

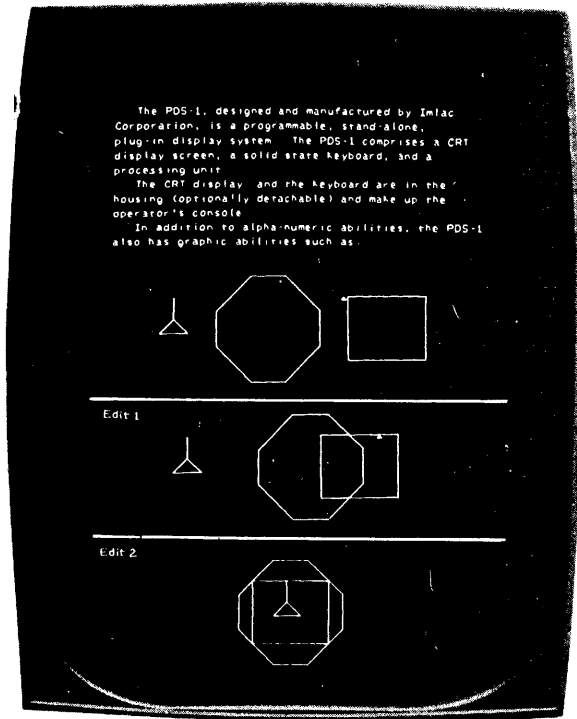
PRELIMINARY TECHNICAL AND OPERATOR MANUAL

IMLAC PDS-1 PROGRAMMABLE DISPLAY SYSTEM

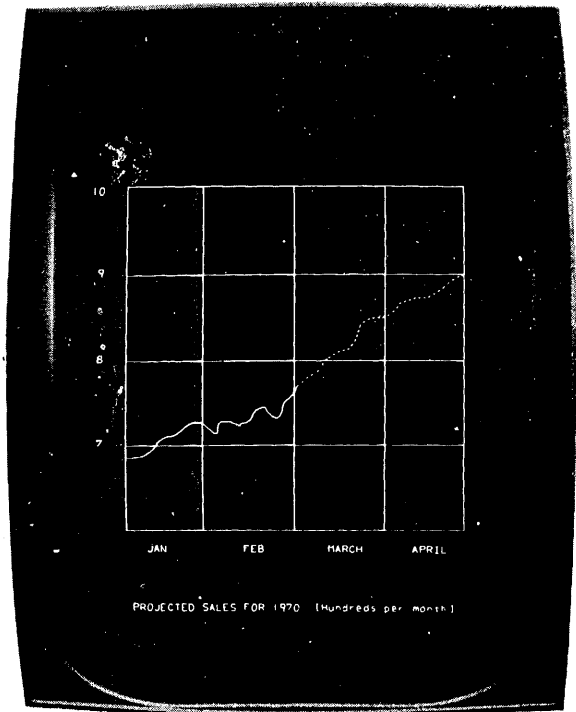
© IMLAC Corporation 1969, 1970

Control No. \_\_\_\_\_

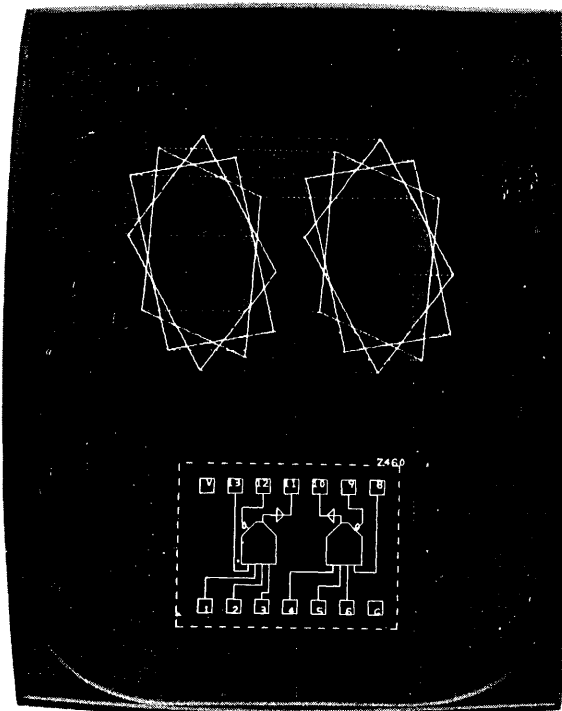
Rev. A 3/70



**Local Alpha-graphics Edit**



**Finance**



**Engineering Design**

These are actual photographs taken directly from the screen of a PDS-1.

The PDS-1 is also capable of moving display animations which are difficult to represent in a still photo.

PRELIMINARY TECHNICAL AND OPERATOR MANUAL

IMLAC PDS-1 PROGRAMMABLE DISPLAY SYSTEM

I.	INTRODUCTION	I-1.1-1.2
II.	FUNCTIONAL DESCRIPTION, OPTIONS AVAILABLE	II-1.1-1.5
III.	SYSTEM DISPLAY SPECIFICATIONS	
	1. Transmission Characteristics	III-1.1
	2. General Characteristics	III-2.1
	3. Terminal Memory	III-3.1
	4. Programmable Features, Keyboard	III-4.1-4.3
IV.	ASSOCIATED OPTIONAL DEVICES	
	1. Optional Input-Output Interface	IV-1.1-1.6
	2. External Program Interrupt Facility	IV-2.1
	3. Print Capabilities	IV-3.1
	4. Lightpen Option	IV-4.1
V.	PROCESSOR DESCRIPTION	V-i-iii
	1. Processor Instructions	V-1.1-1.5
	a. Operator Instructions	
	b. IOT Instructions	
	c. Skip Instructions	
	d. Shift & Rotate Instructions	
	2. Display Instructions	V-2.1-2.3
	a. Display Operate	
	b. Bit Numbers	
VI.	COMPUTER SOFTWARE PACKAGES	
	1. Text and Graphics Editor, Operating Instructions	VI-1.1-1.10
	a. Keyboard Operations	
	b. Cursor Control	
	c. Control Characters	
	d. Transmission Control	
	e. Form Mode Control	
	f. Graphics	
	2. Utility Package	VI-2.1-2.22
	a. Loaders	
	b. Assembler	
	c. Debuggers, Diagnostics	
	d. Illustrative Simple Display Program, Vector Description of Letter "d"	
	3. Interactive Graphics Package	VI-3.1

## I. INTRODUCTION



## I. INTRODUCTION

The PDS-1, designed and manufactured by Imlac Corporation, is a programmable, stand-alone, plug-in display system. The PDS-1 comprises a CRT display screen, a solid state keyboard, and a processing unit.

The CRT display and the keyboard are in the

housing (optionally detachable) and make up the operator's console. The display screen is capable of displaying about 1200 flicker free characters and may be oriented in either the vertical or horizontal direction. The keyboard has two sections, a standard teletype configuration and a console and data control section. Operator instruction time is minimal and requires no programming ability.

The flexibility of the PDS-1 is derived from the central processing unit (CPU). An optional compact control console (see Fig. I-2) provides the operator a means of controlling the CPU directly. The control console enables the operator to visually check the static or dynamic condition of the CPU. This is accomplished with indicator lamps for register contents, as well as switches to give control of the state of machine.

Each of these sections has unique characteristics which, when combined, afford the user a computer peripheral which is compatible with any telecommunication computer system and interchangeable with all existing communications-oriented I/O devices.

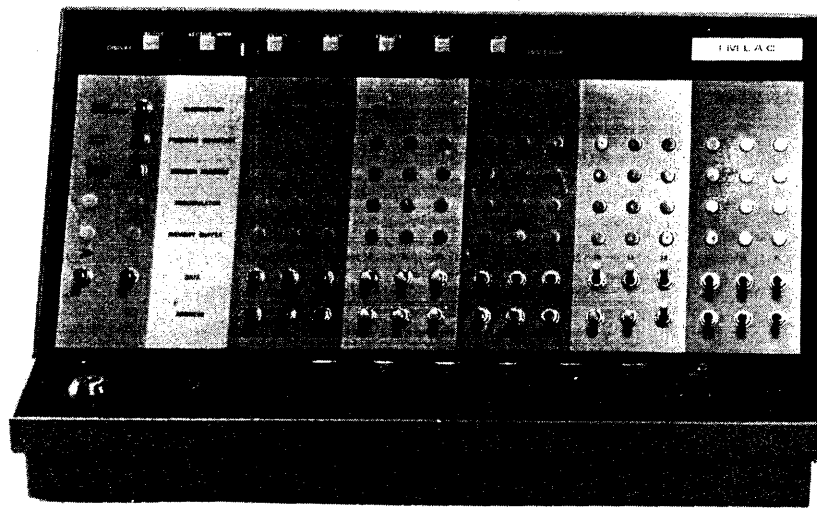


Fig. I-2 Control Console  
(Also called Programmer/Maintenance Console)

## II. FUNCTIONAL DESCRIPTION, OPTIONS AVAILABLE

## II. FUNCTIONAL DESCRIPTION

The display screen of the PDS-1 terminal has a large viewing area on a 14 inch CRT refreshed from a local memory at 40 cycles per second. The system can display about 1200 characters or 500/800 inches of graphics or combinations of characters and graphics, depending on the efficiency of the character and graphics definitions in the display list. The operator is able to make any number of insertions, deletions or changes of characters or graphics. Four character sizes are available; there are upper case, as well as, lower case characters, numerals and other graphic "characters" which may be used by the operator. The resolution of the display system is up to 1,024 points in X and up to 1,024 points in Y.

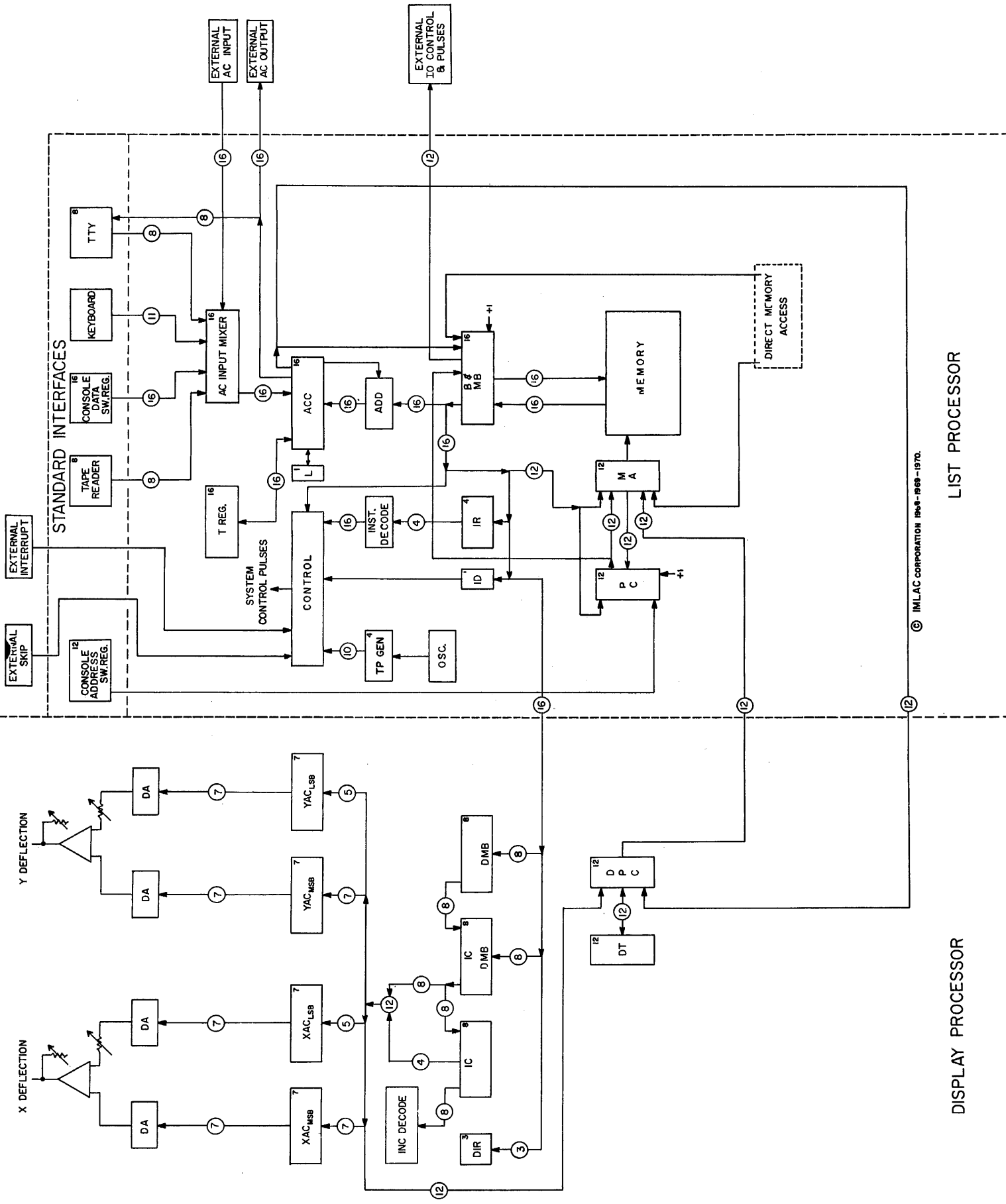
A character is composed of strokes, both visible and invisible defined by software character subroutines. The byte format for a stroke is shown in Figure II-1. The origin of each stroke is the termination of the previous stroke except when the origin is specified by a non-stroke byte. This is what is meant by the "Increment Mode" in vector operation.

The PDS-1 operates two processors simultaneously sharing the same core memory (see Figure II-2). The display processor, normally refreshing the CRT at 40 frames per second, has priority over the data processor, which is a full blown, 16 bit mini-computer. The display processor can "steal" cycles from the data processor if necessary and has its own registers, instructions, and instruction decoding capability.

For the standard text and graphics edit program, roughly half of the 4096 words of core is dedicated to the program itself and half to the display list buffer.



The display processor races through the display list from top to bottom of the buffer, normally at 40 frames per second. The text edit program allows insertions and deletions into the display list which usually results (in an oversimplified description) in the display list being pushed down, or pulled up at the point of insertion or deletion. The display list itself, is generally made up of actual display vector information in IMLAC byte format, or calls to display subroutines, which are in permanent core storage.



© IMLAC CORPORATION 1968-1969-1970.

LIST PROCESSOR

DISPLAY PROCESSOR

Figure II - 2

II-1.3

## II. OPTIONS AVAILABLE

The PDS-1 can be configured with the following available options:

- . Extended Memory (Up to 32K)
- . Extended Input/Output Interface, for parallel transmission between the PDS-1 and other peripherals such as magnetic tapes, cartridges, disc-packs, line printers etc., and other computers.
- . Paper Tape Reader Interface
- . Paper Tape Punch Interface
- . Hard copy to serial printers and teleprinters
- . Lightpen, tablet and joystick interface
- . High Resolution, High Contrast Tubes
- . Read Only Memory Bootstrap Loaders

The PDS-1 can communicate intelligently with an outside network of other PDS-1's. The PDS-1 is either a Very Smart Terminal or a pretty fair stand alone computer with display. The PDS-1 distributes computing power to the operator consoles of a large system, thereby unloading central computing facilities.

BIT NUMBER							
8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7
1 = Increment	1 = Beam ON 0 = Beam OFF	0 = + $\Delta X$ 1 = - $\Delta X$	$\Delta X = 2$	$\Delta X = 1$	0 = + $\Delta Y$ 1 = - $\Delta Y$	$\Delta Y = 2$	$\Delta Y = 1$

Fig. II-1

8 Bit Byte Format Showing the 98 Vector Possibilities  
 49 With Beam ON, 49 With Beam OFF

### III. SYSTEM DISPLAY SPECIFICATIONS

### III. SYSTEM DISPLAY SPECIFICATIONS

Since the PDS-1 system is programmable, many specifications may be chosen by the user. Therefore, these descriptions indicate which specifications are selectable and what the bounds of selection.

Table III-1

Transmission Characteristics	
<u>Features</u>	<u>Specifications</u>
Transmission Rate (bits per second)	definable by user (1 to 9600 baud)
Transmission Code	definable by user (5 to 8 information bits) send & receive codes may be different
Type of Modem	any AT&T approved modem or any acoustic coupler

✓ Since the input and output electronics are quite independent, the codes and transmission rates do not even have to be the same. For example, a 6 bit code at 133 baud may be input to a machine which transmits 8 bit code at 1200 baud.

The CRT display normally is vertically oriented and offers a display field of 80 characters per line by 40 lines. Sixty-four lines is an absolute maximum but a tradeoff must be kept in mind between total flicker-free characters displayable and the display frame rate. Dropping the refresh rate to 30 frames/sec. alleviates the loading, of course.

In the optional horizontal position, the CRT can display 128 characters by 40 lines. This makes the PDS-1, when connected to a hard copy terminal such as the G. E. "Terminet," compatible with that terminal, character space for character space.

Table III-2

General Characteristics	
<u>Features</u>	<u>Specifications</u>
Keyboard	Similar to standard Teletypewriter completely software oriented
Function Keys	standard
Overlay Capability	optional
Hard Copy Capability	optional
Vector Generation	standard
Graphic Input Capability	standard
Display Size (inches - w&h)	7.5 x 8.5
Spot Diameter	10-15 mils nominal
Characters per Line	128 max (horizontally oriented)
No. of Lines of Characters	64 max (vertically oriented)
Maximum No. of Characters/Symbols Displayable	about 1200 flicker free
Character Size (inches - w&h)	variable, programmable and definable
Character Generation Technique	solid stroke
Deflection Method	Magnetic
Brightness (foot lamberts)	50
Contrast Ratio	10:1, options available

For customers desiring more characters on the screen, two options are available; a lower refresh rate (30 Hertz), with attendant slight flicker, or more coarsely drawn character with less legibility. When graphics are the prime emphasis for the use of the machine, the display raster may be adjusted to be square (8" x 8").

Table III-3

Terminal Memory	
<u>Features</u>	<u>Description</u>
Type of Memory	core, up to 32K 16 bits
Memory Size (No. of characters)	variable, over 1500 characters
Memory Refresh Rate (frames per second)	40 (30 or 60) optional

When the application requires, additional core (up to 32K total) may be provided and addressed via the indirect address scheme. The optional I/O system will support tapes, discs, and other devices for more massive local memory requirements.



## Programmable Features

1. Arithmetic
  - a. totals or sub-totals; also sophisticated math routines
2. Controls
  - a. formats
  - b. data validation

Arithmetic - The display's processor, with its computational ability, can be programmed to do specific total and sub totals enabling the operator to check numerical input prior to transmission.

Formats - This is for defined applications where the operator fills in predescribed areas on the screen and transmits only the variable data.

Data Validation - The processor can be programmed to internally validate specific character positions and restrict transmission until corrections are made.

Since the PDS-1 comprises a general purpose 16 bit mini-computer, it can be custom programmed for almost any application.

THE PDS-1 KEYBOARD-SOFTWARE PROGRAMMABLE

Standard ASCII (in general).

<u>Special Keys</u>	<u>Octal Code (8-bit)</u>
TAB	011
EOM	031
FF	014
DEL	177
VT	013
HT	011
HOME	017
INS	003
PAGE XMIT	016
DATA XMIT	002
CR	015
↑	006
→	005
←	010
↓	004

Blank keys are ignored completely on the present model keyboard.

There is no way of generating control characters such as CTRL, A,B,C, etc. directly by hardware from the keyboard.

Other characters such as [, \, [←, ↑, @, escape, altmode, break, must be generated via programming.

Keyboard Test Program

ADDRESS	MNEMONIC	COMMENT	OCTAL CODE
0	KSF	skip on keyboard flag	002020
1	JMP .-1		010000
2	CLA	clear AC	100001
3	KRC	keyboard read & clear	001023
4	JMP .-4		010000

ASCII - 8 level code comes in bits 8-15 of accumulator on console.

"Repeat" does not change the eight bit ASCII code of the key struck. It raises a logical level which gets inclusive OR'D into bit 5 of the accumulator on a keyboard read. This level stays high as long as the repeat key is depressed. Striking the repeat key does not set the keyboard input flag.

"Control" works the same as repeat but on bit 6 of the accumulator.

"Shift" has two functions. It performs the standard ASCII code modification function and words like "control" or "repeat" on bit 7 of the accumulator.

On a keyboard read, the ASCII 8 bit code goes to bits 8-15 of the accumulator.

Bit 8, the parity bit in ASCII, is washed out in all PDS-1 standard programs.

Repeat, Control, and Shift go to bits 5, 6, and 7 of the accumulator respectively, on a read.

#### Keyboard Buffer

On a keyboard read the contents of the 8 bit keyboard buffer are inclusive OR'D into the accumulator along with the logical levels of Repeat, Control, and Shift. On a keyboard read and clear flag or keyboard clear flag the 8 bit keyboard buffer is cleared, but Control, Repeat, and Shift, being logical levels rather than the contents of a buffer are not cleared.

#### IV. ASSOCIATED OPTIONAL DEVICES

#### IV. ASSOCIATED OPTIONAL DEVICES

##### 1. Optional Input-Output Interface

The PDS-1 has been designed to interface easily with a wide variety of external devices. This section of the users manual defines the interface characteristics of the PDS-1 to enable a user to design and construct interfaces for connecting equipment to the PDS-1 using either IMLAC supplied modules or other TTL logic.

The straightforward I/O interface method coupled with the application engineering support available from IMLAC enables the user to connect his own equipment with minimum trouble. IMLAC makes no representation that the use of the logic and circuits described herein will not infringe on existing or future patent rights nor does any description imply the license to use, manufacture or sell any equipment.

Input and output transfers take place under program control. The maximum rate for program controlled transfers of 16 bit data words is in excess of 100KHZ when operations such as status checking are not required. All data transfers, both in and out, are via the 16 bit accumulator. A program skip may be caused as a function of the status of an external bit. The external output, input and skip pulses are generated under program control by external I/O timing and selection cards.

##### INPUT-OUTPUT ELECTRICAL CHARACTERISTICS

The PDS-1 is built from 7400 series logic as manufactured by Texas Instrument and described in the TI Catalog CC201. Some essential characteristics are:

	Min.	Typ.	Max.
Low level output voltage		0.22	0.40 volts
High level output voltage	2.4	3.3	volts
Low level output sink capability			16.0 mA
gates			48.0 mA
buffers			
High level output source capability			400 $\mu$ A
gates			1.2 mA
buffers			
Output reverse current			250 $\mu$ A
Input low level current			-1.6 mA
Input high level current			40 $\mu$ A

INPUT-OUTPUT TIMING

When the I/O Command 001ABP is given AB selects the device and P selects the pulses given to the device as shown in figure 1.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0		1			A			B			P			
							← DEVICE			NUMBER →			P3	P2	P1

Fig. 1 I/O Instruction Decoding

The I/O Instruction is a 2  $\mu$  sec. instruction whose timing is shown in figure 2.

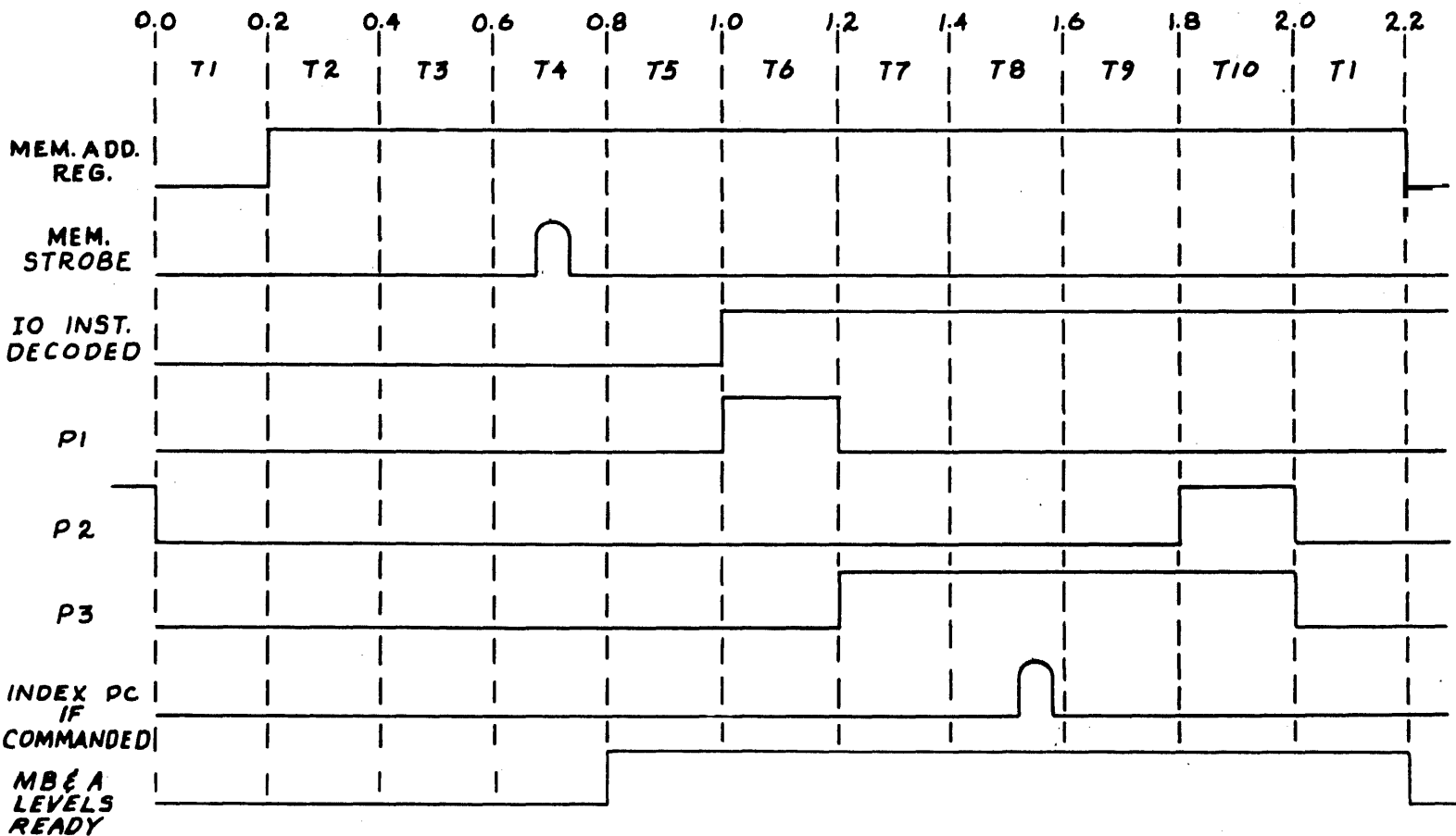


Fig. 2 I/O Timing

SELECTION

The "B" part of the device number is presented at the I/O connector as MB10, MB11, and MB12. The "A" part of the selection number is presented at the I/O connector as six levels with the selected level greater than 2.5 volts and the other five levels less than 0.4 volts. Two of the possible eight "A" levels are used internally and are not available at the I/O connector. Device numbers 20 through 77 may be decoded externally. The Memory Buffer 10-12 bits and the "A" levels are always following the Memory Buffer. The pulses P1, P2, and P3 are only decoded during an I/O command.

To select a particular I/O command a Supplemental I/O Control Board (2034) is needed.

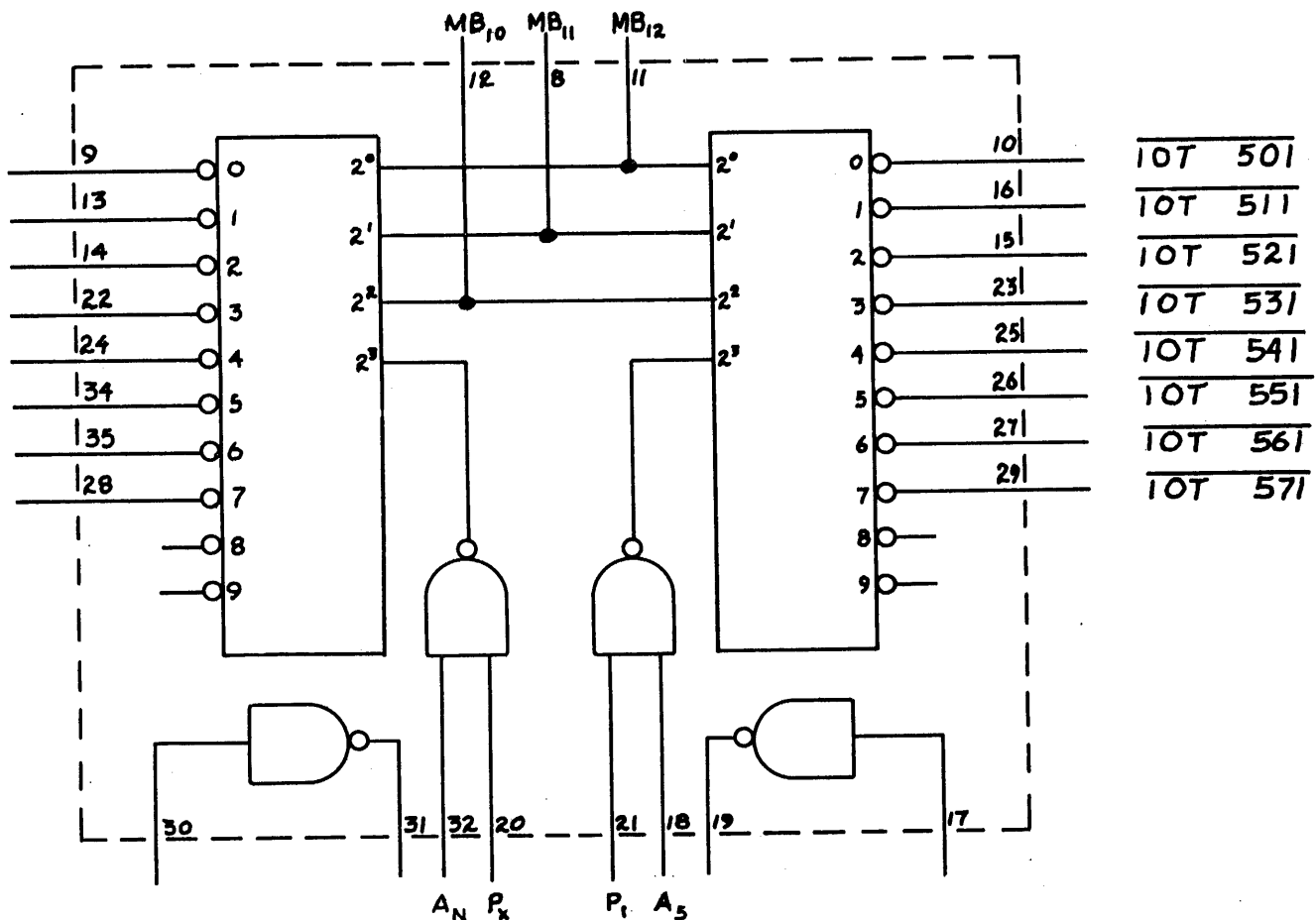


Fig. 3 Generation of I/O Pulse Using Supplemental I/O Control Board

The bits MB 10-11-12 are always connected to pins 11, 8, and 12 respectively. The decoded "A" level is connected to one leg of an input NAND gate and an I/O pulse connected to the other leg. The output is a negative going signal and occurs for the duration of the time pulse. Thus if, as shown in figure 3, A5 were connected to pin 18, P1 were connected to pin 21, and the IOT command 001531 given, a negative going pulse would appear on pin 23 during P1 time. Normally P1 and P2 are used for I/O data transfers and P3 is used for I/O skips.

### I/O SKIP

As I/O pulse can be used to test the condition or status of a device and, if the condition is found to be true, to cause the program to skip the next instruction. The pulse P3 selected by bit 13 must be used for I/O skip to insure that the index Program Counter time (see fig. 2) is bracketed.

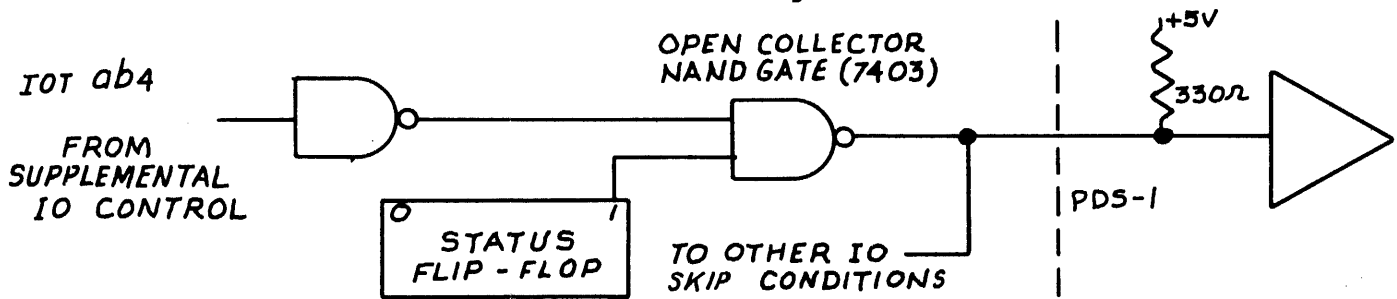


Fig. 4 Logic of an I/O Skip

The logic for an I/O skip is shown in figure 4. The controlling of the status flip-flop can be under program control, under device control, or some combination of each.

### INPUT DATA TRANSFERS

When an external device is ready to send data to the PDS-1, the device sets its status flip-flop. The users program, which should periodically sample the different status flip-flops, senses the ready status and issues an I/O command to read the external devices data into the AC. The AC should be cleared before the read-in command since the resultant word in the AC is the inclusive OR of the input data and the previous AC data.

Each bit of the AC Input (ACI) has an input inverter with 330 ohms to +5 volts on the inverter input. Up to 20 peripheral devices may be hung on a single input inverter in the wired OR connection.



Figure 5 shows the logic for loading data into the AC from an external device.

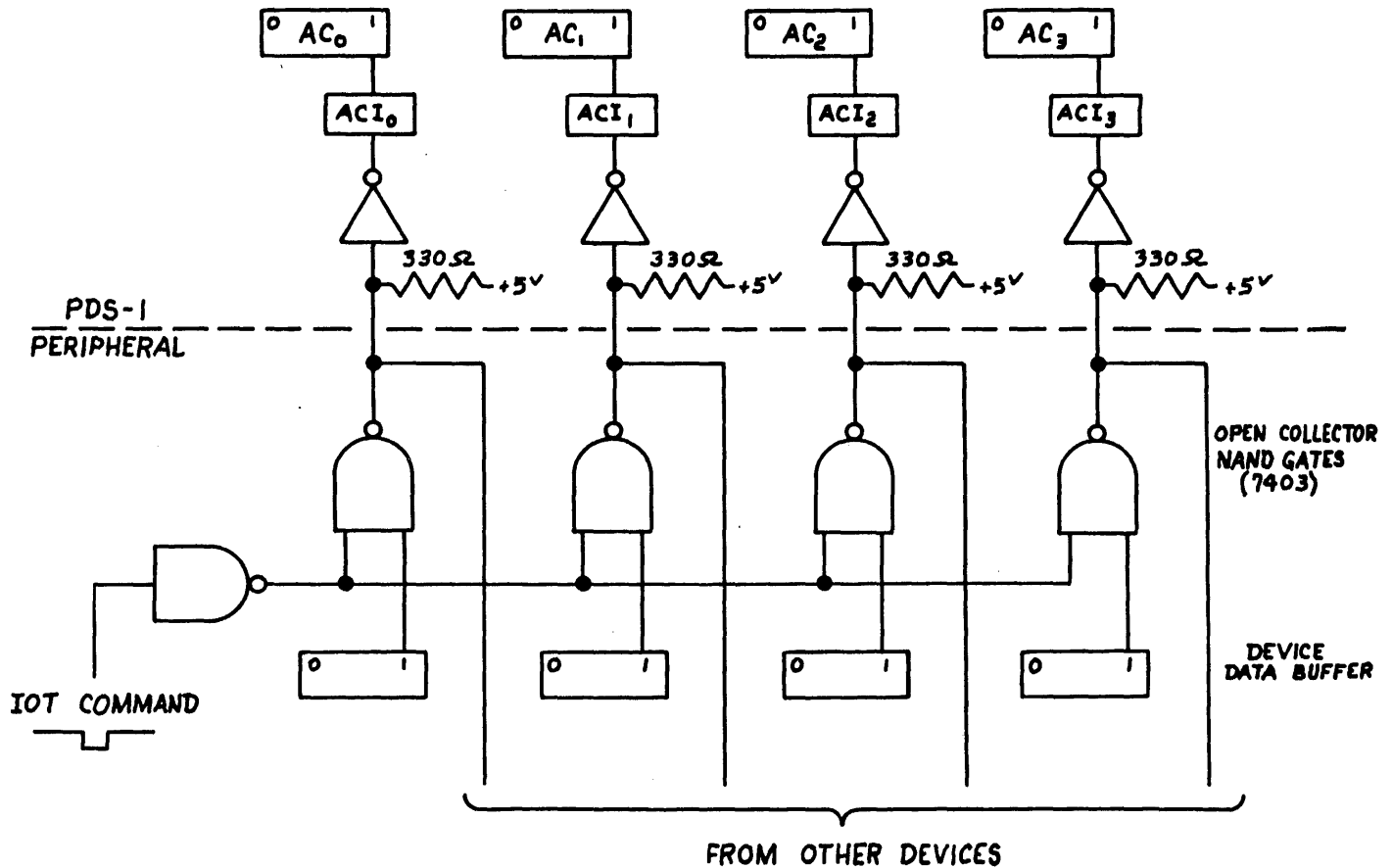


Fig. 5 Logic of Input Data Transfer

As input devices are attached to the input inverters, care should be taken to limit the capacity to about 500 μmf per input inverter.

#### OUTPUT DATA TRANSFERS

When it is desired to send data to an external device, the AC is loaded with the appropriate data and an I/O command is executed. The data word thus transferred may be a character to be operated upon or a status word to establish an operating mode. A maximum of 16 bits may be transferred by any one I/O command.

The true function of each AC bit is brought to the I/O connector. Figure 6 shows the logic for loading data into an external device buffer from the AC. The flip-flop buffer is shown made of D type flip-flops (ex. 7475) with the data on the D inputs and the I/O command on the clock input. If JK flip-flops are used (ex. 7473), it would be necessary to clear the device buffer before reading into it.

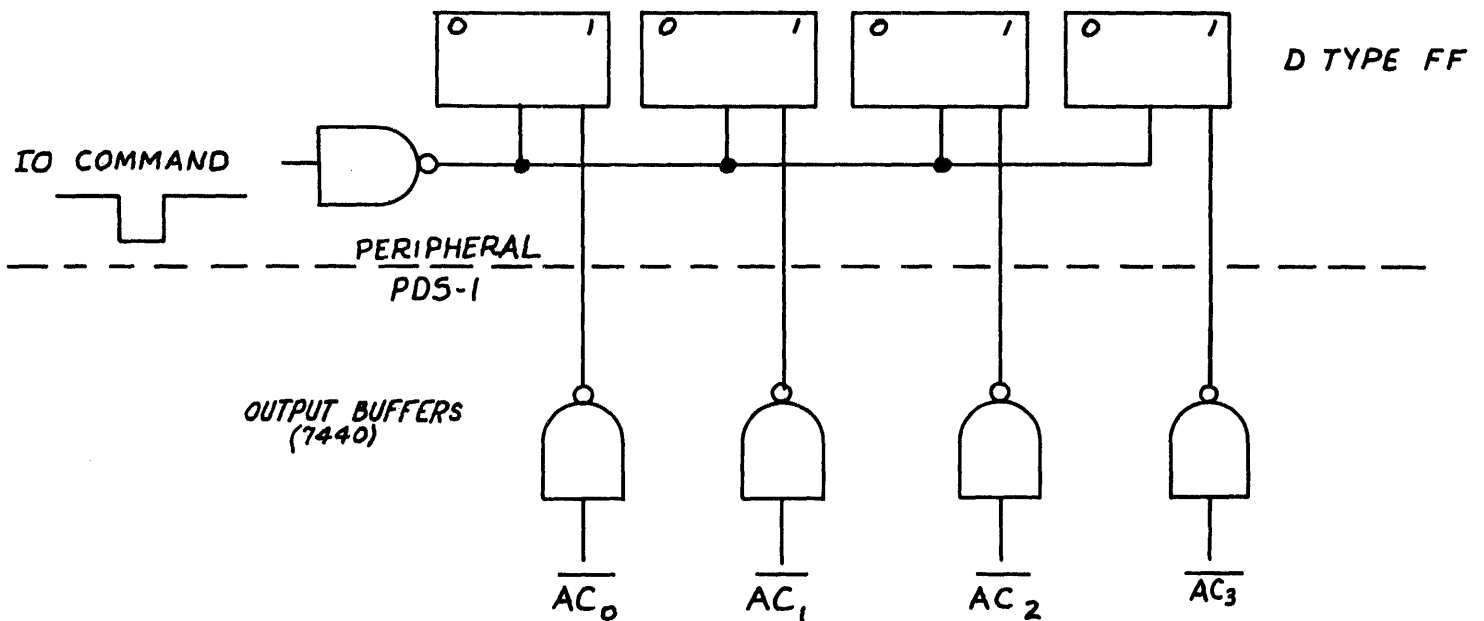


Fig. 6 Logic of Output Data Transfers

#### IV. ASSOCIATED OPTIONAL DEVICES

##### 2. External Program-Interrupt Facility

The PDS-1 program interrupt facility, when enabled under program control (001 162), allows the status flip-flop of an I/O device to force an interrupt upon the completion of the current instruction, thereby alleviating the need for repeated flag checking by the main program. An interrupt is the equivalent of a subroutine jump to memory location 0000. The subroutine starting at zero is used to determine what device caused the interrupt and to take appropriate action.

The interrupt causes the address of the next instruction of the main program to be stored in memory location 0000, the next instruction to be taken from memory location 0001, and the program interrupt facility to be disabled. After the program determines which device has caused the interrupt, the program should clear the corresponding status flip-flop. The final two instructions of the interrupt subroutine should be:

```
001 162   enable interrupt
110 000   indirect jump to 0
```

There is a one instruction delay between the issuance of 001 162 and its execution, so that, if there is another interrupt waiting to be serviced when the enable interrupt is given, the proper contents of the PC for the main program is stored in memory register 0000. The disable interrupt (001 161) is available if the programmer wishes to create logical program interrupt priorities.

The logic for an interrupt is shown in figure 7. Up to twenty gates may be connected to the interrupt input. The determination of which status bit caused the interrupt may be done either by I/O skips as explained with reference to figure 4 or by an input data transfer of the status bits as explained with reference to figure 5.

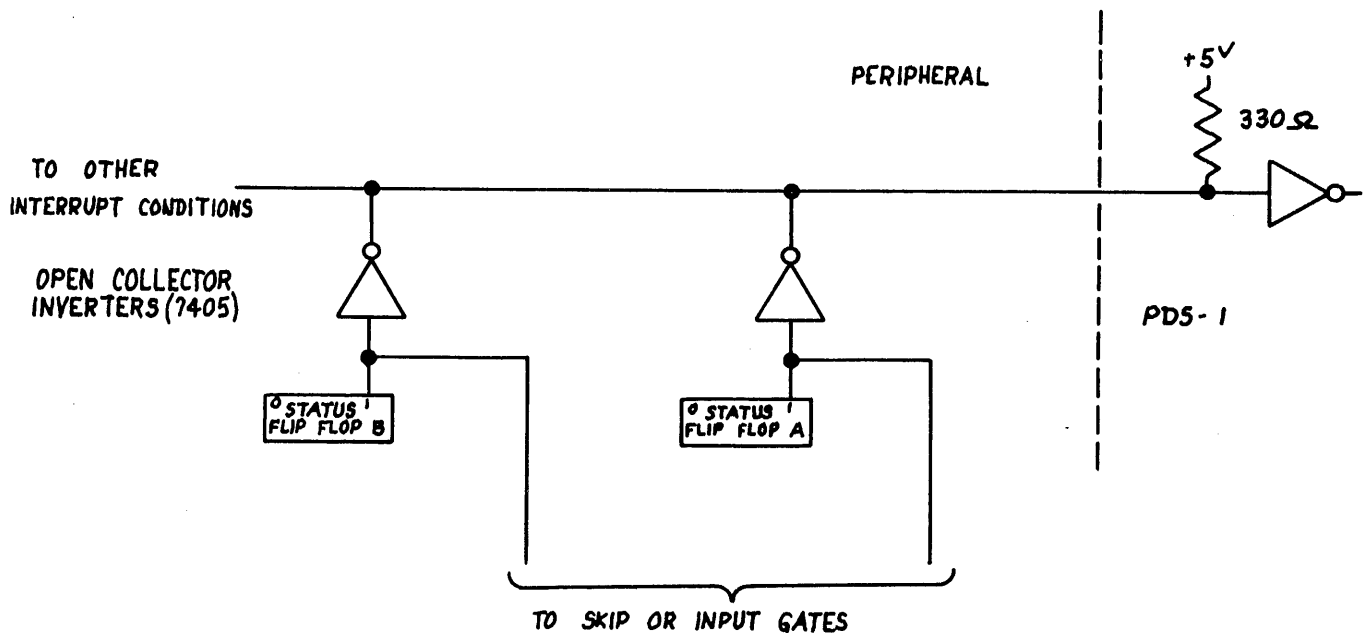


Fig. 7 Logic for an External Interrupt

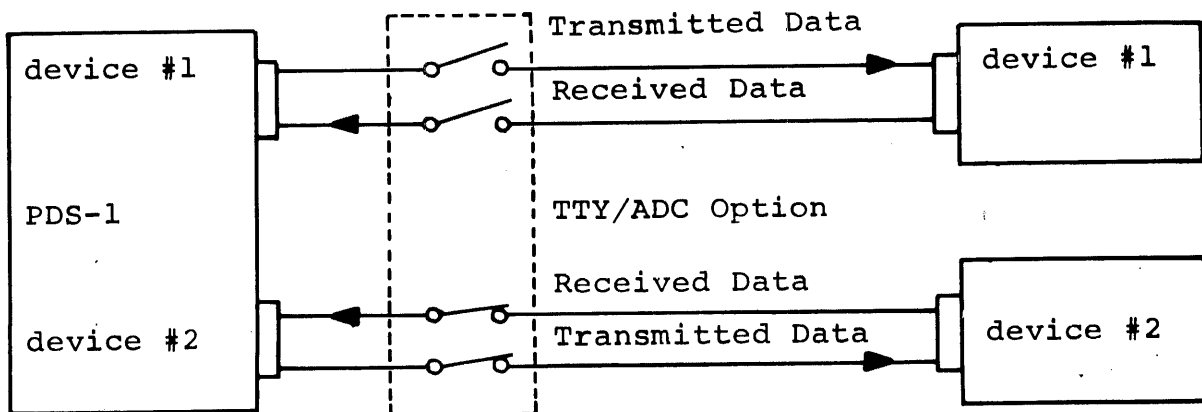
#### IV. ASSOCIATED OPTIONAL DEVICES

##### 3. Print Capabilities

Hard copy reproductions may be produced in the following manner:

1. Local print - Data displayed on the screen may be transferred to a tele-printer.
2. Computer generated print - The CPU may direct a message directly to the printer and not destruct any data that is on the screen.
3. TTY/ADC switch option - The PDS-1 may be equipped with a TTY/ADC (acoustic data coupler) switch option. This option provides the operator with control of a serial interface and eliminates the simultaneous transmission or reception of data by both devices. When both a TTY and an ADC are interfaced with a PDS-1, and the switch option is employed, the operator can communicate with either device.

Normally a 110 baud serial interface is required to interface a TTY to a PDS-1. Because the ADC and the TTY share this interface, the send/receive speed through the ADC is limited. If a 30 character/second printer is used a 300 baud serial interface can be shared. If other speeds are desired, and a mismatch exists, separate interfaces may be bought as options.



#### IV. ASSOCIATED OPTIONAL DEVICES

##### 4. Lightpen Option

A lightpen is available to PDS-1 users for specific graphics or text interactive modes of operation where the keyboard is too slow for the desired human response times.

Hardware prerequisites for the lightpen option are the expanded I/O (IOE-1) and the external interrupt option (ICA-1). The addition of the lightpen includes a Lightpen Buffer Register (LBR) and appropriate operate commands.

When a lightpen "strike" occurs, the computer is interrupted and jumps to the flag service routine which identifies it as a lightpen interrupt. The hardware interface for the lightpen stores the contents of the Display Program Counter (DPC) into the LPR, unless the display program is in the process of executing a display processor subroutine. In this case it waits until completion of the subroutine and returns to the main display list before storing the contents of DPC in the LPR.

Upon receiving the lightpen interrupt the user program can now identify where in the display list it was interrupted and then functionally determine what to do next. Eventually, the user program needs to know where in the main program it was interrupted so that it can resume processing. This is accomplished by the equivalent of a subroutine jump (JMS) to 0000 where the contents of the program counter (PC0) was stored at interrupt time.

## V. PROCESSOR DESCRIPTION

## V. PROCESSOR DESCRIPTION

The flexibility of the IMLAC system derives from its central processing unit. This unit incorporates 4,096 words of 16-bit core storage, some of which is used for display generation and display information storage purposes (Table IV-1). Its order code is comparable to those of other current 16-bit general purpose computers and is quite flexible. An assembly program and numerous utility routines offer the programmer flexibility in operation and debugging of programs.

The PDS-1 operates as a dual processor machine: the display processor steals memory cycles. The use of a common memory provides great flexibility in operation. See Table IV-2 for a list of the instruction repertoire. Tables IV-3 through IV-9 and Figures IV-1 through IV-8 give a more detailed explanation of the processor and display instructions and decoding of bytes.

Table IV-3 describes the action of the CPU orders with appropriate register content changes. The operate class commands are explained by Table IV-4 and Figure IV-1 which shows the bit by bit decoding of an operate class command. Input/output commands are described by Figure IV-2 and Table IV-5; note that some of these instructions are used for other than input/output purposes. The SKIP decoding is illustrated by Figure IV-3 and Table IV-6. Arithmetic shifts and logical rotate instructions are detailed by Figure IV-4, Table IV-7 and associated paragraphs. Figure IV-5 through IV-8 and Tables IV-8 and IV-9 comprise the description of the Display Processor Orders and Byte decoding.

Table V-1

## CENTRAL PROCESSOR HIGHLIGHTS

MEMORY	DESCRIPTION
Word Size (bits)	16
Memory Size (words)	4K - 32K optional
Cycle Time (USEC)	2.0
Memory Parity	no
Memory Protect	optional-sections
Direct Addressing (words)	2K
Indirect Addressing (set bit zero)	32K Single Level (multi-level optional)
CPU	
General Purpose Registers	1 + Link bit for double precision
Index Registers	8 (auto index) in each 2K of memory. $10_8 \rightarrow 17_8$ for example $4010 \rightarrow 4017_8$
Hardware Multiply-Divide	no
Immediate Instructions	yes
Double-Word Instructions	no
Byte Processing	no
I/O	
I/O Word Size (bits)	16
Priority Interrupt Levels	1, optional
Direct Memory Access Channel	optional
I/O Maximum Word Rate (KHz)	500
OTHER FEATURES	
Real Time Clock	optional
Power Fail/Restart	yes
Largest Disk (megabits)	optional
Assembler	2 pass



Table V-2

PROCESSOR ORDERS

OPR N 000  
 LAW N 004  
 JMP 010  
 014  
 DAC 020  
 XAM 024  
 ISZ 030  
 JMS 034  
 040  
 AND 044  
 IOR 050  
 XOR 054  
 LAC 060  
 ADD 064  
 SUB 070  
 SAM 074

Note: Bit 0 indicates  
 Indirect Address  
 Mode for all  
 processors except  
 LAW

OPERATE

HLT 000 XXX Halt  
 NOP 100 000 No operation  
 CLA 100 001 Clear AC  
 CMA 100 002 1's comp. AC  
 STA 100 003 CLA & comp.  
 IAC 100 004 inc. AC  
 COA 100 005 +1=>C(AC)  
 CIA 100 006 2's comp. AC  
 CLL 100 010 Clear L  
 CML 100 020 comp. L  
 STL 100 030 set L  
 ODA 100 040 IOR AC with DS  
 LDA 100 041 C(DS)=>C(AC)  
 CAL 100 011 CLA & CLL

SHIFT & ROT.

RAL N 003 00N  
 RAR N 003 02N  
 SAL N 003 04N  
 SAR N 003 06N  
 DON 003 100 display on

SKIP IF

102 000  
 ASZ 002 001 C(AC) = 0  
 ASN 102 001 C(AC) ≠ 0  
 ASP 002 002 C(AC) +  
 ASM 102 002 C(AC) -  
 LSZ 002 004 C(L) = 0  
 LSN 102 004 C(L) = 1  
 DSF 002 010 Display is on  
 DSN 102 010 Display is off  
 KSF 002 020 Keyb. on flag  
 KSN 102 020 Keyb. on no flag  
 RSF 002 040 TTY has input data  
 RSN 102 040 TTY has no input  
 TSF 002 100 TTY done sending  
 TSN 102 100 TTY not done sending  
 SSF 002 200 40~sync. on  
 SSN 102 200 40~sync. off  
 HSF 002 400 PTR has data  
 HSN 102 400 PTR has no data

IOT

DLZ 001 001 0=>C(DPC)  
 001 002 C(AC) XOR C(DPC)=>C(DPC)  
 DLA 001 003 C(AC)=>C(DPC)  
 CTB 001 011 Clear TTY Break  
 DOF 001 012 Display off  
 KRB 001 021 Keyb. read  
 KCF 001 022 Keyb. clear flag  
 KRC 001 023 Keyb. read & clear  
 RRB 001 031 TTY read  
 RCF 001 032 clear input TTY  
 RRC 001 033 TTY read & clear  
 TPR 001 041 TTY transmit  
 TCF 001 042 clear TTY output status  
 TPC 001 043 TTY print & clear flag  
 HRB 001 051 read PTR  
 HOF 001 052 STOP PTR  
 HON 001 061 START PTR  
 STB 001 062 Set TTY Break  
 SCF 001 071 clear 40/sec sync.  
 IOS 001 072 IOT sync.  
 IOF 001 161 Disable Int. (optional)  
 ION 001 162 Enable Int. (optional)  
 PPC 001 271 Punch&ClearPunchFlag  
 (optional)  
 PSF 001 274 SKIP ifPunchReady  
 (optional)

DISPLAY OPR.

D HLT 000 000 Display Halt  
 D NOP 004 000 Display no operate  
 004 004 set scale = 1/2  
 D STS1 004 005 set scale = 1  
 D STS2 004 006 set scale = 2  
 D STS4 004 007 set scale = 3  
 D RJM 004 040 C(DT)=>C(DPC)  
 D LXM 005 000 inc. XAC  
 D IYM 004 400 inc. YAC  
 D DXM 004 200 dec. XAC  
 D DYM 004 100 dec. YAC  
 D HVC 006 000 H.V. sync. & cont.  
 D HVH 002 000 H.V. sync. & Hlt  
 D DSP 004 020 SinglePointIntensify

DISPLAY ORDERS

D OPR 00  
 D LXA N 01 N=>C(XAC)  
 D LYA N 02 N=>C(YAC)  
 D EIM N 03  
 04  
 D JMS 05  
 D JMP 06

Table V-3

## 1. PROCESSOR INSTRUCTIONS

BIT NO. 1 2 3 4	DECIMAL VALUE	OCTAL VALUE	MNEMONIC	DESCRIPTION
0 0 0 0	0	0 0	OPR N	Operate Class-see Tables IV - 4,5,6,7
0 0 0 1	1	0 4	LAW N LWC N	BIT 0=0; N $\Rightarrow$ C(AC) BIT 0=1; -N $\Rightarrow$ C(AC)
0 0 1 0	2	1 0	JMP Q	Jump to Q: Q $\Rightarrow$ C(PC)
0 0 1 1	3	1 4	-	Not used
0 1 0 0	4	2 0	DAC Q	C(AC) $\Rightarrow$ C(Q)
0 1 0 1	5	2 4	XAM Q	C(AC) $\Leftrightarrow$ C(Q)
0 1 1 0	6	3 0	IS2 Q	C(Q) +1 $\Rightarrow$ C(Q) if result = 0 skip next instruction
0 1 1 1	7	3 4	JMS Q	C(PC) +1 $\Rightarrow$ C(Q); Q+1 $\Rightarrow$ C(PC)
1 0 0 0	8	4 0	-	Not used
1 0 0 1	9	4 4	AND Q	AND AC bit by bit with C(Q)
1 0 1 0	10	5 0	IOR Q	Inclusive OR AC bit by bit with C(Q)
1 0 1 1	11	5 4	XOR Q	Exclusive OR AC bit by bit with C(Q)
1 1 0 0	12	6 0	LAC Q	C(Q) $\Rightarrow$ C(AC)
1 1 0 1	13	6 4	ADD Q	C(AC) + C(Q) $\Rightarrow$ C(AC)
1 1 1 0	14	7 0	SUB Q	C(AC) - C(Q) $\Rightarrow$ C(AC)
1 1 1 1	15	7 4	SAM Q	if C(AC) = C(Q) skip next instruction

BIT NUMBER															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T3	0=HLT	1=Continue								T2 DS $\Rightarrow$ AC	T2 COMP LINK	T1 CLR LINK	T3 +1 to AC	T2 COM AC	T1 CLR AC

Fig. V-1 Operate Word Decoding

a. OPERATE INSTRUCTIONS

Table V-4

<u>OCTAL</u>	<u>MNEMONIC</u>	<u>DESCRIPTION</u>
000 XXX	HLT	Halt after this command
100 000	NOP	No Operation
100 001	CLA	Clear AC
100 002	CMA	Complement AC (1's Complement)
100 003	STA	Clear & Comp. AC: i.e., $-1 \Rightarrow C(AC)$
100 004	IAC	Add 1 to C(AC)
100 005		$+1 \Rightarrow C(AC)$
100 006	CIA	Complement and Increment AC (2's Complement)
100 010	CLL	Clear Link
100 020	CML	Complement Link
100 030	STL	Set Link
100 040	ODA	IOR AC with Data Switches (DS)
100 041	LDA	$C(DS) \Rightarrow C(AC)$



BIT NUMBERS															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Complement Meaning	0	0	0	0	1	0									
							PTR Data Present	40 Cycle Sync.	TTY Output Ready	TTY Input Present	KBD Input Present	Display On	C(L) = 0	C(AC) ≥ 0	C(AC) = 0

Fig. V -3 Skip Instruction Decoding

Table V-6

c. SKIP INSTRUCTIONS

<u>OCTAL</u>	<u>MNEMONIC</u>	<u>SKIP IF DESCRIPTION</u>
102 000		
002 001	AS2	C(AC) = 0
102 001	ASN	C(AC) ≠ 0
002 002	ASP	C(AC) is +
102 002	ASM	C(AC) is -
002 004	LS2	C(L) = 0
102 004	LSN	C(L) = 1
002 010	DSF	Display is on
102 010	DSN	Display is off
002 020	KSF	Keyboard on flag
102 020	KSN	Keyboard on no flag
002 040	RSF	TTY has input data
102 040	RSN	TTY has no input data
002 100	TSF	TTY has completed sending
102 100	TSN	TTY has not completed sending
002 200	SSF	40 ~ sync. on
102 200	SSN	40 ~ sync. off
002 400	HSF	PTR has Data
102 400	HSN	PTR has no Data

BIT NUMBER															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	1	1									
									1=Disp.	0=Rotate 1=Shift	0=Left 1=Right			Shift 2	Shift 1

Fig. V-4 Shift & Rotate Decoding

Shift does not affect the link bit or bit 0 of the accumulator. Shift right puts the sign bit into bit 1. Shift left puts 0 into bit 15 of the accumulator.

Rotate includes the link bit and bit 0 of the accumulator. Rotate left puts bit 0 into link and link into AC bit 15.

Table V-7

d. SHIFT & ROTATE INSTRUCTIONS

<u>OCTAL</u>	<u>MNEMONIC</u>	<u>DESCRIPTION</u>
003 00N	RAL N	Rotate left N times
003 02N	RAR N	Rotate right N times
003 04N	SAL N	Shift left N times
003 06N	SAR N	Shift right N times
003 100	DON	Display On

Note: N may equal 1, 2 or 3

Table V-8

2. DISPLAY INSTRUCTIONS

BIT NO.				OCTAL	MNEMONIC	DESCRIPTION
0	1	2	3			
0	0	0	0	0 0	D OPR	
0	0	0	1	0 1	D LXA N	N => C(XAC)
0	0	1	0	0 2	D LYA N	N => C(YAC)
0	0	1	1	0 3	D EIM N	Enter Inc Mode N is First Byte
0	1	0	0	0 4		
0	1	0	1	0 5	D JMS Q	Q => C(DPC); C(DPC)+1=>C(DT
0	1	1	0	0 6	D JMP Q	Q => C(DPC)
0	1	1	1	0 7		

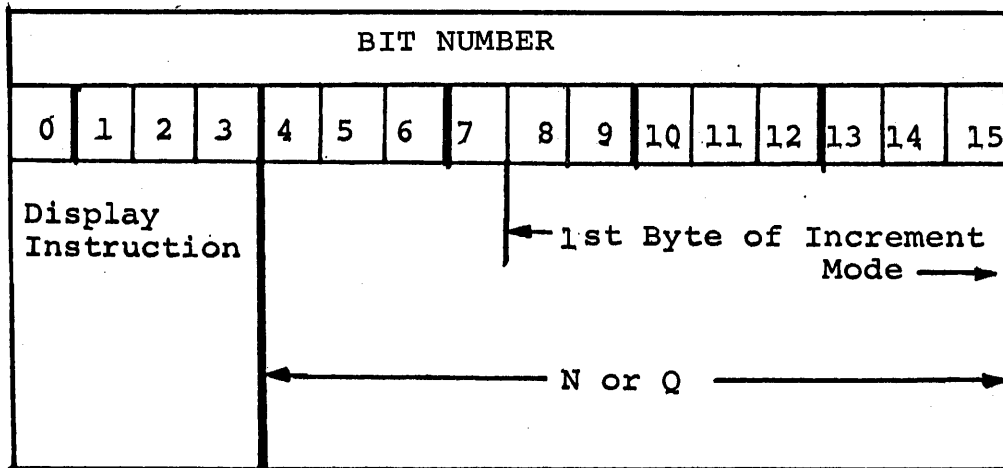


Fig. V-5. Display Instruction Decoding

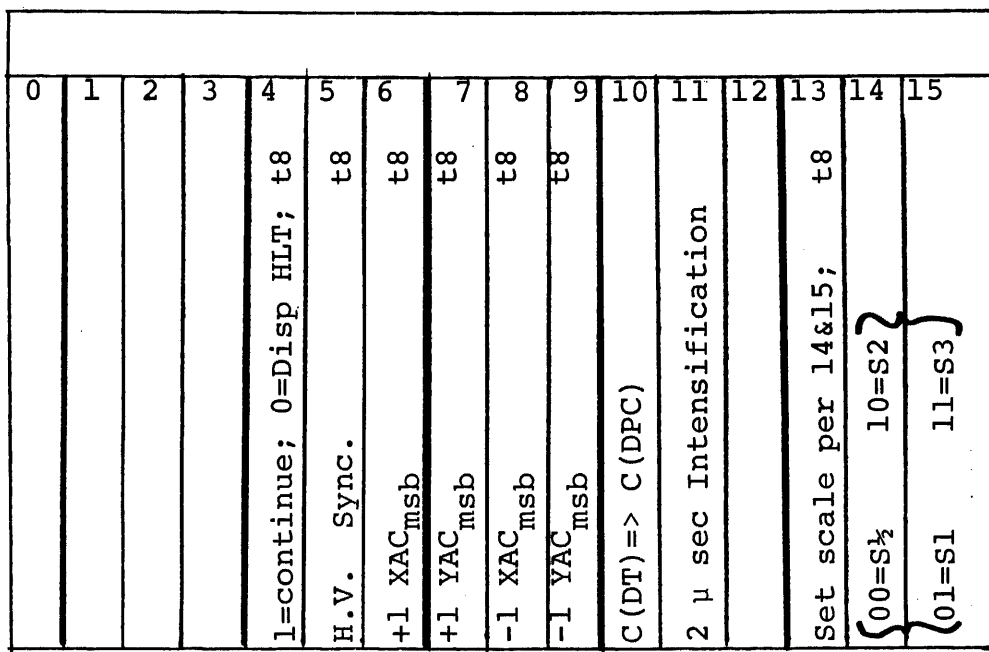


Fig. V-6 Display Operate Decoding

Table V-9

a. DISPLAY OPERATE INSTRUCTIONS		
OCTAL	MNEMONIC	DESCRIPTION
000 000	D HLT	Display Halt
004 000	D NOP	Display No Operate
004 004		Set Scale= $\frac{1}{2}$
004 005	D STS 1	Set Scale=1
004 006	D STS 2	Set Scale=2
004 007	D STS 3	Set Scale=3
004 020	D DSP	2 μ sec Intensification
005 000	D IXM	Inc XAC <sub>msb</sub>
004 040	D RJM	C(DT)=>C(DPC)
004 400	D IYM	Inc YAC <sub>msb</sub>
004 200	D DXM	Decrement XAC <sub>msb</sub>
004 100	D DYM	Decrement YAC <sub>msb</sub>
006 000	D HVC	H.V. Sync & Continue
002 000	D HVH	H.V. Sync & Hlt



b. BIT NUMBERS							
8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7
1 = Increment	1 = Beam on 0 = Beam off	0 = + $\Delta X$ 1 = - $\Delta X$	$\Delta X = 2$	$\Delta X = 1$	0 = + $\Delta Y$ 1 = - $\Delta Y$	$\Delta Y = 2$	$\Delta Y = 1$

Fig. V-7 Increment Byte Decoding

b. BIT NUMBERS							
8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7
0 = non-increment	Escape i.e. { Set Disp BRK CLR inc Mode	C(DT) $\Rightarrow$ C(DPC)	+1 to C(XAC) MSB	0 $\Rightarrow$ C(XAC) LSB		+1 to C(YAC) MSB	0 $\Rightarrow$ C(YAC) LSB

Fig. V-8 Control Byte Decoding

## VI. COMPUTER SOFTWARE PACKAGES

## VI. COMPUTER SOFTWARE PACKAGES

### 1. Text and Graphics Editor

#### Editing Capabilities

##### Insert and Delete

These editing functions permit the operator to return to a specific character location and either insert or delete characters without requiring a complete reconstruction of the text.

##### Insert

The non-destructive cursor is placed under the position where the desired character is to be placed. Upon "insert" of the characters all the succeeding data on the line moves to right.

Example:    Insrt    <sup>te</sup> →    Insert

This motion is automatic after cursor placement and the depression of the appropriate key.

##### Delete

In a similar fashion to the Insert operation, the cursor is placed under the character to be removed. Upon deletion of the character the text would move to the left. The action occurs upon depression of the "Delete" key.

Example:    Delette    →    Delete

### Format

This feature gives the operator the ability to construct a form in which to place variable data. The format is developed via format mode. The use of such a technique prohibits the operator from constructing and transmitting unacceptable messages. The format mode permits use of the standard editing functions-Insert, Delete and Cursor movement-during formation of the form.

Example:

<u>N</u>	<u>A</u>	<u>M</u>	<u>E</u>																
<u>A</u>	<u>D</u>	<u>D</u>	<u>R</u>	<u>E</u>	<u>S</u>	<u>S</u>													
<u>C</u>	<u>I</u>	<u>T</u>	<u>Y</u>								<u>Z</u>	<u>I</u>	<u>P</u>						
<u>A</u>	<u>G</u>	<u>E</u>																	

### Blocking Technique

Using the format concept create two blocks of data. The format can be considered constant and the input data developed by the operator as Variable. Most applications will necessitate that only the variable data be transmitted while the format remains on the display ready for the next entry.

Example:

<u>N</u>	<u>A</u>	<u>M</u>	<u>E</u>		<u>B</u>	<u>I</u>	<u>L</u>	<u>L</u>		<u>L</u>	<u>I</u>	<u>T</u>	<u>T</u>	<u>L</u>	<u>E</u>	<u>F</u>	<u>I</u>	<u>E</u>	<u>L</u>	<u>D</u>				
<u>A</u>	<u>D</u>	<u>D</u>	<u>R</u>	<u>E</u>	<u>S</u>	<u>S</u>		2	9	0		<u>M</u>	<u>A</u>	<u>I</u>	<u>N</u>		<u>S</u>	<u>T</u>						
<u>C</u>	<u>I</u>	<u>T</u>	<u>Y</u>		<u>B</u>	<u>O</u>	<u>S</u>	<u>T</u>	<u>O</u>	<u>N</u>						<u>Z</u>	<u>I</u>	<u>P</u>	1	9	0	0	6	
<u>A</u>	<u>G</u>	<u>E</u>		4	0																			

### Other Edit Capabilities

Line insert/delete	-	yes
Line Shift	-	optional
Split-screen	-	optional
Tabulation	-	horizontal and vertical
Repeat	-	yes
Graphics	-	36 directions and sizes line, and invisible vectors

#### a. Keyboard Operations

Tables VI-1 -- VI-5 describe the operations controlled from the keyboard and how they affect data transmission, data entry, mode entry etc. The flexible design of the keyboard permits an easy transition from any electro-mechanical on-line terminal. The thrust and key touch enable an operator to use touch-type procedures instead of that usually associated with teletypes. All the keys used for editing, transmission, and cursor movement are separated from the standard key layout by two (2) key spaces. This eliminates accidental moves on the operator's part.

Table VI-1

DISPLAYABLE CHARACTERS (See Table VI-6 For Graphics)	
<u>Desired Char. or Action</u>	<u>Key or Combination of Keys</u>
Upper or Lower Case Char.	That Key with or without Shift
Shift Lock (upper case)	CTRL, REPT, U
Shift Lock (lower case)	CTRL, REPT, L
[	Shift and Control and A
\	Shift and Control and B
]	Shift and Control and C
↑	Shift and Control and D
←	Shift and Control and E
@	Shift and Control and F
Clear Screen	Shift and Control and Delete

b. Table VI-2

CURSOR CONTROL	
<u>Desired Char. or Action</u>	<u>Key or Combination of Keys</u>
Move Cursor Right	→
Move Cursor Left	←
Move Cursor Up One Line	↑
Move Cursor Down One Line	↓
Move Cursor to Upper Left Hand Corner of Screen	Home
Move Cursor to Bottom of File	Control and Repeat and B
Move Cursor to Center of Screen	Control and Repeat and C

c. Table VI-3

CONTROL CHARACTERS	
<u>Desired Char. or Action</u>	<u>Key or Combination of Keys</u>
Control (A-Z) (Transmitted Immediately)	Control and (A-Z)
Control Shift (K-P) (Transmitted Immediately)	Control and Shift and (K-P)
Alt. Mode (Transmitted Immediately)	Control and Shift and G
Blank (Leader Trailer) (Transmitted Immediately)	FF

d. Table VI-4

TRANSMISSION CONTROL	
<u>Desired Char. or Action</u>	<u>Key or Combination of Keys</u>
Transmit Entire Page of Data	Page Xmit
Insert CR, Transmit from Previous CR On or from Remote Computer Statement On, whichever is Last	Data Xmit
Stop Transmitting	Control and Repeat and S
Enter Full Duplex Mode (Also called Transmit Immediate Mode)	Control and Repeat and T (Can also be used with Shift Lock for transmission of upper case)
Exit Full Duplex Mode	Control and Repeat and E or Clear Screen
Transmit Break (opens line momentarily)	Shift, and EOM key
Ignore incoming data for 5 sec	EOM key (used to ignore a series of line feeds, etc.)
Read Photo Electric Tape	Control and Repeat and R

e. Table VI-5










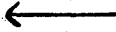
FORM MODE CONTROL	
<u>Desired Char. or Action</u>	<u>Key or Combination of Keys</u>
Clear Form Mode	Control and Repeat and D or Clear Screen
Enter Form Mode	Control and Repeat and F
Transmit Variable Data	Data Xmit (in Form Mode)
Transmit Form + Data	Page Xmit (in Form Mode)
Call Form from Tape or Library	Clear Form Mode Before Calling Form
Call Data From Tape or Library	Clear Form Mode Before Calling Data
Clear Variable Data	Shift and Control and Delete (in Data Mode)
Clear Format	Clear Screen (in Format Mode)
Store Format in Core Memory (This shortens display buffer for Text or Graphic Mode from 1500 to about 1000 characters)	Control and Repeat and P (in Form Mode)
Call Format from Core	Control & Repeat & Q (in Form Mode)
Create Data Fields in Form Mode	Horizontal Right → Cursor Control (If tail begins to form on cursor due to long fields, start again and after delineating the start of the data field, use space bar.)
Contracting Data Fields in Form Mode	Horizontal Left ← Cursor Control
Clear Packed Form (Erase stored format and reini- tialize to text mode. This action also clears screen and is useful in adding buffer storage to display list)	CTRL, REPT, K



f. Graphics

The graphics capability of the PDS-1 Text and Graphics Editor consist of the following keyboard operations:

Table VI-6

GRAPHICS EDITING	
<u>Desired Action</u>	<u>Key Combination</u>
Visible Vector 	CTRL 1
Visible Vector 	CTRL 2
Visible Vector 	CTRL 3
Visible Vector 	CTRL 4
Visible Vector 	CTRL 5
Visible Vector 	CTRL 6
Visible Vector 	CTRL 7
Visible Vector 	CTRL 8
Invisible Vectors	Same as visible but add "shift" key to combination
Delete Vector	Track along graphic display list with cursor keys forward  and  reverse. Then delete appropriate vectors. List will be deleted in reverse direction towards cursor location.
Note: Above vectors are bounded by one character space (unit).	
Repeat Vectors or Deletions by 8	Same as combination above but add REPT key.
Create Independent Graphics Display List or reposition cursor in a graphics file.	Center Cursor(CTRL,REPT,C). Hold CTRL down. Reposition cursor to desired location,create new file.
Clear Screen	CTRL, SHIFT, DELETE
Move a block of graphics display around the screen	Position cursor to beginning of the display list for the graphics of interest. Insert invisible vectors (CTRL,SHIFT,1 through 8) (Space bar also works for shifting graphics to the right. Carriage return also shifts graphics downstream of the cursor). For complex graphics editing the structure of the picture should be separated out to facilitate piecemeal editing without affecting the remainder of the display list.
	(con't.)

TableVI-6 (continued)

GRAPHICS EDITING

Desired Action

Key Combination

Note: The above described graphics vectors of one character unit size can be executed in either text or keyboard graphics mode. The following 2 character size vectors must be drawn in Keyboard Graphics Mode.

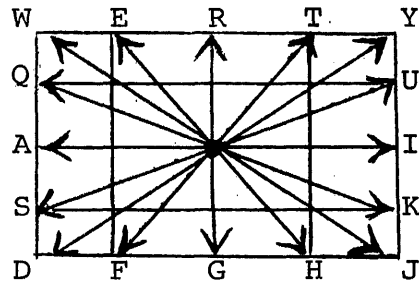
Enter Keyboard Graphics Mode

CTRL, REPT, G

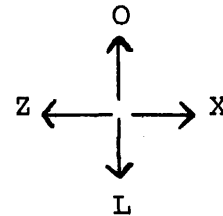
Exit Keyboard Graphics Mode

CTRL, REPT, A

Two character unit vectors in 16 directions



In Keyboard Mode Draw vectors of 8 character unit length along the four major axes



Note: All of the above functions when executed in combination with the REPT key will be multiplied by 8 lengths. The codes generated can be transmitted serially in 8 level code.

Draw centered full screen X, Y axes (undimensional)

CTRL, REPT, X

Table VI-7

SPECIAL COMBINATIONS	
<u>Desired Char. or Action</u>	<u>Key or Combination of Keys</u>
Read 10 lines of text from Paper Tape Reader	CTRL, REPT, W
Single Code for 5 spaces. Tabs are set absolutely in 5 space increments beginning at character 1 in an 80 character line (1, 6, 11, 16, etc.). This number can be changed by software.	TAB key

In order to transmit and receive 8 level Imlac Graphics Code the PDS-1 must be in Graphics Receive Mode. CTRL N (016<sub>8</sub>) places the PDS-1 in this mode. CTRL R (022<sub>8</sub>) exits this mode.

The following table lists the ASCII characters required to receive graphics vectors in Graphics Receive Mode.

ASCII Graphic Character Interpretation (1 unit, 2 units, 8 units or I for invisible)

0	8	S
1	1	E
2	1	SE
3	1	S
4	1	SW
5	1	W
6	1	NW
7	1	N
8	1	NE
9	8	E
:		
;		
<		
=		
>		
?		
@		
A	2	E
B	2	ESE
C	2	SE
D	2	SSE
E	2	S
F	2	SSW
G	2	SW
H	2	WSW
I	2	W
J	2	WNW
K	2	NW
L	2	NNW
M	2	N
N	2	NNE
O	2	NE
P	2	ENE
Q	1I	E
R	1I	SE
S	1I	S
T	1I	SW
U	1I	W
V	1I	NW
W	1I	N
X	1I	NE
Y	8	W
Z	8	N

## 2. Utility Package

### a. Loaders

A given installation site must have either a Programmer/Maintenance Console or a Read Only Memory Bootstrap Loader (ROM) to get on the air.

In the case of the console, the bootstrap must be loaded by toggling-in the instructions manually.

The ROM option offers the convenience of a permanently loaded bootstrap which can be wired to read programs being loaded from either a Teletype, Paper Tape Reader, Acoustic Coupler, EIA Interface, etc.

All operating programs begin at program location 100<sub>8</sub>. The programs are prefaced by a block loader, which is addressed for that particular program's transient area so that when the program runs it is overwritten and does not use up the extra core locations.

The following table shows an example of instructions wired into an ROM specified for Paper Tape Reader Bootstrap. The listing on the following pages is for a Paper Tape Block Loader.

PTR Auto Load From ROM

	PTR AUTO	LOAD (ROM)	2/20/70	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
40	LAC 77	060077			1	1								1	1	1	1	1	1
41	DAC 10	020010				1										1			
42	LWC 76	104076	1					1						1	1	1	1	1	1
43	DAC 20	020020				1										1			
44	HON	001061								1				1	1				1
45	CAL	100011	1													1			1
46	HSF	002400						1			1								
47	JMP 46	010046					1							1			1	1	
50	HRB	001051								1				1		1			1
51	ASN 1	102001	1					1											1
52	JMP 45	010045					1							1			1		1
53	HSF	002400						1			1								
54	JMP 53	010053					1							1		1			1
55	HRB	001051								1				1		1			1
56	RAL 3	003003						1	1										1
57	RAL 3	003003						1	1										1
60	RAL 2	003002						1	1										1
61	HSN	102400	1					1			1								
62	JMP 61	010061					1							1	1				1
63	HSF	002400						1			1								
64	JMP 63	010063					1							1	1				1
65	HRB	001051								1				1		1			1
66	I DAC 10	120010	1				1									1			
67	HSN	102400	1					1			1								
70	JMP 67	010067					1							1	1	1	1	1	1
71	CAL	100011	1													1			1
72	ISZ 20	030020					1								1				
73	JMP 53	010053					1							1	1		1	1	
74	I JMP 76	110076	1				1							1	1	1	1	1	
75																			
76		37700					1	1	1	1	1								
77		37677					1	1	1	1	1	1		1	1	1	1	1	1

## PTR BLOCK LOADER

```
ORG 7700B / PTR BLOCK LOADER SOURCE
HON / START PTR
CAL / CLEAR AC AND LINK
DAC S / DAC SUMCHECK WORD
JMS R1 / JMS READ 1 CHARACTER
ASN / SKIP IF AC NOT 0
JMP *-2
CIA / 2'S COMPLEMENT AC
DAC W / DAC WORD COUNT
JMS R2 / JMS READ 2 CHARACTERS
DAC A / DAC ADDRESS WORD
SAM K / SAM 177777
JMP *+2
HLT / HALT ON END CODE
JMS R2 / JMS READ 2 CHARACTERS
I DAC A / I DAC ADDRESS WORD
JMS S1 / JMS DO SUMCHECK
ISZ A / ISZ ADDRESS WORD
ISZ W / ISZ WORD COUNT
JMP *-5
JMS R2 / JMS READ 2 CHARACTERS
SUB S / SUB SUMCHECK WORD
ASN / SKIP IF AC NOT 0
JMP *-25B / READ ANOTHER BLOCK
HLT / THERE WAS A SUMCHECK ERROR
K OCT 177777B / FOR COMPARING END CODE
S1 BSS 1 / SUMCHECK FORMING SR
CLL / CLEAR LINK
ADD S / ADD SUMCHECK WORD TO C(AC)
LSZ / SKIP IF LINK 0
IAC / INCREMENT AC
DAC S / DAC SUMCHECK WORD
I JMP S1 / RETURN
```

## PTR BLOCK LOADER

```
REL 7740B / RELOCATE TO 7740 FOR BOOTSTRAP AND READING SR'S
HON / START PTR
CAL / CLEAR AC AND LINK
JMS R1 / JMS READ 1 CHARACTER
ASZ / SKIP IF AC 0
JMP R2+3 / JMP INTO ADDRESS FORMING SR
JMP *-4
JMS R2 / JMS READ 2 CHARACTERS
H ASN / SKIP IF ADDRESS NOT 0
HLT / HALT ON 0 ADDRESS
DAC A / DAC ADDRESS WORD
JMS R2 / JMS READ 2 CHARACTERS
I DAC A / STORE CONTENTS OF ADDRESS
JMP H-1
R2 ZRO H / RETURN ADDRESS OF READ 2 CHARACTER SR
CAL / CLEAR AC AND LINK
JMS R1 / JMS READ 1 CHARACTER
RAL 3
RAL 3
RAL 2
JMS R1 / JMS READ 1 CHARACTER
I JMP R2 / RETURN
R1 BSS 1 / RETURN ADDRESS OF READ 1 CHARACTER SR
HSN / SKIP IF PTR DOESN'T HAVE DATA
JMP *-1
HSF / SKIP IF PTR HAS DATA
JMP *-1
HRB / READ PTR
I JMP R1 / RETURN
A BSS 1 / PLACE FOR ADDRESS
S BSS 1 / PLACE FOR SUMCHECK WORD
W BSS 1 / PLACE FOR WORDCOUNT
END
```

PTR BLOCK LOADER

SYM  
A 7774  
H 7747  
K 7730  
h1 7765  
h2 7755  
S 7775  
S1 7731  
W 7776

UNU SYM

ERR

LIT

SAV

MAC

ORG 7700  
END 7776  
LAS 7776



PTR BLOCK LOADER

7700	001061		
7701	100011	I	
7702	023775		DAC 7775
7703	037765		JMS 7765
7704	102001		ASN
7705	013703		JMP 7703
7706	100006		CIA
7707	023776		DAC 7776
7710	037755		JMS 7755
7711	023774		DAC 7774
7712	077730		SAM 7730
7713	013715		JMP 7715
7714	000000		HLT
7715	037755		JMS 7755
7716	123774	I	DAC 7774
7717	037731		JMS 7731
7720	033774		ISZ 7774
7721	033776		ISZ 7776
7722	013715		JMP 7715
7723	037755		JMS 7755
7724	073775		SUB 7775
7725	102001		ASN
7726	013701		JMP 7701
7727	000000		HLT
7730	177777		
7731			
7732	100010		CLL
7733	067775		ADD 7775
7734	002004		LSZ
7735	100004		IAC
7736	023775		DAC 7775
7737	113731	I	JMP 7731

PTR BLOCK LOADER

7740	001061	
7741	100011	
7742	037765	JMS 7765
7743	002001	ASZ
7744	013760	JMP 7760
7745	013741	JMP 7741
7746	037755	JMS 7755
7747	102001	ASN
7750	000000	HLT
7751	023774	DAC 7774
7752	037755	JMS 7755
7753	123774	I DAC 7774
7754	013746	JMP 7746
7755	7747	
7756	100011	
7757	037765	JMS 7765
7760	003003	HAL 3
7761	003003	HAL 3
7762	003002	HAL 2
7763	037765	JMS 7765
7764	113755	I JMP 7755
7765		
7766	102400	
7767	013766	JMP 7766
7770	002400	
7771	013770	JMP 7770
7772	001051	
7773	113765	I JMP 7765
7774		
7775		
7776		

b. Hi Speed Assembler

GENERAL INFORMATION

The Hi Speed Assembler is a two pass assembler which reads a paper tape source program and generates a binary coded paper tape object program. It may also be used to generate several printed tables.

It is now programmed to read input from a photoelectric tape reader and print or punch tape on a teletype. However, it would be easy to change it to use other input or output devices.

Two characters, rubout=177 or 377 and SHIFT-CONTROL-P=000 or 200, are ignored wherever they appear on the source tape.

The object tape is in a format compatible with our block loader programs.

There is no limit to the number of characters on a line.

However, there is a limit to most of the tables stored by the assembler:

macro table	≤	435	total instructions
macro call sequence	≤	32	references
literals	≤	192	literals
errors	≤	96	errors
saves	≤	256	saves
duplications	≤	63	instructions to duplicate

If these limits are exceeded the tables will run over into each other.

The Hi Speed Assembler processes the information it receives from the PTR between reading characters, and it stops the PTR only at the end of each pass, and to punch object tape on the second pass. Therefore, at times when the assembler is asked to generate large quantities of instructions from tables it has previously stored, such as in a macro call or after the last statement in a group to be duplicated, it will miss a few characters on the source tape. To keep this from having damaging effects, enter comments at these critical points of the source tape, at the rate of at least one comment character per ten instructions generated.

SYMBOL      D      OPCODE ADDRESS/COMMENT      CR-LF

#### SYMBOL

A letter or a letter and an octal digit 1-7.  
May be replaced by a space.  
Must not be preceded by a space.

#### I or D

I for indirect addressing  
D is a display opcode follows  
May be omitted

#### OPCODE

A three letter code for an instruction or a pseudo opcode

#### ADDRESS

May be:

an octal no.	375
a symbol	A1
a pt. relative address	.-15
a symbol relative address	A3+14
a literal	=321
	=< JMS B7-2
	=#SAV
a save	#XAC
omitted if instruction does not require an address	
(Address field ends with first space after useful information)	

#### COMMENTS

Must be preceded by a space if there is an address field.  
Need not be preceded by a slash but are neater looking if they are.  
May use entire line if slash is first character on line.  
Are terminated by a line feed.

## LITERALS

A literal is used to introduce an octal constant, an address constant, an instruction constant, or a save address into a program without the bother of labeling it and entering it separately into the source program. Just write the octal constant, address constant, "<instruction" constant, or save, preceded by an equal sign, in the address field(s) of the statement(s) in which it is used. The assembler places all literals after the saves of the program and gives a listing of these addresses and the literals assigned to them.

```
AND =177      /mask rt. seven bits
ADD =A3-5     /ADD A3-5 to AC
LAC =< D JMP C /load the instruction "D JMP C" into the AC
LAC =#SAV     /load the address of #SAV into the AC
```

## SAVES

A save is similar to a literal in that it is a way of introducing a saved memory cell into a program without labeling it and entering it separately into the source program. It is useful for reserving memory cells for counters and other variables. Just assign the variable a 3 letter code, and write that code, preceded by a #, in the address field of the statement(s) in which it is used. The assembler places all saves immediately after the last source statement of the program.

```
DAC #XAC     /DAC in word called XAC
```

## PSEUDOS

```
ORG 21      The following statements start at location 21.
             (used at start of source program)

REL A1+2    The following statements start at location A1+2
             (used anywhere in program)

BSS 5       Reserve the 5 following memory words.

REP 12      Repeat the previous instruction 12 times.

ZRO A1+6    Place the address A1+6 in this location.

OCT 175462  Place the octal No. (175462) in this location.

A2 EQU C7-3 Set the symbolic address A2 equal to the address
             C7-3.

END        This is the end of the source program.

DUP 3 6     Duplicate the following 3 statements 6 times.
```

MCD A 14 Use the following 14 statements to define the macro (A).

MCE This is the end of the macro definition.

MCC A 14 B2 177 X =37 < JMS A2 /Call 4 step macro, A

Insert the 14 step macro A using B2 as the address for @1, 177 for @2, X for @3, the address of the literal =37 for @4, and the instruction (JMS A2) following < in place of INS @5.

INS @7 Reserve this instruction position in the macro we are now defining for an instruction to be specified in the 7th information position in the call for this macro.

INC E B3-2 / Interpret (E B3-2) as two increment mode display bytes.

#### INCREMENT MODE

/ This is the end of the increment mode word / necessary

B Turn the beam on for the following increment bytes.

D Turn the beam off for the following increment bytes.

E Enter increment mode.

N Exit increment mode, zero X and Y LSB. (111)

R Exit increment mode, return from D JMS, zero X and Y LSB. (151)

F Exit increment mode, return from D JMS, add one to X MSB, zero X and Y LSB. (171)

P Pause. (200)

A 273 Make this byte 273.

space Ignore.

+ Form byte.

- Form byte.

0 1 2 3 Form byte.

## MACRO INSTRUCTIONS

The Hi Speed Assembler is capable of storing 26 programmer defined macros of not more than 435 total instructions. In defining a macro one can use constant instructions, instructions with point relative addresses, instructions to be specified in the macro call, and instructions with addresses to be specified in the macro call.

Each of these macros can then be called as often as necessary.

On a macro call a slash is needed after all the variable information is specified. If the call sequence takes up more than one line of characters, type SHIFT-N CR-LF, and continue on the next line.

Examples of macro calls and definitions are included with this write up.

## PRINTOUT

The Hi Speed Assembler does not list the source tape or produce an object listing. A source listing can be obtained by running the teletype on local, and an object listing can be obtained by using one of several listing programs.

The Hi Speed Assembler does, however, list several tables at the end of the first pass and also if you push CONTINUE at the end of the second pass.

### SYMBOL TABLE

Omitted if DS bit 15 down

A list of the defined symbols and the addresses assigned to them.

### UNREFERENCED SYMBOL TABLE

Omitted if DS bit 14 down

A list of the defined but unreferenced symbols and their addresses.

### ERROR TABLE

Omitted if DS bit 13 down

A list of the addresses at which errors took place.

### SAVE TABLE

Omitted if DS bit 12 down

A list of the saved addresses and their corresponding three letter codes.

### LITERAL TABLE

Omitted if DS bit 11 down

A list of the addresses assigned to literals and the corresponding constants.

## MACRO TABLE

Omitted if DS bit 10 down

A list of the defined macros, the instructions in each, and information on how to interpret the calling sequence for the particular macro.

## ORIGIN

Omitted if DS bit 9 down

The address of the origin of the program.

## END

Omitted if DS bit 9 down

The address of the last source statement.

## LAST

Omitted if DS bit 9 down

The address of the last save, literal, or source statement.

## ERRORS

The Hi Speed Assembler recognizes several kinds of errors and lists the addresses at which these errors occurred at the end of pass 1, and pass 2 if you push CONTINUE\*. However, the second pass may be made even if errors were found on the first pass; NOP's will be assembled wherever the source tape has an error.

Usually errors are obvious, but here are some that are not:

Referencing an undefined symbol - (This type of error will only show up on second pass symbol table)

Not putting D before a display command

Having non-printing control characters on the source tape

Trying to define a symbol twice

\*It is a good idea to get a table printout after the second pass as some errors are not recognized until the second pass and the origin, end, last addresses give a good indication as to whether or not your program was interpreted the same on both passes.



## SEPARATE TAPES

If the source tape is in several sections, square off the ends and run them through separately. Push start for the first section and continue for each successive section.



TO USE - Load with either block loader.

Load source tape in PTR

Switch PTR to RUN

Switch TTY to ON LINE

Start computer at 100

When computer encounters END instruction it will type out all tables.

Reload source tape in PTR

Turn punch on (leader will be punched)

Push CONTINUE for pass two

Turn punch off

Push CONTINUE for another set of tables (no symbol table is printed on second pass)

## RECOGNIZED MNEMONICS

HLT	000000	halt
NOP	100000	no operation
CLA	100001	clear AC
CMA	100002	1's complement AC
STA	100003	set AC to all 1's ==-1
IAC	100004	increment AC
COA	100005	+1 to AC
CIA	100006	2's complement AC
CLL	100010	clear link
CML	100020	complement link
STL	100030	set link to 1
ODA	100040	inclusive or data switches to AC
LDA	100041	load data switches into AC
CAL	100011	clear AC and link
DON	003100	start display
DLZ	001001	load 0 into display PC
DLA	001003	load C(AC) into display PC
CTB	001011	clear TTY break
DOF	001012	stop display
KRB	001021	keyboard read
KCF	001022	keyboard clear flag
KRC	001023	keyboard read and clear flag
RRB	001031	TTY read
RCF	001032	TTY clear input flag
RRC	001033	TTY read and clear flag
TPR	001041	TTY print
TCF	001042	TTY clear output flag
TPC	001043	TTY print and clear flag
HRB	001051	PTR read
HOF	001052	stop PTR
HON	001061	start PTR
STB	001062	set TTY break
SCF	001071	clear 40 cycle sync
IOS	001072	IOT sync
ASZ	002001	AC, skip if 0
ASN	102001	AC, skip if not 0
ASP	002002	AC, skip if positive
ASM	102002	AC, skip if negative
LSZ	002004	link, skip if 0
LSN	102004	link, skip if 1
DSF	002010	skip if display on
DSN	102010	skip if display off
KSF	002020	skip if keyboard has char
KSN	102020	skip if keyboard doesn't have char
RSF	002040	skip if TTY has char
RSN	102040	skip if TTY doesn't have char
TSF	002100	skip if TTY done printing
TSN	102100	skip if TTY not done printing
SSF	002200	skip if 40 cycle sync on
SSN	102200	skip if 40 cycle sync off
HSF	002400	skip if PTR has char
HSN	102400	skip if PTR doesn't have char

RAL	00300	rotate AC and link left (1, 2, or 3)
RAR	00302	rotate AC and link right (1, 2, or 3)
SAL	00304	shift AC left (1, 2, or 3)
SAR	00306	shift AC right (1, 2, or 3)
LAW	004	load AC with
LWC	104	load AC with 2's complement of
JMP	010	jump to
DAC	020	deposit AC in
XAM	024	exchange AC and memory with
ISZ	030	index memory and skip if zero
JMS	034	jump subroutine
AND	044	and AC with contents of
IOR	050	inclusive or AC with contents of
XOR	054	exclusive or AC with contents of
LAC	060	load AC with contents of
ADD	064	add to AC contents of
SUB	070	subtract from AC contents of
SAM	074	skip if AC is same as contents of
OPR	100	operate
IOT	001	10 transfer
D HLT	000000	stop display
D NOP	004000	display no operation
D IXM	005000	increment X AC MSB
D IYM	004400	increment Y AC MSB
D DXM	004200	decrement X AC MSB
D DYM	004100	decrement Y AC MSB
D HVC	006000	high voltage sync, continue
D HVH	002000	high voltage sync, halt
D RJM	004040	display subroutine return
D DSP	004020	2 $\mu$ sec intensify
D OPR	004	display operate
D LXA	01	load X AC
D LYA	02	load Y AC
D EIM	03	enter increment mode
D JMS	05	display jump SR
D JMP	06	display jump
D STS	004004	display set scale ( $\frac{1}{2}$ , 1, 2, or 3)

/ MACRO DEFINITION AND CALL

MCD X 4 / DEFINE 4 STEP MACRO, X, TO TRANSFER AND OPERATE ON NUMBERS  
LAC @1 / LAC ADDRESS SPECIFIED IN @1 FIELD OF MACRO CALL  
AND =177400B / GET RID OF RT. HALF OF WORD  
INS @2 / PERFORM INSTRUCTION SPECIFIED IN @2 FIELD OF CALL  
DAC @3 / DAC IN ADDRESS SPECIFIED BY @3 FIELD OF CALL  
MCE / END OF MACRO DEFINITION  
.  
.  
.  
.

S MCC X 4            A            < IOR =100000B            #XAC        / CALL  
/ 4 STEP MACRO, X, WITH THE SPECIFIED @ FIELDS (THE ORDER OF THE  
/ FIELDS IS IMPORTANT, BUT THE SPACING BETWEEN THEM IS NOT)  
/ ALL FORMS OF ADDRESSING ARE PERMISSIBLE IN THE @ FIELDS EXCEPT  
/ POINT RELATIVE ADDRESSING  
/ THE / AT THE END OF THE CALL SEQUENCE IS NECESSARY  
/ A "<" MUST PRECEDE THE INFORMATION TO FILL IN AN INS. IN AN @ FIELD

/ INCREMENT MODE

/ SOURCE CODING TO DRAW AN A  
REL 5010 / RELOCATE TO 5010 FOR A SUBROUTINE  
INC EB13 / ENTER INCREMENT MODE    TURN BEAM ON DRAW 1,3  
INC 1312 / DRAW 1,3    THEN 1,2  
INC 1 - 3            1 -            3        /DRAW 1,-3 TWICE (SPACES DON'T MATTER)  
INC 1-3D-13 / DRAW 1,-3 TURN BEAM OFF DRAW -1,3  
INC -11B-30 / DRAW -1,1 TURN BEAM BACK ON DRAW -3,0  
INC FF / FULL ESCAPE (2 F'S ARE USED AS BOTH BYTES OF WORD  
/ MUST BE FILLED)  
/ MORE CARE IN SPACING AND FEWER COMMENTS COULD MAKE THIS SOURCE  
/ PRINTOUT VERY USEFUL DOCUMENTATION

/ EXAMPLES OF DIFFERENT FORMS OF ADDRESSING

A LAC 10B / ABSOLUTE  
SAM =177B / LITTERAL WITH OCTAL CONSTANT  
JMP .+17B / POINT RELATIVE  
JMS A-62B / SYMBOL RELATIVE  
LAC A / SYMBOLIC  
ADD #REG / SAVED  
I SAM B / INDIRECT SYMBOLIC  
B IOR =A+1 / LITTERAL WITH ADDRESS CONSTANT  
/ B AFTER A NUMBER MEANS THAT IT IS AN OCTAL NUMBER

c. Debuggers

A debugger package for the IMLAC PDS-1 has utility features which allow a programmer to read any core location on to the screen and change it from the keyboard. With a TTY he can get a hard copy core dump in specified block sizes.

Another program is a trace program which allows the user to trace through a program step by step and get the octal representation of the various core locations on the screen.

d. Simple Display Program, Vector Description of letter "d"

*CENTER*

```

ORG 100B / PROGRAM TO DISPLAY HELLO IN UPPER-LEFT OF SCREEN
DOF / STOP DISPLAY
R DSN / SKIP IF DISPLAY OFF      RESTART POINT
  JMP .-1
  SSF / SKIP IF 40 CYCLE SYNC ON
  JMP .-1
  SCF / CLEAR 40 CYCLE SYNC
  LDA / DS GO TO AC
  AND =100000
  DAC .+1
  BSS 1 / HALT OR CONTINUE , DEPENDING ON DS 0
  LAW D / LAW START ADDRESS OF DISPLAY ROUTINE
  DLA / C(AC) GO TO C(DPC)
  DON / START DISPLAY
  JMP R / WAIT FOR NEXT CYCLE OF DISPLAY
D D LXA 0 / LOAD 0 INTO X DISPLAY ACCUMULATOR
D LYA 0 / LOAD 0 INTO Y AC.
D STS 2 / SET SCALE 2
D HVC / HIGH VOLTAGE SYNC AND CONTINUE
D JMS H / DISPLAY JMS TO H CHARACTER DESCRIPTION
D JMS E /      "      E      "
D JMS L /      "      L      "
D JMS L /      "      L      "
D JMS O /      "      O      "
D LXA 4000 / CENTER BEAM TO MINIMIZE LOAD ON DEFLECTION AMPLIFIERS
D LYA 4000
D HLT / STOP DISPLAY
H INC E B03 / H CHARACTER DESCRIPTION
  INC 03 02 /
  INC D30 30 /
  INC B0-3 0-3 /
  INC 0-2 D03 /
  INC 01 B-30 /
  INC -30 F /
E INC E B03 / E CHARACTER DESCRIPTION
  INC 03 02 /
  INC 30 30 /
  INC D-1-3 -1-1 /
  INC B-30 -10 /
  INC D0-3 0-1 /
  INC B30 30 /
  INC F F / (NEEDED 2 BYTES TO FILL OUT WORD)
L INC E B03 / L CHARACTER DESCRIPTION
  INC 03 02 /
  INC A 1 P /
  INC 30 30 /
  INC F F /
O INC E D02 / O CHARACTER DESCRIPTION
  INC B03 23 /
  INC 20 2-3 /
  INC 0-3-2-2 /
  INC -20 -22 /
  INC F F /
END

```

# SIMPLE DISPLAY PROGRAM

SYM

D	116
E	141
H	132
L	151
O	156
R	101

UNU SYM

ERR

LIT

164	100000
-----	--------

SAV

MAC

ORG	100
END	163
LAS	164

# SIMPLE DISPLAY PROGRAM

0100	001012	DOF
0101	102010	DSN
0102	010101	JMP 0101
0103	002200	
0104	010103	JMP 0103
0105	001071	
0106	100041	LDA
0107	044164	AND 0164
0110	020111	DAC 0111
0111	000000	HLT
0112	004116	LAW 0116
0113	001003	DLA
0114	003100	DON
0115	010101	JMP 0101
0116	010000	
0117	020000	
0120	004006	
0121	006000	
0122	050132	
0123	050141	
0124	050151	
0125	050151	
0126	050156	
0127	014000	
0130	024000	
0131	000000	HLT



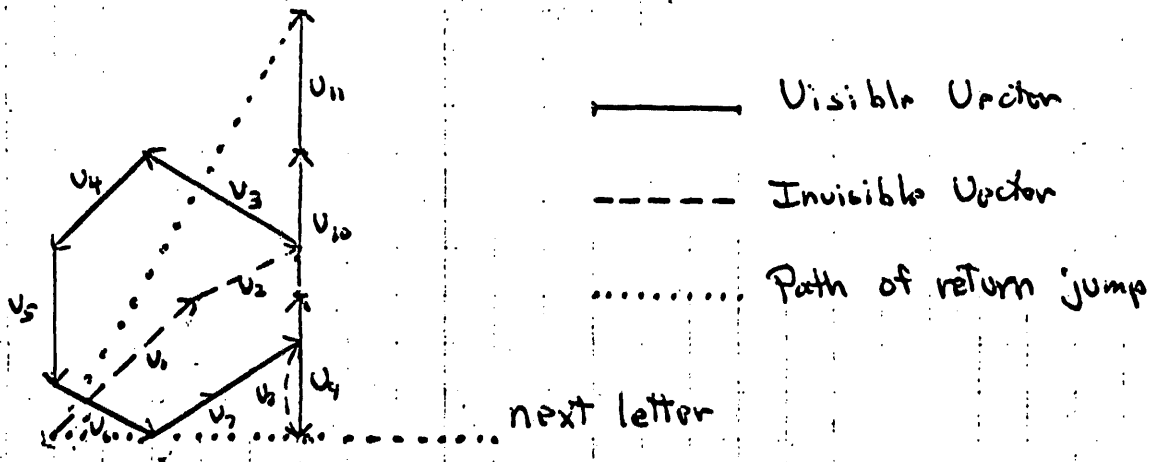
# SIMPLE DISPLAY PROGRAM

0132	030303	ME03
0133	141702	B03E02
0134	114230	I30I30
0135	143707	B0-3B0-3
0136	143203	B0-2I03
0137	100770	I01E-30
0140	174171	B-30F
0141	030303	ME03
0142	141702	B03E02
0143	154330	B30E30
0144	127655	I-1-3I-1-1
0145	174350	B-30E-10
0146	103605	I0-3I0-1
0147	154330	B30E30
0150	074571	B-31F
0151	030303	ME03
0152	141702	B03E02
0153	000600	I01I00
0154	154330	B30E30
0155	074571	B-31F
0156	030202	M102
0157	141723	B03E23
0160	150327	B20E2-3
0161	143766	B0-3B-2-2
0162	170362	B-20E-22
0163	074571	B-31F
0164	100000	NOP

## Vector Description of Letter "d"

Memory Location	Octal Representation	Vector Description			
5440	030233	E	D+3+3	ElM	V 1
5441	110772	D+2+1	B-3+2	V 2	V 3
5442	173307	B-2-2	B+0-3	V 4	V 5
5443	152732	B+2-1	B+3+2	V 6	V 7
5444	103303	D+0-2	B+0+3	V 8	V 9
5445	141703	B+0+3	B+0+3	V 10	V 11
5446	074571	F	F	ESCAPE,+ RETURN JUMP	

E=Enter Increment Mode (sets flag in machine)  
 D=Dim (beam off)  
 B=Beam on  
 F=Non-Increment Mode



### 3. Interactive Graphics Package

A proprietary package, optimized towards the graphical capability of the Imlac PDS-1 is available. It permits line drawing graphics editing from the keyboard as well as limited text editing.

The cursor control keys are used to rapidly position a cursor around the screen. Other keyboard commands connect long lines between pre-selected points.

The capability of this package is developed further.