CHAPTER 6
THE INTELLEC® MONITOR

The Monitor is a control program that provides supervisory functions for the Intellec microcomputer development systems. It processes the commands you enter at the console device. The set of available commands provide the following facilities:

- Displaying and modifying memory and processor registers.
- Initiating execution of your programs.
- Inserting breakpoints into your programs before execution.
- Reading hexadecimal data from an external device into memory.
- Writing hexadecimal data from memory to an external device.
- Accessing user written I/O routines.
- Invoking resident diagnostics.

Your communication with the Monitor is through the system console. When you enter the Monitor, the sign-on message and a prompt character are displayed on the system console. The prompt character is a period (.) at the left margin of the system console.

Command Entry

You can enter commands at the console anytime after the prompt character is displayed at the left margin.

All Monitor commands are single alphabetic characters. Some commands have optional parameters and some have required parameters. For example, the X command displays the contents of all the registers. But if you only want to see the contents of a single register, you can specify that register in the command. However, the D command, which displays the contents of memory, requires that beginning and ending memory locations be included in the command.

Normally, commands are ended by pressing the return key on the keyboard. There are exceptions to this and will be fully explained in the individual command descriptions.

Thus the general syntax of the Monitor commands is:

    <command>[<parameters>]<CR>]

where:

<command> is the single alphabetic character for the command.

<parameters> are one or more variable data supplied with the command. Parameters can be numeric or alphabetic. When a numeric parameter is called for, it must be entered in hexadecimal form and is limited to four hexadecimal digits (0000H through FFFFH). Larger numbers can be entered but only the four rightmost digits are used by the system. For example, the value 123456H is treated as 3456H by the system.

<CR> is the carriage return key on the keyboard.

Where a comma is shown in the syntax, you can use either a comma or a space unless otherwise noted under the individual commands.

Entry Errors

The Monitor checks for several error conditions:

- Invalid characters
- Address value errors
- Checksum errors

The Monitor checks the validity of each character entered at the Console device. As soon as it encounters an invalid character, it displays a cross-hatch (#) and aborts the command. It displays the prompt character on the next line and waits for more input:

```
.4#
```

4 is rejected because it is not a valid command.

The first character entered must be a valid command, otherwise it is rejected by the Monitor.

When the Monitor is expecting an address in hexadecimal (all addresses are entered in hexadecimal) any character other than 0-9 and A-F is rejected:

```
.D1000,1FFG#
```

G is not a valid hexadecimal digit.

Many commands require two addresses where the first address is lower than the second. If the first address is higher than the second, the operation will be performed on the single address specified as the first address. For example, there is a command that fills memory with a constant value. If you meant to say fill memory from address 900 to address 1000 with FF but entered the addresses in the opposite order:

```
.F1000,900,FF
```

The Monitor would place a FF in address 1000 and do nothing else. No indication that an error occurred is given. You will only find the error when you notice that a single byte was filled instead of 100H bytes.

Addresses are evaluated modulo 65,536. That means that the highest address that will be accepted is FFFF in hexadecimal. If addresses higher than FFFF are entered, only the last four digits will be used and the command will be executed. For example, if in the previous addresssing error example, we had entered 10000 instead of 1000:

```
.F10000,900,FF
```

The command would have been evaluated as:

```
.F0000,900,FF
```

and memory from address 0 through 900 would have been filled with FF. This is not what you wanted, in fact, you probably wiped out data you wanted. As you'll find out later in this chapter, it did wipe out some of the memory that the Monitor itself uses. No indication of this error is given except that the Monitor will not function correctly for some commands without rebooting the system.

When the Monitor detects a checksum error when reading from an input device, such as a paper tape reader, it displays a cross-hatch (#) and destroys all the data from the record in which the error occurred. Subsequent records are not read.

Command Categories

The Monitor commands are divided into seven categories:

- Monitor I/O configuration
- Memory control
- Register control
- Paper tape I/O
- Program execution
- Utility
- Diagnostics (the Z$ command, described in *Intellec Series II Installation Manual*)

Monitor I/O Configuration Commands

The Monitor has four system devices defined:

- Console
- Reader
- Punch
- List

You have the option of selecting the actual peripheral device that will perform the required function. For example, the Console device accepts commands and data and presents error and informational messages. To perform these functions you can assign a teletype console, a CRT console, a paper tape reader (in conjunction with a printer), or some non-standard device (for which you must write the driver program).

There are two Monitor commands with which you can control the I/O configuration:

- Assign (A) to change device assignment.
- Query (Q) to find what devices are currently assigned.

To make I/O device assignments, you must understand the characteristics of the system devices.

The Console is an interactive, character-orientated input and output device. To be used as a Console, a device must have all these characteristics. A teletypewriter and a CRT terminal have all these characteristics. A paper tape punch has all but the input characteristic, therefore it can't be assigned as a Console.

The Reader is a character-oriented input device that transfers data on command and notifies the calling system when no more data is available. A paper tape reader meets these qualifications.

The Punch is a character-oriented output device that accepts a character from the calling system and records it on an external medium. A paper tape punch meets these qualifications.

The List device is a character-oriented output device that accepts a character from the calling program and records it on an external medium in human readable form. A line printer meets these qualifications.

One of four actual devices can be assigned to each of the system devices. The devices for which the Monitor has driver programs are:

- Teletype console with a keyboard, printer, paper tape reader, and punch. This type of device can be assigned to all the system devices.
- CRT devices with a keyboard that are compatible with the Intellec system. This type of device can be assigned to the Console or the List device.
- High speed paper tape reader. This type of device can be assigned to the Reader device.
- High speed paper tape punch. This type of device can be assigned to the Punch device.
- Line printer. This type of device can be assigned to the List device.
- Batch. This is a non-interactive mode in which CONSOLE input is read from the assigned READER device and written to the assigned LIST device. In preparing a command file for BATCH input, the user should enter commands in exactly the same way as if the system were in interactive mode. Each command should end with a carriage return/line feed pair. The period (prompt) character which is generated by the Monitor in interactive mode should not appear as part of the command. Since the Monitor will continue to read from the READER until the CONSOLE is reassigned, the last command in the BATCH command file should reassign the CONSOLE to prevent the Monitor from reading off the end of the tape.

A - Assign Command

You can assign one physical device to a system with the Assign command. The format of the comand is:

```
A<logical-device>=<physical-device>
```

where:

A is the Assign command code.

<logical-device> specifies which of the system devices is to to be assigned a <physical-device>. The possible values for <logical-device> are:

C or CONSOLE

R or READER

P or PUNCH

L or LIST

The equal sign (=) must be entered.

<physical-device> specifies which of the physical devices is to be assigned to the <logical-device>. The possible values of <physical-device> for each <logical-device> are:

CONSOLE	T or TTY (teletype terminal) C or CRT (Intellec II compatible CRT terminal) B or BATCH (batch mode) 1 (user-defined device for which a user-written program is present)

READER	T or TTY (teletype terminal) P or PTR (high speed paper tape reader) 1 or 2 (user-defined devices for which user-written driver programs are present)
PUNCH	T or TTY (teletype terminal) P or PTP (high speed paper tape punch) 1 or 2 (user-defined devices or which user-written driver programs are present)
LIST	T or TTY (teletype terminal) C or CRT (Intellec II compatible CRT terminal) L or LPT (line printer) 1 (user-defined device for which a user-written driver program is present)

Examples of the Assign Command: To assign a high speed paper tape reader as the system Reader device:

```
.AR=P<CR>
```

or

```
.AREADER=PTR<CR>
```

To assign a CRT terminal as the system Console device:

```
.AC=C<CR>
```

or

```
.ACONSOLE=CRT<CR>
```

Q - Query Command

The Query command displays the current status of the system I/O devices. It displays a list of the system devices and the physical devices assigned to them. The format of the Query command is:

Q

where:

Q is the Query command code. No parameters are allowed with this command.

Example of the Query Command: To list the current assignments of system devices:

```
.Q<CR>
```

The system displays:

```
C=T
R=T
P=T
L=L
```

This response indicates that a teletype terminal is assigned as the Console, Reader, and Punch devices. A line printer is assigned as the *List* device, not the Console device.

Memory Control Commands

There are four Monitor commands with which you can work with the Intellec memory. The commands that only read memory can be used on RAM as well as PROM and ROM. The commands that write to memory can only effectively be used on RAM. If you specify ROM or PROM with these commands, no error indication is given but the write portion of the command is not executed.

The Memory control commands are:

- Display (D) which displays a specified range of memory.
- Fill (F) which overlays a specified range of RAM with a constant value.
- Move (M) which copies the contents of a specified portion of memory into another RAM location.
- Substitute (S) which modifies RAM on a byte-by-byte basis.

D - Display Command

The Display command displays a section of memory formatted into lines of 16 bytes separated by spaces with the address of the first byte at the left margin.

The format of the Display command is:

```
D<low-address>,<high-address><CR>
```

where:

D is the Display command code.

<low-address> specifies the beginning of the memory range to be printed. The address must be specified in hexadecimal. <low-address> must be less than or equal to <high-address>. If <low-address> is equal to <high- address>, a single byte is printed. Also, if <low- address> is greater than <high- address>, a single byte is printed.

<high-address> specifies the end of the memory range to be printed. The address must be specified in hexadecimal.

Both parameters of the Display command are required.

If the List device is not ready, the Display command will hang. To continue, make the List device ready, or, if that isn't possible, press the interrupt 0 button on the main chassis.

Example of the Display Command: To print the contents of memory locations 109H through 12AH:

```
D109,12A<CR>
```

The system prints:

```
0109 09 0A 0B 0C 0D 0E 0F
0110 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0120 01 02 03 04 05 06 07 08 09 0A
```

F - Fill Command

The Fill command writes a specified 1-byte constant into a specified RAM area. If ROM or PROM is specified, no error is issued, and the command continues to completion even though the memory does not change.

The format of the Fill command is:

```
F<low-address>,<high-address>,<constant><CR>
```

where:

F is the Fill command code.

<low-address> specifies the beginning of the memory range to be filled with the <constant>. The memory address must be entered in hexadecimal.

<high-address> specifies the last byte of the memory range to be filled with the <constant>. The memory address must be entered in hexadecimal.

<constant> is the byte to written to the specified address range. The <constant> must be entered as a hexadecimal number.

All three parameters of the Fill command are required.

If a character other than 0 through F is entered, the command is terminated and the prompt character (.) is displayed.

Example of the Fill Command: To initialize memory locations 20H through 2FH with 00H:

```
F20,2F,00<CR>
```

M - Move Command

The Move command copies a specified area of memory into an area of RAM. The move is done on a byte by byte basis, that is, the first byte of the specified area is copied to the new location, then the second byte is copied to the location following the first new location, and so forth. The data in the original location is not destroyed. Any data existing at the new location is overlaid.

The format of the Move command is:

```
M<start-address>,<end-address>,<destination-address><CR>
```

where:

M is the Move command code.

<start-address> specifies the address of the first byte to be moved. The address must be specified in hexadecimal.

<end-address> specifies the address of the last byte to be moved. The address must be specified in hexadecimal.

<destination-address> specifies the address to which the first byte (start- address) is to be move. Each subsequent byte is moved to a location one higher than the last.

All three parameters are required.

Because the command works on a byte by byte basis, you should be careful when attempting to move a block of data to a location within the block. By the time the command reaches the end of the block, the data will have been overlaid by the first data moved.

Examples of the Move Command: To move the data currently at address 0100H through 0200H to address 0400H through 0500H:

```
M100,200,400<CR>
```

To move the data currently at address 1000H through 1FFFH to address 1500H through 24FFH.

```
M1500,1FFF,1A00<CR>

M1000,14FF,1500<CR>
```

If you tried to do the above example with a single command:

```
M1000,1FFF,1500<CR>
```

the first 500H bytes would be copied as you expected, but the second 500H bytes would be a copy of the first 500H because bytes 1500H through 1FFFH were overlaid by the first 500H bytes.

S - Substitute Command

The Substitute command displays memory locations on an individual basis and gives you the option of modifying each location as it is displayed.

The Substitute command functions differently than most of the Monitor commands. The function of this command is performed before the carriage return (<CR>) is entered. The use of the command is described after the syntax and parameter descriptions. The format of the Substitute command is:

S<address>,[<data-byte>][,[<data-byte>]][...]<CR>

where:

S is the Substitute command code.

<address> specifies a RAM address. The address must be specified in hexadecimal.

<data-byte> specifies a single byte of data in hexadecimal that is to replace the byte currently at the location specified by address. This is an optional parameter. If it is not entered, the byte specified by address is not modified.

The Substitute command functions in the following manner:

1. Enter the command code and the address followed by a comma: S100,
2. The contents of the specified memory location followed by a dash is displayed: S100,FF-
3. You can now:
   - Modify the contents of the address by entering a new byte in hexadecimal: S100,FF-AA
   - Look at the next sequential byte of data by entering a comma: S100,FF-,00-

- End the command and not modify the data byte by pressing the carriage return key: S100,FF-<CR>
- Any combination of the first two and finally ending with a carriage return: S100,FF-AA,00-,11-22<CR>. The example changes the first byte from FFH to AAH, leaves the second byte unchanged, and changes the third byte from 11H to 22H.

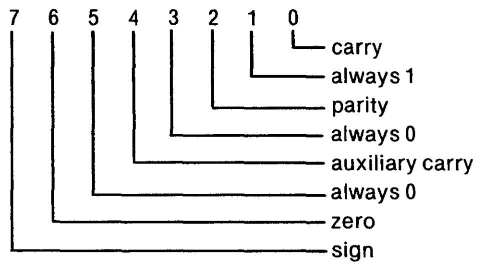
Register Command

There is one command to display register contents and to modify register contents. You can display the contents of all the registers, but to modify a register, you must specify a single register with the command.

The registers you can display and modify and their symbols in the Monitor are:

A	CPU A register
B	CPU B register
C	CPU C register
D	CPU D register
E	CPU E register
F	CPU flag byte
H	CPU H register
I	Intellec interrupt mask
L	CPU L register
M	CPU H and L registers combined
P	CPU program counter
S	CPU stack pointer

The F register is packed with the CPU condition flags:



X - Register Command (Display Form)

This form of the Register command displays the contents of all the registers. To modify register contents, use the modify form of the command. The format of the display form of the command is:

```
X<CR>
```

where:

X is the Register command code.

No parameters are used with this form of the command.

Example of the Display Form of the Register Command: To display the contents of the Intellec registers:

```
.X<CR>
```

The system displays:

```
A=00 B=78 C=00 D=47 E=11 F=02 H=FC I=FC L=20 M=FC20 P=1024 S=CD10
```

X - Register Command (Modify Form)

The modify form of the Register command allows you to display and optionally change the contents of the registers, one at a time.

This form of the Register command functions differently than most of the Monitor commands. The function of the command is performed before the carriage return (<CR>) is entered. The use of the command is described after the syntax and parameter descriptions. The format of the modify form of the Register command is:

X <register>, [<data>] [,[<data>]] [,...] <CR>

where:

X is the Register command code.

<register> specifies a single register by letter.

<data> specifies one or two bytes of data (depending on the register) to be placed in the register. The data must be entered in hexadecimal.

The modify form of the Register command functions in the following manner:

1. Enter the command code and the register letter:
   ```
   XC
   ```
2. The contents of the specified register followed by a dash is displayed:
   ```
   XC FF-
   ```
3. You can now:
   - Modify the contents of the register by entering new hexadecimal data:
     ```
     XC FF-00
     ```
   - Look at the contents of the next sequential register by entering a comma:
     ```
     XC FF-, EE-
     ```
   - End the command and not modify the register by pressing the carriage return key:
     ```
     XC FF-<CR>
     ```
   - Any combination of the first two and finally ending with a carriage return:
     ```
     XC FF- EE,EE-,02-82<CR>
     ```
     This example changes the C register from FFH to EEH, leaves the D register unchanged, and changes the flag byte from 02H to 82H.

Examples of the Modify Form of the Register Command: To examine but not change the M register:

```
.XM 1234-<CR>
```

To examine and change the M register:

```
.XM 1234-5678<CR>
```

To examine all the registers in sequence and change the D and H registers:

```
.XA 00-,11-,22-,33-FF,44-,02-,55-FF,EC-,66-,FF66-,FC9C-,E410-<CR>
```

To examine and change the interrupt mask (I) and the L register:

```
.XI FE-FC, 45-44<CR>
```

Paper Tape I/O Commands

The Monitor has four commands to support your usage of paper tape:

- Read (R), which reads data from a paper tape into the Intellec memory.
- Write (W), which writes data from Intellec memory to paper tape.
- End-of-File (E), which writes an end-of-file record to paper tape.
- Null Leader/Trailer (N), which writes null leader and trailer characters to paper tape.

The Intellec reads and writes paper tape in hexadecimal format. This format is described in detail in Appendix A.

R - Read Command

The Read command reads a paper tape in hexadecimal format from the device assigned as the Reader and loads the data into memory at the location specified in the record. A bias value can be specified in the command. The bias is a value that is added to the address specified in the record. The record is then loaded at a location determined by the address plus the bias value.

The format of the Read command is:

```
R<bias><CR>
```

where:

R is the Read command code.

<bias> specifies a value (modulo 65,536) to be added to the load address contained in the paper tape record. The data is loaded at the memory location specified by the record address and the bias value. The bias value must be specified in hexadecimal. Normally, a value of 0 is used.

NOTE

The addition of the bias value does not imply that the code is relocatable. In some cases, the code would not be executed at the biased location.

The data read is not changed in any way by the specification of a bias value.

Examples of the Read Command: To read a paper tape into memory:

```
.R0<CR>
```

To read a paper tape into a memory location that is 1000H above the address specified in the tape record:

```
.R1000<CR>
```

W - Write Command

The Write command punches the contents of a specified memory area to the assigned punch device. The memory area is specified by the beginning and ending addresses of the data to be punched.

The Write command does not put an end-of-file record on the paper tape. Thus you can punch non-contiguous areas of memory as a single file. The end-of-file record is written by the End-of-File command (E).

The format of the Write command is:

```
W<start-address>,<end-address><CR>
```

where:

W is the Write command code.

<start-address> specifies the first memory location to be punched onto the tape. <start-address> must be specified in hexadecimal.

<end-address> specifies the last memory location to be punched into the tape. <end-address> must be specified in hexadecimal. <end-address> must be higher than <start-address>.

The final record in a paper tape file must be an end-of-file record. After you have written the last memory area to tape, you must write an end-of-file record with the End-of-File command.

Examples of the Write Command: To write the contents of memory locations 200H through 3AFH to paper tape:

```
.W200,3AF<CR>
```

To write the contents of memory locations 450H through 54FH and locations 1000H through 1FFFH to paper tape as a single file:

```
.W450,54F<CR>

.W1000,1FFF<CR>
```

E - End-of-File Command

Every paper tape file must have an end-of-file record as the last record. If the end-of-file record is missing, the reader will read off the end of the tape.

You can specify an entry point address in the end-of-file record written with the End-of-File command. The entry point address is the address of the first instruction in the program to be executed. When this address is specified in the end-of-file record, the address is loaded into the Intellec program counter when the tape is read with a Read command. The program can then be executed by a simple Execute (G) command (described later). If the load address field is 0, the program counter is not altered by the Read command.

The format of the End-of-File command is:

E<entry-point><CR>

where:

E is the End-of-File command code.

<entry-point> specifies the entry point address of the program in the file to which the end-of-file record is being added. <entry-point> must be specified in hexadecimal. Of course an <entry-point> value does not make any sense unless the file contains executable code. A zero should be specified if an entry point address is not wanted.

Examples of the End-of-File Command:

To punch an end-of-file record in a tape that has just been written and specify an entry point address to be used when the tape is read with a Read command:

```
.E0<CR>
```

To punch an end-of-file record in a tape that has just been written and specify an entry point address to be used when the tape is read with a Read command:

```
.E1000<CR>
```

N - Null Command

The Null command punches a 60 null character leader or trailer. The null character is a 00H. You should punch a leader before writing data to a tape and after the end-of-file record. It makes the tape easier to load and saves the data on the tape from the usual damage that tape ends incur through normal handling.

The format of the Null command is:

N<CR>

where:

N is the Null command code.

Example of the Null Command: To punch a leader or trailer in a paper tape:

```
.N<CR>
```

Execute Command

The Execute command lets you transfer control of the Intellec system from the Monitor to your own program. You can specify the starting address and one or two breakpoints with the command. The starting address is optional. If it isn't specified in the command, the address in the program counter will be used. In the following conditions, you can be sure that the desired address is in the program counter:

- You interrupted your program with the interrupt 0 switch on the front of the Intellec and you have done nothing to destroy the program's registers.
- You loaded the program from paper tape and the end-of-file record contained the entry point address. This entry point address is loaded into the program counter by the Monitor.
- You modified the program counter to your entry point address with the Register command.
- Your program returned control to the Monitor because a breakpoint was encountered.

A breakpoint is the address of an instruction within your program that, if fetched, results in the return of control to the Monitor. You may want to do this so you can check the contents of certain registers or data fields at this point in your program. When a breakpoint is reached, the instruction at the address is not executed before control is returned to the Monitor. The instruction at the breakpoint is executed when you return control to your program with the Execute command.

There are two important points you must know when using breakpoints to debug and test your program:

- In saving the CPU status for your program, the Monitor uses the top 12 bytes of your program stack. This pushes the status of your registers and program counter into the stack. You should be aware of this when examining the stack. When control is returned to your program, your registers are restored and the stack pointer is reset as if the breakpoint had never occurred.
- The Intellec interrupt system is enabled when the Monitor is entered. The Monitor cannot determine the state of the interrupt system just prior to exit from your program. It is assumed that the interrupt system was enabled and so interrupts are enabled when control is returned to your program. It is your responsibility to either enable or disable the interrupt system.

G - Execute Command

The Execute command gives control of the Intellec to the program at the address specified or implied in the command. It optionally sets one or two breakpoints in the program to which control is passed.

If breakpoints are specified, the Execute command functions differently than most of the Monitor commands. Before the carriage return is entered, the Monitor prompts for breakpoints if a comma is entered after typing G. This use of the command is described after the syntax and parameter descriptions. The format of the Program Execute command is:

```
G[<start-address>][,<breakpoint1>][,<breakpoint2>]<CR>
```

where:

G is the Execute command code.

<start-address> specifies the address to be placed in the program counter. Control of the Intellec is passed to this address. The address must be specified in hexadecimal.

<breakpoint1> and <breakpoint2> specify points in the program where control will be passed back to the Monitor. The breakpoints are entered in hexadecimal.

When breakpoints are specified, command entry is in the following sequence:

1. Enter the command code and, optionally, the start-address followed by a comma:
   ```
   G1FA,
   ```
2. The Monitor displays a dash:
   ```
   G1FA,—
   ```
3. Enter the first breakpoint address:
   ```
   G1FA,—22C
   ```
4. Enter another comma if a second breakpoint is to be specified or a carriage return if only one breakpoint is wanted. If a second breakpoint is entered, follow it with a carriage return:
   ```
   G1FA,—22C,—24E<CR>
   ```
   or
   ```
   G1FA,—22C<CR>
   ```

If the command contains a syntax error, no breakpoints are set. The command must be re-entered and the breakpoints again specified.

When either of the breakpoints are reached and control is returned to the Monitor, or when control is returned to the Monitor because of an interrupt 0, both breakpoints are eliminated. If you want them when you resume execution of your program you must specify them again.

Examples of the Execute Command: To pass control to the program address in the program counter:

```
.G<CR>
```

To pass control to the program whose entry point is 30A:

```
.G30A<CR>
```

To pass control to the program whose entry point is 30A and to set a breakpoint at address 400 within that program:

```
.G30A,—400<CR>
```

To pass control to the program whose entry point is 30A and to set two breakpoints, at addresses 400 and 500, within that program:

```
.G30A,—400,—500<CR>
```

Utility Command

There is one utility command that does hexadecimal addition and subtraction for you. Many of your early errors in programming will come from mistakes in performing hexadecimal arithmetic.

H - Hexadecimal Command

The Hexadecimal command adds and subtracts two hexadecimal numbers for you. The numbers can be up to and including four hexadecimal digits. The format of the Hexadecimal command is:

```
H<number1>,<number2><CR>
```

where:

H is the Hexadecimal command code.

<number1> specifies the first number to be added. This is number is used as minuend for the subtraction. The number must be entered in hexadecimal.

<number2> specifies the second number to be added. This number is used as the subtrahend for the subtraction.

The command displays two four-digit values. The first is the addition of the two numbers and second is the subraction. Negative numbers must be entered in their twos complement form.

If more than four digits are entered, the command uses the rightmost four digits. The leading digits are lost.

Example of the Hexadecimal Command: To add and subtract E49 (minuend) and 111 (subtrahend):

```
.HE49,111<CR>
```