

PDP-1 COMPUTER
ELECTRICAL ENGINEERING DEPARTMENT
M. I. T.
CAMBRIDGE 39, MASSACHUSETTS

PDP-29
INTRODUCTION TO PDP-1 TS
SYSTEM PROGRAMS

April 15, 1965

INTRODUCTION TO THE PDP-1 TIME SHARING SYSTEM

GENERAL INFORMATION

This memo was written to give the new user a brief introduction to the PDP-1 time sharing system programs. The material presented is sufficient information needed to operate in the TS system; however, only the essential features of the system programs are described. Other useful and time saving features are described in the respective program memo.

The computer has, in addition to a 4096-word core memory, an extra amount of memory in the form of a magnetic drum. The storage space on the drum is broken up into a number of "fields", each of which has the capacity to store 4096 18-bit words. This drum memory (besides being used for the time-sharing operation) can be used to store programs and data when there is not enough room in core. It is also used as a permanent storage space for "utility" programs which can be used by you.

To execute a program that you have written for the PDP-1, you must do the following things:

- 1) Tell the computer, in some way, the sequence of machine-language instructions that make up your program.
- 2) Have the computer translate or assemble these instructions into a sequence of binary words.
- 3) Place this binary program into core memory and have it executed, and
- 4) check your answers to see if your program is working correctly.

The permanently stored utility programs will be of help to you in accomplishing the above tasks. One of the programs, EXPENSIVE TYPEWRITER, assists you in reading in your machine-language program. The MACRO program will assemble your machine-language program, producing a binary program. DDT is an all-purpose program that will see to it that your program is executed. It also allows you to inspect and test your binary program for proper operation.

Each of the three utility programs mentioned above is stored (in binary form) on a different field of the magnetic drum. Of course, in order to use a program you must first place it in core memory. This is done by typing the commands shown in Figure 1.

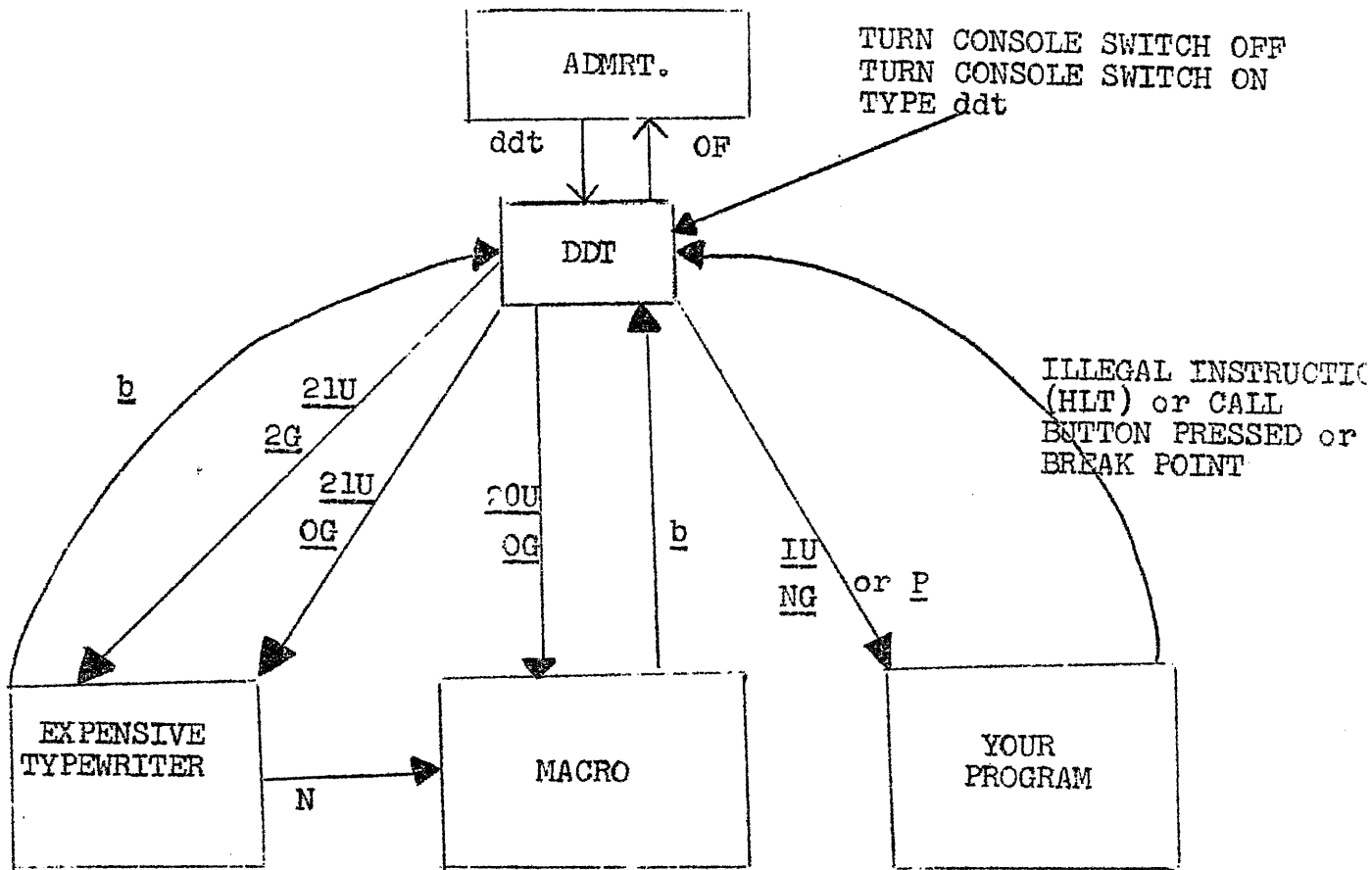


Figure 1.

For example, turning the console switch off, then turning it on, then typing ddt * will bring the program DDT from drum to core. Anything typed by you at this point will be interpreted as a command to DDT. Typing 21U then OG will cause EXPENSIVE TYPEWRITER to be brought into core. Anything typed at this point will be interpreted as a command to EXPENSIVE TYPEWRITER.

The features of each of the programs will now be described.

* The letters typed by you will be underlined in this paper. Those typed by the computer will not be underlined. The symbol ↵ is a carriage return.

EXPENSIVE TYPEWRITER:

This program is used for the reading in or typing in of your machine language program. It is also valuable in modifying your machine language program -- with it you can do such things as changing lines, inserting lines, deleting lines, etc. This program has a buffer storage area where a copy of your machine language program is kept. When you enter EXPENSIVE TYPEWRITER by typing 2IU then OG to DDT, this buffer area is cleared. Entering E.T. by typing 2IU then 2G will not clear this buffer.

E.T. has two operating "modes", control mode and text mode. In the control mode, characters typed in are commands to EXPENSIVE TYPEWRITER; in the text mode, typed input is copied directly into the buffer. The only exception to this is the character "backspace" which in the text mode has the effect of cancelling the previous character. To go from text mode to control mode, type a backspace immediately following a carriage return or after deleting all characters from a line.

Certain commands in the control mode put E.T. into the text mode. In the control mode, the typewriter prints in red, and in the text mode it prints in black. A command is terminated by typing in a carriage return or tab; at any time before the terminator is typed the command can be cancelled by typing the character middle dot (*).

Commands to EXPENSIVE TYPEWRITER:

In the examples below, the letter N is used to signify the number of a line. For example, the title line of your program will be line one, etc.

1. Input:

a: append. Enter the text mode; add the following typed text onto the end of the previous contents of the buffer.

ni: insert. Enter the text mode; insert the following typed text immediately before line N. That is, the first line of the inserted text becomes the new line N. The text is squeezed in between the old lines N-1 and N; no material is lost.

nc: change. Enter the text mode. Line N is entirely deleted and the following typed text is inserted in its place; any number of lines may be inserted.

r: read. The tape in the reader is read in and appended to the end of the buffer.

II. Deletion of Information in the Buffer:

Nd: delete. Line N is deleted.


K: kill. The entire buffer is cleared.

III. Printout:

W: write. The buffer is printed out (in black).

Nl: line. Line N is printed out (in red).

backspace: prints out the next line

: prints out the previous line.

IV. Punching Tape

P: punch. This punches the buffer out on paper tape.

V. Transfer

N: transfer control to MACRO

b: transfer control to DDT

Typical Operating Procedure:

Probably the first time you use the computer you will enter E.T. by typing 21U then OG. Your program buffer will be clear. If you have prepared a FIO-DEC program using the off-line flexowriter, you may read it into the EXPENSIVE TYPEWRITER text buffer by typing r. In reading the tape in, first, insert your tape with the 5-holed side toward you. The input side is the right and the tape feeds from right to left. Turn the reader switch on at the main console by lifting it up. Typing r will read in the tape into the buffer. After the tape is read in, remove it from the reader and turn the reader switch off by pressing it down. If a tape of your program has not been made, typing a, then your program, will put your program into the buffer. After correcting any typing mistakes using the delete, change, etc. commands, you will be ready to enter the MACRO Program where your program will be assembled. To enter MACRO type a capital N.

When you have a program in E.T.'s buffer, you can get a punched copy by typing the command P. Your tape will automatically be punched. After punching, press the tape-feed button on the main console to give you a couple of extra fanfolds of tape, and tear off your tape. This tape can now be used on another attempt to run your program. In this way it is not necessary to retype your entire program every time you run. You should also get a typewritten copy of the machine language program by typing W. Lifting Sense Switch 1 while the buffer is being typed out by a w will stop the typing.

The E.T. has one all-purpose error signal - ?. If a ? is typed by E.T. after you have typed something, examine what you have done carefully.

Important Reminders:

1) The meaning of the backspace key to EXPENSIVE TYPEWRITER differs depending upon the mode of operation. If E.T. is in the control mode, the backspace key means: print out the next line (if no previous line has been mentioned, this will print out the first line). If E.T. is in the text mode, the backspace key has two meanings.

If a letter of text has been typed by you a backspace erases it. If a backspace follows a carriage return typed by you, it means: return to the control mode. Do not attempt to type a command to EXPENSIVE TYPEWRITER unless it is in the control mode.

2) After you type in your program and leave EXPENSIVE TYPEWRITER, you will probably want to return later to make further changes to your program. When entering E.T. from DDT the second time, be sure to enter by typing 2IU then 2G. A OG will erase your program entirely.

3) Remember every command to EXPENSIVE TYPEWRITER must be followed by a carriage return before it is executed.

4) It is a good idea to print out the line you wish to modify using the command Nl before and after the change is made. This way you can be sure that you are changing the right line and that the change was made correctly.

5) After making any changes to your machine-language program with EXPENSIVE TYPEWRITER you must always reassemble your program with MACRO if you want to get a new binary program. Therefore, you should leave EXPENSIVE TYPEWRITER using the N command to MACRO rather than the b command to DDT.

MACRO (*Nightmare Version*)

The Macro program assembles the machine-language program which is located in the buffer of EXPENSIVE TYPEWRITER. The assembled binary program is placed automatically on drum field 1.

Commands to MACRO:

- c: begin assembly
- s: continue assembly
- z: print symbols (if Sense Switch 2 is up)
- b: transfer to DDT

Operating Procedure:

When you enter MACRO with your machine-language program stored in the buffer of EXPENSIVE TYPEWRITER, typing a c will begin pass 1 of assembly. If an error is discovered during assembly an error signal will be printed out in the format

described below and the assembly process will cease.

If an error is found on pass 1, first type g to continue pass 1. Then return to EXPENSIVE TYPEWRITER by typing b then 21U then 2G, in order to correct your machine language program. After the corrections have been made, return to MACRO with N and try pass 1 again. If pass 1 assembles correctly, no error message will be printed out. In this case, type c to initiate pass 2. Error in assembly during pass 2 result exactly as in pass 1 and you should proceed as described above.

If both passes assemble correctly you may, if you desire, receive a print of your table of symbols by: 1) Setting Sense Switch 2 up; 2) Typing z; ~~_____~~
~~_____~~

Error Signals:

Upon detecting an error, MACRO will print out the following:

aaa bbbb ccc dddd eee

aaa is the three letter code indicating the error. bbbb is the octal address at which the error occurred. ccc is the symbolic address at which the error occurred. dddd is the name of the last pseudo-instruction encountered. In the case of an error caused by a symbol, eee will be the symbol. Following is a list of the error indications in MACRO:

ERROR	MEANING
us.?	An undefined symbol has been encountered by MACRO. <u> ? </u> will indicate how the undefined symbol was being used.
a	In a pseudo-instruction argument
w	In a word
c	In a constant
p	In a parameter assignment (assignment with an equal sign)
m	In a MACRO instruction definition
l	In a location assignment
s	In a start pseudo-instruction
d	In a dummy symbol assignment
mdt	Multiple definition of a tag. The definition of this address tag does not correspond to a previous definition.

mdm	Multiple definition of a MACRO instruction The name of this MACRO instruction conflicts in its first 6 characters with a pseudo-instruction or some other MACRO instruction name.
zpa	Illegal parameter assignment
mdv	Multiple definition of a variable.
ipi	An illegal pseudo-instruction
ids	Illegal dummy symbol in the title of a MACRO instruction definition
sce	Storage capacity exceeded
ilf	Illegal format
tmc	Too many constants
tmv	Too many variables
tmp	Too many parameters (dummy symbols)

DDT:

This program is a very valuable aid in debugging and testing your binary program. It allows you to examine the contents of locations of core, to change the contents of these locations, to run your program, and to make use of breakpoints. When in DDT, all typing will be performed in black. After a command is typed by you, DDT will type a carriage return to indicate that the command was carried out. Therefore, do not type a carriage return after a command as you would do in EXPENSIVE TYPEWRITER.

Commands to DDT:

Control

IT. Typed immediately after returning from MACRO assembly. This places your symbols and their values with the DDT program. This allows you to use symbolic addresses when communicating with DDT.

NU: where N is a number. This takes the program from drum field N and places it in core memory so that it can be executed. Your binary program has been placed on drum field 1. EXPENSIVE TYPEWRITER is on drum field 21. MACRO is on drum field 20.

locG: This will cause the computer to begin executing the program which is in core starting at location loc. loc may be either a symbolic expression or an absolute number.

For an example of the above commands:

IT
IU
begG

will clear DDT's old symbol table and enter yours from MACRO, bring in your program from drum to core, and begin executing your program from location beg.

Inspecting and Modifying:

Examine a Register


adr/: Types out the contents of register adr.


"adr" may be a symbolic address or an octal number.

Examples: beg+2/ types out the contents of the register 2 positions past beg. 55/ types out the contents of register 55.

Change a Register

adr/

exp : Changes the contents of register adr to exp.

Example: k/ 4 13  examines the contents of the register k and finds it contains the number 4. This is changed to 13.

Set the Mode

S Sets DDT to type out all words as symbolic instructions.

C Sets DDT to type out all words as octal numbers.

Examples: S
20/ and k
C
20/ 020003 (if k = 3)

U Converts all numeric printouts to decimal

H Resets DDT so that all numeric printouts are in the normal octal mode again.

Examples: C
20/ 020003
U
20/ 8195
H
20/ 020003

Convert an Expression

exp = Types out the value of the expression exp as an octal number.

num -> Types out the octal number num as an instruction.

char ~ Types out the character(s) char as concise code.

Examples: 20/ 20003 -> lac k
(if k = 3)
beg = 20 (if beg is the symbol for location 20)
a40~ 610420

Examining Sequences of Registers

adr/ ... ↑

Types the contents and opens the register preceding adr.

Example: tab/ 13 ↑
 tab-1/ 21 ↑
 tab-2/ 0

Register tab contained 13, tab-1 contained 21, and tab-2 contained 0.

adr/ ... (bs)

Types out and opens the contents of the register next in sequence to adr.

(bs) means back-space

Example: tab/ 13 (bs)
 tab+1 61

Register tab+1 contains 61

adr/ ... >|

Types out the contents of the register addressed by the contents of register adr.

Example: beg/ lac 13 >| 44
Register 13 contains 44

Breakpoint Commands

It is tempting to load a newly coded program into the computer, let it run to completion, and check the final answer for correctness. Because of the high probability of one or more errors in even short programs and the cumulative effect of these errors, this approach is practically worthless (except for very short programs).

A more satisfactory approach is to run only a small portion of the program at a time and check intermediate results. This allows one to catch errors before they have a chance to make the whole operation unintelligible. The breakpoint feature of DDT permits one to take this approach and check out one section of the program at a time.

A breakpoint is a place in the user's program where computation is interrupted and control is transferred to DDT. This is accomplished by removing and saving the instruction at the breakpoint and inserting in its place a jump instruction to DDT.

When the program reaches this jump, the status of the machine (AC, IO, overflow indicator, etc.) is saved and the breakpoint location and contents of the AC are typed out.

At this point one may examine the intermediate results by use of any of the inspection and modification commands listed above. If they are satisfactory, one may proceed with the next section of the program. If they are in error, one may try to correct the errors and re-run the section.

The DDT breakpoint commands are as follows:

locB Prepares DDT to insert a breakpoint at location loc. The actual insertion occurs when a G or X command is given.

B Removes the previous breakpoints.

P Given after a program has been interrupted by a breakpoint. The instruction removed for the breakpoint is executed and control is returned to the user's program.

insX Causes instruction ins to be executed.

Examples:

```
lupB  
begG  
lup)      0  
k/       4  
B  
P
```

This example places a breakpoint at location lup and transfers control to beg. The AC is zero and location k contains 4 at the break. The breakpoint is removed and the program is allowed to proceed.

dzm conX deposits +0 register con