DECUS
PROGRAM LIBRARY

DECUS NO.	12-102
TITLE	A MANUAL FOR THE PDP-12 OPERATOR
AUTHOR	Peter Hiscocks
COMPANY	Ryerson Polytechnical Institute Toronto, Ontario, Canada
DATE	July 4, 1972
SOURCE LANGUAGE	

PURPOSE OF THIS MANUAL

The purpose of this manual is not to replace the detailed documentation already available on PDP-12 programming and the various aspects of interfacing. It is designed to give an "overview" of the operation of the PDP-12, indicating the steps to go through to learn PDP-12 programming, and pointing out additional sources of information.

The reader is also referred to the following sources:

(1) DIAL and PIP Reference Manual

(2) Introduction to Programming
available from Digital Equipment

(3) PDP-12 Reference Manual

Suggestions for improvement will be appreciated.

CONTENTS

Page

USING THIS MANUAL

...ding a Program from the Console Switches

Initialize (P 3)
↓
Load the Program (P 3)
↓
Examine the Program (P 4)
↓
Run the Program (P 4)
↓
Troubleshoot the Program (P 5)
↓
Clear the Working ...ea (P 9)

Loading a Program by Punched Paper Tape, Using Dial and Pip

↓
Punch the Program on an Off Line Teletype (P 21a)
↓
Initialize (P 3)
↓
Start Dial (P 8)
↓
Start Pip (P 15)
↓
Transfer from Punched Paper Tape to Linc Tape (P 18)
↓
Return to Dial (P 16)
↓
Load the Program from Linc Tape into the Working Area (P 11)
↓
Assemble the Program (P 9)
↓
Store the Binary Program (P 10)
↓
Load the Binary Program (P 12)
↓
Start the Binary Program (P 4)

Typing in a Program Directly

↓
Initialize (P 3)
↓
Start Dial (P 8)
↓
Type in the Program (P 9)
↓
Assemble the Program (P 9)
↓
Store the Source Program (P 10)
↓
Store the Binary Program (P 10)
↓
Load the Binary Program (P 12)
↓
Start the Binary Program (P 4)

RYERSON ELECTRICAL DEPARTMENT

BASIC OPERATION OF THE PDP-12 CONSOLE

Initialization

1. Circuit breakers turned on.

2. Main power cord is properly plugged in.

3. Computer POWER SWITCH is turned ON (VERTICAL). The panel lights should come on, and the fans should be heard.

4. STOP key is not set.

   NOTE: A key is set when the part towards you is down.

   Sing STEP, EXEC STOP and FETCH STOP Keys are cleared. (Part towards you is up).

   LINC/MODE/8 key is set (for PDP-8 programming)

   RIGHT SWITCHES are 0000 (octal)

   NOTE: A switch is 1 when the back is down.

   Left switches are 0000

   INST FIELD is 0

   SENSE SWITCHES are 00

   The system is now inialised and ready for your use.

LOADING A PROGRAM

1. Initialise.
2. Press STOP and reset STOP. Pres I/O PRESET
3. Set LEFT SWITCHES to the program's first address
4. Set RIGHT SWITCHES to the program's first instruction
5. Depress FILL. (See Note 1)
6. Depress FILL STEP (See Note 1)

7. Are all the instructions in? No, go to Step 8
   Yes, go to step 9

8. Set RIGHT SWITCHES to the program's next instruction.
   Go back to step 6.

9. Program is loaded.

EXAMINING A PROGRAM:

1. Initialise

2. Set the LEFT SWITCHES to the program's first address

3. Depress the EXAMINE key

4. The address of the instruciton is shown on the MEMORY ADDRESS register. The contents of that address are shown on the MEMORY BUFFER register.

5. Is the MEMORY BUFFER equal to the correct instruction?
   No, go to step 6.
   Yes, go to step 8.

6. Set the LEFT SWITCHES to the address of the wrong instruction.
   Set the RIGHT SWITCHES to the correct value of the instruction.

7. Depress FILL

8. Is this the last instruction? Yes, go to step 10.
   No, go to step 9.

9. Depress STEP EXAMINE
   Go back to step 4.

10. Program is checked.

RUNNING A STORED PROGRAM:

1. Set the LEFT SWITCHES to the address of the first instruction.

2. Depress and reset the STOP key
   Depress I/O PRESET
   Depress START LEFT SWITCHES key

3. The program is running.

SINGLE STEPPING A PROGRAM (a) MANUALLY

1. *Set the LEFT SWITCHES to the address of the first instruction.*

2. *Set the SINGLE STEP key.*

3. *Depress and reset STOP.*
   Depress I/O PRESET
   Depress START LEFT SWITCHES.

4. *The PROGRAM COUNTER register shows the address of the next instruction.*

   The MEMORY ADDRESS shows the address of the last referenced location in memory (the address of the instruction or data last retrieved from memory).

   The INSTRUCTION REGISTER shows the instruction about to be executed.
   The MEMORY BUFFER shows the last instruction or data to be retrieved from memory.
   The ACCUMULATOR register shows the state of the ACCUMULATOR.

5. *Depress CONTINUE*
   Go back to step 4.

SINGLE STEPPING A PROGRAM: (b) USING AUTO RESTART

1. *Set the LEFT SWITCHES to the address of the first instruction.*

2. *Set the SINGLE STEP key.*

3. *Depress and reset STOP*
   Depress I/O PRESET
   Depress START LEFT SWITCHES

4. *Depress AUTO and CONTINUE simultaneously.*

5. *Adjust the AUTO RESTART DELAY COARSE and FINE knobs to set the desired operating speed of the processor. (These controls are found above the ON/OFF key).*

   The AUTO Flip Flop light will be lit when AUTO RESTART is being used.

   AUTO RESTART can only be disabled by depressing I/O PRESET.

NOTE 1

FILL and FILL STEP operate as follows:

FILL causes (1) the contents of the RIGHT SWITCHES to be put into the location specified by the LEFT SWITCHES.

(2) The memory address register to be set to the contents of the LEFT SWITCHES.

FILL STEP causes (1) the contents of the RIGHT SWITCHES to be put into the location specified by the MEMORY ADDRESS register.

(2) The MEMORY ADDRESS register to be incremented (stepped).

OPERATION OF THE RIM LOADER

The RIM (Read In Mode) loader loads the BIN (Binary) loader which loads any paper tape in binary mode. The RIM loader is the logical starting place to operate a machine with only paper tape readers.

The following instructions were taken from Introduction to Programming, Page E1-1, modified for the PDP-12.

1. *Initialize*

2. *Load the RIM loader (shown on the following page) according to the previous instructions for loading a program.*

3. *Examine the RIM loader according to the previous instructions for examining a program.*

4. *Put low speed reader (LSR) to FREE*

RIM LOADER

LOCATION	LOW SPEED READER	HIGH SPEED READER
7756	6032	6014
7757	6031	6011
7760	5357	5357
7761	6036	6016
7762	7106	7106
7763	7006	7006
7764	7510	7510
7765	5357	5374
7766	7006	7006
7767	6031	6011
7770	5367	5367
7771	6034	6016
7772	7420	7420
7773	3776	3776
7774	3376	3376
7775	5356	5357

5. Place the Binary Loader paper tape in the L.S.R.

6. Turn the LSR to START

7. Start the computer at 7756

8. If the Binary Loader tape does not read in, check the RIM loader.

OPERATION OF THE BINARY LOADER (To load any binary coded tape)

1. Load the RIM Loader and the BINARY LOADER

2. Turn LSR to FREE

3. Place the binary coded tape in the L.S.R.

4. Turn the LSR to START

5. Start the BINARY LOADER at 7777.

6. The tape should read in until it reaches the trailer tape. It should then stop. The program should halt with the accumulator equal to zero.

7. If AC = 0, you may start your binary program at its starting address.

8. If AC ≠ 0, reload the BINARY LOADER.

OPERATION OF THE DIAL PROGRAM

INTRODUCTION

We have previously outlined two methods of loading a program into the PDP-12. The first involved "toggling in" a machine language program by means of the console switches. (SEE Basic operation of the PDP-12 console). The second, which assumed that a binary tape (a machine language tape) of the program was available, involved the use of the RIM loader and the BIN loader. The RIM loader was used to load the BIN loader, the BIN loader was used to load the binary tape.

Both these processes are very tedious when a long program is being developed. To aid in the process of developing programs the DIAL (Digital Interactive Assembly Language) was developed. DIAL is referred to as a MONITOR program because it supervises all the other programs. An overview of the various DIAL operations is shown on the "DIAL PROGRAM", at the end of this section.

STARTING DIAL

To start up dial, it is necessary to go through the following drill. (Dial resides on tape, and must be pulled into core memory to operate. The following drill causes DIAL to be loaded and started).

1. Initialize the system (As previously discussed)

2. Mount the DIAL tape on unit Ø and a second tape on unit 1

3. Set the tape units to WRITE and REMOTE

4. Set the left switches to 0701
   Set the right switches to 7300

5. Set the processor mode switch to LINC

6. Press STOP, I/0 PRESET and DO in that order, one after the other

7. Wait for the tape to stop

8. Press I/0 PRESET
   Press START 20

FUNCTIONS OF DIAL

DIAL will perform the following functions:

1. LOAD A SOURCE PROGRAM

DIAL enables a program in SOURCE mode to be typed in and edited by means of the teletype (TTY) and cathode ray tube (CRT).

(A program is written in SOURCE mode has machine language instructions replaced by mneumonics. For instance, the instruction to clear the accumulator and linc is written as 7300 in machine language; as CLA CLL in source language. See Introduction to Programming, chapter 3 for further detail).

An area of the tape known as the Source Working Area on unit Ø is used to store the program as it is developed. Programs entered in this fashion are the same as discussed in Introduction to Programming with the following addition. Before entering the program, type

→ CL↲

to clear the "working area" where your program will be stored. The first instruction in a PDP-8 mode program must be the pseudo-operator

P MODE

The first instruction ina LINC mode program must be the pseudo operator

L MODE

Details on manipulating and editing a program are discussed in the DIAL reference manual, chapter 2, kept by the computer.

2. ASSEMBLE A PROGRAM

When the program has been edited into its final form in source mode language, it may be converted into a binary program by executing the command

→ AS↲

where → indicates Line Feed

↲ indicates Carriage Return

Alternatively, the program may be listed, on the teletype, in its Source form and its Binary form by executing the insturction

→ LI↲

In both cases, a list of errors in the program is generated. Error messages are explained in the DIAL manual.

3. STORING A SOURCE PROGRAM

A source program may be stored on at any time by executing the instruction

→ SP ␣ NAME, U ↲

where ␣ represents space

NAME is the name by which the file will be tagged

U is the number of tape transport (Ø or 1) which is to be used to store the program

Example

→ SP ␣ MEAT, Ø ↲

causes the source program in the working area to be labelled "MEAT" and stored on unit Ø.

The name of the file and the tape blocks that it occupies, are added to the tape directory (see below).

4. STORING A BINARY PROGRAM

After a program has been assembled, its binary version is on tape unit 1 in the Binary Working Area. It may be moved from the Binary Working Area and stored on tape by executing the instruction

→ SB ␣ NAME, U ↲

It is convenient to name the binary version the same as the source version.

5. USING THE TAPE DIRECTORY

Dial maintains a directory of all the programs stored on the tape. Each program is listed by name, together with its block number (BN) (the first block of tape where the program is stored) and the total number of blocks it occupies. (BLKS)

(The first block number and the total number of blocks for a program are used when transferring programs from tape to tape.)

The Tape Directory of a particular tape may be displayed on the CRT by typing

→ DX, U ↲

where U is either 1 or Ø depending on the tape directory to be displayed.

The display may be manipulated by typing

"ALT MODE" and "Q" simultaneously

(forward one frame), or

"ALT MODE' and "W" simultaneously

(back one frame).

A permanent record of the tape direcotry may be obtained by typing

→ PX, U ↲

The Tape Directory will be printed out on the teletype.

Type ↲ to return to DIAL

6. <u>LOADING A SOURCE PROGRAM FROM TAPE INTO THE SOURCE WORKING AREA</u>

(for modification, for instance).

If the source program is stored on tape, it may be called into the Source Working Area by typing

→ AP ␣ NAME, U ↲

where "NAME" is the name of the program

U is the tape unit on which it is stored

If the CRT displays

NO

then you probably asked for a non existent program - mis-spelled name or wrong unit. Check the tape directory.

The source program having been loaded into the source working area, it may be modified. The modified version (in the working area) may be mode to replace the old version (stored on tape) by typing

→ EX ↲

("EX" stands for "EXIT")

Otherwise, the following dialogue takes place:

YOU: → SP ␣ MEAT, Ø ↲

PDP 12: REPLACE? (Displayed on CRT)

YOU: R ↲ (typed)

PDP 12: (obediently replaces the old version on tape Ø with the new version in the source working area)

7. LOADING A BINARY PROGRAM INTO CORE MEMORY FOR EXECUTION

If the program is in the binary working area- that is, if it was previously assembled - it may be loaded from the binary working area by typing

→ LO ↲

The program may be executed by starting the program as described previously.

If the program has been filed on tape, it may be loaded into core by typing

→ LO ␣ NAME, U ↲

If the CRT displays

NO

You asked for a non existant program. Chech the tape directory.

ERASING A PROGRAM ON TAPE

Look up the tape directory by typing

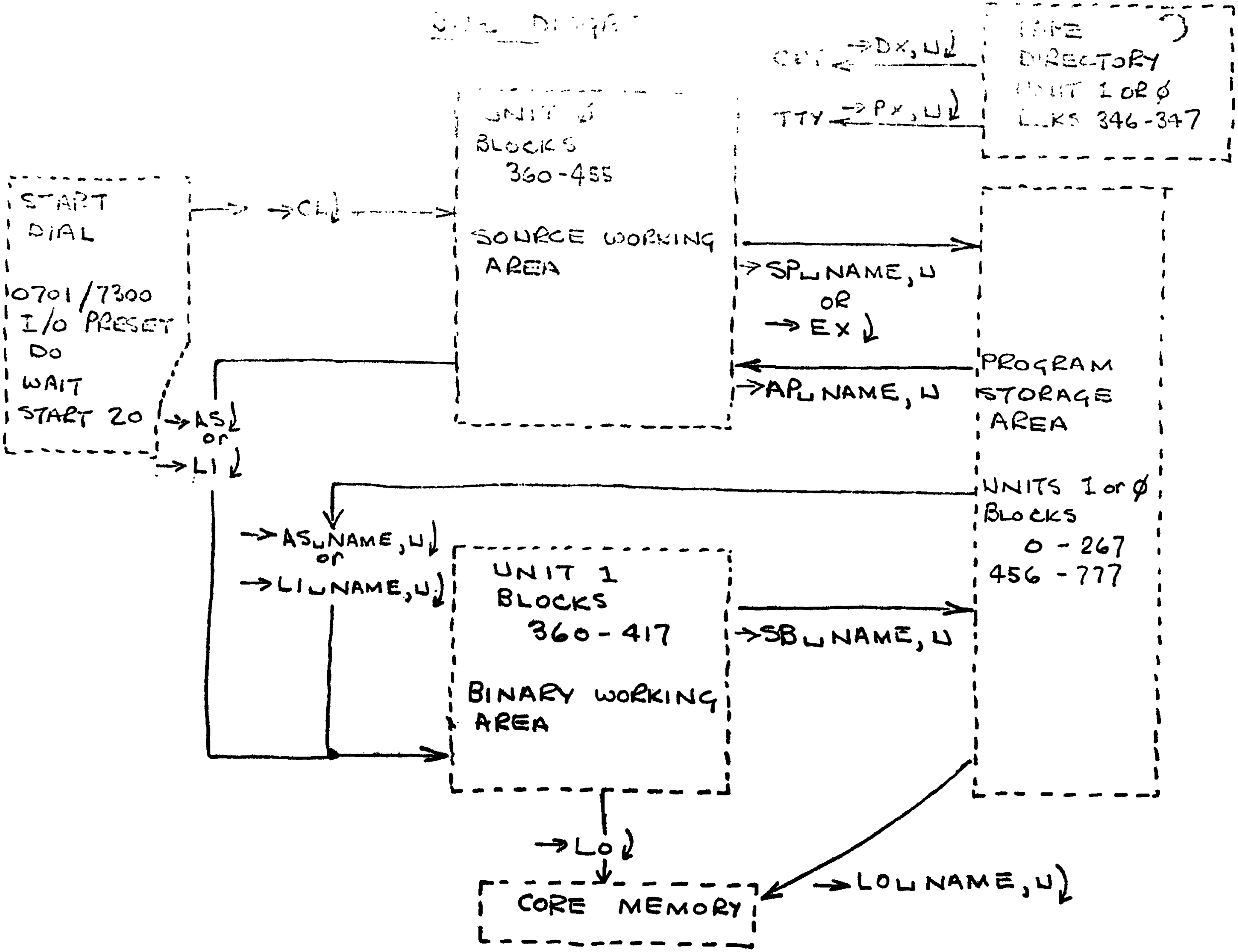
→ DX, U ↵

where U is the tape unit (Ø or 1)

Manipulate the display by typing "Q" or "W", until the program to be removed is the last program displayed.

(i.e.) at the bottom of the list.

Depress the "RUBOUT" key of the teletype, followed by " : ". The program will be removed.


DIRECTORY
346-347
→Dx,␣↓
TTY →Px,␣↓
START
DIAL
0701/7300
I/O PRESET
DO
WAIT
START 20
→CL↓
BLOCKS
360-455
SOURCE WORKING
AREA
→SP␣NAME,␣
OR
→EX↓
→AP␣NAME,␣
PROGRAM
STORAGE
AREA
UNITS 1 or Ø
BLOCKS
0 - 267
456 - 777
→AS↓
or
→LI↓
→AS␣NAME,␣↓
or
→LI␣NAME,␣↓
UNIT 1
BLOCKS
360-417
BINARY WORKING
AREA
→SB␣NAME,␣
→LO↓
CORE MEMORY
→LO␣NAME,␣↓

MOVING PROGRAMS AROUND

*It is frequently necessary to move a program from a magnetic tape to another magnetic tape from a magnetic tape to punched paper tape;** from puneched paper tape to magnetic tape; and so forth.*

The program which accomplishes this for you is known as PIP for Periferal Interchange Program. PIP is part of the DIAL system. Details on PIP are contained in the manual by the computer.

STARTING PIP

Initialize the computer
(see previous instructions)

Start DIAL
(see previous instructions)

Type →PI ↲

The display flashes

"I could run faster if I had 4K instead of 8K"

and then PIP is ready to be used.

USING PIP

PIP carries on an initial dialogue with the operator, which is as follows:

PIP OPTIONS

a-----auxilary mode

b-----binary mode

s-----source mode

*** A long program is always safer on punched paper tape than on magnetic tape. Always store your precious programs on punched paper tape.*

AUXILARY MODE (See figure 1 and figure 2) (PAGES 19 and 20).

Auxilary mode is used to transfer information from one linc tape to another:

To copy a program from one tape to another when the block numbers are known (the first block and the number of blocks). (When the program in known by name, it is more convenient to use binary or source mode - see below).

OR

To duplicate tape Ø to tape 1

OR

To copy the DIAL system from one tape to another

OR

To copy one complete tape onto another

To choose auxilary mode, type

A↲

COPYING DIAL

To copy DIAL, type

S↲

Then type

LØ↲

to indicate that the DIAL program is to be copied from LINC tape Ø, type

L1↲

to indicate that the DIAL program is to be copied onto linc tape 1.

RETURNING TO DIAL or PIP

Type CTRL and D
simultaneously to return to DIAL:
type CTRL and P
simultaneously to return to PIP

BINARY or SOURCE MODE (See figure 1, PAGE 19)

Binary and Source mode are used when a program is on linc tape, known by name and to be transferred to another linc tape or punched out on paper tape

or

when a program is on punched paper tape and to be transferred to linc tape and given a name.

TRANSFERRING A PROGRAM FROM LINC TAPE TO LINC TAPE

1. Start PIP
2. Choose binary mode (type B↓) if the program is in machine language. Choose source mode (type S↓) if the program is in source language.
3. To specify the INPUT DEVICE:
   type

   L n ; NAME

   where n is the number of the LINC tape to be copied (Ø or 1)

   NAME is the name of the program to be copied

   If the CRT displays NO
   you probably asked for a non existent program.
   Check the tape directory.
4. To specify the OUTPUT DEVICE:
   type
   L n ; NEW NAME

   where n is the number of the LINC tape where the program is to be stored

   NEWNAME is the name to be given to the program when it is stored, for future reference.
5. You may check the tape directory to make certain the program was properly transferred.

TRANSFERRING A PROGRAM FROM LINC TAPE TO PUNCHED PAPER TAPE

1. Repeat steps 1 to 3 above

4. Turn on the paper tape punch

5. To sepcify the output device type T↵
   The paper tape should now be punched

6. Label your paper tape with its name, your name, and its type (binary or source). All paper tapes look the same!.

TRANSFERRING A PROGRAM FROM PUNCHED PAPER TAPE TO LINC TAPE

1. Repeat steps 1 and 2 above

3. To specify the input device type T↵

4. Specify the output device by typing Ln : NAME
   where n is the number of the LINC tape to be used
   NAME is the name which you wish your program to be called.

   e.g. LØ; MEAT

   The paper tape should now be read in.

5. Turn the low speed reader to FREE
   Put your paper tape in the teletype low speed reader
   Turn the reader to START

6. When finished, PIP restarts: Type: CTRL D
   to return to DIAL
   If PIP does not restart, Type: CTRL &
   PIP comes back with "MORE INPUT?"

   N↵ (no more)

   R↵ (read more)

   Type: CTRL D to return to DIAL

7. You may check the tape directory to make certain the program was properly read in.

   → DX,n↵

PIP OPTIONS

a auxilary → A → **OPTIONS**

- c copy specified blocks
- d duplicate tape Ø onto tape 1
- s copy system
- U copy unit → (See figure 2)

b binary → B → INPUT DEVICE

s source → S → **INPUT DEVICE**

- h high speed reader (~~nonexistant~~)
- l linc tape
- t teletype
- c card reader (nonexistant)

↓

OUTPUT DEVICE

- h high speed punch (~~nonexistant~~)
- l linc tape
- t teletype
- p line printer (nonexistant)

Figure 1

PIP OPTIONS

AUXILARY MODE

c--copy specified blocks	d-- duplicate tape Ø onto tape 1	s--copy system	U--copy unit
C ↲	D ↲	S ↲	U ↲
Input Device		Input Device	Input Device
LØ;Bn, BKS ↲		LØ ↲	LØ ↲
or		or	or
L1;Bn, BKS ↲		L1 ↲	L1 ↲
Output Device		Output Device	Output Device
L1,Bn ↲		L1 ↲	L1 ↲
or		or	or
LØ,Bn ↲		LØ ↲	LØ ↲

('BN' is the number of the first block to be transferred
BKS is the number of blocks to be transferred)

eg) LØ; 300,26

LINC TAPE Ø
FIRST BLOCK = 300
NO OF BLOCKS = 26

Figure 2

PIP Auxilary Mode Options

(j) ***MARKING A BLANK TAPE***

A blank tape or one which is to be cleaned off for re-use must be re-marked by the MARK 12 program. This program lays down timing trancks and divides the tpae into blocks.

To mark (or re mark) a tape

1. *Mount the DIAL tape on unit Ø and set to WRITE AND REMOTE .*
2. *Mount the tape to be marked on the right reel of unit 1. Wind a few turns from the right reel on to the left ree.*
3. *Start DIAL (described previously)*
4. *Load MARK 12 by typing*

   LO␣MARK12, Ø ↵

5. *Put unit 1 on WRITE and REMOTE*
6. *Press the MARK switch on the console*

The tape will then be marked and may be used to store DIAL or any other program.

The paper tape may then be loaded onto magnetic tape using the procedure given under "Moving Programs Around", "Transferring from punched paper tape to Linc tape".

Typing a program on to punched paper tape on an "OFF LINE" Teletype *

1. Turn the rotary switch on the front panel of the teletype to "LOCAL".

2. Press the "ON" button on the teletype punch.

3. Generate paper tape leader by pressing the "HERE IS" key on the teletype.

4. Type in your program as per normal.

   If you make a mistake, leave it. It can be corrected after being loaded into the computer.

5. Generate paper tape trailer by pressing "HERE IS".

6. Turn the teletype rotary switch to "OFF".

PROGRAMMING THE PDP-12 LINC TAPE

Digital Equipment manufacture two small tape transports for use on their computers; the DEC tape unit and the LINC tape unit. The LINC tape unit (in contrast to teh DEC tape unit) contains a small "sub processor" which carries out transfers between the LINC tape and the PDP-12 core memory. (The DEC tape unit is controlled by program in the main core memory).

Storage Capacity:

Each LINC tape is divided into 777_8 "blocks". One block contains 400_8 locations.
In decimal notation, we have

One tape = 512_{10} blocks (777_8)

One block = 256_{10} locations (400_8)

One core = 4096_{10} locations (7777_8)

= 16_{10} blocks

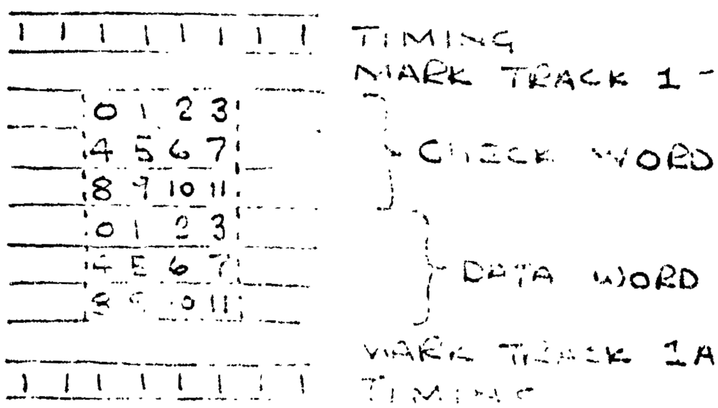
so

One tape = 32_{10} core images (40_8)

Each block length is approximately 2".
Each word transfer takes approximately 133μ sec.

Tape Format:

The tape has a total of ten tracks. Four are used for mark and timing; the remaining six are data tracks. This is shown below:



Notice that the data word is recorded redundantly as a "check word". The numbers shown refer to the bits of the data word. The data word and check word are logically ORed together to prevent "bit dropout".

The timing track is recorded at constant frequency, irregardless of tape speed. This makes it unnecessary to control the tape speed and simplifies the mechanical construction of the tape transport. As a result, the spacing between timing pulses will vary.

The mark track tells the tape control how to interpret the timing track. The tape is divided into end zones (at the ends of the tape, where no data is recorded) inter block zones (between blocks of data), and data blocks. The mark track indicates the zone to the tape control.

Tape Instructions:

Linc tape instructions are in a two word format:

Word 1: What to do with the data
Word 2: Where to take data from or where to put the data

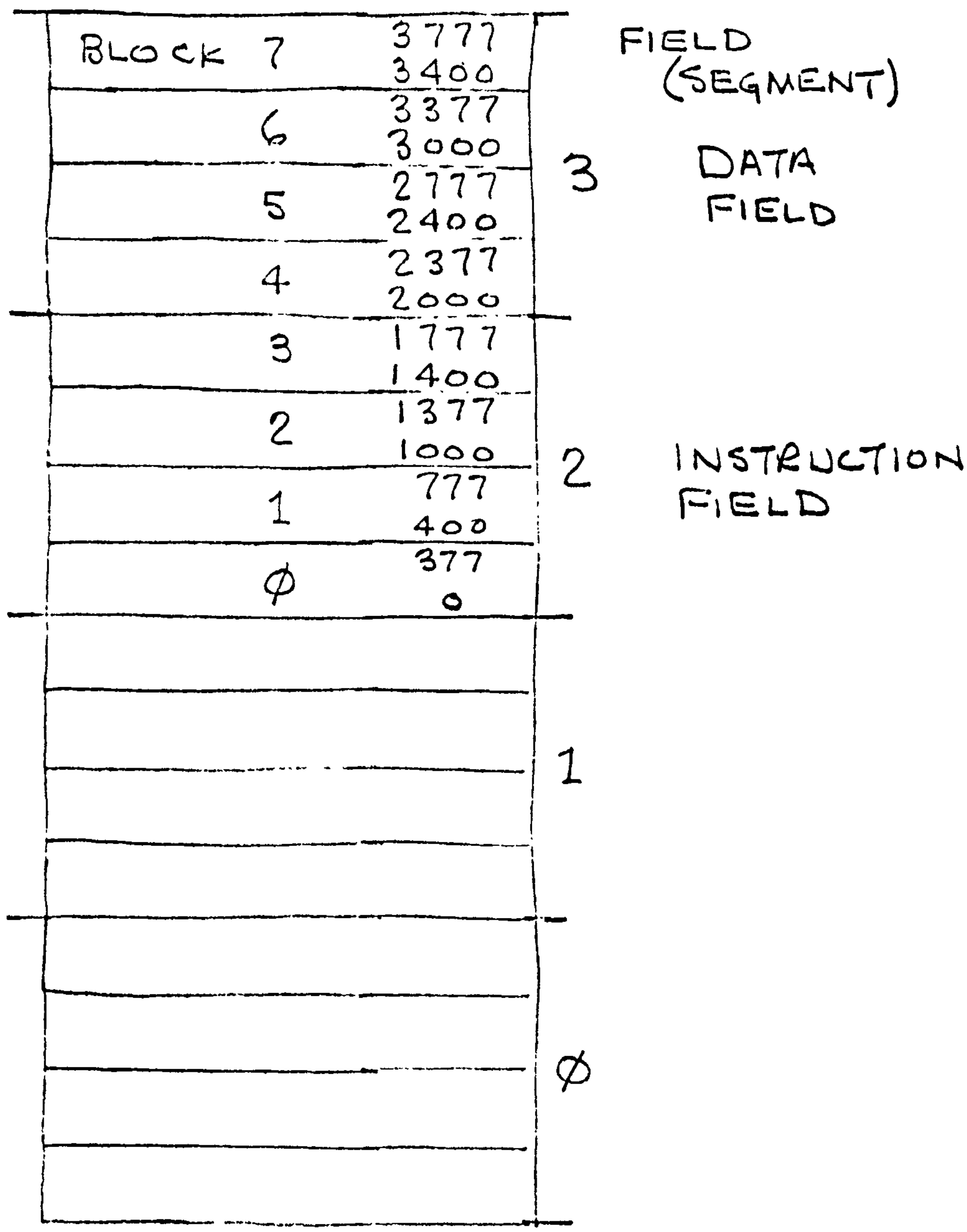
Word 2 Format

Recall that 2 LINC programs operate (normally) in segments 2 and 3 of core memory.

Segment 2 (locations 4000 to 5777) is the Instruction Field (I.F.); Segment 3 (locations 6000 to 7777) is the Data Field (D.F.).

For the purpose of LINC tape programming, these two core segments are broken up into blocks, each 400_8 locations, corresponding to the length of one tape block. This is shown below:

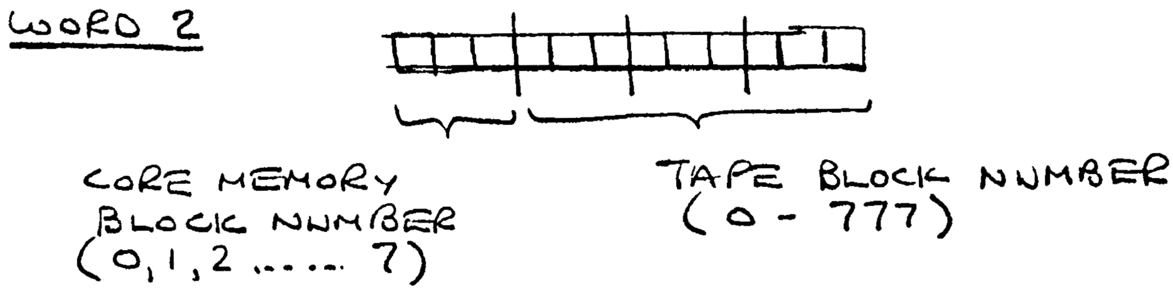
(SEE PAGE 2A)


BLOCK 7
3777
3400
6
3377
3000
5
2777
2400
4
2377
2000
3
1777
1400
2
1377
1000
1
777
400
Ø
377
0
3
2
1
Ø
FIELD
(SEGMENT)
DATA
FIELD
INSTRUCTION
FIELD

Notice that the addresses given inside each block correspond to the relative address in the field.

For instance, a program running in instruction field 2 will treat absolute address 4000 as relative address 0.

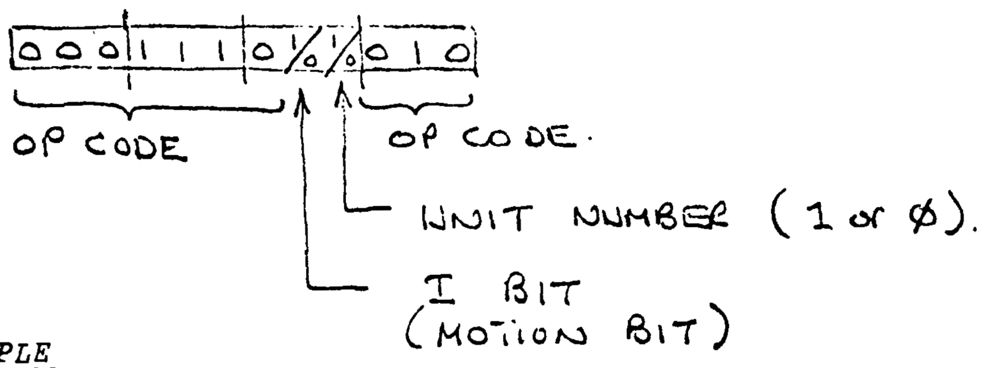
The format of word 2 of a tape instruction is shown below:



READ TAPE   *RDE I n*

Read tape as specified in word 2 from unit n

If I = 1, remain in motion in same direction
I = ∅, reverse direction



EXAMPLE

RDE I ∅
6055

Read from unit ∅, remain in motion in same direction
Read into core block 6, from tape block 55

(What relative core locations are involved?
What absolute core locations are involved?)

READ TAPE AND CHECK TRANSFER * RDC I n

(If reading is no good, repeat transfer;
if reading is OK, continue program).

WRITE BLOCK WRI I n

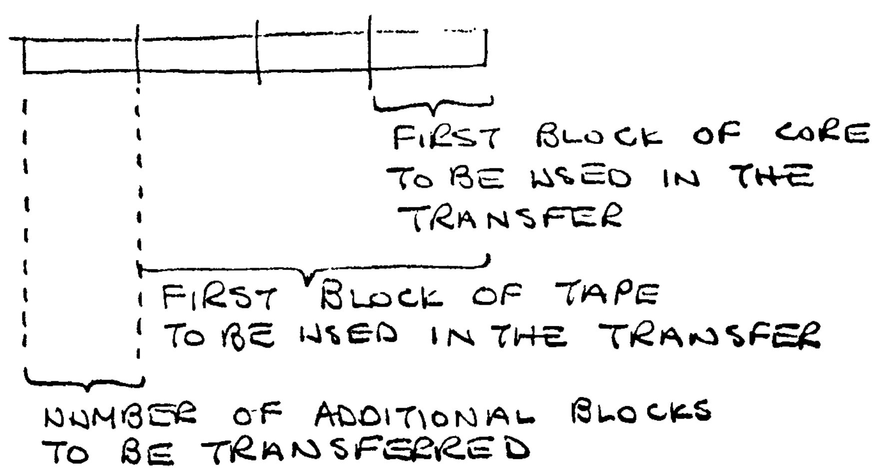
WRITE AND CHECK BLOCK WRC.I

The next two instructions refer to transfers of groups of blocks. Word 2 format is different in these cases (see below).

READ AND CHECK GROUP RCG n

WRITE AND CHECK GROUP WCG n

Word 2 Format



Example

The instruction used to start the DIAL program is "0701 7300"

which decodes as shown below:

0701 RCG Ø
(Read and check group, from tape unit Ø)

7300 Load first block into core block Ø

Load first block from tape bloc 3ØØ

Load 7 additional blocks

* Checking is performed by a checksum addition at the end of the transfer of the block.

Utility Tape Instructions

No data transfer is involved with these instructions.

MTB *Move towards block (specified in word 2)*

CHK *Check tape block (specified in work 2)*
(The tape runs through the specified block, generating a checksum in its own checksum register. At the end of the block, this checksum is compared with the checksum recorded on the tape).

TAC *Read the tape accumulator*

LINC TAPE ASSIGNMENT

1. a) Write a program to flash the LINK bit of the accumulator on and off at about a one second rate.

b) Write a second program to store this first program on LINC tape and clear the core locations used by this first program.

c) Write a third program to load the first program back into core from LINC tape.

2. Write a program which will perform the following functions:

a) Sample the output of a potentiometer or analogue input and store the digitized values on tape.

b) Display the digitized values on the CRT screen.

A total of 1777_8 values are to be sampled.

The display is to consist of half these values
- the lower half if sense switch 1 = 1;
the upper half if sense switch 1 = Ø.

REFERENCES

A picture of the TU30 (Linctape) transport and a brief description of the transport controls and indicators is given on pages 2-11 and 2-12 of the PDP-12 User's handbook.

Details of the operation of the Linctape unit are presented on pages 3-36 to 3-46 of the PDP-12 User's Handbook.

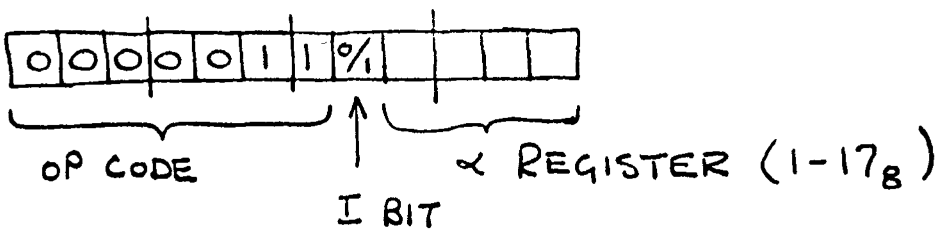
PROGRAMMING THE PDP-12 DISPLAY

DISPLAYING A SINGLE POINT

The instruction for displaying a single point is:

DIS I α DISPLAY A POINT

The binary coding of this instruction is:



A single point on the screen in intensified.

A vertical position of this point is specified by bits 3 to 11 of the accumulator.

The horizontal position of this point is specified by bits 3 to 11 of the designated α register. (See Fig. 2).

Bit Ø of the designated α register determines the channel on which the point is displayed.

The DEC VE-12 oscilloscope has a knob on the front panel which selects channel Ø, channel 1, or both. Both channels are displaed on an extension oscilloscope.

The "I BIT" in display instruction controls whether or not the contents of the α register are to be incremented before they are used.

For I = 0, no incrementing

I = 1, increment before using.

When I is set to 1, and the α register is incremented each time the display instruciton is used, this will have the effect of moving the displayed point horizontally to the right, producing a horizontal straight line.

THE CATHODE RAY TUBE:

The CET is divided into 512_{10} x 512_{10} points. This is 777_8 x 777_8 points. The points are numbered from 000 to 777 horizontally

from-377 to +377 vertically.

Negative numbers in the vertical direction are represented in complementary notation, so that -1 is 777, -2 is 776 and -377 is 400. This is shown in Figure 1.

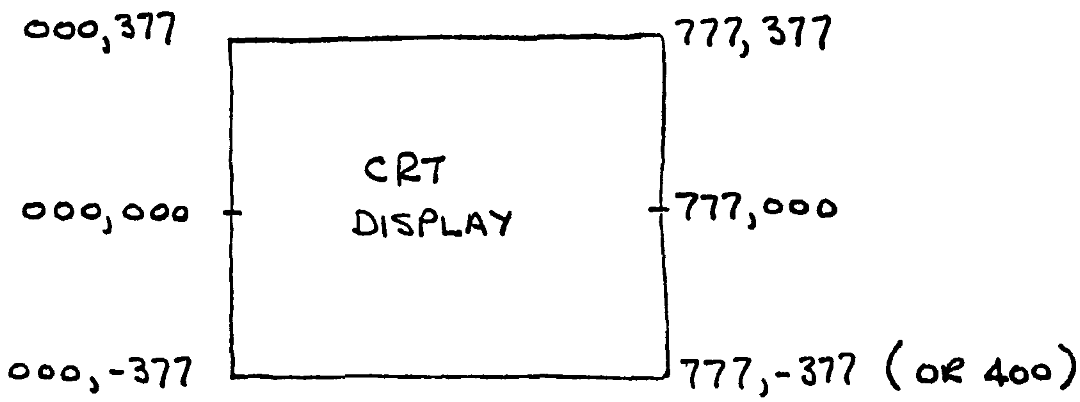
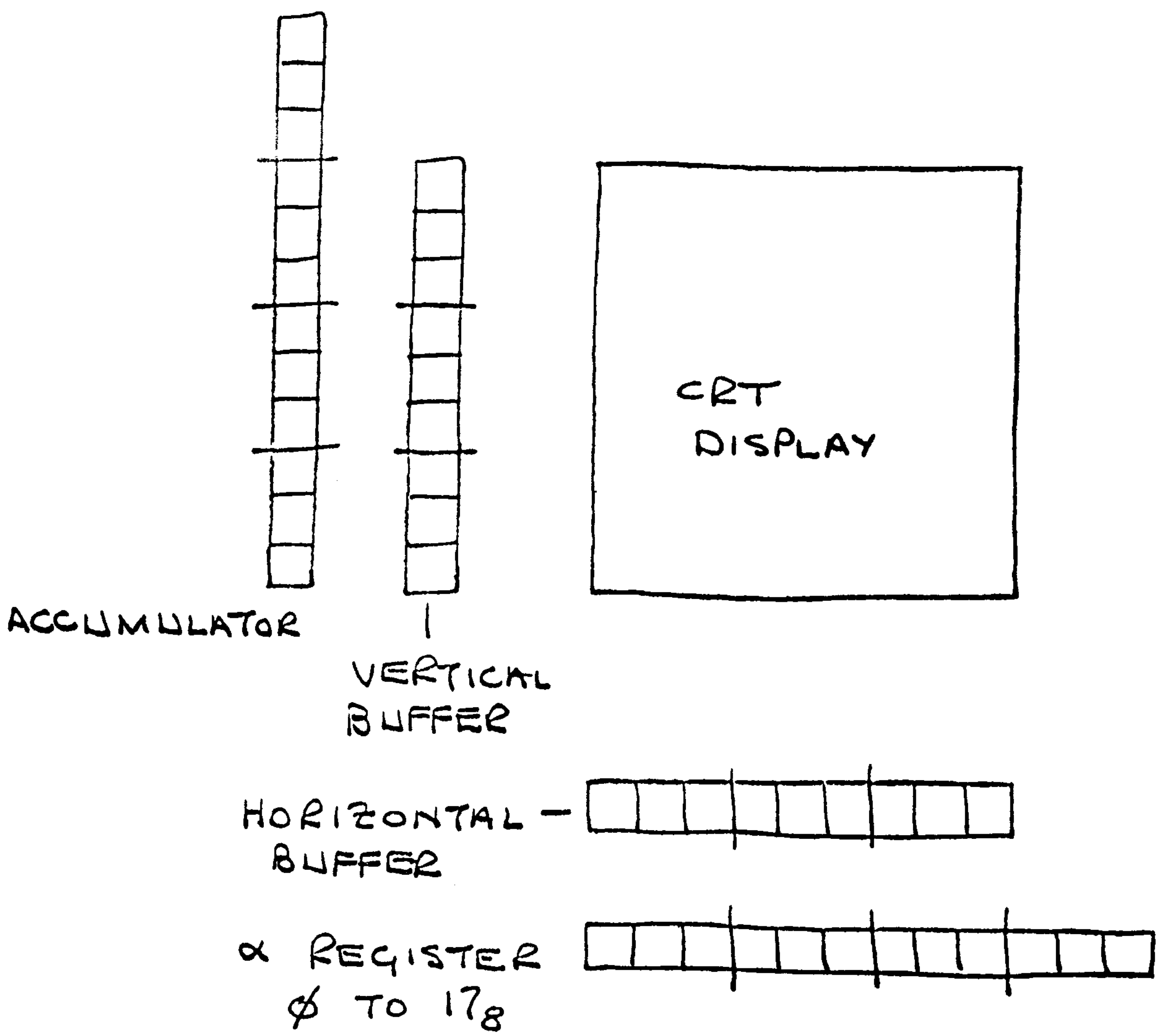


Figure 1

CRT COORDINATES



NOTE: The Vertical and Horizontal Buffers Retain the Last - Specified Vertical and Horizontal Coordinates.

Figure 2

Example: A programme to display one point at screen centre.

(x = 400,   y = 000)

```
0200        CLR          / Clear AC and Link

0201        ADD x        / Load horizontal coordinate into AC

0202        STC HOR      / Deposit in α register number 5 in order to
                           specify the horizontal position of the spot.

0203        DIS HOR      / Display the point.  Notice that the AC = Ø
                           because of the DCA instruction in 0202 so the
                           vertical position will be screen centre.

0204

0205        JMP 0203     / Return and repeat.

0206        HOR, 0005    / α  register # 5.

0207        x, 0400      / Horizontal coordinate.
```

Coded into octal this would be:

```
0200
0201
0202
0203
0204
0205
0206
0207
0208
0209
0210
```

Example: A programme to display a horizontal straight line at screen centre (y = 0)

```
0200        CLR           /Clear AC LINK
0201        DIS I HOR     / Increment ∝ register 5 and display a point
0202        JMP 0201      / Return and repeat
0203        HOR, 0005
```

Coded into octal this would be:

```
0200
0201
0202
0203
0204
0205
```

PROBLEMS

1. Write a programme to display a square on the CRT.

2. Write a programme to display a diamond on the CRT. (a square turned through 45°)

3. Write a programme to display a graph which is to represent the octal values of a table of data words in memory.

   Note: It will be necessary to scale the values of the data words so that they will fit on the CRT, Why?

DISPLAYING A CHARACTER

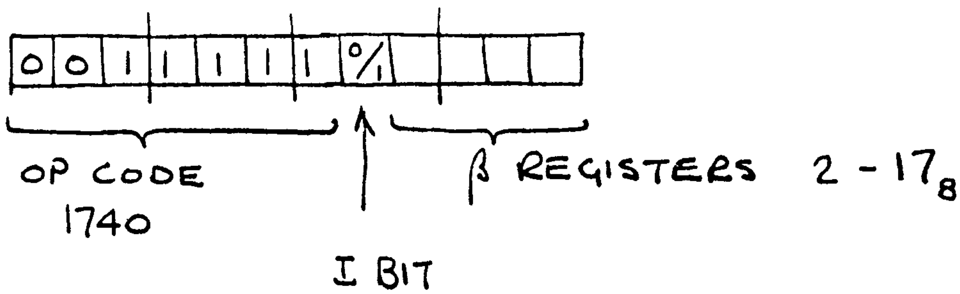
Characters could be displayed on the CRT by displaying a series of dots, using the DIS instruction. However, this would involve a great deal of programming effort and the computer would be occupied a great deal of the time. What has been done is the processor has been quipped with a "half character generator," which automatically writes out one half a character. The half character generator automatically moves the electron beam of the scope to a series of twelve points, in a 6 x 2 matrix (see page8 •). The points of this matrix which are to be intensified are specified by a twelve bit computer word which ahs been moved into the "intersification buffer". Where there is a 1 in the computer word, the corresponding dot in the matrix is brightened.

At the end of the display of one half character, the matrix is moved horizontally to the right. (This is done automatically, by the contents of increasing memory location 0001, which holds the horizontal position information for the half character matrix).

This is discussed in detail below.

THE DSC INSTRUCTION

DSC I β



The operation of this instruction is as follows:

Bits 0 through 6 indicate that a "display character" is to take place.

Bits 8 through 11 direct the computer to a β register. The content of the β register indicates where the intensification word is to be found. In other words, an indirect addressing operation take place, with the specified register containing the pointer address.

Bit 7, the "I" bit, indicates whether the content, of the specified β register is to be incremented before it is used.

For I = 0, do not increment before using.

For I = 1, increment before using.

EXAMPLE:

0015	0300
0200	DIS I 15 (coded 001, 111, 111, 101)
0300	
0301	7741
0302	1477

When the computer executes the DSC instruction in 0200, it refers to β register 0015. Because the I bit is 1, the content of 0015 is incremented from 0300 to 0301.

The intensification word 7741 is then moved to the intensification buffer and used to brighten up the appropriate points in the half character matrix.

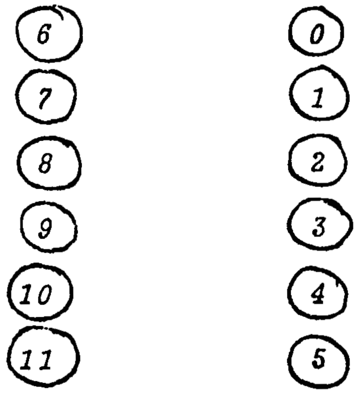
THE HALF CHARACTER MATRIX

The dots of a half character are shown below. The numbers represent the corresponding bits of the intensification word.

For instance:

100001111111 would represent the left half of an "0"

111111100001 would represent the right half of an "0"



CHARACTER SIZE:

Two character sizes are available: full size and half size. Half size characters are specified by executing the LINC made instruction:

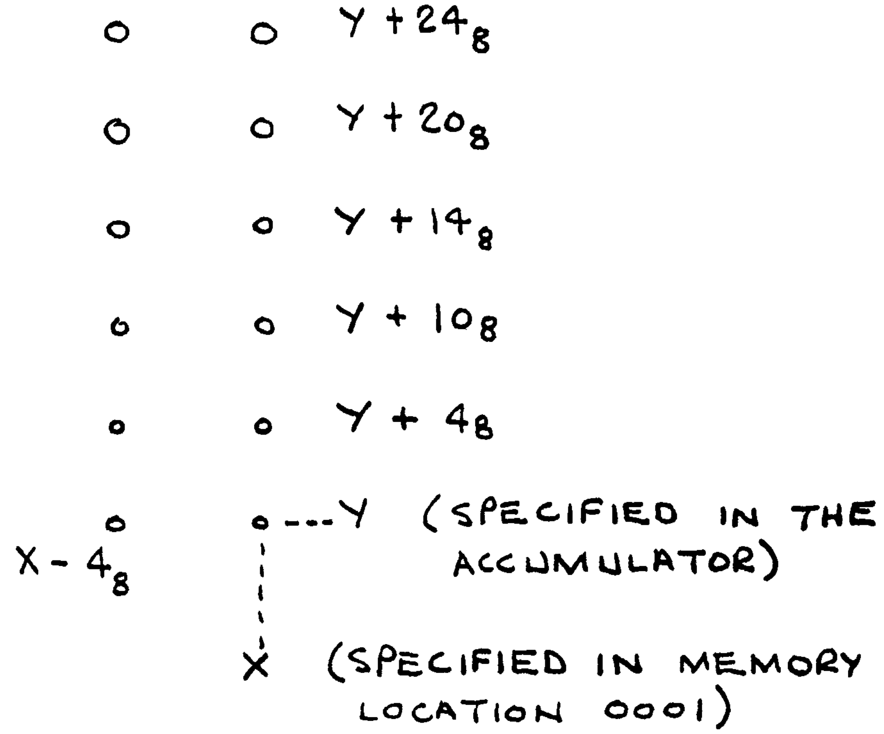
ESF (0004) EXECUTE SPECIAL FUNCTION - with the accumulator containing 0400. Full size characters are executed after I/O PRESET is generated.

CHARACTER LOCATION:

The horizontal position of a character is specified in memory location 0001.

The vertical position is specified in the accumulator

SPACING OF FULL SIZE CHARACTER MATRIX



CHARACTER SPACING:

a) Horizontal:

*Each time the DSC instruction is executed, the horizontal coordinate location register 0001 is incremented by 10_8*to automatically space the characters horizontally. It is thus not necessary for the programmer to modify the contents of 0001 except at the end of a line of text when the next character is to start at the left side of the CRT.*

b) Vertical:

When the ESC instruction is executed, the following operations take place:

1. *bits 7 to 11 of the accumulator are cleared*
2. *bits 3 to 11 of the accumulator are transferred into the vertical buffer*
3. *the character is displayed with its bottom row at the level specified by the accumulator*

* *for full size characters. For half size the spacing is* 4_8

EXAMPLE

If the accumulator contains 0273_8 when the DSC instruction is executed, then 0240_8 will be transferred to the vertical display buffer and the character will have its base at $Y = 0240_8$.

4. When the DSC instruction is finished, the accumulator has 30_8 added to it.

In the example above, the accumulator would contain 0270 at the finish of the DSC instruction.

Thus to move the character line up one, at least 10_8 must be added to the accumulator. To move down one line, 40_8 must be subtracted from the accumulator.

<u>EXAMPLE:</u> A programme to display the letter 'B' in the centre of the CRT screen.

```
0001    0000           / Channel select and horizontal coordinate register

0005    0000           / Address of the character table

0200    CLA CLL        / Clear AC and Link
0201    TAD SX         / Load starting horizontal coordinate into AC
0202    DCA   1        / Deposit in β 1
0203    TAD STABLE     / Load starting address of character table into AC
0204    DCA   5        / Deposit in β 5
0205    LINC           / Change to LINC mode
0206    DSC   5        / Display the character referenced by β 5
0207    DSC I  5       / Index the contents of β 5 and display the
                         character referenced by β 5.
0210    PDP            / Change to EIGHT mode
0211    JMP 0200       / Return and repeat
0212    SX, 0400       / Starting x coordinate
0213    STABLE, 0214   / 1st address in the character table
0214    5177           / Left half of pattern word for letter 'B'
0215    2651           / Right half of pattern word for letter 'B'
```

EXAMPLE : A programme to display "DURHAM on the CRT display COLLEGE"

```
0001    0000            / Horizontal coordinate
0005    0000            / Pattern word address
0200    CLA CLL
0201    TAD M 17        / Load - 17 into AC
0202    DCA K 5A        / Deposit in K5A
0203    TAD M16
0204    DAC  K5B
0205    TAD 5x          / Initialize horizontal
0206    DCA    1        / Position register
0207    TAD STABLE -1   / Load 0227
0210    DCA   5         / And deposit in β5
0211    TAD 5Y          / Load vertical coordinate into AC
0212    DURHAM, LINC
0213    DSC I 5         / Increment β5  and display character referenced
0214    PDP
0215    15Z K 5A        / Finished first line?
0216    JMP DURHAM      / No, continue with the first line
0217    JMS RESET       / Yes, alter horizontal and vertical coordinattes
0220    COLLEGE, LINC
```

```
0221        DSC I 5
0222        PDP
0223        I5K K5B          / Done second line?
0224        JMP COLLEGE      / No, continue with the second line
0225        JMP 0200         / Yes, repeat entire programme
0226        STABLE -1, 0227
0227        SC, 0344
0230        4177
                    D
0231        3641
0232        0000              SPACE
0233        0176
                    U
0234        7601
0235        0000              SPACE
  .           .
  .           .            }  OTHER PATTERN WORDS
  .           .
0274        4577
                    E
0275        4145

0300        RESET, Ø         / Subroutine for setting up horizontal and vertical
                                coordinates set start of second line.
0301        TAD R            / Read new horizontal coordinate into AC
0302        DCA   1          / Deposit in β1.
```

```
0303        TAD M40         / Lead - 40 (0740) into AC to cause Y to be
                              decreased by 1 line

0304        JMP I RESET     / Return to main programme

0310        M, 315

0320        M16, 7761

0321        K5A, 0000

0322        K5B, 0000

0323        5Y, 0000

0324        M40, 0740

0325        M17, 7754
```

Programme developed by

Mike Creamer and

Frank Nielson

OPERATION OF THE PDP-12 A/D CONVERTER

There are two methods of inputting analogue voltages into the PDP-12 in order to have them converted to digital form. The first is via analogue channels 0_8 through 7_8 - the control knobs located on the power switch panel to the left of the main control panel. These are ten-turn potentiometers which produce between plus one volt and minus one volt, corresponding to $+777_8$ and -777_8 in digital form.

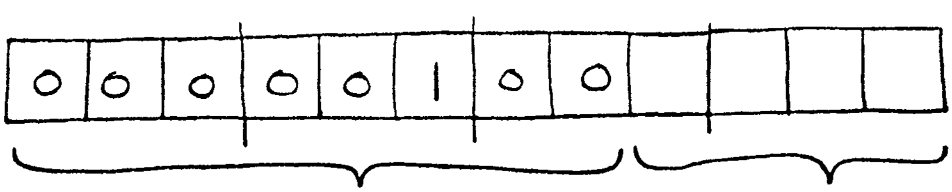
The second method of connecting inputs into the PDP-12 is via the phone jacks located below the power switch panel. These jacks correspond to input channels 10_8 through 17_8 and have a permissible voltage input range of plus one volt to minus one volt.

The PDP-12 has only one A-D converter, so it is necessary that only one input at a time be connected to the converter. The selection of the particular input (from the sixteen knob and jack inputs) is doen by a "multiplexer," the multiplexer being controlled by the computer programme instruction.

In addition, a particular analogue input may be changing at the time it is to be converted to digital form. For this reason, a "sample and hold" circuit is used to sample the voltage input and hold this sample while the analogue to digital conversion is taking place.

The "Sample" Instruction: SAM N

This is a LINC mode instruction, so the processor must be in LINC mode when the instruction is executed.



OPERATION CODE *SELECTED CHANNEL*

Ø - 17_8

The sequence of operations for a SAM instruction is as follows:

1. *The input called for by the instruction is selected by the multiplexer.*

2. *The analogue input is sampled by the sample-and-hold circuit and converted to a digital number by the A-D converter.*

3. *The accumulator is cleared.*

4. *The digital output from the A-D converter is transferred into the accumulator.*

The total length of time for one complete sampling operation is 18.2 microseconds.

The Sampled Output:

The sampled output from the A-D converter ranges between $+777_8$*(0777) and* -777_8*(1000). Negative numbers are represented in complimentary notation.*

Displaying the Sampled Output

The sampled output is 9 bits plus sign. The CRT display accepts 8 bits plus sign. It is thus necessary to scale the output from the A-D converter, one bit position to the right, before displaying.

No special provision for negative numbers need be made, since both the A-D converter and x-y display use complimentary notation.

Fast Sample Mode

In some situations, 18.2 microseconds is unsatisfactorily long for sampling time, in which case one may use "fast sample" mode. Fast sample is enabled by executing the instruction:

ESF (0004)

with 0100 in the accumulator.

Normal sampling can be restored by pressing I/O PRESET

When SAM N is executed with fast sample enabled, the order of events is as follows:

1. *The accumulator is cleared*

2. *The digital number in the A/D converter is transfered into the accumulator.*

3. *A new sample is taken and converted to binary form.*

Steps 1 and 2 take only 1.8 msec so no time is lost waiting for the conversion to take place. Step 3, the time consuming process, can take place while other programming instructions are taking place. However, a total of 18.2 micro-seconds must elapse before a new sample is called for. If the D/A converter is not ready, the processor will automatically wait until it is.

Problems

1. *Write a programme to display a square on the CRT screen, the size of which is to be controlled by potentiometer number 3.*

2. *Write a programme to display a graph of the output of potentiometer, versus time.*

3. *Write a programme to retrieve a data word from memory and display its value on the CRT. The address of the data word is to be selected by potentiometer number 2.*