1.2 DDT Prompt Invocation Method

You respond to the system prompt with the following command:

```
A>DDT  RETURN
```

The DDT utility will identify itself with the following display:

```
DDT VERS n.n
-
```

where "n.n" is the version number; and

where "-" (the hyphen character) is the DDT prompt, at which you can enter special DDT commands.

To load a file from the default disk into memory under this invocation method, you must now make the following entries, in order: the letter **I**, (no space), the complete file name of the program to be loaded, a carriage return, the letter **R**, and another carriage return.

The DDT utility will **I**nsert the name of the program into memory, **R**ead the named program into memory, and display values for both the "NEXT" available address and the "PC" program counter. For example, if you desire to debug the program file named "PROGRAM.HEX", the console display of these transactions might appear as follows:

```
A>DDT  RETURN
DDT VERS 2.2
-IPROGRAM.HEX  RETURN
-R  RETURN
NEXT PC
3E80 0100
-
```

Where "3E80" is the next available memory address after the end of the program (or the address after the last address occupied by the loaded program);

where "0100" is the current value of the program counter (or the first memory address occupied by the program); and

where "-" (the hyphen character) is the DDT prompt, at which you can enter special DDT commands.

LOADING A PROGRAM FROM A NON-DEFAULT DRIVE

Using the second DDT invocation method, you can access a file from a non-default disk by doing the following, in order:

- Enter the letter **I**, the complete file name of the program to be loaded, and a carriage return.
- Enter the character string **S5C** and a carriage return. This entry will produce a six-character display.
- Enter the two-digit drive number for the drive containing the program that is to be debugged, and a carriage return. (See the table below to find the drive number that corresponds to the appropriate drive letter. This table will not be displayed on the console.)

Drive Number	Drive Letter
00	DEFAULT
01	A
02	B
03	C
04	D
05	E
06	F

NOTE: "DEFAULT" is the drive logged before DDT invocation.

The drive number you entered will be displayed on the right of the six-digit display, and a second six-digit display will appear beneath.

- Enter a . (period) and a carriage return. This entry will be displayed on the right of the second six-digit display.

- Enter the letter **R** and a carriage return. DDT will read the program file into memory, and display values for both the "NEXT" available address and the "PC" program counter.

For instance, to access the file named "PROGRAM.HEX" from the disk within non-default drive B (02), you must interact with DDT as shown in the following display:

```
A>DDT  RETURN
DDT VERS 2.2
-IPROGRAM.HEX  RETURN
-S5C  RETURN
005C 00 02  RETURN
005D 57 .  RETURN
-R  RETURN
NEXT  PC
3880 0100
```

2 SAVING A DEBUGGED PROGRAM

You can preserve the results of a debugging session by copying a block of data from the Transient Program Area (TPA) to a disk file. The SAVE resident command will assist in this task. SAVE copies a user-specified number of pages from the TPA to a user-specified file name on a disk.

To SAVE a program that has been loaded into memory and debugged by DDT, first exit from the DDT utility to the operating system. If the exit is performed properly, the image of the program in the TPA will remain undisturbed as SAVE copies from it to a disk file.

You can exit from the DDT utility by either of the following two methods:

- **Performing a Warm Boot** — Any time the DDT hyphen (-) prompt is displayed, enter a ***CTRL-C*** (by pressing the **C** key while holding down the ***CTRL*** key). CP/M will display the system prompt.
- **Jumping to the Operating System's Execution Address** — Any time the DDT hyphen (-) prompt is displayed, you can trigger execution of the operating system, by entering the DDT command **G0** (where "0" is zero). This command sends the program counter to computer memory 0000H (zero), where the operating system resides. CP/M will display the system prompt.

You must proceed immediately to entry of the SAVE command. The SAVE command is entered in the form:

A>**SAVE {pages} {file name}** ***RETURN***

Where {**pages**} are 256-byte units of data that are expressed in decimal (not hexadecimal) numerals, and

where {**file name**} specifies the file name under which you wishes to store the results of the debugging session.

You can SAVE the entire program by determining the (decimal) number of pages between the "pppp" value and the "aaaa" value displayed when DDT initially loads the program into the TPA, as shown:

```
NEXT  PC
aaaa pppp
```

Unless the program has been moved from the beginning of the TPA, the two left-hand digits in the "aaaa" value will be the hexadecimal number of pages the program occupies, as long as the two right-hand digits are not zeros. If the two right-hand digits are zeros, one should be subtracted from this total to determine the number of hexadecimal pages. In either case, the hexadecimal number of pages should be converted to a decimal number suitable for the SAVE command.

3 DDT COMMANDS

The DDT utility has its own assortment of commands. They are entered in response to the DDT hyphen (-) prompt. DDT command lines begin with a single command letter. These command letters are defined in the following list:

A	Assemble: Assembly language mnemonics are inserted
D	Display: Display memory contents in hexadecimal an ASCII form
F	Fill: Fill a block of memory with a specified data constant
G	Go: Go to specified address to run a program
H	Hex: Hexadecimal computation of sums and differences
I	Insert: Insert file name into file control block
L	List: List assembly language mnemonics of a program
M	Move:Move a data block
R	Read: Read a file into memory from disk
S	Substitute: Substitute hexadecimal values
T	Trace: Trace program execution
U	Untrace: Untrace program execution
X	eXamine: Examine or change registers or flags

Most of these command letters can be followed by parameters such as hexadecimal values or a file name. All DDT command lines must end with a carriage return.

When hexadecimal parameters are used, these values consist of one to four digits. (Longer numbers are automatically truncated on the right.) One, two, or three such values can be entered in some command lines. Values are separated by commas or single blank spaces.

Only one DDT command can be entered in response to a single DDT hyphen (-) prompt. Each DDT command line, however, can be composed using most of the same line editing keys and techniques as are used for commands entered at the CP/M system prompt.

No DDT command line can exceed 32 characters in length. If a thirty-third character is entered, it is interpreted as a carriage return and execution begins based on the first 32 characters in the command line.

Many DDT commands operate under a "CPU state" which corresponds to the program being tested. The CPU state holds the program's registers. Initially, all registers and flags contain zeroes—except for the program counter (P) and the stack pointer (S), which default to the value 100H.

The program counter is a CPU register that is used as a movable reference point for DDT commands. It keeps track of the last hexadecimal address that was displayed and/or altered by a DDT command. The address immediately after this last address will be the starting address for the next DDT command you enter (unless the next command specifies a different starting address).

3.1 A Assembly Language Mnemonics are Inserted

The A (assembly) command enables you to insert assembly language instructions into the program being tested. The command is entered in the form:

As

Where **s** is the memory address at which you desire to start inserting assembly language instruction statements.

DDT responds to such an entry by echoing the value ("s") entered. You can then enter an assembly language statement to the right of the echoed "s" value. The statement must end with a carriage return.

DDT will display the next available memory address after the new statement is appended to the program. You can enter another statement to the right of the displayed address, or enter a carriage return alone to end A command operations and retrieve the DDT prompt.

For example, if you want to insert a "MOVE IMMEDIATE to register C" statement into a program at memory address 0104H, the following entry should be made in response to the DDT prompt:

A104 ***RETURN***

DDT will echo the address with the display:

```
0104
```

To the right of the address display, you can insert the statement:

MVI C,{data} ***RETURN***

Where **{data}** is the data to be moved into register C.

DDT will then display the value for the next available memory location. Since the "MVI" statement takes up two locations, DDT displays:

```
0106
```

You can enter another statement at location 0106H or end the operations of the A command by entering a carriage return alone.

NOTE: When the A command inserts a statement at a particular memory address, the statement(s) that formerly occupied that part of memory will be overwritten, and therefore destroyed. If you insert an statement that does not occupy the same number of locations as the statement(s) being replaced, the meaning of subsequent statements might be changed. You should use the L command immediately after finishing A command operations, to verify that the desired results were achieved during use of the A command.

The following example demonstrates this problem. You want to replace a "jump" instruction (JMP) with a "return from subroutine" statement (RET). The JMP statement occupies three locations, and the RET instruction will occupy one. Inserting the one-byte RET into the first location of the three-byte JMP will leave the last two-thirds of the JMP statement in the program. This partial statement could cause problems when the program is run.

3.2 D Display Memory Contents in Hexadecimal and ASCII Form

The D (display) command allows you to view the contents of memory in both hexadecimal and ASCII formats. The display appears in the following form:

```
aaaa bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb cccccccccccccccc
```

Where "aaaa" is the address of the first memory location displayed in this line;

where "bb" represents the hexadecimal contents of a memory location; and

where "cccccccccccccccc" represents the ASCII translation of the contents of each memory location.

If the contents of a memory location cannot be displayed as an ASCII character, a period (.) will be displayed instead.

The display address acts as a pointer in memory which is initially set to 100H. As each memory location is displayed, the pointer is incremented by one so at the the end of a D command, the pointer is positioned ready to display the next 256 memory locations.

The four forms of the command are:

D Display memory from the current display address until one page of data (256 locations) have been shown. DDT will display 16 lines, each representing 16 bytes of data.

Ds Change the starting display address to "s", then display memory beginning with address "s" and continuing until one page of data (256 locations) is displayed.

Ds f Change the starting display address to "s", then display memory beginning with address "s" and continuing until memory address "f" is reached.

D,f Display memory from the current display address until address "f" is reached.

Displays triggered by any of these commands can be suspended if a ***CTRL-S*** is entered during the display. The display will resume if anything other than CTRL-C is entered.

A display will be aborted if any character other than CTRL-S is entered. However, it is recommended that the display be intentionally aborted by pressing the DELETE key, because other characters will appear at the next DDT hyphen (-) prompt if they are used to abort the display.

For example if the file SYSGEN.COM had been loaded into memory by DDT and the user wanted to see a hexadecimal and ASCII display of it, then **D** should be entered at the DDT hyphen prompt. This entry would cause a display something like the following:

```
0100 C3 79 02 43 4F 50 59 52 49 47 48 54 20 28 43 29 .y.COPYRIGHT (C)
0110 20 31 39 37 38 2C 20 44 49 47 49 54 41 4C 20 52 1978, DIGITAL R
0120 45 53 45 41 52 43 48 20 6F 26 00 29 29 29 29 29 ESEARCH o&.)))))
0130 29 29 C9 0E 01 CD 05 00 FE 61 D8 FE 7B D0 E6 5F ))........a..{..
0140 C9 5F 0E 02 CD 05 00 C9 3E 0D CD 41 01 3E 0A CD . ......>..A.>..
0150 41 01 C9 E5 CD 48 01 E1 7E B7 C8 E5 CD 41 01 E1 A....H..~....A..
0160 23 C3 58 01 D5 4F 2A 01 00 11 18 00 19 D1 E9 2A #.X..O*........*
0170 01 00 11 1B 00 19 E9 2A 01 00 11 1E 00 19 E9 2A .......*.......*
0180 01 00 11 21 00 19 E9 2A 01 00 11 24 00 19 E9 2A ...!...*...$...*
0190 01 00 11 27 00 19 E5 21 14 07 3A 12 07 BE F5 35 ...'...!..:.....5
01A0 CA AE 01 F1 0E 00 C2 B7 01 0E 02 C3 B7 01 F1 3A ...............:
01B0 12 07 32 14 07 0E 01 E1 E9 0E 14 C3 05 00 0E 0F ..2.............
01C0 C3 05 00 21 00 09 22 08 07 3E FF 32 05 07 21 05 ...!..".>.2..!.
01D0 07 34 7E B7 CA EB 01 3A 0F 07 32 0C 07 3A 13 07 .4~....:..2..:..
01E0 32 12 07 E5 2A 10 07 22 0D 07 E1 3A 12 07 32 14 2...*.."...:..2.
01F0 07 3A 0B 07 BE CA 78 02 4E CD 6F 01 3E FF 32 06 .:....x.N.o.>.2.
```

3.3 F Fill a Block of Memory with a Specified Data Constant

The F (fill) command allows you to fill a block of memory with a specific constant. The form of the command is:

Fs f c

Where **s** is the address at which the filling should begin;

where **f** is the address at which the filling should end; and

where **c** is the data constant that should occupy each memory address in between.

Any data that resided between addresses "s" and "f" prior to the entry of the command will be overwritten by the constant, and therefore destroyed.

Only hexadecimal values should be entered in such a command, and value "f" must be greater than value "s". If "s" is greater than "f", the operation will not be executed and the DDT prompt will reappear.

For example, the following command:

- F9200 B400 E5 ***RETURN***

would fill every memory location from address 9200H through address B400H with the hexadecimal value "E5".

3.4 G Go to Specified Address to Run a Program

The G (Go to) command enables you to begin execution of the program from any address, and to specify one or two execution breakpoints if desired. (A breakpoint is the address of an instruction which, when reached, will stop the execution of the program and redisplay the DDT prompt.) Execution begins with the instruction at the memory address immediately following the one specified in the command. The instruction at the specified address is not executed.

If no breakpoint is entered, the only other way in which the control of the program may be returned to you is if an "RST 7" instruction is encountered within the program. This instruction will immediately stop program execution and redisplay the DDT prompt to allow further DDT commands from you.

The G command can be entered in the following forms:

G Begins execution of the program at the current value of the program counter, with no breakpoints set. The program will run to completion.

Gs Sets the program counter to address "s" and begins execution of the program from that address, with no breakpoints set. The program will run to completion.

G,b Begins execution of the program at the current value of the program counter and continues until the instruction at address "b" (the breakpoint) is reached. Then program execution stops.

Gs b Sets the program counter to "s" and begins execution of the program at address "s". When the instruction at address "b" (the breakpoint) is reached, program execution stops.

G b c Begins execution of the program at the current value of the program counter and continues until either address "b" or address "c" is reached. When either of these breakpoint addresses is reached, program execution stops.

Gs b c Sets the program counter to address "s" and begins execution of the program at this address. When either address "b" or address "c" is reached, program execution stops.

At a breakpoint, program execution stops and DDT displays:

```
*bbbb

-
```

Where "bbbb" is the address at which program execution stopped; and

where "-" is the DDT prompt

For example, you could "go to" the very beginning of computer memory (address 0000H) and trigger execution of the program that is situated there. This program is, of course, the CP/M operating system. Its execution can be triggered by entry of the following command:

-G0 ***RETURN***

The operating system would respond by displaying the system prompt, as shown:

```
A>
```

This command has the same effect as a warm boot.

3.5 H Hexadecimal Computation of Sums and Differences

The H (hexadecimal value) command simultaneously adds and subtracts two hexadecimal values. This command is entered in the following form:

Ha,b

Where **a** is a hexadecimal value; and

where **b** is another hexadecimal value.

The resulting display appears in the form:

```
ssss dddd
```

Where "ssss" represents the sum of two values; and

where "dddd" represents the difference between the two values.

This command is helpful in determining addresses to which programs will be relocated with DDT command "M".

For example, if you have a program that begins at address 0311H, and wishes to move this program 0126H bytes higher in memory, then the H command could be used to calculate the new starting address, as shown:

```
-H311,126  RETURN
0437 01EB
```

0437H would be the new starting address for the program.

However, if you enter an "a" value that is smaller than the "b" value, the sum ("ssss") will be the same, but the difference ("dddd") will be equal to 10000H minus the amount that "b" is greater than "a".
Such a case is demonstrated by the following entry:

-H1,2 ***RETURN***

which will produce the following displayed solution:

```
0003 FFFF
```

3.6 I Insert File Name Into File Control Block

The I (input) command allows you to insert a file name into the area of memory that is used to store the names of files to be read from the disk. This area of memory begins at address 5CH. This is one of the memory areas from which the Console Command Processor (a functioning part of the CP/M operating system) distributes control to utilities and resident commands. This DDT command is entered in one of the following forms:

-I{primary file name} ***RETURN***

-I{primary file name}.{extension} ***RETURN***

If the second form of the command is used and the {extension} entered is either "HEX" or "COM", then subsequent R commands can be used to read the pure binary or hexadecimal machine code.

The I command will not read the file from the disk and store it into memory. It will only insert the filename into the File Control Block portion of the Console Command Processor, so that a subsequent R command can read the named file into memory.

3.7 L List Assembly Language Mnemonics of a Program

The L (list) DDT command enables you to disassemble the instructions wthin a span of memory, and to display the assembly language mnemonics of the disassembled code on the console. The command can be entered in any of the following three forms:

L Lists 12 lines of disassembled machine code, beginning at the current list address. The list address acts as a pointer in memory which is initially set to 100H. As each memory location is disassembled, the pointer is incremented by one.

Ls Changes the list address to "s", and then lists 12 lines of disassembled machine code beginning at address "s".

Ls f Lists disassembled code from starting address "s" to the final address "f".

The list appears in the form:

```
aaaa mmmm oooo
```

Where "aaaa" is the address of the instruction,

where "mmmm" is the mnemonic of the operator, and

where "oooo" is the operand.

Listings triggered by an L command can be suspended if a ***CTRL-S*** is entered during the listing. The listing will resume if anything other than CTRL-C is entered.

A listing will be aborted if any character other than CTRL-S is entered during the listing. If the listing is to be aborted intentionally, we recommend that the DELETE key be pressed, because other characters might appear at the next DDT hyphen (-) prompt if they are used to abort the listing.

If an invalid mnemonic is encountered in a statement of a disassembled program, question marks ("??") will be used to represent the invalid mnemonic in the listing.

The disassembled mnemonics from address 0919 through address 091D of the program in memory can be listed with the entry of the following command:

-L0919,091D ***RETURN***

Such a listing might appears as follows:

```
0919 OUT  F2
091A INX  H
091B MOV  A,M
091D ORA  A
```

3.8 M Move a Data Block

The M (move) command allows you to move a block of data from one area of memory to another. This command is entered in the form:

Ms f d

Where **s** is the starting address;

where **f** is the final address of the block of data to be moved; and

where **d** is the starting point of the memory area to which the data is moved.

The data is moved to the area of memory beginning at the address "d". An example of the command follows:

M100 200 1000 ***RETURN***

This command would take the contents of the block of memory starting at address 0100H and running through address 0200H, and move these contents to the area of memory beginning with the address 1000H.

3.9 R Read a File Into Memory From Disk

The R (read) command is used after the I command to read COM and HEX files from the diskette into the transient program area in preparation for a debugging operation. The R command requires a previous I command, specifying the name of the HEX or COM file to be read. The command can be entered in either of the following two forms:

R Reads the file whose name is in the file control block at address 5CH from the disk and places it in the Transient Program Area. (The file name was placed in this location with the I command.)

Rb Reads the file whose name is in the file control block at address 5CH from the disk and places it in the Transient Program Area with the addition of a bias factor, "b", which is a hexadecimal number added to each program instruction address or data address as it is read. This factor allows you to locate the program at any location in memory. When the bias factor is omitted, then b = 0000 is assumed.

The read operation must not place the file in the first page of memory (0-0FFH) because this would write over the system parameters which are stored in this area. If the file specified in the preceding I command is a HEX file, the load address is derived from each individual HEX record. If the file to be loaded is a COM file, a load address of 100H is assumed. Any number of R commands may be issued following an I command to reread the program under test.

The R command reads the desired file from the default drive. If the desired file resides on a non-default drive, the command **S5C** should be entered, and the value for the non-default drive should be substituted according to the following table:

Drive Number	Drive Letter
00	DEFAULT
01	A
02	B
03	C
04	D
05	E
06	F

This substitution should be performed between the I command and the R command.

When the R command loads a named file into the Transient Program Area, a message in the following form is displayed:

```
NEXT PC
nnnn pppp
```

Where "nnnn" is the the address immediately following the loaded program; and

where "pppp" is the current value of the program counter (100H for COM files, or it is taken from the last record if a HEX file is specified.)

The next address "nnnn" can be used to determine the size of the file which was loaded. If the beginning address is 100H, then subtracting 100H from "nnnn" will display the size of the program in bytes. The size derived in this manner is in hexadecimal units, and may have to be converted to decimal units before it is used.

3.10 S Substitute Hexadecimal Values

The S (substitute) command enables you to examine—and optionally alter—the contents of specified memory locations. This command is entered in the form:

Sb

Where **b** is the hexadecimal address of the first memory location to examine.

DDT responds with a display of addresses and bytes in the form:

```
aaaa cc
```

Where "aaaa" is the hexadecimal address, and

where "cc" is the hexadecimal contents of the memory location.

You may substitute a new value for "cc" by entering the new value (in one or two hexadecimal digits) and a carriage return when DDT displays "aaaa" and "cc". Your entry will appear on the right side of this display, and it will replace "cc" in the memory image of the program.

The next address "aaaa" and its contents "cc" are then displayed, inviting you to substitute a new value for this "cc". When you are finished altering address contents in this sequence of addresses, a period (.) and a carriage return should be entered, rather than a new value. Your alterations will be retained in memory, and the DDT hyphen (-) prompt will reappear.

If you wish to skip an address without changing it, then a carriage return (without a period) should be entered in response to one of the "aaaa" "cc" displays. The DDT hyphen (-) prompt will reappear.

For example, if you enter the following command:

-S100 ***RETURN***

then memory addresses and their contents will be displayed on the screen, starting with address 0100H, as shown. your substituted values for the address contents are in boldface print on the right side of the following example display:

```
0100 C3 3C RETURN
0101 C0 C RETURN
0102 01 10 RETURN
0103 20 RETURN
0104 43 RETURN
0105 4F F4 RETURN
0106 50 . RETURN
-
```

3.11 T Trace program execution

The T (trace) command allows to trace the execution of one to 65,535 (0FFFFH) program steps. During the trace, the contents of all registers and the status of all flags within the central processing unit (CPU) are displayed. This command can be entered in either of the following forms:

T Displays the contents of the CPU registers and the status of the flags; then executes one program instruction. The DDT hyphen prompt (-) reappears.

Tn Displays the contents of the CPU registers and the status of the flags; then executes "n" program instructions and stops. The DDT hyphen prompt (-) reappears.

Displays caused by the T command take the following form:

```
CfXfMfEfIf A=bb B=dddd D=dddd H= dddd S=dddd P=dddd inst *hhhh
```

Where "f" is a 0 or 1 flag value;

where "bb" is a byte value;

where "dddd" is a double byte quantity corresponding to a register pair;

where the "inst" field contains the disassembled instruction which occurs at the location addressed by the program counter; and

where "hhhh" is the next address available for execution.

The display address (used in the D command) is set to the value of the H and L registers. The list address (used in the L command) is set to the value of "hhhh" so it will be ready to list the next program steps to be executed if desired. Since the state of the flags and registers of the CPU displayed by the T command occur before each instruction is executed, it may be helpful to use an X command to view the state of the CPU after the trace command.

The second form of the T command will trace the execution for "n" steps ("n" is a hexadecimal value) before a program breakpoint occurs. A breakpoint can be forced during long trace displays by using the DELETE key. The state of the CPU is displayed before each program step is executed in the trace mode.

If the program being tested must access the disk or input/output (I/O) devices through the CP/M system, the program tracing is discontinued at the interface to CP/M, and resumes after returning from CP/M to the program being tested. Thus, CP/M functions which access I/O devices, such as the disk drive, operate at the proper speed (real time), thereby avoiding I/O timing problems. Programs running in the trace mode execute aproximately 500 times slower than real time because DDT gets control after each user instruction is executed. In programs which use interrupt instructions, the interrupts are always enabled during the trace mode.

3.12 U Untrace Program Execution

The U (untrace) command allows you to trace the execution of one to 65,535 (0FFFFH) program steps. During the untrace, the contents of all registers and the status of all flags within the central processing unit (CPU) are displayed. Intermediate program steps are not displayed. This command can be entered in either of the following forms:

U Displays the contents of the CPU registers and the status of the flags. Then executes one program instruction. The DDT hyphen prompt (-) reappears.

Un Displays the contents of the CPU registers and the status of the flags. Then executes "n" program instructions, and stops. The DDT hyphen prompt (-) reappears.

Displays caused by the T command take the following form:

```
CfXfMfEfIf A=bb B=dddd D=dddd H=dddd S=dddd P=dddd inst *hhhh
```

Where "f" is a 0 or 1 flag value;

where "bb" is a byte value;

where "dddd" is a double byte quantity corresponding to a register pair;

where the "inst" field contains the disassembled instruction which occurs at the location addressed by the program counter; and

where "hhhh" is the next address available for execution.

The display address (used in the D command)1 is set to the value of the H and L registers. The list address (used in the L command) is set to the value of "hhhh" so it will be ready to list the next program steps to be executed if desired. Since the state of the flags and registers of the CPU displayed by the U command occur before each instruction is executed, it may be helpful to use an X command to view the state of the CPU after the trace command.

The second form of the U command will untrace the execution for "n" steps ("n" is a hexadecimal value) before a program breakpoint occurs. A breakpoint can be forced during long untrace displays by using the DELETE key. The state of the CPU is displayed before each program step is executed in the untrace mode.

If the program being tested must access the disk or input/output (I/O) devices through the CP/M system, the program tracing is discontinued at the interface to CP/M, and resumes after returning from CP/M to the program being tested. Thus, CP/M functions which access I/O devices operate at the proper speed (real time), thereby avoiding I/O timing problems. Programs running in the untrace mode execute aproximately 500 times slower than real time because DDT gets control after each user instruction is executed. In programs which use interrupt instructions, the interrupts are always enabled during the untrace mode.

3.13 X eXamine or Change Registers or Flags

The X (examine) command enables you to display and alter the state of the registers and flags of the CPU at any time during the debugging process. This command can be entered in either of the following forms:

X

Xr

Where **r** is one of the 8080 CPU registers in the following table:

8080 CPU REGISTER SYMBOL	REGISTER NAME	RANGE OF REGISTER CONTENTS
C	Carry Flag	(0/1)
Z	Zero Flag	(0/1)
M	Minus Flag	(0/1)
E	Even Parity Flag	(0/1)
I	Inter-Digit Carry	(0/1)
A	Accumulator	(0-FF)
B	BC Register Pair	(0-FFFF)
D	DE Register Pair	(0-FFFF)
H	HL Register Pair	(0-FFFF)
S	Stack Pointer	(0-FFFF)
P	Program Counter	(0-FFFF)

The first form of the command displays the state of the CPU in the following form:

```
CfXfMfEfIf A=bb B=dddd D=dddd H=dddd S=dddd P=dddd inst
```

Where "f" is a 0 or 1 flag value;

where "bb" is a byte value;

where "dddd" is a double byte quantity corresponding to a register pair; and

where the "inst" field contains the disassembled instruction which occurs at the location addressed by the program counter.

The second form of the command displays the flag or register value of the specified register, and allows alteration of the hexadecimal value within this flag or register. You can substitute a new value for the value held in the register. You makes this substitution by entering the new value and a carriage return to the right of the existing value.

This example demonstrates how the value in a register can be altered.

```
-XS
S=00FE EF00   RETURN
-
```

When you substitute a value into a flag or register, ending the substitution with a carriage return, the DDT hyphen prompt (-) reappears. If you wish to make no changes to the values, then a carriage return alone should be entered.

If a value in the proper range is typed, then the flag or register value is altered. Note that BC, DE, and HL are displayed as register pairs. Thus you enter the entire register pair when B, C, or the BC pair is altered.

4 DDT Error Signals

DDT does not display entire messages when you make an erroneous entry. However, DDT will display a question mark (?) if your entry does not conform to valid entry syntax restrictions.

In addition, DDT will display question marks (??) in place of invalid mnemonics that it encounters when dealing with the assembly language form of a program.

DIR

The Resident Command that Displays Disk File Directories

The DIR resident command is issued to determine the presence of (1) all of the files on a disk, (2) a specified file, or (3) a group of specified files. After command entry, DIR displays file names to the console (4). Some file names cannot be accessed by DIR (5).

1 DIRECTORY OF All FILES ON A DISK

DIR can be used to determine the names of all files on a disk by answering the system prompt with the following entry:

A>**DIR** ***RETURN***

If you desire a DIRectory of the files on a disk that does not reside in the default drive (drive B for instance) the DIR command should be entered with a drive specification, as shown:

A>**DIR B:** ***RETURN***

NOTE: Because DIR is a resident command, it is automatically loaded into the computer with the rest of the operating system. Therefore, it is never necessary (or valid) to specify a drive at the **beginning** of a DIR command line. For example, the command A>**B:DIR** ***RETURN*** is invalid.

2 DIRECTORY OF A SPECIFIED FILE

To find out whether one particular file resides on a disk, the complete name of that file is entered one space after the resident command specification "DIR". For example, the entry of the following command line will check the disk in default drive A for the file named "THISFILE.DOC":

A>**DIR THISFILE.DOC** ***RETURN***

The presence of a specific file on a disk in a non-default drive can be determined by entering the appropriate drive name and a colon immediately before the name of the specified file.

3 DIRECTORY OF A GROUP OF FILES

To inquire about several files that belong to a group with similar names, the you can enter an ambiguous file name (a file name with wildcard characters "*" and "?"). For instance, to check the default drive disk for all of the files with the extension "BAK", you would enter the following command line:

A>**DIR *.BAK** ***RETURN***

The command line:

A>**DIR PROGRAM?.HEX** ***RETURN***

will check the disk for files such as "PROGRAM1.HEX", "PROGRAM2.HEX", and "PROGRAM3.HEX". In addition, the command:

A>**DIR S*.COM** ***RETURN***

will check the disk for files such as "SC.COM", "STAT.COM", and "SYSGEN.COM".

4 CONSOLE DISPLAY OF FILE DIRECTORY

The entry of a DIR command line will produce a console display showing up to four file names in a horizontal line, with the name of the logged drive preceding each line. The following example shows a DIRectory display of all of the files on the disk in drive A:

```
A: MOVCPM5 COM : LIST     COM : PIP      COM : SUBMIT   COM
A: STAT    COM : XSUB     COM : ED       COM : ASM      COM
A: DDT     COM : LOAD     COM : CONFIGUR COM : SYSGEN   COM
A: DUMP    COM : DUMP     ASM : DUP      COM : FORMAT   COM
A: CPM48   COM : BATCH    SUB : DUMP     PRN : DUMP     HEX
```

If a specified file or group of files is not found on the disk being investigated, the console will display the message:

```
NO FILE
```

This message will also appear if a DIR command line references files on an empty disk.

5 FILES NOT ACCESSIBLE BY DIR

DIR commands will not produce a display indicating the presence of files that maintain the "SYS" status. (See STAT 2.3.)

Files assigned to an unlogged user area are also inaccessible to a DIR command unless a USER command is issued before the DIR command. (See USER.)

DUMP

The Utility that Displays a File in Hexadecimal Form

The DUMP utility is invoked with the name of a file (1) to display the hexadecimal contents of each address in file (2). The file contents appear in lines containing 16 bytes of data each. To the left of each line is the address of the first byte in each line. This display will continue to the end of the program, unless suspended or aborted (3).

1 DUMP INVOCATION

The hexadecimal contents of a file can be displayed on the console by responding to the system prompt with the command:

A>**DUMP {file name}** ***RETURN***

Where **file name** is the complete file name of the disk file that you wish to examine in a hexadecimal display.

2 EXAMPLE DUMP DISPLAY

The command **DUMP THISFILE.HEX *RETURN*** might produce this display:

```
0000 C3 C0 01 20 43 4F 50 59 52 49 47 48 54 20 28 43
0010 29 20 31 39 37 39 2C 20 44 49 47 49 54 41 4C 20
0020 52 45 53 45 41 52 43 48 20 44 49 53 4B 20 4F 52
0030 20 44 49 52 45 43 54 4F 52 59 20 46 55 4C 4C 24
0040 46 49 4C 45 20 45 58 49 53 54 53 2C 20 45 52 41
0050 53 45 20 49 54 24 4E 45 57 20 46 49 4C 45 24 2A
0060 2A 20 46 49 4C 45 20 49 53 20 52 45 41 44 2F 4F
etc. . .
```

3 SUSPENDING LONG SCROLLING DISPLAYS

The displays produced by the DUMP utility often scroll by quickly on the console. However, they can be halted temporarily by entering a ***CTRL-S*** at any time during the scroll. The display can then be resumed by entering any keyboard character except CTRL-C (which executes a warm boot and aborts the program). A run of the DUMP utility can be aborted altogether by entering any keyboard character (other than CTRL-S) during the display.

4 DUMP ERROR MESSAGE

```
NO INPUT FILE PRESENT ON DISK
```

EXPLANATION: The file specified to be dumped does not exist on the specified disk. Command should be re-entered specifying the proper file name or drive name, or inserting the proper disk.

DUP

The Utility for Copying and/or Verifying Entire Disks

The DUP utility can be used to duplicate **all** of the data from one disk to another disk. It can also compare the two disks to verify whether the data recorded on one disk correspond exactly to the data recorded on another disk. If desired, DUP will even perform both operations consecutively, to ensure accurate duplication of a disk.

NOTE: Both of the disks involved in a DUP operation must be prepared in the exact same fashion by the FORMAT utility. Thus the density and number of sides used for data storage on each disk must be identical. Furthermore, you can not duplicate disks that were initialized with a different operating system, such as HDOS or MS™-DOS.

You can use the DUP utility through either of two methods: the DUP Prompt Method or the System Prompt Method.

1 DUP PROMPT METHOD

With this DUP method, you invoke the DUP utility from a disk by entering a command at the system prompt. Then you answer a series of DUP prompts to define the duplication operation.

1.1 DUP Invocation

Invoke DUP by typing a command at the system prompt in the following form:

A>**DUP** ***RETURN***

The DUP utility will present the following display when invoked:

```
Disk Utility Program
Version 2.04

Do you want to:

     A  copy and verify
     B  copy only
     C  verify only

     Z  exit to operating system

Selection:
```

This display includes a menu listing the four operations DUP offers. You can begin execution of an operation by typing the letter listed to the left of that operation. Each operation is explained in the following sections.

1.2 A copy and verify

The "copy and verify" operation makes an exact duplicate of a disk, and then verifies that the operation was performed flawlessly, by comparing the two disks.

To begin this operation, type **A** at the "Selection:" prompt. (No carriage return is necessary.)

DUP will ask, in a series of consecutive prompts, for the letter of the drive that contains the disk to be copied **from** ("Source unit:") and for the letter of the drive that contains the disk to be copied **to** ("Destination unit:"). Enter the appropriate drive letter for each prompt. (No carriage return is necessary.)

When you have specified both a source and destination, DUP will instruct you to put the appropriate disks in the specified drives (even if you have already done so). Then the screen display will look something like the following:

```
Source unit:C
Destination unit:D
Put source disk in drive C.
Put destination disk in drive D.

Press RETURN to begin:
```

The last three lines in this display give you a final opportunity to make certain that you have specified the proper drives and inserted the proper disks. (It is important to make certain of these factors, lest you accidentally duplicate the contents of a blank disk over to a disk that has useful data.)

When both disks are positioned in the proper drives, enter a carriage return to start the copying process. The lights on the specified drives will glow alternately to signify DUP activity. (The duration of the copy and verify operation varies depending on the density and number of sides used on the disks.) When finished copying, DUP will display the message:

```
Copy finished.
```

The verification process will begin automatically. DUP will compare the source and destination disks to verify the accuracy of the copy. Then DUP will display the message:

```
Verification finished.
```

and redisplay the DUP selection menu.

1.3 B copy only

The "copy only" operation makes an exact duplicate of a disk.

To begin this operation, type **B** at the "Selection:" prompt. (No carriage return is necessary.)

DUP will ask, in a series of consecutive prompts, for the letter of the drive that contains the disk to be copied **from** ("Source unit:") and for the letter of the drive that contains the disk to be copied **to** ("Destination unit:"). The screen display for this operation may appear as:

```
Source unit:C
Destination unit:D

Put source disk in drive C.
Put destination disk in disk D.

Press RETURN to begin:
```

The last three lines in this display give you a final opportunity to make certain that you have specified the proper drives and inserted the proper disks. (It is important to make certain of these factors, lest you accidentally duplicate the contents of a blank disk over to a disk that has useful data.)

When both disks are positioned in the proper drives, enter a carriage return to start the copying process. The lights on the specified drives will glow alternately to signify DUP activity. (The duration of the copy operation varies depending on the density and number of sides used on the disks.) When finished, DUP will display:

```
Copy finished.
```

Then DUP will redisplay the selection menu.

1.4 C verify only

The "verify only" operation helps you determine whether two disks are identical in media and data contents.

To begin this operation, type **C** at the "Selection" prompt. (No carriage return is necessary.)

DUP will ask, in two consecutive prompts, for the letter of the drives that contain the disks to be compared. (These prompts will ask you to specify "Source unit:" and "Destination unit:", although "source" and "destination" are not pertinent to this operation.)

Answer each prompt with the name of one of the drives containing a disk to be compared. The screen display for this operation might appear as follows:

```
Source unit:C
Destination unit:D
Put source disk in drive C.
Put destination disk in drive D.

Press RETURN to begin:
```

Enter a carriage return to start the verification process. The lights on the specified drives will glow alternately to signify DUP activity. (The duration of the verification operation varies depending on the density and number of sides used by the disks.)

If DUP finishes comparing the two disks and finds that they are absolutely identical, DUP will display the message:

```
Verification finished
```

Then DUP will redisplay the selection menu.

If the two disks do not correspond exactly, in data content or data position, then DUP displays the following message:

```
Verification error
```

Then DUP will redisplay the selection menu.

NOTE: Conceivably, two disks could contain the exact same data, but in different positions on the disk surface. DUP verification would regard such disks as **different**.

1.5 Z exit to operating system

When the DUP selection menu is displayed, you can end the program and return to the operating system by typing the Z alternative at the "Selection:" prompt. Such an entry will produce the prompt:

```
Place a bootable disk in drive A and type any character:
```

At this prompt, you should insert the disk you used to perform bootstrap into drive A (if you have removed it since bootstrap) and enter a carriage return.

1.6 Invalid Entries During a DUP Prompt Operation

If you answer a DUP prompt with an invalid character, either the prompt or the menu will be repeated in most cases. However, if you specify a drive that does not exist in the hardware environment (when responding to the "Source unit:" prompt or the "Destination unit:" prompt) then the terminal may "hang up", freezing the keyboard. In such a case, you must reset the computer and perform bootstrap to proceed with any CP/M operation.

2 SYSTEM PROMPT METHOD

The System Prompt Method enables you to include all of the specifications necessary for the DUP operation in a single command line entered at the CP/M system prompt.

2.1 Command Line Entry

System Prompt Method DUP commands are entered in the following form:

A>**DUP {destination}:={source}:{[option]}** ***RETURN***

Where **DUP** is the command line function;

where {**destination**} is the letter of the drive containing the blank disk that you wish to receive the copied data;

where {**source**} is the letter of the drive containing the data disk that you wish to duplicate; and

where {**[option]**} represents letters enclosed in square brackets [] and separated by a comma , to specify how the DUP operation should be conducted. One or two of the letters **C, V,** or **N** can be used, although none of these options are mandatory.

NOTE: In a command line equation, the data source is always on the right and the data destination is always on the left.

2.2 DUP Options

Through the System Prompt Method, you can enter the following options to perform a DUP operation as indicated. These option letters should always be enclosed in square brackets. If two option letters are used, they should be separated by a comma.

C Copy only: DUP will copy all of the data from the source disk to the destination disk, without comparing them to verify the accuracy of the copy.

V Verify only: DUP will compare the source and destination disks to verify that they contain the exact same data in the exact same locations on the disk surface. When this option is specified, either disk can be the "source" or "destination".

N No inquiry prompt: DUP will perform the operation you have specified without displaying a prompt to confirm whether the disks are in the appropriate drives.

Without the C or V Options Copy and verify: DUP will copy all of the data from the source disk to the destination disk, and then automatically compare the two disks to verify that the copying was accurate. This operation will be performed by default if you enter a DUP command with source drive and destination drive specifications but without specifying a C or V option.

2.3 DUP Defaults

If you enter a System Prompt Method DUP command line (specifying destination and source drives) and abstain from specifying certain options, the following default conditions will be in effect during the DUP operation:

- Copy and Verify operation will be performed. Hence, the entire contents of the source disk will be copied to the destination disk and then the two disks will be compared to verify whether they are exact duplicates. Occurs if neither the C nor the V options are specified.

- Prompts are displayed to encourage you to insert the source and destination disks in the specified drives, and then to trigger the start of the DUP operation with a carriage return. These prompts appear as shown:

```
Disk Utility Program
Version 2.04


Put source disk in drive A.
Put destination disk in drive B.

Press RETURN to begin:
```

Prompts in this form will appear if the N option is not specified.

2.4 System Prompt Examples

A>**DUP B:=A:** ***RETURN***

DUP will prompt you to insert the proper disks into drives B and A, then copy all of the data from drive A to drive B, and then verify whether the copy was performed accurately.

A>**B:DUP C:=D:[C,N]** ***RETURN***

The DUP utility, in this case, is stored on the disk in non-default drive B. It will copy all of the data from the disk in drive D to the disk in drive C. As specified in the options, DUP will **not** prompt you to insert the disks in the appropriate drive and DUP will **not** verify the accuracy of the copy.

A>**DUP D:=E:[V,C]** ***RETURN***

If your command line options are contradictory, (both C and V use in same line) DUP will acknowledge only the last one. Hence, in this case, DUP will prompt you to insert the proper disks into drives D and E, then **copy** all of the data from drive E to drive D, DUP will **not** verify whether the copy was performed accurately.

A>**dup c:=e:[v,n]** ***RETURN***

DUP will verify that the data stored on the disks in drives C and E is exactly the same and arranged in the same locations on the surfaces of these disks. As specified in the options, DUP will **not** copy any data and it will **not** prompt you to insert your disks. Notice that the letters in a DUP command line do not have to be upper case (capitalized).

NOTE: Conceivably, two disks could contain the exact same data, but in different positions on the disk surface. A DUP verification operation would regard such disks as **different**, and display a "Verification Error" message.

3 DUP ERROR MESSAGES

```
Media incompatible on diskettes.
```

EXPLANATION: Disks used for duplication must be identical in size, density, number of sides, and tracks per inch.

```
Drives incompatible for copy operation.
```

EXPLANATION: DUP operations can only be performed between two drives that write the same type of disk media. If you have Z-37 drives, make certain that the drives you have selected as your "Source unit" and "Destination unit" have been set to write data at the same step rate and number of tracks per inch (tpi). Specify drives that write to identical disk types for your "Source unit" and "Destination unit".

```
Drive not available in current configuration.
```

EXPLANATION: Drives specified as source or destination must be drives that are connected in the hardware environment, turned on and recognized by the operating system. Specify such drives.

```
Hard read error on source disk. Copy/Verify aborted.
```

EXPLANATION: DUP failed in an attempt to read data from a source disk. Try the operation again. If DUP failures persist, use the PIP utility to copy files from the source disk, and the SYSGEN utility to copy the operating system from the source disk.

```
Hard read error on destination disk. Copy/Verify aborted.
```

EXPLANATION: DUP failed in an attempt to write data to the destination disk. Use the FORMAT utility to prepare the destination disk before using DUP. If DUP failures persist, use a different disk as the data destination. Make certain that the destination disk media is the same as the source disk media.

```
Verification error.
```

EXPLANATION: DUP's comparison of two disks found them to be different. Determine which disk contains the desired data, call this disk the "source disk." Then use the FORMAT utility to erase and prepare the inferior disk, and perform DUP's "copy and verify" operation. If the second verification of these two disks produces this error, use a new disk of the same media type for the destination and repeat the "copy and verify" operation.

```
Source and Destination cannot be the same drive.
```

EXPLANATION: Different drive units must be specified as source and destination unit.

```
Unable to copy to this disk. It is write protected.
```

EXPLANATION: Disk should be write enabled by removing the adhesive tab from the write-enable notch in the disk jacket.

```
Command line syntax error
```

EXPLANATION: System Prompt Method command line was entered incorrectly. Re-enter using the entry form explained in "2.1 Command Line Entry."

```
Unknown command line option
```

EXPLANATION: System Prompt Method command line was entered with invalid characters used as options. Re-enter using only the options listed in "2.2 DUP Options."

```
Invalid Source Unit
```

EXPLANATION: The entry made at the "Source unit:" prompt did not correspond to a valid disk drive within the hardware environment. The DUP operation must be reselected, and a valid drive name letter entered.

```
Invalid Destination Unit
```

EXPLANATION: The entry made at the "Destination unit:" prompt did not correspond to a valid disk drive within the hardware environment. The DUP operation must be reselected, and a valid drive name letter entered.

ED

The Line Editing Utility that Creates and Edits Text Files

The ED utility enables you to compose, alter, and manipulate files containing ASCII characters. The files composed by ED are often referenced or manipulated by other system utilities and commands.

When invoking ED, you either create a new file or summon an old file into the computer (1). ED works on files using an area of computer memory known as the memory buffer (2). Files being edited can be moved, altered, or displayed when you enter various ED commands (3).

1 ED INVOCATION

The ED utility is invoked by the entry of a command line in the following form:

A>**ED {file name}** ***RETURN***

Where {**file name**} is the complete name of a file that you wish to compose or edit. you must specify the name of the text file here.

If the file resides in a non-default drive, this drive should be specified immediately before the file name in the command line. If the file does not yet exist at all, ED will create it on the disk in the default drive, or on a disk in a specified drive.

The following entry, for example, would cause ED to open the file named "THISFILE.TXT", which resides on the disk in non-default drive B:

A>**ED B:THISFILE.TXT** ***RETURN***

When ED "opens" a file, it checks the disk directory for the name of the file. Then ED reserves the computer's Transient Program Area (TPA) as a memory buffer to be used for text editing and file manipulation.

If the name of the file is not in the disk directory, ED creates the empty file and displays both the "NEW FILE" message and the ": *" prompt. Then you can begin inserting text into the empty file by using the "I" command at the prompt.

If the name of the file does exist in the disk directory, then ED locates it, and displays the ": *" prompt on the console. You can then move the existing file into the memory buffer by using the "A" command at the prompt.

2 ED STRUCTURE AND FEATURES

Text sections 2.1, 2.2, and 2.3 explain the structure and features of the ED utility. It is recommended that you understand these concepts before trying to implement ED commands.

2.1 Text Files in the CP/M Environment

To be properly read from, written to, and transferred, CP/M text files must be composed entirely of American Standard Code for Information Interchange (ASCII) characters. Text files must end with the entry of the CTRL-Z "end-of-file" character. Text can be moved to and from the disk in units of "lines". A line is defined as a string of ASCII characters that ends with the carriage return and line feed characters. (The "carriage return and line feed" combination can be entered by pressing the RETURN key.)

2.2 The Memory Buffer

The memory buffer is an area in the computer's memory that the ED utility uses as a "scratch pad" on which to compose and edit text before it is transferred to a disk for storage.

You can send a specified number of lines of text to the buffer from the disk or send text characters into the buffer by entering them directly through the keyboard. The memory buffer in some computers can hold about 36,000 text characters at one time. When full, you can purge the buffer of text by writing a specified number of text lines from the full buffer to the disk.

2.3 The Character Pointer

Text in the buffer is usually arranged on numbered lines. To help you to access locations within the text, the buffer contains an invisible character pointer, which can be moved to specific locations within the text by various user commands.

The character pointer resides before the first character in the text, after the last character, or between any two text characters. When it is moved to a position in the text, a specified number of characters or lines can be inserted or deleted starting at the pointer's current position. Text is inserted through the use of special commands, sometimes followed by direct keyboard typing.

The character pointer is positioned by user commands that move it up or down to different lines, left or right along a line, or to the top or bottom of the file. The character pointer can also search through the text to locate a user-specified text string.

All ED commands are executed starting at the current position of the character pointer.

3 ED COMMANDS

Commands used within the CP/M EDitor are entered in response to the ":*" prompt, or to a prompt in the form "n:*" where "n" is the number of the text line upon which the character pointer resides. All ED commands (except those which end an ED session) can be entered in a series on the same command line. The last command entered on any line must be followed by a carriage return to initiate command processing.

The ED utility uses four commands (A, W, X, and R) to transfer text lines between the disk and the memory buffer (3.1).

When a file is in the memory buffer, four commands (B, C, L, and n) can be used to move the character pointer to a specified line or character (3.2).

Once the character pointer is in position, four commands (I, D, and K) are used to insert text into or delete text from the buffer (3.3).

Whenever text is within the memory buffer, two commands (T and P) can be used to display it to the screen. Three other commands (Z, V, and U) are used to alter characteristics of screen display (3.4).

When a file is in the memory buffer, four commands (F, S, N, and J) can be used to move the character pointer to an occurrence of a specified text string (3.5).

To remove an entire file from the memory buffer, four commands (E, Q, H, and O) can be used to send it from the buffer to the disk, or to dispose of it in some manner. These commands must be entered alone and followed immediately by a carriage return (3.6).

One command (M) is used to trigger multiple execution of other ED commands (3.7).

3.1 Moving Text to or from Memory Buffer

nA **Append lines from disk to buffer**

This command will copy "n" lines of text from the disk file specified in the invocation command to the memory buffer, where the text image can be edited.

The nA command must be implemented when you wish to edit text from an existing file. This command is entered in response to the ":*" prompt, and causes the "1:*" prompt to be displayed. The counterpart of the nA command, the nW command, transfers edited text lines back to the disk.

If you do not specify the number (n) of lines to be appended, ED will append one line from the disk. If the "#" character is entered in place of "n" (#A), then all of the text lines in the disk file will be copied to the memory buffer.

nW **Write lines from buffer to disk**

This command transfers "n" lines of edited text from the memory buffer to the disk. Text that is written to the disk in this manner will no longer exist in the memory buffer.

The W command is implemented when the memory buffer becomes full during text editing. It is entered in response to the ":*" or "n:*" prompt, and causes the ":*" prompt to be displayed.

The W command always starts at the top of the buffer, transferring the first line in the buffer through the "nth" line in the buffer. The buffer text line that occurs after the "nth" line then becomes the first line, moving up to the top of the buffer.

If you do not specify the number of lines to be transferred to the disk, one line will be transferred. If the "#" character is entered in place of "n" (#W), then all of the text lines in the memory buffer will be transferred to the disk, and the buffer will be empty.

As the edited text is written back to the disk, a few lines at a time, it accumulates in a temporary file that has a "$$$" extension.

nX **eXtradite text block from buffer to temporary library file**

This command transfers a block of text from the memory buffer to a temporary disk file, so that it can be transferred back to the buffer (by the R command) at a desired location.

The block of text begins with the line containing the character pointer, and ends "n" lines later. The block is stored on the disk in a temporary file which is automatically named $$$.LIB (the standard name for a temporary library file). After this file is created on the disk, you should implement other ED commands to move the character pointer to the text location at which the temporary file should be inserted.

When the character pointer is at the desired location, the R command is used to transfer the $$$.LIB file back to the buffer at the desired location. The letter "R" and a carriage return should be entered to insert the text at the current location. The same text can then be inserted at another text location by moving the character pointer and, again, entering "R" and a carriage return.

Before a different block of text is transferred to the $$$.LIB file, the "0X" (a zero followed by an X) command should be entered to clear the old text from the file.

If no number is specified in the space preceding the "X", then one line of text (starting from the current character pointer location) is transferred to the temporary file. If the "#" symbol is specified, then all of the text lines within the buffer which follow the character pointer are transferred.

Rf **Read library file f from disk to buffer**

This command copies the text from a disk library file to the memory buffer, inserting this text at the current location of the character pointer.

The file being read into the buffer should usually be specified in place of the "f" in the command "Rf". However, these files are always assumed to have the "LIB" extension. Therefore, you need only enter the primary file name in such a command. For example, the text from library file "ROUTINEX.LIB" could be read into the buffer with the entry of the following command:

RROUTINEX

If you wish to read a temporary library file into the buffer text, then no part of the file name need be entered with the R command. The standard temporary library file name "$$$.LIB" will be assumed.

3.2 Positioning the Character Pointer

+/− **B Beginning/Bottom of text character pointer movement**
This command will move the character pointer to the beginning of the first line of the text in the buffer (if entered in the form **B**), or to the end of the last line in the buffer (if entered in the form **−B**).

+/−nL **Line down/up character pointer movement**
This command will move the character pointer from its current line within the memory buffer text to another line a specified number (**n**) of lines away.

When the command is entered in the form **nL**, the character pointer will move ahead (down) the specified number of lines to the beginning of a text line.

When the command is entered in the form **−nL**, the character pointer will move backward (up) the specified number of lines to the beginning of a text line.

If the number of lines to move is not specified when the **L** command is entered, the pointer will move one line ahead (down).

In order to move the pointer to the beginning of the line upon which it currently resides, enter the "0L" command (with a zero preceding the "L").

+/−n **Advance to a line and display it**

This command moves the character pointer a specified number of lines (n) and displays the text of the line on which it lands. This command produces the same results as the simultaneous entry of both the "L" and "T" commands. If no number is entered before the carriage return, ED assumes the number one.

+/−nC **Character pointer movement to right/left**

This command moves the character pointer a specified number (n) of character spaces, usually toward the right or left edge of a text line. (When a command-driven character pointer reaches the edge of a text line, the carriage return and line feed characters will cause it to change its direction momentarily.)

When entered with a plus sign ("+"), this command will move the pointer the specified number of spaces to the right, and/or down to successive lines. When entered with a minus sign ("−"), this command will move the pointer the specified number of spaces to the left, and/or up to preceding lines. If you wish to move the character pointer past the edge of a text line using the "C" command, the number specified in the command will have to include two character spaces to get past the carriage return and line feed characters at the end of the line.

3.3 Inserting or Deleting Text

I **Insert characters from keyboard to buffer**

The "I" command enables you to insert characters directly into the text at the current position of the character pointer. This command is entered in response to the ": *" or "n: *" prompt.

If you enter a carriage return immediately after the "I" command, then the "n:" prompt appears on the next line and text characters can be inserted on successive lines until you enter a CTRL-Z "end-of-file" character. The entry of a CTRL-Z causes the ": *" prompt to be displayed. After such an insertion operation, the character pointer will be positioned at the end of the last inserted text line.

If text is inserted on the same line as the "I" command, then the insertion operation will end when the next carriage return is entered, and a "*" command prompt will appear at the left edge of the screen. After such an insertion operation, the character pointer will be positioned at the beginning of the line following the line of inserted text.

If upper and lower case insertion text is desired, enter the "I" command with a lower case "i". Entering the command with an upper case "I" will automatically translate all inserted text to upper case.

+/−nD Delete characters from buffer text

This command will delete a specified number (**n**) of characters from memory buffer text, starting at the location of the character pointer.

Deletions will take effect to the right of the pointer if the specified number of deletions is preceded by a "−" sign. If the specified number of deletions is preceded by a "+" sign or by nothing, then characters to the right of the pointer will be deleted.

If no number of characters is specified for the deletion operation, then one character will be deleted. If "#" is specified as the number of characters to be deleted, then all text characters before or after the pointer (depending on the sign preceding the number) will be deleted.

The carriage return and line feed characters at the end of each text line are counted as two separate characters, even though they are produced by pressing only the RETURN key.

+/−nK Kill lines from buffer text

This command will delete a specified number (**n**) of lines from the memory buffer text, starting at the position of the character pointer.

If a "+" (plus sign) precedes the number of lines to be killed, then that number of lines occurring **after** the character pointer will be killed. If a "−" (minus sign) precedes the number, then that number of lines occurring **before** the pointer will be killed.

If the character pointer is positioned in the middle of a line during a "K" command, the portion of the text line to the left or right of the pointer will be deleted as if it were one entire line.

If no number (**n**) is specified for the deletion operation, then **one** line will be deleted. If "#" is specified as the number of lines to be deleted, then **all** text lines before or after the pointer (depending on the sign preceding the number) will be deleted.

3.4 Displaying Text to Console

+/−nT Type text lines on console

This command will cause a console display of a specified number (n) of text lines, starting at the position of the character pointer.

If the character pointer is in the middle of a line, the portion of the line between the pointer and the end of the line will be counted as an entire line. If such a command begins with a minus sign ("-"), then the specified number of lines before the line containing the character pointer and the line containing the pointer are displayed. If the pointer is positioned in the middle of a line and a zero is specified in the command, then only the portion of the line from its beginning to the pointer will be displayed.

If no number (**n**) of lines is specified, then one line will be displayed. If the "#" symbol is specified, then all of the lines in one direction will be displayed.

You can interrupt a console display which scrolls too quickly by entering a ***CTRL-S*** character. The scroll will resume when another CTRL-S is entered. You can abort a long scrolling screen display by entering any other keyboard character while the display is in progress.

The "T" command will not effect the position of the character pointer. Therefore, at the end of a "T" operation, the pointer marks the position at which the operation began. This position will be indicated by the number in the "n:*" prompt.

The "T" command can be entered in response to a "n:*" or ":*" prompt.

nP **Page display on console**

This command causes text to be displayed on the video screen in units of one page (23 lines), and deposits the character pointer at the end of the display.

If a display of the first page (23 lines) beginning at the character pointer is desired, then a zero should precede the **P** in such a command. Hence, a **0P** command has the same effect as a **23T** command.

If the number one or no number is specified before the "P", then one page (23 lines) of text, starting 23 lines past the character pointer, will be displayed. The command **2P** will cause the display of two pages starting 23 lines past the pointer. The command **3P** will cause the display of three pages starting 23 lines past the character pointer, and so on. Hence, a **2P** command has the same effect as a **23L46T-23L**.

nZ **Zone interruption of text display scroll**

This command can be entered into a command line in front of the "T" or "P" display commands to interrupt a long console display scroll at time-regulated intervals so you can view the text one zone at a time.

When "T" and/or "P" commands are entered in a series within the same command line, "Z" commands can be placed in between to interrupt their execution for time periods determined by the number preceding the "Z".

The number (n) preceding the **Z** in the command stands for the number of half seconds that the display scroll will be interrupted. Hence, if a command line contains a **10Z** between two display commands, the scrolling caused by the display commands will be interrupted for five seconds.

+/−V **aVert or replace line numbers in console displays**
If you prefer not to use the line numbers, the **−V** command will eliminate them from the console display.

The command **V** will restore line numbers to the console display.

A special form of this command, in which a zero precedes the V (**0V**), will produce a display showing how many locations remain unused in the memory buffer (r), and the total number of buffer locations that are accessible through the ED utility (t). The display appears in the form: "r/t".

+/−U **Uppercase/lowercase text translation**
If you would like all characters entered into text through the ED utility to be put into uppercase, the "U" command can be entered.

The "−U" command can be entered to allow the inserted text to be displayed in both lower and uppercase.

3.5 Searching for Text Strings

nFt **Find text string t within buffer text**
This command is used to locate the specified number of occurrences (n) of a particular string of characters (t) within the text.

The string of characters (**t**) being sought is specified immediately after the **F** in the command,and ended with the entry of a ***CTRL-Z*** character and a carriage return.

If you do not specify the number of occurrences (n) of the string to be found within the text, then only the first occurrence will be found.

If you desire to locate a string of characters (t) that contains the carriage return and line feed characters, these two characters can be specified in the command line with the ***CTRL-L*** character.

The string specified in the command (t) must match the actual text in spelling, spacing, capitalization, etc.

The specification of a string of characters that is not found in the text will produce the error message:

```
BREAK "#" AT
```

sending the character pointer back to its position before the search operation failed.

nSdt **Search and replace text string**

This command performs the operations of the "F", "D", and "I" commands simultaneously, by finding a specified string (d) within the buffer text, deleting it, and inserting a second specified string (t) at the same location.

The number (n) in the command represents the number of text string substitutions desired by you throughout the text. The omission of this number will cause a substitution to be made only at the first occurrence of the sought-after string (d). The entry of the "#" symbol in place of this number will cause the substitution to be made at every occurrence of the sought-after string throughout the text.

The string of characters to be found and deleted in the buffer text (d) is specified immediately after the "S" in the command, and ended by a CTRL-Z character. Immediately after this CTRL-Z, the text string to be inserted (t) is entered and ended by a second CTRL-Z and a carriage return.

The specification of a string of characters that is not found in the text will produce the error message:

```
BREAK "#" AT
```

sending the character pointer back to its position before the failed search and replace operation.

nNt **fiNd text string on disk**

This command performs the same search operation as the "F" command except that it can search an entire file for a text string (t).

If the specified string (t) is not found in the memory buffer, then this command will automatically write the contents of the buffer to the disk (into a temporary file, as the "W" command does) and append an image of another portion of the disk file's text to the buffer (as the "A" command does) until the entire file has been searched for the string the specified number of times (n).

nJftd **Juxtaposition substitution and deletion**

This command finds a first string (f), inserts a second string (t) after the first, and then deletes all of the text between the end of the inserted string (t) and the beginning of the third string (d).

The "J" in the command is immediately followed by the first text string (f), a CTRL-Z, the text string to be inserted (t), a CTRL-Z, the third text string (d), a CTRL-Z, and a carriage return.

The third string (d) serves as a restraining boundary for the text deletion.

This multi-faceted operation is performed a specified number of times (n), or once, if no number is specified. If the "#" is specified, the operation will be performed for all occurrences of the first text string (f).

If the third command line string cannot be found in the buffer text, then no text is deleted.

3.6 Closing a Text File

E **End session while buffer text becomes permanent disk file**
All text in the memory buffer is copied to the disk, where it is combined with any text that has accumulated in a temporary file, and assigned the original file name.

At the same time, the version of the file that was copied to perform the edit is assigned the extension ".BAK" in place of its original extension.

The operating system then regains control and displays the system prompt.

Q **Quit session by deleting edited copy of file**
All text in the memory buffer and/or any temporary file created during the session is deleted, and any existing versions of the file on the disk maintain the status and names they held prior to the editing session.

If the file being edited existed before the session, then the original version remains intact, as if the editing session never took place.

If the file in the buffer is an original composition (was created as a "NEW FILE" during this editing session) then all copies or versions of this file will be destroyed.

Since the accidental use of this command could delete important text composed or edited during the session, its entry will produce the "Q- (Y/N)?" confirmation message. The **Y** character must be entered before the deletion will be executed. If the **N** is entered, the current editing session will continue.

H **Halt session temporarily to save alterations**
All alterations made to text (or any text composed) will be saved under the active file name, and the editing session will continue with an image of the currently edited file automatically appended to the memory buffer. This command has the same effect as entering a combination of both the "E" and "A" commands.

O **Omit recent alterations and restart edit session**
Any text in the memory buffer or in a temporary disk file is deleted, and the editing session continues, using the same text with which it began.

In effect, this command nullifies any text alterations or composition performed in the most recent ED session and starts the session over, as if the "Q" and "A" commands had been entered consecutively.

3.7 Causing Multiple Command Execution

nM **Multiple command execution**
This command allows you to execute one or more commands a specified number of times (n) without additional command entries. Commands are entered on the same line, following the **nM** command in a string terminated by either a carriage return or a CTRL-Z.

All commands following the **nM** will be executed the number of times specified at the beginning of the entire command line (n), or until an error condition is encountered. If no number (n) is specified, then the operations invoked by the command line will be implemented from the position of the character pointer through the end of the text, or until an error condition is encountered.

This command is commonly used with the search and replace command ("nS"), to facilitate text string substitution throughout a large text area. When such a search reaches the end of the text in the memory buffer, an error condition to indicate that the substitution can no longer be executed.

Multiple commands are executed from the position of the character pointer toward the end of the text. Hence, the pointer should be positioned at the beginning of the buffer text if multiple commands are to be executed throughout the text.

4 THE FILE EDITING CYCLE

The following sequence of diagrams shows the file named "REPORT.DOC" as it undergoes ED's file editing cycle. The left side of the diagrams represents the memory buffer, and the right side of the diagrams represents the logged disk.

In Figure 2-1, you have opened the file "REPORT.DOC" by entering **ED REPORT.DOC** at the "A>" system prompt. You then enter ED's I command, and begin typing text into the memory buffer file. No text has yet been recorded on the disk.

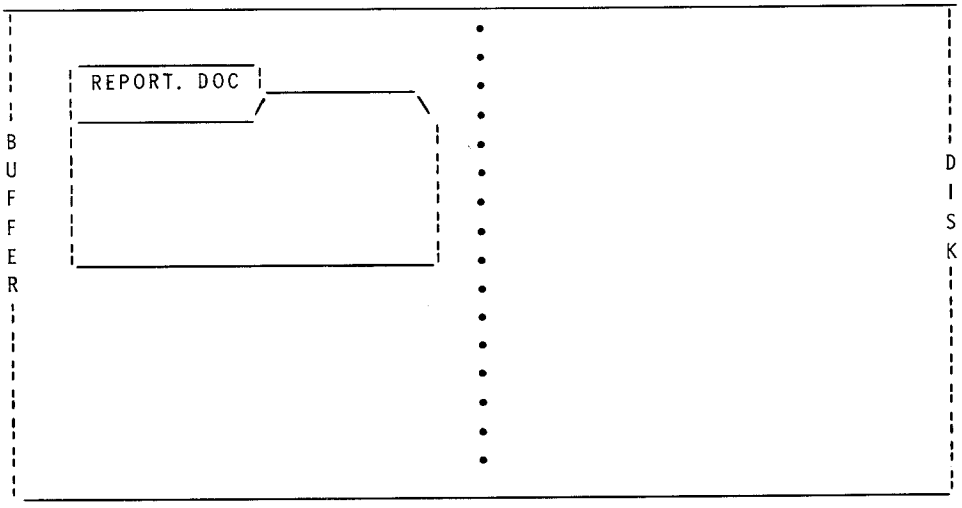

Figure 2-1

In Figure 2-2, you have inserted text into the file, and want to save some number of text lines. First you end the insert by entering a CTRL-Z. Then to save the text, you enter ED's **W** command to send a specified number of text lines from the buffer to a temporary file on the disk. ED gives this temporary file the name "REPORT.$$$". The buffer file remains.

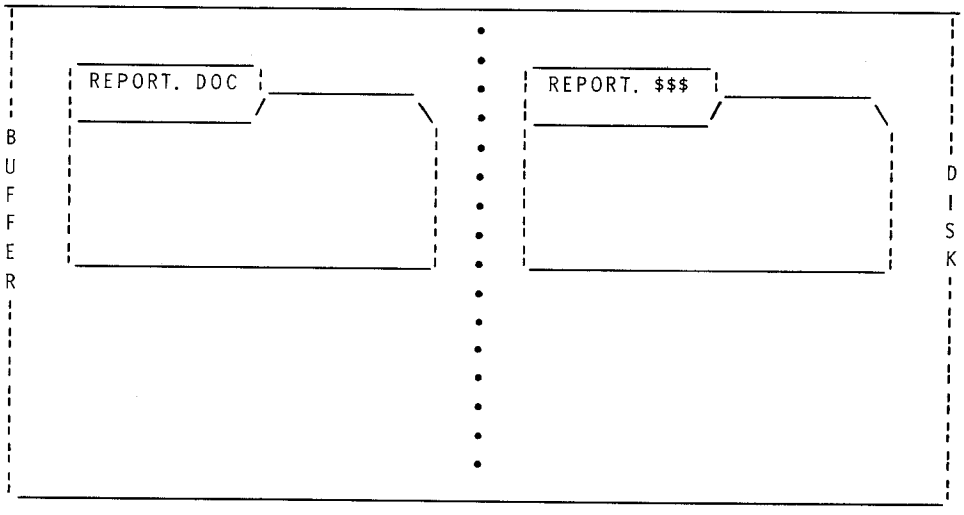

Figure 2-2

In Figure 2-3 you end the editing session and save all of the text composed for "REPORT.DOC" by entering ED's **E** command. This saved text is recorded on the disk under the file name "REPORT.DOC". Both the buffer file "REPORT.DOC" and the temporary disk file "REPORT.$$$" are erased.

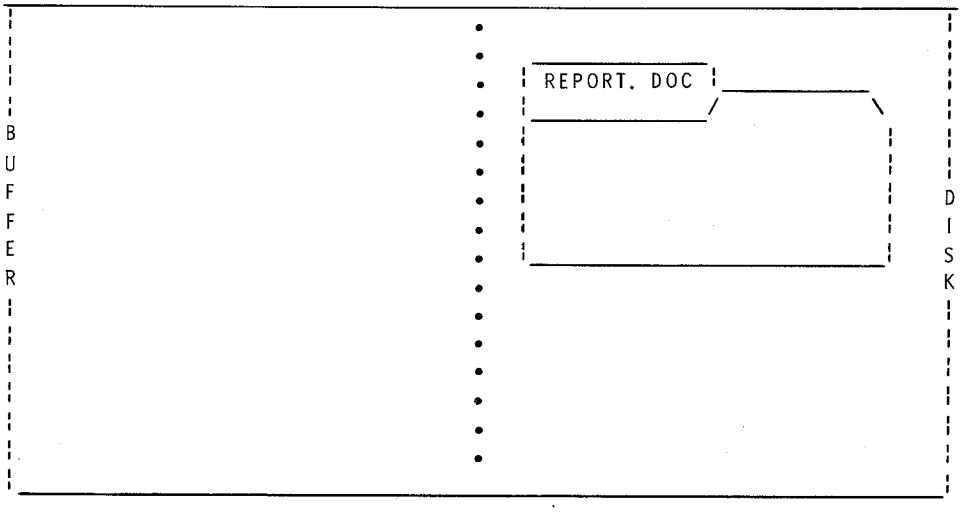

Figure 2-3