HES

HES MON 64™

By Terry Peterson

Instruction Manual

HES MON 64™

By Terry Peterson

If You've Never Used a 'Machine Language Monitor' Before

The following section is intended for people who are unfamiliar with the uses of a machine language (M.L.) monitor program. However, it is not a tutorial in the architecture of the C64 or the 6502. Nor is it intended to teach 6502 assembly language programming. In fact, some knowledge of assembler language will be most helpful. It IS intended to help the beginner get started in using HESMON. Even those who know nothing about the 6502 or the C64 will find some of HESMON's commands useful (see, for example, the Interpret Memory command).

If you are familiar with the C64's screen editor, you should have no trouble entering and editing HESMON commands. HESMON commands are entered and edited just as are BASIC direct mode commands. They consist of a single character usually followed by one or more 'parameters' and a RETURN. The parameters consist of hexadecimal numbers or character strings and are separated from one another by spaces. With one exception (the '#' command) numeric parameters must be hexadecimal and do not need to be prefixed with '$'. String parameters are identified by enclosing them in double quotes ("). If HESMON doesn't understand a command it will print '?', usually just to the right of the bad command. If the command is understood, but the result is impossible or illegal, e.g., trying to save HESMON itself on tape, HESMON prints a '?' on the following line.

To use HESMON, turn your C64 off, insert the HESMON cartridge into the expansion slot in the C64 and then turn the power on. You will see the HESMON version number, the programmer's name, the H.E.S. copyright message, and the 'cold start' register display:

```
C*
  PC   IRQ   SR  AC  XR  YR  SP
;0000 EA31  27  00  00  00  FA
```

The meaning of this rather cryptic display is as follows: The first line 'C*' identifies a cold start of HESMON, that is, starting up from power-on. The next line identifies the pseudo 6502 registers maintained by HESMON:

```
 PC = program counter
IRQ = interrupt request vector
 SR = status register
 AC = accumulator
 XR = X register
 YR = Y register
 SP = stack pointer
```

NOTE: "6502" is used synonomously for "6510" in this document.

The register contents are shown on the third line. The quantities shown in the register display (except the IRQ) are not the actual register contents, they are the numbers HESMON will use to set the 6502 registers when instructed to begin execution of a M.L. program. IRQ is not a 6502 register, but a RAM 'vector' that points to an IRQ interrupt service routine. Beginners may ignore this location — but better not change it! The ';' at the

beginning of the last line is really a HESMON command. It tells HESMON (if the RETURN key is pressed with the cursor on this line) to put the seven numbers that follow into the corresponding pseudo registers. Just before beginning execution of a M.L. program HESMON copies the pseudo register contents to the 6502 registers. So, for example, if we want the C64 to print 'HI.', we could first move the cursor up to the ';' line and alter it to read:

```
1200  EA31  27   48   49   2E   FA
```

When we press RETURN, the 6502 pseudo program counter is set to $1200; while the accumulator, and X and Y pseudo registers are set to $48 (ASCII H), $49 (ASCII I), and $2E (ASCII.). Now, if we write a program at $1200 to print the AC, XR, and YR it will print 'HI.' when we execute the HESMON Go command. Let's write such a program using the HESMON Simple Assembler command, 'A'. Type in the following lines:

```
A1200  JSR  FFD2
TXA
JSR  FFD2
TYA
JSR  FFD2
BRK
```

The 'A' beginning the first line tells HESMON we wish to assemble, that is, translate assembly mnemonics into machine code. As you press RETURN after typing each of the above lines, you will see HESMON reprint the line, showing the machine code generated from the assembly language instruction. HESMON will then prompt for the next line of program by printing the 'A' command and the next available address followed by a space. So you don't have to keep track of what the next address is, just type in the assembly language instructions. When you've finished the program, just press RETURN and HESMON will exit this mode. By the way, $FFD2 is one of the 'Kernal' routines in the C64's ROMs. It prints the contents of the accumulator to the current output file — the screen in this case. For further information on this and other useful ROM routines, consult the Commodore 64 Programmers' Reference Guide" published by Commodore

Now type 'G' and hit RETURN. You should see:

```
G
HI.
B*
  PC   IRQ   SR  AC  XR  YR  SP
;120C EA31  30  2E  49  2E  FA
```

Notice after the 'HI.' is another register display, the break entry display identified by 'B*'. This means we've re-entered HESMON by executing a BRK instruction — the one at the end of our short program. Now examine the register contents. The PC points one address higher than the BRK instruction. The X and Y registers and stack pointer are unchanged. The accumulator now has the $2E transferred into it by the TYA instruction at $1207. Let's play with this a bit. Type 'D1200 120B'. This command instructs HESMON to 'disassemble' the program you just entered.

Now, move the cursor to the last line, at address $120B, and type the following, with the 'A' replacing the ',' (also

be sure to blank out any characters left on the screen after the '8'):

```
A120B  LDA  #48
JMP  1200
```

We now have a M.L. program that will print 'HI.' forever — or until we stop it. Type 'G1200'. When you tire of watching the stream of 'HI.HI.HI.'s, press — no, not the STOP key — the RESTORE key by itself. The RESTORE key is HESMON's super-STOP key. It will halt just about any M.L. program (except HESMON itself) when HESMON is plugged in. (Exception: If you attempt to use RS232 files all bets are off. Also, correct operation of RS232 files is not guaranteed with HESMON installed.) To get back to our example: after pressing RESTORE you should see a clear screen with the following:

```
S*
  PC   IRQ  SR  AC  XR  YR  SP
;XXXX EA31  XX  XX  XX  XX  XX
```

This is the RESTORE entry display, identified by the 'S*'. The X's are not actually what you will see. The register contents will depend upon exactly when you pressed RESTORE.

If you want to enter a series of bytes into memory, use the Memory Modify command (:). For example, to enter the sequence $01, $02, $03, $04, $05, $06, $07 . . . starting at $1234, you type:

```
:1234 01 02 03 04 05 06 07 08
```

HESMON will respond by reprinting the line and will prompt for another line by printing the next available address. As with the Assemble command, you may exit by typing RETURN.

Besides entering programs and data into memory, one of the functions of a M.L. monitor is to examine programs and data already in memory. HESMON has several commands for this purpose; including Disassembly (D), Memory Display (M), and Interpret Memory (I). These three commands are special in that the cursor-up and cursor-down key may be used to 'scroll' their displays forward and backward through memory. The action of this scrolling is easier to use than to describe. Think of the text on the screen as being on a drum which may be rolled up or down using the cursor up/down key. The scrollable display type found closest to the edge of the screen where new lines will appear is continued in the scroll direction. I said it was hard to describe! Try it. Just type 'DAAD7' and hit RETURN. Then press and hold the cursor-down key. To scroll up, go to the top of the screen and then hold down the cursor-up key.

Other commands allow you to hunt for a particular sequence of bytes in memory (H), compare two blocks of memory for differences (C), or transfer a block of memory to a different location (T). There are also two advanced functions: N—relocate absolute memory references in a program, and E—change the external references in a program. Finally, there are number base conversion and hexadecimal arithmetic functions.

Alphabetical List and Description of HESMON Commands

The following section lists the HESMON commands in alphabetical order describing each in detail and giving example(s) of its usage.

A — The Simple Assembler

The HESMON simple assembler provides an easy way to enter short M.L. programs. It does not have all the features found in a complete assembler such as HESBAL in HES's 6502 Professional Development System for the VIC and Commodore 64, but it provides increased convenience compared to POKEing from BASIC or entering hexadecimal codes using a more primitive monitor. The syntax of HESMON's Assembler command is as follows:

A 1111 MMM OOOOO

where '1111' is a four digit hexadecimal address in the C64's RAM, 'MMM' is a standard three character assembler mnemonic for a M.L. operation code (op-code), such as JSR, LDA, etc. 'OOOOO' is the 'operand' of the op-code. It is beyond our scope here to discuss fully the meaning of those parameters — for a complete discussion, consult a book on 6502 assembly language programming. See Section I for a simple example of A's usage. Notice that since all numeric operands MUST be in hexadecimal notation the customary '$' preceding these numbers is optional; as is the ',' preceding 'X' or 'Y' in indexed instruction operands. If HESMON understands the line, it will reprint it showing the corresponding byte(s) of M.L. between the address and the assembly code. HESMON will then prompt for the next line of assembly code by displaying the next address followed by a space and the input cursor. If HESMON cannot interpret the line, it will print a '?' instead of prompting for the next line. For example, you type:

A 1200 LDA #41

HESMON responds by overprinting your line and then prompting for the next line as follows:

A 1200 A9 41 LDA #$41
A 1202

Note — HESMON ignores anything to the right of a ':' on the line.

B — Breakpoint Set

There are three different methods to return to HESMON from a M.L. program. The Breakpoint Set command is one of them. This command allows you to designate an address in a program as a 'breakpoint,' that is, a place where the program is to be halted and control is to be returned to HESMON. Breakpoint Set also allows you to specify the number of times the instruction at this address is to be executed before the breakpoint is activated. The breakpoint defined with Breakpoint Set is effective ONLY when the C64 is executing HESMON's Quick Trace command. For example, to halt a program, that starts at address $1200, on the fifth repetition of the instruction at address $1234, you would type:

B 1234 0005
Q 1200

The first line above sets the breakpoint at $1234 and the repeat count to five. The second line initiates the Quick Trace mode of program execution (see the Quick Trace command). When address $1234 has been reached for the fifth time HESMON will halt execution of the program, display the current values of the 6502 registers, and enter the single-step mode of execution (see the Walk command).

The second method to return to HESMON from a M.L. program is to insert a 6502 'BRK' instruction into the program. Obviously, since this method requires program modification, it may be used only with programs in RAM. Finally, HESMON may be called by simply pressing the RESTORE key. In either of these last two cases HESMON will be re-entered whether or not the Quick Trace mode was active. If a BRK instruction was encountered, the 'break' entry register display will be printed showing the contents of the 6502 registers. Similarly, if the RESTORE key is pressed, the RESTORE entry register display is shown. In the latter case, the screen is cleared first. The RESTORE key method of HESMON re-entry will work any time the HESMON cartridge is plugged in — unless an RS232 file has been accessed or the 6502 has attempted to execute an undefined op-code (one that disassembles as '???'). After an RS232 file has been attempted HESMON may be re-entered from BASIC via a BRK instruction. Type 'SYS8' to cause a break entry.

C — Compare Memory Blocks

This command compares two sections of memory and reports any differences by printing the address of one member of the mismatched pair(s). The syntax is as follows:

C 1111 2222 3333

where 1111 is the start address of the first section, 2222 is the end address of the first section, and 3333 is the start address of the second section — the one to be compared with the first section. This command may be stopped (in case a large number of addresses are printed) with the STOP key. For example, suppose you have two disk files containing (you thought) the same M.L. program residing at locations $1400 to $147F. However, when you used the BASIC command VERIFY, it said 'VERIFY ERROR'. Naturally, you wonder just where the difference is. VERIFY can only tell you they differ SOMEWHERE. Compare Memory Blocks may be used to find out: First use HESMON's Load command to load one of the files (See Load). Then move that program to $1500 using the HESMON Transfer Memory Block command: T 1400 147F 1500. Next Load the other file. Now compare the two files using Compare Memory Block:

C 1400 147F 1500

HESMON will print a list of all the memory locations which differ between the two programs.

D — Disassemble Memory

This command is the inverse of the Assemble command. It interprets memory contents as M.L. instructions and displays the assembly language equivalent. Disassemble is used in

two distinct ways. First, it may be used to disassemble a section of memory by specifying an address range, such as:

D 1111 2222

where 1111 is the start address and 2222 is the end. This type of disassembly is convenient when used in conjunction with HESMON's Output Divert command to produce a hardcopy listing of a M.L. program. Second, the disassemble command may be started by entering a single parameter, the beginning address:

D 1111

This mode is handy for examining a M.L. program on the screen because, once the first line is displayed, preceding or subsequent lines of code may be disassembled by pressing the cursor-up or cursor-down key respectively.

You may alter a program in RAM using the Disassemble command's output. If you move the cursor to the line you wish to alter, change the byte display (not the mnemonic), and press return; HESMON will alter the memory contents and retype the line showing the altered bytes and the corresponding disassembly. Then HESMON will prompt for the next line by printing the next address and leaving the input cursor on the same line. To exit this mode type RETURN, just as with the Simple Assembler command.

E — External Relinker

This command is rather difficult to understand, but the effort is worth it! Basically, this command facilitates the transport of M.L. programs from one 6502-based computer to another (PET, VIC, etc.) by translating the system calls of one computer to those of another. Of course the capabilities of these computers are different so one cannot always achieve a perfect translation, but at least a functioning version can be made without completely rewriting the program. The heart of this command is a table of corresponding addresses. This table contains four-byte entries consisting of pairs of addresses. These address pairs are the addresses in the respective computer operating systems that perform a given task. Typically these will be addresses in the ROM firmware of the computers. The correspondence table must be supplied by you. Lists of common ROM routine addresses in various 6502 computers have appeared in several places, most notably in COMPUTE! magazine (e.g., "VIC Memory Map Above Page Zero", COMPUTE! Vol. 4, No. 1, P. 181); "Butterfield on Commodore", Commodore Magazine, Oct./Nov., 1982, pp. 81 ff.; and, for the PET, in "PET/CBM Personal Computer Guide" by Osborne and Donahue.

For example, suppose you have loaded into your C64 a M.L. program intended to run in a PET with BASIC 4.0 ROMs. We will assume it is in locations $1200 to $13FF. Many of its external subroutine calls are probably of the form JSR $FFxx. The subroutines at these addresses are all almost identical in function to those of the same address in the C64 because these entry points are in a 'jump table' set up for the purpose of standardizing system calls between the different Commodore ROM sets. So what's the

problem? Any subroutine call in the address range $B000 to $FF00 probably also has an equivalent in the VIC, but it's at a different address! This is where External Relinker comes in. External Relinker will find such subroutine calls and replace them with the corresponding C64 ROM routine calls — if we can identify the correct replacement (this is where the published ROM maps come in). If we already have a correspondence table constructed in an earlier session with External Relinker, we simply load it using the Load command. But, if we don't have a table, External Relinker will use our answers to its queries to construct one we may save for future use. For the present example, suppose we have no table, just two ROM maps. We want to construct a table starting at $1000, so we start it by entering four zeroes (four zeroes denote the last entry in the table) using the Fill Memory Block command.

F 1000 1003 00

Then we start External Relinker:

E 1200 13FF 1000 B000 FF00

The first two parameters tell External Relinker where the start and end of the program we are working on are. The third says where the correspondence table starts. The last two give the address range we're interested in relinking. At this point External Relinker will start disassembling our program from $1200 to $13FF, looking for references to addresses in the specified range of $B000 to $FF00. When it finds such an address it will first consult the correspondence table which starts at $1000 — if no entry for the address is found, it will show the disassembled line containing the unknown address and wait for the entry of the correspondence address. We will look up the PET address in the published table, find its equivalent in the C64 table, type the VIC address over the one on the screen, and press RETURN. HESMON will add the new correspondence to its table, alter the address reference in the program and then continue its search. On subsequent occurrences of this address HESMON will automatically make the specified replacement.

F — Fill Memory Block

This command is used to set a section of memory to a particular value. The syntax is as follows:

F 1111 2222 33

where 1111 and 2222 are the first and last addresses (inclusive) of the section to be filled and 33 is the hexadecimal quantity to be written. See, for example, the usage in the example of External Relinker.

G — Go (execute program)

This command transfers control of the C64 to a M.L. program; that is, it starts execution of the M.L. program. It may be used with or without an address parameter. If no address parameter is given, execution is begun at the address shown in the program counter (PC) of the Register Display command. For example you may exit HESMON and 'warm start' BASIC by typing:

G A474

The C64 will respond, "READY.". For another example, see Section 1.

H — Hunt for a Sequence

This command locates a specific sequence of bytes in memory. It has two forms, as follow:

H 1111 2222 33 44 55 . . . .
H 1111 2222 "ABCDE . . . . "

where 1111, 2222 are the first and last addresses of the range of memory to be searched and 33, 44, etc., are the hexadecimal byte(s) to be found, separated by spaces. The second form allows the bytes to be specified as characters enclosed by quotes. For example to find all subroutine calls to the character output routine (AB47) in the C64 ROM's we would type:

H A000 FFFF 20 47 AB

HESMON responds with a list of all such subroutine calls. Note that, as usual, the low and then high order bytes of the address were specified.

To find all occurrences of the string 'READY' (there is only one, at $A378), we would type:

H A000 FFFF "READY"

I — Interpret Memory

This command displays the contents of memory as 'ASCII' characters. It is similar to the Memory Display command except that it shows 32 characters per line. It may be used with either one or two parameters and its output may be scrolled just as with the Disassemble command. For example, to see the table of BASIC's keywords and error messages, type:

I A000 A300

L — Load 'Program'

This command 'loads' (i.e., reads) a 'program' into memory from an external device such as tape or disk. The loaded material need not actually be a program. For example, it may be a section of memory containing a data table for External Relinker that was saved to tape or disk using the Save command. However, the most common use of Load is to retrieve M.L. programs from tape or disk. Note that HESMON's Load should NOT normally be used to load a BASIC program. The syntax of Load is as follows:

L "programname" 11

where 'programname' is the name of the file to be loaded (be sure to include the double quote marks) and '11' is the device number from which to load. If the device number is omitted, the tape drive will be assumed; if the filename is also omitted, the first file found on the tape will be loaded. For example:

L "YAHTZEE" 08

The above loads YAHTZEE from device eight, the disk drive.

M — Memory Display

This command displays the contents of memory in hexadecimal notation. It This command displays the contents of memory in hexadecimal notation. It is similar to the Disassemble command in that it may take either one or two addresses as parameters. The two-parameter form displays from the first address to the second; the one-parameter form shows eight bytes beginning with the address given. Also like the Disassemble command, the output of Memory Display may be

scrolled up or down with the cursor-up and cursor-down key. For example:

M A000 A040

shows from $A000 through $A047 in hex and in characters, eight bytes per line. To see more, press cursor-up or -down.

N — New Locator

This command is a relative of the External Relinker command. It has a different general purpose, however. New Locator is designed to convert absolute address references in a M.L. program from one memory range to another. It is typically used following a Transfer Memory Block command to relocate a program in memory. For example, suppose you have just moved a M.L. program from $1200-$1280 to $1300-$1380 using T. Any address references within the program now point $0100 too low. New Locator can fix this. Type:

N 1300 1380 0100 1200 1280

The meaning of the above line is as follows: Disassemble from $1300 to $1380 checking for addresses in the range $1200 to $1280. Add $0100 to any such addresses. If we had moved a table of addresses, for example a 'jump table' (pairs of numbers of addresses, low byte followed high byte), instead of actual machine code; we would put a 'W' following the last parameter to tell New Locator to treat the memory contents as pairs of address bytes rather than M.L. The general Syntax for New Locator is the following:

N 1111 2222 3333 4444 5555 [W]

where 1111 and 2222 specify the actual memory range to scan, 3333 is the 'offset' to add to adjusted addresses, 4444 and 5555 specify the address range of references which are to be adjusted, and W (if present) specifies that the scanned range is a table of 'words' with no op-codes. If not in the 'word table' mode, New Locator will halt and display any line of machine code it can't disassemble.

O — Output Divert

This command is HESMON's equivalent to BASIC's CMD command. It allows HESMON's output to be printed on the C64 printer or stored in a disk file instead of being displayed on the screen. This is the preferred method to get HESMON's output on a device other than the screen. Output Divert has a number of options. The complete syntax of the command is:

O11 22 "filename"

where '11' is the device address where the output is to be sent (normally 04 for the printer), '22' is the 'secondary address' of the device (typically 02 to 0E for the disk drive), and 'filename' is the filename to be used for storing the output (see your disk drive documentation). All of these parameters are optional. If you merely type 'O' HESMON will open a file to device 4, the printer, and start diverting its output. If you type 'O' when the output is already being diverted, the file will be closed and the output will be directed to the screen again. That is, typing 'O' 'toggles' Output Divert on and off. If you want explicitly to revert to screen output, type 'O3F'. The secondary address and filename default to 'none' since they are not needed by the printer. For more information about

filenames and secondary addresses, consult the documentation for the device to which you wish to divert HESMON's output.

P — Print Screen

This command is a limited version of Output Divert. It copies the current screen display to printer or disk. It's just like having a snapshot of the current screen image. The parameters of Print Screen are the same as for Output Divert, except there is no toggling because Print Screen automatically reverts to screen output at the completion of the screen copy. Note: Print Screen will NOT copy high resolution graphics.

Q — Quick Trace

This command is used after the Breakpoint Set command in debugging M.L. programs. It takes one or zero parameters just like the Go command. If specified, the parameter gives the address at which to begin execution. If omitted, execution begins at the PC shown in the register display. The difference between Quick Trace and Go is that a breakpoint, defined with the Breakpoint Set command, is only recognized in the Quick Trace mode of execution — the breakpoint will be ignored if execution is begun with the Go command. Program execution is much slower with Quick Trace than with Go because Quick Trace is really just a fast version of the Walk (single step) command. Using Quick Trace, instructions are executed one at a time and HESMON is re-entered after each. This process continues until the defined breakpoint is reached. For an example of Quick Trace usage, see the Breakpoint Set command.

R — Register Display

This command displays HESMON's current 6502 pseudo register contents as well as the current interrupt request (IRQ) RAM vector. The IRQ vector is shown as a convenience to the programmer who wishes to use this vector to run interrupt-driven or 'background' routines. This vector may be altered like any of the register contents; however, extreme caution must be exercised in so doing because the replacement is made *IMMEDIATELY*, not at the time of execution of a Go command. Therefore, the interrupt handling routine must be in-place BEFORE the IRQ vector is altered.

There are no parameters for the Register Display command, just type 'R'. To alter the register contents, move the cursor to the line beginning with ';' and overwrite the display. Then hit RETURN and the contents will be altered. Note that the display, except as noted for the IRQ vector, shows the contents of the 6502 registers at the time HESMON was entered. These registers will be set by HESMON to the values shown in the register display just prior to beginning execution of a program using the Go, Quick Trace, or Walk commands. For a fuller discussion of the meaning of this display, see Section I.

S — Save 'Program'

This command saves the contents of a specified range of memory to an external device as a non-relocating 'program' file. The 'non-relocating' part means that the program may be

reloaded from tape using BASIC's LOAD command. The syntax of Save is as follows:

S "filename" 11 2222 3333

where 'filename' is the filename to be used (don't forget the double quote marks), '11' is the device number on which to save (01 for the tape and 08 for the disk drive). '2222' is the beginning address. '3333' is the last address PLUS ONE of the memory area to be saved. All the parameters must be given, except that in tape saves the 'filename' may be null (""). For example, to save a M.L. program residing from $1500 to $1DFF to the disk as 'APROGRAM', type:

S "A PROGRAM" 08 1500 1E00

Again, notice the last parameter is *one byte higher* than the last program address. Also, note that HESMON's Save should NOT be used to save BASIC programs because HESMON saves programs as absolute, not relocatable, files.

T — Transfer Memory Block

This command transfers the contents of a block of memory to another area. Its syntax is as follows:

T 1111 2222 3333

where 1111, 2222 are the first and last address (not last-plus-one) of the block to move and 3333 is the starting address where the block is to be moved to.

U — (Test Color RAM)

U has no parameters. It tests the color RAM for proper function and prints 'OK' if they are working. If there is a bad byte, its address will be printed.

V — Verify RAM Function

This command tests a section of RAM for proper function. Its syntax is:

V 1111 2222

where 1111, 2222 are the first and last memory locations of the block to test. HESMON will keep cycling the test over the address range specified until the STOP key is pressed (it may be necessary to hold it down for a second or two). At the successful completion of each test of the memory block, HESMON will print a '.' to show it is working. If a memory location fails the test, HESMON will print its address followed by a binary number showing the data incorrectly stored. The bits of the number are shown most significant (bit 7) to least significant (bit 0) left to right. The bits of the RAM location that are different from the test data are printed in reverse field. Using the information printed on the screen it will usually be possible to pinpoint the bad RAM IC(s). Note that if you 'test' addresses that contain no RAM, a seemingly random pattern of numbers will be printed.

W — Walk Program

This command causes single-step execution of a M.L. program under user control. It, like Go and Quick Trace, may be used without a parameter to begin at the register display 'PC' location; or it can accept one parameter that specifies the starting address. To exit the Walk mode, press the STOP key. To step as rapidly as the registers can be printed, press the SPACE bar. To step at the key repeat rate, press a normally repeating key, e.g., the cur-

sor down key. To take one step only, press a normally non-repeating key, e.g., the left-arrow key. The 'J' key has a special function in Walk mode. It causes HESMON to continue execution at full speed until a return-from-subroutine instruction is executed. For example, type:

```
W AAD7
```

HESMON will begin execution at $AAD7 the carriage return, linefeed output ROM routine. After executing the instruction at that address HESMON will halt, showing the register contents and a disassembly of the next instruction the C64 will execute if Walk is continued. The display in the above example is as follows:

```
25 0D 00 00 FA
,AAD9 20 47 AB JSR $AB47
```

The first of the two lines above shows the 6502 register contents in the same order as the Register Display command: SR AC XR YR SP. This example assumes HESMON has just been cold started, otherwise the registers — except the accumulator — may differ from those shown here. The second line shows that the C64 will next do a subroutine call to $AB47, the character output routine used by BASIC. To continue, press any key except STOP or 'J' (no need to hit RETURN). Suppose we press the left-arrow key once. HESMON will now show two more lines:

```
25 OD 00 00 F8
,AB47 20 0C E1 JSR $E10C
```

Now we see the C64 is at location $AB47 about to execute a subroutine call to $E10C. Notice the stack pointer (SP) has been decremented by two because the return address for the JSR instruction was 'pushed' on the stack before the jump to $AB47 was executed. Let's press the left-arrow once more:

```
25 OD 00 00 F6
,E10C 20 D2 FF JSR $FFD2
```

Here we finally get to a place where the C64 is going to a 'Kernal' routine we can recognize: the character output routine $FFD2. Since this routine is documented in the C64 literature, we know exactly what it will do: print the character $0D in the accumulator. Therefore, we needn't single step further through that routine. So we press the 'J' key. HESMON shows (after a blank line — where the carriage return was printed):

```
20 0D 00 06 F6
,E10F B0 E8    BCS $E0F9
```

Now the C64 is at the point just following the JSR $FFD2 instruction. The 'carry' bit (bit 0) of the status register (SR = $20) is clear (0), so the branch on carry set (BCS) will not be taken. At this point we may continue to single step through this subroutine by pressing left-arrow; return to the next higher level of code (SP = $F8) by pressing 'J'; or quit the Walk command by pressing STOP.

X -Exit to BASIC

This command gives control to the C64's BASIC interpreter. It has two forms. The first form 'XC' has the same effect as if the C64 were turned off and then back on without the HESMON cartridge plugged in except that HESMON may be entered by pressing RESTORE. The second form 'X' causes a 'warm start' of BASIC,

similar to pressing RESTORE when HESMON is not plugged in. **Your first exit to BASIC from HESMON after turning on the C64 should be an 'XC', otherwise BASIC may misbehave.** While in BASIC, to achieve the same effect as pressing STOP & RESTORE without HESMON: First press RESTORE. Then type 'X' and hit RETURN.

 # — Convert Decimal to Hexadecimal

This command prints the hexadecimal equivalent of a decimal number. If the decimal number is negative it shows the two's complement 16-bit hex equivalent and the corresponding positive decimal number. For example:

1234

HESMON shows (on the same line):

1234 = $04D2 1234

$ — Convert Hexadecimal to Decimal

This command prints the decimal equivalent of a hexadecimal number. For example:

$ ABCD

HESMON shows (on the same line):

$ ABCD 43981

+ — Hexadecimal Addition

This command prints the sum of two hexadecimal numbers in hex and decimal. All four digits, including leading zeroes if needed, must be used. Example:

+ 1234 5678

HESMON shows (beginning on the same line):

+ 1234 5678 = $68AC 26796

– — Hexadecimal Subtraction

This command prints the difference of two hexadecimal numbers in hex and decimal:

– 1234 5678

HESMON shows (beginning on the same line):

– 1234 5678 = $BBBC 48060

Notice that the decimal number in this example is positive even though we would expect the result of this subtraction to be negative. This is because the two-byte number $BBBC doesn't retain the information that the result is negative. If you want to know the true negative decimal result, either type in the operands in the reverse order, or type:

– 0000 BBBC = $4444 17476

So, the true decimal value of the difference $1234 – $5678 is – 17476.

Things to be careful about when using HESMON

The BASIC interpreter has control of the C64 at all times when BASIC is running. This means that the worst that's likely to happen if your BASIC program has an error is that BASIC will issue a 'SYNTAX ERROR' message and stop your program. A M.L. monitor, on the other hand, must allow its user to take complete control of the C64 to execute certain commands. So, if your M.L. program has an error and you attempt to execute it using the Go command, the likely result is that the C64 will go catatonic — that is, even the RESTORE key may not bring back HESMON. In this event you will have to turn the power off and back on to get back to HESMON. You may avoid this catastrophe by using the Walk command to check out your program. Nevertheless, you can still send the C64 to never-never land by attempting to Walk through an instruction that disassembles as '???'. These instructions are 'unimplemented op-codes'. They do not have a defined result. Many of them cause the 6502 to 'crash' — that is, enter a state from which it may be recovered only by powering on again.

HESMON uses 33 bytes near the bottom of the machine stack ($120-$141) for its variable storage. Most M.L. programs do not use a sufficiently large amount of the stack to interfere with this storage — but it is a possibility to be aware of. Large, complex BASIC programs sometimes do use enough of the stack to interfere with these locations. And finally, RS 232 files will **not** work correctly when HESMON is plugged in.

Acknowledgements

The seeds of HESMON are contained in the public domain monitor programs for the PET/CBM computers known as MICROMON and EXTRAMON. These programs, while not directly useful in the C64 environment, provided at least the general framework and the philosophy of user-friendliness which distinguish them and HESMON from other M.L. monitors of the author's experience.

Appendix A

The HESMON Commands in Brief

The following is a condensed list of HESMON's commands for quick reference. Brackets ([]) denote optional parameters.

A 1111 MMM OOOOOO — Simple Assembler
B 1111 2222 — Breakpoint Set
C 1111 2222 3333 — Compare Memory Block
D 1111 [2222] — Disassemble
E 1111 2222 3333 4444 5555 [W] — External Relinker
F 1111 2222 33 — Fill Memory Block
G [1111] — Go
H 1111 2222 33 44 55 . . . . or
  1111 2222 "XXXXX . . . ." — Hunt for sequence
I 1111 [2222] — Interpret Memory
L "name" 11 — Load Program
M 1111 [2222] — Memory Display
N 1111 2222 3333 4444 5555 [W] — New Locator
O [11 [22 ["name"]]] — Output Divert
P [11 [22 ["name"]]] — Print Screen
Q [1111] — Quicktrace
R — Register Display
S "name" 11 2222 3333 — Save Program
T 1111 2222 3333 — Transfer Memory Block
U — Test Color RAM
V 1111 2222 — Verify RAM
W [1111] — Walk
X[C] — Exit to BASIC
11111 — Decimal to Hex
$ 1111 — Hex to Decimal
+ 1111 2222 — Hex Addition
– 1111 2222 — Hex Subtraction
: 1111 22 33 44 55 66 77 88 — Memory Modify
; 1111 2222 33 44 55 66 77 — Register Modify
, 1111 11 [22 [33]] XXXX — Disassembly Modify

Human Engineered Software
71 Park Lane
Brisbane, California 94005
Telephone (415) 468-4110