condition occurs if instruction terminates before the L count is exhausted.

F — total number of field separating symbols in pattern field.

K — number of control characters in pattern field.

L — total number of bytes specified by L field.

L_1 — number of bytes in first operand field.

L_2 — number of bytes in second operand field.

SUMMARY OF INSTRUCTIONS (Cont'd)

Logical Instructions (Cont'd)

Instruction	Op(16)	Mnemonic	Format	Interrupt Action	Condition Code	Timing (μsec) (Average and Includes Staticizing) 70/46
Move	D2	MVC	SS	1. Address error.	Unchanged.	12.06 + 1.44L
	92	MVI	SI			5.04
Move Numerics	D1	MVN	SS	1. Address error.	Unchanged.	13.02 + 2.22L
Move Zones	D3	MVZ	SS	1. Address error.	Unchanged.	13.02 + 2.22L
Or	56	O	RX	1. Address error.	0 — Result is zero. 1 — Result is not zero. 2 — Not used. 3 — Not used.	8.40
	D6	OC	SS			12.07 + 2.22L
	96	OI	SI			6.96
	16	OR	RR			5.28
Shift Left Single Logical	89	SLL	RS	None.	Unchanged.	Under 16 = 7.92 + 0.48 (N) 16 to 31 = 11.04 + 0.48 (N-16) 32 to 47 = 14.16 + 0.48 (N-32) 48 to 63 = 17.28 + 0.48 (N-48)
Shift Right Single Logical	88	SRL	RS	None.	Unchanged.	Under 16 = 8.88 + 0.48 (N) 16 to 31 = 11.04 + 0.48 (N-16) 32 to 47 = 13.20 + 0.48 (N-32) 48 to 63 = 13.20 + 0.48 (N-48)
Shift Left Double Logical	8D	SLDL	RS	1. Address Error.	Unchanged.	Under 16 = 7.68 + 0.96 (N) 16 to 31 = 11.76 + 0.96 (N-16) 32 to 47 = 15.84 + 0.96 (N-32) 48 to 63 = 19.92 + 0.96 (N-48)
Shift Right Double Logical	8C	SRDL	RS	1. Address Error.	Unchanged.	Under 16 = 7.44 + 0.96 (N) 16 to 31 = 10.56 + 0.96 (N-16) 32 to 47 = 13.68 + 0.96 (N-32) 48 to 63 = 16.80 + 0.96 (N-48)
Store Character	42	STC	RX	1. Address Error.	Unchanged.	5.04

Test Under Mask	91	TM	SI	1. Address Error.	0 — Selected bits all zero, or mask all zero. 1 — Selected bits mixed zero and one. 2 — Not used. 3 — Selected bits all one.	6.48
Translate	DC	TR	SS	1. Address Error.	Unchanged.	9.84 + 5.04L
Translate and Test	DD	TRT	SS	1. Address Error.	0 — All accessed function bytes all zeros. 1 — Non-zero function byte encountered. 2 — Last function byte non-zero. 3 — Not used.	14.64 + 4.08B
Test and Set	93	TS	SI	1. Machine check. 2. Addressing. 3. Power failure.	0 — Leftmost bit of byte specified is zero. 1 — Leftmost bit of byte specified is one.	6.96

Branching Instructions

Branch and Link	45	BAL	RX	None.	Unchanged.	5.52
	05	BALR	RR			Branch = 4.80 No Branch = 3.84
Branch on Condition	47	BC	RX	None.	Unchanged.	Branch = 4.56· No Branch = 4.56
	07	BCR	RR			Branch = 3.84 No Branch = 3.36
Branch on Count	46	BCT	RX	None.	Unchanged.	Branch = 7.92 No Branch = 6.96
	06	BCTR	RR			Branch = 5.76 No Branch = 5.28
Branch on Index High	86	BXH	RS	None.	Unchanged.	Branch = 11.60 No Branch = 11.12

Legend: B — total number of bytes processed. This condition occurs if instruction terminates before L count is exhausted.

L — total number of bytes specified by L field.
N — number of bits shifted.

SUMMARY OF INSTRUCTIONS (Cont'd)

Branching Instructions (Cont'd)

Instruction	Op(16)	Mnemonic	Format	Interrupt Action	Condition Code	Timing (μsec) (Average and Includes Staticizing) 70/46
Branch on Index Low or Equal	87	BXLE	RS	None.	Unchanged.	Branch = 11.60 No Branch = 11.60
Execute	44	EX	RX	1. Address error.	May be set by instruction being modified and executed.	6.96 + EX

Floating-Point Arithmetic Instructions

Add Normalized (Long)	6A	AD	RX	1. Address error. 2. Significance error. 3. Exponent overflow. 4. Exponent underflow.	0 — Result mantissa zero. 1 — Result mantissa less than zero. 2 — Result mantissa greater than zero. 3 — Result exponent overflow.	27.69
	2A	ADR	RR			22.63
Add Normalized (Short)	7A	AE	RX			19.20
	3A	AER	RR			16.08
Add Unnormalized (Long)	6E	AW	RX	1. Address error. 2. Significance error. 3. Exponent overflow.	0 — Result mantissa zero. 1 — Result mantissa less than zero. 2 — Result mantissa greater than zero. 3 — Result exponent overflow.	26.81
	2E	AWR	RR			21.77
Add Unnormalized (Short)	7E	AU	RX			18.96
	3E	AUR	RR			15.84
Compare (Long)	69	CD	RX	1. Address error.	0 — Operands equal. 1 — Operand specified by 1st address low. 2 — Operand specified by 1st address high. 3 — Not used.	23.52
	29	CDR	RR			18.48
Compare (Short)	79	CE	RX			15.36
	39	CER	RR			12.24
Divide (Long)	6D	DD	RX	1. Address error. 2. Exponent overflow. 3. Exponent underflow. 4. Divide error.	Unchanged.	280.27
	2D	DDR	RR			275.68
Divide (Short)	7D	DE	RX			83.00
	3D	DER	RR			79.88

Halve (Long)	24	HDR	RR	1. Address error.	Unchanged.	8.16
Halve (Short)	34	HER	RR			6.00
Load Complement (Long)	23	LCDR	RR	1. Address error.	0 — Result mantissa zero. 1 — Result mantissa less than zero. 2 — Result mantissa greater than zero. 3 — Not used.	8.16
Load Complement (Short)	33	LCER	RR			6.00
Load (Long)	68	LD	RX	1. Address error.	Unchanged.	13.68
	28	LDR	RR			8.64
Load (Short)	78	LE	RX			9.84
	38	LER	RR			6.72
Load Negative (Long)	21	LNDR	RR	1. Address error.	0 — Result mantissa zero. 1 — Result mantissa less than zero. 2 — Not used. 3 — Not used.	7.68
Load Negative (Short)	31	LNER	RR			5.52
Load Positive (Long)	20	LPDR	RR	1. Address error.	0 — Kesult mantissa zero. 1 — Not used. 2 — Result mantissa greater than zero. 3 — Not used.	7.68
Load Positive (Short)	30	LPER	RR			5.52
Load and Test (Long)	22	LTDR	RR	1. Address error.	0 — Result mantissa zero. 1 — Result mantissa less than zero. 2 — Result mantissa greater than zero. 3 — Not used.	8.16
Load and Test (Short)	32	LTER	RR			6.00
Multiply (Long)	6C	MD	RX	1. Address error. 2. Exponent overflow. 3. Exponent underflow.	Unchanged.	186.55
	2C	MDR	RR			181.51
Multiply (Short)	7C	ME	RX			49.42
	3C	MER	RR			46.40
Store (Long)	60	STD	RX	1. Address error.	Unchanged.	11.28
Store (Short)	70	STE	RX			8.40

Legend: EX — object instruction execution time.

SUMMARY OF INSTRUCTIONS (Cont'd)

Floating-Point Arithmetic Instructions (Cont'd)

Instruction	Op$_{(16)}$	Mnemonic	Format	Interrupt Action	Condition Code	Timing (μsec) (Average and Includes Staticizing) 70/46
Subtract Normalized (Long)	6B	SD	RX	1. Address error. 2. Significance error. 3. Exponent overflow. 4. Exponent underflow.	0 — Result mantissa zero. 1 — Result mantissa less than zero. 2 — Result mantissa greater than zero. 3 — Result exponent overflow.	27.69
	2B	SDR	RR			22.63
Subtract Normalized (Short)	7B	SE	RX			19.20
	3B	SER	RR			16.08
Subtract Unnormalized (Long)	6F	SW	RX	1. Address error. 2. Significance error. 3. Exponent overflow.	0 — Result mantissa zero. 1 — Result mantissa less than zero. 2 — Result mantissa greater than zero. 3 — Result exponent overflow.	26.81
	2F	SWR	RR			21.77
Subtract Unnormalized (Short)	7F	SU	RX			18.96
	3F	SUR	RR			15.84

Notes:
1. Time for $L_1 > L_2$ and no End Around Carry. Additional time must be added if $L_2 > L_1$ or End Around Carry.
2. If the two fields are equal $B = L$ since all bytes must be examined. If the fields are unequal the instruction is terminated upon examining the first pair of unequal bytes. In this case, B is less than L.
3. Each 127 words stored or loaded requires an extra 0.96 microseconds to effect wrap around.
4. If Debug Mode, 19.20 + EX.

APPENDIX B

LIST OF PROGRAM INTERRUPTS

Priority	Condition	State Initiated	Explanation	Timing (If Interrupt Taken) 70/46
1	Power Failure	4	Power failure in processor or memory.	11.64
2	Machine Check	4	Parity error or equipment malfunction.	11.64
3	External Signal 1	3	Signal received on one of the six external lines associated with the direct-control feature.	11.64
4	External Signal 2	3		11.64
5	External Signal 3	3		11.64
6	External Signal 4	3		11.64
7	External Signal 5	3		11.64
8	External Signal 6	3		11.64
9	Interval Timer	3	Lapse of Interval Timer.	14.64
10	Selector 1 Terminate	3	A device on the associated selector or multiplexor channel has terminated.	18.86 + CRT
11	Selector 2 Terminate	3		18.86 + CRT
12	Selector 3 Terminate 70/46	3		18.86 + CRT
13	Not Specified	3		
14	Not Specified	3		
15	Not Specified	3		
16	Multiplexor Terminate	3		25.90 + CRT
17	Elapsed Time Clock	3	Elapsed time count has expired.	13.08
18	Console Request	3	Manual request for interrupt by the operator.	13.08
19	Paging Error	3	Improper use of Virtual Memory.	15.60
20	Paging Queue	3	Translation Table Interrupt.	15.60
21	Supervisor Call	3	Result of execution of Supervisor Call instruction to utilize programmed routines.	13.08
22	Privileged Operation	3	Privileged instruction attempted in non-privileged mode.	13.08
23	Op-Code Trap	3	Op Code attempted which is invalid for this model.	13.08
24	Address Error	3	Invalid address, specification, or memory protect violation.	13.08
25	Data Error	3	Sign of operand incorrect in decimal arithmetic and editing, or incorrect field overlap.	13.08

LIST OF PROGRAM INTERRUPTS (Cont'd)

Priority	Condition	State Initiated	Explanation	Timing (If Interrupt Taken) 70/46
26	Exponent Overflow	3	Result characteristic of floating-point operation is greater than 127.	13.08
27	Divide Error	3	Rules pertaining to Divide instruction have been violated.	13.08
28	Significance Error	3	Result of floating-point or subtract has zero fraction.	13.08
29	Exponent Underflow	3	Result characteristic of floating-point operation is less than zero.	13.08
30	Decimal Overflow	3	Result field is too small to contain the result of a decimal operation.	13.08
31	Fixed-Point Overflow	3	High-order carry or high-order significant bits lost in fixed-point operation.	13.08
32	Test Mode	3	Allows program control over processor during program testing.	13.08
Priorities 1 thru 16			If interrupt not taken.	5.76
Priorities 17 thru 32			If interrupt not taken.	5.76

APPENDIX C

INPUT/OUTPUT SERVICE REQUEST

I/O Channel Service Times and Processing Mode	Spectra 70/46 Processor Times (Microseconds)
	70/46
A. *Selector Basic Times*	
1. Read/Write (Scratch Pad)	NA
2. Read/Write (Main Memory Normal)	1.44 (2 bytes)
3. Read/Write (Main Memory less than 4-bytes; CCW specified)	NA
4. End Service	3.12
B. *Selector Add'l Times* (to be added to above times)	
1. Data Chaining	7.20 (read) 8.16 (write)[c]
2. Command Chaining	3.04 (read) 6.0 (write)[c]
3. Transfer in Channel	3.84
4. Status Modifier	1.92
5. Incomplete Read (Device Terminated in middle of word; not indicated by CCW)	NA
C. *Multiplexor Basic Times*	
1. Read/Write (Multiplex Mode; non-catch-up)	13.92[a]
2. Read/Write (Burst Mode)	1.92[a]
3. END SERVICE (Mux Mode)	10.08[a, d]
4. END SERVICE (Burst Mode)	7.68[a, d]
D. *Multiplexor Add'l Times* (to be added to above items)	
1. Data Chaining Mux Mode	12.96
2. Data Chaining Burst Mode	13.44
3. Command Chaining Mux Mode	12.96[b]
4. Command Chaining Burst Mode	6.72[b]
5. Transfer in Channel	4.32
6. Status Modifier	1.92
7. Catch-up each additional byte	1.92
E. *Processing Mode Times*	
1. START I/O (addr. the selector)	32.64[b]
2. START I/O (addr. the multiplexor)	39.6[b]
3. Multiplexor Program Interrupt	25.90[b]
4. Selector Program Interrupt	18.86[b]

NOTES:

a. Because of odd/even ROM addressing, banking may result in loss of 2 EO cycles (or 0.96 μs); which will probably occur both at the beginning and at the ending of the service; randomly this is a 50% change, or an additional time of 0.48 μs.

b. Times are for processor servicing and are extended by Channel Response Time (CRT).

c. The additional time required for the command and data chaining for write commands is needed for buffer loading in the selector channel on the 70/46.

d. Buffered devices require two (2) end services for multiplexor operations. If the multiplexor is operating in the Burst Mode, the first end service is done in the Burst Mode and the second end service is done in the normal Mux Mode.

APPENDIX D

EXTENDED BINARY-CODED-DECIMAL INTERCHANGE CODE (EBCDIC)

0123 ◄—— 4567 ——►

HEX →		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
↓		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	NUL				PF	HT	LC	DEL								
1	0001					RES	NL	BS	IL								
2	0010					BYP	LF	EOB	PRE			SM					
3	0011					PN	RS	UC	EOT								
4	0100	SPACE										¢	.	<	(	+	|
5	0101	&										!	$	*	)	;	¬
6	0110	–	/									∧	,	%	_	>	?
7	0111											:	#	@	'	=	"
8	1000		a	b	c	d	e	f	g	h	i						
9	1001		j	k	l	m	n	o	p	q	r						
A	1010			s	t	u	v	w	x	y	z						
B	1011																
C	1100		A	B	C	D	E	F	G	H	I						
D	1101		J	K	L	M	N	O	P	Q	R						
E	1110			S	T	U	V	W	X	Y	Z						
F	1111	0	1	2	3	4	5	6	7	8	9						⌑

Bit Positions: 0 1 2 3 4 5 6 7

Significance: 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

Control Characters:

NUL	— All Zero-Bits	BYP	— Bypass
PF	— Punch Off	LF	— Line Feed
HT	— Horizontal Tab	EOB	— End of Block
LC	— Lower Case	PRE	— Prefix
DEL	— Delete	SM	— Set Mode
RES	— Restore	PN	— Punch On
NL	— New Line	RS	— Reader Stop
BS	— Backspace	UC	— Upper Case
IL	— Idle	EOT	— End of Transmission

APPENDIX E

USA STANDARD CODE FOR INFORMATION INTERCHANGE (USASCII)
(Extended to 8 Bits)

76X5 ←——————— 4321 ———————→

HEX	→	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
↓		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	0001	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	0010																
3	0011																
4	0100	SP	!	"	#	$	%	&	'	(	)	*	+	,	—	.	/
5	0101	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
6	0110																
7	0111																
8	1000																
9	1001																
A	1010	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
B	1011	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	___
C	1100																
D	1101																
E	1110	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
F	1111	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Bit Positions: 7 6 X 5 4 3 2 1

Significance: 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

Control Characters:

NUL — Null
SOH — Start of Heading (CC)
STX — Start of Text (CC)
ETX — End of Text (CC)
EOT — End of Transmission (CC)
ENQ — Enquiry (CC)
ACK — Acknowledge (CC)
BEL — Bell (audible or attention signal)
BS — Backspace (FE)
HT — Horizontal Tabulation (punch card skip) (FE)
LF — Line Feed (FE)
VT — Vertical Tabulation (FE)
FF — Form Feed (FE)
CR — Carriage Return (FE)
SO — Shift Out
SI — Shift In
DLE — Data Link Escape (CC)
DC1 — Device Control 1
DC2 — Device Control 2
DC3 — Device Control 3
DC4 — Device Control 4 (stop)
NAK — Negative Acknowledge (CC)
SYN — Synchronous Idle (CC)
ETB — End of Transmission Block (CC)
CAN — Cancel
EM — End of Medium
SUB — Substitute
ESC — Escape
FS — File Separator (IS)
GS — Group Separator (IS)
RS — Record Separator (IS)
US — Unit Separator (IS)
DEL — Delete

SP — Space (normally non-printing)

(CC) — Communication Control
(FE) — Format Effector
(IS) — Information Separator

APPENDIX F

CHARACTER CODES

Decimal	Hexadecimal	EBCDIC	Character Set Punch Combination	Printer Graphics
0	00	0000 0000	12,0,9,8,1	
1	01	0000 0001	12,9,1	
2	02	0000 0010	12,9,2	
3	03	0000 0011	12,9,3	
4	04	0000 0100	12,9,4	
5	05	0000 0101	12,9,5	
6	06	0000 0110	12,9,6	
7	07	0000 0111	12,9,7	
8	08	0000 1000	12,9,8	
9	09	0000 1001	12,9,8,1	
10	0A	0000 1010	12,9,8,2	
11	0B	0000 1011	12,9,8,3	
12	0C	0000 1100	12,9,8,4	
13	0D	0000 1101	12,9,8,5	
14	0E	0000 1110	12,9,8,6	
15	0F	0000 1111	12,9,8,7	
16	10	0001 0000	12,11,9,8,1	
17	11	0001 0001	11,9,1	
18	12	0001 0010	11,9,2	
19	13	0001 0011	11,9,3	
20	14	0001 0100	11,9,4	
21	15	0001 0101	11,9,5	
22	16	0001 0110	11,9,6	
23	17	0001 0111	11,9,7	
24	18	0001 1000	11,9,8	
25	19	0001 1001	11,9,8,1	
26	1A	0001 1010	11,9,8,2	
27	1B	0001 1011	11,9,8,3	
28	1C	0001 1100	11,9,8,4	
29	1D	0001 1101	11,9,8,5	
30	1E	0001 1110	11,9,8,6	
31	1F	0001 1111	11,9,8,7	
32	20	0010 0000	11,0,9,8,1	
33	21	0010 0001	0,9,1	
34	22	0010 0010	0,9,2	
35	23	0010 0011	0,9,3	
36	24	0010 0100	0,9,4	
37	25	0010 0101	0,9,5	
38	26	0010 0110	0,9,6	
39	27	0010 0111	0,9,7	
40	28	0010 1000	0,9,8	
41	29	0010 1001	0,9,8,1	
42	2A	0010 1010	0,9,8,2	
43	2B	0010 1011	0,9,8,3	
44	2C	0010 1100	0,9,8,4	
45	2D	0010 1101	0,9,8,5	
46	2E	0010 1110	0,9,8,6	
47	2F	0010 1111	0,9,8,7	
48	30	0011 0000	12,11,0,9,8,1	
49	31	0011 0001	9,1	
50	32	0011 0010	9,2	
51	33	0011 0011	9,3	
52	34	0011 0100	9,4	
53	35	0011 0101	9,5	
54	36	0011 0110	9,6	

CHARACTER CODES (Cont.)

Decimal	Hexadecimal	EBCDIC	Character Set Punch Combination	Printer Graphics
55	37	0011 0111	9,7	
56	38	0011 1000	9,8	
57	39	0011 1001	9,8,1	
58	3A	0011 1010	9,8,2	
59	3B	0011 1011	9,8,3	
60	3C	0011 1100	9,8,4	
61	3D	0011 1101	9,8,5	
62	3E	0011 1110	9,8,6	
63	3F	0011 1111	9,8,7	
64	40	0100 0000		Space
65	41	0100 0001	12,0,9,1	
66	42	0100 0010	12,0,9,2	
67	43	0100 0011	12,0,9,3	
68	44	0100 0100	12,0,9,4	
69	45	0100 0101	12,0,9,5	
70	46	0100 0110	12,0,9,6	
71	47	0100 0111	12,0,9,7	
72	48	0100 1000	12,0,9,8	
73	49	0100 1001	12,8,1	
74	4A	0100 1010	12,8,2	¢ (cents)
75	4B	0100 1011	12,8,3	. (period)
76	4C	0100 1100	12,8,4	< (Less than)
77	4D	0100 1101	12,8,5	((open parenthesis)
78	4E	0100 1110	12,8,6	+ (plus)
79	4F	0100 1111	12,8,7	| (vertical)
80	50	0101 0000	12	& (ampersand)
81	51	0101 0001	12,11,9,1	
82	52	0101 0010	12,11,9,2	
83	53	0101 0011	12,11,9,3	
84	54	0101 0100	12,11,9,4	
85	55	0101 0101	12,11,9,5	
86	56	0101 0110	12,11,9,6	
87	57	0101 0111	12,11,9,7	
88	58	0101 1000	12,11,9,8	
89	59	0101 1001	11,8,1	
90	5A	0101 1010	11,8,2	! (exclamation)
91	5B	0101 1011	11,8,3	$ (dollar sign)
92	5C	0101 1100	11,8,4	* (asterisk)
93	5D	0101 1101	11,8,5	) (close parenthesis)
94	5E	0101 1110	11,8,6	; (semicolon)
95	5F	0101 1111	11,8,7	¬ (logical NOT)
96	60	0110 0000	11	— (minus)
97	61	0110 0001	0,1	/ (slash)
98	62	0110 0010	11,0,9,2	
99	63	0110 0011	11,0,9,3	
100	64	0110 0100	11,0,9,4	
101	65	0110 0101	11,0,9,5	
102	66	0110 0110	11,0,9,6	
103	67	0110 0111	11,0,9,7	
104	68	0110 1000	11,0,9,8	
105	69	0110 1001	0,8,1	
106	6A	0110 1010	12,11	∧ (logical AND)
107	6B	0110 1011	0,8,3	, (comma)
108	6C	0110 1100	0,8,4	% (percent)
109	6D	0110 1101	0,8,5	__ (underline)

CHARACTER CODES (Cont.)

Decimal	Hexadecimal	EBCDIC*	Character Set Punch Combination	Printer Graphics
110	6E	0110 1110	0,8,6	> (greater than)
111	6F	0110 1111	0,8,7	? (question mark)
112	70	0111 0000	12,11,0	
113	71	0111 0001	12,11,0,9,1	
114	72	0111 0010	12,11,0,9,2	
115	73	0111 0011	12,11,0,9,3	
116	74	0111 0100	12,11,0,9,4	
117	75	0111 0101	12,11,0,9,5	
118	76	0111 0110	12,11,0,9,6	
119	77	0111 0111	12,11,0,9,7	
120	78	0111 1000	12,11,0,9,8	
121	79	0111 1001	8,1	
122	7A	0111 1010	8,2	: (colon)
123	7B	0111 1011	8,3	# (number sign)
124	7C	0111 1100	8,4	@ (at the rate of)
125	7D	0111 1101	8,5	' (apostrophe)
126	7E	0111 1110	8,6	= (equals)
127	7F	0111 1111	8,7	" (quote)
128	80	1000 0000	12,0,8,1	
129	81	1000 0001	12,0,1	
130	82	1000 0010	12,0,2	
131	83	1000 0011	12,0,3	
132	84	1000 0100	12,0,4	
133	85	1000 0101	12,0,5	
134	86	1000 0110	12,0,6	
135	87	1000 0111	12,0,7	
136	88	1000 1000	12,0,8	
137	89	1000 1001	12,0,9	
138	8A	1000 1010	12,0,8,2	
139	8B	1000 1011	12,0,8,3	
140	8C	1000 1100	12,0,8,4	
141	8D	1000 1101	12,0,8,5	
142	8E	1000 1110	12,0,8,6	
143	8F	1000 1111	12,0,8,7	
144	90	1001 0000	12,11,8,1	
145	91	1001 0001	12,11,1	
146	92	1001 0010	12,11,2	
147	93	1001 0011	12,11,3	
148	94	1001 0100	12,11,4	
149	95	1001 0101	12,11,5	
150	96	1001 0110	12,11,6	
151	97	1001 0111	12,11,7	
152	98	1001 1000	12,11,8	
153	99	1001 1001	12,11,9	
154	9A	1001 1010	12,11,8,2	
155	9B	1001 1011	12,11,8,3	
156	9C	1001 1100	12,11,8,4	
157	9D	1001 1101	12,11,8,5	
158	9E	1001 1110	12,11,8,6	
159	9F	1001 1111	12,11,8,7	
160	A0	1010 0000	11,0,8,1	
161	A1	1010 0001	11,0,1	
162	A2	1010 0010	11,0,2	
163	A3	1010 0011	11,0,3	
164	A4	1010 0100	11,0,4	

APPENDIX G

POWERS OF TWO TABLE

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 45
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5

APPENDIX H

HEXADECIMAL-DECIMAL NUMBER CONVERSION

General

◆ The table provides for direct conversion of hexadecimal and decimal numbers in these ranges:

Hexadecimal	*Decimal*
000 to FFF	0000 to 4095

Hexadecimal-Decimal Number Conversion Table

◆ In the table, the decimal value appears at the intersection of the row representing the most significant hexadecimal digits (16^2 and 16^1) and the column representing the least significant hexadecimal digit (16^0).

Example: $C21_{16} = 3105_{10}$

HEX	*0*	*1*	*2*
C0	3072	3073	3074
C1	3088	3089	3090
C2	3104	3105	3106
C3	3120	3121	3122

For numbers outside the range of the table, add the following values to the table figures:

Hexadecimal	*Decimal*	*Hexadecimal*	*Decimal*
1000	4,096	C000	49,152
2000	8,192	D000	53,248
3000	12,288	E000	57,344
4000	16,384	F000	61,440
5000	20,480	10000	65,536
6000	24,576	20000	131,072
7000	28,672	30000	196,608
8000	32,768	40000	262,144
9000	36,864	50000	327,680
A000	40,960	60000	393,216
B000	45,056	70000	458,752

Example: $1C21_{16} = 7201_{10}$

Hexadecimal	*Decimal*
C21	3105
+1000	+4096
1C21	7201

HEXADECIMAL-DECIMAL NUMBER CONVERSION TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
01	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
02	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
03	0048	0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060	0061	0062	0063
04	0064	0065	0066	0067	0068	0069	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079
05	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095
06	0096	0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	0110	0111
07	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127
08	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143
09	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
0A	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175
0B	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191
0C	0192	0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204	0205	0206	0207
0D	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223
0E	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239
0F	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
10	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271
11	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287
12	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300	0301	0302	0303
13	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319
14	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0332	0333	0334	0335
15	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351
16	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367
17	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383
18	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399
19	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415
1A	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431
1B	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447
1C	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463
1D	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479
1E	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495
1F	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
20	0512	0513	0514	0515	0516	0517	0518	0519	0520	0521	0522	0523	0524	0525	0526	0527
21	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543
22	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559
23	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575
24	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591
25	0592	0593	0594	0595	0596	0597	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607
26	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623
27	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639
28	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655
29	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671
2A	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687
2B	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699	0700	0701	0702	0703
2C	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719
2D	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735
2E	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751
2F	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
30	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783
31	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799
32	0800	0801	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815
33	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831
34	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847
35	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863
36	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879
37	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895
38	0896	0897	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911
39	0912	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927
3A	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943
3B	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959
3C	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975
3D	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991
3E	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007
3F	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023

HEXADECIMAL-DECIMAL NUMBER CONVERSION TABLE (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
40	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039
41	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055
42	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071
43	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
44	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
45	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119
46	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
47	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151
48	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167
49	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
4A	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199
4B	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215
4C	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
4D	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247
4E	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
4F	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
50	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295
51	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
52	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327
53	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
54	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359
55	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375
56	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391
57	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407
58	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423
59	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439
5A	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455
5B	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471
5C	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487
5D	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
5E	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519
5F	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
60	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551
61	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567
62	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
63	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599
64	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615
65	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631
66	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647
67	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1662	1663
68	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679
69	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695
6A	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711
6B	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727
6C	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
6D	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759
6E	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775
6F	1776	1777	1778	1779	1780	1781	1782	1783	1784	1785	1786	1787	1788	1789	1790	1791

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
70	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	1803	1804	1805	1806	1807
71	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
72	1824	1825	1826	1827	1828	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839
73	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855
74	1856	1857	1858	1859	1860	1861	1862	1863	1864	1865	1866	1867	1868	1869	1870	1871
75	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887
76	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902	1903
77	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919
78	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935
79	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951
7A	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967
7B	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
7C	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
7D	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
7E	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031
7F	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047

HEXADECIMAL-DECIMAL NUMBER CONVERSION TABLE (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
80	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063
81	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079
82	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095
83	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111
84	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
85	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143
86	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159
87	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175
88	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191
89	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207
8A	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223
8B	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239
8C	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255
8D	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271
8E	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287
8F	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
90	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319
91	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335
92	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351
93	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367
94	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383
95	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399
96	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
97	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431
98	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447
99	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463
9A	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479
9B	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495
9C	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511
9D	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527
9E	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543
9F	2544	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A0	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A1	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A2	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A3	2608	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	2620	2621	2622	2623
A4	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639
A5	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655
A6	2656	2657	2658	2659	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671
A7	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687
A8	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703
A9	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719
AA	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735
AB	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751
AC0	2752	2753	2754	2755	2756	2757	2758	2759	2760	2761	2762	2763	2764	2765	2766	2767
AD0	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783
AE0	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
AF0	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
B0	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
B1	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
B2	2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863
B3	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879
B4	2880	2881	2882	2883	2884	2885	2886	2887	2888	2889	2890	2891	2892	2893	2894	2895
B5	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911
B6	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
B7	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943
B8	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959
B9	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975
BA	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991
BB	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
BC	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023
BD	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039
BE	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055
BF	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071

HEXADECIMAL-DECIMAL NUMBER CONVERSION TABLE (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C0	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
C1	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103
C2	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119
C3	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135
C4	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151
C5	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
C6	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183
C7	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199
C8	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209	3210	3211	3212	3213	3214	3215
C9	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231
CA	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
CB	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263
CC	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279
CD	3280	3281	3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295
CE	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311
CF	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
D0	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343
D1	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359
D2	3360	3361	3362	3363	3364	3365	3366	3367	3368	3369	3370	3371	3372	3373	3374	3375
D3	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391
D4	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
D5	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423
D6	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439
D7	3440	3441	3442	3443	3444	3445	3446	3447	3448	3449	3450	3451	3452	3453	3454	3455
D8	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471
D9	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
DA	3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503
DB	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519
DC	3520	3521	3522	3523	3524	3525	3526	3527	3528	3529	3530	3531	3532	3533	3534	3535
DD	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551
DE	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
DF	3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E0	3584	3585	3586	3587	3588	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599
E1	3600	3601	3602	3603	3604	3605	3606	3607	3608	3609	3610	3611	3612	3613	3614	3615
E2	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631
E3	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
E4	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
E5	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679
E6	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693	3694	3695
E7	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711
E8	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
E9	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743
EA	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759
EB	3760	3761	3762	3763	3764	3765	3766	3767	3768	3769	3770	3771	3772	3773	3774	3775
EC	3776	3777	3778	3779	3780	3781	3782	3783	3784	3785	3786	3787	3788	3789	3790	3791
ED	3792	3793	3794	3795	3796	3797	3798	3799	3800	3801	3802	3803	3804	3805	3806	3807
EE	3808	3809	3810	3811	3812	3813	3814	3815	3816	3817	3818	3819	3820	3821	3822	3823
EF	3824	3825	3826	3827	3828	3829	3830	3831	3832	3833	3834	3835	3836	3837	3838	3839

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
F0	3840	3841	3842	3843	3844	3845	3846	3847	3848	3849	3850	3851	3852	3853	3854	3855
F1	3856	3857	3858	3859	3860	3861	3862	3863	3864	3865	3866	3867	3868	3869	3870	3871
F2	3872	3873	3874	3875	3876	3877	3878	3879	3880	3881	3882	3883	3884	3885	3886	3887
F3	3888	3889	3890	3891	3892	3893	3894	3895	3896	3897	3898	3899	3900	3901	3902	3903
F4	3904	3905	3906	3907	3908	3909	3910	3911	3912	3913	3914	3915	3916	3917	3918	3919
F5	3920	3921	3922	3923	3924	3925	3926	3927	3928	3929	3930	3931	3932	3933	3934	3935
F6	3936	3937	3938	3939	3940	3941	3942	3943	3944	3945	3946	3947	3948	3949	3950	3951
F7	3952	3953	3954	3955	3956	3957	3958	3959	3960	3961	3962	3963	3964	3965	3966	3967
F8	3968	3969	3970	3971	3972	3973	3974	3975	3976	3977	3978	3979	3980	3981	3982	3983
F9	3984	3985	3986	3987	3988	3989	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
FA	4000	4001	4002	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4014	4015
FB	4016	4017	4018	4019	4020	4021	4022	4023	4024	4025	4026	4027	4028	4029	4030	4031
FC	4032	4033	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043	4044	4045	4046	4047
FD	4048	4049	4050	4051	4052	4053	4054	4055	4056	4057	4058	4059	4060	4061	4062	4063
FE	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	4075	4076	4077	4078	4079
FF	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095

APPENDIX I

SCRATCH-PAD MEMORY LAYOUT AND REGISTER ASSIGNMENTS

Hexadecimal	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
0 1	PROCESSOR UTILITY								GENERAL PURPOSE REGISTER NO. 8 P4	GENERAL PURPOSE REGISTER NO. 9 P4	GENERAL PURPOSE REGISTER NO. 10 P4	GENERAL PURPOSE REGISTER NO. 11 P4	INTERRUPT MASK REGISTER P4	INTERRUPT STATUS REGISTER P4	PROGRAM COUNTER P4	GENERAL PURPOSE REGISTER NO. 15 (WEIGHT) P4
2 3	PROCESSOR UTILITY		I/O CHANNEL REGISTERS – MULTIPLEXOR CHANNEL ADDRESS REGISTER	CHANNEL COMMAND REGISTER II	CHANNEL COMMAND REGISTER I	STATUS REGISTER	PROCESSOR UTILITY				I/O CHANNEL REGISTERS – SELECTOR NO. 1 CHANNEL ADDRESS REGISTER	CHANNEL COMMAND REGISTER II	CHANNEL COMMAND REGISTER I	ASSEMBLY STATUS REGISTER	CBA	PROCESSOR UTILITY
4 5	INTERRUPT MASK REGISTER P1	INTERRUPT STATUS REGISTER P1	PROGRAM COUNTER P1	INTERRUPT FLAG REGISTER	INTERRUPT MASK REGISTER P2	INTERRUPT STATUS REGISTER P2	PROGRAM COUNTER P2	GENERAL PURPOSE REGISTER NO. 7 P3	INTERRUPT MASK REGISTER P3	INTERRUPT STATUS REGISTER P3	PROGRAM COUNTER P3	GENERAL PURPOSE REGISTER NO. 11 P3	GENERAL PURPOSE REGISTER NO. 12 P3	GENERAL PURPOSE REGISTER NO. 13 P3	GENERAL PURPOSE REGISTER NO. 14 P3	GENERAL PURPOSE REGISTER NO. 15 (WEIGHT) P3
6 7	PROCESSOR UTILITY		I/O CHANNEL REGISTERS – SELECTOR NO. 2 CHANNEL ADDRESS REGISTER	CHANNEL COMMAND REGISTER II	CHANNEL COMMAND REGISTER I	ASSEMBLY STATUS REGISTER	CBA	PROCESSOR UTILITY			I/O CHANNEL REGISTERS – SELECTOR NO. 3 CHANNEL ADDRESS REGISTER	CHANNEL COMMAND REGISTER II	CHANNEL COMMAND REGISTER I	ASSEMBLY STATUS REGISTER	CBA	PROCESSOR UTILITY
8 9	GENERAL PURPOSE REGISTER NO. 0 P2	GENERAL PURPOSE REGISTER NO. 1 P2	GENERAL PURPOSE REGISTER NO. 2 P2	GENERAL PURPOSE REGISTER NO. 3 P2	GENERAL PURPOSE REGISTER NO. 4 P2	GENERAL PURPOSE REGISTER NO. 5 P2	GENERAL PURPOSE REGISTER NO. 6 P2	GENERAL PURPOSE REGISTER NO. 7 P2	GENERAL PURPOSE REGISTER NO. 8 P2	GENERAL PURPOSE REGISTER NO. 9 P2	GENERAL PURPOSE REGISTER NO. 10 P2	GENERAL PURPOSE REGISTER NO. 11 P2	GENERAL PURPOSE REGISTER NO. 12 P2	GENERAL PURPOSE REGISTER NO. 18 P2	GENERAL PURPOSE REGISTER NO. 14 P2	GENERAL PURPOSE REGISTER NO. 15 P2
A B	PROCESSOR UTILITY		I/O CHANNEL REGISTERS – SELECTOR NO. 4 CHANNEL ADDRESS REGISTER	CHANNEL COMMAND REGISTER II	CHANNEL COMMAND REGISTER I	ASSEMBLY STATUS REGISTER	CBA	PROCESSOR UTILITY								
C D	GENERAL PURPOSE REGISTER NO. 0 P1	GENERAL PURPOSE REGISTER NO. 1 P1	GENERAL PURPOSE REGISTER NO. 2 P1	GENERAL PURPOSE REGISTER NO. 3 P1	GENERAL PURPOSE REGISTER NO. 4 P1	GENERAL PURPOSE REGISTER NO. 5 P1	GENERAL PURPOSE REGISTER NO. 6 P1	GENERAL PURPOSE REGISTER NO. 7 P1	GENERAL PURPOSE REGISTER NO. 8 P1	GENERAL PURPOSE REGISTER NO. 9 P1	GENERAL PURPOSE REGISTER NO. 10 P1	GENERAL PURPOSE REGISTER NO. 11 P1	GENERAL PURPOSE REGISTER NO. 12 P1	GENERAL PURPOSE REGISTER NO. 13 P1	GENERAL PURPOSE REGISTER NO. 14 P1	GENERAL PURPOSE REGISTER NO. 15 P1
E F	FLOATING-POINT REGISTER NO. 0		FLOATING-POINT REGISTER NO. 2		FLOATING-POINT REGISTER NO. 4		FLOATING-POINT REGISTER NO. 6		PROCESSOR UTILITY							

* Word Address is in Hexadecimal; e.g., 2A Program Counter P3.