

PDP
4
MANUAL

PROGRAMMED DATA
PROCESSOR-4
MANUAL

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

Foreword

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. However, Appendix 4 explains the meaning and use of special characters used in the programming examples.

Copyright 1962 Digital Equipment Corporation

Table Of Contents

	Page
CHAPTER 1: SYSTEM DESCRIPTION	5
CHAPTER 2: ARITHMETIC AND CONTROL ELEMENT	11
Functions	11
Control States	15
Instructions	16
CHAPTER 3: INPUT-OUTPUT EQUIPMENT FUNCTIONS AND PROGRAMMING	25
Input-Output Commands	25
Device Selector	25
Information Collector	26
Information Distributor.....	28
Input-Output Skip Facility	28
Program Interrupt Control	28
Input-Output Status Instruction	29
Clock/Timer	29
Input-Output Devices	30
Precision CRT Display, Type 30A	30
Light Pen, Type 32	32
Precision CRT Display, Type 30D and Light Pen, Type 32	33
High Speed Analog-to-Digital Converter (Typical Input Device)	34
Low Speed Analog-to-Digital Converter (Typical Input Device)	35
Perforated-Tape Reader	36
Printer-Keyboad and Control, Type 65	38
Perforated-Tape Punch and Control, Type 75	42
Card Reader and Control, Type 41-4	44
Card Punch Control, Type 40-4	46
Automatic Line Printer and Control, Type 62	49
CHAPTER 4: THE INTERFACE ELECTRICAL CHARACTERISTICS	52
Appendix 1 Instruction Lists	57
Appendix 2 Codes	61
Appendix 3 Read-In Mode Sequence	65
Appendix 4 Assembly Program	67
Appendix 5 Multiply and Divide Subroutines.....	70
Appendix 6 Programming Aids	73
Appendix 7 Powers of 2	75



Typical PDP-4 System

CHAPTER 1

SYSTEM DESCRIPTION

Summary

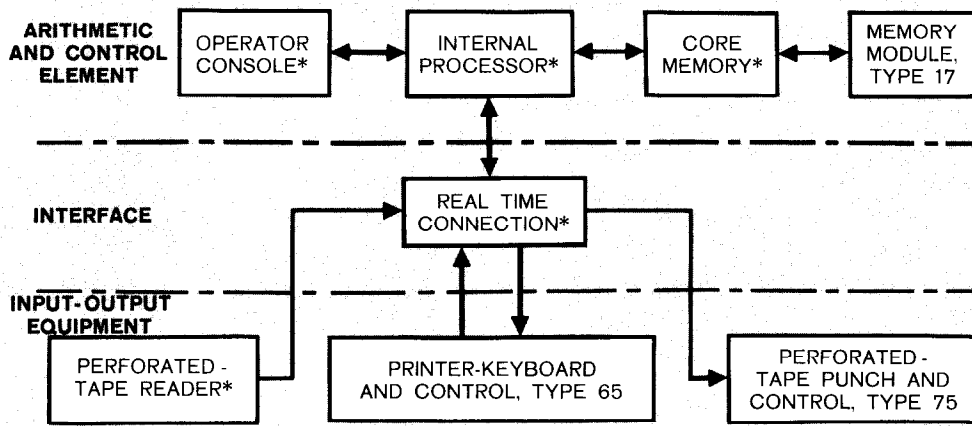
The Digital Equipment Corporation Programmed Data Processor-4 (PDP-4) is designed to be the control element in an information processing system. PDP-4 is a single address, parallel, binary machine with an 18-bit word length using 1's or 2's complement arithmetic. Standard features of the machine are stored program operation, a random access magnetic-core memory, a complete order code, and indirect addressing.

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



*Included in a
Standard PDP-4

Figure 1 — PDP-4 System with Real-Time Connection

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 optional Extended Arithmetic Control Unit, Type 22, gives PDP-4 a multiply, divide, and arithmetic shifting capability without the use of sub-routines.

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.

Memory capacities of from 1,024 to 32,768 words are available for PDP-4. 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. See Chapter 2 for detailed functions of the Arithmetic and Control Element.

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 Control, Type 75, and the Printer-

Keyboard and Control, Type 65. 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. See Chapter 3 for details of functions.

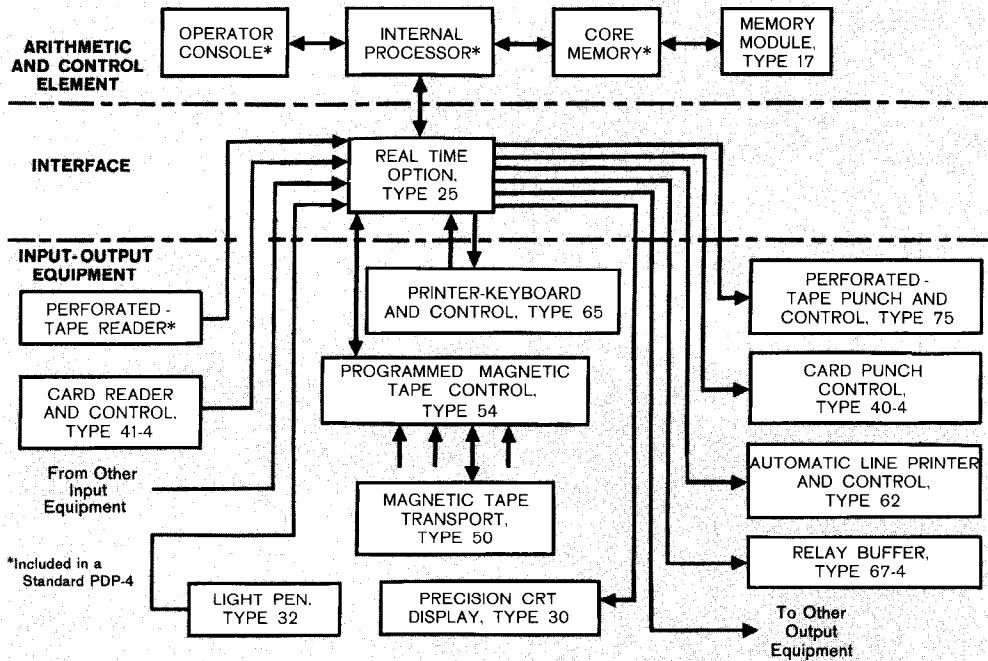
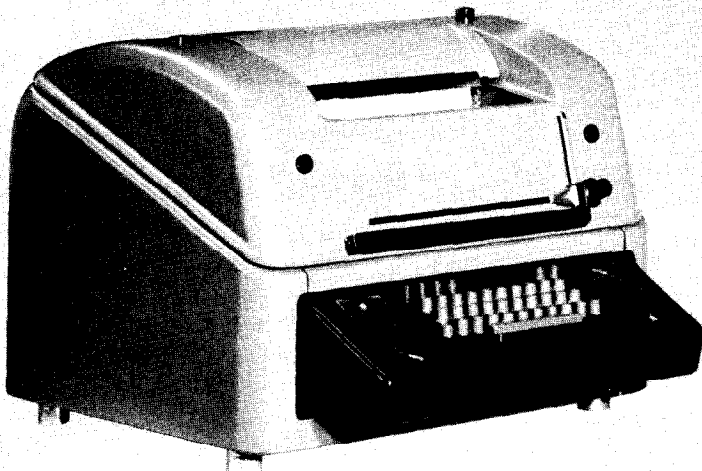
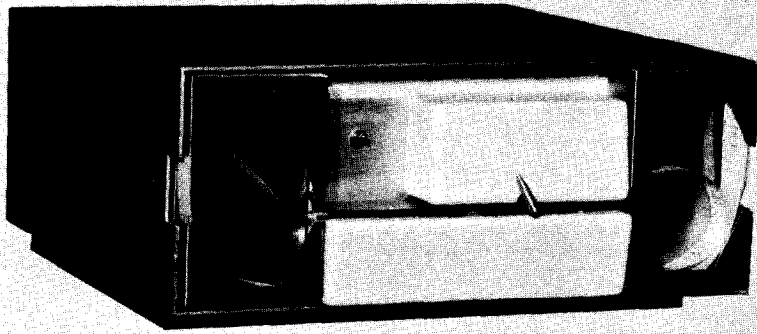


Figure 2 — PDP-4 System with Real-Time Option

THE DEVICE SELECTOR consists of decoding elements to select and establish the state of an 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; 8×18 bits of information may be collected, broken into variable-sized words.



The Perforated-Tape Reader (top) and Printer-Keyboard (bottom).

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 8×18 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 1. Eight skip conditions may be sampled.

THE PROGRAM INTERRUPT permits one of 11 lines (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; 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 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 PRECISION CRT DISPLAY, TYPE 30, displays data on a $9\frac{1}{4}$ " by $9\frac{1}{4}$ " area. Information is plotted point by point 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. Upon signal

from the Light Pen, the computer carries out previously programmed instructions. Requires Real-Time Option.

THE 18-BIT RELAY BUFFER, TYPE 67-4, 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 PROGRAMMED MAGNETIC TAPE CONTROL, TYPE 54, controls up to four Magnetic Tape Transports, 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 TRANSPORTS, TYPE 50, are used with the Programmed Magnetic Tape Control, Type 54.

THE AUTOMATIC LINE PRINTER AND CONTROL, TYPE 62, 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 41-4, operates at a rate of up to 200 cards per minute. Cards are read column by column. Column information may be read in alpha-numeric or binary mode. The alpha-numeric 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 40-4, enables the operation of a standard IBM Type 523 Summary Punch with PDP-4. Cards are punched row by row 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.

PROGRAMMING AIDS

Several programs are supplied with each PDP-4 to assist the programmer in routine tasks. They include: The PDP-4 Assembly Program, the DDT-4 debugging tape, double-precision floating point routines, maintenance routines, a tape reproducer, punch routines, an octal debugging routine, an algebraic compiler, and a floating point functions program which will enable various functions, such as double precision floating-point sine, to be computed. See Appendix 6.

CHAPTER 2

ARITHMETIC AND CONTROL ELEMENT

In this chapter the functions of the Arithmetic and Control Element are described in detail. The operations of the machine instructions are explained and listed.

Functions

INTERNAL PROCESSOR

The Internal Processor performs arithmetic operations, controls memory access, and handles information entering and leaving the machine. It consists of the Information Processor Control, which oversees all activities, and six registers: Accumulator, Link, Memory Buffer, Memory Address, Instruction, and Program Counter. The elements of the Internal Processor are shown within the broken line in Figure 3.

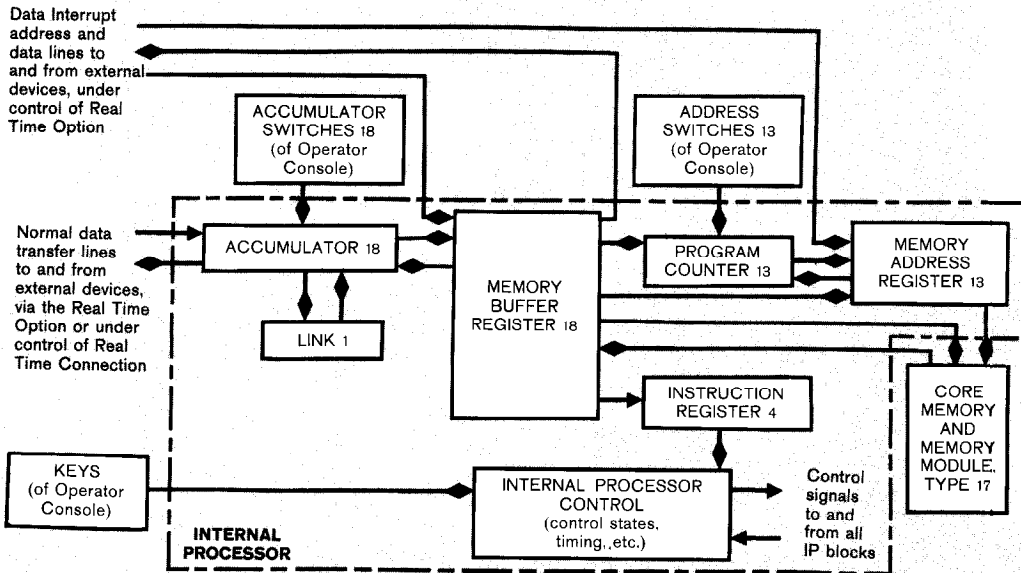


Figure 3 – Arithmetic and Control Element

ACCUMULATOR (AC): Arithmetic operations are performed in this 18-bit register. The AC may be cleared and complemented. Its contents may be rotated right or left with the Link. The contents of the Memory Buffer may be added to the contents of the AC with the result left in the AC. The contents of both these registers may be combined by the logical operations AND and Exclusive OR, the result remaining in the AC. The Inclusive OR may be formed between the AC and the Accumulator Switches on the Operator Console (see below), and the result left in the AC.

The Accumulator also acts as an input-output register. Under normal operation all information transfers between core memory and an external device must pass through the Accumulator.

LINK (L): This is a one-bit register used to extend the arithmetic facility of the Accumulator. In 1's complement arithmetic, the Link is an overflow indicator; in 2's complement it functions as a carry register. The Link may be cleared and complemented and its state sensed independent of the AC. It is included with the AC in rotate operations.

MEMORY BUFFER (MB): All information transferred between Core Memory and the AC, Instruction Register, or Program Counter passes through the MB. Information is read from a memory cell into the MB and rewritten into the cell in one cycle time (8 microseconds). Instructions are brought from memory into the MB to be decoded. The MB serves also as a buffer for information transferred between Core Memory and an external device in a Data Interrupt. The contents of the MB may be incremented by one.

MEMORY ADDRESS REGISTER (MA): The address of the Core Memory cell currently being accessed is contained in the 13-bit MA. Information may enter the MA from the MB, Program Counter, or from an external device operating in a Data Interrupt.

INSTRUCTION REGISTER (IR): This is a 4-bit register which contains the operation code of the instruction currently being performed by the computer. Information enters the IR from the MB.

PROGRAM COUNTER (PC): The program sequence, that is, the order in which instructions are performed, is determined by the PC. This 13-bit register contains the address of the memory cell from which the next instruction will be taken. Information may enter the PC from the MB, MA, or the Address Switches of the Operator Console.

MEMORY

The memory contains stored information for processing, and the instructions of the program being run. Memory capacities of from 1,024 to 32,768 words are available in PDP-4. Standard models PDP-4A and PDP-4B come with 1024-word and 4096-word memories, respectively. The two models are identical in all other respects. 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 an 8192-word capacity. With the addition of the Magnetic Core Memory Extension Control Type 16, memory modules may be added to build a memory of 32,768 words. Further increase in storage capacity can be gained by adding the Magnetic Drum System Type 24, available in three capacities: 16,384, 32,768, and 65,536 words.

OPERATOR CONSOLE

The Operator Console contains all the switches and controls necessary to run the machine, and lights which indicate the current status of the Internal Processor. The functions of the lights and controls are described in the following tables.

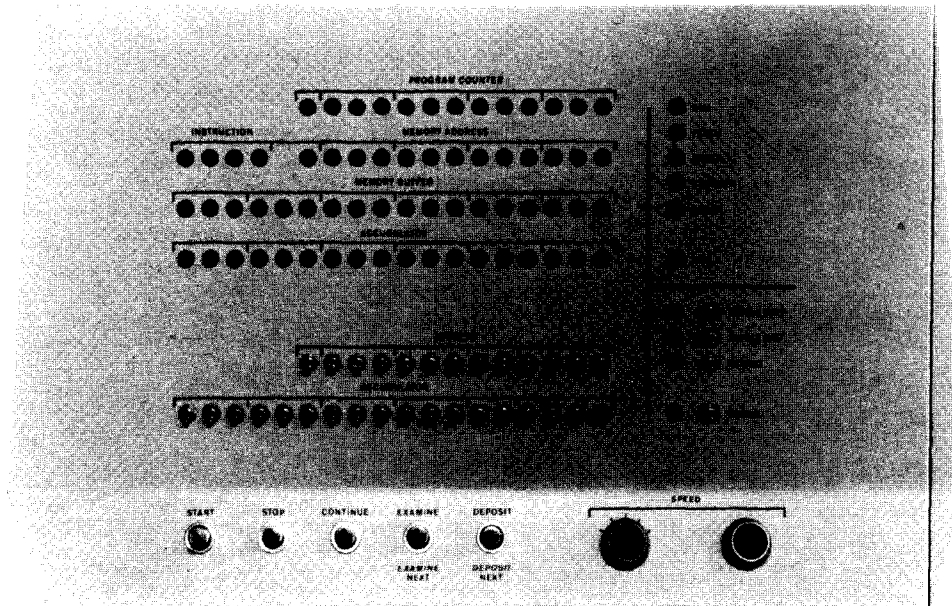


Figure 4 — Operator Console

Console Switches	Function
ADDRESS	A group of 13 switches which establishes the memory address for the START, EXAMINE, and DEPOSIT operations.
ACCUMULATOR	A group of 18 switches, the setting of which determines the word to be placed in memory by the DEPOSIT and DEPOSIT NEXT operations, or to be placed in the AC under program control.
POWER	Controls the primary power to the computer and all external devices attached to it.
SINGLE STEP	Causes the computer to stop at the completion of each memory cycle. Repeated operation of CONTINUE while this switch is on will step the program one cycle at a time.
SINGLE INSTRUCTION	Causes the computer to stop at the completion of each instruction. Repeated operation of CONTINUE while this switch is on will step the program one instruction at a time. When both switches are on, SINGLE STEP takes precedence over SINGLE INSTRUCTION.
REPEAT	Causes the operations initiated by pressing CONTINUE, EXAMINE NEXT, or DEPOSIT NEXT, to be repeated as long as the key is held on. The rate of repetition is controlled by the setting of the SPEED knobs.
SPEED	Two controls that vary the REPEAT interval from approximately 40 microseconds to 8 seconds. The left knob is a five-position coarse control, the right knob a continuously variable fine control. For both knobs, slowest speed is obtained in extreme left position.

Console Light	Indication
ACCUMULATOR	The contents of the AC.
MEMORY BUFFER	The contents of the MB.
LINK	The contents of the Link.
MEMORY ADDRESS INSTRUCTION	The contents of the MA register.
PROGRAM COUNTER	The contents of the IR.
RUN	The contents of the PC.
FETCH, DEFER, EXECUTE, BREAK	The computer is executing instructions.
	The primary control state of the next memory cycle.

Console Key	Function
START	Starts the processor. The first instruction is taken from memory cell specified by the setting of the ADDRESS switches. The START operation clears the AC and Link, and turns off the Program Interrupt.
STOP	Stops the processor at the completion of the memory cycle in progress at the time of key operation.
CONTINUE	Causes the computer to resume operation from the point at which it was stopped by the last previous operation of STOP or one of the EXAMINE or DEPOSIT keys. Besides the normal off and momentary on positions, CONTINUE has a latched on position obtained by raising the key instead of depressing it.
EXAMINE	Places the contents of the memory cell specified by the ADDRESS switches in the AC and MB. The contents of the ADDRESS switches appear in the MA. The PC contains the address of the next cell.
EXAMINE NEXT	Places the contents of the cell specified by the PC in the MB and AC. The C(PC) are incremented by one. The MA contains the address of the register examined.
DEPOSIT	Deposits the contents of the AC switches in the memory cell specified by the ADDRESS switches. The C(AC switches) remain in the AC and MB. The contents of the ADDRESS switches appear in the MA. The PC contains the address of the next cell.
DEPOSIT NEXT	Deposits the contents of the AC switches in the memory cell specified by the PC. The C(PC) are then incremented by one. The C(AC), C(MB), and C(MA) are the same as for DEPOSIT.

Control States

The PDP-4 operates in one of four primary control states during a memory cycle: Fetch, Defer, Execute, or Break. The next control state is established at the completion of the current one. All states except Break are determined by the instructions themselves.

FETCH: A new instruction is obtained when this state occurs. The contents of the memory cell specified by the PC are placed in the MB,

and the instruction part (bits 0-4) of this word are placed in the IR. The C(PC) are then incremented by one.

If a two-cycle instruction is fetched, the following control state will be either Defer or Execute. If a one-cycle instruction is fetched, the operations specified will be performed during the last part of the Fetch cycle. The next state will be Fetch.

DEFER: When bit 4 of a memory reference instruction is a 1, the Defer state is entered to perform the indirect addressing. The process of indirect addressing is often referred to as deferring, in the sense that access to the operand is deferred once to another memory cell. This is why the primary control state in which this operation is performed is called Defer. Bit 4 of a memory reference instruction is referred to interchangeably as the Indirect or the Defer Bit.

EXECUTE: This state is established only when a memory reference instruction is being performed. The contents of the memory cell addressed are brought into the MB, and the operation specified by the C(IR) is performed.

BREAK: When this state is established, the sequence of instructions is broken for a Data Interrupt or a Program Interrupt. In both cases, the break occurs only at the completion of the current instruction.

The Data Interrupt allows information to be transferred between memory and an external device; when this transfer has been completed, the program sequence is resumed from the point of the break. The Program Interrupt causes the sequence to be altered. The C(PC) and the C(L) are stored in location 0000 and the program continues from location 0001.

Instructions

The instruction code is specified by bits 0-3 of a word. There are two types of instructions: Memory Reference and Augmented.

MEMORY REFERENCE INSTRUCTIONS

The bit assignment of the memory reference instruction is shown in Figure 5. Bits 0-3 determine the operation to be performed. Bits 5-17 specify the address of the memory cell containing the operand. If bit 4 is a 1, then indirect addressing occurs. In the following discussion, *i* is the mnemonic symbol used to indicate indirect addressing.

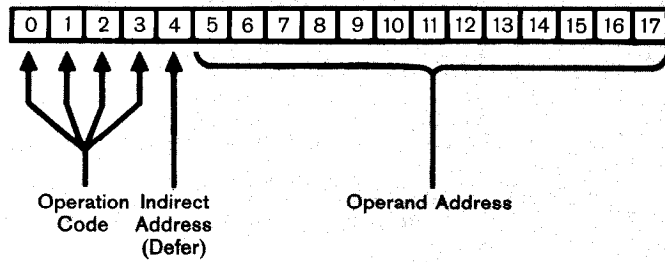


Figure 5 — Memory reference instruction format

INDIRECT ADDRESSING

When indirect addressing is specified, the address part (bits 5-17) of a memory reference instruction is interpreted as the address of a cell containing not the operand, but the address of the operand. Consider the instruction `add A`. Normally, `A` is interpreted as the address of the cell containing the quantity to be added to the AC. Thus, if cell 100 contains the number 576, the instruction

`add 100`

will cause the quantity 576 to be added to the AC. Now suppose that cell 576 contains the number 1135. The instruction

`add i 100`

(where `i` signifies indirect addressing) will cause the computer to take the number 576, which is in cell 100, as the effective address of the instruction, and the number in cell 576 as the operand. Hence this instruction will result in the quantity 1135 being added to the AC.

If, when indirect addressing is indicated, the memory cell addressed by the instruction is one of those in locations 10-17, the contents of that cell are incremented by one and the result taken as the effective address. This feature is called auto-indexing. If memory cell 12 contains the number 200, the instruction

`add i 12`

will cause the number in cell $200 + 1$ to be added to the AC.

1'S COMPLEMENT ARITHMETIC

When two numbers are added together in 1's complement arithmetic (see `add` instruction in following table), a 1 carried out of the high-order position will be added to the low-order digit, as follows:

$$\begin{array}{r}
 110101001100011 \\
 011001010111101 \\
 \hline
 1\ 001110100100000 \\
 \xrightarrow{\hspace{1.5cm}} 1 \\
 001110100100001
 \end{array}$$

Since bit 0 of a word is used for the sign of a number, the largest positive number that can be represented is $2^{17} - 1$. If, in 1's complement addition, the addends are of like sign and the sign of the sum is different, overflow is said to have occurred and the Link is set to 1.

2'S COMPLEMENT ARITHMETIC

In 2's complement addition (see tad instruction), a carry out of the high-order bit is not added into the low order position. Instead, if a carry occurs, the Link is complemented. The signs of the addends and sum are not examined. Two's complement addition is used primarily in multiple precision arithmetic.

All memory reference instructions require an Execute cycle (see Control States above) to transfer data between Core Memory and the MB and execute the instruction. When indirect addressing is specified, an extra cycle is required to determine the effective address. The jmp instruction, while it requires an address, does not require an operand; an Execute cycle is thus not needed, and the instruction is performed in only one cycle.

MEMORY REFERENCE INSTRUCTIONS

Explanation of Special Terms

C(A)	contents of A	∇	exclusive OR
A => B	A replaces B	\vee	inclusive OR
Y ₁₋₄	bits 1 - 4 of Y	\wedge	AND
Y _j	a given bit in Y	\bar{A}	1's complement of A

MNEMONIC SYMBOL	OCTAL CODE (BITS 0-3)	TIME (μ sec)	OPERATION
lac Y	20	16	Load AC. The C(Y) are loaded into the AC. The previous C(AC) are lost. C(Y) => C(AC).
dac Y	04	16	Deposit AC. The C(AC) are deposited in the memory cell at location Y. The previous C(Y) are lost; the C(AC) are unchanged. C(AC) => C(Y).
dzm Y	14	16	Deposit Zero in Memory. Zero is deposited in memory cell Y. The original C(Y) are lost. The AC is unaffected by this operation. 0 => C(Y).
add Y	30	16	Add (1's complement). 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 C(Y) are unchanged. The Link is set to 1 on overflow. (See text). C(Y) + C(AC) => C(AC).

MNEMONIC SYMBOL	OCTAL CODE (BITS 0-3)	TIME (μ sec)	OPERATION															
tad Y	34	16	Two's complement Add. The C(Y) are added to the C(AC) in 2's complement arithmetic. The result is left in the AC and the original C(AC) are lost. The C(Y) are unchanged. A carry out of the 0 bit complements the Link. $C(Y) + C(AC) \Rightarrow C(AC)$.															
xor Y	24	16	Exclusive OR. The logical operation Exclusive OR is performed between the C(Y) and the C(AC). The result is left in the AC and the original C(AC) are lost. The C(Y) are unchanged. Corresponding bits are compared independently. $C(Y_j) \vee C(AC_j) \Rightarrow C(AC_j)$. <p style="text-align: center;">Example</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>$C(AC)_j$ original</th> <th>$C(Y)_j$</th> <th>$C(AC)_j$ final</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	$C(AC)_j$ original	$C(Y)_j$	$C(AC)_j$ final	0	0	0	0	1	1	1	0	1	1	1	0
$C(AC)_j$ original	$C(Y)_j$	$C(AC)_j$ final																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
and Y	50	16	AND. The logical operation AND is performed between the C(Y) and the C(AC). The result is left in the AC, and the original C(AC) are lost. The C(Y) are unchanged. Corresponding bits are compared independently. $C(Y_j) \wedge C(AC_j) \Rightarrow C(AC_j)$ <p style="text-align: center;">Example</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>$C(AC)_j$ original</th> <th>$C(Y)_j$</th> <th>$C(AC)_j$ 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>	$C(AC)_j$ original	$C(Y)_j$	$C(AC)_j$ final	0	0	0	0	1	0	1	0	0	1	1	1
$C(AC)_j$ original	$C(Y)_j$	$C(AC)_j$ final																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
sad Y	54	16	Skip if AC is Different from Y. The C(Y) are compared with the C(AC). If the numbers are the same, the computer proceeds to the next instruction. If the numbers are different, the next instruction is skipped. The C(AC) and the C(Y) are unchanged. $\text{If } C(AC) \neq C(Y) \text{ then } C(PC) + 1 \Rightarrow C(PC)$.															

MNEMONIC SYMBOL	OCTAL CODE (BITS 0-3)	TIME (μ sec)	OPERATION
isz Y	44	16	Index and Skip if Zero. The C(Y) are incremented by one in 2's complement arithmetic. If the result is 0, the next instruction is skipped. If not, the computer proceeds to the next instruction. The C(AC) are unaffected. C(Y) + 1 => C(Y). If result = 0, C(PC) + 1 => C(PC).
jmp Y	60	8	Jump to Y. The next instruction to be executed is taken from memory cell Y. Y => C(PC).
jms Y	10	16	Jump to Subroutine. The C(PC) and the C(L) are deposited in memory cell Y. The next instruction is taken from cell Y + 1. C(L) => C(Y ₀). 0 => C(Y ₁₋₄). C(PC) => C(Y ₅₋₁₇). Y + 1 => C(PC).
cal	00	16	Call Subroutine. The address portion of this instruction is ignored. The action is identical to jms 20. The instruction cal i is equivalent to jms i 20.
xct Y	40	8 + time of instruction being executed	Execute. The instruction in memory cell Y will be executed. The computer will act as if the instruction located in Y were in the place of the xct.

AUGMENTED INSTRUCTIONS

None of the augmented instructions require a memory reference. Bits 4-17 of an augmented instruction are used to specify operations, many of which may be combined in a single instruction. There are three classes of augmented instructions:

- a. Operate class: includes operations on the AC and Link, the skip group, and the halt instruction.
- b. The special instruction, law.
- c. Input-output transfer class: includes all the instructions which initiate transfers of information between the Internal Processor and an external device and those that sense the status of the devices.

OPERATE CLASS

The instructions of the Operate class require one cycle for their execution. The octal code (bits 0-3) for this class is 74. The operations specified by bits 4-17 are called micro-instructions. The functions of each micro-instruction are described in the following table. The Event Time indicates when the operation is performed in the course of the cycle. Times 0, 1, and 2 occur in that order in the latter part of the cycle.

Except for the restrictions indicated at the end of the table, micro-instructions may be combined in a single instruction. The bit assignment of the Operate class micro-instructions is shown in Figure 6.

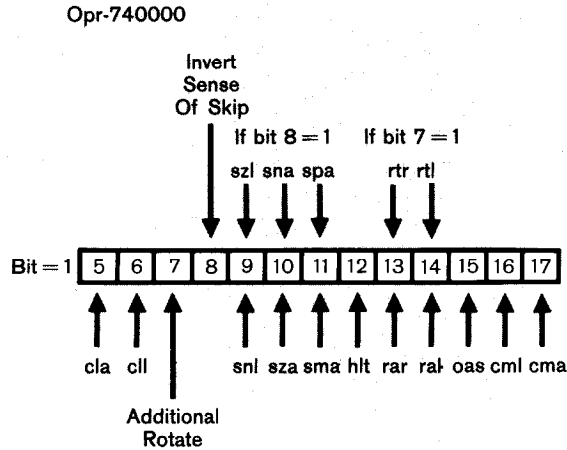


Figure 6 — Operate class instruction — bit assignment

MNEMONIC SYMBOL	OCTAL CODE	EVENT TIME	OPERATION
opr	740000		Operate. Indicates the Operate class. When used alone, performs no operation; the computer proceeds to the next instruction.
cla	750000	2	Clear AC. The AC is cleared to 0. $0 \Rightarrow C(AC)$.
cma	740001	3	Complement AC. Each bit of the AC is complemented. $\overline{C(AC)} \Rightarrow C(AC)$.
cll	744000	2	Clear Link. Link is set to 0. $0 \Rightarrow C(L)$.
cml	740002	3	Complement Link. $\overline{C(L)} \Rightarrow C(L)$.

MNEMONIC SYMBOL	OCTAL CODE	EVENT TIME	OPERATION
ral	740010	3	Rotate AC Left. The C(AC) and the C(L) are rotated left one place. $C(AC_j) \Rightarrow C(AC_{j-1})$ $C(AC_0) \Rightarrow C(L)$. $C(L) \Rightarrow C(AC_{17})$
rtl	742010	2, 3	Rotate Two places Left. Equivalent to two successive ral's.
rar	740020	3	Rotate AC Right. The C(AC) and the C(L) are rotated one place right. $C(AC_j) \Rightarrow C(AC_{j+1})$ $C(L) \Rightarrow C(AC_0)$ $C(AC_{17}) \Rightarrow C(L)$
rtr	742020	2, 3	Rotate Two Places Right. Action taken is equivalent to two successive rar's.
oas	740004	3	OR AC Switches. The Inclusive OR of the C(AC) and the C(AC switches) is placed in the AC. A switch up is interpreted as a 1. $C(AC \text{ Switches}) \vee C(AC) \Rightarrow C(AC)$.

Example

$C(AC)_j$ original	$C(Y)_j$	$C(AC)_j$ final
0	0	0
0	1	1
1	0	1
1	1	1

sma	740100	1	Skip if Minus AC. If the AC is negative, the next instruction is skipped. If $AC_0 = 1$, then $C(PC) + 1 \Rightarrow C(PC)$.
spa	741100	1	Skip if Plus AC. If the AC is positive, the next instruction is skipped. If $AC_0 = 0$, then $C(PC) + 1 \Rightarrow C(PC)$.
sza	740200	1	Skip if Zero AC. If C(AC) are 0, the next instruction is skipped. If $C(AC) = 0$, then $C(PC) + 1 \Rightarrow C(PC)$.
sna	741200	1	Skip if Non-zero AC. If $C(AC) \neq 0$, then $C(PC) + 1 \Rightarrow C(PC)$.

MNEMONIC SYMBOL	OCTAL CODE	EVENT TIME	OPERATION
snl	740400	1	Skip if Non-zero Link. If C(L) is 1, the next instruction is skipped. If C(L) ≠ 0, then C(PC) + 1 => C(PC).
szl	741400	1	Skip if Zero Link. If C(L) = 0, then C(PC) + 1 => C(PC).
hlt	740040	immediately after the completion of the cycle.	Halt. Stops the computer.

If skips are combined in a single instruction, the Inclusive OR of the conditions to be met will determine the skip. For instance, if both *sza* and *snl* are indicated (octal code 740600), the next instruction will be skipped if either the AC is zero or the Link is non-zero, or both.

If *ral* or *rar* is specified, *cma*, *cml*, *oas* may not be specified, and conversely. If *rtl* or *rtr* is specified, *cma*, *cml*, *cla*, *cll*, *oas* may not be specified, and conversely.

THE INSTRUCTION, law

The octal code for this instruction is 760000. Bits 5-17 are used to specify a quantity to be placed in the AC. The effect of the law instruction is to place itself in the AC.

law Y	76	8 μsec	Load AC With law Y.
-------	----	--------	---------------------

INPUT-OUTPUT TRANSFER CLASS

The instructions in this class are used to effect information transfers between the Internal Processor and external devices, via the Interface.

iot	760000	8 μsec	Input-Output Transfer. Bits 4-13 of an iot instruction determine the device and sub-device to be selected. The presence of a 1 in bit 14 will cause the AC to be cleared at Event time 1. Bits 15-17 determine when pulses are to be sent to the selected device.
-----	--------	--------	---

The bit assignment of the iot instruction is shown in Figure 7. The instructions of the iot class are described in Chapter 3.

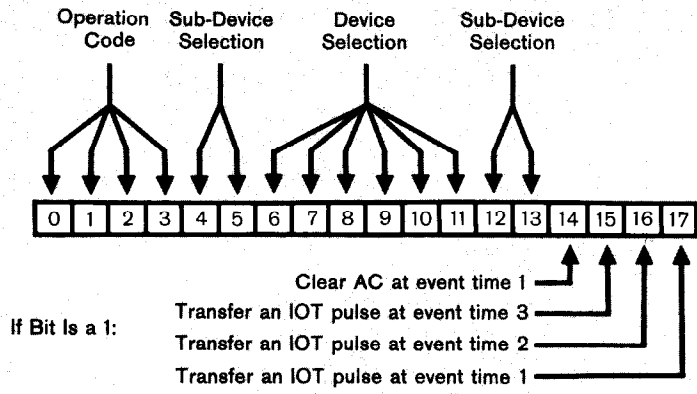


Figure 7 – Bit assignment for input-output transfer instruction (iot)

CHAPTER 3

INPUT-OUTPUT EQUIPMENT FUNCTIONS AND PROGRAMMING

PDP-4 is capable of operating with the ten input-output devices described in Chapter 1 and with a variety of others. 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 (see Figure 8).

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 (DS)

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 (e.g. asks for a line of perforated tape to be read and assembled into a word, 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 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 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 been completed; 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 (see below). Flags may be read into the AC using the iors (in-out read status) instruction. Most flags are connected to the Program Interrupt (see below).

The Device Selector selects an input-output device or subdevice according to the address code of the device in bits 4-13 in the iot instruction. It then generates IO pulses at event times 0, 1, and 2 if the appropriate micro-instruction code bits are present in bits 17, 16, and 15. Pulse iot 0 occurs near the end of an iot instruction, followed by iot 1 in 2 microseconds. Pulse iot 2 occurs at the beginning of the next instruction, 1.2 microseconds after iot 1. This timing enables one iot instruction to perform multiple operations.

INFORMATION COLLECTOR (IC)

The Information Collector enables information to be collected from eight 18-bit word input devices. The AC must contain 0 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 0)
- 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)

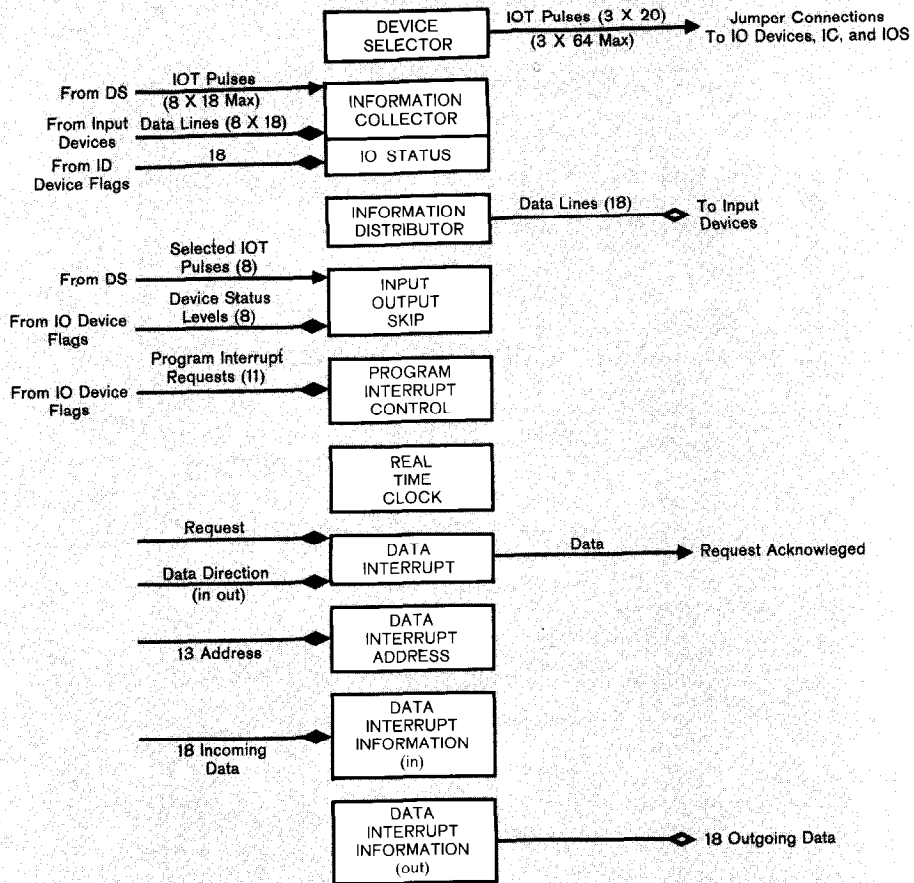


Figure 8 — Real Time Option, Type 25

INFORMATION DISTRIBUTOR (ID)

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:

iac Y	Load the AC with C(Y)
iot	Clear selected output register to 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 microprogrammed together in one instruction)

INPUT-OUTPUT SKIP FACILITY (IOS)

The Input-Output Skip facility enables the program to skip (or branch) according to various external device 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 (PIC)

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 interrupt program will then 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

INPUT-OUTPUT STATUS INSTRUCTION

The iors (in-out read status) instruction, 700314, enables the status of all IO devices to be read into the AC and sampled. Various IO device states are indicated by the presence of a 1 or 0 in the bit positions allocated for that device (see Figure 9).

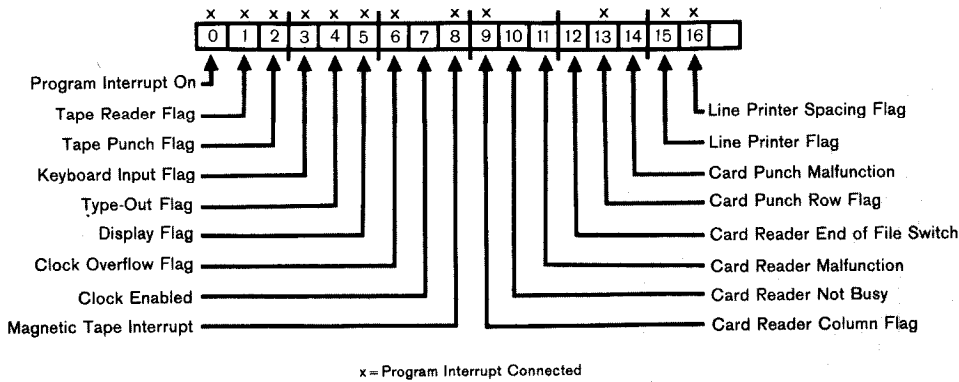


Figure 9 — Input-Output Status instruction, bit assignment

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 contents of memory cell 7 are 0 after the addition, the Clock flag is set to 1, which initiates a Program Interrupt if the Interrupt is on. 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 after a timed interval.

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 Control, and the Printer-Keyboard and Control. All devices except the Perforated-Tape Reader are optional. This section is arranged in the order of increasing complexity of connection.

PRECISION CRT DISPLAY, TYPE 30A

Data points are displayed on a 9¼ inch by 9¼ inch area. Information is plotted point by point 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, or every 50 microseconds.

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.

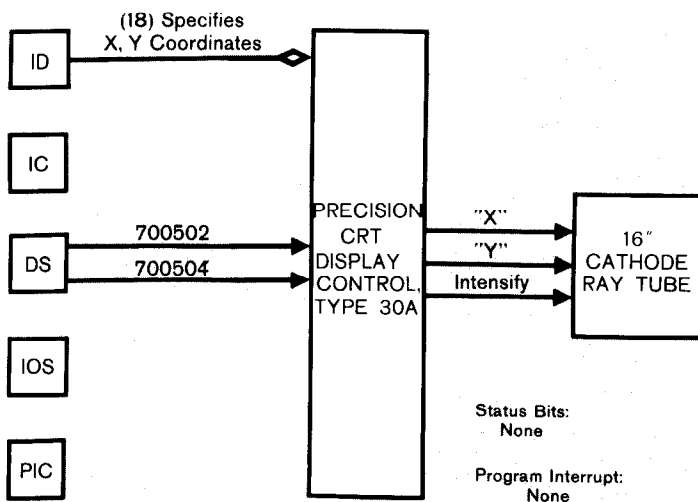


Figure 10 — Precision CRT Display, Type 30A programming logic

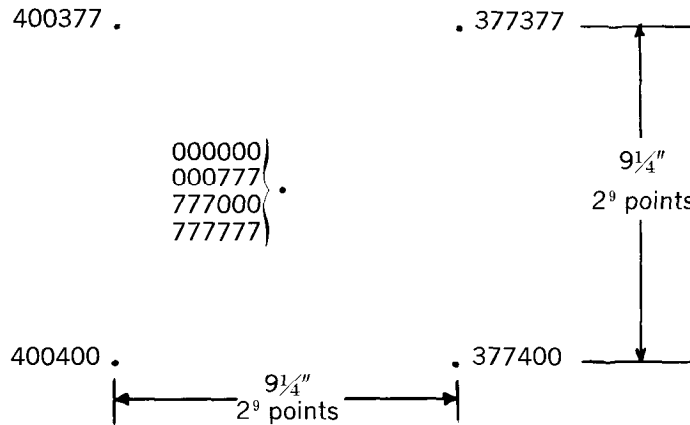
The CRT, Type 30A is selected when the numbers 0 and 5 (octal) are specified in bits 8 and 9 respectively, of the iot instruction. The display commands are:

dls — 700506 — 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:



A program sequence is given in PDP-4 Assembly 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, bits 9-17 y.
40/ lac 10 /place xy co-ordinate in ac
        /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.

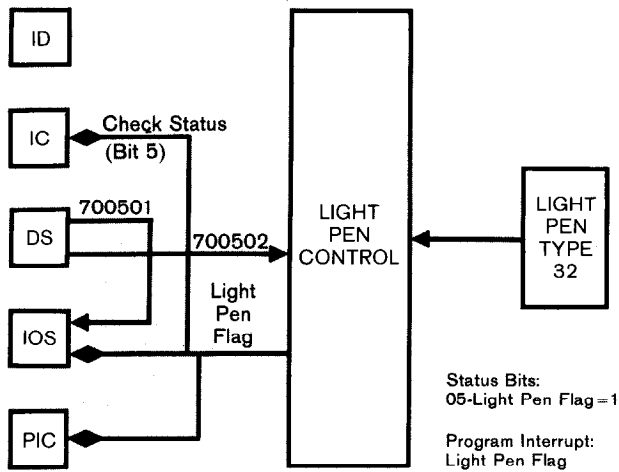


Figure 11 — Light Pen programming logic

PRECISION CRT DISPLAY, TYPE 30D AND LIGHT PEN, TYPE 32

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, AC₈₋₁₇.

The instructions are:

- dsf – 700501 – Skip if the Display flag is a 1. The Display Flag is set to 1 when the Light Pen senses light.
- dcf – 700601 – Clear the Display flag.
- dxi – 700506 – Load the C(XB) with C(AC₈₋₁₇).
- dyl – 700606 – Load the C(YB) with C(AC₈₋₁₇).
- dxs – 700546 – Load the C(XB) with C(AC₈₋₁₇). Plot the point: C(XB), C(YB).
- dys – 700646 – Load the C(YB) with C(AC₈₋₁₇). Plot the point: C(XB), C(YB).
- dlb – 700706 – Load the Brightness Register with AC bits 15-17. The bits of AC specify the brightness of the points displayed. Clear the Display flag.

- 700502 – Clear XB.
- 700504 – C(SB) ∨ C(AC) => C(XB). Display a point.
- 700602 – Clear YB.
- 700604 – C(YB) ∨ C(AC) => C(YB). Display a point.

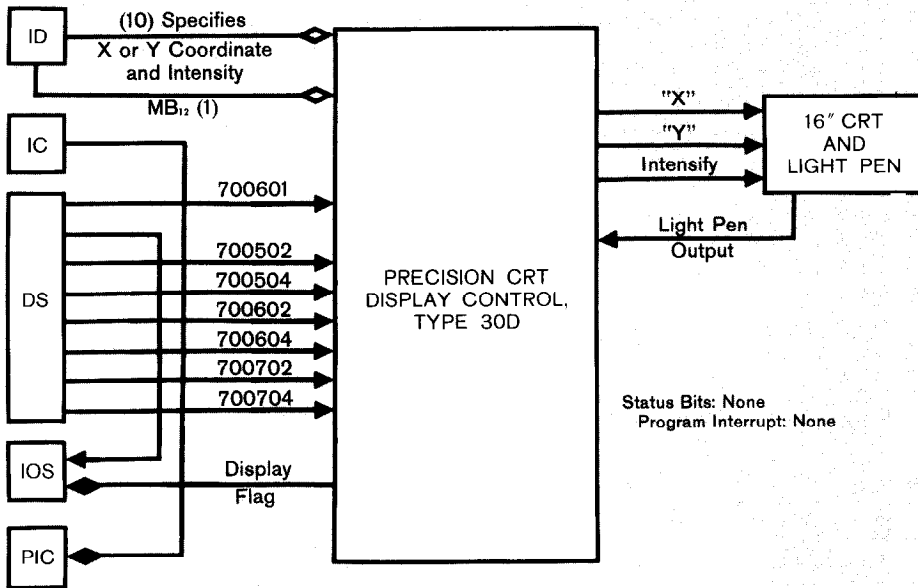
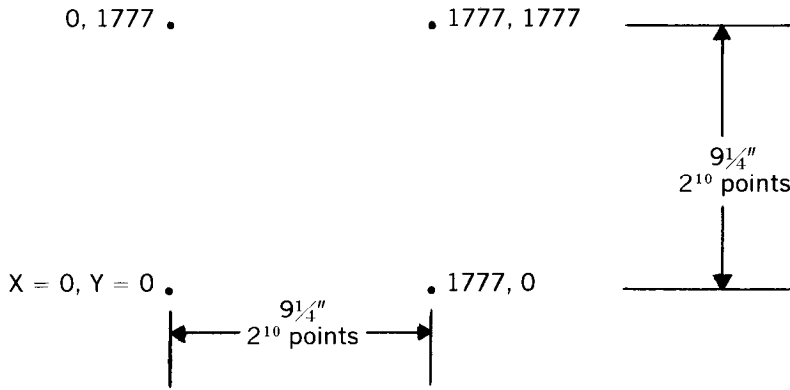


Figure 12 – Precision CRT Display, Type 30D, and Light Pen, Type 32

The Display flag is connected to the Program Interrupt and to bit 5 of the iors instruction. The co-ordinates of the corners 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

```

HIGH SPEED 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 micro-instruction starts the converter. In a period of 2 microseconds the converter will form an 8-bit number proportional to the analog input.
- 701104 – C(A-D converter) \vee C(AC) \Rightarrow A(AC).

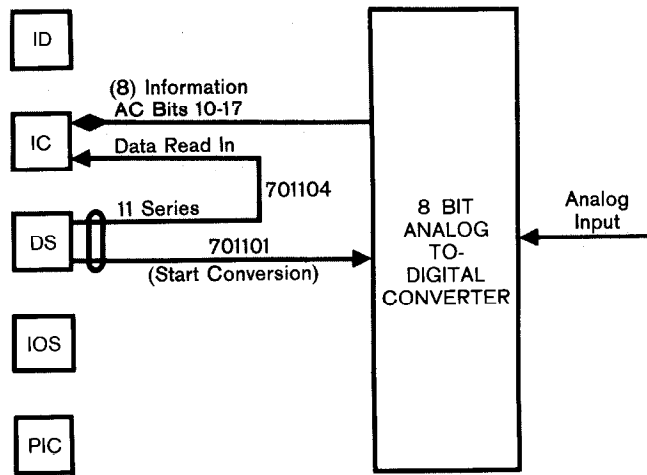


Figure 13 — High-speed analog-to-digital converter programming logic

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                           /places sample in AC
     dac 10                         /deposit result

```

LOW SPEED ANALOG-TO-DIGITAL CONVERTER (TYPICAL INPUT DEVICE)

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 micro-instruction which clears the converter flag, and $C(\text{converter buffer}) \vee C(\text{AC}) \Rightarrow C(\text{AC})$.

The converter flag might connect to the Program Interrupt.

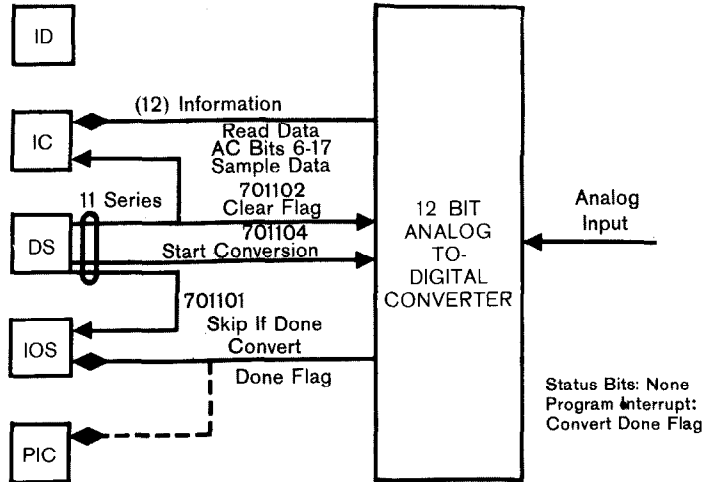


Figure 14 — Slow-speed analog-to-digital converter programming logic

PERFORATED-TAPE READER

The Tape Reader senses 5-, 7-, or 8-hole perforated-paper (or Mylar) tape photoelectrically at 300 characters (or lines) per second. The Reader control requests Reader movement, assembles data from the Reader into a Reader Buffer (RB), and signals the computer when incoming data 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 one or three lines from the tape. The $C(\text{RB})$ can be read into the AC. The Reader flag becomes a 1 when a character or word has been assembled in the RB.

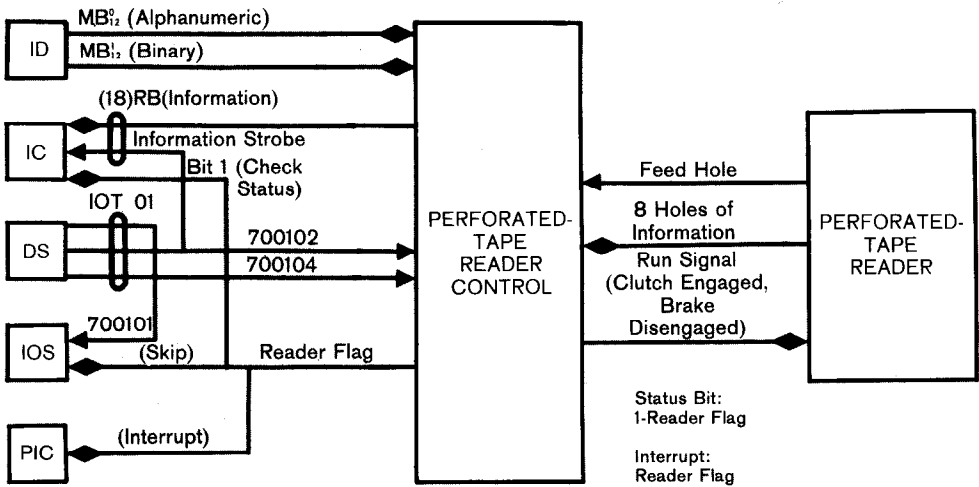


Figure 15 — Perforated —Tape Reader programming logic

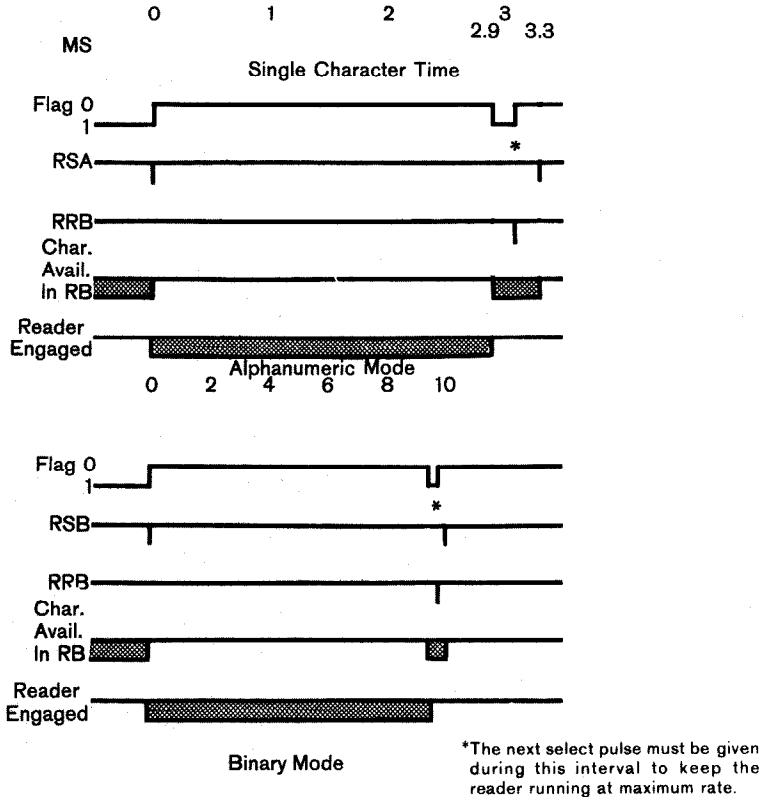


Figure 16 — Perforated-Tape Reader timing

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. The first, second, and third six-bit characters are the left, middle, and right thirds, respectively, of the 18-bit word. The reader commands, iot select series 01, are:

- rsf — 700101 — Skip if Reader flag is a 1, i.e., character or word present.
- 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.
- rrb — 700112 — Read RB. Clear the Reader flag, and transfer the contents of RB to the AC.
- rcf — 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 computation time are available 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/	rsa	/select reader alphanumeric
	rsf	/begin loop to look for character arrival
	jmp 41	/end loop to look for arrival
	rrb	/fetch character from reader buffer

By changing instruction 40 to 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 2, 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 17 illustrates the current pattern produced by the binary code 10110.

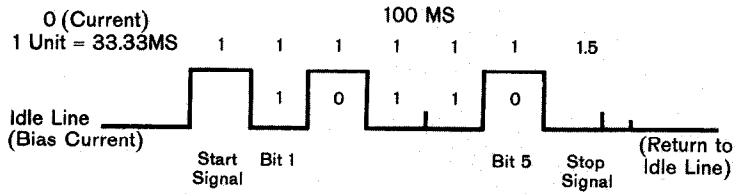


Figure 17 — Teletype timing of information code 10110

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 1, 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, i.e., character present.
- krb — 700312 — Read Keyboard buffer. Clear the Keyboard flag. $C(KB) \Rightarrow C(AC)$
- 700302 — Clear the Keyboard flag. $C(KB) \vee C(AC) \Rightarrow C(AC)$

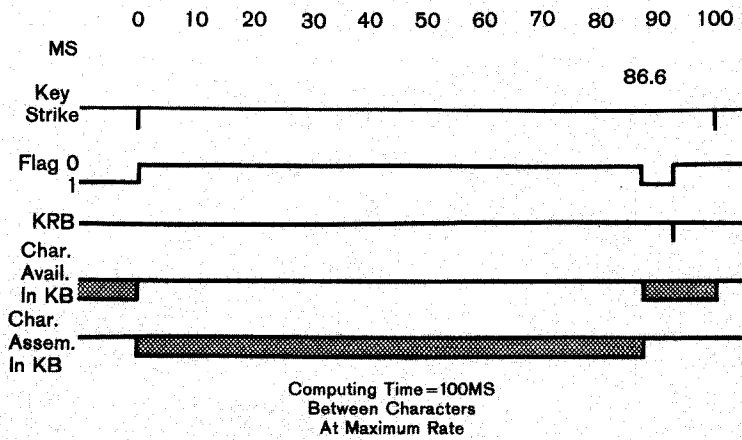


Figure 18 — Keyboard timing

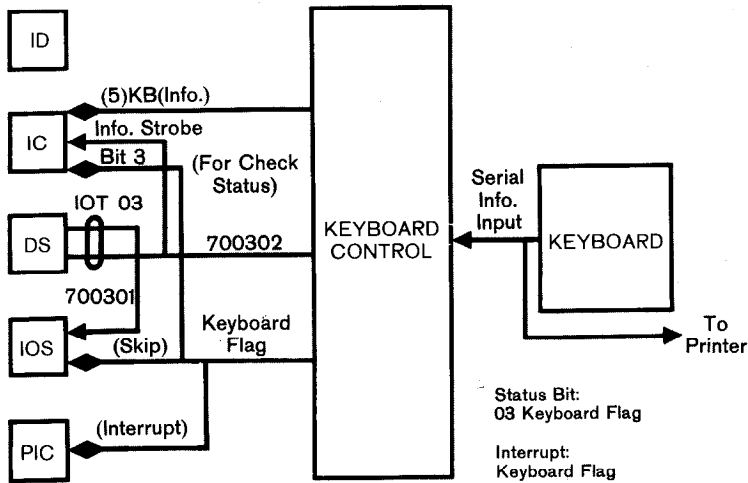


Figure 19 — Keyboard programming logic

The Keyboard flag is connected to the Program Interrupt Control and the `iors` instruction, bit 3. A simple sequence which “listens” for keyboard inputs is:

PROGRAM SEQUENCE

```

/listen loop for keyboard
400/      ksf      /skip when a character arrives from keyboard
          jmp 400
          krb      /read in the character

```

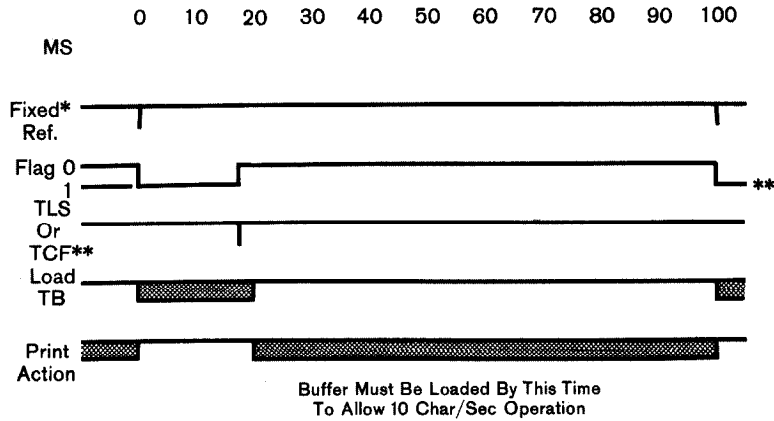
The sequence following the listen sequence beginning in 403 may operate for up to $100 + 13.3$ milliseconds before returning to listen for the next character without missing the next character. The average computing time between any two characters must be less than 100 milliseconds (for an input rate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

TELEPRINTER

The Teleprinter is given 5 bits of information from AC bits 13 to 17, 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 Program Interrupt and to bit 4 of the iors instruction. The printing rate is ten 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) ∨ C(TB). Print a character.



**if TCF, Flag Will Not Come On Until Next TLS Complete
 *Determined By Printer

Figure 20 – Printer timing

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
    :

```

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

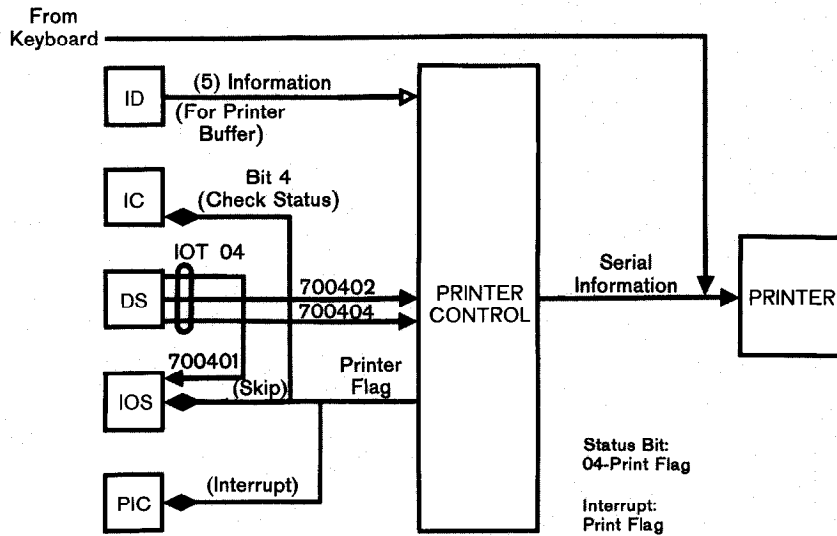
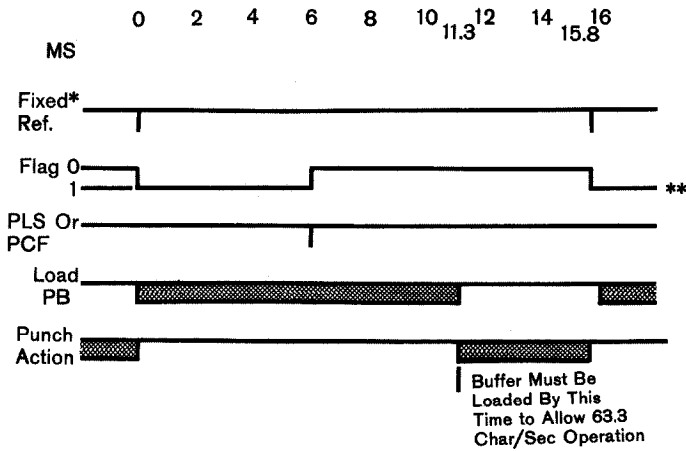


Figure 21 — Printer programming logic

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, signaling that new information may be read into Punch Buffer (PB) (and punching initiated). The Punch flag is connected to the Program Interrupt and to the iors instruction bit 2. The 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).



*Determined By Punch
 **PCF Flag Will Not Come On Until Next P Is Complete

Figure 22 — Perforated-Tape Punch timing

PROGRAM SEQUENCES

```

/punch the contents of AC and wait
    pls      /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.

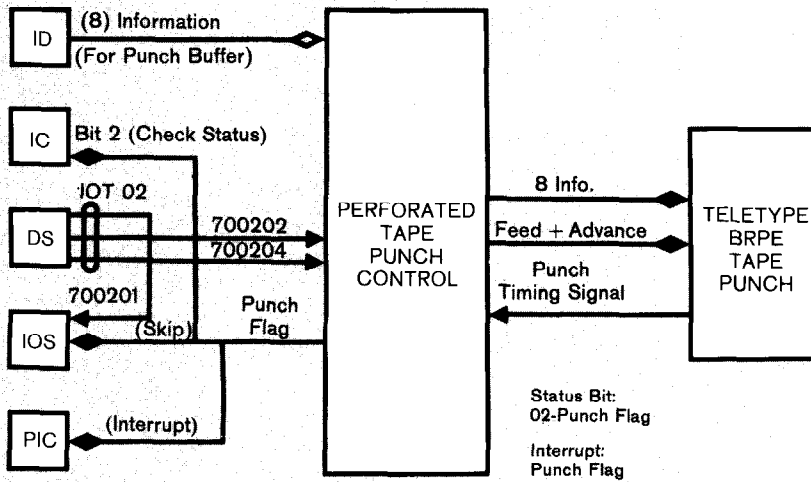


Figure 23 — Perforated-Tape Punch programming logic

CARD READER AND CONTROL, TYPE 41-4

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. 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, 6 information bits are placed into AC bits 12-17. 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 transfers 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.

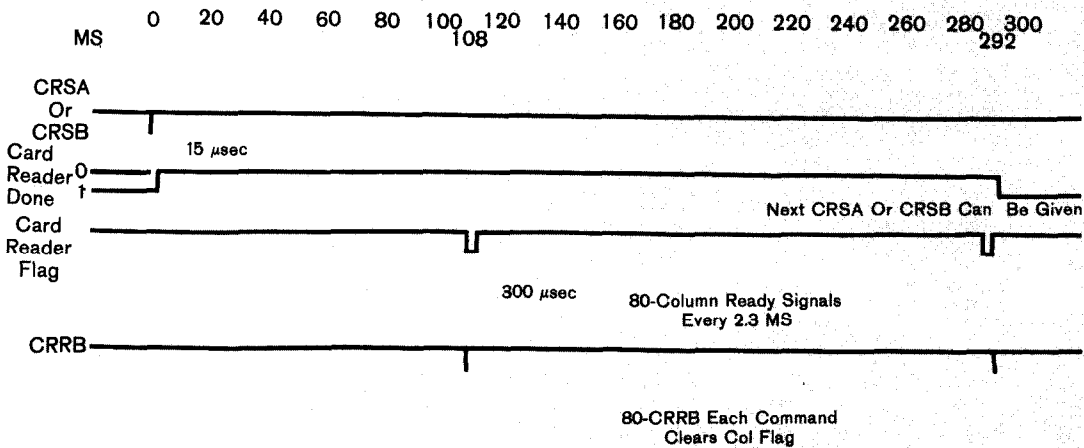


Figure 24 – Card Reader timing

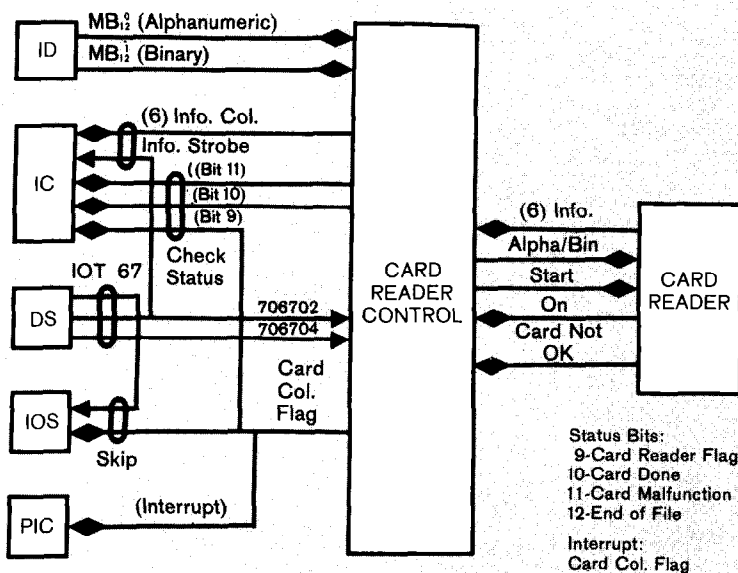


Figure 25 – Card Reader programming logic

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. 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 CONTROL, TYPE 40-4

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 are 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. The commands for the Card Punch Control, iot series 64, are:

cpsf – 706401 – Skip if Card Punch flag is a 1. The Card Punch flag indicates the Punch buffer is available, and should be loaded.

cpcl – 706402 – Clear Card Punch flag.

cpse – 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.

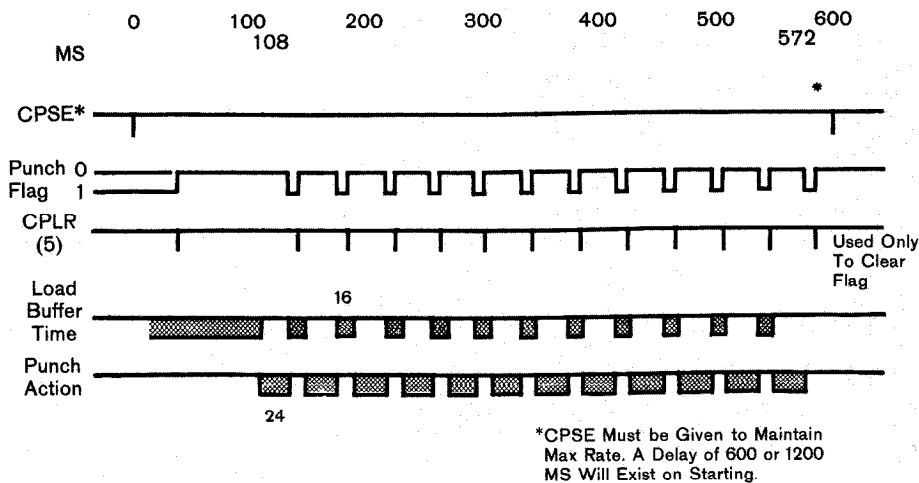


Figure 26 – Card Punch timing

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 instructions fill the first 72 bits (or columns); the fifth loads the remaining 8 bits of the buffer from AC bits 10-17.

After the last row punching is complete, 28 milliseconds are available to select the next card for continuous 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 select 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 iors instruction. Faults occurring in the punch are detected by status bit 14 of the iors and signify the punch is disabled, the stacker is full, or the hopper is empty.

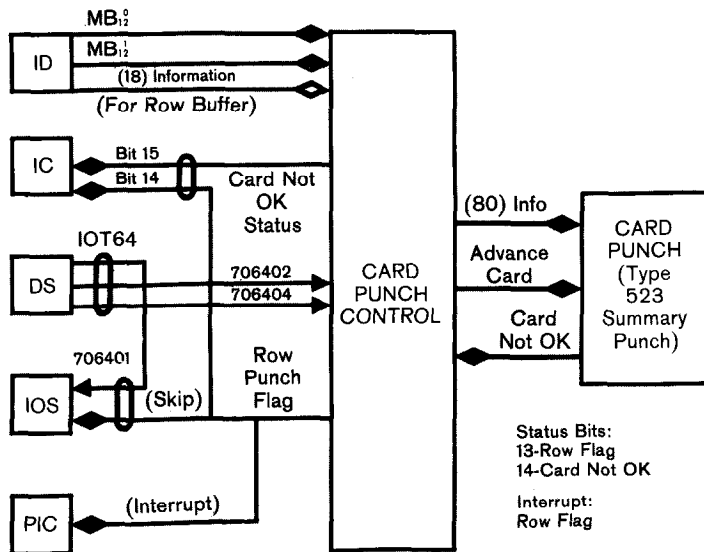


Figure 27 – Card Punch programming logic

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      /select the card
             lac punloc /initialize the card image
             dac 10
             lac rowct
             dac temp1  /initialize the row counts, 12.
/loop1,     lac grpct  /initialize the 5 groups per row
             dac temp2
             cpsf      /sense punch load availability
             jmp.-1
loop2,     lac i 10    /5 groups of 18 bit per row
             cplr      /load buffer command

```

```

        isz temp2
        jmp loop2
        isz temp1      /test for 12 rows
        jmp loop1
        hlt           /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

```

AUTOMATIC LINE PRINTER AND CONTROL, 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 two 66-line pages) per second.

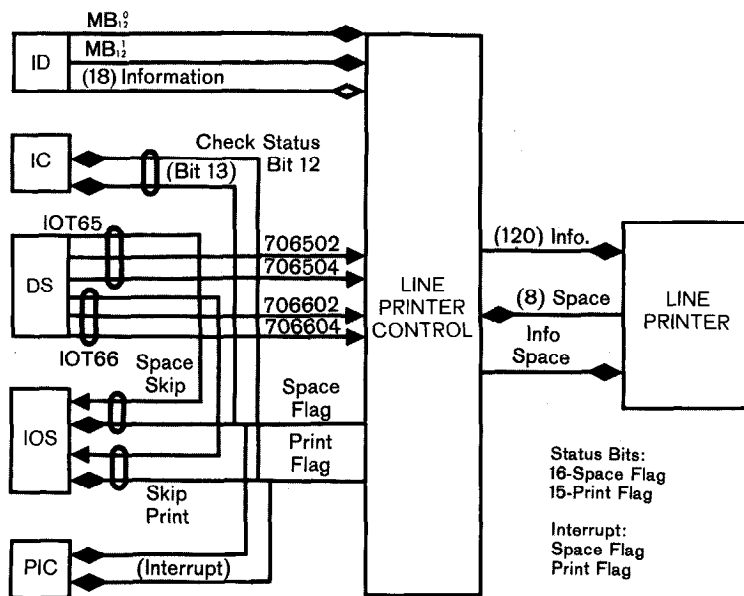


Figure 28 – Line Printer programming logic

A complete line, or 120 columns of information, is placed in the printing buffer. Six bits specify each character (the codes are given in Appendix 2). 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 inside 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 AC bits 15-17, and giving an instruction to space paper. The spacing flag becomes a 1 when the spacing action is complete. A recommended control tape has the following characteristics, where the middle column indicates the number of lines between successive holes in the channel:

Channel	Spacing	Time
0	1 line	16 ms
1	2 lines	$< 2 \times 16$ ms
2	3 lines	$< 3 \times 16$ ms
3	6 lines	$< 6 \times 16$ ms
4	11 lines (1/6 page)	$< 11 \times 16$ ms
5	22 lines (1/3 page)	$< 22 \times 16$ ms
6	33 lines (1/2 page)	$< 33 \times 16$ ms
7	restores page	520 ms for 66 lines

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 Printing buffer.

lpse — 706506 — Select the Printer. Print the contents of the Printing 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 AC bits 15-17 and select spacing. Clear the spacing 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
/characters 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
            lsls + 10     /space 1 line (0 in AC)iot 10 clears
                        /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 Printer-Keyboard. 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.

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 29). 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.

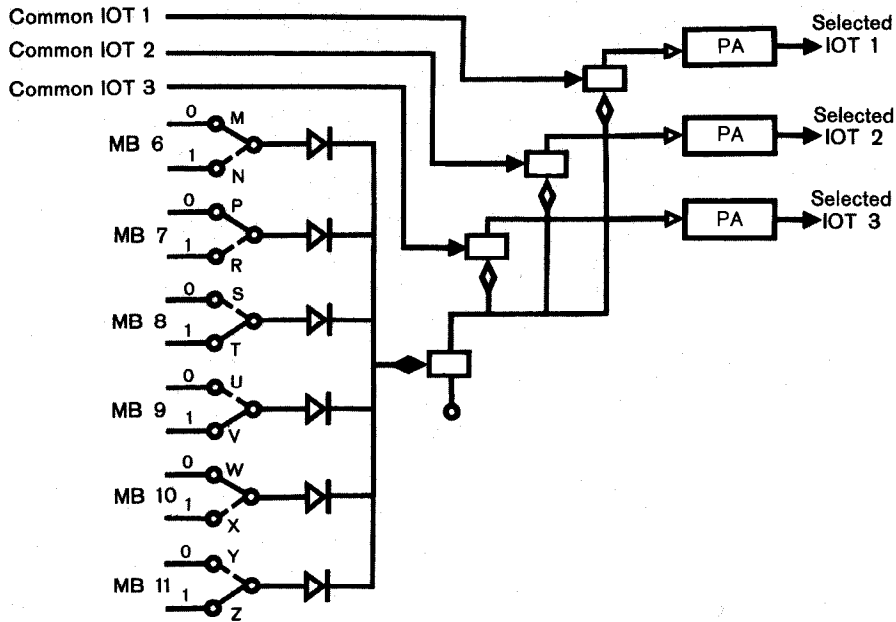


Figure 29-Typical Pulse Amplifier, Type 4605, used in PDP-4 Device Selector. Example shown is wired to pass the iot address 001101. The six-level AND gate will pass only that address if it is present in the instruction word from the Memory Buffer, thus enabling three AND gates to pass three IO pulses to the pulse amplifier.

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.

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 $IC_{j,k}$, where j 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 of the iot instruction.

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

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 30. 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.

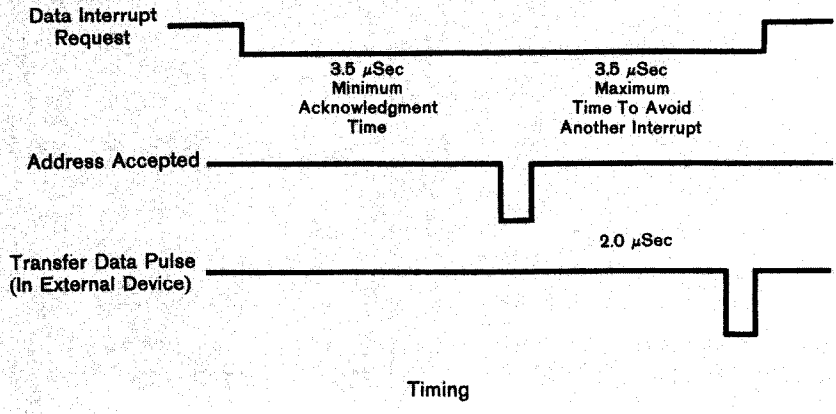
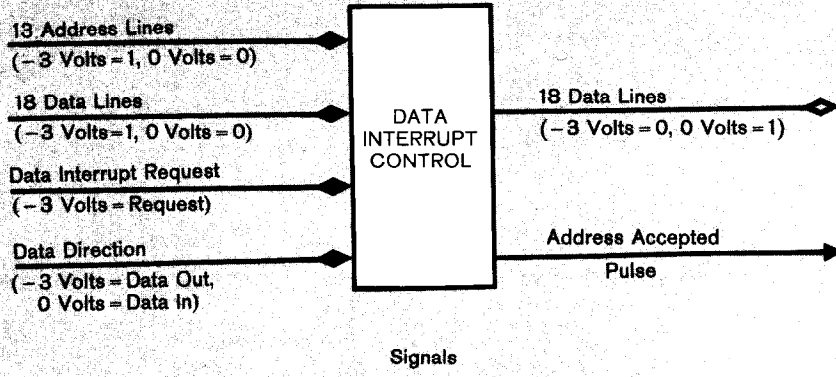


Figure 30 — Data Interrupt Control signals and timing

APPENDIX 1

Instruction Lists

MEMORY REFERENCE INSTRUCTIONS

MNEMONIC CODE	OCTAL CODE	TIME (μ sec)	OPERATION
cal Y	00	16	Call Subroutine. Y is ignored jms 20 if bit 4 = 0, jms i 20 if bit 4 = 1.
dac Y	04	16	Deposit AC. $C(AC) \Rightarrow C(Y)$
jms Y	10	16	Jump to subroutine. $C(PC) \Rightarrow C(Y_{5-17})$, $C(L) \Rightarrow C(Y_0)$, $Y + 1 \Rightarrow C(PC)$
dzm Y	14	16	Deposit zero in memory. $0 \Rightarrow C(Y)$
lac Y	20	16	Load AC. $C(Y) \Rightarrow C(AC)$
xor Y	24	16	Exclusive OR. $C(AC) \vee C(Y) \Rightarrow C(AC)$
add Y	30	16	Add (1's complement). $C(AC) + C(Y) \Rightarrow C(AC)$
tad Y	34	16	2's complement add. $C(AC) + C(Y) \Rightarrow C(AC)$
xct Y	40	8+	Execute.
isz Y	44	16	Index and skip if 0. $C(Y) + 1 \Rightarrow C(Y)$, if $C(Y) + 1 = 0$, then $C(PC) + 1 \Rightarrow C(PC)$
and Y	50	16	AND. $C(AC) \wedge C(Y) \Rightarrow C(AC)$
sad Y	54	16	Skip if AC and Y differ. If $C(AC) = C(Y)$, then $C(PC) + 1 \Rightarrow C(PC)$
jmp Y	60	8	Jump. $Y \Rightarrow C(PC)$
law N	76	8	Load AC with law N. $1 \Rightarrow C(AC_{0-4})$, $N \Rightarrow C(AC_{5-17})$

OPERATE INSTRUCTIONS

MNEMONIC CODE	OCTAL CODE	EVENT TIME	OPERATION
opr	740000	—	Operate.
nop	740000	—	No Operation.
cma	740001	3	Complement, $\overline{C(AC)} \Rightarrow C(AC)$
cml	740002	3	Complement Link, $\overline{C(L)} \Rightarrow C(L)$
oas	740004	3	Inclusive OR AC Switches. $C(ACS) \vee C(AC) \Rightarrow C(AC)$
las	750004	2, 3	Load AC from Switches. $C(ACS) \Rightarrow C(AC)$
ral	740010	3	Rotate AC + Link left one place. $C(AC_j) \Rightarrow C(AC_{j-1}), C(L) \Rightarrow C(AC_{17}),$ $C(AC_0) \Rightarrow C(L)$
rcl	744010	2, 3	Clear Link, then ral. $0 \Rightarrow C(L)$, then ral
rtl	742010	2, 3	Rotate AC left twice. Same as two ral instructions
rar	740020	2	Rotate AC + Link right one place. $C(AC_j) \Rightarrow C(AC_{j+1}), C(L) \Rightarrow C(AC_0),$ $C(AC_{17}) \Rightarrow C(L)$
rcr	744020	2, 3	Clear Link, then rar. $0 \Rightarrow C(L)$, then rar
rtr	742020	2, 3	Rotate AC right twice. Same as two rar instructions
hlt	740040	4	Halt. $0 \Rightarrow \text{RUN}$
sza	740200	1	Skip on zero AC. Skip if $C(AC) = \text{positive zero}$
sna	741200	1	Skip on non-zero AC. Skip if $C(AC) \neq \text{positive zero}$
spa	741100	1	Skip on positive AC. Skip if $C(AC_0) = 0$
sma	740100	1	Skip on negative AC. Skip if $C(AC_0) = 1$
szl	741400	1	Skip on zero Link. Skip if $C(L) = 0$
snl	740400	1	Skip on non-zero Link. Skip if $C(L) = 1$
skp	741000	1	Skip, unconditional. Always skip
cll	744000	2	Clear Link. $0 \Rightarrow C(L)$
stl	744002	2, 3	Set the Link. $1 \Rightarrow L$
cla	750000	2	Clear AC. $0 \Rightarrow C(AC)$
clc	750001	2, 3	Clear and Complement AC. $\text{—}0 \Rightarrow C(AC)$
glk	750020	2, 3	Get Link. $0 \Rightarrow C(AC), C(L) \Rightarrow C(AC_{17})$

BASIC IOT INSTRUCTIONS

MNEMONIC CODE	OCTAL CODE	OPERATION
Interrupt		
iof	700002	turn off interrupt
ion	700042	turn on interrupt
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 beyboard 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

BASIC IOT INSTRUCTIONS

(continued)

MNEMONIC CODE	OCTAL CODE	OPERATION
		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 flag

APPENDIX 2

Codes

FIO-DEC CODE

		High order bits			
		00	01	10	11
a A	61				
b B	62				
c C	63				
d D	64				
e E	65	0000	space	0 →	
f F	66	0001	1 " ' / ?	j J	a A
g G	67	0010	2 ' / ?	k K	b B
h H	70	0011	3 ~	l L	c C
i I	71	0100	4 ∩	m M	d D
j J	41	0101	5 ∨	n N	e E
k K	42	0110	6 ^	o O	f F
l L	43	0111	7 <	p P	g G
m M	44	1000	8 >	q Q	h H
n N	45	1001	9 ↑	r R	i I
o O	46	1010			lower case
p P	47	1011	stop	, =	. X
q Q	50	1100		black	- +
r R	51	1101		red)]
s S	22	1110		tab	-
t T	23	1111		([car ret
u U	24				
v V	25				
w W	26				
x X	27				
y Y	30				
z Z	31				
0 →	20			stop code	13
1 "	01			lower case	72
2 ' / ?	02	/ ?	21	upper case	74
3 ~	03	, =	33	black	34
4 ∩	04	. X	73	red	35
5 ∨	05	- +	54	tab	36
6 ^	06)]	55	backspace	75
7 <	07	([57	carriage return	77
8 >	10	. _	40	space	00
9 ↑	11	-	56		

code delete punches seventh channel

TELETYPE CODE

Low order bits	00	High order bits		11
		01	10	
000		line feed	E 3	A —
001	T 5	L)	Z "	W 2
010	car ret	R 4	D \$	J ' 1
011	O 9	G &	B ?	figures
100	space	I 8	S bell	U 7
101	H #	P 0	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			

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	A
H 70					
I 71	0010	2	S	K	B
J 41					
K 42	0011	3	T	L	C
L 43					
M 44	0100	4	U	M	D
N 45					
O 46	0101	5	V	N	E
P 47					
Q 50	0110	6	W	O	F
R 51					
S 22	0111	7	X	P	G
T 23					
U 24	1000	8	Y	Q	H
V 25					
W 26	1001	9	Z	R	I
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					

HOLLERITH CARD CODE

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

LINE PRINTER CODE

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

APPENDIX 3

Read-In Mode Sequence

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.

READ-IN LOADER

The purpose of the read-in loader is to load programs punched in "read-in mode," such as the block format loader described below. The read-in loader must be loaded by means of the console toggle switches. It loads tapes of the following format:

```
  dac A
  c(A)
  dac B
  c(B)
  .
  .
  .
  jmp Y
  dummy word
```

Read-in mode tapes consist of word pairs giving a dac into an address, followed by the contents of that address. They are terminated by a jmp to the program followed by a dummy word (e.g., 0).

To load a read-in mode tape, place the tape in the reader, set the ADDRESS switches to 7770, and press START.

LOCATION	OCTAL CODE		MNEMONIC	REMARKS
7762/	0	r,	0	/read one binary word
7763/	700101		rsf	
7764/	607763		jmp . - 1	/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	go,	rsb	/enter here, start reader going
7771/	107762	g,	jms r	/get next binary word
7772/	47775		dac out	
7773/	407775		xct out	/execute control word
7774/	107762		jms r	/get data word
7775/	0	out,	0	/store data word
7776/	607771		jmp g	/continue

BLOCK FORMAT 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
Check sum	/sum of every word in block, except check sum

The routine occupies register 7737 to 7761, and uses the read-in loader subroutine 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 of the block and pressing the CONTINUE switch on the console.

LOCATION	MNEMONIC	REMARKS
7737/	rsb	
a,	jms r	/block format loader
	dac s	
	xct s	
	dac cks	
	jms r	
	dac out	
b,	add cks	/loop
	dac cks	
	jms r	
	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	
	cks = 7777	

APPENDIX 4

PDP-4 Assembly Program

The more important characteristics of the PDP-4 Assembly Program are mentioned briefly here to provide the background necessary to understand the programming examples in this manual. The program and its complete description are furnished to purchasers of PDP-4.

CHARACTER SET: The 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
Δ 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
= equals sign	assign symbol on left of =
/ slash	begin comments; set current address
↵ carriage return	termination character
→ tab	termination character
- overbar	variable indicator

The characters Δ, ↵, and → are nonprinting.

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

SYMBOLS: Any sequence of alphanumeric characters, the first of which must be a letter. Symbols are identified by the first six characters only.

'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

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

Examples:

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

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 equals 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:

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

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 the address into which the next instruction or data word will go. Normally, the first instruction or data word goes into register 22 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 22, then a slash may be used to set the new address.

INDIRECT ADDRESSING: Indirect addressing is indicated by the symbol, i which has the value 20000.

Example: lac i abc

THE CHARACTER PERIOD (.): The period (.) has as its value the current address.

Example: dac . is equivalent to
a, dac a

PSEUDO INSTRUCTIONS

FLEXOWRITER INPUT PSEUDO INSTRUCTIONS: The pseudo-instruction, flex $\Delta\alpha\beta\gamma$ causes the (six-bit) FIO-DEC 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 β 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 FIO-DEC code, the sixth a 1 for upper case or a 0 for lower case.

Example: flex Δ boy

The pseudo-instruction, char $\Delta Z\gamma$ causes the (six-bit) FIO-DEC character, γ to be assembled into the left, middle, or right six bits of the word, depending on whether Z is r, m, or l.

Example: char r0
char m(
char la

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 (α).

Examples of the use of constants:

```
add (1)↵  
lac (add z 1)↵  
lac (-760000)↵  
lac (flexo abc)↵
```

START: The pseudo-instruction, start indicates the end of the English tape. Instruction, start A must be followed by a carriage return. "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 two.

DECIMAL: The pseudo-instruction, decimal indicates all numbers are to be considered decimal.

OCTAL: The pseudo-instruction, octal indicates all numbers are to be considered octal.

APPENDIX 5

Multiply and Divide Subroutines

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, approximately 100 microsec. for zero.

```
mult,    0
         dzm mp5
         sna
         jmp mpz
         spa + cll — 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 + cll — opr
         tad mp2
         rar
         dac mp5
         isz mp3
         jmp mp4

mpsign,  0
         dac mp5
         lac mp1
         rar
         xct mpsign

mpz,     isz mult
         jmp i mult

start
```

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 ms

```
divide, 0
        spa + cll — opr
        cma + cml — opr
        dac dvd
        xct i divide
        spl
        cma
        dac quo
        jms dv5

dv5,    0          /remainder has sign of dividend
        isz divide
        xct i divide
        sma + cml — opr
        cma + cml — opr
        jms dv4

dv4,    0
        cll
        tad (1
        dac dvs
        tad dvd
        isz divide
        spl
        jmp i divide
        lac (-22
        dac dv1
        jmp dv2

dv3,    lac dvd
        ral
        dac dvd
        tad dvs
        spl
        dac dvd
```


DIVIDE SUBROUTINE

(continued)

```
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 + cll — opr
      jmp i divide

start
```

APPENDIX 6

Programming Aids

The following programming aids are supplied with the PDP-4.

PDP-4 ASSEMBLY PROGRAM—A one-pass assembler which allows mnemonic symbols to be used for addresses and instructions. Constants are automatically assigned. Text statements may be written for printing at run time, and a decimal mode may be specified. Up to six character symbols may be used, and the symbol table may be punched on paper tape for use with the debugging tape below.

DDT-4 DEBUGGING TAPE—Provides communication with a program via the on-line typewriter. Registers may be examined (using mnemonic codes) and modified. Communication is entirely in symbolic language. Programs may have break points inserted and then run under DDT-4 control, similar to a tracing routine. A program may be searched for particular words.

DOUBLE-PRECISION FLOATING POINT PACKAGE—Provides floating-point arithmetic with a 36-bit mantissa and 18-bit exponent. The routines include plus, minus, divide, multiply, fix-to-float, and float-to-fix, with decimal input and output.

MAINTENANCE ROUTINES—There are five maintenance routines. These tests are also used as DEC's standard acceptance test routines.

- (a) **CONTEST (CONTinuous TEST)**—Verifies that all machine functions are performing properly. Each instruction is tested, a core checkerboard pattern is run, a tape is punched and read, and a message is typed. The test then repeats itself.
- (b) **INSTEP (INStruction TEst Programs)**—Test all machine instructions under various modes.
- (c) **Checkerboard Program**—Provides continuous memory testing with four different patterns.
- (d) **Reader and Punch Test**—Checks the start time of the reader and checks the reader using different patterns and variable times. The punch is tested by providing tapes for the reader test.
- (e) **Teleprinter Test.**

TAPE REPRODUCER—Reproduces tape using the Interrupt Mode.

PUNCH ROUTINES—Allow punching in either block format or read-in mode format.

OCTAL DEBUG — A simple debugging routine.

MISCELLANEOUS INPUT-OUTPUT ROUTINES—Octal, decimal, double precision input and output and special Teletype conversion routines.

DEMONSTRATION PROGRAMS—Included are: Three Point Display (Tri-Pos), Pen Follow, Type-in Character Display, and Character Punch.

FLOATING POINT FUNCTIONS — Allows various functions to be computed, such as double precision sine, cosine, tangent, exponents, log base e, and square root. Inquire at DEC for the completion date of these sub-routines.

ALGEBRAIC COMPILER — Inquire at DEC for the completion date of this FORTRAN compiler.

APPENDIX 7

Powers Of Two

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 808 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 848	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 081 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625
562 949 953 421 312	49	0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5
1 125 899 906 842 634	50	0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25
2 251 799 813 985 248	51	0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125
4 503 599 627 370 496	52	0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 668 164 062 5
9 007 199 254 740 992	53	0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 834 582 031 25
18 014 398 509 481 984	54	0.000 000 000 000 000 055 511 151 231 257 827 021 171 513 417 041 015 625
36 028 797 018 963 968	55	0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 708 520 507 812 5
72 057 594 037 927 936	56	0.000 000 000 000 000 013 877 787 807 814 456 755 215 395 854 260 253 906 25
144 115 188 075 855 872	57	0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 927 130 126 953 125
288 230 376 151 711 744	58	0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 963 565 063 476 562 5
576 460 752 303 423 488	59	0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 782 531 738 281 25
1 152 921 504 606 846 976	60	0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 891 265 869 140 625

digital

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS