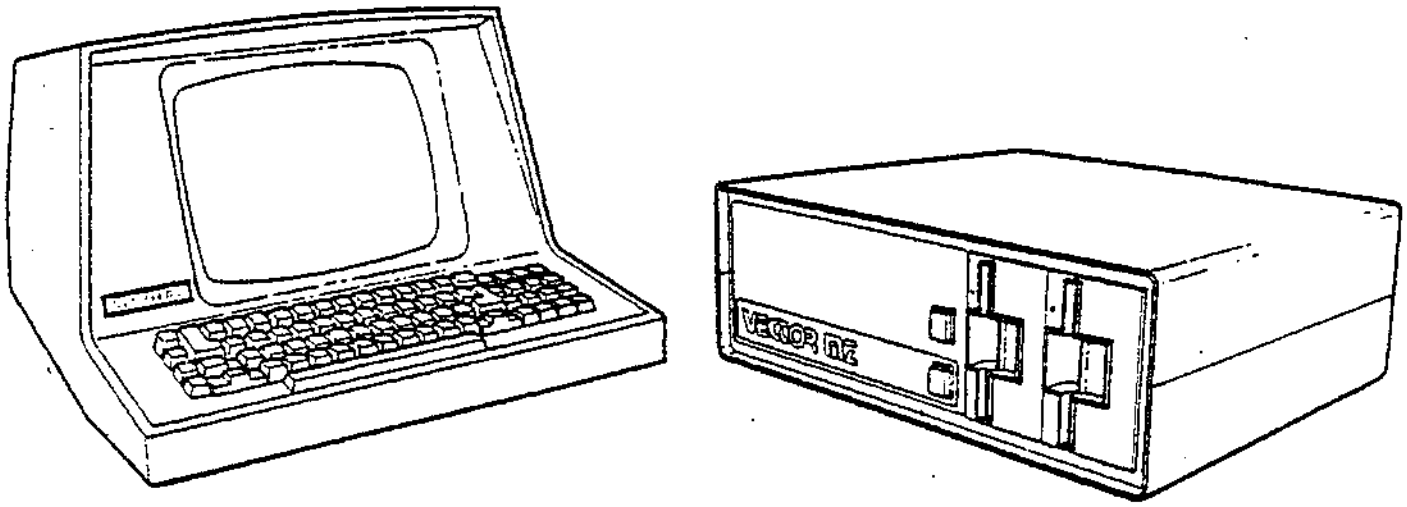
RAID

RApid Interactive Debugger

USERS MANUAL



VECTOR
VECTOR GRAPHIC, INC.

RAID - RAPID INTERACTIVE DEBUGGER

Version 1.1

REFERENCE MANUAL

Revision A

May 30, 1980

Disclaimer

Vector Graphic makes no representations or warranties with respect to the contents of this manual itself, whether or not the product it describes is covered by a warranty or repair agreement. Further, Vector Graphic reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Vector Graphic to notify any person of such revision or changes, except when an agreement to the contrary exists.

Revisions

The revision letter such as A or B changes if the MANUAL has been improved but the PRODUCT itself has not been significantly modified. The date and revision on the Title Page corresponds to that of the page most recently revised. When the product itself is modified significantly, the product will get a new revision number, as shown on the manual's title page, and the manual will revert to revision A, as if it were treating a brand new product. EACH MANUAL SHOULD ONLY BE USED WITH THE PRODUCT IDENTIFIED ON THE TITLE PAGE.

FOREWORD

Audience

This manual is intended for ASSEMBLY LANGUAGE programmers. It assumes an in depth knowledge of the Z80 CPU and the Vector Graphic computer, as well as familiarity with the CP/M operating system.

Scope

This manual will describe the features and commands of Vector Graphic's RAID - Rapid Interactive Debugger - an advanced full-screen simulator/debugger.
As a reference manual, it is not intended to be a training manual for novice programmers.

Organization

Chapter 1 reviews the implications of RAID in the context of the Vector computer system and of the CP/M operating system. Chapter 2 discusses the structure of the debugger in general. Chapter 3 gives each of the commands offered by RAID, listing them alphabetically.

SUMMARY OF RAID COMMANDS

Loading RAID

RAID [<d:filename>.HEX]	Load RAID for screen at F000H and load <filename> as in Disk operations below.

Load file operations

L <d:filename.filetype>	Load file from disk at 0100H.
L <d:filename>.HEX	Load HEX file from disk at its own load address (not necessarily 0100H) and convert to executable format.

Transitions

P <address>	Set PC to <address>.
G <address>	Go to <address> and execute (RAID not in control).
Q	Quit RAID; return to calling system.
<ESC>	Go to Extended Systems Monitor executive.
I	Initialize screen (clear off extraneous material).

Execute under RAID control

XS [<rate>]	Execute slow and refresh entire screen. Higher <rate> slows execution speed.
XF [<rate>]	Execute fast and refresh entire screen. Higher <rate> speeds execution.
XP [<rate>]	Execute fast and refresh partially (only registers and stack). Higher <rate> speeds execution.
XN [<address> <address>...]	Execute with refresh only at given addresses (4 max).
XD [<address> <address>...]	Execute direct (no simulation), with hard breakpoints at given addresses (max 6).
XI <hex instruction>	Execute <hex instruction> immediately (4 bytes maximum.)
<TAB>	Single step: execute instruction at PC with full screen refresh.
<LF>	Execute direct and return

Halts during simulation

HP <address>	Halt when PC = <address>.
HO <opcode>	Halt when executed opcode = <opcode>.
HR <register or pair> <value>	Halt when <register> or <register pair> = <value>.
HM <address> <byte>	Halt when contents of <address> = <byte>.

Dump

DA <address> [<line>[-]]	Dump ASCII, beginning at <address> in memory display area. <line>[-] limits applicable portion of display area.
DH <address> [<line>[-]]	Dump hex, as described for DA.

Enter

EH <address> <byte> <byte>	Put <byte>s in memory <address>.
EA <address> <char> <char>	Put ASCII characters in memory <address>.
ER <register or pair> <byte or word>	Put <byte or word> in <register or register pair>.

Stack

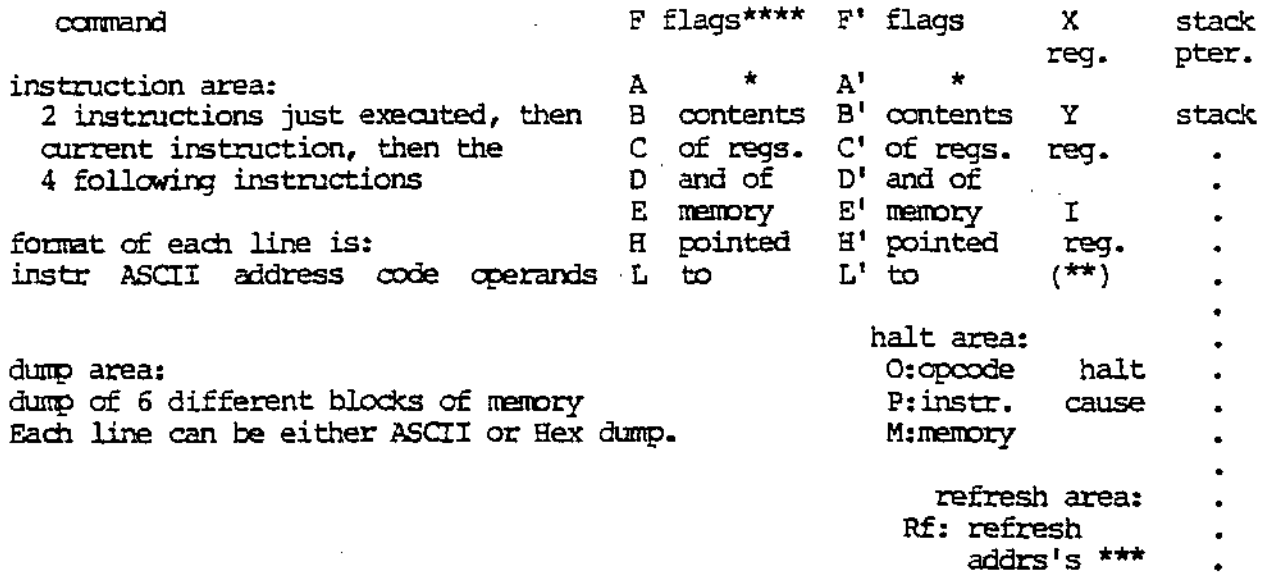
SS <address>	Set start of stack displayed on screen.
SC	Clear extraneous material from stack display.
SP	Pop word (2 bytes) off stack.
SH <2 byte word>	Push word onto stack.

Help

?	Display HELP screen at any time.

Fill

F <start address> <end address> <byte>	Fills working memory from the start address to the end address with the byte specified.



* contents of the A and A' registers include the ASCII equivalent
** (di) means interrupts are disabled. (ei) means they are enabled.
*** address of current refresh is highlighted.
**** flag symbol appears when flag is set.

RAID Display

TABLE OF CONTENTS

I. PERSPECTIVE

1.1 The value of RAID

Vector Graphic's memory mapped systems, such as the System B, offer powerful capabilities for the assembly language programmer, capabilities not available on other small general purpose computers. SCOPE, Vector's sophisticated screen-oriented program editor is a uniquely efficient tool. With the addition of RAID - the Rapid Interactive Debugger - Vector's memory mapped systems reach a level of sophistication for assembly language software development that rivals true microprocessor development systems.

The program to be debugged is called the "target program". RAID can debug a target consisting of any valid Z-80 operations. The use of RAID allows the programmer to execute the target program instruction-by-instruction, while viewing the effect on the CPU and memory. The status of the CPU and significant parts of memory are displayed constantly on the screen. This process is called "simulation" because RAID takes each instruction out of the target program in order, executes it, and then displays the result of the operation in the status display on the screen. RAID is actually in control of the system, but the user only sees on the screen what would be happening if the target program were running by itself in the system. During simulation The presence of RAID in the system is transparent to the operator, except for the periodic refreshing of the status display on the screen.

Simulation is the heart of RAID. All RAID operations revolve around the use of simulation to debug progams. By seeing the state of the CPU and memory after each instruction is executed, the programmer can find the instruction or routine which causes unexpected effects. There are also RAID commands for changing small sections of program, memory in general, or CPU registers, in order to determine the effect, but simulation is the main purpose for using RAID.

The fast memory-mapped screen used in Vector Graphic systems enables RAID's unique form of full-screen status display. After the execution of each target instruction is simulated, the current state of all registers, alternate register, flags, alternate flags, index registers, the stack, the stack pointer, six selectable blocks of memory, the last two instructions simulated, the instruction about to be simulated, and the four following instructions. The user can re-define the location of each of the six selected blocks of memory at any time, and can also cause each of them to appear in either hex or ASCII representation. The portion of the stack displayed can also be selected.

Further, instructions are disassembled as they are displayed, using Vector's 8080-superset Z-80 mnemonics. The ASCII equivalent of each opcode and operand also appears, in case the codes in fact represent ASCII characters.

The most significant aspect of this simulation process is that the entirety of this information is refreshed on the screen instantaneously (unless supressed) after each instruction's simulated execution. Most of the screen is used, except for the lower 6 lines which are reserved for the target program to display video messages.

This use of the full screen for system status can only be accomplished with memory-mapped video, a Vector Graphic hallmark. It is a powerful departure from the single line status display of development systems using serial terminals. Since the status of most important aspects of the system are on the screen all the time, the user does not have to repeatedly request status on various aspects. This significantly reduces the development time of new software.

The user is given a number of ways to simulate execution of a program under control of RAID: single stepping, slow (one instruction each 1/2 second), fast with full status display refresh, faster with partial status refresh, and faster yet with status refresh occuring only when program simulation passes selected addresses in memory. A program can also be executed "directly", with no simulation, as if RAID were not present. The user can halt the simulated execution at preselected points. Such a halt can take place when the program passes a selected address, when a selected opcode is executed, when a selected register contains a certain value, or when a selected memory location contains a certain byte. These features are useful during the debugging process, particularly when it is necessary to execute rapidly during certain parts of the target program, but to slow down or stop at certain places or under certain conditions in order to isolate an event.

As further aides to debugging, the user can simulate execution of single instructions (not part of a program) to view the affect, can change the value of any location in memory or of any register without writing a program to do so, and can pop words off of or push words onto the stack without writing a program.

The assembly language programmer will find RAID an indispensible tool which dramatically reduces the debugging effort, particularly in comparison with the less sophisticated 8080 debuggers already available for Vector Graphic systems. Debugging more quickly means a program available at an earlier date. In this way RAID significantly increases the value of software written on Vector Graphic systems.

.2 The range of RAID

Upon entering RAID, the user is asked where RAID will be located in this session. RAID can be located anywhere in memory having space for a 6K program.

RAID does not reserve for itself the use of any registers or flags, nor the system scratch pad area. Therefore, the operation of most target programs is not generally limited by the presence of RAID in the system.

Although RAID is loaded under the CP/M operating system, and though it has facilities for loading a target program from a CP/M compatible disk, RAID operates independently of any operating system during the simulation process. Once the user has succeeded in loading both RAID and a target program into system memory, RAID can be used to debug programs running in conjunction with any operating system.

The limitations of RAID are as follows:

It cannot be used to simulate real-time software such as disk drivers and communications programs. However, RAID can simulate those parts of such programs which are not time-dependent, and can execute directly (no simulation) those parts which are time dependent.

Interrupts cannot be simulated because an interrupt will interrupt RAID as well as the target program. However, the interrupt service routine can usually be simulated if it is entered without using a real interrupt.

RAID cannot usually jump from simulation mode, to direct execution, then back to simulation, if during the direct execution portion the stack pointer is moved more than a few locations. This combination of events will cause undesirable effects.

RAID cannot simulate RAID itself.

The Extended Systems Monitor firmware included in all Vector systems must not be removed or replaced, unless the replacement has exactly the same keyboard and video I/O routines. RAID does its console I/O through the Monitor.

1.3 Protocol used in this manual

Underlined material is material you have to type, except where underlining is used for emphasis. The use will be clear from the context.

Material enclosed in angle brackets, such as <file name>, describes what is to be typed rather than giving it literally. Angle brackets sometimes enclose the name of a key to be depressed, such as <TAB>.

Square brackets, such as [N], indicate that the user can optionally type or not type the enclosed material.

II. THE STRUCTURE OF RAID

2.1 Loading RAID

The disk on which RAID comes contains the program which is designed to be used in 56K systems only whose screen memory is located at address F000H.

To load RAID, first get into the CP/M command executive, as indicated by the A> prompt. Load the RAID diskette in drive A and type:

A>RAID [<d:filename.filetype>] <RETURN>

The filename is optional. If typed, it can include a drive designator and a filetype extension, using the standard CP/M format. It causes the selected unambiguous file to be loaded from disk into memory. IMPORTANT: if the filetype is .HEX, then RAID will first convert the file into binary format, and then load it at the load address of the file specified when the file was assembled. This enables you to assemble and debug programs that run at any location, rather than 0100H as is normal for CP/M. If the filetype is anything else, the RAID will assume the file is already in binary format (having passed through the CP/M LOAD procedure) and will load it at 0100H.

After you depress RETURN, you will see the following on the screen...

VECTOR GRAPHIC SIMULATING DEBUGGER VS. x.x

Recommended Location Range is:

3000 to 8000 for a 48K system

3000 to A000 for a 56K system

Debug Module Relocation Address:

Type 4 digits following the colon, giving the desired start address for RAID. It can be any address between the recommend locations, but it must start on an even page boundary, in other words, end in ..00H. Addresses greater than the above are not recommended because this would cause RAID to overlay CP/M, which is not normally desired. However, you may overlay some or all of CP/M if the target program does not use the overlayed part of CP/M. It is not recommended to use the lower part of the TPA (Transient Program Area) because this is the usual area for the target program.

With the exception of the HELP screen, the bulk of the RAID program will load from the specified load address upwards. The HELP screen occupies approximately 1440 bytes from the load address downwards. Thus the simulated program may overlay the help area in memory and, if so, the help screen will not be displayed when the help command is input. If the HELP screen data has been disturbed the phrase "HELP SCREEN NOT AVAILABLE" will be displayed in the

command area.

As soon as you type the four digits, the screen will be erased and replaced with a large banner reading...

RAID

A SCREEN ORIENTED Z-80 SIMULATOR FOR VECTOR GRAPHIC SYSTEMS

Below the banner will be a list of the RAID commands, similar to the list found at the beginning of this manual. This list is a "help screen" which you can view during normal use of RAID by simply inputting a question mark (?) in the command mode. At the bottom of the list, you will see...

PRESS ANY KEY TO CONTINUE

Simply depress the space bar or any other key. As soon as you do, the RAID debugging display will appear. If you specified a program when loading RAID, this program will be loaded at this time. If it does not exist on the disk, then you will see a ? in the command area of the display. You may now proceed to run RAID. At any time, if you want to load a CP/M file from disk, you may use the Load File commands found in Chapter 3.

.2 A RAID session

As mentioned in Perspective, simulation is the heart and purpose of RAID. After RAID is loaded, you may load a target program using the Load command, unless a target program was specified when loading RAID. Then use the Program Counter command (P) to set the value of the PC to the beginning of the target program. Next, you may use one of the execute commands to begin simulation. For example, you can simply depress the TAB key in order to simulate execution of the one instruction pointed to by the PC. More often, you will initiate continuous simulation. A complete list of the Execute command options is found in the "Execute operations" section of Chapter 3. For example, the Execute commands contain a way to cause the status display to be refreshed only when the target program passes pre-selected points during continuous simulation.

The purpose of simulation is to be able to view the status of the system after each instruction or group of instructions is executed. Therefore, the screen contains at all times a continuously updated status display. This is referred to as the "RAID display." The Dump and Stack commands can be used at any time to control what part(s) of memory are displayed in the dump and stack areas of the RAID display.

You will usually watch simulation occuring for a while. You can stop it at any time, in order to change what part(s) of memory are displayed, or to change the value of a register or memory location to see the effect, or to change the value of PC in order to resume simulation at a different point. You can also set a "halt location," a point where simulation will stop automatically when certain conditions are met, such as a register or memory location having a certain value, or the PC reaching a certain value, or a specific opcode being executed. After entering such commands, you can then resume simulation. When simulation is not taking place, RAID is waiting for commands in the command mode.

For those sections of programs that cannot be simulated (see Perspective), you can execute them "directly," that is, actually execute them without simulation. To stop direct execution in mid-stream, you can specify "breakpoint" addresses in advance. When the target program's direct execution reaches one of these addresses, control returns to the RAID command mode, which then waits for another command.

To practice and demonstrate the use of RAID, a demo target program is loaded into the CP/M TPA at the same time RAID is loaded, if no other file is specified while loading. The demo program starts at address 0103H. For example, to watch RAID rapidly simulate this program, first load RAID, then enter the command P 103 <RETURN> followed by XF <RETURN>.

To exit RAID and return to CP/M, execute the Quit command by typing Q <RETURN> on the command line. To return to the Extended Systems Monitor, just depress the ESC key at any time while RAID is in control.

2.3 The RAID display

A drawing of the RAID display is found at the beginning of this manual, for reference purposes. It occupies the top 18 lines of the screen, which are referred to as the "reserved" portion.

This display will always be on the screen during RAID operation, without change. The only exception is if the target program changes the screen directly, or moves the cursor into the reserved part of the screen. If the target program simply displays its output line-by-line according to cursor location, using the normal Extended Systems Monitor video driver, without changing the cursor location, then this output will scroll through the bottom six lines of the screen only. (This is a feature made possible by Vector's Extended Systems Monitor.) In order to make effective use of RAID, it is not wise to debug programs which write directly to the reserved part of the screen.

Whenever there is a change to any part of the system whose status is displayed on the screen, that change will appear immediately on the screen. For example, if the operator pops a word off the stack using the SP command, this word will disappear from the stack display. If the operator enters a value into RAM memory using the EH command, then this value will appear in the dump area of the display, assuming that the appropriate part of memory is represented in the dump area. If during simulation, the target program changes the stack, memory, registers, or flags, the changes will immediately appear. This process is called "refreshing the display."

The only time the display is not refreshed, though changes are taking place in the system, is when the operator specifically requests that it not be refreshed, using some of the various Execute commands (XP, XN, or XD).

The following sections will clarify what the various parts of the RAID display show. Refer to the drawing at the beginning of this manual, or to the screen itself.

.4 Command area

You can type commands in the command area of the display whenever RAID is in the command mode, and the two small white spots are at the left side of the command area, and there is nothing typed in the area. For more information about entering commands, see the beginning of Chapter 3.

2.5 Instruction area

There are seven lines in the instruction area of the screen. Each of these seven lines show one instruction from the target program. The first two lines show the two instructions most recently simulated. They are not necessarily consecutive in memory, because they may be jumps, calls, returns, etc. The third line, the highlighted one, shows the instruction about to be simulated. The last four lines show the four instructions following the current instruction in memory. They may or may not be the next instructions to be simulated.

Each line gives the following information:

(1) The instruction and no, one, or two operands, in hexadecimal notation with no separation between insruction and operands. These are one, two, three or four bytes represented by two, four, six or eight hexadecimal digits.

(2) The ASCII representation of the instruction and operands. Since there is one ASCII character per byte, there are one, two, or three ASCII characters shown. If the character is in reverse video, this means the high-order bit is a one. Control character (those whose lower seven bits are between 00 and 1F) are represented differently depending on the system's character set. Usually, they are represented by graphics characters consisting of from one to six small white squares in various combinations. A chart showing the correspondence between graphics characters and ASCII codes is found at the rear of the Flashwriter II User's Manual which comes with every system.

(3) The address of the instruction, in hexadecimal notation.

(4) The assembly language representation of the instruction followed by its operands. Vector's 8080-superset mnemonics are used.

.6 Dump area:

Six lines are in the dump area of the display. Each line begins with a four-digit hexadecimal memory address, followed by a colon, which indicates the beginning of the memory area shown on that line. Each line consists of sixteen bytes, showing the contents of memory beginning at the address given. The bytes on each line can either be two hexadecimal digits or one ASCII character. When RAID is first loaded, they are all hexadecimal and begin at address 0000H. Each line increases by 10H.

You can change the beginning address of any one line or any group of lines, and you can change the representation to ASCII or back to hexadecimal, by using the Dump commands, described in Chapter 3.

2.7 Flags and registers

Next to the instruction area of the display, there are a group of areas which display the contents of all the Z-80 registers and flags. The left-hand side of this area shows the main flags and registers, the next section shows the alternate flags and registers, and the third section shows the contents of the X register, the Y register, the I register, and the interrupt flag.

The top line shows the flags, indicated by "F" and "F'". On this line, if any flag is set, its symbol will appear, otherwise the symbol will not be visible. The symbols are:

symbol	flag
S	sign
Z	zero
H	half-carry
P	parity
N	add/subtract
C	carry

The contents of each register are shown as a hexadecimal byte next to the register symbol. Further, the contents of the A and A' registers is shown as an ASCII equivalent, between double quotes. Last, the contents of the memory addresses pointed to by the register pairs BC, DE, HL, BC', DE', and HL' are given in parenthesis next to the C, E, L, C', E', and L' register contents respectively.

The contents of the X, Y, and I registers appear in hexadecimal notation directly beneath the register symbols. The status of the interrupt flag appears below the contents of the I register, between parentheses, with (di) indicating interrupts are disabled, and (ei) indicating they are enabled.

.8 Stack pointer and stack

Along the right side of the display is a column with the symbol "SP" at the top, standing for "stack pointer". The contents of the stack pointer appear immediately below, in hexadecimal notation. Below this is found the contents of the stack itself, as much as will fit in the column: 13 hexadecimal words (2 bytes in each word).

RAID keeps track of a value known as the "start of the stack". The start of the stack is the address just above the location of the first word pushed onto the stack. For example, if the stack pointer is pointing at A002H when a program pushes BBBBH onto the stack as the first word on the stack, then the start of the stack is at A002H and BBBBH will be stored at A000H and A001H. (The processor of course decrements the stack pointer by 2 before pushing a word onto the stack, because it always points to the last word pushed onto the stack.)

How does RAID know where the start of the stack is? When RAID is first loaded, RAID assumes the stack is empty. In other words, it assumes the start of the stack is equal to the initial setting of the stack pointer.

The hexadecimal word at the top of the stack column is always the contents of the two locations just below the start of the stack. Using the above example, if the start of the stack is at A002H, then the word stored at A000H and A001H will be the top word in the display. The bottom word in the column is always the contents of the location pointed to by the stack pointer. When the stack pointer decreases by 2, due to popping the stack, the bottom word disappears. When the pointer increases by 2, due to pushing a word onto the stack, the new word is added on the bottom. Hence the display really does show the stack. As words are pushed onto and popped off the stack, the column will grow longer and shorter respectively.

Note that there is a Stack command used for changing the setting of the "start of the stack," which in turn alters what part of memory is displayed in the stack column.

Stack commands are discussed in Chapter 3.

2.9 Halt area

The halt area of the display is used in conjuction with the Halt commands HO, HP, and HM. These commands cause simulation to halt when a pre-selected opcode is executed, when a pre-selected PC value occurs, or when the contents of a pre-selected memory location take on a given value. After you enter one of these commands, the entered opcode, PC value, or memory location and contents will be displayed next to the O:, P:, or M:, respectively. This part of the display simply reminds you of the Halt command(s) you had entered.

In the upper right-hand corner of the halt area is a small square area. When a halt does occur as a result of a Halt setting, the type of halt is indicted in this area. You will see either "HO", "HP", or "HM".

The HO, HP, and HM commands are discussed in Chapter 3.

.10 Refresh area

Below the halt area is an area marked with "Rf:". This is used in conjuction with the XN command. XN causes continuous simulation, but with the display refreshing only when the PC counter passes pre-selected addresses. You can pre-select as many as four such addresses. They will then appear in the refresh area as a reminder. Further, each time a refresh does occur during simulation, the address which caused the refresh will be highlighted within the refresh area of the display.

The XN command is discussed in Chapter 3.

III. RAID COMMANDS

You can type commands in the command area of the display whenever RAID is in the command mode, and the two small white spots are at the left side of the command area, and there is nothing typed in the area.

To clear undesired material from the area, depress the DEL key.

Commands are typed normally, and edited with the BACK SPACE key. To execute a command after typing it in, depress RETURN.

When a command has a parameter following it, a space before the parameter is optional. If there is more than one parameter, then there must be space between each.

When asked to type an <address> or a <word> as a command parameter, leading 0's are optional. An address can have up to four hexadecimal digits. You cannot type an "H" at the end.

When you are asked to type a <byte>, type two hexadecimal digits.

When you are asked to type a <char>, standing for <character>, depress the key or keys that cause an ASCII character to be generated. This can be a single key depression, or the simultaneous depression of the CTRL key and/or the SHIFT key along with a character key.

When you are asked to type a <register or register pair>, you may either specify a register or a register pair. To specify a register, type one of the letters A, B, C, D, E, H, or L. To specify a register pair, type BC, DE, or HL. To specify an alternate register, follow the letter with a single quote mark, such as in A'. To specify an alternate register pair, follow the pair with a single quote mark, as in CD'.

A ? character appears on the right side of the area when RAID cannot understand a command.

To halt simulation at any time and return to the command mode, depress control-shift-brace. In other words, simuiltaneously hold down the CTRL and SHIFT keys while you depress the brace (curley bracket) key.

3.1 Dump operations

Dump ASCII	DA <address> [<line>[-]]
Dump Hexadecimal	DH <address> [<line>[-]]

There are two dump commands. They both control what part(s) of memory are displayed in the dump area of the display. Dump ASCII (DA) causes the specified part of the dump area to appear in ASCII representation, one character per location. Dump Hexadecimal (DH) causes the specified part to appear in hex notation, two digits per location.

In both commands, the <line> is a number between 0 and 5. 0 refers to the top line in the dump area, 1 to the next line, and so on, with 5 as the last line.

In each command, if you type no line number, then the entire dump area will be affected: the contents of the specified address will be displayed in the upper left-hand corner, and the subsequent 96 locations (16 on each line) will be displayed.

If you type a line number, then only that line will be affected. The 16 locations beginning with the specified address will be displayed on that line.

If you type a line number followed by a dash, then that line and all following lines in the dump area will be affected, displaying the locations beginning with the specified address.

.2 Enter operations

Enter Hex	EH <address> <byte> [<address> <byte>...]
Enter ASCII	EA <address> <char> [<address> <char>...]
Enter Register	ER <register or register pair> <byte or word>

These commands change the specified memory location or register. The new register contents will immediately appear on the display. The new memory contents will immediately appear if the address is included in the dump area.

With the Enter Hex and Enter ASCII commands, you may make more than one entry on the same command line. Simply separate each parameter with a space as usual.

With the Enter Register command, you may either specify a register or a register pair. To specify a register, type one of the letters A, B, C, D, E, H, or L. To specify an alternate register, follow the letter with a single quote mark, such as in A'. Then type a 2-digit byte. To specify a register pair, type BC, DE, or HL. To specify an alternate register pair, follow the pair with a single quote mark, as in CD'. Then type a 4-digit word.

Execute operations: See Section 3.11

3.3 Fill

Fill F <start address> <end address> byte

Fills working memory with the byte specified from the start address to the end address.

3.4 Go

Go G <address>

This command transfers control out of RAID and into some other program. The CPU will begin executing automatically at the address specified.

.5 Halt operations

Halt on PC	HP <address>
Halt on Opcode	HO <opcode>
Halt on Register	HR <register or register pair> <byte or word>
Halt on Memory	HM <address> <byte>

One or more of these commands can be active at any given time. Once they are activated, then simulation will automatically halt when the one of the specified conditions occurs. The HP, HO, and HM commands cause the selected values to appear on the display in the halt area. Then, when simulation halts, the cause of the halt appears at the upper right-hand corner of the halt area, as "HP", "HO", or "HM".

In Halt on Register, you may either specify a register or a register pair. To specify a register, type one of the letters A, B, C, D, E, H, or L. To specify an alternate register, follow the letter with a single quote mark, such as in A'. Then type a 2-digit byte. To specify a register pair, type BC, DE, or HL. To specify an alternate register pair, follow the pair with a single quote mark, as in CD'. Then type a 4-digit word.

3.6 Initialize Screen

Initialize Screen I

This command simply clears any extraneous material off the RAID display. All areas of the display will then reflect the current status of the system.

.7 Load file operations

Load Hex File	L <filename>.HEX
Load Executable File	L <filename.filetype>

These commands are used to load CP/M object files from a disk. The drive designater can be included, in standard CP/M format, if the desired file is not in the logged-in drive.

Load Executable File simply loads the file into the TPA at 0100H. It does not execute it. This is an alternative to specifying a filename when loading RAID.

Load Hex File allows you to simultaneously convert an assembled HEX file into executable form and then load it at its own load address in memory, which does not have to be 0100H. The load address would have been determined by an ORG pseudo-op when assembling the file. This is an alternative to using CP/M utilities before loading RAID in order to load the file.

3.8 Set Program Counter

Set PC P <address>

This command simply sets the Program Counter to the desired address. The instruction at this address will immediately appear on the third line of the instruction area of the display. When you begin simulating, the target program will be executed beginning at this address.

3.9 Quit

Quit RAID **Q**

This transfers control back to the program which called RAID in the first place, which is normally the CP/M executive.

3.10 Stack operations

Set Start of Stack SS <address>

The Set Start of Stack operation resets the start of the stack (explained in Section 2.8) to a value other than the value set when RAID was loaded. It will immediately affect the section of memory displayed in the stack area of the display. For example, if you increase the start of stack by 2, one additional word will be displayed at the top of the column. If you decrease it by 2 the top word of the column will be removed.

Note, if you decrease the start of stack value, this will leave extraneous material in the stack display. To clear this out, use the Clear Stack Column command, SC.

Clear Stack Column SC

The Clear Stack Column command only clears extraneous material from the stack display. It leaves the actual stack displayed.

Pop Stack SP

Push Stack SH <word>

Pop Stack and Push stack do what they appear to do. When typing the <word> in the Push Stack command, leading 0's do not have to be typed. The results of these commands will be displayed immediately in the stack area of the display.

.11 Execute operations

Execute Slowly — XS [<rate>]

This causes simulation to begin at the Program Counter, at the rate of approximately one instruction every half second. The status display will be fully refreshed after each instruction.

The optional <rate> parameter can be any hexadecimal number, but is usually between 1 and 10. The higher the value, the slower the rate of simulation.

Execute Fast; Full Refresh — XF [<rate>]

Without the optional <rate> parameter, this causes simulation to begin at the Program Counter, at the maximum possible rate, while still refreshing all parts of the display after each instruction.

If you include the <rate> parameter, simulation will take place even faster, because the display will not be refreshed after each instruction. The higher the <rate> parameter, the faster the simulation. You will be able to notice the difference when the <rate> is in excess of 1000, and even more so when it is above 8000. The <rate> can be any hexadecimal number.

Execute Fast; Partial Refresh — XP [<rate>]

Similar to the XF command, XP causes an even faster rate of simulation, because only the register and stack areas are refreshed when the display is refreshed. Again the <rate> is an optional hexadecimal number. You will begin to see a noticably quicker rate of simulation when <rate> is greater than 80000.

Execute with No Refresh, except as specified — XN [<address> <address>...] (up to 4 addr's)

This results in faster simulation than even the XP command, because no display refresh is done at all, except at specified addresses. You may optionally specify from one to four addresses. When the PC of the target program passes one of these specified addresses, the display will be refreshed. The chosen refresh addresses will be displayed in the refresh area of the display (indicated by the "Rf:"), and when any refresh takes place during simulation, the causative address will be highlighted.

Execute Directly, with specified breakpoints — XD [<address> <address>...] (up to 6 addr's)

This command causes the target program to execute independently of RAID, beginning at the set value of the PC. It is not a simulation, but a real execution, in real time. You may include up to six optional break-point addresses. When execution of the target program reaches one of these addresses, execution will stop and control will return to RAID. This is accomplished by replacing the opcodes at the specified addresses with FF restart codes.

Execute Directly and return — (LF)

This command is executed directly from the keyboard using the (LF) key. It will cause the target program to execute directly, in real time, any called routine. This is particularly useful for disk access or print routines which cannot be simulated in RAID. This command causes the real time execution of the called subroutine to halt at the next memory location past the calling instruction when the subroutine returns. This is of particular value when debugging a program which calls routines which have already been debugged. Because of the limitation of not being able to set break points, this command will not work for routines in PROM.

Execute Instructions — XI <hex instruction> [<hex instruction>...]

This simply simulates execution of the instructions listed on the command line. Any valid Z-80 code and operands may be listed, with spaces to separate them. The effects will be seen in the display.

Execute Next Step — <TAB>

This simulates execution of the instruction pointed to by the PC (the third instruction in the instruction area of the display). The effects will be seen in the display.

.12 Help screen

Help screen ?

This displays the Help screen at any time by simply inputting a question mark when the command mode. If the target program has altered the HELP screen memory area, the phrase "HELP SCREEN NOT AVAILABLE" will appear in the command area. To get back to the Raid screen, hit any key.