

PROGRAMMED DATA PROCESSOR



**PROGRAMMING
MANUAL**

PDP-4
PROGRAMMING MANUAL

TABLE OF CONTENTS

CHAPTER 1 SYSTEM DESCRIPTION

	<u>Page</u>
Summary	1
System Description	1
Arithmetic and Control Element	3
Interface	3
Input-Output Devices	6

CHAPTER 2 ARITHMETIC AND CONTROL ELEMENT

Functions	9
Internal Processor	9
Control States	15
Instruction Operation	16
Memory Reference Instructions and Auto Indexing	16
Augmented Instructions	20

CHAPTER 3 INPUT-OUTPUT EQUIPMENT FUNCTIONS AND PROGRAMMING

Input-Output Commands	27
Device Selector	27
Information Collector	29
Information Distributor	30
Input-Output Skip Facility	30
Program Interrupt Control	30
Clock/Timer	31

	<u>Page</u>
Input-Output Devices	33
Visual CRT Display, Type 30A	33
Light Pen, Type 32	35
Visual CRT Display, Type 30D and Light Pen, Type 30D	35
Analog-to-Digital Converter (Typical Input Device)	36
Low Speed Analog-to-Digital Converter	39
Perforated-Tape Reader and Control	39
Printer-Keyboard and Control, Type 65	43
Perforated-Tape Punch and Control, Type 75.	50
Card Reader and Control, Type 40-200-4.	53
Card Punch, Type 41-523-4	56
Line Printer, Type 62.	61

CHAPTER 4

THE INTERFACE ELECTRICAL CHARACTERISTICS

Real-Time Option	63
The Device Selector	63
Information Collector.	65
Information Distributor	66
Input-Output Skip Facility	66
Program Interrupt Control.	68
Data Interrupt Control	68

Appendix 1	Memory Reference Instructions	A-1
Appendix 2	Operate Instructions	A-2
Appendix 3	Basic IOT Instruction Group	A-3
Appendix 4	Codes	A-6
Appendix 5	Read-In Mode Sequence	A-10
Appendix 6	Assembler Program	A-13
Appendix 7	Multiply Subroutine.	A-18

		<u>Page</u>
Appendix 8	Divide Subroutine.	A-19
Appendix 9	Input-Output Equipment Characteristics	A-21
Appendix 10	Powers of 2	A-22

FOREWARD

This manual is for programmers and users of the Programmed Data Processor-4, a high speed, stored program, digital computer manufactured by the Digital Equipment Corporation.

Chapters 2 and 3 contain the detailed information necessary to make use of the machine.

Chapter 1 summarizes the machine's electrical and logical design. Chapter 4 presents information helpful in making the electrical connections to input-output devices. Appendices provide detailed data which may be helpful in specific programming assignments. Although program examples are given in this document, no attempt has been made to teach programming techniques.



Standard PDP-4 system

CHAPTER 1

SYSTEM DESCRIPTION

SUMMARY

The Digital Equipment Corporation Programmed Data Processor-4 (PDP-4) is designed to be the control element in an information processing system. PDP-4 is a single address, parallel, binary machine with an 18-bit word length using one's or two's complement arithmetic. Standard features of the machine are stored program operation, a random access magnetic-core memory, a complete order code, and indirect addressing.

Standard core memory size is 1024 or 4096 words, expandable to 8192 words. The memory cycle time is 8 microseconds. Instruction enactment times are multiples of the 8-microsecond memory cycle, with two-cycle instructions such as add, deposit, load, etc., performed in 16 microseconds. Indirect addressing requires an additional 8 microseconds.

Flexible, high-capacity input-output capabilities of the PDP-4 enable it to operate in conjunction with a variety of peripheral devices, such as perforated-tape readers and punches, punched-card readers and punches, teletype printer-keyboard, line printers, magnetic tape transports, and analog-to-digital converters.

The machine is completely self-contained, requiring no special power sources, air conditioning, or floor bracing. From a single source of 115-volt, 60-cycle, single-phase power, PDP-4 produces circuit operating dc voltages of -15 volts (± 1) and +10 volts (± 1) which are varied for marginal checking. Total power consumption is 900 watts. It is constructed with standard DEC 4000 series system modules and power supplies. Solid-state components and built-in marginal checking facilities insure reliable machine operation.

SYSTEM DESCRIPTION

The basic PDP-4 system is shown diagrammatically in Figure 1. Three portions of the system are delineated according to function: the Arithmetic and Control Element, the Interface, and the Input-Output Equipment. Information originates not only from the Input-Output Equipment

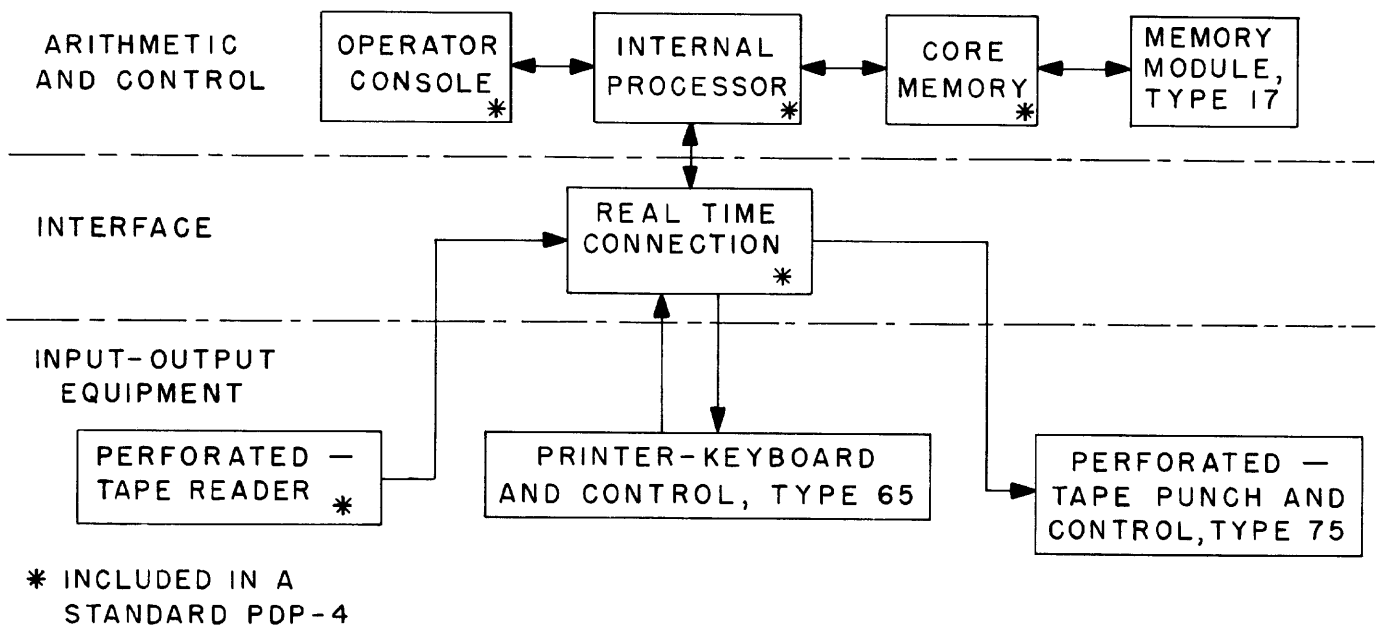


Figure 1 PDP-4 System with Real-Time Connection

but can be entered manually and modified at the Operator Console.

Arithmetic and Control Element

The Operator Console, Internal Processor, and Core Memory constitute the Arithmetic and Control Element. The Internal Processor carries out the arithmetic and logical operations, and controls the real-time connection and the core memory. Binary arithmetic with a fixed point is employed.

The Console is used to observe and control the action of the program and the Internal Processor, and to alter the contents of Internal Processor registers. The contents of Core Memory can be examined or new information deposited. All Internal Processor registers are displayed continuously.

Three memory capacities are available in PDP-4: 1024, 4096, or 8192 words. Standard models PDP-4A and PDP-4B come with 1024-word and 4096-word memories, respectively. The two models are identical in every other respect. The smaller memory has a 32 by 32 by 18 core array, the larger a 64 by 64 by 18 core array. A Memory Module Type 17, containing a 64 by 64 by 18 core array may be added to PDP-4B to give it a 8192-word storage capacity.

The cycle time (the time required to read information from memory and rewrite information back into memory) is 8 microseconds. The access time (the time required to read information from memory) is 2 microseconds. In the event of power failure, the contents of the Core Memory remain unaltered.

Interface

The Real-Time Connection, furnished as standard equipment, provides communication between the Internal Processor and the Perforated-Tape Reader, the Perforated-Tape Punch, and the Keyboard-Printer. The Real-Time Option Type 25, gives the system the additional capability to operate efficiently over a wide range of information handling rates (from seconds per event to 125,000 words per second) and with a large variety of input-output devices (see Figure 2). The Real-Time Option consists of a Device Selector, an Information Collector, an Information Distributor, an Input-Output Skip connection, a Program Interrupt facility, a Data Interrupt facility, and a Clock/Timer.

The Device Selector consists of decoding elements to select and establish the state of an

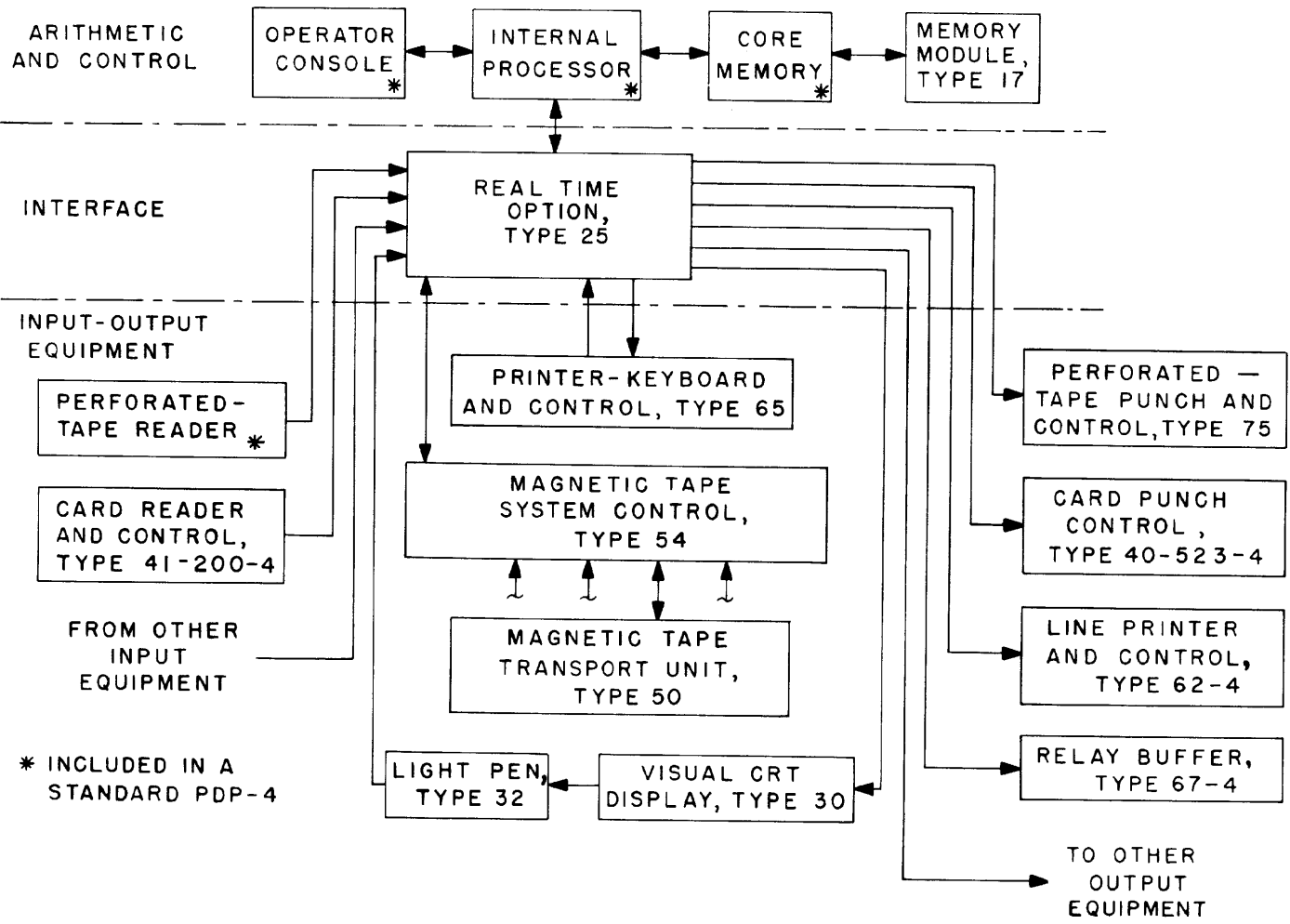


Figure 2 PDP-4 System with Real-Time Option

external device when the program issues an input-output transfer instruction. The direction of information transfer (in or out of the Internal Processor) is controlled by signals produced by the Device Selector. Up to 64 Input-Output devices can be selected and these, in turn, may cause the selection of many more. The standard Device Selector has provisions for twenty selector elements.

The Information Collector receives information from input devices (selected by the Device Selector) and transfers the information to the Internal Processor. Up to 18 bits of information can be collected simultaneously; 8 x 18 bits of information may be collected, broken into variable-sized words.

The Information Distributor distributes information from the Internal Processor to all output devices. Only the output device selected (or addressed) by the Device Selector samples and reads in the information contained in the Information Distributor. Up to 8 x 18 bits may be distributed.

The Input-Output Skip Connection provides a program skip instruction conditioned by the state of a given Input-Output device logic line. The instruction following the skip instruction will not be executed if the line is a ONE. Eight skip conditions may be sampled.

The Program Interrupt permits one of eleven lines (or conditions) or Input-Output devices to interrupt the program and initiate a subroutine which may return to the original program when the cause for interruption has been processed. The machine state is preserved during a Program Interrupt. This type of interrupt is suited for information or event rates in the range of 0 to 2,000 cycles per second.

The Data Interrupt allows a device to automatically interrupt the program and deposit or extract data from the Core Memory at an address specified by the device. The Data Interrupt is suited for high speed information transfers, since up to 125,000 18-bit words may be transferred per second.

The Clock/Timer produces a signal which increments a core memory register at a rate of 60 cycles per second. When the register overflows, a Program Interrupt occurs.

Input-Output Devices

All of the input-output Devices are optional except the Perforated-Tape Reader.

The Perforated-Tape Reader senses 5-, 7-, or 8-hole perforated-tape information at the rate of 300 lines per second. Either one line of tape (alphanumeric) or 3 lines of tape (binary word) may be read.

The Perforated-Tape Punch and Control, Type 75, perforates 5-, 7-, or 8-hole paper tape at a rate of 63.3 lines per second.

The Printer Keyboard and Control, Type 65, includes a Teletype Model KSR-28 Printer and Keyboard with an allowable input or printing rate of ten characters per second. Typed information may be monitored by a program. A program may print information.

The Visual CRT Display, Type 30A or 30D, displays data on a 9 1/4" by 9 1/4" area. Information is plotted on a point by point basis to form either graphical or tabular data. Operation of this device requires the Real-Time Option.

The Light Pen, Type 32, is a photoelectric device which detects information displayed on the Type 30 Visual CRT Display. The Light Pen may be used to draw functions, in effect, on the CRT by monitoring displayed information. Requires Real-Time Option.

The 18-bit Relay Buffer provides contacts which operate devices of higher power rating. The relays have form "D" contacts, which open and close in approximately 3 milliseconds. Requires Real-Time Option.

The Magnetic Tape System Control, Type 54, controls up to four Magnetic Tape Transport Units, Type 50. Information is read from or written on the tape. The format on the tape may be programmed to be compatible with IBM tapes having a density of 200, 6+1 bit characters per inch. Requires Real-Time Option.

The Magnetic Tape Transport Units, Type 50, are used with the Magnetic Tape System Control, Type 54.

The Line Printer and Control, Type 62-4, operates at up to 600 lines per minute, 120 columns per line. Each column may print one of 64 characters. Spacing format is controlled by a punched format tape in the Printer. Once a command to print or space is given, the Internal Processor is not required. Approximately one per cent of program running time is required to operate the Line Printer at a 600 line per minute rate. Requires Real-Time Option.

The Card Reader and Control, Type 41-200-4, operates at a rate of up to 200 cards per minute. Cards are read column by column. Column information may be read in alphanumeric or binary mode. The alphanumeric mode converts the 12-bit Hollerith Code of one column into the six-bit binary-coded decimal code with code validity checking. The binary mode reads a 12-bit column directly into the PDP-4. Approximately one per cent of a Card Reader program running time is required to read the 80 columns of information at the 200 cards per minute rate. Requires Real-Time Option.

The Card Punch Control, Type 40-523-4, enables the operation of a standard IBM Type 523 Summary Punch with PDP-4. Cards are punched on a row by row basis at a rate of 100 cards per minute. Approximately 0.3 per cent of program running time is required to operate the Card Punch at the 100-card-per-minute rate. Requires Real-Time Option.

CHAPTER 2

ARITHMETIC AND CONTROL ELEMENT

In this section the functions of the Arithmetic and Control Element, summarized in the previous section, are described in detail. The instruction codes and operations are then explained and listed.

FUNCTIONS

Internal Processor

The Internal Processor performs arithmetic operations, controls the memory, and handles information entering and leaving the machine. It consists of six registers, shown within the dotted line in Figure 3: the Accumulator, Link, Memory Buffer, Memory Address, Program, and Instruction registers.

Accumulator (AC): an 18-bit register, which together with the Memory Buffer, performs the arithmetic operations. The AC may be cleared, complemented, and rotated right or left together with the Link. The contents of the AC and the contents of the Memory Buffer may be added together, the logical AND formed, and the exclusive OR formed and placed in the AC. The logical OR of the AC and the Accumulator Switches (ACS) on the Operator Console may be placed in the AC. The AC also acts as an input-output buffer register.

Link (L): a one-bit register to extend the facilities of the AC. It may be cleared, complemented, and shifted (as part of the AC). It is used as an overflow flip-flop for 1's complement arithmetic and as a carry register for 2's complement arithmetic. It functions as a program flag, an overflow flag, and a carry extension register.

Memory Buffer Register (MB): an 18-bit register which holds information read from a selected memory address (specified by MB). The reading of information from a cell (the 18-bit word stored at one memory address), clears the contents of the cell. The MB also specifies the information written back into the selected cell. The time required

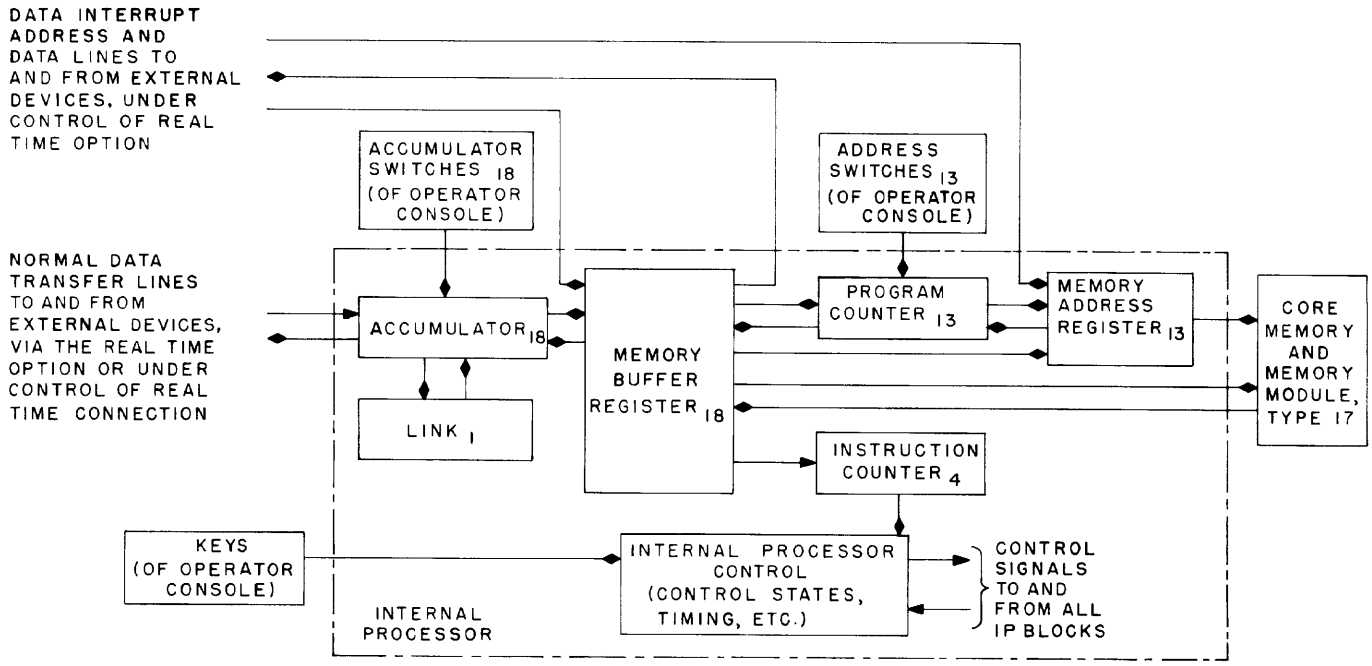


Figure 3 Arithmetic and Control Element

to read information from a cell and rewrite information into the cell (memory cycle time) is 8 microseconds.

The register may be cleared and information read in from the Program Counter, the Accumulator, or external sources for Data Interrupt. The register may be advanced by one.

Memory Address Register (MA): a 13-bit register used to address (or select) a memory cell. The MA may be cleared and information read in from the MB, the Program Counter, or external sources for Data Interrupt. The 13 bits allow addressing of 8192 words of Core Memory.

Program Counter (PC): a 13-bit register which contains the address of the cell in memory from which the next instruction will be taken. The register may be cleared and information read in from the MA, MB, or ADDRESS switches of the Operator Console. The register may be advanced by one.

Instruction Register (IR): a 4-bit register which contains the instruction currently being carried out by the machine. Information is read into the IR from the MB. Instruction word bits 0 through 3 are stored in the IR. These bits determine control to enact the instruction.

Core Memory and Memory Module, Type 17: The Memory stores information being collected or distributed and instructions for the Internal Processor. As described in the previous section, memory capacities of 1024, 4096, and 8192 words are available in PDP-4.

Operator Console: All controls and indications necessary to operate the PDP-4 are located on the Operator Console, shown in Figure 4. The functions of the keys, switches, and indicators on the Operator Console are given below.

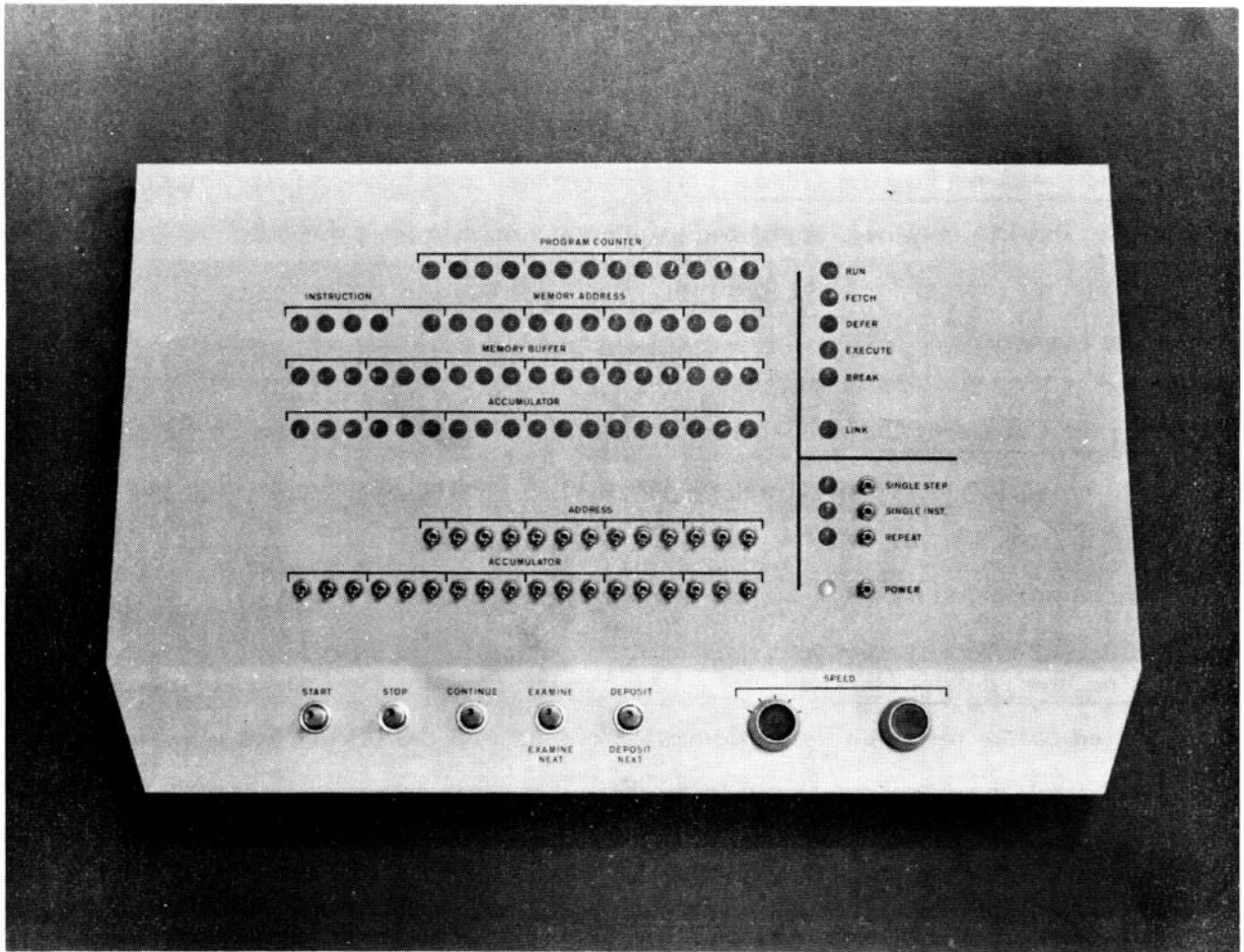


Figure 4 Operator Console

Console Key	Function
START	Starts the computer. The first instruction is taken from the Core Memory at the address indicated by the ADDRESS switches. The START key clears the AC and L, and turns off the Program Interrupt.
STOP	Causes the computer to stop at the completion of the memory cycle in progress at the time of key operation.
CONTINUE	Causes the computer to resume operation, beginning at the address specified by the PC. The key has three positions: off, on and holding, and on with spring return to off.
EXAMINE	Sets the contents of the memory cell selected by the ADDRESS switches into the AC and MB. The MA will contain the address of the memory cell being examined. The PC will contain the address of the next memory cell.
EXAMINE NEXT	Sets the contents of the memory, at the address specified by the PC, into the AC and the MB. The C(PC) (the contents of the PC) are indexed by one. The MA will contain the address of the register examined.
DEPOSIT	Sets the word selected by the ACCUMULATOR switches into the memory at the location specified by the ADDRESS switches. The results will remain in the AC and MB. The MA will contain the address of the memory cell holding the information. The PC will contain the address of the next cell.
DEPOSIT NEXT	Sets the contents of the ACCUMULATOR switches into the memory at the location specified by the PC. The MA will contain the address of the register holding the information.

Console Switch	Function
POWER	Controls the primary power to the computer.
ADDRESS	A group of 13 switches which establish the memory address for START, EXAMINE, and DEPOSIT keys.

Console Switch	Function
ACCUMULATOR	A group of 18 switches which establish the contents of a word to be manually deposited into memory by means of the DEPOSIT or DEPOSIT NEXT key, or to be brought into the AC under program control.
SINGLE STEP	Causes the computer to halt at the completion of each memory cycle. Operation of the CONTINUE key will step the program one cycle at a time.
SINGLE INSTRUCTION	Causes the computer to stop at the completion of each instruction. Operation of the CONTINUE key will step to the next instruction.
REPEAT	Causes operations initiated by a key to be repeated as long as the key is depressed. The operations are performed at the rate set by the SPEED switch and SPEED control.
SPEED	Varies the REPEAT interval from approximately 40 microseconds to 8 seconds.

Console Indicator	Indication
ACCUMULATOR	The contents of the AC
MEMORY BUFFER	The contents of the MB
INSTRUCTION	The binary code of the instruction being executed
MEMORY ADDRESS	The contents of the MA
PROGRAM COUNTER	The contents of the PC
LINK	The contents of the L (one bit)
BREAK, EXECUTE, DEFER, FETCH	The primary control state of the <u>next</u> memory cycle
RUN	The computer is executing instructions

CONTROL STATES

The PDP-4 operates in one of four primary control states during a core memory cycle: Fetch, Execute, Control, or Break. The instruction establishes the control state. A decision for the next state or cycle is made at the completion of each state.

Fetch (F): brings a new instruction into the MB from memory. It is initiated before the beginning of each new instruction. The instruction is taken from the location specified by the PC. The instruction part of the word (bits 0 through 3) is then set into the IR and the PC is advanced by one.

If a two-cycle instruction is fetched, the next state will be either a Defer or an Execute. If a one-cycle instruction is fetched, the instruction will be enacted and the next cycle or state will be another fetch.

Execute (E): enacts the instruction as the last cycle of an instruction. Any instruction which contains a memory address as part of the word, such as add C(Y) to C(AC), draws the contents of the indicated memory address into the MB and performs the indicated operation during the execute state.

Defer (D): obtains an effective address from memory during the defer cycle. This state occurs when a 1 is in bit 4 of a memory reference instruction. When deferring is specified, the contents of the deferred address cell are used to form the effective address. The instruction portion, and bit 4, of the deferred cell is ignored when obtaining the effective address.

Break (B): breaks the sequence of the main program for Data Interrupt or a Program Interrupt. The Data Interrupt breaks the program only at the completion of an instruction, and allows information to be transferred between memory and an external (IO) device.

The Program Interrupt breaks the program only at the completion of an instruction, to enter a subroutine in register 0. The Program Interrupt stores the C(PC) and the Link in location 0000 and enters the routine at location 0001.

INSTRUCTION OPERATION

All instructions in the machine utilize bits 0 through 3 to define the instruction code and bits 5 through 17 indicate the Core Memory address of the operand or of operations performed which do not reference the memory. The instructions may be divided into memory reference, which require an operand from memory, and augmented classes, which do not require an operand.

MEMORY REFERENCE INSTRUCTIONS AND AUTO INDEXING

Memory reference instructions employ bit 4 as the indirect address bit. The bit allotment is indicated in figure 5.

A memory reference instruction which is to use an indirect address will have a 1 in bit 4 of the instruction word. The original address, Y , of the instruction will not be used to locate the operand of the instruction, as is the normal case. Instead it is used to locate a memory register whose contents in bits 5 through 17 will be used as the address. Thus Y is not the location of the operand but the location of the location of the operand. Bit 4 is ignored during the indirect or defer cycle. If the memory register containing the indirect address is registers $10_8 - 17_8$, a 1 is added to the contents of the register before the indirect addressing occurs. The indirect reference to registers $10_8 - 17_8$ is known as auto-indexing.

In the list of memory reference instructions which follows, attention is called to the instructions add Y and tad Y , which initiate 1's complement and 2's complement addition, respectively.

In 1's complement arithmetic, negative numbers are represented by a 1 in the sign position (bit 0 or leftmost bit) of the word. Each digit of the word decreases in significance from bit 1 to bit 17. The negative of a 1's complement number is formed by complementing the number. The complement of a binary number is formed by changing all ones to zeros and all zeros to ones. In 1's complement convention, -0 (all ONES) results when adding -0 to 0 or adding $-n$ to n .

In 2's complement arithmetic, a negative number is represented by a 1 in the sign position. The negative of a 2's complement number is formed by complementing it and adding 1 to the result.

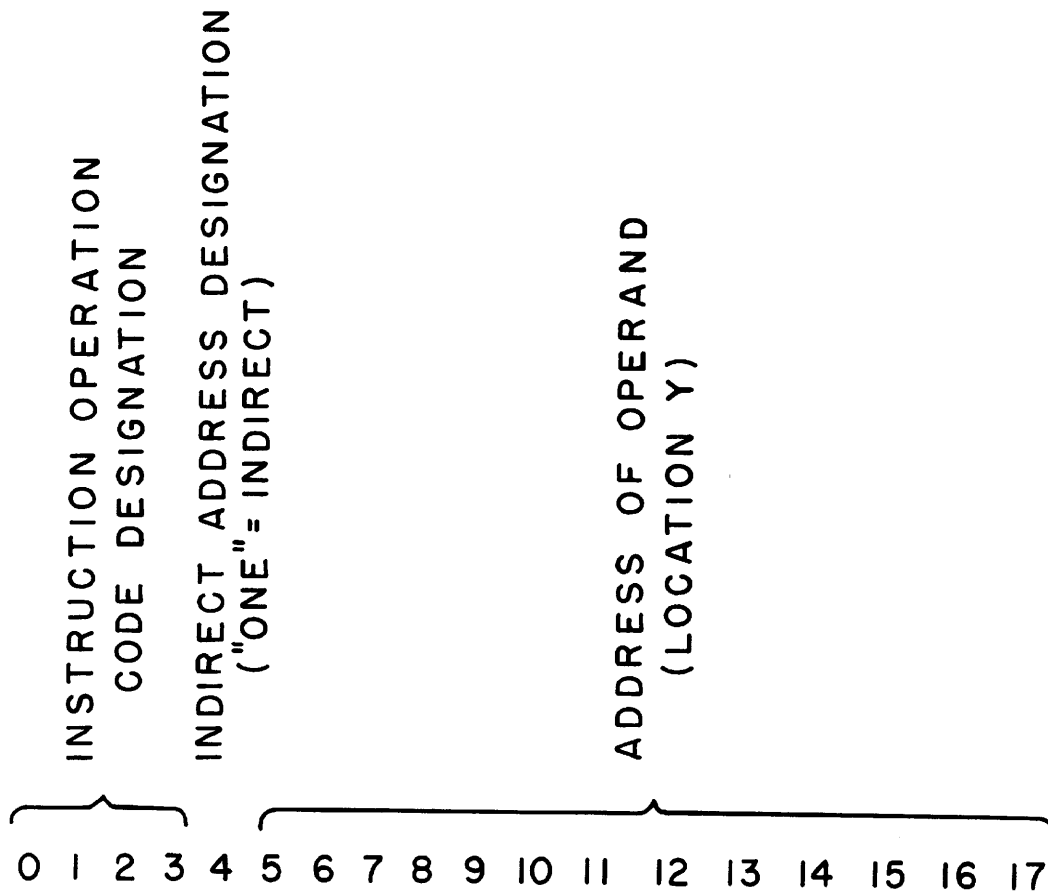


Figure 5 Memory Reference Instruction Format

MEMORY REFERENCE INSTRUCTIONS

Mnemonic Code	Octal Code	Time (μ sec)	Operation
cal Y	00	16	<p>Call subroutine. The address portion of the instruction, Y, is ignored. If the indirect bit = 0, the instruction takes the action jms 20. If the indirect bit = 1, the action is jms i 20.</p> <p>If $MB_4 = 0$, then $C(L) \Rightarrow C(20_0)$, $0 \Rightarrow C(20_{1-4})$, $C(PC) \Rightarrow C(20_{5-17})$. $21 \Rightarrow C(PC)$.</p> <p>If $MB_4 = 1$, then $C(L) \Rightarrow C(X_0)$, $0 \Rightarrow C(X_{1-4})$, $C(PC) \Rightarrow C(X_{5-17})$. $X + 1 \Rightarrow C(PC)$. $X = C(20_{5-17})$</p>
dac Y	04	16	<p>Deposit AC. The C(AC) are deposited in memory register Y. The C(AC) are unaffected by this operation. $C(AC) \Rightarrow C(Y)$.</p>
jms Y	10	16	<p>Jump to subroutine. The C(PC) and Link are deposited in memory location Y. The next instruction will be taken from Y + 1, the beginning of the subroutine. $C(L) \Rightarrow C(Y_0)$, $0 \Rightarrow C(Y_{1-4})$, $C(PC) \Rightarrow C(Y)$, $Y+1 \Rightarrow C(PC)$.</p>
dzm Y	14	16	<p>Deposit 0 in memory. The C(Y) are changed to 0. The original C(Y) are lost. $0 \Rightarrow C(Y)$.</p>
lac Y	20	16	<p>Load AC. The C(Y) replace the C(AC). The previous C(AC) are lost. $C(Y) \Rightarrow C(AC)$.</p>
xor Y	24	16	<p>Exclusive OR. The exclusive OR logical function is performed on a bit-by-bit basis between the C(AC) and C(Y). The result is left in the AC and the original C(AC) are lost. $C(AC)_i \oplus C(Y)_i \Rightarrow C(AC)_i$.</p>

C(AC) _i original	Example	
C(AC) _i original	C(Y) _i	C(AC) _i final
0	0	0
0	1	1
1	0	1
1	1	0

Mnemonic Code	Octal Code	Time (μsec)	Operation																		
add Y	30	16	1's complement add. The C(Y) are added to the C(AC) in 1's complement arithmetic. The result is left in the AC and the original C(AC) are lost. The Link bit is set to a one if the magnitude of the sum of C(Y) and C(AC) is greater than $2^{17}-1$. $C(AC) + C(Y) \Rightarrow C(AC)$																		
tad Y	34	16	2's complement add. The C(Y) are added to C(AC) in 2's complement arithmetic. If there is a carry out of bit 0 of the AC, the Link will be set to 1. This feature is useful in multiple precision arithmetic. $C(AC) + C(Y) \Rightarrow C(AC)$																		
xct Y	40	8+ time of instruction being executed	Execute. The instruction in register Y will be executed. The computer will act as if the instruction located in Y were in the place of the xct Y.																		
isz Y	44	16	Index and skip if 0. The C(Y) are replaced by C(Y) + 1. The C(AC) are unaffected by this instruction. The addition is done using 2's complement arithmetic. If the resulting sum is 0, the instruction following the isz is skipped. $C(Y) + 1 \Rightarrow C(Y)$, if $C(Y) + 1 = 0$, then $C(PC) + 1 \Rightarrow C(PC)$																		
and Y	50	16	Logical AND. The logical AND function is performed on a bit-by-bit basis between C(AC) and C(Y). The original C(AC) are lost. $C(Y)_i \wedge C(AC)_i \Rightarrow C(AC)_i$																		
<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="3">Example</th> </tr> <tr> <th>$C(AC)_i$ original</th> <th>$C(Y)_i$</th> <th>$C(AC)_i$ final</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>				Example			$C(AC)_i$ original	$C(Y)_i$	$C(AC)_i$ final	0	0	0	0	1	0	1	0	0	1	1	1
Example																					
$C(AC)_i$ original	$C(Y)_i$	$C(AC)_i$ final																			
0	0	0																			
0	1	0																			
1	0	0																			
1	1	1																			
sad Y	54	16	Skip if AC is different than Y. The C(Y) are compared with the C(AC). If the two numbers are different, the next instruction in the sequence is skipped. The C(AC) and C(Y) are both unaffected by the instruction.																		

Mnemonic Code	Octal Code	Time (μ sec)	Operation
			If $C(AC) \neq C(Y)$ then $C(PC) + 1 \Rightarrow C(PC)$.
jmp Y	60	8	Jump. The $C(PC)$ are reset to address Y. The next instruction to be executed is taken from memory register Y. The original contents of the PC are lost. $Y \Rightarrow C(PC)$

Augmented Instructions

Augmented instructions use bits 4 through 17 to specify combinations of microcommands. There are three augmented instructions:

- a. Operate instructions perform operations on the AC and Link, and allow a skip to take place as a function of the AC and Link.
- b. Load AC with a word allows the command to specify a constant with which to load the AC.
- c. Input-output transfer which pulses external (peripheral) equipment to initiate an information transfer. Input-output instructions are explained in detail in Section 3.

Since augmented instructions are microprogramming instruction, actions are specified by bits within the address portion of the instruction. Several actions may be called for in the same instruction word.

Operate Instruction

The operate (opr) augmented instruction is used for branching or skipping, modifying the contents of the AC, and rotating the contents of the AC. This is a single-cycle instruction which is enacted in 8 microseconds and is initiated by an instruction code of 74_8 . Bit allotment of this instruction is indicated in Figure 6 and the basic operate microinstructions are listed below. Bits of the instructions can be combined to form instructions which perform various operations in sequence.

<p>The Operate Group (Operation Code 11110)</p>	
0	
1	
2	
3	
4	
5	CLA Instruction If A "One"
6	CLL Instruction If A "One"
7	A Second Rotate Instruction
8	Do Not Skip On Condition If A "One"
9	Skip If Link Non Zero, SNL, If A "One"
10	Skip If AC=0, SZA, If A "One"
11	Skip If AC ₀ = 1, SMA, If A "One"
12	HLT Instruction If A "One"
13	RAR Instruction If A "One"
14	RAL Instruction If A "One"
15	OAS Instruction If A "One"
16	CML Instruction If A "One"
17	CMA Instruction If A "One"

Figure 6 Operate augmented instruction format

BASIC OPERATE INSTRUCTION GROUP

opr 740000

Bit 4 = 0

Mnemonic Code	Octal Code of Address Part	Operation	Sequence
sma	100	Skip if the AC is minus. If $AC_0 = 1$, then $C(PC) + 1 \Rightarrow C(PC)$.	0
sza	200	Skip if the AC is 0 (+0). If AC_{0-17} bits are all 0, then $C(PC) + 1 \Rightarrow C(PC)$.	0
snl	400	Skip on non-zero link. If $L = 1$, then $C(PC) + 1 \Rightarrow C(PC)$.	0
rcs	1000	Reverse the condition for a skip; i.e., do not skip if any of the above skip conditions are present. Bits 9-11 are microprogrammed, and allow the logical expressions to be formed; e.g., $sma \vee sza \vee snl$ in various combinations. For a list of additional skip instructions see the following pages.	0
	2000	Allow a second rotate to take place at event time 1 if a rar or ral is used. (If this bit is a ONE, and ral or rar are used, no other instruction affecting AC may be given; namely, cma, cml, oas, cla, cll.)	1
cll	4000	Clear Link, $0 \Rightarrow C(L)$	1
cla	10,000	Clear AC, $0 \Rightarrow C(AC)$	1
cma	1	Complement AC, $C(AC) \Rightarrow C(AC)$	2
oas	4	"Inclusive OR" AC switches with AC. $C(ACS) \vee C(AC) \Rightarrow C(AC)$	2
ral	10	Rotate AC and Link left one place. $C(AC_i) \Rightarrow C(AC_{i-1})$ $C(AC_0) \Rightarrow C(L)$ $C(L) \Rightarrow C(AC_{17})$	2
rar	20	Rotate AC and L right one place. $C(AC_i) \Rightarrow C(AC_{i+1})$ $C(L) \Rightarrow C(AC_0)$ $C(AC_{17}) \Rightarrow C(L)$	2

Mnemonic Code	Octal Code of Address Part	Operation	Sequence
---------------	----------------------------	-----------	----------

(If ral or rar are given, cma, cml, oas may not be given.)

hlt	40	Halt the machine. 0 => C(RUN)	3
-----	----	-------------------------------	---

The following mnemonic words have been assigned to these microprogrammed instructions in the operate class.

opr	740000	Operate. No effect.
nop	740000	No operation. No effect.
las	750004	Load the AC with the ACCUMULATOR switches. C(AC Switches) => C(AC).
rtl	742010	Rotate the C(AC) and C(L) left two places. This is identical to giving two ral instructions.
rtr	742020	Rotate the C(AC) and C(L) right two places. Identical to two rar instructions.
stl	744002	Set the Link. (Clear the Link then complement the Link.) $1 \rightarrow C(L)$
clc	741001	Clear the AC, then complement the AC. $-0 \Rightarrow C(AC)$
skp	741000	Skip. $C(PC) + 1 \Rightarrow C(PC)$
spa	741100	Skip if AC is positive. (Do not skip if AC is negative.) If $AC_0 = 0$, then $C(PC) + 1 \Rightarrow C(PC)$
sna	741200	Skip if AC is non-zero. If $AC \neq 0$, then $C(PC) + 1 \Rightarrow C(PC)$
szl	741400	Skip if Link is 0. If $L = 0$, then $C(PC) + 1 \Rightarrow C(PC)$

The Load-Accumulator-With-Word Instruction

law N	760000	Load AC with address portion, N, of the instruction. Place ones in remaining bits of AC. The indirect bit specifies law if a 1 or opr if a 0. $1 \Rightarrow C(AC_{0-4}), N \Rightarrow C(AC_{5-17})$
-------	--------	--

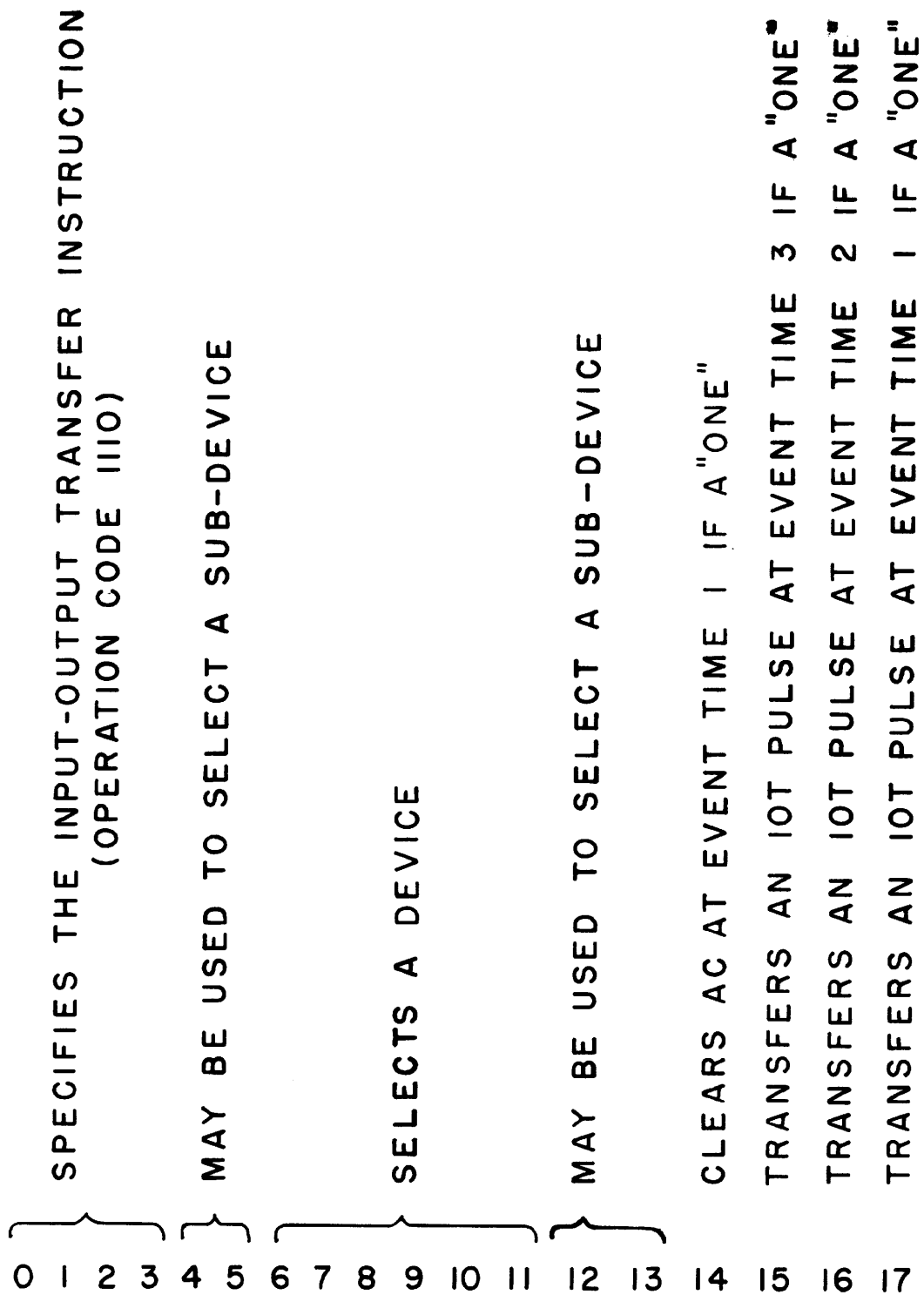


Figure 7 Input-Output Transfer, Augmented Instruction Format

The Input-Output -Transfer Instruction

The input-output transfer (iot) augmented instruction causes the Interface to produce iot pulses which select the IO devices and transfer information. This is a single-cycle instruction which is enacted in 8 microseconds and is initiated by an instruction code of 70_8 . Bit allotment of this instruction is indicated in Figure 7.

Operations caused by this instruction occur at three event times which are related to the internal timing of the Arithmetic and Control element. Event times 1 and 2 occur near the end of the cycle, and event time 3 occurs at the beginning of the next instructions. This timing allows one iot instruction to perform multiple operations.

Before information can be transferred into the AC from an IO device, the AC must be cleared. Clearing of the AC, when necessary, during an iot instruction is programmed to occur at event time 1 by placing a 1 in bit 14. Use of the iot pulses to establish states or transfer data in IO devices is discussed in Chapter 3.

CHAPTER 3

INPUT-OUTPUT EQUIPMENT FUNCTIONS AND PROGRAMMING

PDP-4 is capable of operating with the ten input-output devices described in Section 1 and with a variety of others, the maximum number depending on their data rates. The computer can operate with most of the devices simultaneously. The Interface, consisting of the Real-Time Connection or the Real Time Option, issues commands to the devices, monitors their state of availability, transfers information to them, and receives information from them. Since the Internal Processor can store or read out data much faster than the devices can operate, the Interface and the individual devices provide buffering to minimize the amount of program time consumed in transfers.

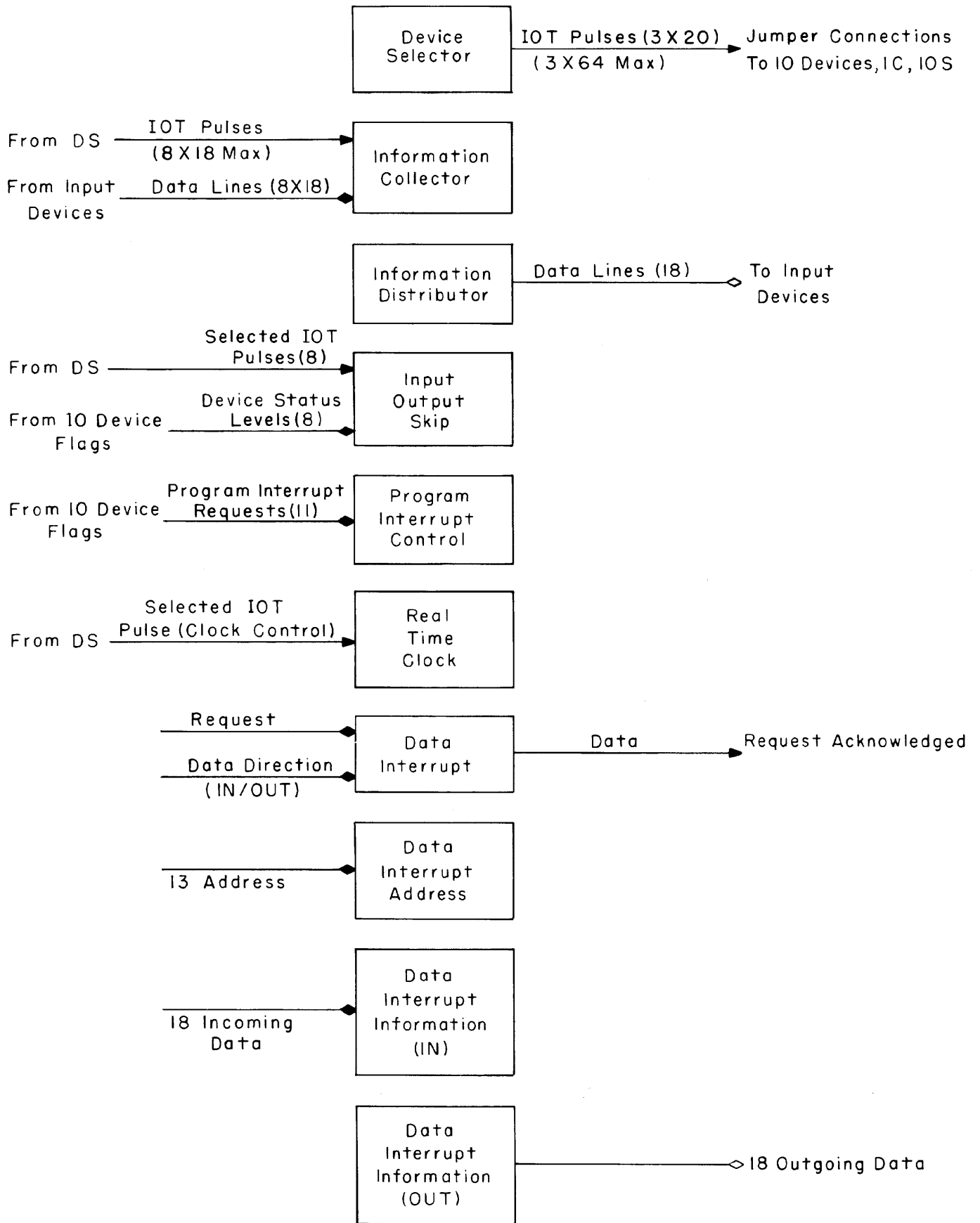
The Real-Time Connection, furnished as standard equipment, provides communication between the Internal Processor and the perforated-tape reader, the perforated-tape punch, and the keyboard-printer. The Real-Time Option, Type 25, gives the system the additional capability to operate efficiently over a wide range of information handling rates, from seconds per event to 125,000 words per second, and with a large variety of input-output devices. The Real-Time Option consists of the Device Selector, the Information Collector, the Information Distributor, the Input-Output Skip Facility, the Program Interrupt Control, the Data Interrupt Control, and the Clock/Timer.

The coupling of input-output equipment to PDP-4 is similar for all devices. The electrical characteristics of the coupling are discussed in Chapter 4. The logical functions and programming instructions are given below.

INPUT-OUTPUT COMMANDS

Device Selector

The input-output transfer (iot) augmented instruction causes the Interface to produce pulses which select IO devices and transfer information. Upon receipt of an instruction, the Device Selector in the Interface performs one of the following functions:



- (a) Starts a device (eg. asks for a line of perforated tape to be read and assembled into a word, or a card to be moved to a reading or punching station, etc.)
- (b) Transfers data from the information buffer of an input device to the AC, through the Information Collector
- (c) Transfers information from the AC, through the Information Distributor (ID) to the buffer of an output device
- (d) Senses the flag(s) associated with a device to determine its availability
- (e) Resets the flags. These commands dismiss a device without asking for additional action.

The flags referred to above are binary signals generated by an external device upon completion of its assigned task. This technique allows the internal processor to resume its arithmetic operations after issuing an instruction to a relatively slow input-output device (data rate of less than 20,000 words per second). When a flag is set to 1 by the device, it signifies that:

- (a) an output action (punch out, etc.) has occurred; the Arithmetic and Control Element may transmit data to the device.
- (b) an input action (card or tape input, etc.) has occurred; information is available for the Arithmetic and Control Element
- (c) an alarm condition exists

Flags may be sensed, and a program skip take place, using the Output Skip Facility. Flags may be read into the AC using the iors instruction. Most flags are connected to the Program Interrupt (see below).

The Device Selector selects an input-output device according to the address code of the device in bits 6-11 in the iot instruction. It then generates IO pulses at event times 1, 2, and 3 if the appropriate micro-instruction code bits are present in bits 17, 16, and 15. Pulse iot 1 occurs near the end of an iot instruction, followed by iot 2 in 2 microseconds. iot 3 occurs at the beginning of the next instruction, 1.2 microseconds after iot 2. This timing enables one iot instruction to perform multiple operations.

Information Collector

The Information Collector enables information to be collected from eight 18-bit word input devices. The AC must contain ZERO at the time the inputs are sampled. A word can be

broken into smaller words according to the word size requirements of the input device. The program steps for reading the contents of a group of static parallel data bits are:

cla	Clear the AC (AC must equal zero)
iot	Selected device (sample the selected device outputs)
dac Y	Deposit C(AC). The C(AC) are sent to a particular memory cell, Y. (the first two steps may be microprogrammed together in one instruction)

Information Distributor

The Information Distributor presents the static data contained in the AC to each output device requiring AC information. The devices sample the Information Distributor using the program-controlled pulses from the Device Selector. The program steps for transmitting information from a particular memory cell are:

lac Y	Load the AC with C(Y)
iot clear selected output register	Prepare for information
iot transmit	The information is sampled and placed in the register of the input-output device (the second two steps may be combined in one instruction)

Input-Output Skip Facility

The Input-Output Skip facility enables the program to skip (or branch) according to various external device (or IO) states. There are eight inputs to the Skip facility. The iot pulses from the Device Selector strobe an input line and if a logic condition is present, the instruction following the iot is skipped. The iot skip pulse must occur at event time 1.

Program Interrupt Control

The program interrupt allows a logic line state to interrupt the program. It is used to speed the processing of input-output device information, or to allow certain alarm conditions to be sensed by the computer. The interrupt may be enabled or disabled by the program.

When the interrupt occurs, the contents of the Program Counter and the Link are stored in memory location 0 (bits 0, 5 . . . 17) and an interrupt program begins in memory location 1.

This action disables the interrupt mode. The interrupt program is responsible for finding the signal causing the interruption, for removing the condition, and for returning to the original program.

When the condition for interruption is removed, an iot signal to re-enable the program interrupt is given, followed by the instruction, `jmp indirect 0`, or `620000`. The interrupted program will thus resume. If a program interrupt request is waiting, it will be serviced after the `620000` instruction.

If a second interruption condition occurs and the interrupt program is running, the signal will have no effect. That is, there is only one level of interruption. The start key disables the program interrupt system. The iot instructions for the program interrupt are:

`iof - 700002 - Disable the program interrupt`

`ion - 700042 - Enable the program interrupt`

The state of the program interrupt may be examined using the `iors` instruction, `700314`.

If the program interrupt is on, a 1 is read into bit 0 of the AC. The other bits of the AC contain the status of other devices.

Clock/Timer

The Clock produces a pulse every $1/60$ second (16.6 milliseconds) which temporarily interrupts the program (in the same manner as the data interrupt) and a 1 is added to the contents of memory cell 7 using 2's complement addition. If the content of memory cell 7 is 0 after the addition, the Clock Flag is set to 1, which initiates a program interrupt. Depressing the START key on the Operator Console clears the Clock Flag and disables the Clock. The iot instructions associated with the Clock are:

`csf - 700001 - Skip the next instruction if the Clock Flag is a 1`

`cof - 700004 - Disable the Clock and clear the Clock Flag`

`con - 700044 - Enable the Clock and clear the Clock Flag`

Register 7 is identical to other core memory registers, that is, its contents may be examined or modified. By presetting register 7 to a number, a program interrupt will occur when the register overflows or after a timed interval.

The status of the Clock may be examined using the `iors` instruction, `700314`. If bit 6 is a 1, a Clock overflow program interrupt has occurred. If bit 7 is a 1, Clock counting is enabled.

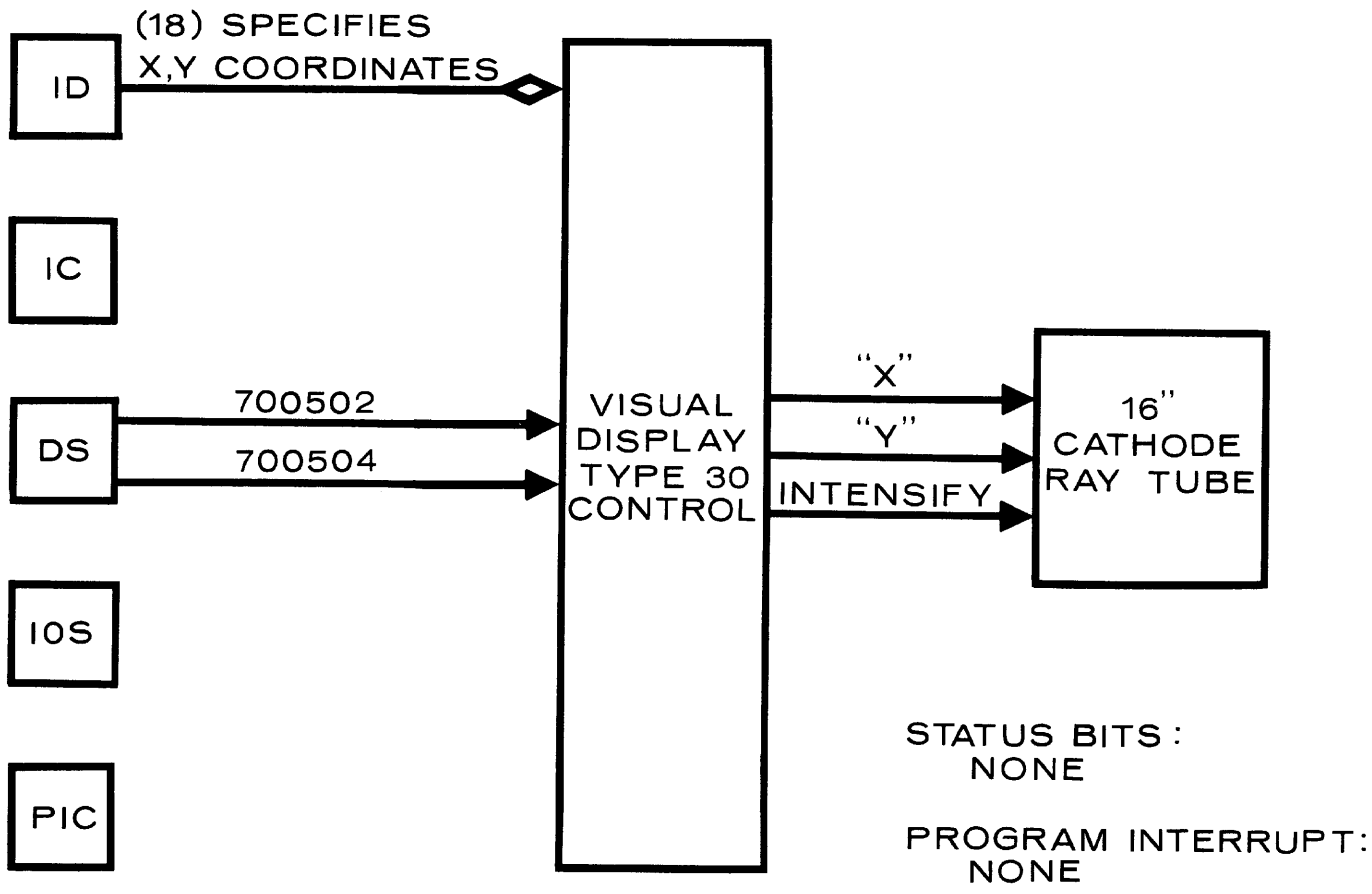


Figure 8 Visual CRT Display programming logic

INPUT-OUTPUT DEVICES

All of the Input-Output Devices discussed below can be controlled by the Real-Time Option, Type 25. The Real-Time Connection, furnished as standard equipment, provides communication between the internal processor and the perforated tape reader, the perforated tape punch, and the keyboard-printer. All devices except the perforated tape reader are optional. This section is arranged in the order of increasing complexity of connection.

Visual CRT Display, Type 30A

Data points are displayed on a 9 1/4 inch by 9 1/4 inch area. Information is plotted on a point by point basis to form either graphical or tabular data. Two digital-to-analog converters drive the deflection yokes in the X and Y directions. Data can be plotted at a 20 kc rate.

The program loads the AC with a point to be plotted. Bits 0 through 8 specify the X co-ordinate of the point and Bits 9 through 17 the Y co-ordinate. The C(AC) are then transferred to the Display Buffer. The specifying of the point initiates the plotting of the point on the CRT.

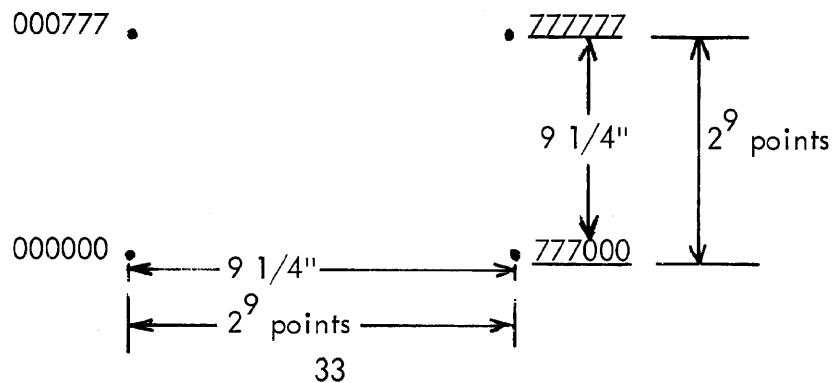
The CRT, Type 30A is selected when the numbers 0 and 5 (octal) are specified in bits 8 and 9 respectively, of the iot instruction. The display commands are:

dls - 7000506 - Load the Display Buffer and select the display. The program loads the Display Buffer from the AC. A point is plotted as specified by the C(Display Buffer). The plotting requires 50 microseconds, after which another dls can be given. The Light Pen Flag or Display Flag is cleared with dls.

700502 - Clear the X and Y display buffers. 0 => C(Display Buffer).

700504 - C(AC) V C(Display Buffer) => C(Display Buffer). Plot the point specified by the C(Display Buffer).

The points specified in the AC are plotted as unsigned quantities, beginning in the lower left hand corner of the cathode ray tube. The point locations are:



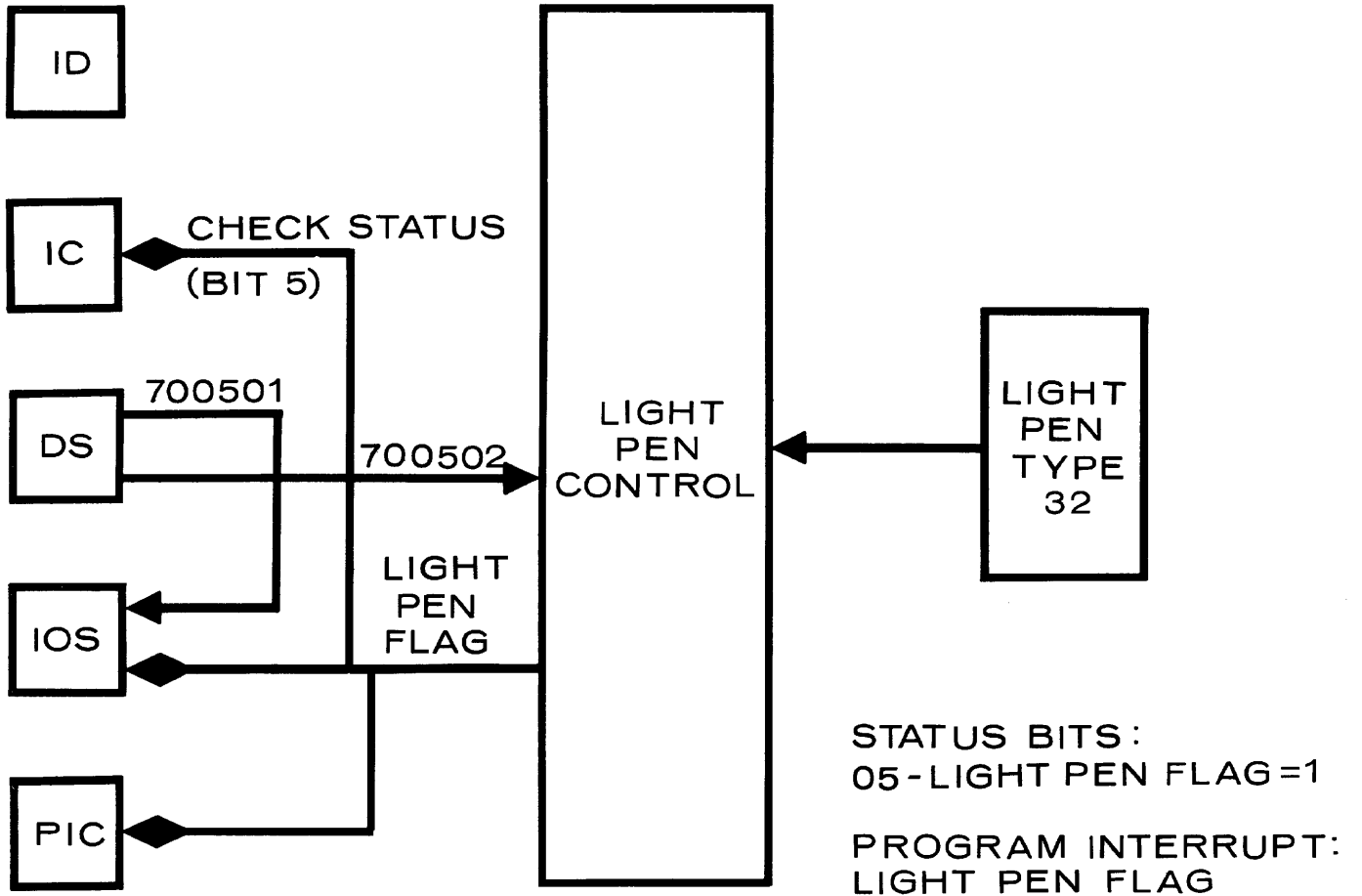


Figure 9 Light Pen programming logic

A program sequence is given in PDP-4 MACRO language below. The program begins in register 40, and plots a point, xy , as specified by Core Memory register 10.

PROGRAM SEQUENCE		
/display a point 30a		
10/ ...		/xy bits 0-8 x, bits 9-17 y.
40/ lac 10		/place xy co-ordinate in ac
700506		/display the point, next dls command must wait 50 microsec.

Light Pen, Type 32

The Light Pen is a photosensitive device which detects the presence of information displayed on a CRT. If the Light Pen is held in front of the CRT at a point displayed, the Display Flag will be set to a 1. The Pen is specified by 0 and 5 in bits 8 and 9 of the iot instruction. The commands are:

dsf - 700501 - Skip if Display Flag is a 1.

dcf - 700502 - Reset the Display Flag to a 0.

The Display Flag is connected to bit 5 of the iors instruction, and to the Program Interrupt.

Visual CRT Display, Type 30D and Light Pen, Type 30D

The type 30D display plots points at a 20KC rate. The x and y co-ordinate buffers (XB and YB) are loaded from the 10 bits, AC_{8-17} .

The instructions are:

dsf - 700501 - Skip if the Display Flag is a 1. The Light Pen is connected to the Display Flag.

dcf - 700601 - Clear the Display Flag.

dxi - 700506 - Load the C(XB) with $C(AC_{8-17})$.

dyl - 700606 - Load the C(YB) with $C(AC_{8-17})$.

dxs - 700546 - Load the C(XB) with $C(AC_{8-17})$. Plot the point: C(XB), C(YB).

dys - 700646 - Load the C(YB) with $C(AC_{8-17})$. Plot the point: C(XB), C(YB).

dlb - 700706 - Load the Brightness Register with $AC_{16} - AC_{17}$. The bits of AC specify the brightness of the points displayed. Clear the Display Flag.

700502 - Clear XB.

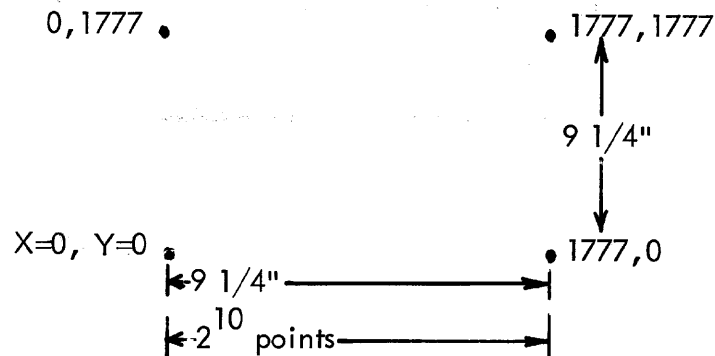
700504 - C(XB) V C(AC) => C(XB). Display a point.

700602 - Clear YB.

700604 - C(YB) V C(AC) => C(YB). Display a point.

The Display Flag is connected to the Program Interrupt and to bit 05 of the iors instruction.

The displayed point locations are:



PROGRAM SEQUENCE

```
/display a point 30d
10/ ..      /x bits 8-17
    ..      /y

40/ lac 10
    dxl      /load x
    lac 11
    dys      /load y and plot the point
```

Analog-to-Digital Converter (Typical Input Device)

An analog-to-digital converter with a resolution of 8 bits and a conversion time of 2 microseconds may be connected to the Real Time Option. The input-output transfer instructions, series 11, for the converter are:

sci- 701115 - Sample the analog input. Convert the sampled quantity to digital form and load the AC with the converted number.

701101 - This microinstruction starts the converter. In a period of 2 microseconds the converter will form an 8-bit number proportional to the analog input.

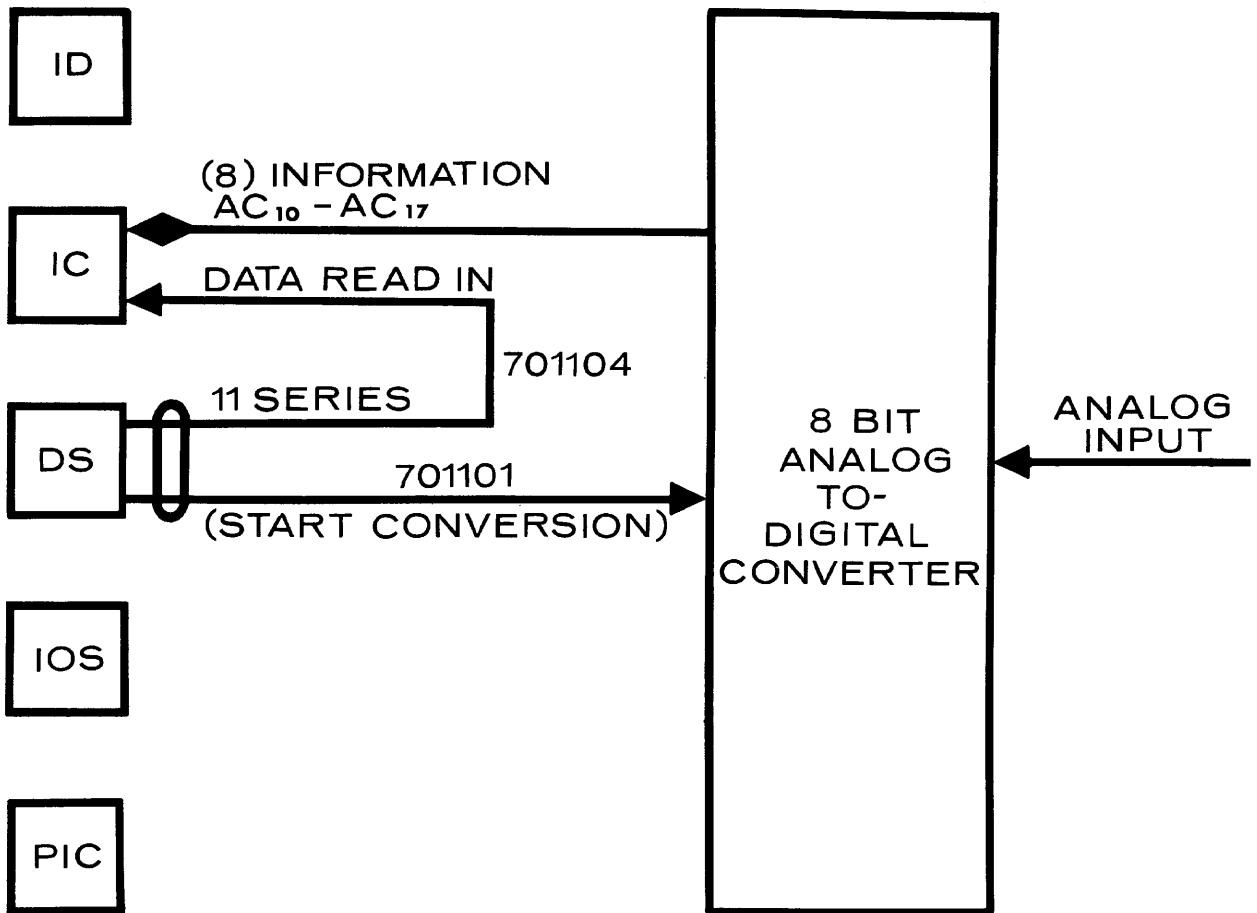


Figure 10 High-speed analog-to-digital converter programming logic

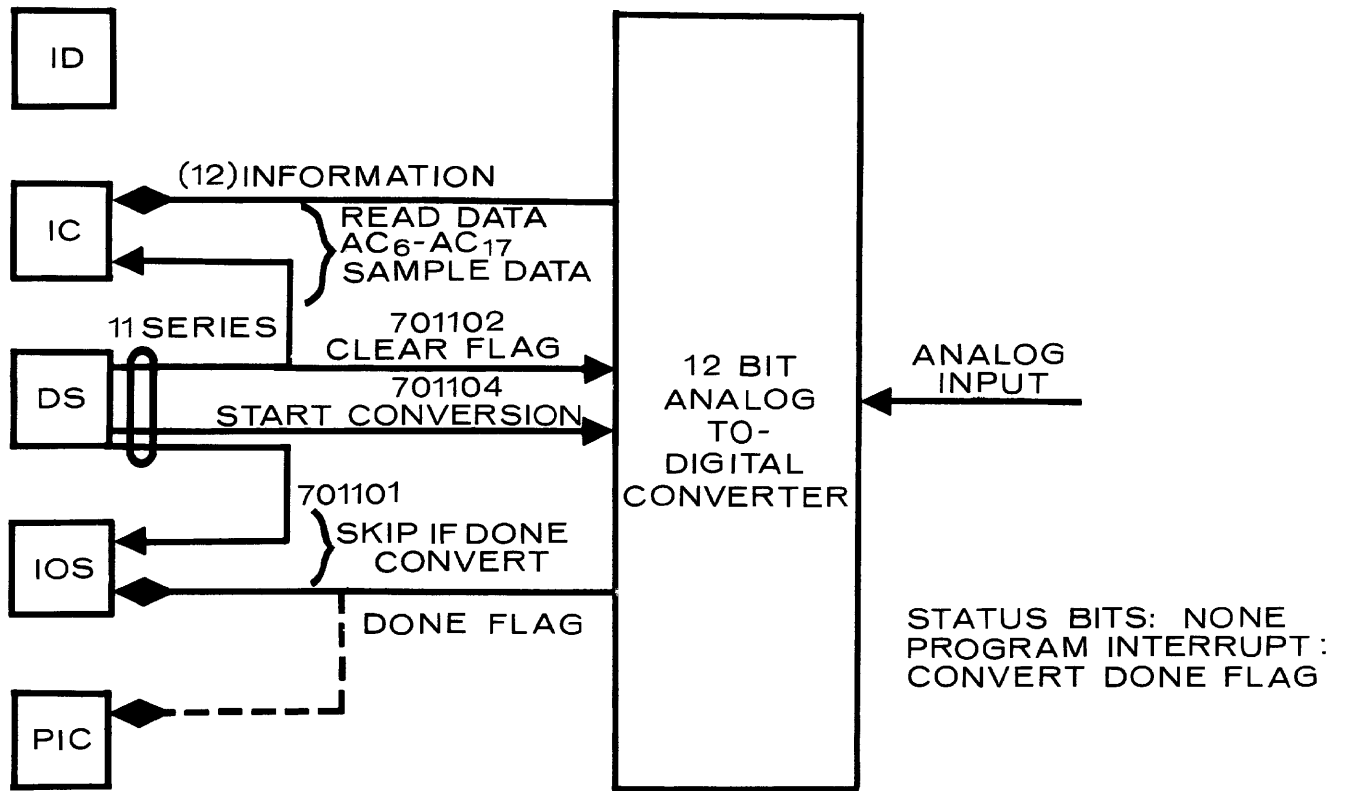


Figure 11 Slow-speed analog-to-digital converter programming logic

701104 - C(A-D Converter) V C(AC) => A(AC)

A program sequence to sample a function at the input to the converter, and store the result in memory register 10 would be:

PROGRAM SEQUENCE	
/analog-to-digital converter	
10/	/location of sampled result
42/ sci	/701115, places sample in AC
dac 10	/deposit result

Low Speed Analog-to-Digital Converter

An analog-to-digital converter with a resolution of 12 bits and a conversion time of 60 microseconds can be connected to PDP-4. The converter is given an iot command to sample the analog function, and in 60 microseconds the converter will contain a 12-bit number proportional to the input. At the completion of the sample, the Converter Flag is set to a 1, signifying that the input data is ready.

The contents of the Converter Buffer are read into the AC with a program command. The action which transfers the information from the converter to the AC also resets the Converter Flag. An iot skip instruction is used which skips if the conversion is complete; i.e., the Converter Flag is a 1. The program instructions, iot series 11, are:

asf - 701101 - Skip if the Converter Flag is a 1.

arb - 701112 - Read Converter Buffer and clear Converter Flag.

ase - 701104 - Start the converter and clear the Converter Flag.

701102 - A microinstruction which clears the Converter Flag, and C(Converter Buffer) V C(AC) => C(AC)

The Converter Flag might connect to the Program Interrupt.

Perforated-Tape Reader and Control

The tape reader senses 5-, 7-, or 8-hole perforated-paper (or Mylar) tape at 300 characters (or lines) per second photoelectrically. The reader control requests reader movement, assembles data from the reader into a Reader Buffer (RB), and signals the computer when incoming data

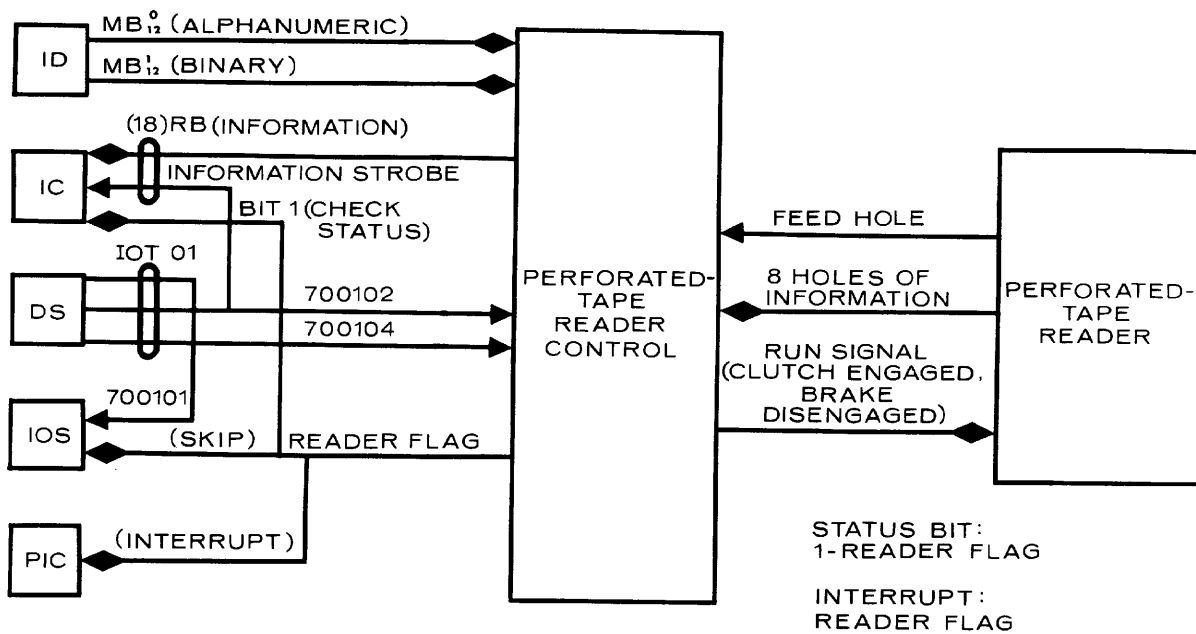


Figure 12 Perforated-Tape Reader programming logic

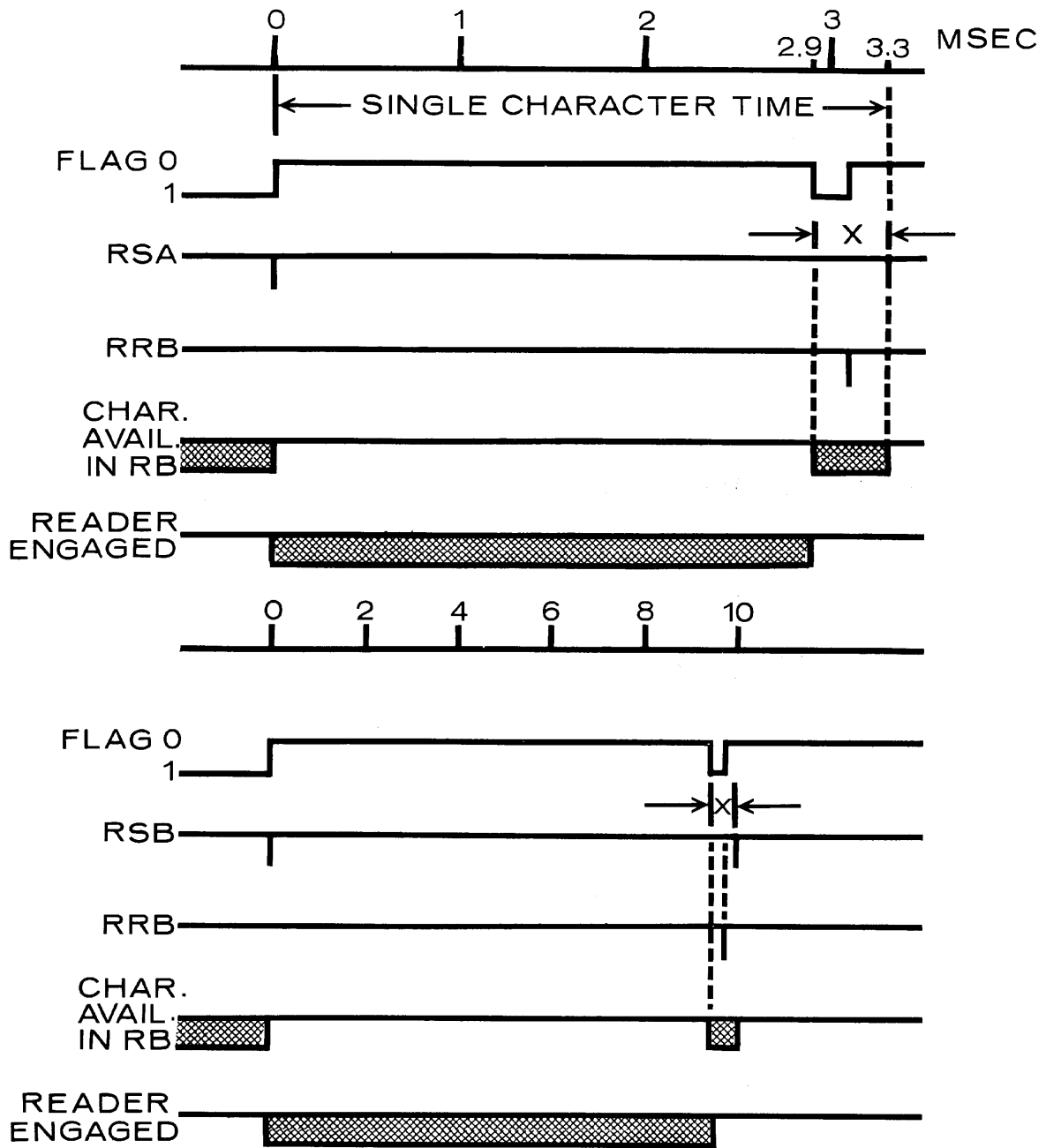


Figure 13 Perforated-Tape Reader timing

is present. Reader tape movement is started by the reader control request to release the reader brake and simultaneously engage the clutch.

In addition to the reader movement control logic, the control unit contains an 18-bit Reader Buffer (RB) which can collect 1 or 3 lines from the tape. The C(RB) can be read into the AC. The Reader Flag becomes a 1 when a character or word has been assembled in RB. A timing diagram of the Reader operation is shown in Figure 13.

An alphanumeric character is one line (5, 7, or 8 holes) on tape. A binary word consists of three consecutive characters (18 bits) on tape which have the 8th hole present. Only 8-hole tape is used in the binary mode; the 7th hole is ignored, and the six remaining bits of each character form one third of the 18-bit word. The reader commands, iot select series 01, are:

rsf - 700101 - Skip if Reader Flag is a 1.

rsa - 700104 - Select reader and fetch one alphanumeric character from tape. Clear the Reader Flag. Reset RB. The character is read into RB bits 10-17. Turn on the Reader Flag when character is present.

rsb - 700144 - Select reader and fetch a binary word from tape. Clear the Reader Flag. Reset the RB. Fetch the next three characters (with 8th holes present) from perforated tape and place in RB bits 0-5, 6-11, and 12-17. Turn on Reader Flag when a word is assembled.

rsf - 700101 - Skip if Reader Flag is a 1.

rrb - 700112 - Read RB. Clear the Reader Flag, and transfer the contents of RB to the AC.

700102 - Clear the Reader Flag. $C(RB) \vee C(AC) \Rightarrow C(AC)$

The Reader Flag is connected to the Program Interrupt Control and to bit 0 of the iors instruction.

Several methods may be used to program the reader. The following sequence reads a character from tape and places it in the AC. Up to 400 microseconds of program may be given between the end of the sequence and the next command to read a character or word from tape. The sequence, starting in register 40 is:

PROGRAM SEQUENCE		
/perforated-tape reader		
40/	700104	/rsa-select reader alphanumeric

700101	/rsf begin loop to look for character arrival
jmp 41	/end loop to look for arrival
700112	/rrb-fetch character from reader buffer

By changing instruction 40 to 7000144, or rsb, the sequence would fetch a binary word.

Printer-Keyboard and Control, Type 65

The printer-keyboard is a Teletype Model 28 (KSR, keyboard send-receive) which can print or receive ten characters per second. A five-bit code, given in Appendix 4, represents the characters. The printing (output) and keyboard (input) functions have separate commands and control logic.

The signals to and from the KSR to the control logic are standard serial, 7.5-unit-code Teletype signals. The signals are: start (1.0 unit), information bits 1-5 (1.0 unit each), and stop (1.5 units). Figure 14 illustrates the current pattern produced by the binary code 10110.

Keyboard

The keyboard control contains a 5-bit buffer (KB) which holds the code for the last key struck. The Keyboard Flag signifies that a character has been typed and its code is present in the Keyboard Buffer. The Keyboard Flag and Keyboard Buffer are cleared each time a character starts to appear on the teletype line. The Keyboard Flag becomes a one, signifying the buffer is full 0.5 ± 0.125 units after the end of information bit 5, or 86.6 milliseconds after key strike time. The instructions to manipulate the Keyboard are:

ksf - 700301 - Skip if the Keyboard Flag is a 1.

krb - 700312 - Read Keyboard Buffer. Clear the Keyboard Flag. $C(KB) \Rightarrow C(AC)$

700302 - Clear the Keyboard Flag. $C(KB) \vee C(AC) \Rightarrow C(AC)$

The Keyboard Flag is connected to the Program Interrupt Control and the iors instruction, bit 03. A timing diagram for the Keyboard is given in Figure 15, and an interconnection diagram is shown in Figure 16.

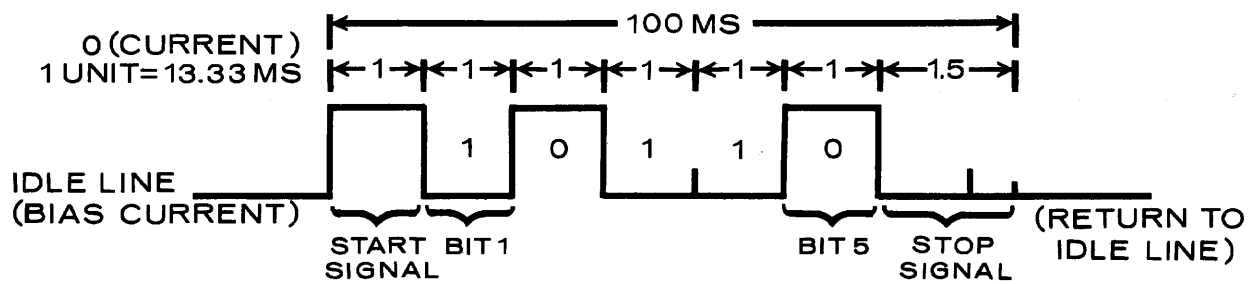


Figure 14 Teletype timing of information code 10110

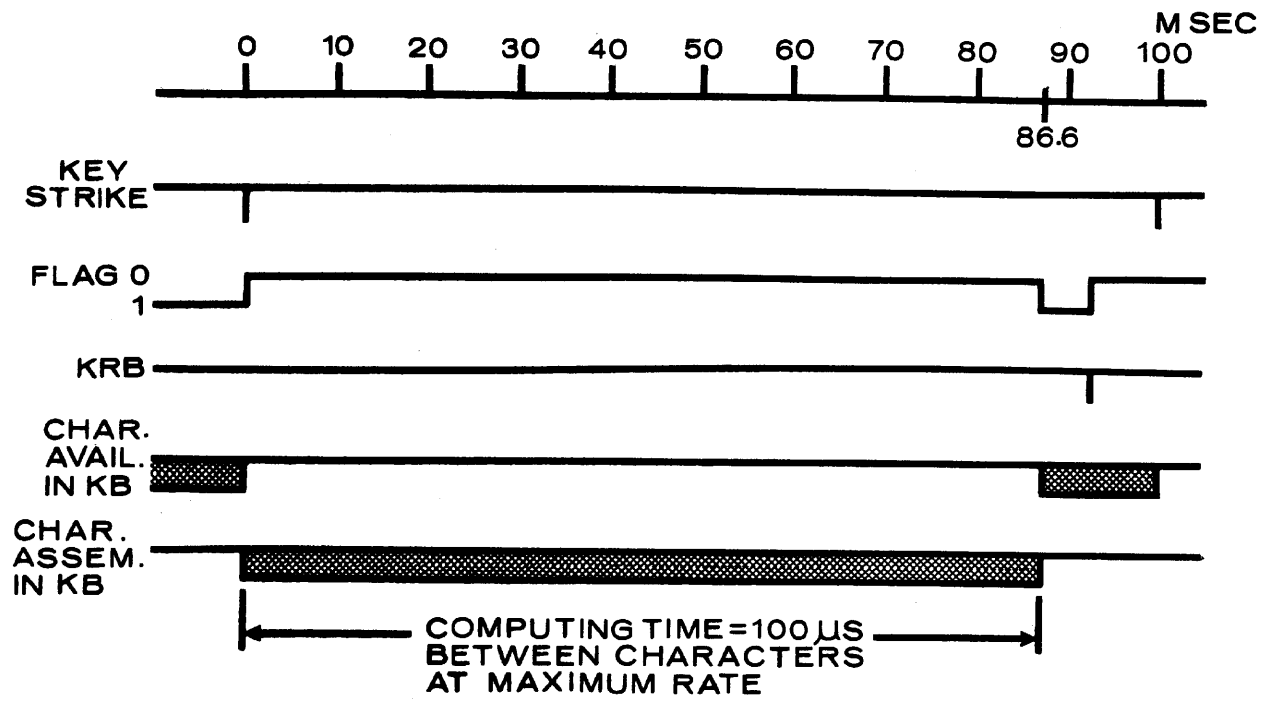


Figure 15 Keyboard timing

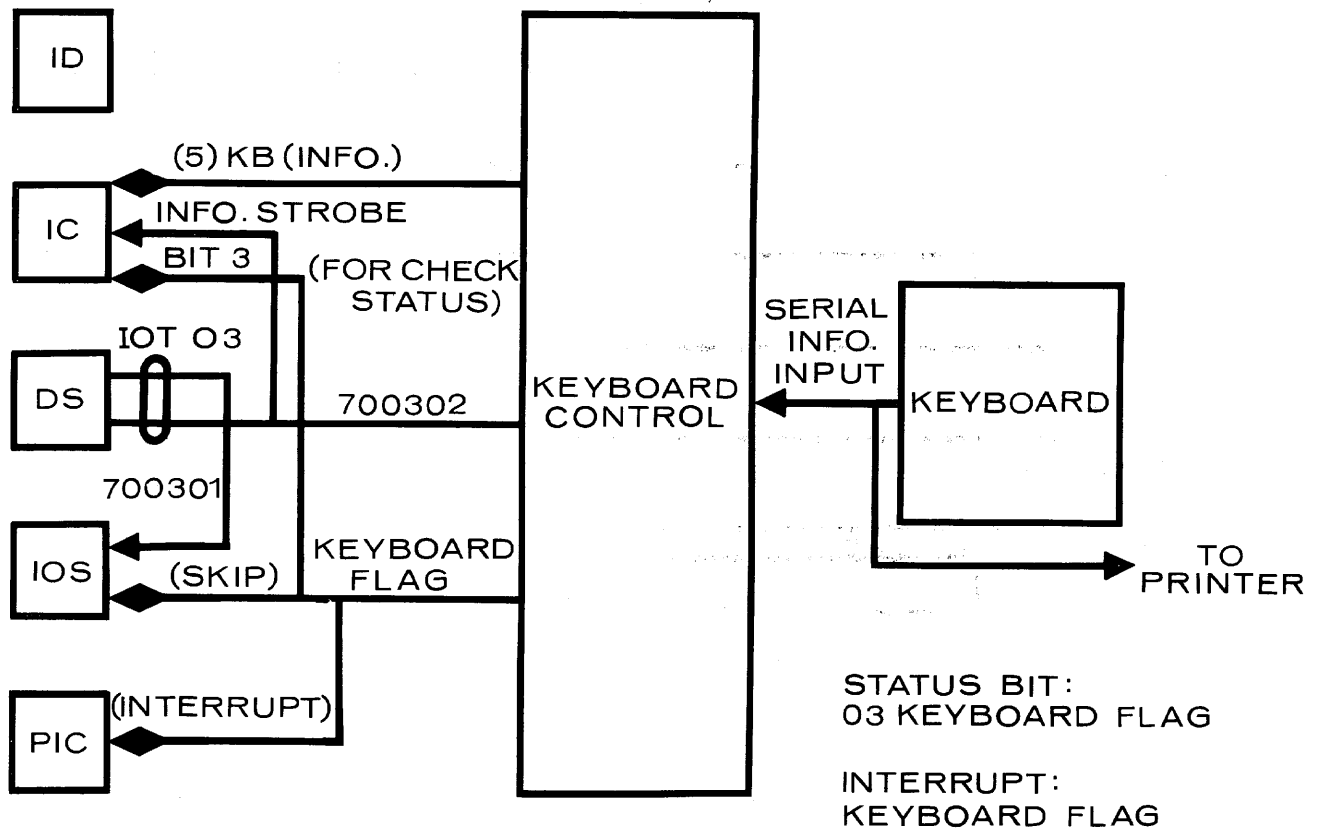


Figure 16 Keyboard programming logic

A simple sequence which "listens" for keyboard inputs is:

PROGRAM SEQUENCE	
/listen	loop for keyboard
400/	700301 /ksf-skip when a character arrives from keyboard
	jmp 400
	krb /700312-read in the character

The sequence following the listen sequence, beginning in 403 may operate for up to 100 + 13.3 milliseconds before returning to listen for the next character without missing the next character. The average computing time between any two characters must be less than 100 milliseconds (for an input rate of 10 characters per second).

Printer (Teleprinter)

The printer is given five bits of information from the AC, coding the character to be printed. The Teleprinter Buffer (TB) receives this information, transmits it to the Teleprinter serially, and when finished turns on the Teleprinter Flag. The Flag is connected to the PI and to bit 04 of the iors instruction. A timing diagram for the Teleprinter is shown in Figure 17 and the interconnection diagram is shown in Figure 18. The printing rate is 10 characters per second. The instructions for the printer are:

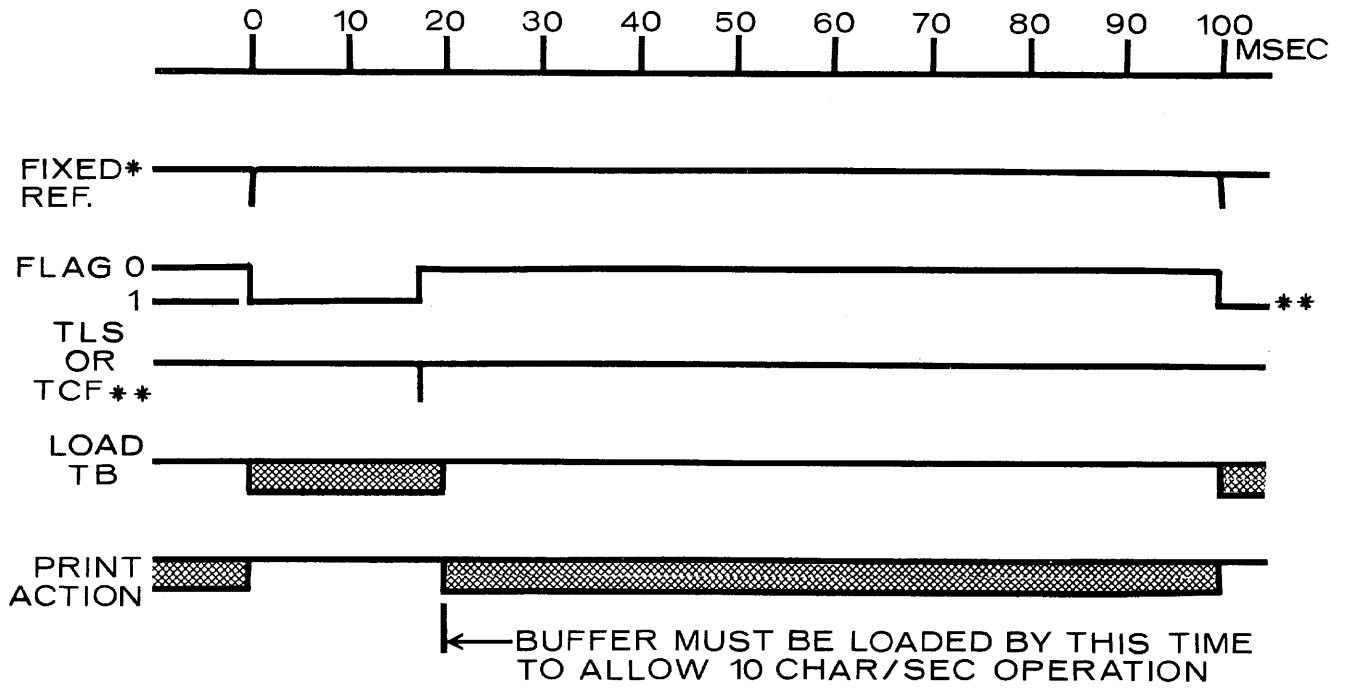
tsf - 700401 - Skip if Teleprinter Flag is a 1.

tls - 700406 - Load the Teleprinter from AC bits 13-17, clear the Teleprinter Flag. Select the Teleprinter for printing.

tcf - 700402 - Clear the Teleprinter Flag.

700404 - C(AC) V C(TB). Print a character.

PROGRAM SEQUENCES	
/print and wait for Teleprinter	
tls	/print the character from AC bits 13-17
tsf	/begin listen loop for printing completion
jmp.-1	/return to previous instruction or listen loop again
:	
/wait for previously printed character completion, then print	
tsf	/wait loop until previous character printed
jmp.-1	/return to wait loop beginning
tls	/print the new character
:	



** IF PCF, FLAG WILL NOT COME ON UNTIL NEXT TLS COMPLETE
 * DETERMINED BY PRINTER

Figure 17 Printer timing

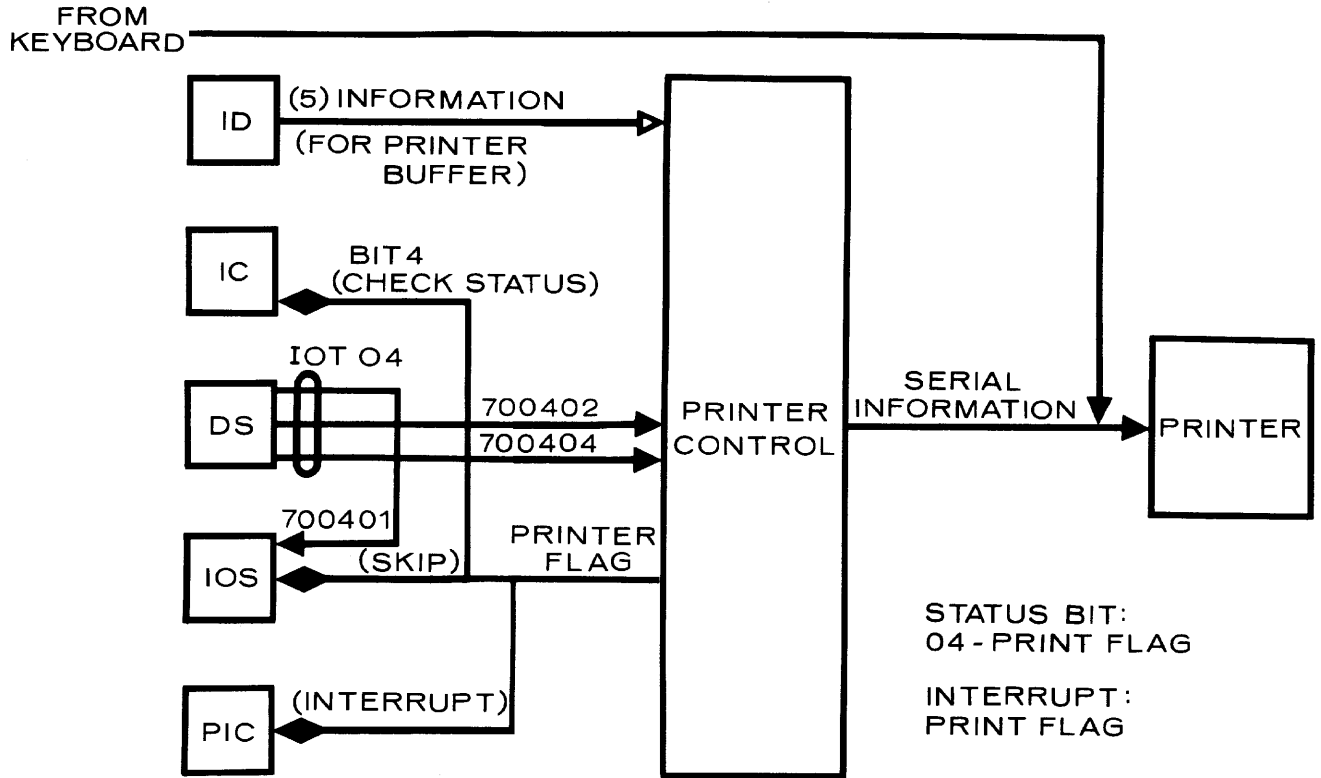


Figure 18 Printer programming logic

In the first sequence above, 20 milliseconds of program time is available between that tfs and the next one that can be given. In the second sequence, 100 milliseconds of program time is available between that tfs and the next one that can be given.

Perforated-Tape Punch and Control, Type 75

The Teletype BRPE paper tape punch perforates 5-, 7-, or 8-hole tape at 63.3 characters (lines) per second.

Information to be punched on a line of tape is loaded on an 8-bit buffer (PB) from the AC bits 10 through 17. The Punch Flag becomes a 1 at the completion of punching action, signalling that new information may be read into PB (and punching initiated). The Punch Flag is connected to the PI, and to the iors instruction bit 02. The timing for the punch is shown in Figure 19. The Real Time Option connections of the punch and control are shown in Figure 20. The Perforated-Tape Punch instructions, iot series 02, are:

psf - 700201 - Skip if the Punch Flag is a 1.

pcf - 700202 - Clear the Punch Flag.

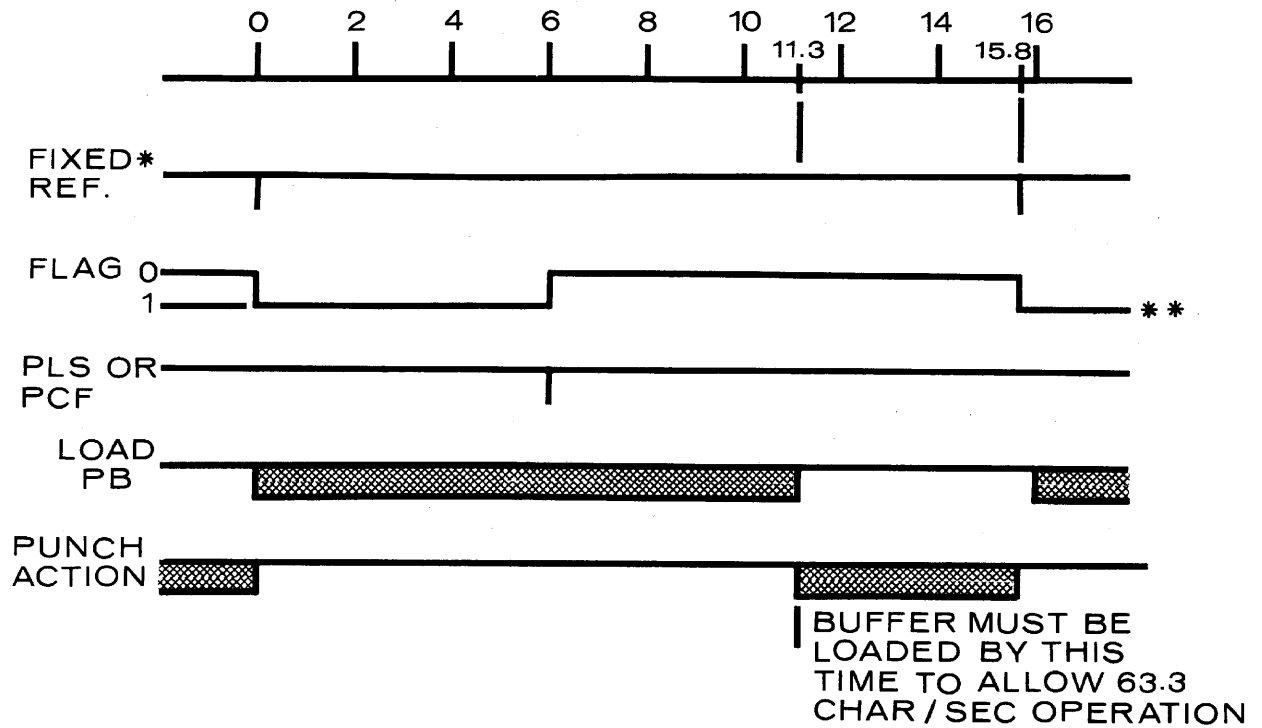
pls - 700206 - Load a character into PB from AC bits 10-17. Clear the Punch Flag. Punch the specified character.

700204 - $C(PB) \vee C(AC) \Rightarrow C(PB)$. Punch the $C(PB)$.

PROGRAM SEQUENCES

/punch the contents of AC and wait	
pls	/700206 punches AC 10-17
psf	/wait till done loop beginning
jmp .-1	/wait till done loop end
/wait for previous punching, then punch next	
psf	/wait loop for previous character punching
jmp.-1	/wait loop end
pls	/punch the next character on tape

In the first sequence above, 11.3 milliseconds of program time is available between the instruction following the wait loop and the next pls that can be given. In the second sequence, 15.8 milliseconds or more program time is available between the pls and the next time a pls can be given.



- * DETERMINED BY PUNCH
- ** IF PCF FLAG WILL NOT COME ON UNTIL NEXT P IS COMPLETE

Figure 19 Perforated-Tape Punch timing

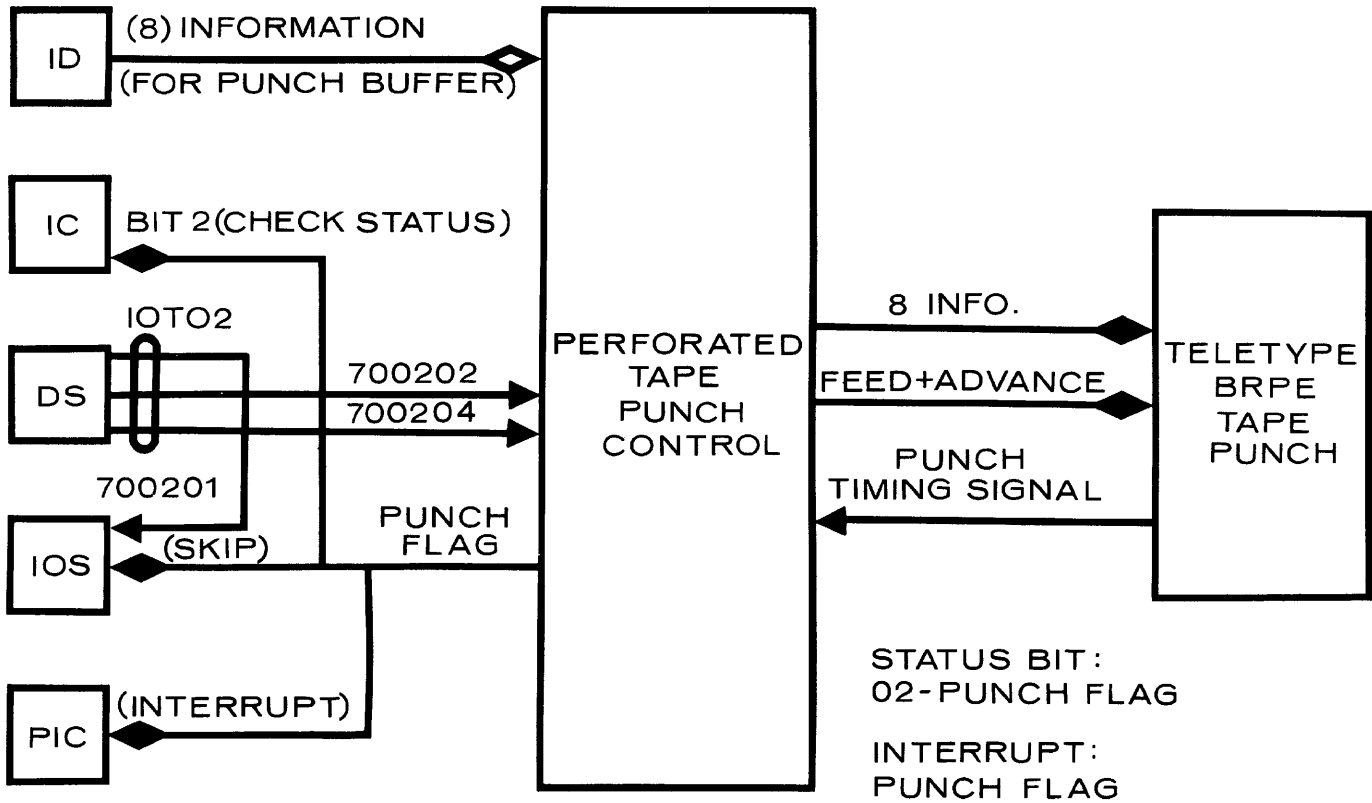


Figure 20 Perforated-Tape Punch programming logic

Card Reader and Control, Type 40-200-4

The control of the card reader is different than the control of other input devices, in that the timing of the read-in sequence is dictated by the device. Once the command to fetch a card is given, the reader will read all 80 columns of information in order. To read a column, the program must respond to a flag set as each new column is started. The instruction to read the column must come within 300 microseconds after the flag is set. The interval between flags is 2.3 milliseconds. Figure 21 shows the timing sequence following a command to read one card. The commands for the card reader, iot series 67, are:

crsf - 706701 - Skip if Card Reader Flag is a 1. If a card column is present for reading, the instruction will skip.

crrb - 706712 - Read the card column buffer information into AC and clear the Card Reader Flag. One crrb reads alphanumeric information. Two crrb instructions read the upper and lower column binary information.

crsa - 706704 - Select a card in alphanumeric mode. Select the card reader and start a card moving. Information will appear in alphanumeric form.

crsb - 706714 - Select a card in binary mode. Select the card reader and start a card moving. Information will appear in binary form.

Upon instruction to read the card reader buffer, six information bits are placed into AC₁₂ through AC₁₇. Alphanumeric (or Hollerith) information on the card is encoded or represented with these six bits. The binary mode enables the 12 bits (or rows) of each column to be obtained. The first read buffer instruction transfers the upper six rows (Y, X, 0, 1, 2, and 3), the second instruction the lower six rows (4, 5, 6, 7, 8, and 9). The mode is specified with the Card Read Select instruction. The mode can be changed while the card is being read.

The Card Read Flag is connected to the Program Interrupt Control and to bit 9 of the iors instruction. The Card Read Done status level bit is connected to bit 10 of the iors instruction. A Card Read Malfunction status is connected to bit 11 of the iors instruction. Card Read Malfunction status indicates one or more of the following conditions: reader not ready (power off, etc.), hopper empty, stacker full, card jam, validity check error (if validity is on), or real circuit failure.

Bit 12 of the iors instruction is connected to the END OF FILE switch at the Card Reader.

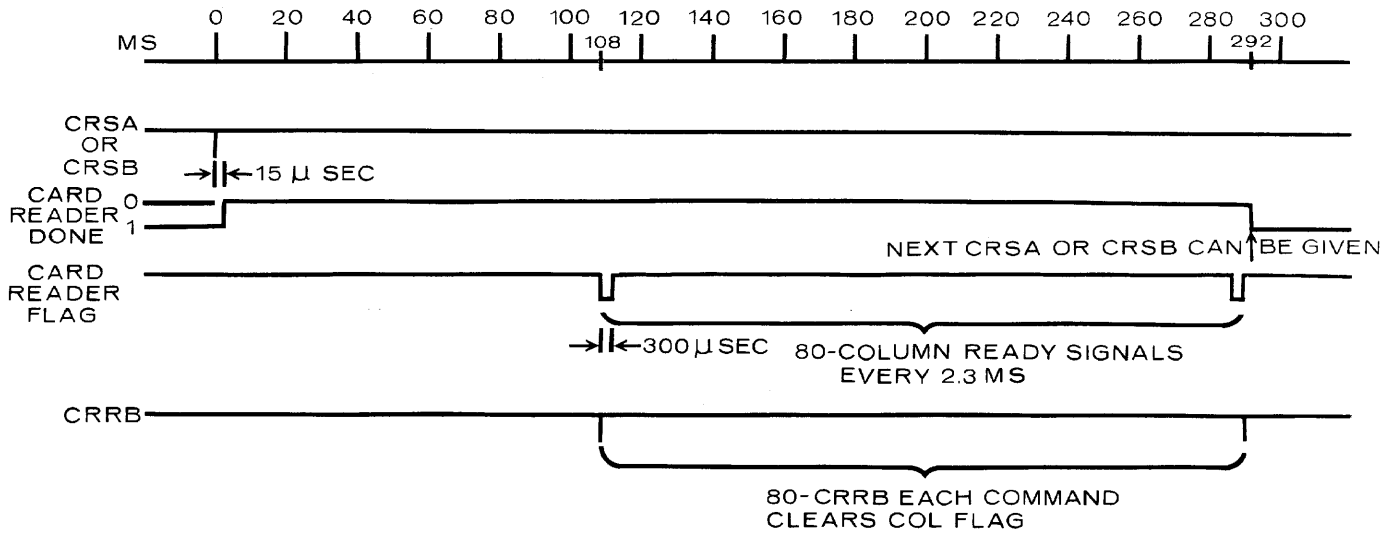


Figure 21 Card Reader timing

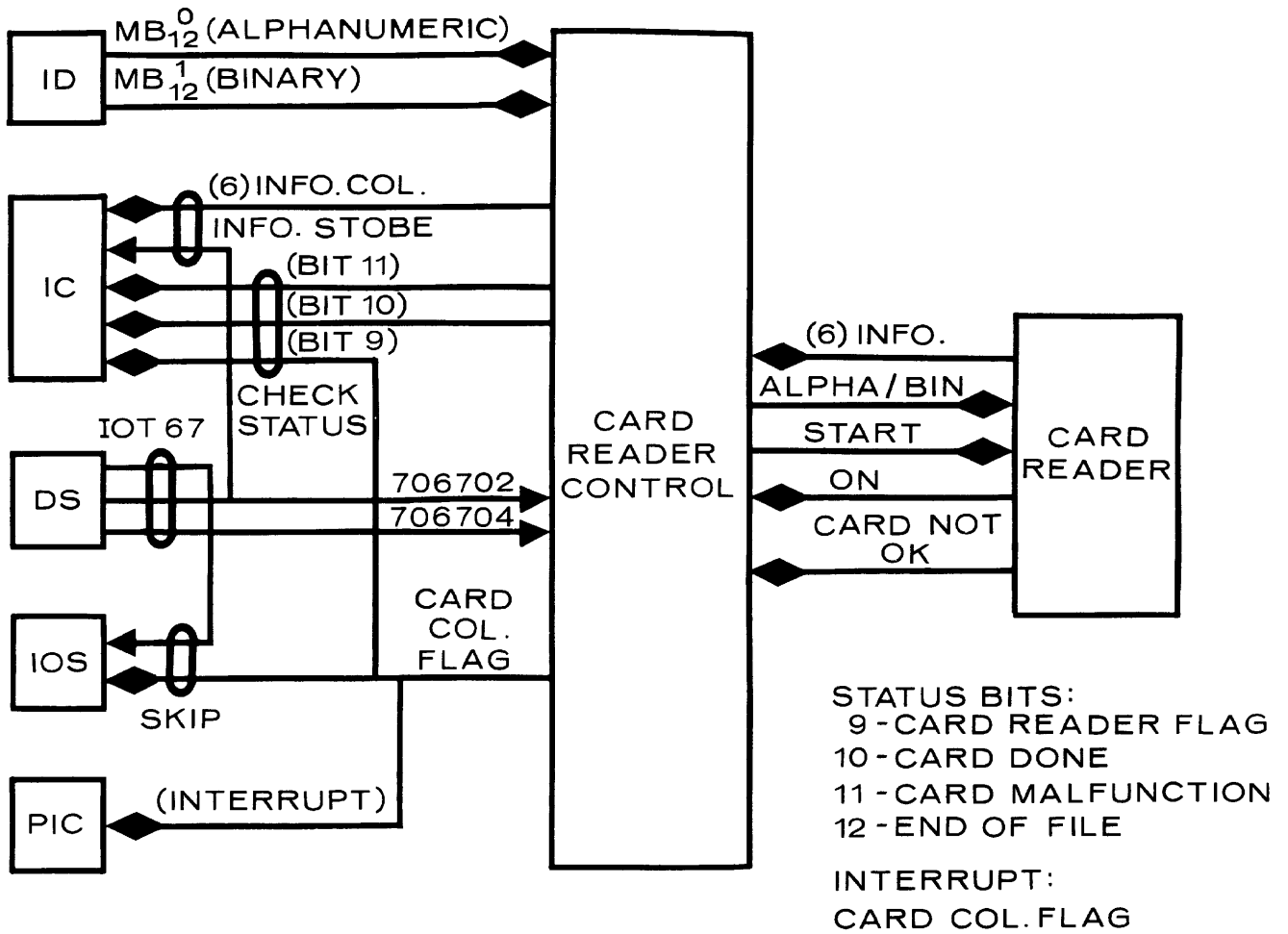


Figure 22 Card Reader programming logic

The switch is activated manually, and when depressed, holds until the RESET END OF FILE switch is depressed.

PROGRAM SEQUENCE

```

/sequence to read an 80-column card and place alphanumeric codes
/in register 1000-1117 (octal). Program begins in register cardrd.

cardrd,      crsa          /read card in alphanumeric mode
             lac cardlo    /initialize card location table
             dac 10        /place in indexable register
             lac cardct    /initialize card count 80 (decimal)
             dac temp

cdloop,      crsf          /wait for column loop
             jmp cdloop
             crrb          /place column information in ac
             dac i 10      /info to 1000, 1001...1117
             isz temp
             jmp cdloop
             hlt          /finish of card, and halt

cardlo,      1000-1        /location of card table

cardct,      -120+1       /80 column counter initial value

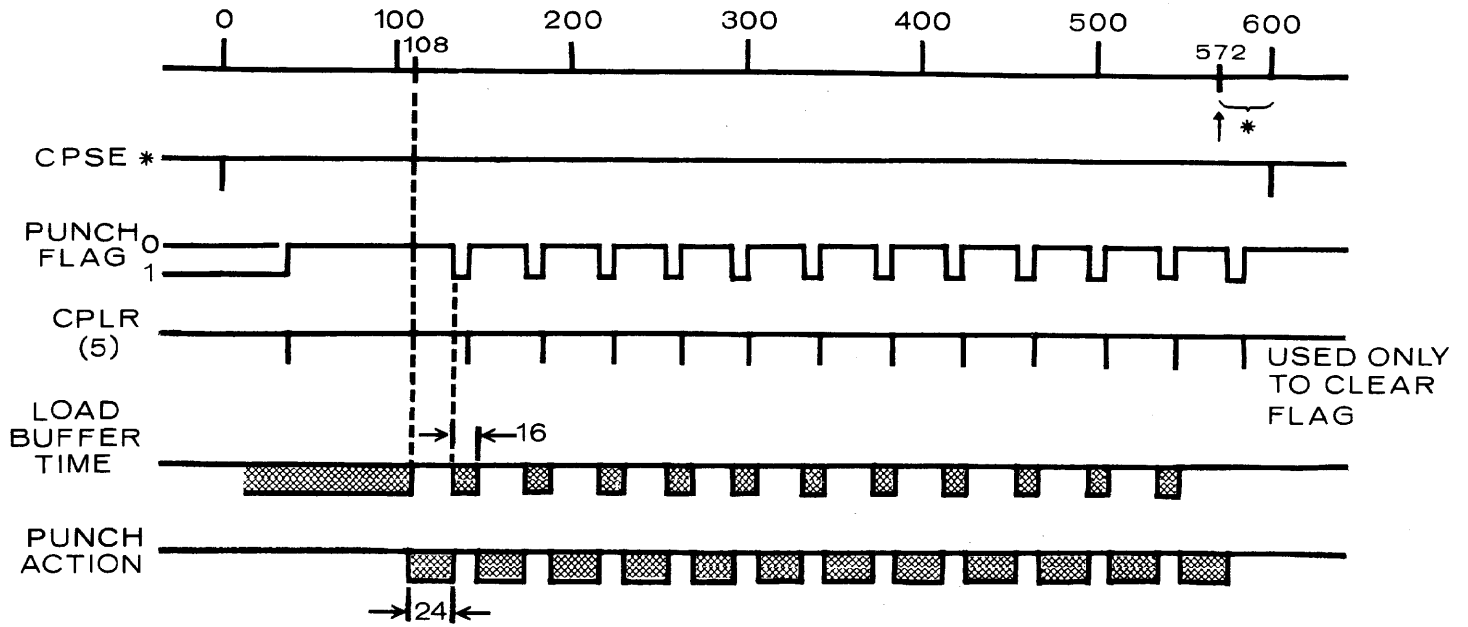
temp,        0            /reserved for column counter

```

Card Punch, Type 41-523-4

The card punch dictates the timing of a read-out sequence, much as the Card Reader controls the read-in timing. Once a card has started, all 12 rows will be punched at intervals of 40 milliseconds. Punching time for each row is 24 milliseconds, leaving 16 milliseconds to load the buffer for the next row. A flag indicates that the buffer is ready to load. Figure 23 shows the timing sequence following a command to read one card. The commands for the card punch, iot series 64, are:

- cpsf - 706401 - Skip if Card Punch Flag is a one. The Card Punch Flag indicates the punch buffer is available, and should be loaded.
- cpcf - 706402 - Clear Card Punch Flag.
- cpsc - 706442 - Select the Card Punch. Transmit a card to the 80 column punch die from the hopper.
- cplb - 706406 - Load the Card Punch Buffer from the C(AC). Five load instructions must be given to fill the buffer.



*CPSE MUST BE GIVEN TO MAINTAIN MAX RATE. A DELAY OF 600 OR 1200 MS WILL EXIST ON STARTING.

Figure 23 Card Punch timing

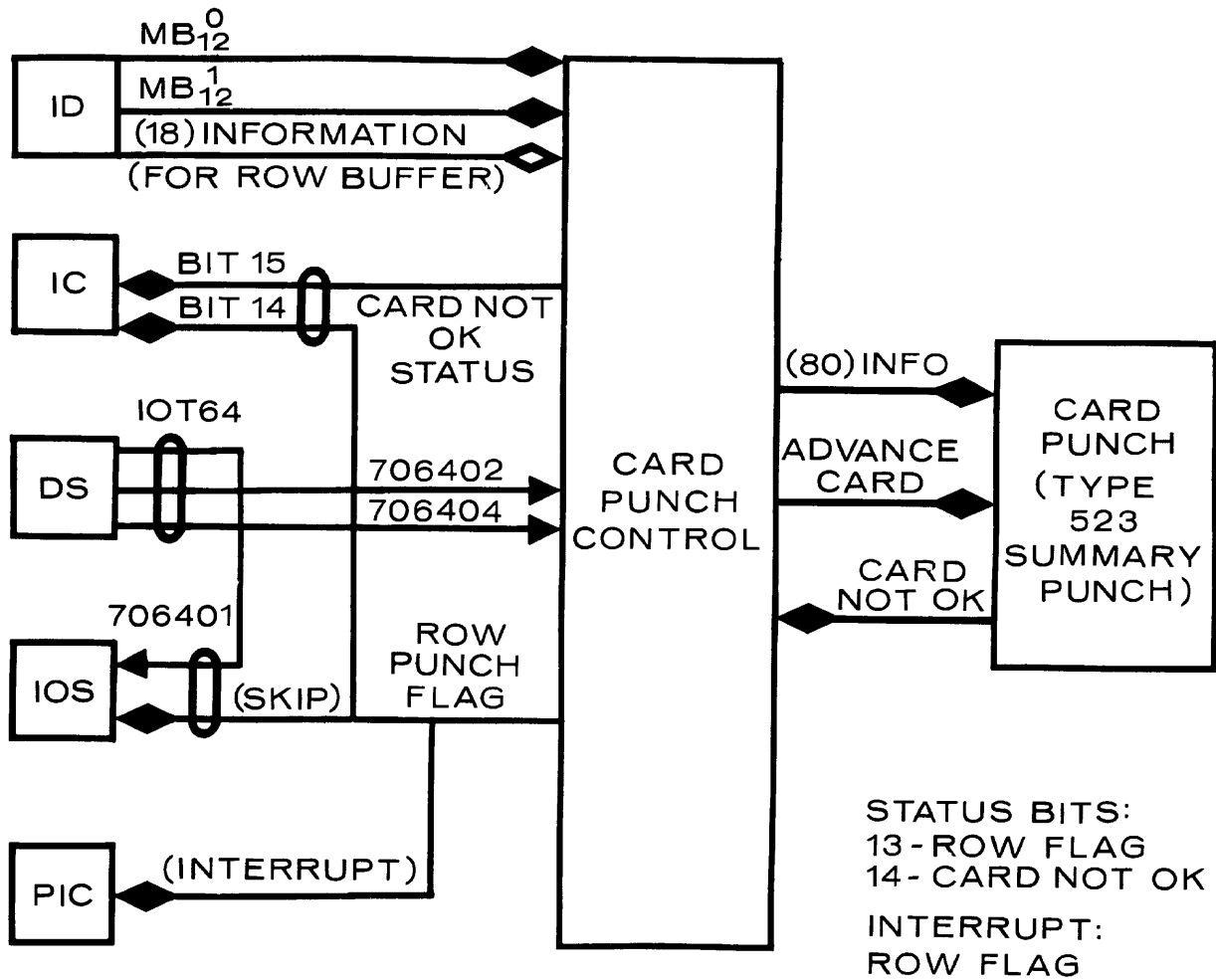


Figure 24 Card Punch programming logic

Since 18 bits are transmitted with each iot instruction, 5 iot instructions must be issued to load the 80-bit row buffer. The first four loading instruction fill the first 72 bits (or columns); the fifth loads the remaining 8 bits of the buffer from AC₁₀ through AC₁₇.

After the last row punching is complete, 28 milliseconds are available to select the next card for punching. If the next card is not requested in this interval, the card punch will stop. The maximum rate of the punch is 100 cards per minute in continuous operation. A delay of 1308 milliseconds follows the command to read the first card; a delay of 108 milliseconds separates the reading of cards in continuous operation.

The Card Punch Flag is connected to the Program Interrupt, and to bit 13 of the instruction. Faults occurring in the punch are detected by status bit 14 of iors and signify the punch is disabled, or the stacker is full, or the hopper is empty.

PROGRAM SEQUENCE

/sequence to punch 12 rows of data on a card. Each row is stored in
/5 consecutive registers beginning in location 100. The program begins
/in register cardph.

cardph,	cpse lac punloc dac 10 lac rowct dac temp1	/select the card /initialize the card image /initialize the row counts, 12.
/bop1,	lac grpct dac temp2 cpsf jmp.-1	/initialize the 5 groups per row /sense punch load availability
loop2,	lac i 10 cplr isz temp2 jmp loop2 isz temp1 jmp loop1 hlt	/5 groups of 18 bit per row /load buffer command /test for 12 rows /end punching 1 card
punloc,	100-1	/location of card image
rowct,	-14+1	/12 rows per card
grpct,	-5+1	/5 groups per row
temp1,	0	/row counter
temp2,	0	/group counter

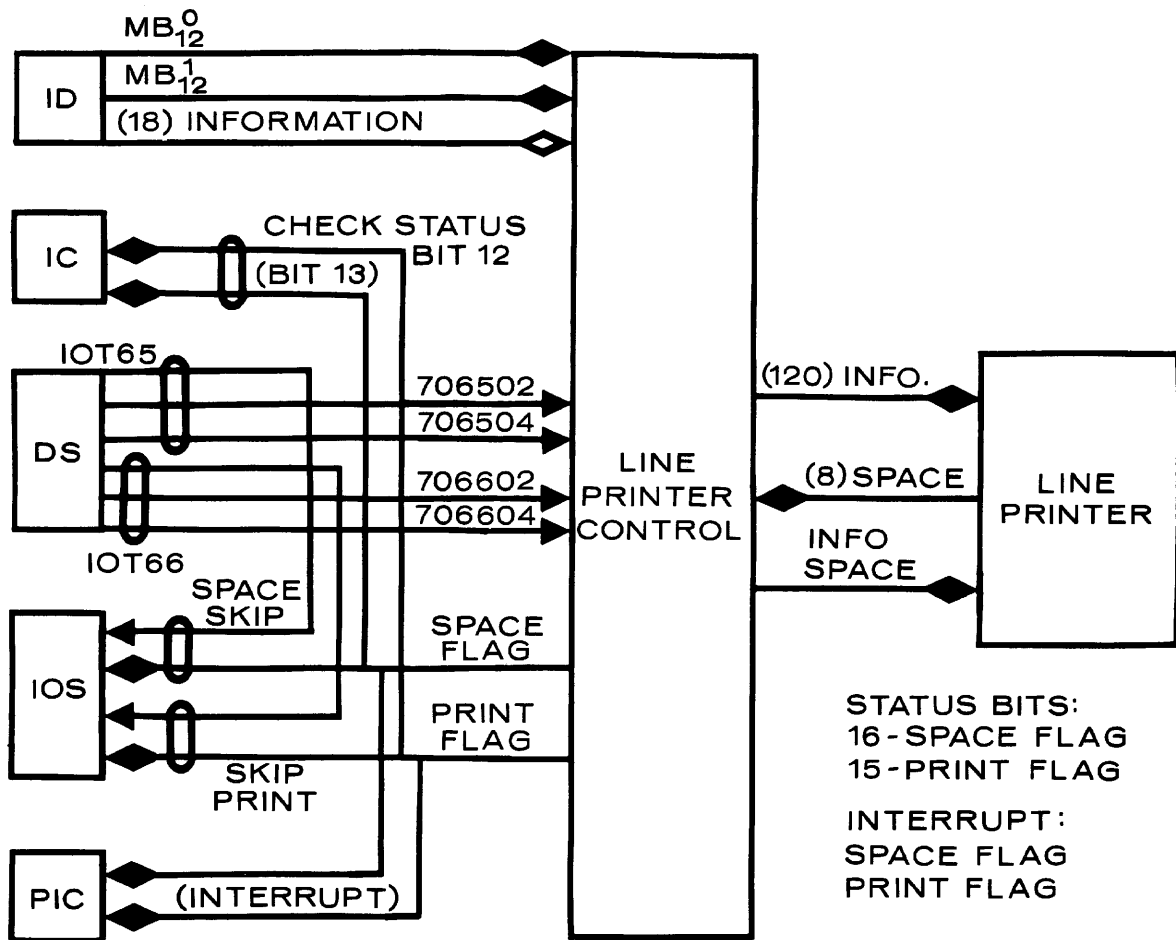


Figure 25 Line Printer programming logic

Line Printer, Type 62

The Line Printer can print 600 lines of 120 columns per minute. Each column has 64 characters. Spacing rate is approximately 132 lines (or 2-66 line pages) per second.

A complete line, or a 120 columns of information, is placed in the Printing Buffer. Six bits specify each character (the codes are given in Appendix 4). The information is transferred to the Printing Buffer through the AC, three characters at a time from AC bits 0-5, 6-11, and 12-17. Forty load print buffer instructions fill the 120-column line.

After the Printing Buffer is loaded, a print instruction is given which prints the contents of the Buffer. The action of printing does not disturb the Printing Buffer. When a column of information has been printed, the Printing Flag becomes a 1. Approximately 80 milliseconds are required to print one line.

An eight-channel format-control tape within the printer moves in synchronism with the paper and specifies how far the paper is to be spaced. Holes punched in each channel of the format tape signify the next paper position. The channel is selected by placing a three-bit code in AC₁₅ through AC₁₇, and giving an instruction to space paper. The Spacing flag becomes a one when the spacing action is complete. The tape has the following characteristics:

Channel (AC ₁₅₋₁₇)	Action	Time
0	Space one line	16 msec.
1	Restore page	520 msec. for 66 lines
2	Space two lines	< 2 x 16 msec.
3	Space three lines	< 3 x 16 msec.
4	1/4 page	
5	1/2 page	
6	1/6 page	
7	Not used	

The line printer printing and spacing instructions, iot series 65 and 66, are:

lpsf - 706501 - Skip if the Printing Flag is a 1.

lpcf - 706502 - Clear the Printing Flag.

lpld - 706542 - Load the Line Printer Buffer.

lpsc - 706506 - Select the printer. Print the contents of the Printer Buffer. Clear the Printing Flag. (The Printing Flag becomes a 1 at the completion of the printing.)

lssf - 706506 - Skip when the Spacing Flag becomes a 1.

lscf - 706602 - Clear the Spacing Flag.

lsls - 706606 - Load the Spacing Buffer from AC₁₅₋₁₇ and select spacing. Clear the Space Flag. (The spacing flag becomes a 1 when spacing is complete.)

The Printing and Spacing Flags are connected to the Program Interrupt and to the iors instruction bits 15 and 16.

PROGRAM SEQUENCE

/sequence to print a line of 120 columns. Output stored 3 columns
/per word.
/Data begins in register 2000. Sequence assumes printer is
/in process of printing a line previously assigned. "Print" is
/begin of prog.

```
print,  lpsf      /wait till previous printing done
        jmp.-1
        cla
        lsls     /space 1 line (0 in ac)
        lac (2000-1) /location of data
        dac 10   /print table initialize
        lac (-50+1) /40x3 characters
        dac temp

ldloop, lac i 10  /load print buffer loop
        lpld     /load from ac
        isz temp
        jmp ldloop

space,  lssf     /test for spacing done before proceeding
        jmp space
        lpse     /print activate...end of printing a line
```

CHAPTER 4

THE INTERFACE ELECTRICAL CHARACTERISTICS

As explained in previous sections, the standard Interface contains the Real-Time Connection, which can operate only with the perforated-tape reader, the perforated-tape punch, and the keyboard-printer. The Real-Time Option can operate with a variety of external devices over a wide range of information handling rates. In this section the location of the Real-Time Option, its electrical characteristics, and its connections to input-output devices are presented.

REAL-TIME OPTION

A coordinate system locates modules and connectors in PDP-4 with a four place, alphanumeric code. Bays are numbered 1 and 2, panels are lettered alphabetically downward, connectors or modules are numbered left to right in the panels (blank spaces included), and terminals are lettered alphabetically downward on the connectors or modules. The Real-Time Option is located in panels 2E, 2F, and 2H. Connections to external control units are made through a cable connector in positions 2J1-6.

The Device Selector (location 2F6-25)

The standard Device Selector contains provisions for up to 20 selector modules, each of which is a Pulse Amplifier, Type 4605. The amplifiers are pulsed with standard DEC 4000 Series negative logic pulses which can drive 18 units of base load.

Each module is wired to respond to one address code only (see Example, Figure 26). The 6-bit address portion of the iot instruction will therefore pass only through the six-level AND gate of those modules wired to the same combination of ones and zeros. The output of the AND gate enables three AND gates to pass the common iot 1, 2, and 3 pulses. These pulses are available at terminals E, H, and K, respectively, of modules 2F6-25.

The Device Selector Modules are delivered with jumpers across the address terminals. The user can remove appropriate jumpers to establish the module select mode according to the table below.

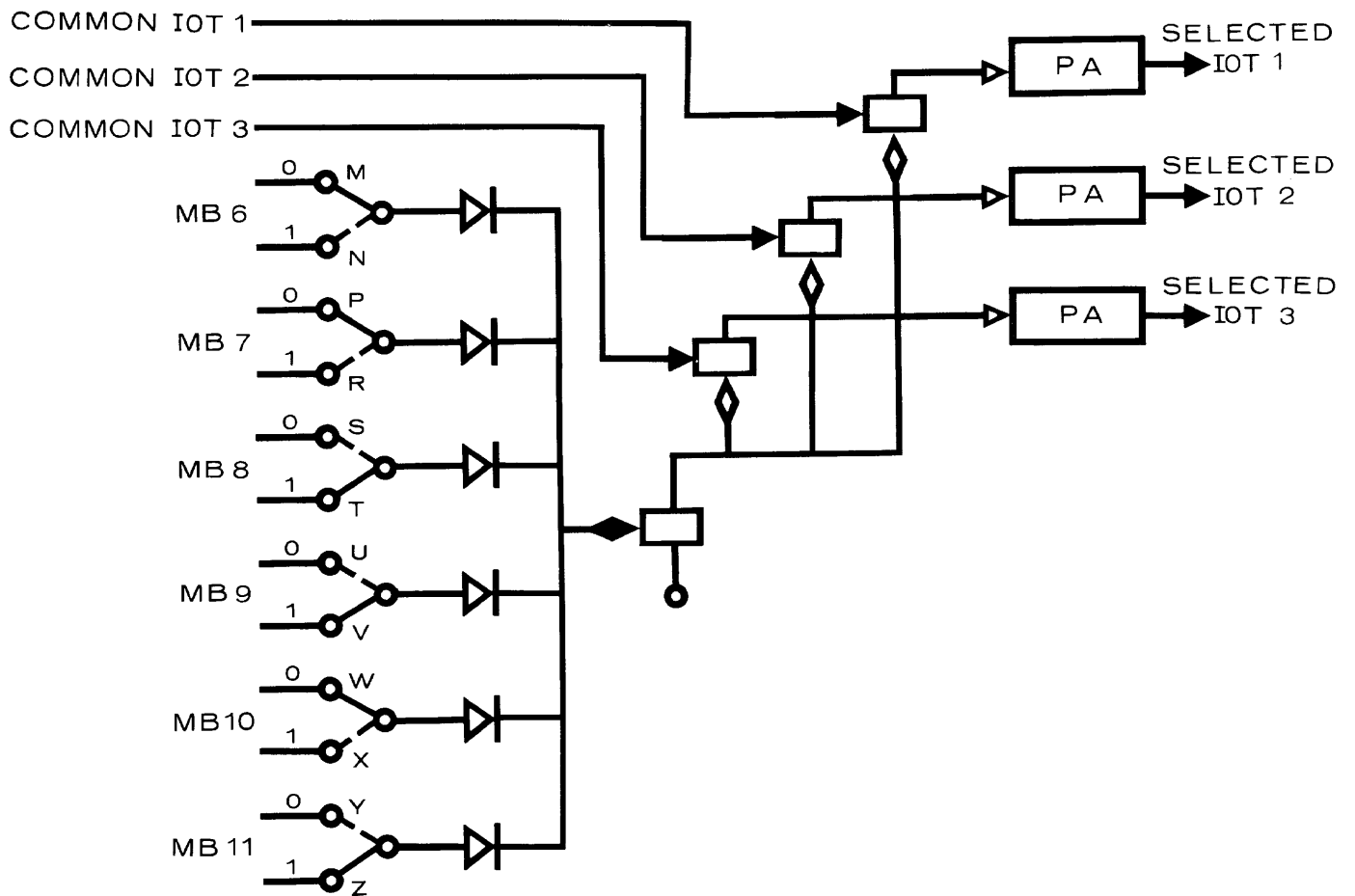


Figure 26 Typical Pulse Amplifier, Type 4605, used in PDP-4 Device Selector. Example shown is wired to pass the IOT address 001101. The six-level AND gate will pass only that address if it is present in the instruction word from the Memory Buffer, thus enabling three AND gates to pass three IO pulses to the pulse amplifier.

Instruction Word Bit	ZERO Input Terminal	ONE Input Terminal
6	M	N
7	P	R
8	S	T
9	U	V
10	W	X
11	Y	Z

Information Collector (location 2H8-25)

The information collecting sequence begins with an iot pulse from the Device Selector applied to the strobe input of the Information Collector. The IC then ANDS with the Input Device information present level and the results are transmitted to the AC. The results of the AND functions are mixed, or ORed together, to enable eight 18-bit-word devices to read data into the AC. Two or more devices requiring less than 18 bits could share a word, provided their bit-position requirements did not conflict. In such cases, more than eight input devices could be handled by the IC. The incoming information signal polarities are:

0 volts	0 bit transmitted to AC
-3 volts	1 bit transmitted to AC

The IC consists of 18 modules, one for each bit of the word, starting with bit 0 in module 2H8. All eight input channels are wired to each module. The convention for designating bits is $IC_{i,k}$, where i specifies the bit number and k the channel number. The eight input-level terminals and associated iot-pulse terminals are:

Channel (k)	Data-Bit Input	Associated iot Input
0	E	F
1	H	J
2	K	L
3	M	N
4	S	T
5	U	V
6	W	X
7	Y	Z

Information Distributor (location 2H1-3)

The Information Distributor presents the static data contained in the AC to an output device when the Device Selector commands the device to sample the ID. The signal polarities are:

-3 volts		AC bit contains a 0
0 volts		AC bit contains a 1

Eight groups of 18 outputs are available in the ID. The module driving the output bus is a Type 1690 or 1685 Bus Driver supplying up to 15 ma at 0 or -3 volts. All eight groups must share the bus.

Connections to the ID are made at three taper-pin terminal blocks, 2H1, 2H2, 2H3. Each block has 3 columns of 20 terminals each. Each column represents a group; the first 18 terminals (A-U) in the column represent AC bits 0-17 and the last two (V, W) the bipolar bit 12 in the Memory Buffer. V and W may be used to select a subdevice. The terminals are tied together horizontally to form 20 rows.

Input-Output Skip Facility (location 2H06)

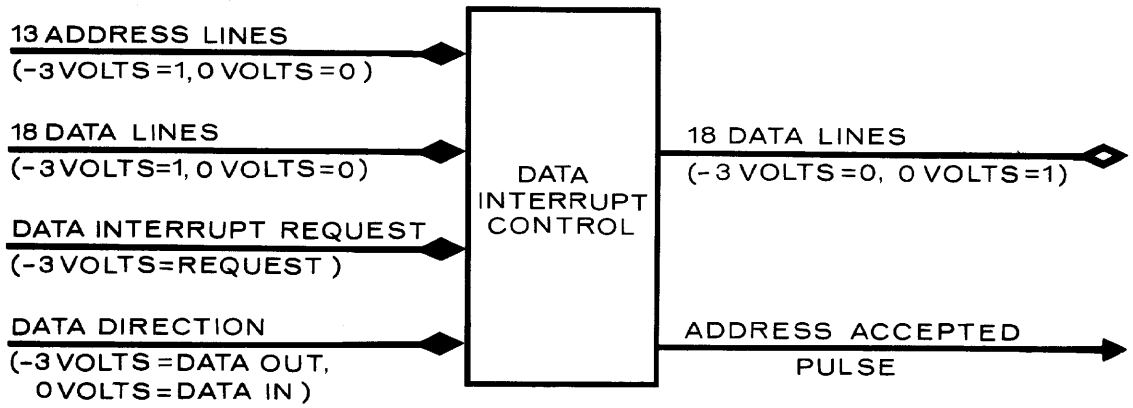
There are 8 inputs to Input-Output Skip. The iot pulses from the Device Selector strobe an input line and if a logic condition is present, the instruction following the iot will be skipped. The conditions for skipping are:

-3 volts		skip
0 volts		do not skip

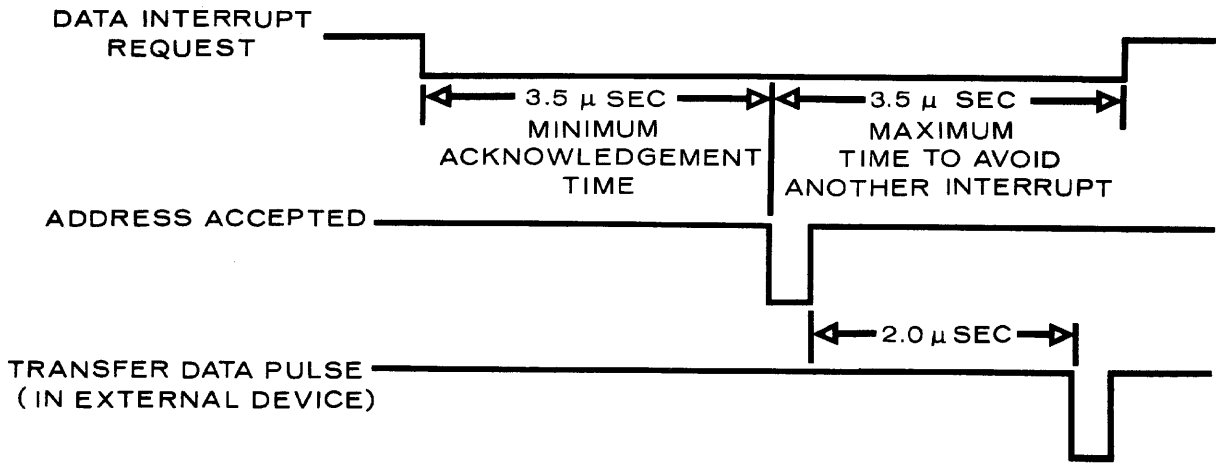
The iot skip pulse must occur at event time 1.

The IOS consists of a Capacitor-Diode Gate, Type 4129. The input connections are:

IO Device Input Connection	Device Selector Pulse Connection
F	E
J	H
L	K
N	M
T	S
V	U
X	W
Z	Y



(a) DIC SIGNALS



(b) DIC TIMING

Figure 27 Data Interrupt Control signals and timing

Program Interrupt Control (location 2H05)

Eleven Program Interrupt lines are available. Any one of the 11 signals may cause an interruption of a program. All signals are identical; the polarities are:

-3 volts		interrupt the program
0 volts		no effect

The connections from IO devices which request program interrupt are made to module 2H05 at pins E, F, H, J, S, T, U, W, X, Y, and Z.

Data Interrupt Control (location 2E13)

The signal levels associated with the DI are shown in Figure 27. In transferring data, the Memory Address is first transmitted to the Memory Address Register on 13 lines from the external source. Data is next transferred to or from the MB on 18 + 18 lines.

Incoming data is received from 18 lines and placed in the Memory Buffer and on into Memory.

Outgoing data from the core memory addressed is transferred to the Memory Buffer and appears on 18 lines for sampling by the IO device.

APPENDIX 1
TABLE A.1. MEMORY REFERENCE INSTRUCTIONS

Octal Code	Mnemonic Code	Time (μ sec)	Name	Operation
00	cal Y	16	Call Subroutine	Y is ignored jms 20 if bit 4=0, jmsi 20 if bit 4=1.
04	dac Y	16	Deposit AC	$C(AC) \Rightarrow C(Y)$
10	jms Y	16	Jump to sub routine	$C(PC) \Rightarrow C(Y,),$ $Y + 1 \Rightarrow C(PC)$
14	dzm Y	16	Deposit zero in memory	$0 \Rightarrow C(Y)$
20	lac Y	16	Load AC	$C(Y) \Rightarrow C(AC)$
24	xor Y	16	Exclusive OR	$C(AC) \oplus C(Y) \Rightarrow C(AC)$
30	add Y	16	Add (one's complement)	$C(AC) + C(Y) \Rightarrow C(AC)$
34	tad Y	16	Two's add (two's complement)	$C(AC) + C(Y) \Rightarrow C(AC)$
40	xct Y	8+	Execute	
44	isz Y	16	Index and skip if zero	$C(Y) + 1 \Rightarrow C(Y),$ if $C(Y) + 1 = 0$, then $C(PC) + 1 \Rightarrow C(PC)$
50	and Y	16	AND	$C(AC) \wedge C(Y) \Rightarrow C(AC)$
54	sad Y	16	Skip if AC and Y differ	If $C(AC) = C(Y)$, then $C(PC) + 1 \Rightarrow C(PC)$
60	jmp Y	8	Jump	$Y \Rightarrow C(PC)$
76	law N	8	Load a Word	$1 \Rightarrow C(AC_{0-4}),$ $N \Rightarrow C(AC_{5-17})$

APPENDIX 2

TABLE A.2. OPERATE INSTRUCTIONS

Octal Code	Mnemonic Code	Name	Operation
740000	opr	Operate	None
740000	nop	No Operation	None
740001	cma	Complement	$C(AC) \Rightarrow C(AC)$
740002	cml	Complement Link	$C(L) \Rightarrow C(L)$
740004	oas	Inclusive OR ACS	$C(ACS) \vee C(AC) \Rightarrow C(AC)$
750004	las	Load AC from Switches	$C(AC) \text{ Switches}) \Rightarrow C(AC)$
740010	ral	Rotate AC + Link left one place	$C(AC_i) \Rightarrow C(AC_{i-1}),$ $C(L) \stackrel{\pm}{\Rightarrow} C(AC_{17})^{i-1},$ $C(AC_0) \Rightarrow C(L)$
742010	rtl	Rotate AC left twice	Same as two ral instructions
740020	rar	Rotate AC + Link right one place	$C(AC_i) \Rightarrow C(AC_{i+1}),$ $C(L) \stackrel{\pm}{\Rightarrow} C(AC_0)^{i+1},$ $C(AC_{17}) \Rightarrow C(L)$
742020	rtr	Rotate AC right twice	Same as two rar instructions
740040	hlt	Halt	$0 \Rightarrow \text{RUN}$
740200	sza	Skip on zero AC	Skip if $C(AC) = \text{positive zero}$
741200	sna	Skip on non-zero AC	Skip if $C(AC) \neq \text{positive zero}$
741100	spa	Skip on positive AC	Skip if $C(AC_0) = 0$
740100	sma	Skip on negative AC	Skip if $C(AC_0) = 1$
741400	sfl	Skip on zero Link	Skip if $C(L) = 0$
740400	snl	Skip on non-zero Link	Skip if $C(L) = 1$
741000	skp	Skip, unconditional	Always skip
744000	cll	Clear Link	$0 \Rightarrow C(L)$
744002	stl	Set the Link	$1 \Rightarrow L$
750000	cla	Clear AC	$0 \Rightarrow C(AC)$
741001	clc	Clear and Complement AC	$-0 \Rightarrow C(AC)$

APPENDIX 3

TABLE A.3. BASIC IOT INSTRUCTION GROUP

```

/Interrupt
  iof=700002 /turn off interrrupt
  ion=700042 /turn on interrrupt

/IO Equipment
  iors=700314 /read status of io equipment

/Clock
  clsf=700001 /skip if clock flag is 1
  clof=700004 /turn off clock, clear clock flag
  clon=700044 /turn on clock, clear clock flag

/Paper tape reader
  rsf=700101 /skip if reader flag is a 1
  rsa=700104 /select reader for alphanumeric,clear reader flag
  rsb=700144 /select reader for bry, clear reader flag
  rrb=700112 /read the reader buffer into AC, clear reader flag

/Paper tape punch
  psf=700201 /skip if punch flag is a 1
  pls=700206 /load punch buffer and select punch, clear punch flag
  pcf=700202 /clear punch flag

/Keyboard input from teleprinter
  ksf=700301 /skip if keyboard flag is a 1
  krb=700312 /read the keyboard buffer into the AC, clear keyboard flag

/Teleprinter
  tsf=700401 /skip if teleprinter flag is a 1
  tls=700406 /load teleprinter buffer and select, clear teleprinter flag
  tcf=700402 /clear the teleprinter flag

/Display type 30A
  dsf=700501 /skip if display flag is a 1
  dls=700506 /load display buffer and select, clear display flag
  dcf=700502 /clear display flag

/Display type 30D
  dsf=700501 /skip if display flag is a 1 (light pen)
  dcf=700601 /clear display flag
  dxl=700506 /load x co-ordinate
  dxs=700546 /load x co-ordinate and select
  dyl=700606 /load y co-ordinate
  dys=700646 /load y co-ordinate and select
  dlb=700706 /load brightness register

```

TABLE A.3. BASIC IOT INSTRUCTION GROUP (continued)

```

/Magnetic tape type 54
  mci=707001 /clear tape instruction and character buffer
  mrs=707012 /read tape status into AC
  mli=707005 /load instruction buffer
  msc=707101 /skip if character is present for reading
  msi=707201 /clear interrupt flag and select interrupt
  msf=707301 / skip if the tape flag is a 1 (end of record)

  mrl=707112 /clear AC, read character buffer into AC left
              /clear character buffer
  mrm=707202 /read character buffer into AC middle
              /clear character buffer
  mrr=707302 /read character buffer into AC right
              /clear character buffer

  mwl=707104 /write a character from AC left
  mwm=707204 /write a character from AC middle
  mwr=707304 /write a character from AC right

/Card reader
  crsf=706701 /skip if reader character flag is a 1
  crsa=706704 /select card reader for alphanumeric
  crsb=706744 /select card reader for binary
  crrb=706712 /read card column buffer into AC

/Card punch
  cpsf=706401 /skip if the card punch flag is a 1
  cpse=706444 /select a card, set card punch flag
  cplr=706406 /load row buffer, clear punch flag
  cpcf=706442 /clear punch flag

/Line printer
  lpsf=706501 /skip if printing flag is a 1
  lpcf=706502 /clear printing flag
  lpld=706542 /load the printing buffer
  lpse=706506 /select printing, clear printing flag

  lssf=706601 /skip if spacing flag is a 1
  lscf=706602 /clear spacing flag
  lsls=706606 /load spacing buffer and select spacing, clear spacing fla

```

start

.

TABLE A.3.1 IORS COMMAND BIT ASSIGNMENTS AND PROGRAM INTERRUPT CONNECTIONS

IORS Bit	Device	Program Interrupt Connected	Status if a 1
0	Prog. Interrupt	x	Prog interrupt is on
1	Tape Reader Flag	x	Reader buffer has a character
2	Tape Punch Flag	x	Punching is complete, punch is available
3	Keyboard Flag	x	Keyboard buffer has a character
4	Teleprinter Flag	x	Character has been printed, and is available
5	Display Flag	x	Light pen flag is a one
6	Clock Flag	x	Clock has overflowed
7	Clock Status		Clock counting is enabled
8	Magnetic Tape Flag	x	End of record
9	Card Reader Flag	x	Card column is available
10	" " "		Card is at reading station
11	" " "		Card Malfunction
12	" " "		End of File button is pressed
13	Card Punch Flag	x	Punch buffer is available to load
14	" " "		Card malfunction
15	Line Printer Printing Flag	x	Printing is completed
16	Line Printer Space Flag	x	Spacing is completed

APPENDIX 4

TABLE A.4.1 FIO-DEC CODE

		High order bits			
		00	01	10	11
a A	61				
b B	62	Low order bits			
c C	63				
d D	64				
e E	65	0000	space	0 →	• _
f F	66				
g G	67	0001	1 "	/ ?	j J a A
h H	70				
i I	71	0010	2 ' .	s S	k K b B
j J	41				
k K	42	0011	3 ~	t T	l L c C
l L	43				
m M	44	0100	4 ∩	u U	m M d D
n N	45				
o O	46	0101	5 V	v V	n N e E
p P	47				
q Q	50	0110	6 ^	w W	o O f F
r R	51				
s S	22	0111	7 <	x X	p P g G
t T	23				
u U	24	1000	8 >	y Y	q Q h H
v V	25				
w W	26	1001	9 ↑	z Z	r R i I
x X	27				
y Y	30	1010			lower case
z Z	31				
0 →	20	1011	stop	, =	• x
1 "	01				
2 ' .	02	1100		black	- + upper case
3 ~	03				
4 ∩	04	1101		red) backspace
5 V	05				
6 ^	06	1110		tab	-
7 <	07				
8 >	10	1111			(car ret
9 ↑	11				
/ ?	21				
, =	33	• _	40		
• x	73				
- +	54				
)	55	-	56		
(57				

stop code 13
 lower case 72
 upper case 74
 black 34
 red 35
 tab 36
 backspace 75
 carriage return 77
 space 00

code delete punches seventh channel

TABLE A.4.2 TELETYPE CODE

		High order bits			
Low order bits		00	01	10	11
000			line feed	E 3	A -
001	T 5		L)	Z "	W 2
010	car ret		R 4	D \$	J '
011	O 9		G &	B ?	figures
100	space		I 8	S bell	U 7
101	H #		P O	Y 6	Q 1
110	N ,		C :	F !	K (
111	M .		V ;	X /	letters
letters	37			figures	33
A	30			0	15
B	23			1	35
C	16			2	31
D	22			3	20
E	20			4	12
F	26			5	01
G	13			6	25
H	05			7	34
I	14			8	14
J	32			9	03
K	36			(36
L	11)	11
M	07			.	07
N	06			,	06
O	03			-	30
P	15			?	23
Q	35			:	16
R	12			\$	22
S	24			bell	24
T	01			&	13
U	34			#	05
V	17			!	32
W	31			;	17
X	27			/	27
Y	25			!"	26
Z	21			"	21
space	04		carriage return		02
line feed	10				

TABLE A.4.3 CARD READER CODE

	Low order bits	High order bits			
		00	01	10	11
A	61				
B	62				
C	63				
D	64				
E	65	0000	blank	-	+ [&]
F	66				
G	67	0001	1	/	J
H	70				
I	71	0010	2	S	K
J	41				
K	42	0011	3	T	L
L	43				
M	44	0100	4	U	M
N	45				
O	46	0101	5	V	N
P	47				
Q	50	0110	6	W	O
R	51				
S	22	0111	7	X	P
T	23				
U	24	1000	8	Y	Q
V	25				
W	26	1001	9	Z	R
X	27				
Y	30	1010	0		
Z	31				
0	12	1011	= [#]	,	\$
1	01				
2	02	1100	' [©]	([%]	*) [□]
3	03				
4	04				
5	05				
6	06				
7	07				
8	10				
9	11				
+	60				
-	40				
/	21				
=	13				
,	33				
\$	53				
:	73				
;	14				
(34				
*	54				
)	74				
blank	00				

TABLE 3a. HOLLERITH CARD CODE

digit	no punch	Zone			
		no zone	12	11	0
		blank	+ [&]	-	0
1	1	A	J	K	/
2	2	B	L	M	S
3	3	C	N	O	T
4	4	D	P	Q	U
5	5	E	R	S	V
6	6	F	T	U	W
7	7	G	V	W	X
8	8	H	X	Y	Y
9	9	I	Z		Z
8-3		= [#]	.	\$,
8-4		' [©]) [□]	*	([%]

TABLE A.4.4 HIGH-SPEED LINE PRINTER CODE

		High order bits			
		00	01	10	11
A	61				
B	62	Low order			
C	63	bits			
D	64				
E	65	0000	space	0	·
F	66				—
G	67	0001	1	/	J
H	70				A
I	71	0010	2	S	K
J	41				B
K	42	0011	3	T	L
L	43				C
M	44	0100	4	U	M
N	45				D
O	46	0101	5	V	N
P	47				E
Q	50	0110	6	W	O
R	51				F
S	22	0111	7	X	P
T	23				G
U	24	1000	8	Y	Q
V	25				H
W	26	1001	9	Z	R
X	27				I
Y	30	1010	'	"	\$
Z	31				×
0	20	1011	~	,	=
1	01				·
2	02	1100	∩	>	—
3	03				+
4	04	1101	∨	↑)
5	05]
6	06	1110	∧	→	—
7	07				
8	10	1111	<	?	(
9	11				[
·	40				
/	21		—		60
'	12		"		32
~	13		,		33
∩	14		>		34
∨	15		↑		35
∧	16		→		36
<	17		?		37
\$	52		×		72
=	53		·		73
—	54		+		74
)	55]		75
(57		[77
—	56				76
space	00				

APPENDIX 5

READ-IN MODE SEQUENCE

A5.1 GENERAL

The initial data input to PDP-4 is made using the keys and switches on the operator console. A small program read in manually can be used to read in a somewhat larger program from perforated tape. An example of such a routine is given below. It can also be used to read in other programs from perforated tape.

Manually set the read-in loader into the machine by means of the ADDRESS and ACCUMULATOR switches and the DEPOSIT key on the console. When the manual program has been established, load the tape reader, set the ADDRESS switches to enter address, and operate the START key. The manual loader starts at 7770 and automatically transfers to the perforated tape loader when finished. The program keeps track of check sum, and halts at 7755 if an error is detected.

A5.2 MANUAL LOADER

The routine is loaded using the toggle switches on the console. Execution starts at register 7770. Since the subroutine to read one binary word always gives a read tape command after reading the buffer, the jmp instruction at the end of the tape must be followed by a dummy last word if the user wishes the tape to stop upon termination of read (necessary to prevent reader chatter when reading block format tapes).

<u>Location</u>	<u>Octal Code</u>	<u>Mnemonic</u>	<u>Remarks</u>
7762/r,	0	0	/read one binary word
7763/	700101	rsf	
7764/	607763	jmp i	/wait for word to come in
7765/	700112	rrb	/read buffer
7766/	700144	rsb	/read another word
7767/	627762	jmp i r	/exit subroutine
7770/	700144	rsb	/enter here, start reader going
7771/g,	107762	jms r	/get next binary word
7772/	47775	dac out	

<u>Location</u>	<u>Octal Code</u>	<u>Mnemonic</u>	<u>Remarks</u>
7773/	407775	xct out	/execute control word
7774/	107762	jms r	/get data word
7775/out,	0	0	/stored data word
7776/	607771	jmp g	/continue

A5.3 PERFORATED-TAPE LOADER

The block format loader will read a block format binary tape of the following format:

dac A	A is the address of the first data word
-N	/complement of number of data words in block
N data words	/data words
Checksum	/sum of every word in block

The routine occupies register 7737 to 7760, and uses the manual loader sub-routine to read each binary word. Upon completing a block, the computed check sum is compared with the read check sum and the loader halts if these differ. The block may be re-read by pulling the tape back to the beginning and pressing the CONTINUE switch on the console.

<u>Location</u>	<u>Mnemonic</u>	<u>Remarks</u>
7737/a,	rsb	
	jms r	/block format loader
	dac s	
	xct s	
	dac cks	
	jms r	
	dac out	
b,	add cks	/loop
	dac cks	
	jms r	

<u>Location</u>	<u>Mnemonic</u>	<u>Remarks</u>
	isz out	/check count, last word read is check sum
	jmp s	
	sad cks	
	jmp a	/sum checks, continue
	hlt	/stop on check sum error
	jmp a-1	/out
s,	xx	
	isz s	
	jmp b	

APPENDIX 6

PDP-4 ASSEMBLER PROGRAM

A6.1 GENERAL

The characteristics of the MACRO assembler program are defined here to provide the background necessary to understand the programming examples in this manual. A separate manual is available at DEC which describes the program and explains its use in detail.

A6.2 CHARACTER SET

The MACRO character set includes digits 0 through 9, letters a through z, and the following punctuation characters:

<u>Punctuation Characters</u>	<u>Meaning</u>
+ plus	add values
- minus	subtract values
△ space	add values
∧ and	combine values by logical AND
∨ or	combine values by INCLUSIVE OR
(left parenthesis	enclose constant word
) right parenthesis	enclose constant word
. period	has value of current address
, comma	assign address tag
= equal sign	assign symbol on left of =
/ slash	begin comments; set current address
↵ carriage return	termination character
→ tab	termination character
— overbar	variable indicator

The characters △ , ↵ , and → are used for invisible characters.

A6.3 NUMBERS

Any sequence of digits delimited on the left and right by a punctuation character.

A6.4 SYMBOLS

Any sequence of letters or digits, the first of which must be a letter. Symbols may be any length, but all characters over six are ignored.

'Value symbols' are those symbols which have a numerical value assigned to them, either in the permanent symbol table, or during assembly. Value symbols may be assigned by the use of a comma, indicating the symbol to the left of the comma is an address tag; or by an equals sign, indicating the symbol to the left of the equals sign is to be assigned the value of the word to the right of the equals sign.

Example: a, dzm 100
 b = -1
 c = a + b

A6.5 SYLLABLES

A syllable can take several forms. It can be a value symbol, a period (.), a flexo-writer input pseudo-instruction ("flex" or "char"), or a constant (a word enclosed in parentheses).

Examples of syllables are:

al
100
1z2
flex abc
flex now
(add a + 1)
lac abcdef

A6.6 WORDS

A word is a string of syllables connected by the arithmetic operators plus, minus, space, AND or OR, delimited on the left by tab, carriage return, left parenthesis, or equals sign; and on the right by a tab or carriage return. A word may be a single number or symbol so delimited, or a string of symbols connected by the operators. If the word is delimited on the left by an equals sign then the symbol to the left of the equal sign is assigned a value equal to that of the word. Otherwise, the word

is a storage word and will become part of the binary version of the program being assembled. The arithmetic operators, plus and space both mean add, while the operator minus means subtract.

Examples of words:

```
sad K ↙  
lac a ↙  
1000-20 ↙  
add b+2 ↙  
jmp -2 ↙  
a+b-c-2 ↙  
lac (add a+1) ↙
```

A6.7 THE CHARACTER SLASH (/)

The slash has two meanings: if immediately preceded by a tab or carriage return then slash initiates a comment, which is terminated by the next tab or carriage return. If slash is preceded by a word, then the address part of the word indicates to MACRO the address into which the next instruction or data word will go. Normally, MACRO translates the first instruction or data word into register 20 and succeeding instructions or data words into succeeding registers. If the programmer wishes to break this sequence or wishes to start translating into some register other than 20, then a slash may be used to set the new address.

A6.8 INDIRECT ADDRESSING

Indirect addressing is indicated by the symbol "i" which has the value 20000.

Example: lac i abc

A6.9 THE CHARACTER PERIOD (.)

The character (.) has as its value the current address.

Example: dac. is equivalent to

a, dac a

A6.10. PSUEDO - INSTRUCTIONS

A6.10.1 FLEXOWRITER INPUT PSUEDO INSTRUCTIONS

flex Δ α β γ

The psuedo-instruction flex causes the (six-bit) Teletype codes for the three characters following the space (Δ) to be read into one word which is taken as the value of the syllable. The code for the character α will go into bits 0 - 5 of the word, for into bits 6 - 11, and for γ into bits 12 - 17. The code is a six-bit character, the first five of which are the Teletype code, the sixth a 1 for upper case or a 0 for lower case.

Example: flex Δ boy

char Δ Z γ

The psuedo-instruction char causes the (six-bit) Teletype character γ to be read into the left, middle, or right six bits of the word, depending on whether Z is r, m, or l.

Example: char Δ r0

 char Δ la

A6.10.2 CONSTANTS

The MACRO assembly system has available a facility by which the program constants may be automatically stored. A constant must follow the rules for a word and is delimited on the left by a left parenthesis. The right delimiter may be a right parenthesis, carriage return, or tab. The value of the syllable (α) is the address of the register containing α . The constant α will be stored in a constants block at the end of the program, and the address of α will replace

Examples of the use of constants:

add (1) ↵

lac (add z 1) ↵

lac (-760000) ↵

lac (flexo abc) ↵

A6.10.3 START

The psuedo-instruction "start" indicates the end of the English tape. Instruction "start A" must be followed by a carriage return. The "A" is the address at which execution of the program is to begin, and causes a jmp A instruction punched at the end of the binary tape on pass 2.

A6.10.4 DECIMAL

The psuedo-instruction "decimal" tells MACRO to take all numbers as decimal.

A6.10.5 OCTAL

The psuedo-instruction "octal" tells MACRO to take all numbers as octal.

APPENDIX 7

MULTIPLY SUBROUTINE

/PDP-4 ones complement single precision multiplication subroutine

/calling sequence: /lac multiplier

/jms mult

/lac multiplicand

/return; low order product in AC, high order product in mp5

/time = 2.6 msec. for non-zero cases, approximatley 100 microsec. for zero.

```

mult, 0
    dzm mp5
    sna
    jmp mpz
    spa+c11-opr
    cma+cml-opr
    dac mpl
    xct i mult
    sna
    jmp mpz
    spa
    cma+cml-opr
    dac mp2
    lac (360000
    , ral
    dac mpsign
    lac (-21
    dac mp3
  
```

```

mp4, lac mpl
    rar
    dac mpl
    lac mp5
    spl+c11-opr
    tad mp2
    rar
    dac mp5
    isz mp3
    jmp mp4
  
```

```

mpsign, 0
    dac mp5
    lac mpl
    rar
    xct mpsign
  
```

```

mpz, isz mult
    jmp i mult
  
```

start

APPENDIX 8

DIVIDE SUBROUTINE

/PDP-4 ones complement divide subroutine
 /calling sequence: /lac high order dividend
 /jms divide
 /lac low order dividend
 /lac divisor
 /return; quot. in AC, rem. in dvd. if high dividend is
 /greater than divisor, no divide takes place and L => 1. Time = 3.1 msec.

```

divide,  0
         spa+c11-opr
         cma+cml--opr
         dac dvd
         xct i divide
         spl
         cma
         dac quo
dv5,     0      /remainder has sign of dividend
         isz divide
         xct i divide
         sma+cml-opr
         cma+cml-opr
         jms dv4
dv4,     0
         c11
         tad (1
         dac dvs
         tad dvd
         isz divide
         spl
         jmp i divide
         lac (-22
         dac dv1
         jmp dv2
dv 3,    lac dvd
         ral
         dac dvd
         tad dvs
         spl
         dac dvd
  
```

```
dv2,   lac quo
        ral
        dac quo
        isz dv1
        jmp dv3

        lac dv5
        ral
        lac dvd
        spl
        cma
        dac dvd
        lac dv4
        ral
        lac quo
        spl
        cma+c11-opr
        jmp i divide
```

start

TABLE A.9 CHARACTERISTICS OF INPUT-OUTPUT EQUIPMENT

Equipment (Device and Control)	Within Main Console					External to Main Console											
	Panels Reqd.	DC Power Dissipation (Watts)	Weight Added (Pounds)	Real Time Option Requirements				Maximum Distance from Console (Feet)	Size (Inches)			Weight (Pounds)	AC Power Dissipation				
				ID	IC	DS	IOS		PIC	IO ^R	S		Width	Depth	Height	KVA	Watts
Perforated- Tape Reader	1	20	10	0	18	1	1	1	4	20 1/4	12	8 3/4	30	0.132	125		
Perforated- Tape Punch	1	20	10	8	0	1	1	1	4	7 3/4	15 1/4	13	24 1/2	0.220	65		
Printer- Keyboard	1	20	10	5	5	2	2	2	4	21 1/4	20	14 3/4	55	0.220	65		
Visual CRT Display -30A,D	0	0	0	18	0	1	0	0	25	34	44	48	250	0.88	920		
Light Pen	1	<1	10	0	0	1	1	1	6	Approx. 7" long			1/8	0	0		
Relay Buffer	2	50	20	18	0	1	0	0	0	0	0	0	0	0	0		
High Speed Printer	0	0	0	18	0	2	2	2	25	66 3/4	31	61 1/4	1100	2.20	2300		
Card Reader	0	0	0	0	6	1	2	1	4	30	17 3/4	42	150	0.132	40		
Card Punch	0	0	0	18	0	1	1	1	25	40	26	49 1/2	678	1.10	1150		
Magnetic Tape System Type 54	1	20	10	18	18	4	2	1	15	28	22	69	550	1.32	1380		

APPENDIX 10

POWERS OF TWO

2^n	n	n^{-2}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.0625
32	5	0.03125
64	6	0.015625
128	7	0.0078125
256	8	0.00390625
512	9	0.001953125
1024	10	0.0009765625
2048	11	0.00048828125
4096	12	0.000244140625
8192	13	0.0001220703125
16384	14	0.00006103515625
32768	15	0.000030517578125
65536	16	0.0000152587890625
131072	17	0.00000762939453125
262144	18	0.000003814697265625
524288	19	0.0000019073486328125
1048576	20	0.00000095367431640625
2097152	21	0.000000476837158203125
4194304	22	0.0000002384185791015625
8388608	23	0.00000011920928955078125
16777216	24	0.000000059604644775390625
33554432	25	0.0000000298023223876953125
67108864	26	0.00000001490116119384765625
134217728	27	0.000000007450580596923808125
268435456	28	0.0000000037252902984619140625
536870912	29	0.00000000186264514923095703125
1073741824	30	0.000000000931322574615478515625
2147483648	31	0.0000000004656612873077392578125
4294967296	32	0.00000000023283064365386962890625
8589934592	33	0.00000000011641532182693481453125
17179869184	34	0.0000000000582076609134674072265625
34359738368	35	0.00000000002910383045673370361308125
68719476736	36	0.000000000014551915228366851806640625
137438953472	37	0.0000000000072759576141834259033203125
274877906944	38	0.00000000000363797880709171295166015625
549755813888	39	0.000000000001818989403545856475830078125
1099511627776	40	0.0000000000009094947017729282379150390625
2199023255552	41	0.00000000000045474735088646411895751953125
4398046511104	42	0.000000000000227373675443232059478759765625
8796093022208	43	0.000000000000113686837216160297393798828125
17592186044416	44	0.00000000000005684341886080801486968994140625
35184372088832	45	0.000000000000028421709430404007434844970703125
70368744177664	46	0.0000000000000142108547152020037174224853515625
140737488355328	47	0.00000000000000710542735760100185871124267578125
281474976710656	48	0.000000000000003552713678800500929355621337890625
562949953421312	49	0.0000000000000017763568394002504646778106689453125
1125899906842624	50	0.00000000000000088817841970012523233890533447265625
2251799813685248	51	0.000000000000000444089209850062616169452667236328125
4503599627370496	52	0.0000000000000002220446049250313080847263336681640625
9007199254740992	53	0.00000000000000011102230246251565404236316683458203125
18014398509481984	54	0.000000000000000055511151231257827021171513417041015625
36028797018963968	55	0.0000000000000000277555756156289135105907917085205078125
72057594037927936	56	0.0000000000000000138777878078145675521539585426025390625
144115188075855872	57	0.000000000000000006938893903907228377647697927130126953125
288230376151711744	58	0.0000000000000000034694469519536141888238489635650634765625
576460752303423488	59	0.00000000000000000173472347597680709441192448178253173828125
1152921504606846976	60	0.000000000000000000867361737988403547205962240891265869140625

