Being revised

PDP-X Technical Memorandum # 13

Title: PDP-X Processor Description

Author(s): H. Burkhardt
L. Selighman

Index Keys: Architecture
Instruction Set
Processor

Distribution Keys: A, B, C

Obsolete: None

Revision: None

Date: July 24, 1967

Index

1.0 Introduction

1.1 Models

2.0 Processor Architecture

2.1 Instruction Format
2.2 Data Formats
2.2.1 Fixed Point Arithmetic Operations
2.2.2 Logical Operations
2.2.3 Floating-Point Operations
2.2.4 Character Operations
2.2.5 Byte Pointer
2.3 Addressing
2.4 General Registers
2.5 Program Status Word
2.5.1 Traps
2.5.2 Condition Codes
2.6 Instructions
2.6.1 Basic Instructions
2.6.2 Extended Operation Class
2.6.3 Extended Arithmetic Group
2.6.4 Character Group
2.6.5 Logical Compare and Modify Group
2.6.6 Push Down Group
2.6.7 IO Instructions
2.7 Priority (Interrupt) System
2.8 Protection Feature
2.8.1 Instruction Protection
2.8.2 Memory Protection
2.8.3 Monitor Calls
2.8.4 Instructions for Memory Protection System
2.8.5 Summary

3.0 IO System

3.1 Devices and Controllers
3.2 Modes of Data Transfer
3.3 Operation of the Multiplexor Channel and Interrupt
3.4 Basic Peripheral Structure
3.4.1 Status Register
3.4.2 Output Device - No Interrupt
3.4.3 Output Device - Interrupt Mode
3.4.4 Input Device - No Interrupt
3.4.5 Input Device - Interrupt Mode

3.5 Paper Tape Peripherals
3.5.1 Paper Tape Reader/Punch
3.5.2 Keyboard/Printer
3.5.3 Paper Tape Peripherals Status Bytes
3.6 Device Assignment Table
3.7 IO Bus
3.7.1 General Characteristics
3.7.2 Operation
3.7.3 Line Definitions
3.7.4 Basic Bus Timing/Flow Diagrams
3.7.5 Basic Control Organization

4.0 Appendices

4.1 Assembly Language Conventions
4.2 Instructions (Alphabetic)
4.3 Flow Chart Conventions

1.0 Introduction

PDP-X is a modern, very high performance, third generation, binary, two's complement computer family designed for the small computer market. Upward and downward (limited) program compatibility permits easy system growth and enhances application programming. Standard IO and Memory interfaces are used for all processor models and all peripheral devices. The architecture lends itself to fourth generation hardware implementation and the development of multiprocessor systems.

The system architecture of the PDP-X computer family is described below. Two particular members of the family are suggested; however, details of their implementations are beyond the scope of this document. PDP-X/I may be thought of as a PDP-8 class machine, PDP-X/II as a PDP-9 class machine.

1.1 Models

The smallest PDP-X model has two-priority levels, main program and interrupt level, both general-register sets reside in core memory. The Model I instruction set consists of only the basic instructions; all EOP class instructions trap. Memory is expandable from 4K to 32K. All peripheral equipment will run on Model I within the constraints of the single interrupt level.

The Model I processor may not be upgraded to a Model II processor. All of the other components of the system, however; may be used with a Model II processor.

The medium sized PDP-X model has two sets of hardware general registers for its two priority levels: main program and interrupt level. The Model II instruction set consists of both the basic instructions and extended instructions. The Basic configuration includes High-Speed Paper Tape, 33 KSR Teletype, and an 8K memory.

a. The Priority Interrupt System may be added. This option adds six sets of general registers, the priority interrupt structure, and special instructions to modify the state of the interrupt system.

b. The Protection Option may be added. This option consists of user mode/ executive mode, the memory paging system and certain special instructions to alter the state of the paging hardware. The Priority Interrupt system is required before this option may be added.

c. Power Fail, Memory Parity, Machine Check hardware may be added. Parity may be added without any alterations to the existing memory system.

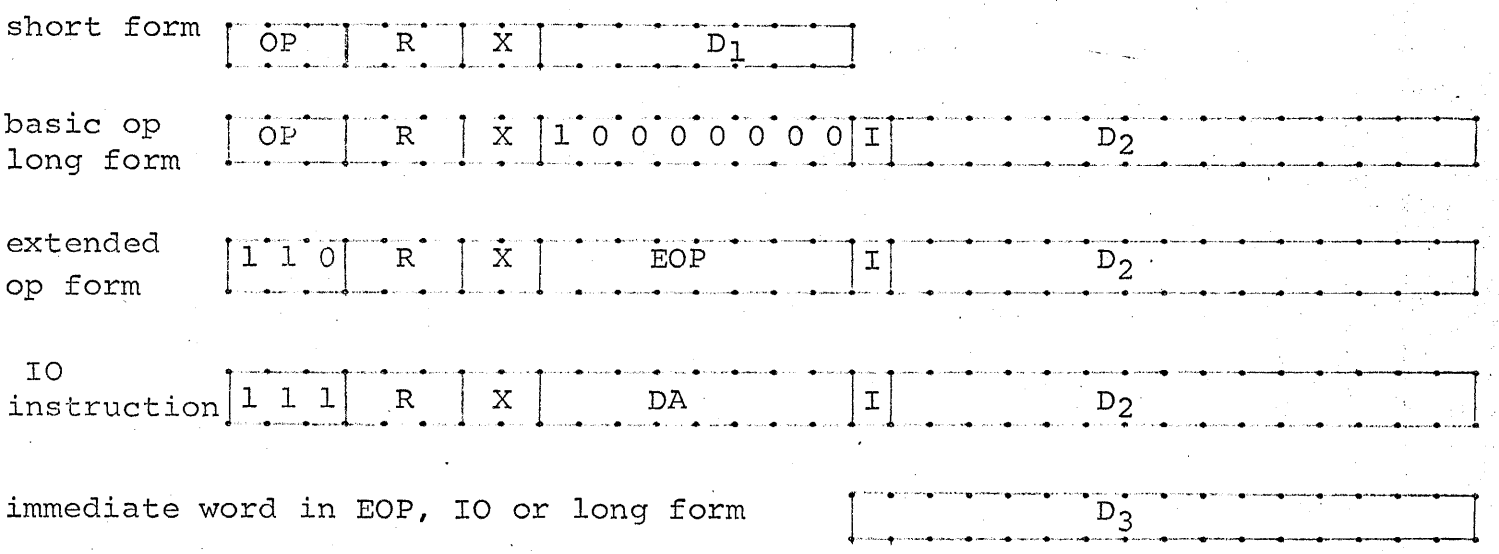
d. The memory system may be increased to 32K or to 128K, if the Protection Option is installed.

Models

Model	Register Sets	Instruction Set	Standard Memory	Standard IO	Interrupt Structure	Protection System	Add Time	Multiply Time (Signed)	Index Time
I_a	2, core	Basic	4k - 8μ	33 ASR	2 levels	None	32μ	subroutine	8μ
I_b	2, core	Basic	4k -.75μ	33 ASR	2 levels	None	3μ	subroutine	.75μ
II_a	2, hard-ware	Extended Set	8k -.75μ	33 ASR High-Speed Paper Tape	2 levels	None	1.5μ	< 14μ	0
II_b	8, hard-ware	Extended Set	8k -.75μ	33 ASR High-Speed Paper Tape	8 Fully Nested Levels	None	1.5μ	< 14μ	0
II_c	8, hard-ware	Extended Set	16k-.75μ	33 ASR High-Speed Paper Tape	8 Fully Nested Levels	User/Exec Modes Paging	1.5μ	< 14μ	0

2.0 Processor Architecture

2.1 Instruction Format



mnem	bits	definition
OP	3	basic operation code specifying major instruction class
R	3	general register specification or sub function selection for non-accumulator reference instructions
X	2	index register and address mode selector
D_1	8	short form address
D_2	15	long form address
D_3	16	immediate operand
I	1	indirect addressing specification
EOP	8	extended operation code specifying instruction
DA	8	IO device address and bus selection

2.2 Data Formats

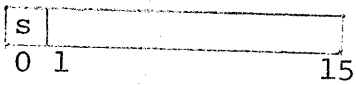
The hardware and software capabilities include operations on single and double precision fixed point data, short and long form floating point data, single precision logical data, and character bytes. On the larger processors, most of the operations will be performed by the hardware. On smaller processors, the operations may be performed by resident subroutines.

2.2.1 Fixed Point Arithmetic Operations

The basic arithmetic operand is the 16-bit fixed-point binary word. To preserve precision, all products and dividends are 32-bits long.

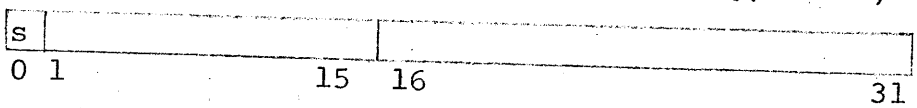
Since the 16-bit word size accommodates a 15-bit address, fixed point arithmetic can be used both for integer operand arithmetic and for address arithmetic. Since integer and addressing arithmetic often requires repeated references to operands or to intermediate results, the use of multiple registers is advantageous in arithmetic sequences and address calculations.

Additions, subtractions, multiplications, divisions and comparisons are performed upon one operand in a register and another operand either in a register or main storage. Two's complement notation is used to facilitate multi-precision arithmetic.



Signed (2's complement) single precision integer

$$-(2^{15}-1) \leqslant \text{word} \leqslant (2^{15}-1)$$



Signed (2's complement) double precision integer

$$-(2^{31}-1) \leqslant \text{word} \leqslant (2^{31}-1)$$

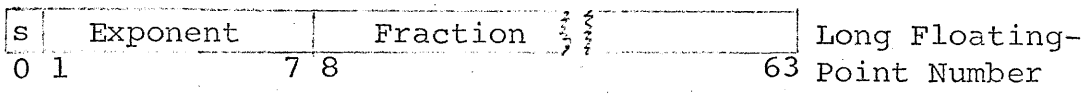
The address of a double precision quantity is the address of the high-order word. This address must be even (i.e. $EFA_{15} = 0$).

2.2.2 Logical Operations

Logical Operations are performed on 16-bit binary words with one operand in a register and another operand either in a register or in storage.

2.2.3 Floating-Point Operations

Floating-point numbers occur in either of two fixed length formats--short and long. These formats differ only in the lengths of the fractions



Operands are either 32 (two words) or 64 (four words) bits long. The short form, equivalent to seven decimal places of precision, permits a maximum number of operands to be placed in storage and gives the shortest execution times. The long form gives up to 17 decimal places of precision.

The fraction of floating-point number is expressed in hexadecimal (base 16) digits each consisting of four binary bits and having the values 0-15. In the short format, the fraction consists of six hexadecimal digits occupying bit positions 8-31. In the long format the fraction has 14-hexadecimal digits occupying bit positions 8-63.

The radix point of the fraction is assumed to be immediately to the left of the high-order fraction digit (between bits 7 and 8). To provide the proper magnitude for the floating-point number, the fraction is considered to be multiplied by a power of 16. The characteristic portion, bits 1-7 of both formats, is used to indicate the power. The characteristic is treated as an excess 64 number with a range -64 to +63 corresponding to 0-127. This permits

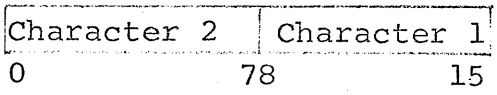
representation of decimal numbers with magnitudes in the range 10^{-78} to 10^{75}, Four 64-bit floating-point registers are provided (actual memory locations 32-47). Arithmetic operations are performed with one operand in a register and another either in a register or from stroage. The result is developed in a register. The availability of several floating-point registers eliminates much storing and loading of intermediate results.

The addresses of short-form floating-point numbers must be even ($EFA_{15} = 0$). The addresses of long form floating-point numbers must be evenly divisible by four ($EFA_{14-15} = 0$).

2.2.4 Character Operations

Two 8-bit bytes (characters) may be stored as a single computer word.

Packed Characters



Data transfers to and from external devices are done through an 8-bit channel. If, during an I/O read or write operation the device requests a two byte transfer, character 1 is received (transmitted) and then character 2. If the device does not request a two byte transfer, only character 1 of the effective word is effected.

2.2.5 Byte Pointer

The left 15-bits of the byte pointer select the word address of a byte pair. Bit 15 selects one of the 2 bytes:

0 = right byte

1 = left byte

2.3 Addressing

Addresses are generated by either long or short format instructions. In either case, the processor forms a 15-bit EFfective Address (EFA) which it sends to the memory system. The left byte (high order 7-bits) of the address is called the field; there are 128 fields of 256 words each.

The available addressing modes are:

a. direct (no indexing) to any word in memory

b. relative ($\pm 127_{10}$) to the instruction

c. immediate (the next location is the effective address)

d. indexed

e. linked (the subroutine linkage register is used to pick up arguments or to make returns).

In the short format, the displacement (D1) is treated as a signed two's complement number, unless the addressing mode is direct. In this case, D1 is a field 0 address. Long format instructions are specified whenever D1 = 128_{10} or whenever the instruction implicitly forces this format (all IO and extended op code instructions).

The address mode is, thus, a function of format (short or long) and of the X bits of the instruction:

Addressing Table (Effective Address Calculation)

X	Short	Long
0	D1 (Field 0)	D2 (Direct)
1	±D1+PC (Relative)	PC+1 (Immediate)
2	±D1+R2 (Linked) indexed by subroutine linkage register	D2+R2 (Linked) indexed by subroutine linkage register
3	±D1+R3 (Indexed)	D2+R3 (Indexed)

The basic addressable unit is the word (two bytes, 16-bits), although certain instructions do reference bytes or double-words. The contents of the effective address is called the EFfective Word (EFW). Words in storage are consecutively numbered starting with 0. The 15-bit address field accommodates a maximum of 32,768 words. When only a part of the maximum storage capacity is available in a given installation, the available storage is continguously addressable from 0. A nonexistant memory trap occurs when any operand is located beyond the installed capacity. The invalid address is recognized when the data is accessed and a program trap occurs with bit 3 of the Program Status Word set.

2.4 General Registers

Each level of priority contains a set of 16 general registers, a Register Group (RG) 8 of which may be used by the program as accumulators, index registers, etc. The Program Status Word (PSW) occupies registers 0 and 1. These registers occupy field 0 words 0 to 7 in the memory space as well as the R bits in the instruction; hence, register to register instructions are possible. The registers may be stored, loaded, added into, etc., depending on the operation code of the particular instruction used. The second group of 8 registers contain the trap locations for unimplemented EOP instructions, push down words, and hardware traps. These may be modified or read as memory words but are not explicitly referenced as accumulators. In Model II, the first 8 registers may be fast (flip-flop) registers instead of core memory as in Model I.

register use

0	R_0	status word, contains condition code, etc.
1	R_1	status word, contains program counter (PC)
2	R_2	accumulator, subroutine linkage register, or secondary index
3	R_3	accumulator or main index register
4	R_4	accumulator
5	R_5	accumulator
6	R_6	accumulator
7	R_7	accumulator

register	use
8	EOP, receives the updated program counter
9	EOP, receives instruction itself
10	EOP, receives effective address
11	EOP, contains the entry point into the EOP handler, loaded into PC
12	contains the push down pointer
13	contains the push down counter
14	TRAP, receives the updated program counter
15	TRAP, contains the entry point into the TRAP handler, loaded into PC

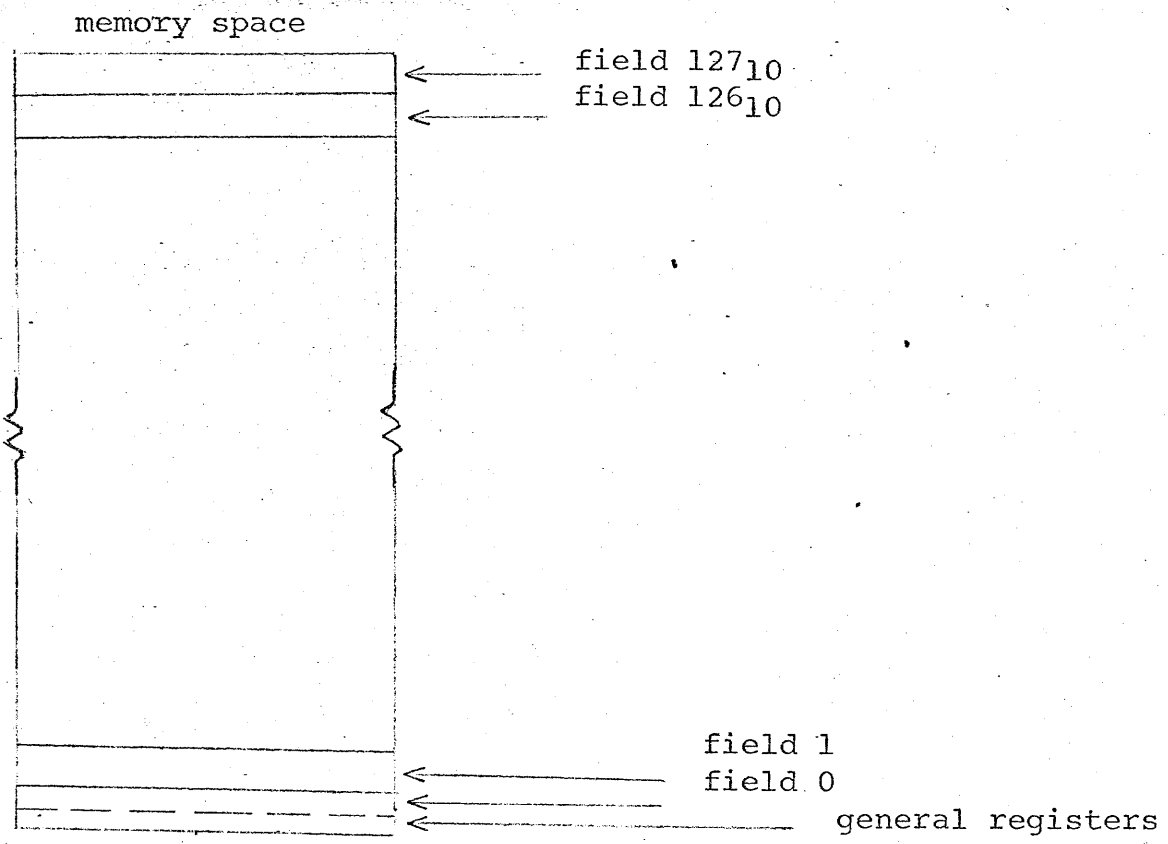
For each level of machine priority, both background and IO, there exists a set of general registers; in addition, the hardware insures that the applicable set is available at locations $0\text{-}15_{10}$ in (apparent) memory address space. Thus, the general registers need not be stored and restored during interrupts.

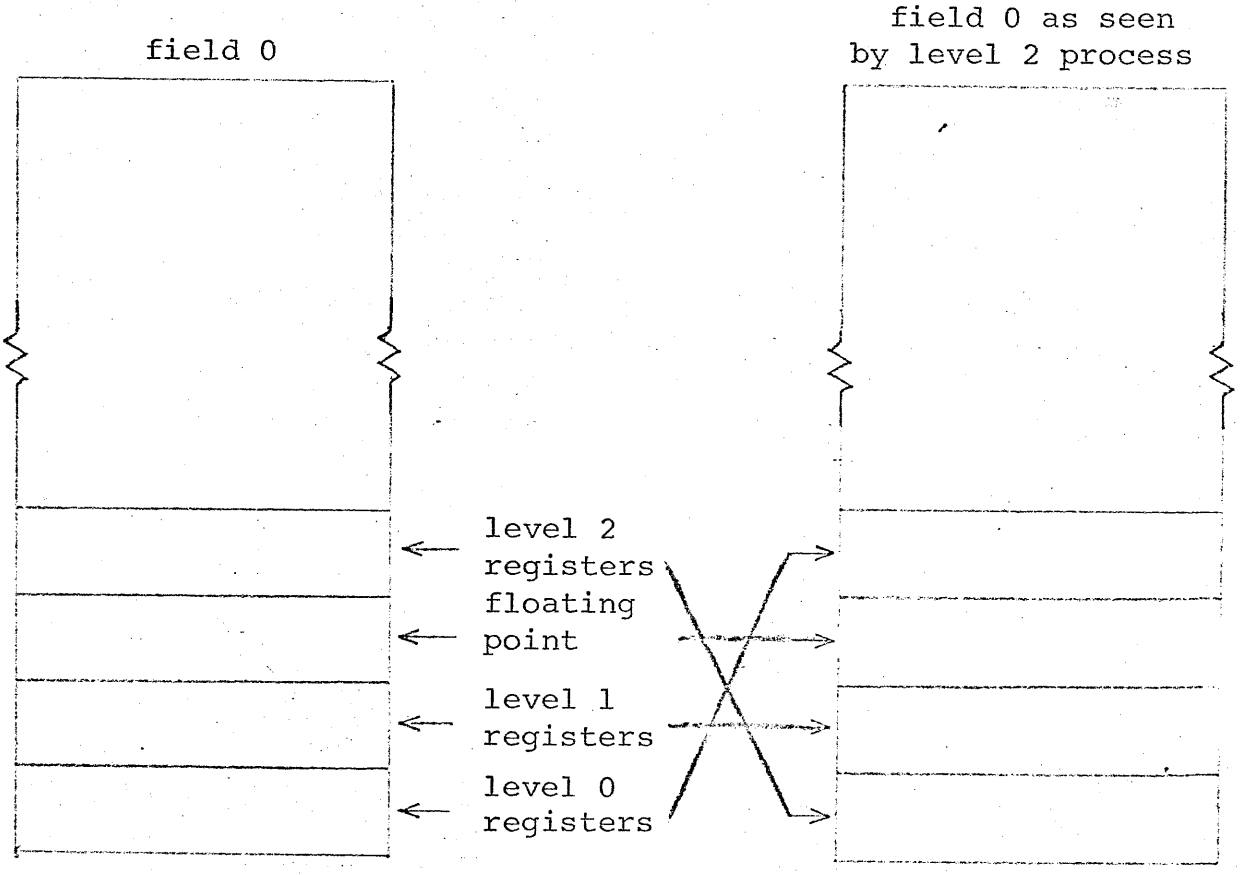
The lowest (background) priority level contains floating point registers. These registers are not available for use on the higher priority levels unless they are explicitly stored and restored under program control. Each of the 4 floating point registers is 64-bits (4 words) long, permitting multiple precision floating point instructions. In all floating operations the R bits of the instructions specify these registers. Only the low order 2-bits of R are used.

The set of general registers map onto the main memory space in field 0. The following figure shows the entire memory space; the figure on the right is an exploded view of physical memory. Apparent memory is the memory space as seen by the running process; this differs from physical memory in the location of its general registers as is shown for a priority level 2 process in the bottom figure. The hardware operates as follows:

a. whenever an address in the range $0\text{-}15_{10}$ is encountered, the RG bits are added to the address in bit positions $9\text{-}11_{10}$.

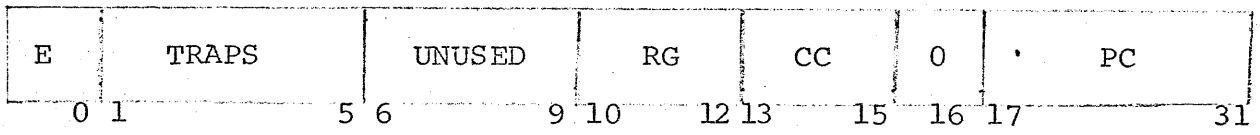
b. whenever bits 9-11 of the address are equal to the RG bits and address bits 1-8 are zeros, bits 9-11 of the address are cleared.


memory space
field 127_{10}
field 126_{10}
field 1
field 0
general registers


field 0
field 0 as seen
by level 2 process
level 2
registers
floating
point
level 1
registers
level 0
registers

2.5 Program Status Word

The collection of bits that constitute the state of the processor between instructions is called, collectively, the Program Status Word (PSW). This state word occupies the doubleword at memory locations 0 and 1 of the active process, corresponding to general registers R_0 and R_1.



Bit(s)	Definition
0	Arithmetic trap enable
1	Arithmetic (add, divide, shift, floating, etc.); enabled if bit 0 = 1
2	Push down list error
3	Nonexistent memory (reference to an address not in the memory system)
4	Privileged instruction (attempt to execute a system instruction while in user mode)
5	Protection violation (attempt to access a protected memory area)
6-9	Unused
10-12	Priority of active process (current Register Group)
13	Condition code bit 0
14	Condition code bit 1
15	Condition code bit 2
16	Unused, always 0
17-31	Program counter of active process

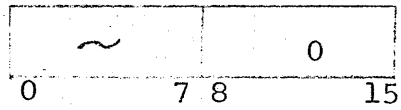
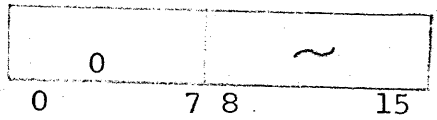
2.5.1 Traps

Program status word bits 0-5 constitute the trap indicators and the arithmetic trap enable bit; a trap occurs when an unusual condition is detected by the hardware. The trap source, bit, and new register group is given in the table below. Refer to the section on the protection feature for further explanation of the privileged instruction and protection violation traps. No arithmetic unusual condition may cause a trap unless the arithmetic trap enable bit is set.

During a trap, the updated program counter (usually points to the instruction following the error source) is stored in register 14_{10}, a new address, contained in register 15_{10}, is loaded into the program counter and the appropriate PSW trap bit is set.

In the case of a priority change, the change in general register groups occurs before the PC is stored and reloaded.

PSW bit	Source	New Priority and RG
1	Arithmetic Error	Same
	ADD: Magnitude of sum greater than register capacity	
	SUB: Magnitude of difference greater than register capacity	
	DIV, LDIV: Magnitude of quotient greater than register capacity	
	SHFT: (Arithmetic left shift only) sign bit (0) changed during shift	

PSW bit	Source	New Priority and RG
2	Push Down List Error Execution of PUSH type instruction with counter word of form  Execution of POP type instruction with counter word of form  	Same
3	Nonexistant Memory Attempt to address a word not available in the memory system, i.e., no memory responds to a processor request. In a time sharing environment, this error indicates an attempt to use a page whose assignment was never requested (see Section 2.8)	Same
4	Priviledge Instruction Attempt by user mode (see protection system Section 2.8) program to execute an IO class instruction	Monitor
5	Read Only Violation Attempt by the user mode program to write into a read-only page (see Section 2.8).	Monitor

2.5.2 Condition Codes

The condition codes (PSW bits 13, 14, 15) are used to determine conditional branches as the result of arithmetic and certain other operations. The codes are normally not changed during load, store, and branch instructions; the appendix contains an instruction list which indicates which instructions effect these PSW bits.

These bits, when changed, reflect the result of the operation just performed; previous condition code information is lost. Hence, the code bits must be tested before the next instruction which changes these bits is executed.

BIT	Normal Meaning
13	Carry during add, borrow during subtract, end bit for certain rotate operations
14	Negative result, unusual device response
15	Nonzero result

2.6 Instructions

Instructions may be divided into two groups, basic and extended. The basic instructions appear in both models and may be in either long or short format. Extended instructions are implemented in Model II, they trap when executed in Model I; extended instructions exist in long format only. The instruction class is determined by the three Op Code bits (0,1,2) of the instruction word. EOP (extended) instructions are characterized by a 110 pattern in the Op Code and the specific operation in the D_1 bits.

Instructions may also be classified by the type of the operand effected. These include:

Class:	Operand:
arithmetic	signed words
logical	unsigned words
floating	floating point double/ quadruple words
branch	address pointers
IO	IO system

Class	OP	mnem	Definition
load	0	LDA	LoaD specified Accumulator (R) with the effective word; the condition code bits remain unchanged.
store	1	STA	STore specified Accumulator (R) into memory at the effective address; the condition code bits remain unchanged.
and	2	AND	AND specified accumulator (R) with the effective word, place result in specified accumulator; condition code 0 remains unchanged 1 set if negative result, cleared otherwise 2 set if nonzero result, cleared otherwise
add	3	ADD	ADD contents of specified accumulator (R) with the effective word following the rules of two's complement arithmetic, place result in specified accumulator; condition code 0 set if carry out of bit 0, cleared otherwise 1 set if negative result, cleared otherwise 2 set if nonzero result, cleared otherwise
branch	4		general conditional branch and subroutine linkage instruction; R bits specify particular operation. If the branch is taken, the effective address is loaded into the program counter. When R = 7 the program counter, updated to point to the instruction following the branch, is saved in accumulator/index register 2; the previous contents of 2 are lost. The condition code bits remain unchanged for all branch instructions.
			R Condition
		BCN	0 branch if condition code bit 0 set (Branch if Carry Nonzero)

Class	OP	mnem	Definition
			R Condition
		BM	1 branch if condition code bit 1 set (Branch if Minus result)
		BN	2 branch if condition code bit 2 set (Branch if Nonzero result)
		B	3 unconditional Branch
		BCZ	4 branch if condition code bit 0 NOT se (Branch if Carry Zero)
		BP	5 branch if condition code bit 1 NOT se (Branch if Positive result)
		BZ	6 branch if condition code bit 2 NOT se (Branch if Zero result)
		BAL	7 Branch And Link

Note: BAL is the basic subroutine call instruction and it is used in the following manner:

```
        BAL     SUBR    ; CALL
        ARG 1           ; FIRST ARGUMENT
        .
        .
        .
        ARG N           ; Nth ARGUMENT

        -               ; RETURN, INSTRUCTION EXECUTED
                        ; WHEN SUBROUTINE DONE, MORE
                        ; THAN 1 RETURN IS POSSIBLE JUST
                        ; AS THERE MAY BE MORE THAN
                        ; 1 ARGUMENT

SUBR:   -               ; FIRST INSTRUCTION OF SUBROUTINE

        LDA 4, I(2)     ; PICK UP I + 1th (I ≤ N)

        -               ; ARGUMENT

        B   N(2)        ; RETURN TO CALLING ROUTINE
```

In order to nest subroutines, the subroutine linkage register must be stored in a temporary storage memory

word, usually within the subroutine.

If the arguments were addresses, rather than data words, the instruction to pick up data would need to specify indirect addressing, i.e.:

```
LDA 4,   @ I (2)
```

Class	OP	mnem	Definition
modify	5		general memory modification instruction; the R bits specify a particular operation. Condition code bit 0 is changed only by the two rotate instructions (R = 4,5). Condition code bits 1 and 2 are set as follows for all modify instructions: 1 set if negative result, cleared otherwise 2 set if nonzero result, cleared otherwise Note: in short format these instructions may modify the accumulators or other general registers.
			R Operation
		TST	0 TeST, no operation but condition code bits 1 and 2 are set to reflect the state of the effective word.
		COM	1 logical COMplement, the effective word is complemented on a bit-by-bit basis.
		INC	2 INCrement, one is added to the effective word.
		NEG	3 NEGate, the effective word is negated (complemented then incremented).
		RR	4 Rotate Right, the effective word and condition code bit 0 are rotated together as a 17 bit register one place to the right, loading condition code bit 0 from bit 15.
		RL	5 Rotate Left, the effected word and condition code bit 0 are rotated left together as a 17 bit register, loading condition code bit 0 from bit 0 of the memory word.

Class	OP	mnem	Definition
			R Operation
		SWP	6 SWaP bytes, the left and right bytes of the effective word are interchanged.
		CLR	7 CLeaR, the effective word is set to a zero.

2.6.2 Extended Operation Class

Class	OP	mnem	Definition
EOP	6		Extended OPeration code class; forced long format; D_1 bits specify particular operation to be performed. The effect on the condition code bits depends upon the particular operation performed.

D_1 codes 0 through 63_{10} are reserved for 64_{10} programmed operators. These codes specify UUO's (UnUsed Operation codes). Codes 0 through 31_{10} are reserved for user program/monitor communication since they can function as protected entry points. See section 2.8 on protection features.

If the operation specified has not been implemented in the machine or if it is UUO, a trap occurs as follows:

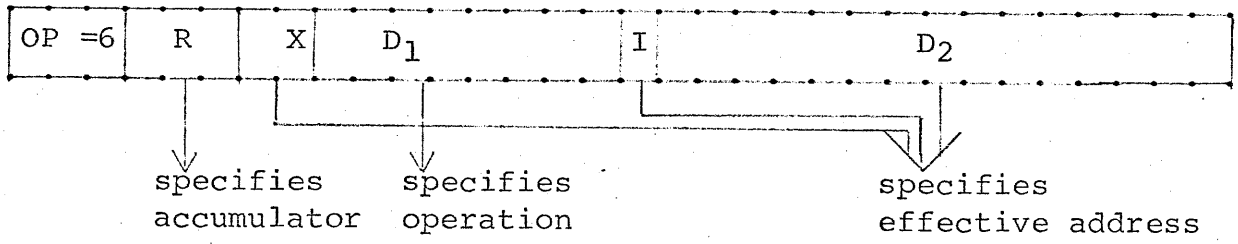
location 8_{10} receives the updated program counter

9 EOP instruction

10 effective address

11 contains the entry point into the EOP handler. This word is loaded into the program counter.

EOP class instruction double word:



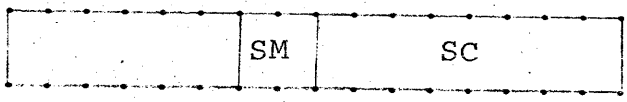
2.6.3 Extended Arithmetic Group

D_1	mnem	Definition
112	SUB	SUBtract the effective word from the contents of the specified accumulator (R) following the rules of two's complement arithmetic; place result in the specified accumulator. condition code bit 0 set if carry out of bit 0, cleared otherwise 1 set if negative result, cleared otherwise 2 set if nonzero result, cleared otherwise
101	MUL	MULtiply. The effective word is algebraically multiplied by the low order word of the doubleword specified by R. If R is even, the doubleword product replaces the doubleword at R and is properly signed. If the specified accumulator (R) is odd, the high order part of the doubleword product is discarded. The multiplier and multiplicand are taken to be signed. No overflow is possible. condition code bit 0 remains unchanged 1 is set if negative product, cleared otherwise 2 is set if either half of the doubleword product is nonzero, cleared otherwise
100	LMUL	Logical MULtiply. The effective word is logically multiplied by the multiplier as described for MUL, above, except that the operands are taken to be positive 16 bit logical quantities. condition code bit 0 remains unchanged 1 is set if the product is $\geq 2^{31}$ 2 is set if either half of the doubleword product is nonzero, cleared otherwise
103	DIV	DIVide. The signed arithmetic doubleword beginning at the specified accumulator (R) (if odd divide overflow is forced) is algebraically

D_1	mnem	Definition
		divided by the effective word. The signed quotient developed replaces the low order word of the dividend, the remainder, signed the same as the dividend replaces its high order part. When the relative magnitude of dividend and divisor is such that the quotient cannot be expressed by a 16 bit signed integer, a divide overflow trap occurs, no division takes place, and the dividend may be lost. condition code bit 0 set for divide overflow, cleared otherwise 1 set for negative quotient, cleared otherwise 2 set for nonzero quotient, cleared otherwise
102	LDIV	Logical DIVide. The logical doubleword beginning at the selected register (R) (if odd divide overflow is forced) is divided by the effective word. The operation performed is the same as DIV except that the operands are treated as 16 and 32 bit positive integers. The results are also 16 bit positive integers. condition code bit 0 set for divide overflow, cleared otherwise 1 set for quotient 7, 2^{15} 2 set for nonzero quotient
111	CMP	algebraic CoMPare. The specified accumulator (R) and the effective word are algebraically compared as signed integers. Neither accumulator nor effective word are changed, but the condition code is set according to the result. condition code bit 0 remains unchanged 1 set if register < memory word, cleared if ≥ memory word 2 set if register ≠ memory word, cleared otherwise
110	LCMP	Logical CoMPare. The specified accumulator (R) and the effective word are logically compared as 16 bit positive integers. Operation proceeds and condition code is set as in CMP.

D_1	mnem	Definition
113	SHFT	SHIFT. The contents of the specified accumulator (R) is shifted as indicated by the effective word. The right half (byte) of the effective word is used as a signed shift count. A positive value indicates a left shift. Bits 6 and 7 of the effective word indicate the type of shift to be performed. (Condition code bit 0 is changed only by the Rotate with CC0 mode.) Condition code bits 1 and 2 are set as follows for all shift instructions: 1 if negative result, cleared otherwise 2 set if nonzero result, cleared otherwise) 00 - arithmetic shift - performs two's complement multiplication by powers of two. The sign is unchanged. When going to the right, the sign is shifted into bit 1. Ones or zeros leaving bit 15 are lost. When going to the left, zeros enter bit 15. The arithmetic error bit (bit 1 of the PSW) is set if, during shifting, the sign bit and bit 1 become unequal. 01 - rotate with CC0 - bits leaving one end enter condition code bit 0. CC0 enters at the other end. 10 - rotate - bits leaving one end enter at the other end. 11 - logical shift - bits leaving one end are lost and zeros enter the other end.

Shift Control Word



SM = Shift Mode

00 - Arithmetic Shift

01 - Rotate with CCO

10 - Rotate

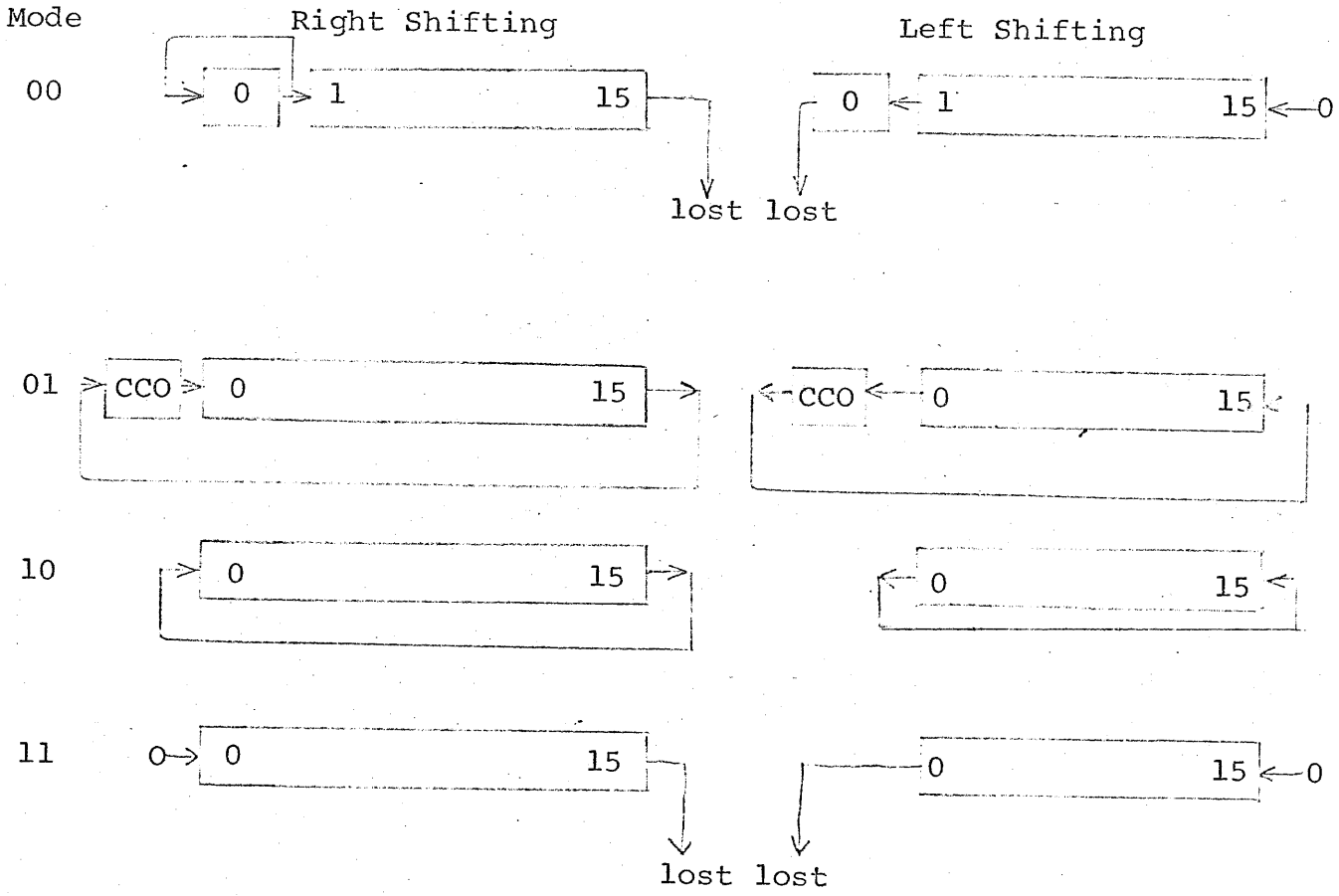
11 - Logical Shift

SC = Signed Shift Count

>0 : Left

<0 : Right

=0 : No Shift



2.6.4 Character Group

D_1	mnem	Definition
114	LDC	LoaD Character. The effective word is used as a character pointer to locate an 8-bit byte. This byte is loaded into the right half of the specified accumulator (R). The left half is cleared. The addressed memory word is left unchanged. The condition code remains unchanged.
115	STC	STore Character. The effective word is used as a character pointer to locate an 8-bit byte. The right half of the specified accumulator (R) is stored at the indicated character position. The other half of the addressed word is unaltered. The content of R remains unchanged. The condition code remains unchanged.

CHARACTER (BYTE) POINTER WORD

ADDR = address C = character selection
0-right byte
1-left byte

Note: This is identical to the BA word in the multiplexor channel.

2.6.5 Logical Compare and Modify Group

Bits of the specified accumulator (R) that are masked by bits of a memory word may be tested and/or modified to determine a conditional branch. The bits to be tested and/or modified are selected by ones in the effective word. Condition code bit 2 is cleared if all of the selected bits of the specified accumulator (R) are zero; otherwise, it is set to a one. Condition code bit 1 is set to a one if bit 0 is selected and bit 0 of the specified accumulator (R) is a one; otherwise, it is cleared. The selected bits of the specified accumulator (R) are then modified or not depending upon the operation being performed. Condition code bit 0 is undisturbed.

D_1	mnem	Definition
104	TSTN	TeST but change Nothing. Test the content of the specified accumulator (R) against the effective word. Set condition code bits 1 and 2 according to the result.
105	TSTZ	TeST and Zero selected bits. Test the content of the specified accumulator against the effective word. Set condition code bits 1 and 2 according to the results. Clear selected bits in the specified accumulator (R) (i.e., for every one in the effective word, clear the corresponding bit in the specified accumulator). This performs the logical extract operation.
106	TSTO	TeST and set selected bits to Ones. Test the content of the specified accumulator (R) with the effective word. Set condition code bits 1 and 2 according to the result. Set selected bits in the specified accumulator (R) (i.e., for every one in the effective word, set the corresponding bit in the specified accumulator). This performs the logical function inclusive or.

D_1	mnem	Definition
107	TSTC	TeST and Complement selected bits. Test the content of the specified accumulator with the effective word. Set condition code bits 1 and 2 according to the result. Complement selected bits in the specified accumulator (R) (i.e., for every one in the effective word, complement the corresponding bit in the specified accumulator). This performs the logical function exclusive or.

2.6.6 Push Down Group

Words are pushed into (popped from) memory under control of pointer and counter words. General register 12_{10} is the push down pointer. Its contents indicate the first free location on the push down list. General register 13_{10} is the push down counter which insures that the space allotted to the list is not exceeded. Maximum push down list length is 256 words. Exceeding the list capacity while either storing or retrieving causes a push down error trap with bit 2 of the PSW set. The registers are incremented/ decremented on each instruction.

The list control registers are initialized by placing the starting (lowest) address of the list in the pointer word and the length of the list (positive integer $\leq 255_{10}$) in the counter word. During a push type instruction, the pointer is used then incremented; the counter left byte is incremented; the right byte decremented. During a POP type instruction, the pointer is decremented then used; the counter left byte is decremented, the right byte is incremented. A trap occurs whenever a byte whose value is 0 is decremented.

All 8 Push/Pop class instructions manipulate the pointer and counter words. PUSHBL and POPBL move 2 words onto or off of the push down list.

D_1	mnem	R	Definition
116	PUC	0	PUsh Count. The pointer and counter are modified as for a push type instruction but no data is put onto the list.
	PUSH	1	PUSH. The effective word is placed in the next location on the push down list.
	PUB	2	PUsh and Branch. The program counter is placed in the next location on the push down list. The effective address replaces the program counter.
	PUL	3	PUsh, Branch and Link. The subroutine linkage register is placed in the next location on the push down list.

D I	mnem	R	Definition
			The program counter is placed in the subroutine linkage register. The program counter is placed in the next location on the push down list. The effective address replaces the program counter.
	POC	4	POp Count. The pointer and counter are modified as for a POP type instruction but no data is removed from the list.
	POP	5	POP. The last word placed on the push down list is moved to the content of the effective address.
	POB	6	POp and Branch. The sum of the effective word and the last word placed on the push down list replaces the program counter. This is the return instruction for PUB.
	POL	7	POp, branch and Link. The sum of the effective word and the last word placed on the push down list replaces the program counter. Then, the new last word on the push down list is then placed in the subroutine linkage routine. This is the return instruction for PUL.

2.6.7 IO Instructions

Class	OP	mnem	Definition
IO	7		Input/Output instruction class; R bits specify particular operation; forced long format. D_1 is taken to specify the device, lower order 6 bits specify a Device Address (DA), the higher order 2 bits specify one of four possible busses.

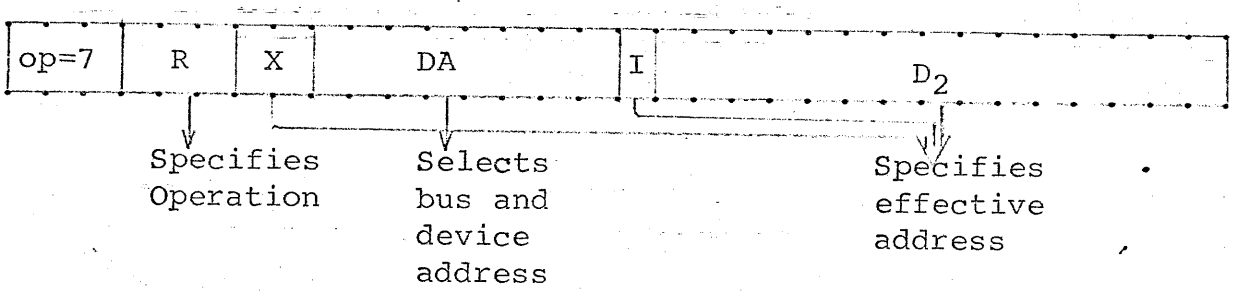
condition code bit 0 remains unchanged
1 set if no device responds, cleared otherwise
2 set if the byte or word resulting from the operation is now zero, cleared otherwise

Bit 2 has particular meaning following the IO test status (IOT) instruction. A byte is normally transmitted to (received from) the device from the right half of the effective address, some devices will automatically take (send) the second byte also. Every IO instruction calculates an effective address even if not used.

R	Operation	
IOR	0	IO Read. The data buffer of the addressed input device replaces the effective word.
IOS	1	IO read Status. The status register of the addressed device replaces the effective word.
IOT	2	IO Test status. The device status and the effective word are logically ANDed as in the TSTN instruction. Note that the resulting condition code bits may be tested by a subsequent conditional branch instruction.

Class	OP	mnem	Definition
			R Operation
		IOW	4 IO Write. The effective word replaces the data buffer of the addressed output device. Certain output devices are started.
		IOC	5 IO Command. The effective word replaces the status register of the addressed device.
		IOD	6 Internal IO Device control. See below.
		IOX	7 IO Xor status. Similar to IOC, the effective word and the device status register are XORed; the result replaces the device status register. Note that this instruction permits a portion of the status register to be changed.

IO Class Instruction Doubleword



The IOD instruction is used to change the state of the IO system. The DA field is used to decode specific functions, unused DA codes lead to no operation.

DA	mnem	Definition
0	RIO	Reset IO system; all devices are cleared; no interrupt may occur from any device unless that device is specifically reinitialized. This instruction is accomplished by transmitting the reset all code on the IO bus. All devices, both DEC and customer-designed must use this code to reset to a known, non-operating state, i.e.: clear ALL flip-flops in the devices and in the controllers. Note: Refer to section on priority structure for further explanation of the priority system.
4	PSI	Priority System Inhibit - the priority of the currently running process is raised by setting an inhibit indicator to the value specified by the effective word (i.e., the content of the effective address). The low order three bits of the effective word is compared with the highest priority level active indicator. If its value is higher than the indicator, then the priority level inhibit indicator corresponding to this value is turned on. If it is less than or equal to the active indicator, no operation results. In either case, the RG bits are not changed. This instruction is normally used together with its return, PSC, in calling a subroutine which is not re-entrant. Such cases occur in changing queue parameters of certain interrupt service routines. A typical call is of the form: PSI N ; raise priority to N BAL SUBR ; call subroutine ARG ; argument(s) PSC ; restore priority

DA	mnem	Definition
		N is chosen to be sufficiently such that no routine that runs at a priority level greater than N calls SUBR.
5	PSR	Priority System Request - The priority level request indicator corresponding to the value of the low order three bits of the effective word is set. An interrupt at that level is requested; this request is handled as if it were an external IO device. When the interrupt is granted, the interrupt address is determined according to the following table:

Note: Refer to section

Priority Level	Interrupt Address
0	No interrupt requested
1	422
2	424
:	:
7	436

6	PSC	Priority System Clear - The priority level of the currently running process is returned, by clearing an inhibit indicator, to the priority preceding the last PSS instruction issued. The priority level inhibit indicators are compared with the priority level active indicators. If the highest inhibit level is greater than the highest active level, then that inhibit indicator is turned off. Note that RG is not changed.
7	PSD	Priority System Dismiss - The currently active process is terminated and control is returned to the interrupted process. The highest active level indicator is

DA	mnem	Definition

cleared and the RG bits are set to the value of the new highest active level indicator.

This instruction is normally used at the completion of an interrupt service routine to return control to the interrupted process.

2.7 Priority (Interrupt) System

The Priority System (PS) controls the eight (maximum) levels of priority possible in the processor including the main program at the lowest level. The interrupt due to an internal source (PSI instruction) or an external source (IO device) causes the new, appropriate set of general registers to be substituted for the previously operating set. Since the state of the processor is stored and reloaded almost instantaneously, the interrupt service routine can begin immediately. The priority levels are fully nested so that, for example, an interrupt at level 6 would occur into a process running at level 4, but an interrupt at level 2 could never occur into such a process.

Four elements of the hardware are important to the programmer; these include: the Register Group (RG) bits of the PSW, the priority level inhibit indicators, the priority level active indicators, and the priority level request indicators. The register set currently in use is specified by the RG bits. These priority levels are normally used as shown in the following table:

Level (RG)	Use	Traps
0	Main Program	Arithmetic
1	Monitor, Lowest Hardware Device	Monitor
2-6	Device	
7	Highest Hardware Device	Machine Check

The priority level active indicators are a set of seven flip-flops, one for each of levels 1-7, that determine which levels have active processes associated with them. These indicators are set only by external devices and cleared only by the PSD instruction. If the main program were interrupted first by a level 2 device, and then by a level 6 device before the former device was completely serviced, then the indicators for levels 2 and 6

would be on. No interrupt may occur at a level equal to or less than the highest active level as stored in the indicators.

The priority level inhibit indicators are a set of seven flip-flops, one for each of levels 1-7, that inhibit interrupts at all levels equal to or lower than the highest inhibited level. These indicators are set by the PSS instruction and cleared by the PSR instruction. If either an inhibit indicator or an active indicator is set, only interrupts with a priority above the highest indicator may occur.

The priority level request indicators are associated only with the PSI instruction. Execution of this instruction causes the specified (1-7) bit of the request indicators to be set. These indicators are treated as devices requesting interrupts at each of the seven external priority levels. Associated with each level is an interrupt address, the branch instruction stored therein is executed when the priority of the priority level request indicator is higher than any active process and is not inhibited. When the interrupt occurs, the request indicator is cleared.

Level (RG)	Interrupt Address
1	422
2	424
3	426
4	430
5	432
6	434
7	436

When an interrupt occurs, the RG bits of the new PSW are set to the new priority level and the appropriate active indicator is set. Subsequent instructions will use the interrupt process PC and register set. The interrupt is cleared with a PSD instruction which clears the highest active level indicator and loads the RG register with the value of the new highest-active level indicator.

2.8 Protection Feature

The protection feature consists of three items to allow multi-user operation of PDP-X: (a) instruction protection, (b) memory protection, (c) monitor calls. The protection feature is only available on Model II processors with the Priority System option.

2.8.1 Instruction Protection

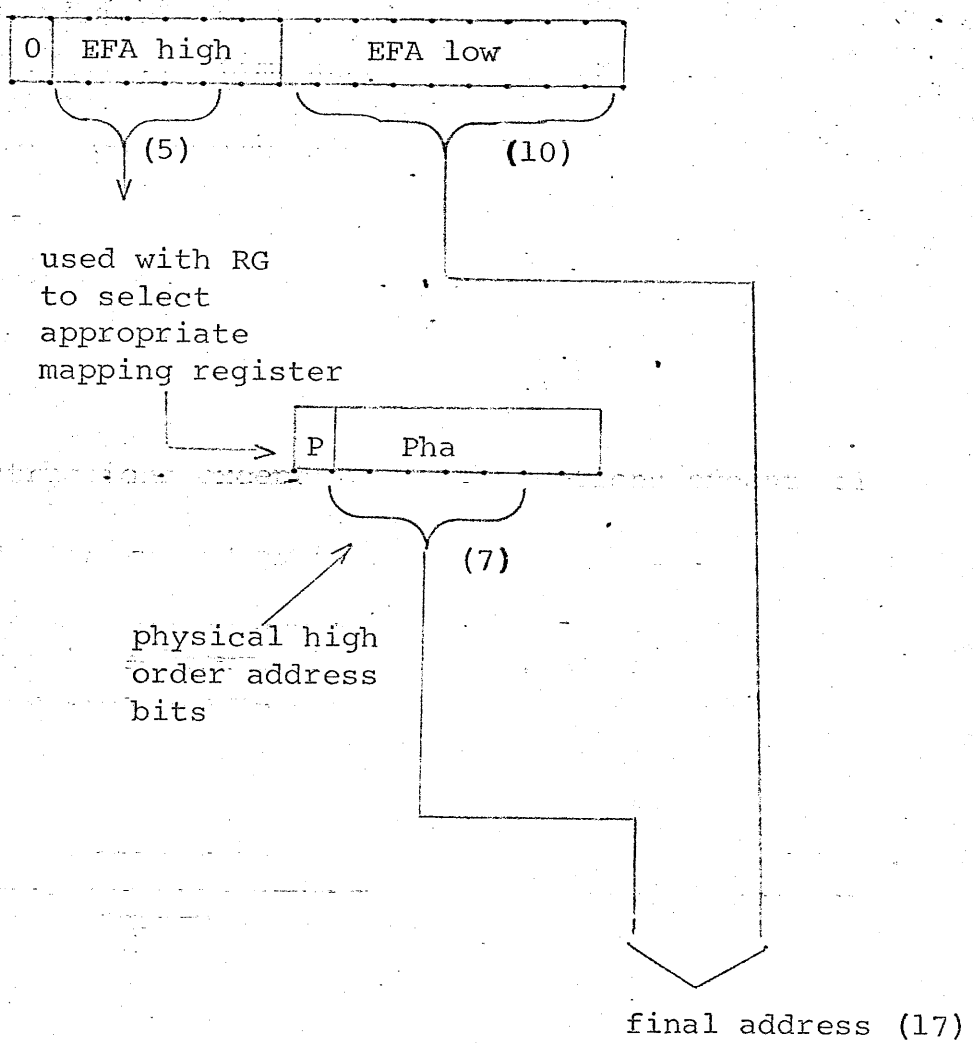
With the protection feature, PDP-X runs in two distinct modes: user mode and monitor mode. The PDP-X is in user mode when RG = 0. In user mode, the program may execute all instructions except the IO class instructions, thus the user program may in no way alter the state of (a) the IO system, (b) the Priority System, (c) the protection system. Monitor mode is entered whenever RG becomes nonzero, in other words, whenever priority is raised to levels 1-7. In monitor mode, all instruction classes may be executed.

If a program in user mode executes an IO instruction, the priority is raised to level 1 (RG set to 1), bit 4 of the PSW is set and a trap occurs. Similarly, if the user mode program violates memory protection (see 2.8.2), a similar sequence will occur setting either bit 3 or bit 5 of the PSW.

2.8.2 Memory Protection

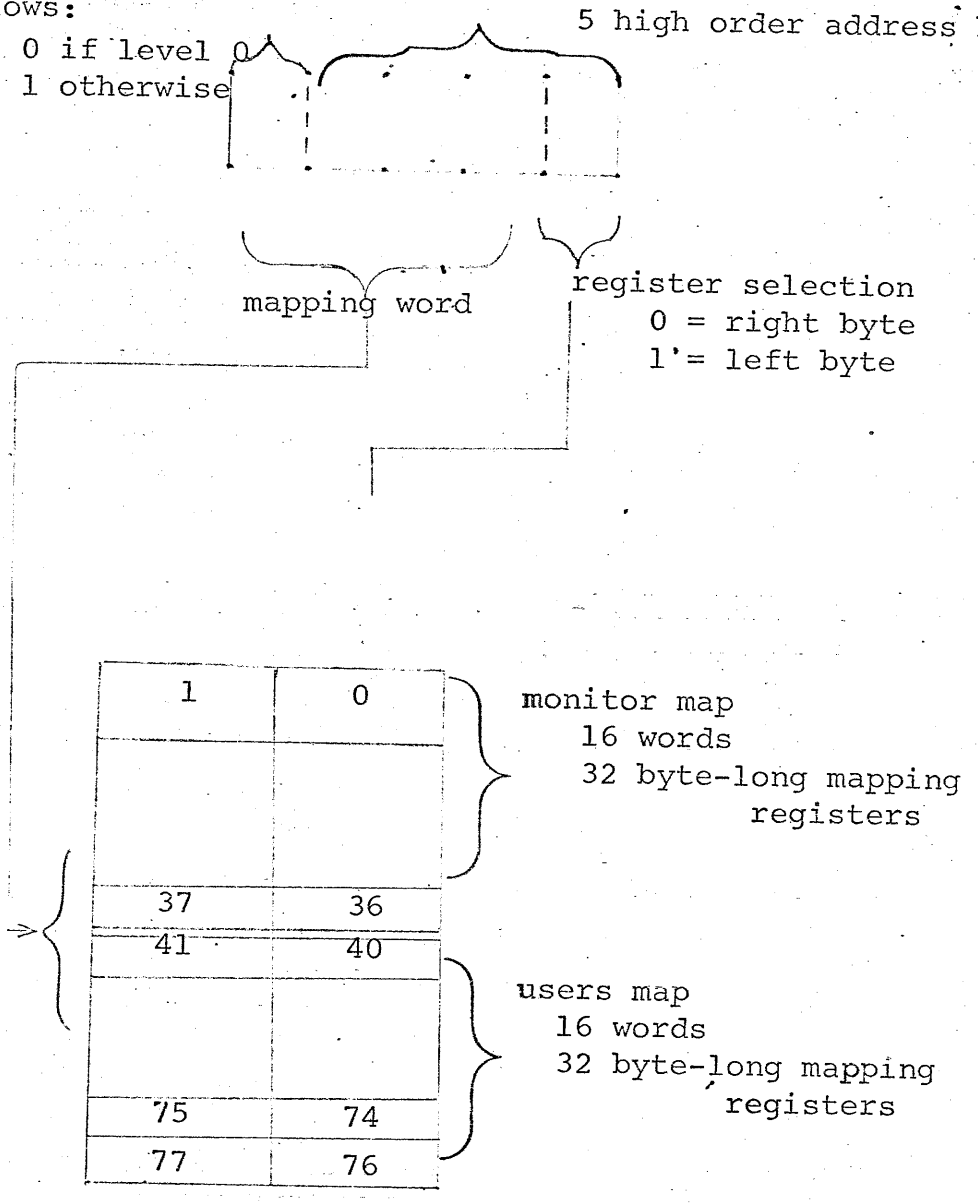
Memory protection is accomplished through memory paging. The high order address bits formed by the processor are not sent directly to the memory system; instead, together with the RG bits of the PSW, they are used to select one of several mapping registers. The contents of the selected mapping register is sent to the memory system along with the unaltered low order bits.

Each mapping register contains one (read only) control bit and seven address bits which substitute for the processor-generated five high order bits. Page size is, therefore, 1K. The diagram on the following page outlines this mapping.



A maximum of 64_{10} protection/mapping registers, each one byte long, may be accommodated. One map (32 mapping registers) is provided for priority level 0 (user mode) which will be referred to as the user map. Another map is provided for the other seven levels (monitor mode) which will be referred to as the monitor map since the monitor program normally runs at priority level 1 and the monitor controlled IO routines run at levels 2-7.

The address of the mapping register is formed as follows:



If the P bit is set and PHA is nonzero, then the specified memory page is taken to be read only and no user mode program may write into that page. An instruction executed in user mode that attempts to write into a read only page causes a read only violation which: (a) raises priority to level 1, (b) sets bit 5 of the PSW, and (c) initiates a trap sequence (see section 2.5.1).

If the P bit is set and PHA is zero, then no memory page has been assigned to the program. A user mode program that attempts to reference a word in such a page causes a non-existent memory violation which: (a) raises priority to levels 1, (b) sets bit 3 of the PSW, and (c) initiates a trap sequence (see section 2.5.1).

2.8.3 Monitor Calls

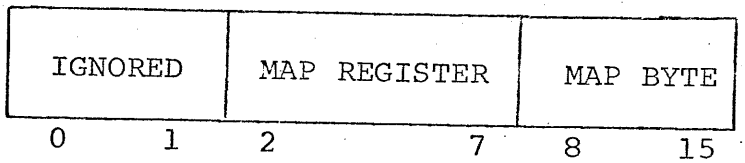
If a program running in user mode executes an EOP class instruction with D_1 in the range $0 \leq D1 \leq 37_8$, the priority is raised to level 1 and the EOP is executed using the monitor's register group. If a user mode program executes an EOP class instruction with D_1 in the range $40_8 \leq D_1 \leq 77_8$, the priority is unaltered and the EOP is executed using the user program's register group (see section 2.6).

This mechanism allows a convenient means for the user program to communicate with the monitor.

2.8.4 Instructions for Memory Protection System

Four IOD class instructions are added to PDP-X with the addition of the protection option. These instructions allow the monitor mode to load and modify the user and monitor maps.

DA	mnem	Definition
10	LMM	Load Monitor Map. The 16 words (32 bytes) starting at the effective address are loaded into the monitor map registers. Monitor map register 0 is always 0 and is not changed by this instruction (although the byte is read from memory). The user map is left unchanged.
11	LUM	Load User Map. The 16 words (32 bytes) starting at the effective address are loaded into the user map registers. The monitor map is left unchanged.
12	LBM	Load Both Maps. The 32 words (64 bytes) starting at the effective address are loaded successively into the monitor map (32 bytes) and the user map (32 bytes). Monitor map register 0 is always 0 and is not changed by this instruction.
13	LSM	Load Selected Map. The effective word is interpreted as:



The map register field is used to select one of the 64 map registers. The map byte is then loaded into the selected map register. Monitor map 0 is always 0 and may not be changed by this instruction.

2.8.5 Summary

The PDP-X memory protection system is designed to provide the following capability:

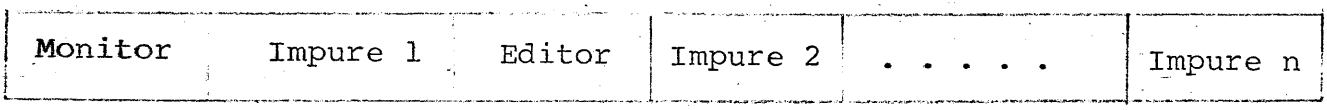
a) efficient memory management and protection in a multi-user environment

b) re-entrance for systems software packages such as the editor, assembler and compiler

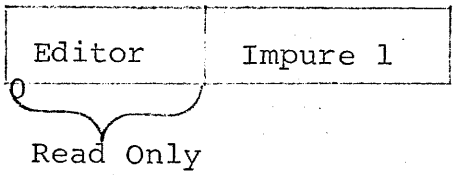
A paging system is employed for memory management and protection. Each user may have a virtual memory space of up to 32K words while the physical memory may contain 128K words. The individual user pages need not be contiguous in the physical memory allowing for the separate maintenance of IO buffers, temporary storage and data. Jobs may be swapped out and a new job swapped in without "shuffling" as is done in PDP-10. IO buffer areas need not be moved in physical space and, hence, they need not be swapped or shuffled.

The major systems programs may be run in a re-entrant environment if they do not modify themselves during the course of execution. Each program will have two parts termed the pure portion (that portion not modified, i.e., instructions and constants) and the impure portion (that portion that is modified, i.e., buffers, data and temporary storage). For several jobs to use the re-entrant program, it is only necessary to have one copy of the pure portion in core. Each job would have its own impure portion.

In this case, the physical space might appear as:



When job 1 is to be run, the user map is arranged so that user virtual memory appears as:



Thus, only one copy of the editor need be resident in core for many users. The impure portions may be swapped in and out.

Paging is not sufficient, however, to accommodate the general case of arbitrarily-shared procedures. This case must be accommodated by well-defined conventions.

3.0 IO System

The PDP-X IO system is both byte and full-word oriented. For devices whose media is no more than a byte wide, the multiplexor channel packs (unpacks) bytes directly into (from) memory, eliminating the need for assembly (disassembly) registers in the devices.

For devices whose natural word is larger than a byte, both the IO instructions and the channels transfer a whole word.

Interrupt sources are automatically identified by the basic hardware and the priority levels at which the devices interrupt are fully nested. In addition, no special device hardware is required in order to operate a device on the multiplexor channel; the processor channel hardware generates the same control sequences as do the IOR and IOW instructions.

3.1 Devices and Controllers

The hardware involved in IO operation is logically divided into four parts: IO section, IO bus, controller, and device. The IO bus is described in detail below. Controllers and devices are generally different for each type of IO media; from the programming point of view, most controller functions merge with IO device functions.

In all cases, the controller function is to provide the logic and buffering capabilities necessary to operate the associated IO device. Each controller functions only with the IO device for which it is designed, but each controller has standard signal connections with regard to the IO bus. The teletype device (keyboard, printer), for example, connects to the IO bus through teletype controller logic (single character data buffering and interrupt logic). The detailed meaning of the command/status bits read under program control through the IO section from controller type to type, but the general format remains unchanged.

3.2 Modes of Data Transfer

There are three basically different modes of data transfer available in the IO system: program-controlled, multiplexor channel, and selector channel. All three use the standard IO bus interface; the third provides an additional physical bus interface and additional control logic at the processor end. Maximum data transfer rate for each mode varies with processor model, but the program-controlled rate is always lowest and the selector channel rate highest. In all cases, transfer sequences are initiated by IO instructions issued to the appropriate controller, rather than to the channel.

Program-controlled transfer, while slowest, provides the greatest flexibility. Data may be modified, limit checked, or otherwise monitored as it is transmitted; control sequences required by special purpose or custom-designed IO equipment may be generated. For the slower devices, expecially paper tape or teletype, direct program control of IO leads to simpler programming.

Multiplexor channels are provided in both the basic Model I and II processors. When a device requires channel servicing, the state of the processor is dumped, the device serviced, and the state of the processor restored; the program is simply subjected to a short delay. The multiplexor channel is capable of sustaining concurrent IO operations with several devices. Bytes of data are interleaved together and routed to or from the selected IO devices and to or from the desired locations in main storage. The channel's single data path is time-shared by the concurrently operating devices.

Selector channels are capable of operating only one device at a time; however, they permit data rates over a million bytes per second. Devices such as disc files operate only with selector channels, other devices operate in all data transfer modes. As with the multiplexor channel, the selector channel is invisible to the program; all instructions are directed at the device rather than the channel.

3.3 Operation of the Multiplexor Channel and Interrupt

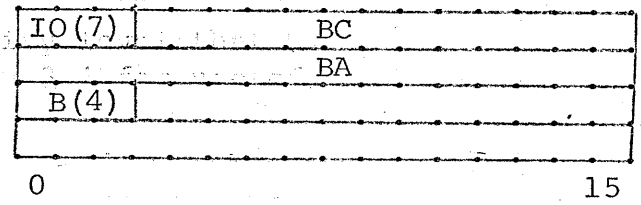
Devices require attention at the completion of every task. When operating in the multiplexor channel mode, this attention is automatically provided by the hardware without the need for program intervention until the transfer of the block of data is completed. Data transfers under program control require program intervention for every byte (word).

A device signals that it requires attention by requesting service at the priority level that has been assigned. (See section 3.4 for use of REQ, ENABLE, and LOW status bits.) When the priority of the active process drops below the priority of the request, the interruption occurs. The state of the old process is stored as part of its general register set (R_0 and R_1 contain the PSW) and a new general register set is switched in. Processor hardware then requests the device to transmit its address and its LOW bit; this 7-bit number is ORed into bit positions 8-14 of a word with bit position 7 set to 1 and all other positions 0. This address, called the interrupt address, lies somewhere in field 1.

Subsequent operation depends on the word found at the interrupt address. Any instruction class other than IO is executed; for predictable results, such an instruction must be an unconditional branch to the appropriate IO service routine for program-controlled transfer operation. An IO class instruction signifies that the device is under multiplexor channel control. A byte (word) of data is read from (written to) the device and packed into memory (unpacked from memory). The byte address pointer and byte counter are updated. If the byte counter went to zero indicating that the last byte (word) had been transferred or that the device indicated an unusual condition, the device will re-interrupt and, hence, the instruction following the IO class instruction is also executed. This is normally a branch to an IO service routine that re-initializes the device and channel for subsequent operations. If no unusual condition was detected and the byte counter did not overflow,

the device continues operating and will re-interrupt with the next data byte.

The format of the words starting at the interrupt address for multiplexor channel operation are given below. The doubleword at that address for program-controlled transfer is a simple single word or doubleword branch instruction. Branch instructions are normally unconditional, direct.



BC stands for byte counter and maintains a count of data bytes as they are transferred to and from the device. Prior to each transfer, BC is incremented to determine whether or not this is the last byte. When initializing a device for multiplexor channel operation, the programmer must load BC with the two's complement of the number of bytes to be transferred. At the end of channel operation, the entire word at the interrupt address will be set to zero. Exceptional conditions which cause termination before the specified number of bytes is read (written) leave the word nonzero.

BA stands for byte address and maintains the address of the data byte next to be transferred to and from memory. Prior to each transfer, BA is incremented to form the byte address of the data byte. This byte address is shifted right before use to form a word address, the end bit determines which half of the word the data byte will be loaded into. A 1 indicates the left byte, a zero the right byte. When the program initializes the channel, it must load BA with the byte address of the first byte to be transferred.

Unless the word executed at the interrupt address is a branch class instruction, control immediately returns to the interrupted process. The priority level is restored and the processor continues with the old (pre-interruption) program counter, status, and other general registers.

3.4 Basic Peripheral Structure

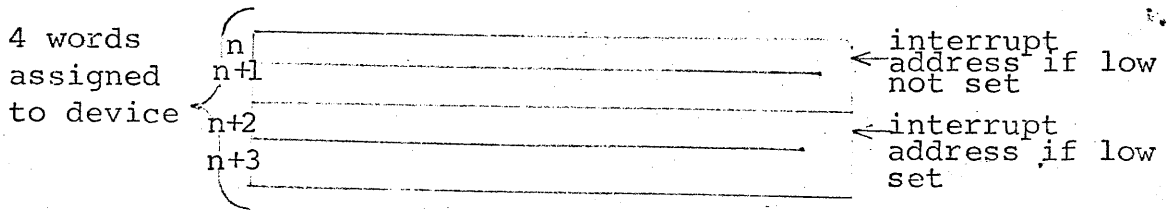
3.4.1 Status Register

All PDP-X peripherals are normally controlled by loading their status register using IOC or IOX commands; however, certain simple devices (paper tape or teletype) may respond directly to IOW and IOR; the status register may be sensed using the IOS or IOT commands. All PDP-X peripheral unit data transfers are accomplished using IOR and IOW instructions which may also modify the status register. The six low order bits comprise the basic section of the status register which is common to all peripherals.

UNUSUAL	DIR	REQ	BUSY	LOW	ENABLE
10	11	12	13	14	15

ENABLE - The ENABLE bit connects the peripheral to the interrupt system; with ENABLE set, either REQ or UNUSUAL cause an interrupt. With this bit cleared, the peripheral may in no way affect the operation of the rest of the system. ENABLE is cleared through the command line RESET on the IO Bus during the power up/down sequence, by instruction, and from the console.

LOW - The LOW bit indicates which of two possible interrupt addresses will be used by the processor when the peripheral interrupts. Refer to the figure below. In addition, LOW normally specifies which of two priority levels will be used to request the interrupt, LOW = 1 indicating a lower priority.



LOW is set whenever UNUSUAL is set. In normal use, LOW = 0 indicates normal data transfer on the multiplexor channel, a byte is transferred each time REQ rises. When the block of bytes is completely transferred (overflow), LOW and REQ are set, causing an interrupt at the second interrupt address.

BUSY - The BUSY bit indicates that the device is performing the requested function. It may be explicitly set by an IOC instruction or implicitly by an IOW or IOR instruction. BUSY is cleared when the device has completed its operation, REQ is set by the device when BUSY is cleared.

REQ - The REQ bit is set whenever the device requires attention under normal conditions. If ENABLE is also set, an interrupt will occur. REQ may be cleared by an explicit IOC instruction and is implicitly cleared during byte transfers on the multiplexor channel.

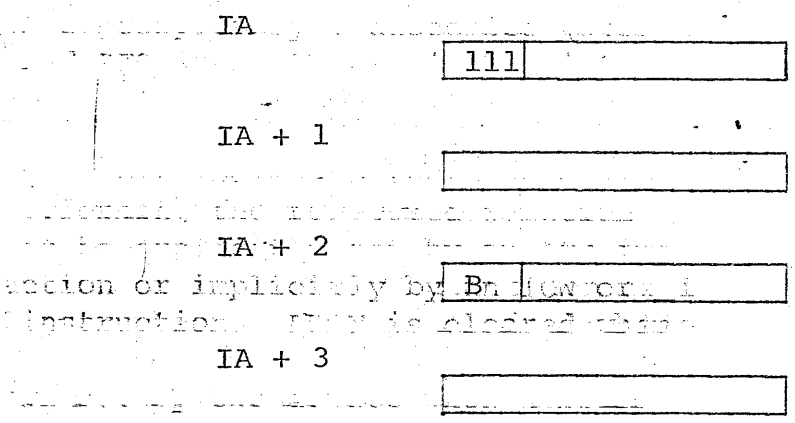
DIR - The DIR bit indicates the transfer direction of the device. Since it is a function either of the device type or device function, it may not be explicitly changed, although it may be sensed. DIR is sensed by the multiplexor channel to determine the direction of data transfer.

1 = out of processor into device
0 = out of device into processor

UNUSUAL - The UNUSUAL bit indicates error status or that some non-normal event has occurred. Some unusual conditions may only be cleared by operator intervention. Others may be cleared by clearing an error bit elsewhere in the status register. When UNUSUAL is raised, so is LOW. If ENABLE is also set, an interrupt will occur.

The interrupt address is computed as follows:

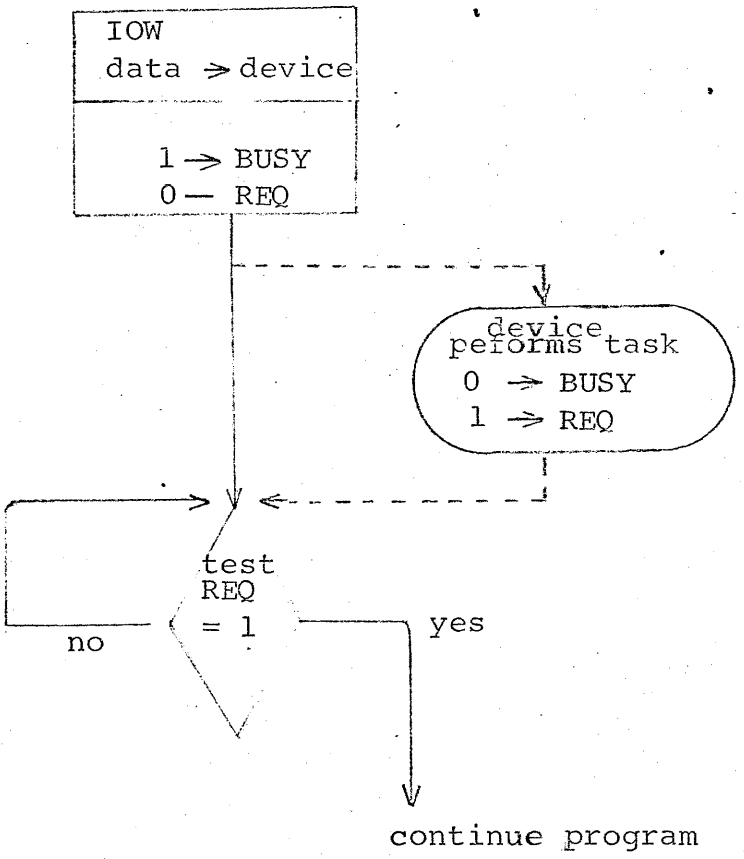
$IA = 400_8 + 2*\ (2*\ DEVICE\ \#\ +\ LOW)$



```
; General Subroutine to Punch a Block of Data
; Called by BAL PUNCH
;           Number of Data Bytes
;           Byte Pointer to Data
;           Return when Punch has Started
PUNCH:      TST PUNLOK          ; IS PUNCH ROUTINE BUSY?
            BZ  PUNCH           ; IF ZERO, IT IS STILL BUSY, WAIT
            STA 3, PUNLOK       ; IT IS DONE - SAVE AC3
            LDA 3, (2)          ; GET COUNTER
            NEG 3               ; FORM TWO's COMPLEMENT
                                ; AS REQUIRED BY CHANNEL
            STA 3, PUNINT       ; STORE COUNTER IN CHANNEL
            LDA 3, 1 (2)        ; GET POINTER
            STA 3, PUNINT + 1   ; STORE POINTER IN CHANNEL
            LDA 3, PUNLOK       ; RESTORE AC3
            CLR    PUNLOK       ; SET LOCK TO ZERO
                                ; START UP PUNCH BY ENABLING
            IOC, PTP, [11]      ; TO INTERRUPT AND SETTING
                                ; REQ, PUNCH WILL START
                                ; IMMEDIATELY
            B       2(2)        ; EXIT ROUTINE
PUNLOC:     -1                  ; 0 MEANS BUSY, -1 MEANS
                                ; NOT BUSY
; Unusual or usual end interrupt handler
PUNDON:     IOT  PTP, [40]      ; TEST UNUSUAL BIT
```

```
                BN    PUNERR          ; OUT OF TAPE IN PUNCH, GO TO ERROR
                COM   PUNLOK          ; MAKE LOCK NONZERO (NON-BUSY)
                IOC   PTP, [0]        ; CLEAR OUT PUNCH
                PSD                   ; DEBREAK AND DISMISS
; Interrupt address
LOC PUNINT
                BLOCK 2               ; 2 LOCATIONS FOR CHANNEL
                B     PUNDON          ; BRANCH TO DONE
                                      ; ON OVERFLOW OR
                                      ; OUT OF TAPE
```

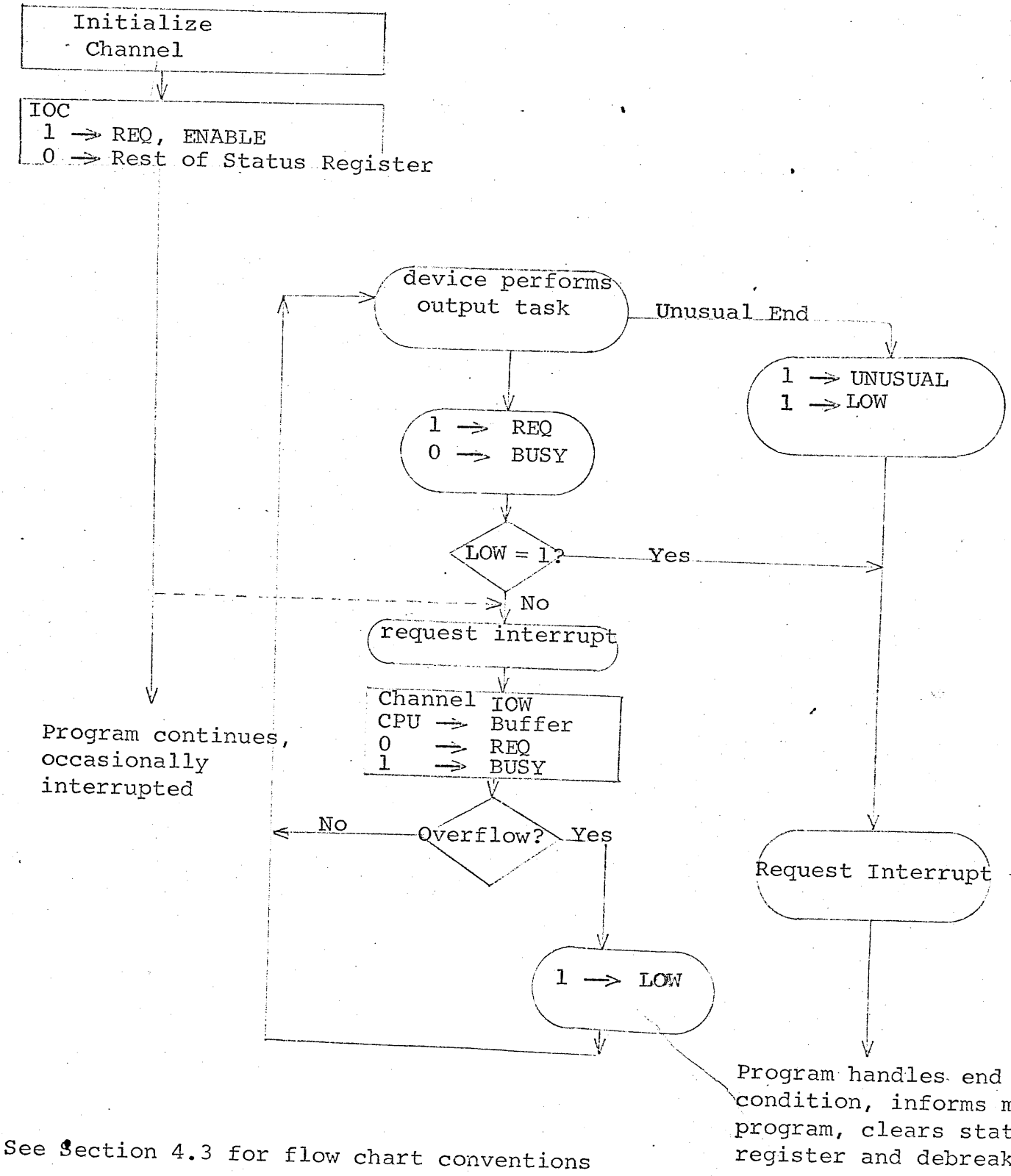
3.4.2 Output Device - no interrupt, output one character




```
; example for paper tape punch

    IOW   PTP,  data          ; load byte in punch
                              ; set BUSY clear REQ
    IOT   PTP,  [10]          ; test  REQ (done)

     BZ .-1                   ; not set, go back to test
```

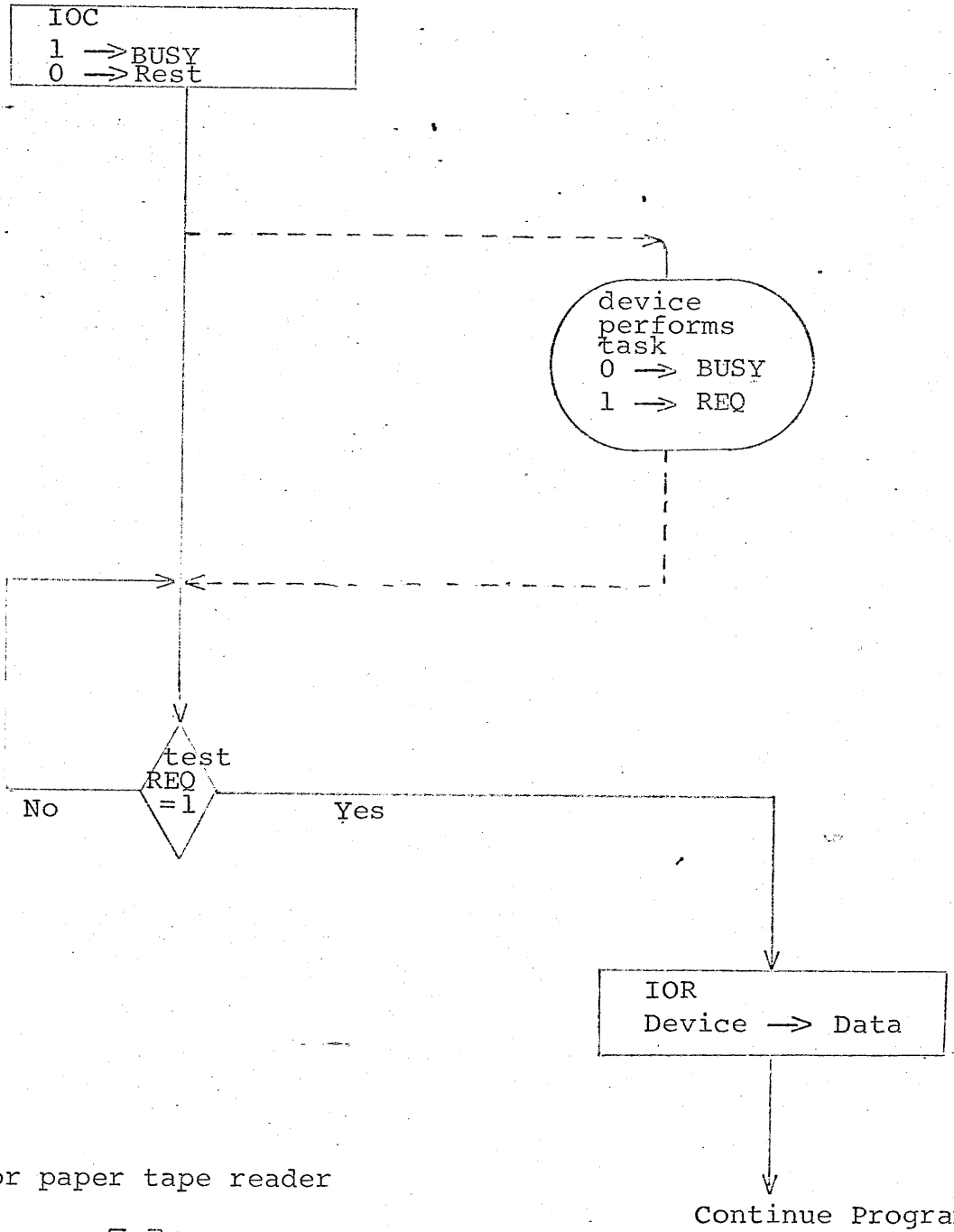
(See Section 4.3 for flow chart conventions.)

3.4.3 Output Device - interrupt mode (multiplexor channel)



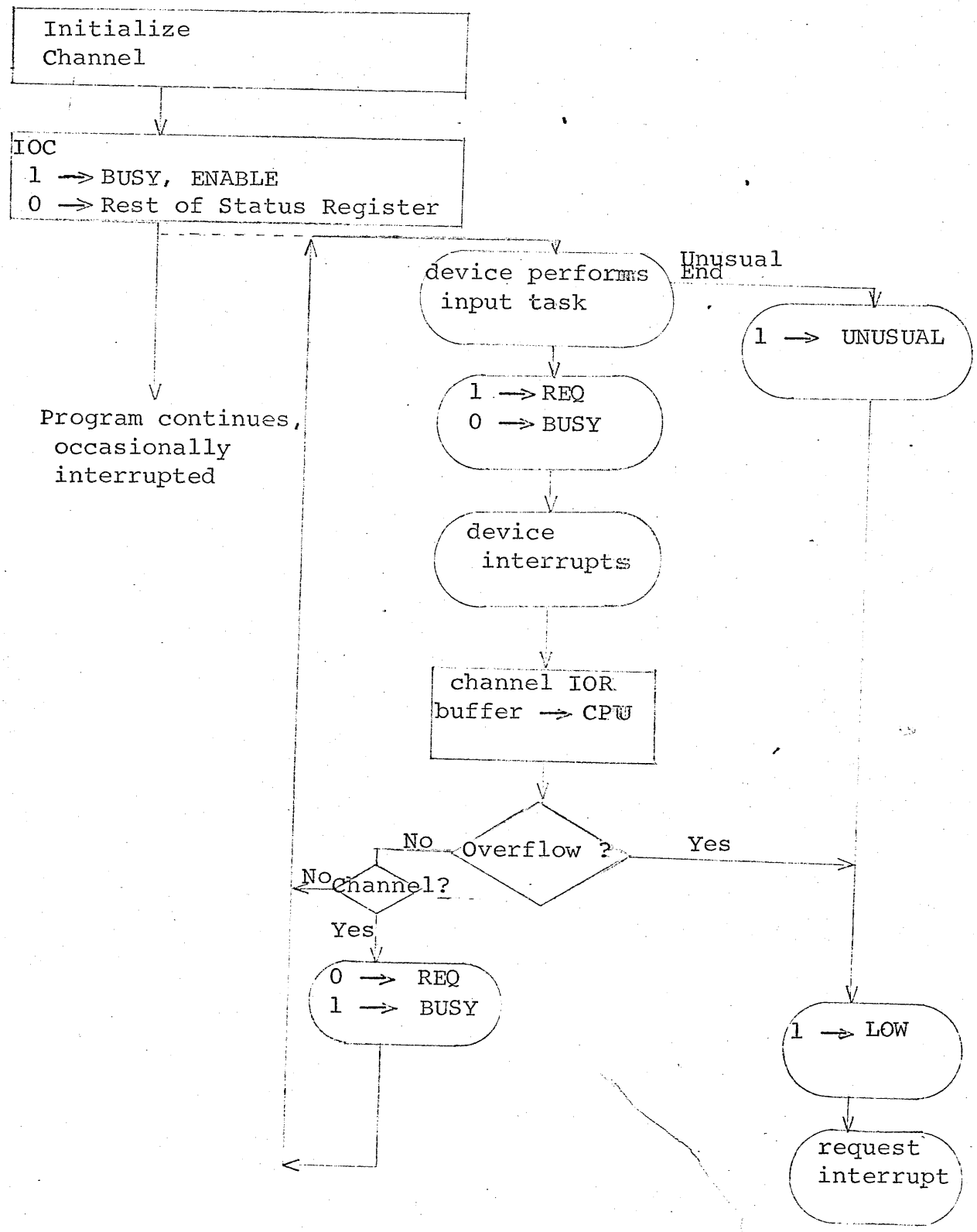
See Section 4.3 for flow chart conventions

3.4.4 Input Device - no interrupt, input one character




```
; example for paper tape reader

        IOC   PTR, [4]    ; start reader

        IOT   PTP, [10]   ; test REQ

        BZ    .-1         ; not set, go back to test

        IOR   PTR, data   ; read in data byte
```

3.4.5 Input Device - interrupt mode (multiplexor channel)



3.5 Paper Tape Peripherals

(Note that all are independent and may be simultaneously operating at their maximum rates.)

3.5.1 Paper Tape Reader/Punch

The paper Tape Reader/Punch (PTR/PTP) is built around the PC01 Reader/ Punch mechanism. It is intended for use where high-speed paper tape operations are required. Fan fold tape is normally used.

The tape reader operates at a maximum rate of 300 bytes (tape lines) per second; each byte may be program interpreted as either binary or alphanumeric (ASCII) data. A byte is read every 3.3 milliseconds, the reader stops if not reselected (set BUSY bit to 1 within 1.6 milliseconds after REQ comes up) between characters. Maximum reading rate is only obtained after reading approximately 10 characters consecutively.

The tape punch operates at a maximum rate of 50 bytes (tape lines) per second. Power to the punch motor is program controlled, "warm up" before punching the first character is 1 second; power remains up for 5 seconds after the last character is punched. Operation of the punch is synchronized with the motor, thus, the full rate may be obtained only if BUSY is set within 5 milliseconds of the previous REQ rise.

The status register of both the reader and the punch follow the general format. The UNUSUAL bit is raised to indicate that the device is out of tape; all 1's will be read by the reader, the punch may still punch approximately 10 lines. The out-of-tape condition, if ignored, does not stop the punch.

3.5.2 Keyboard/Printer

The keyboard/printer (TTI, TTO) is built around Teletype Corporation's Models 33, 35, and 37 ASR or KSR mechanism. The connection between the mechanism and logic is a simple four-wire cable so that the mechanism is easily remoted from the

processor. The ASR models have an attached paper tape reader/punch. Both the keyboard and printer operate at a maximum rate of 10 bytes per second; since operation is full duplex (completely independent), the program must echo any characters from the keyboard it wants printed. The status byte for both follow the standard format; there is no UNUSUAL bit.

BUSY is set in the keyboard when a key is struck; it is lowered and REQ set when the byte is available for input. If busy is set by the program, the (optional) reader mechanism is started (if enabled by the switch on the mechanism). The (optional) punch mechanism punches whatever is printed (when enabled by the switch on the mechanism).

3.5.3 Paper Tape Peripherals Status Bytes

PTR Paper Tape Reader		UNUSUAL (Out of Tape)	DIR =0	REQ	BUSY	LOW	ENABLE
PTP Paper Tape Punch		UNUSUAL (Out of Tape)	DIR =1	REQ	BUSY	LOW	ENABLE
TTI Keyboard			DIR =0	REQ	BUSY	LOW	ENABLE
TTO Printer			DIR =1	REQ	BUSY	LOW	ENABLE

X indicates permanently 0

3.6 Device Assignment Table

Number	Type	(Field 1) Location
00-02	internal features (reserved)	
03	power fail/parity	414
04-07	initiated priority interrupts	420-437
10	TTO	440
11	TTI	444
12	PTP	450
13	PTR	454
14	Small Display with Light Pen	460
15	Card Readers	464
16	Incremental Plotter	470
17	Relay Buffer	474
20-27	A/D/A; multiplexor, etc.	500
30-37	four extra full duplex TTY's	540

3.7 IO Bus

3.7.1 General characteristics:

1. Electrical - A twisted pair signaling system similar to CDC 3000 series is used.

2. Interlock - DC interlocked control signals are used to insure reliable operation over extremely long distances and permit arbitrarily fast devices physically close to the processor.

3. Bi-directional - The signaling systems permits bi-directional signal flow; this will be utilized on the data lines.

4. General line format -

   8 Bi-directional address/data lines

   8 Outward control lines

   8 Inbound control lines

   8 Inbound priority request lines

   8 Outbound priority interrupt grant lines

5. Power control and ground return

3.7.2 Operation

The connection between the processor and the IO device control units is called the IO Bus. The interface consists of signal lines that connect the control units to the processor; except for the signal used to establish priority, all communication lines to and from the processor are common to all control units. At any one instant, however, only one control unit may be logically connected to the processor. The logical connection is maintained from the time it is first extablished by the processor until it is broken by the processor. The rise and fall of all signals transmitted over the interface are controlled by interlocked responses. This interlocking removes the dependence of the interface on circuit speed and bus length, making it applicable to a wide variety of circuits and data rates.

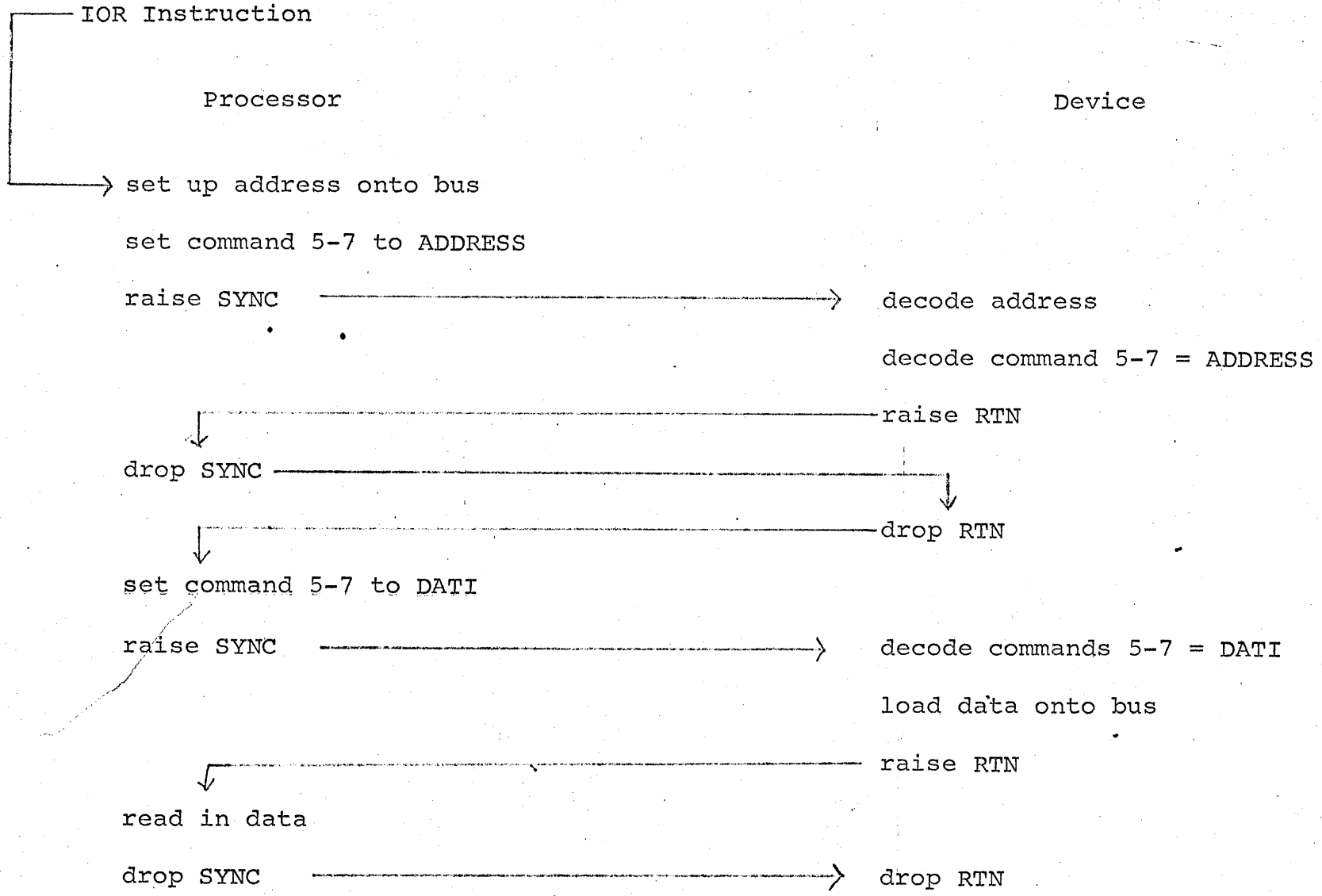
48 signals comprise the bus including 32 control signals, data signals, power, ground, and spares. The priority signals are retransmitted by each device, all others are common to every device. The data lines are timing-shared between data and address information. At the beginning of an IO operation, these lines are used by the processor to transmit a device address, later in the operation, they are used to transmit the data bytes. The data/address bus is bi-directional.

The priority lines are used to synchronize requests from devices at the various priority levels. A device requesting attention at a particular priority level does so by transmitting its request on the appropriate line of the eight available. The processor IO section grants such a request by transmitting on the associated priority out line from the processor. Devices not requesting retransmit the grant signal; the first one requesting (nearest to the processor on that level) blocks it and transmits its address to the processor on the data lines.

The control signals are used to establish the mode of operation on the bus. One control line out (SYNC) is the synchronizing signal; its rise indicates that all other control and address information is correct and may be strobed. One control line in (RTN) is similarly used. The other seven lines are decoded to indicate data/command, in/out, multiple/single byte, etc. Upon a) receipt of SYNC, b) control lines decode to address, c) and its address is on the data lines, then a device becomes selected and will respond to subsequent commands. The device indicates that it has become selected by transmitting the RTN signal. Refer to the figure.

With the rise of RTN which indicates that a device has become selected and is ready to accept a command, the processor drops SYNC, resets the control lines to the appropriate command code, and prepares to raise SYNC again as soon as RTN drops. The subsequent RTN rise, SYNC fall, RTN fall triplet completes the byte transfer. A word transfer requires raising SYNC followed by another such triplet.

If during a selection sequence a device fails to respond (e.g., addressing anon-existent or disconnected device) automatic time out circuitry causes the processor to continue, indicating this failure to the program.


IOR Instruction
Processor
Device
set up address onto bus
set command 5-7 to ADDRESS
raise SYNC
decode address
decode command 5-7 = ADDRESS
raise RTN
drop SYNC
drop RTN
set command 5-7 to DATI
raise SYNC
decode commands 5-7 = DATI
load data onto bus
raise RTN
read in data
drop SYNC
drop RTN

3.7.3 Line Definitions

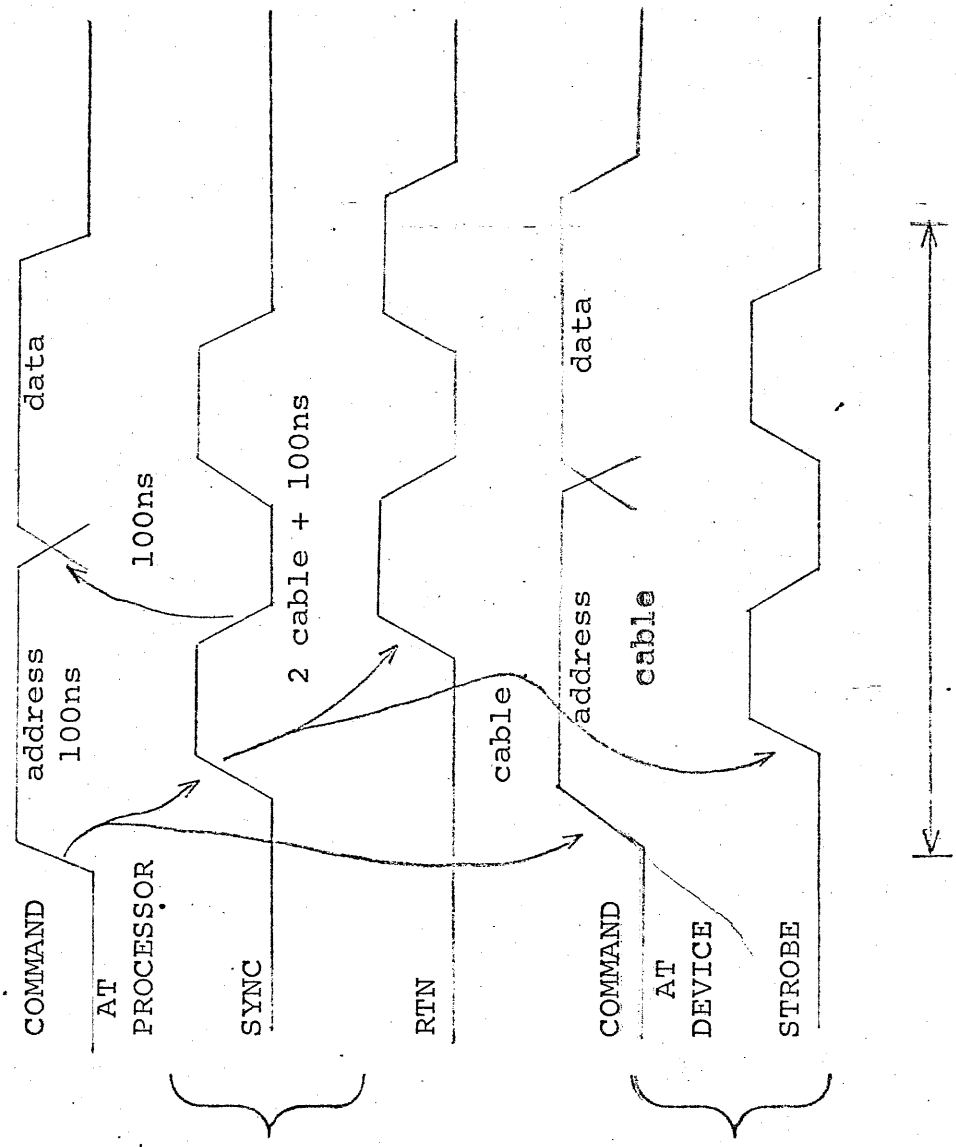
Line (direction)	X	State	Function
Command X Out			Processor → device control information
	0		SYNC - assertion indicates validity of other command lines.
	1,2		Reset conditions
		0	normal operating state
		1	processor has stopped
		2	load (reset to state for automatic read-in of bootstrap program)
		3	reset, clear all device status registers
	3,4		spare
	5,6,7		command mode, information content of data lines
		0	DATO - data to device
		1	DATI - data to processor, channel operating
		2	ADDRESS - address to device
		3	DATI END - data to processor, no channel

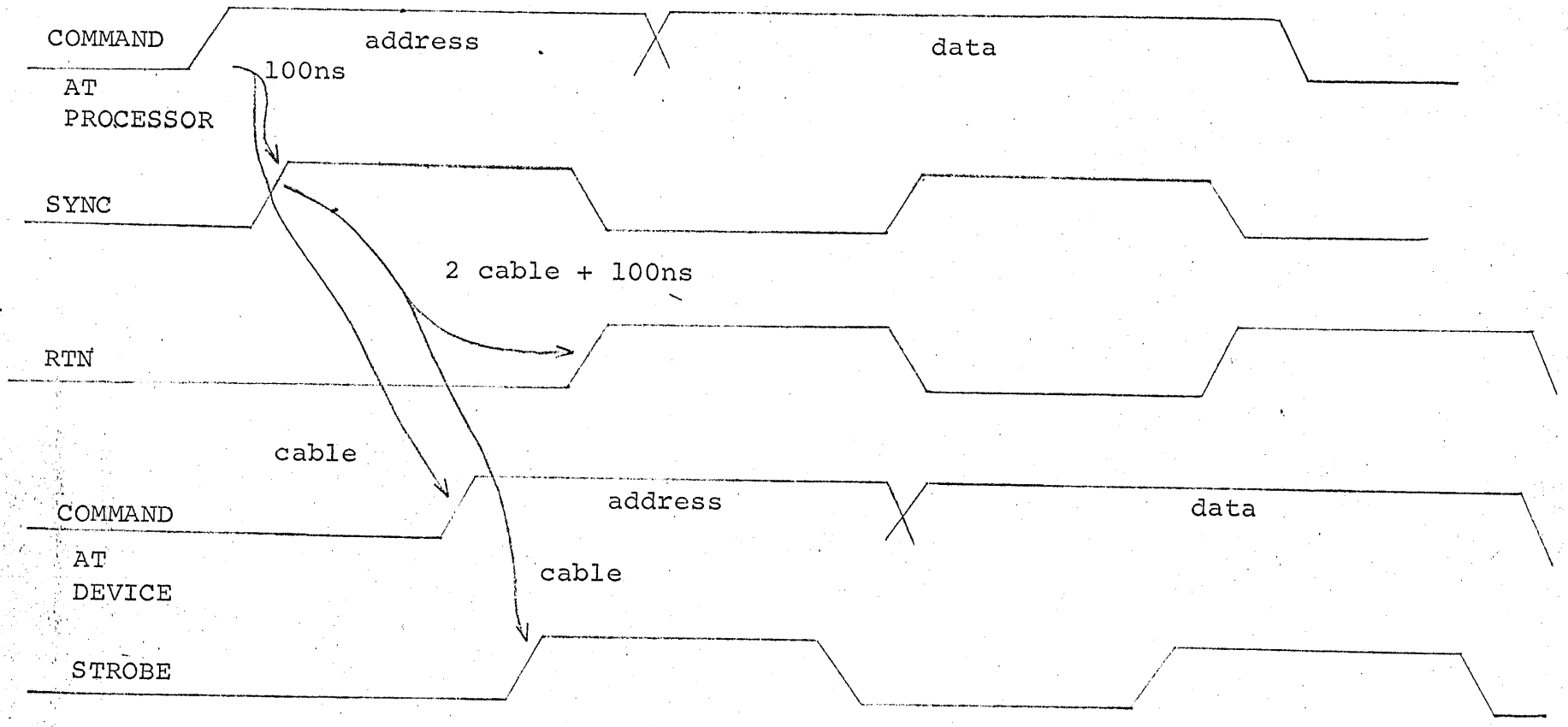
Line (direction)	X	State	Function
		4	DATO LAST - data to device, channel overflow
		5	DATI LAST - data to processor, channel overflow
		6	CONO - command to device
		7	CONI - device status to processor
Response X In			Device → processor response to command
	0		RTN - assertion in response to SYNC to interlock bus, assertion indicates validity of other response lines
	1		16D - 16-bit data word required
	2		16C - 16-bit command word required
	3		spare
	4,5		special interrupt controls for channel
		0	normal
		1	inhibit BA count in channel
		2	force last cycle of channel operation after transmitting 16-bit address
		3	permit only BC cycle of channel operation

Line (direction)	X	State	Function
	6,7		Direction control
		0	OUT - processor to device
		1	
		2	IN - device to processor
		3	ADD IN - add device data to processor
Request X In	1-7		Device → processor interrupt request lines; seven is highest priority.
Data X In/Out	0-7		Device ↔ processor data/address lines.
Priority X Out	1-7		Processor → device interrupt request grant lines; instructs device to transmit its address; these lines are retransmitted if used; seven is highest priority, one lowest.
Gnd	1		Heavy ground wire interconnection
Pwr	1		Power supply for remote turn on/off and priority line receiver/transmitter

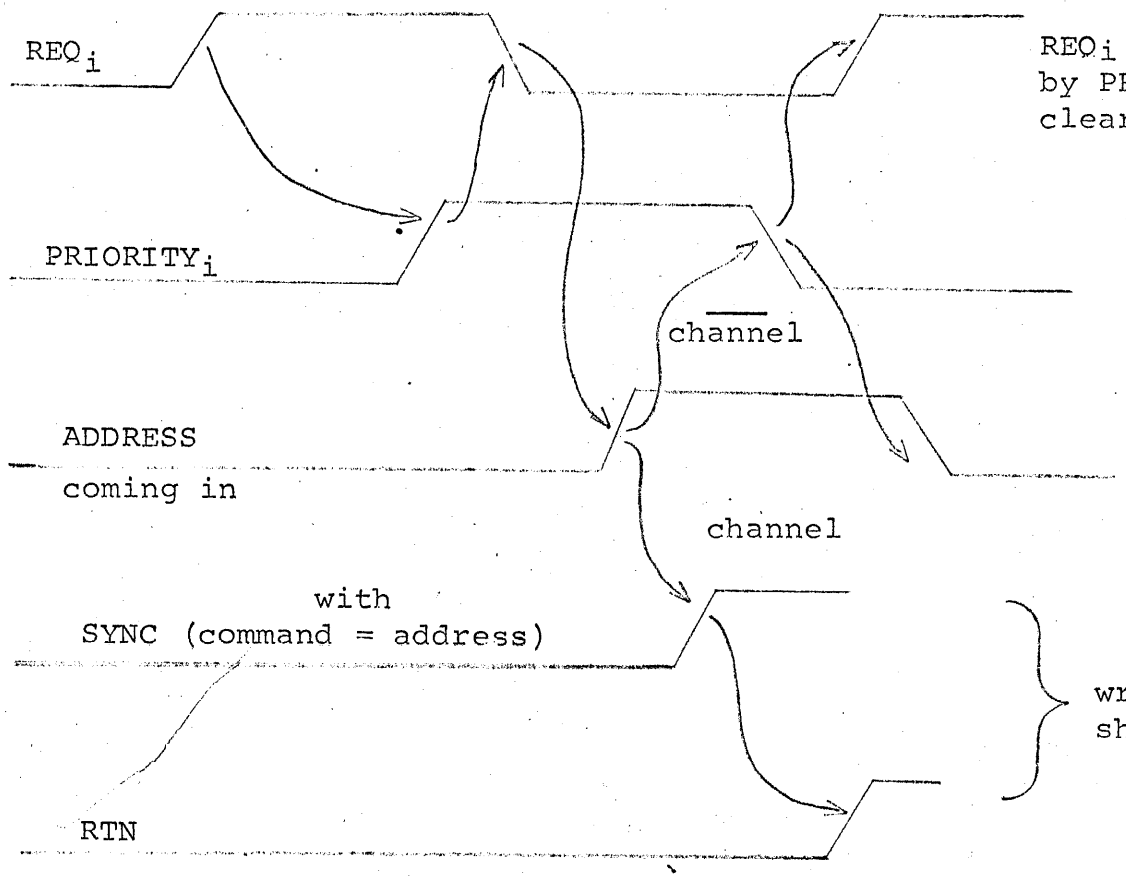
3.7.4 Basic Bus Timing/ Flow Diagrams

Write Operation (Single Byte)
For 25 feet of cable; period = 850ns



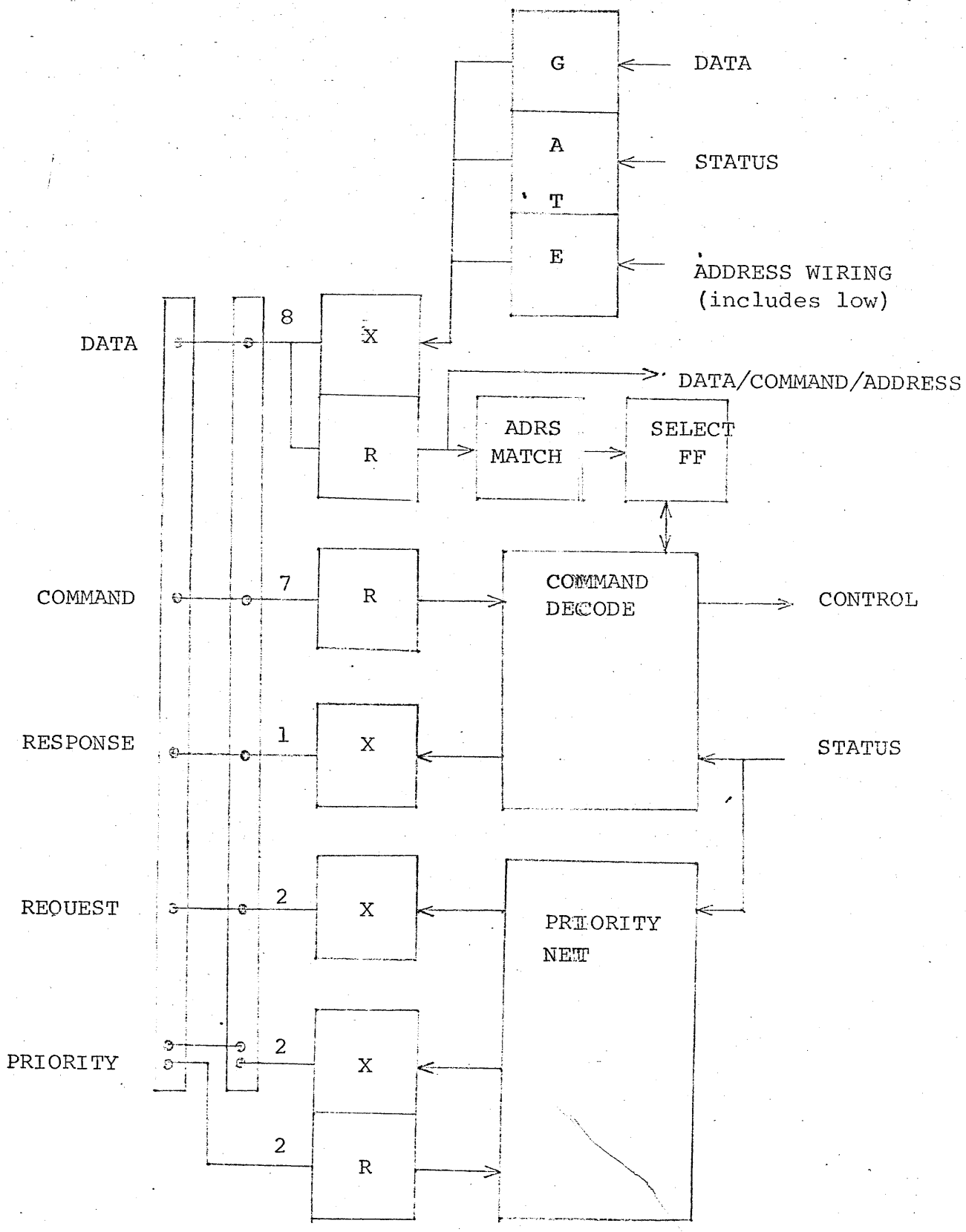


Write operation (single byte)
For 200 feet of cable; period = 4μs



INTERRUPT SYNCHRONIZING SEQUENCE

3.7.5 Basic Control Unit Block Diagram



4.0 Appendices

4.1 Assembly Language Conventions

To facilitate the interchange of code for the PDP-X, some assembly language conventions will be briefly stated below. It is to be noted that these are in no way complete nor are they necessarily indicative of the final form of the assembly language for the PDP-X.

The general syntax is derived from MACRO-10. The generalized instruction statement will have the form:

TAG: OPCODE R, @ ADDRESS (INDEX) ; COMMENT

This may be summarized by

1. Symbols may be six characters long, must begin with a letter and may contain any of the characters A-Z, 0-9, %, . .

2. : Delimits the tag field. The symbol to the left of the colon is assigned a value equal to the current location counter.

3. , Separates the R-field and the address field.

4. @ Indicates indirect addressing and, in an instruction, forces long form. In an address constant, it sets bit 0 to a 1. For example, @A would generate a 16-bit constant with bits 1-15 containing the value of A and with bit 0 a 1.

5. ( ) Surround the index field. If no index field is specified, the assembler will pick the appropriate addressing mode. For example, in location 1000 (octal), the instruction LDA 2, 100 would generate an index field of 0 and D_1 would contain 100.

   LDA 2, 1010 would generate an index field of 1 and D_1 would contain 010.

   LDA 2, 100 (3) would generate a short form indexed instruction. D_1 would contain 100.

LDA 2, 1010 (3) would generate a long form indexed instruction. D_1 would contain 200, D_2 would contain 001010.

LDA 2, 200 would force long form. D_1 would contain 200, D_2 would contain 000200.

6. ; Delimit comments. Comments extend to the next CR-LF pair.

7. · Has the value of the current location counter.

8. = Indicates assignment. The symbol to the left of the = is assigned the value of the expression to the right of the =.

9. In I/O instructions, the device number replaces the R-field. For example, if TTY has the value of the teletype device number, IOW TTY, MEM will generate a doubleword with the device number in D_1, X = 0, D_2 = MEM.

10. In instructions where the R-field is interpreted as part of the operation code, the comma may be omitted. For example, BZ , Y and BZ Y are both legal and equivalent, whereas BZ 3, Y is illegal. The R-field is ignored and the error flagged.

11. [ ] Are used to indicate literals.

12. ↑ used to indicate a local radix change (see MACRO-10 User's Manual). Valid forms are:

    ↑B binary (0-1)
    ↑O octal (0-7)
    ↑D decimal (0-9)
    ↑X hexadecimal (0-9, A-F)

13. PSEUDO instructions

    a. RADIX EXPRESSION – Global radix is changed.

    b. BOUND EXPRESSION – Where expression has a value equal to some power of

two. This will move the location counter to the appropriate word boundary. For example, BOUND 4 will move the location counter to the next multiple of 4.

c. LOC EXPRESSION - Will set the location counter.

d. PHASE EXPRESSION
DEPHASE - See MACRO-10 User's Manual.

e. BLOCK EXPRESSION - Generates a block of zeros.

f. LPOOL EXPRESSION - Generates a literal pool of size equal to the value of the expression. (See SDS Sigma 2 Symbol Reference Manual.)

14. Expressions - Any atom (symbol or number) can be combined with the operator's

+ Add
- Subtract
/ Divide
* Multiply
& Logical and
! Logical or
( ) for evaluation procedure

(See MACRO-10 User's Manual for effect upon relocation.)

15. In general, the assembler will optimize short program segments and produce relocatable binary output. The size of the largest optimizable segment will be determined by the available memory size. Programs will consist of several of these segments loaded by a linking-loader. Core images can then be saved.

4.2 Instructions (alphabetic)

mnem	Definition	OP	D_1 (octal)	R	CC	Models
ADD	ADD	3	--	--	0,1,2	I,II
AND	AND	2	--	--	1,2	I,II
B	Branch	4	--	3	--	I,II
BAL	Branch And Link	4	--	7	--	I,II
BCN	Branch if Carry Nonzero	4	--	0	--	I,II
BCZ	Branch if Carry Zero	4	--	4	--	I,II
BM	Branch if result Minus	4	--	1	--	I,II
BN	Branch if result Nonzero	4	--	2	--	I,II
BP	Branch if result Positive	4	--	5	--	I,II
BZ	Branch if result Zero	4	--	6	--	I,II
CLR	CLeaR	5	--	7	1,2	I,II
CMP	CoMPare	6	111	--	1,2	II

mnem	Definition	OP	D_1 (octal)	R	CC	Models
COM	COMplement	5	--	1	1,2	I,II
DIV	DIVide	6	103	--	0,1,2	II
INC	INCrement	5	--	2	1,2	I,II
IOC	IO Command	7	--	5	1,2	I,II
IOR	IO Read	7	--	0	1,2	I,II
IOS	IO Status	7	--	1	1,2	I,II
IOT	IO Test status	7	--	2	1,2	I,II
IOW	IO Write	7	--	4	1,2	I,II
IOX	IO Xor status	7	--	7	1,2	I,II
LBM	Load Both Maps	6	12	6	--	IIc
LCMP	Logical CoMpare	6	110	--	1,2	II
LDA	LoaD Accumulator	0	--	--	--	I,II
LDC	LoaD Character	6	114	--	--	II
LDIV	Logical DIVide	6	102	--	0,1,2	II

mnem	Definition	OP	D_1 (octal)	R	CC	Models
LMUL	Logical MULtiply	6	100	--	1,2	II
LMM	Load Monitor Map	7	10	6	--	IIc
LSM	Load Selected Map	7	13	6	--	IIc
LUM	Load User Map	7	11	6	--	IIc
MUL	MULtiply	6	101	--	1,2	II
NEG	NEGate	5	--	3	1,2	I,II
POB	POp and Branch	6	116	6	--	II
POC	POp and Count	6	116	4	--	II
POL	POp, branch and Link	6	116	7	--	II
POP	POP	6	116	5	--	II
PSC	Priority System Clear	7	6	6	--	IIb
PSD	Priority System Dismis	7	7	6	--	I,II
PSI	Priority System Inhibit	7	4	6	--	IIb
PSR	Priority System Request	7	5	6	--	IIb

mnem	Definition	OP	D_1 (octal)	R	CC	Models
PUSH	PUSH	6	116	1	--	II
PUB	PUsh and Branch	6	116	2	--	II
PUC	PUsh Count	6	116	0	--	II
PUL	PUsh, branch and Link	6	116	3	--	II
RIO	Reset IO system	7	001	6	--	I,II
RL	Rotate Left	5	--	5	0,1,2	I,II
RR	Rotate Right	5	--	4	0,1,2	I,II
SHFT	SHiFT	6	113	--	0,1,2	II
STA	STore Accumulator	1	--	--	--	I,II
STC	STore Character	6	115	--	--	II
SUB	SUBtract	6	112	--	0,1,2	II
SWP	SWaP bytes	5	--	6	1,2	I,II
TST	TeST	5	--	0	1,2	I,II
TSTC	TeST and Complement	6	107	--	1,2	II

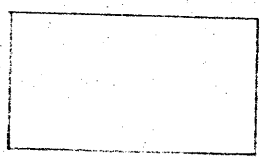
mnem	Definition	OP	D_1 (octal)	R	CC	Models
TSTN	TeST but change Nothing	6	104	--	1,2	II
TSTO	TeST and set to Ones	6	106	--	1,2	II
TSTZ	TeST and Zero	6	105	--	1,2	II

4.3 Flow Chart Conventions

. . . . . .

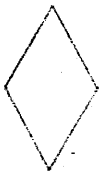
Implicit transfer of control. Since the device proceeds asynchronously with the processor.

Explicit transfer of control.



Operation performed by the processor either under control of the program or under control of the multiplexor channel.

Operation performed by the device.



A test made either by the device (implicitly) or by the processor under program control.