High-level Software Analyzers

Model 64330
Model 64340

HEWLETT PACKARD

Technical Data May 1986

Description

Hewlett-Packard High-level Software Analyzers offer HP 64000 system users an advanced, yet easy-to-use, feature set for analysis of programs written in Pascal or C. The analyzers are processor specific, for troubleshooting and debugging software written for the target microprocessor. Measurements are specified and displayed in the high-level context used in generating the software. This simplifies correlations between executing software and written programs, making the analyzers powerful tools for testing and revising software written in high-level languages. Two series of high-level software analyzers are available: HP 64330 High-level Software Analyzers and HP 64340 Real-time High-level Software Analyzers. The HP 64330 requires an HP 64000 emulator and the HP 64302A Emulation Bus Analyzer; the HP 64340 analyzer requires only an HP 64000 emulator.

Both series of analyzers trace program and data flow in executing code. The HP 64330 High-level Software Analyzer adds no extra code to the software under test, but it does stop program execution periodically by inserting software traps to accommodate the analyzer. By contrast, in real-time mode, the HP 64340 Real-time High-level Software Analyzer is fully transparent to the system under test; it meets all criteria for real-time analysis: the processor is not halted, program execution is not stopped, and additional code and traps are not added to the target software.

Hierarchical measurement modes provide a spectrum of perspectives on software operation. First, problems are isolated at a general level. Then specific errors related to statements and variables are pinpointed. Additionally, the HP 64340 analyzer measures the absolute time used by an executing module and counts the number of times specified source statements are executed.

Measurements may be specified in terms of static and dynamic variables; files, programs, procedures, and function names; as well as source code line numbers. A close link to the HP 64000 emulation subsystem adds the further advantages of displaying/modifying variables and controlling execution of a program under test. Directed-syntax softkeys simplify transitions from software analysis to software synthesis, and back again. After identifying a programming error, it takes only a few keystrokes to correct, recompile, relink, run the modified program, and then return to the software analyzer to verify the modification.

Features

- Measurements for global and detailed views of high-level software execution of both C and Pascal programs
- Variable values are displayed in their native data type (Boolean, integer, real, scalar, structured types, etc.)
- Measurements may be specified using static and dynamic variable names; file, procedure, and function names; as well as high-level source line numbers
- HP 64000 emulation subsystems may be controlled from the high-level software analyzers
- Command files speed measurement set-up and execution, facilitating automatic measurements

In addition, the HP 64340 real-time high-level software analyzers also offer

- Tracing of high-level functions, procedures, statements, and variables without breaking—true real-time trace
- Module timing to detect anomalies and analyze performance
- Counts of specified statements to verify software coverage
- Additional measurement parameters to define trigger sequences and a measurement window
- Display any source files without exiting the analyzer
- Time tagging of modules or statements as an elementary performance check
- Interactive operation with other HP 64000 analysis and emulation subsystems

DesignCenter

Measurement Hierarchy

High-level analysis measurements are frequently applied hierarchically. The top-down sequence of measurements is particularly useful when there is little initial information about the cause of a software failure. At a "coarse" or global level, the Trace Modules measurement verifies that procedures and functions are executed in the proper sequence and at the appropriate nesting level. If an incorrect sequence or nesting level is found, the Trace Statements measurement can determine the precise location of a software fault. But, if the modules occur in the correct sequence and level, the Trace Data Flow measurement can point out improper parameter values and global variables passed to and from selected modules.

Assuming module execution sequences and parameter values are correct, the Trace Statements measurement displays program flow in more detail. This measurement, showing executed source lines and values of global and local variables referenced, allows designers to distinguish between errors caused by programming flaws and those due to unexpected variable values. Then, a Trace Variables measurement can be applied to isolate the causes of improper variable assignments.

Once the software is executing properly, the Count Statements and Time Module measurements on the HP 64340 analyzer allow coverage testing and performance analysis of the software modules. Invaluable for quality testing of software, the Count Statements measurement shows the number of times a source statement or range of source statements are executed. The Time Module measurement measures up to four modules for real-time execution speed, pinpointing bottlenecks that may require recoding.

The flowchart in figure 1 represents a typical scenario for the application of this hierarchical measurement structure.

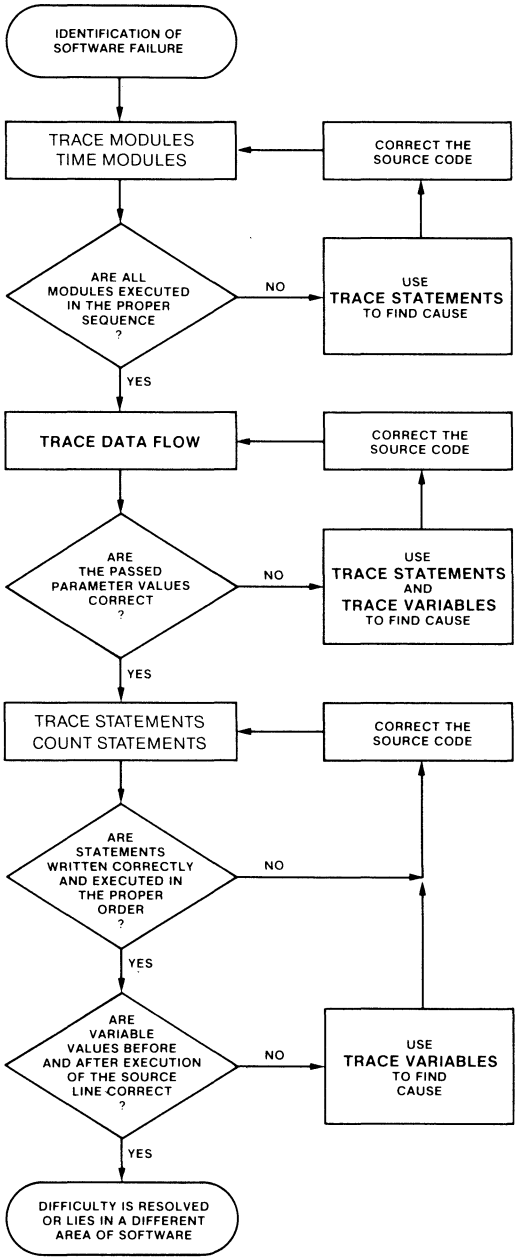

Figure 1. *A logical and well-ordered debugging process is possible due to the hierarchical arrangement of the measurements of the high-level software analyzers.*

Trace Modules

Trace Modules measurement monitors program execution at the procedure level. It provides an overview of total system activity, capturing the sequence in which software modules are executed. By rapidly isolating software failures, this capability can quickly verify that procedures are called in the proper sequence, before undertaking the more time-consuming task of debugging at the statement level. For instance, a subroutine multiplying two floating-point numbers located on a stack fails unless the operands have been pushed onto the stack. What might appear to be a failure of the multiply subroutine itself may actually result from improperly calling the multiply routine before calling the subroutine that transfers the operands from CPU registers onto the stack. A procedural-level view greatly reduces the time necessary to identify and correct these kinds of problems.

Indentation (figure 2) represents the nesting level of the monitored procedures. Adding further information, the source statement causing a procedure to be executed is displayed, as well as the procedure end point. A designer can determine immediately which procedures are functioning as subroutines to main procedures, and can verify the order and nesting levels of execution. The number of calls to a subroutine by the main program may also be determined quickly. Recursive procedures and functions are traced.

The Trace Modules measurement is performed in real time using the HP 64340 analyzer, tracing up to 288 modules. The HP 64330 analyzer can trace a maximum of 128 modules using software breakpoints to detect entry and exit points.

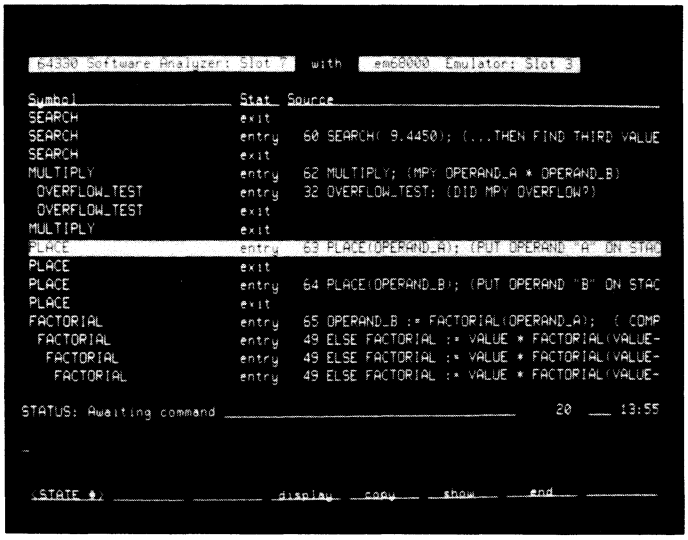

Figure 2. *A Trace Modules measurement indicates that the MULTIPLY subroutine was called before the operands were placed on the stack by procedure PLACE.*

Trace Data Flow

Trace Data Flow measurement monitors the flow of data to and from program subroutines and functions. The measurement allows the designer to determine the values of parameters and variables on entry to and/or exit from selected procedures. Verifying proper parameter and variable values is extremely important in isolating software failures.

A procedure that reads a file from disc may require several parameters: the starting sector number, load address, and number of disc sectors to be read. If any parameter is incorrect, the file cannot be read correctly, even if the file-reading subroutine is functioning properly. The Trace Data Flow measurement provides a means to quickly verify the values of the parameters.

Parameter variable values on entry and values of data elements on exit from a procedure are equally important in trouble-shooting. A "search" procedure that accepts an input value, then outputs the number of the first table containing that value, is one such case. Here, the correct input value is necessary to the proper functioning of the search procedure itself. A correct table value is necessary to the continued operation of the remainder of the program.

If, as in figure 3, the data flow measurement indicates the input value to be incorrect, it can be concluded that something is wrong prior to evoking the search routine. If the input value is correct, but the table number is wrong, the search routine itself may be at fault. The search routine is likely to be operating as intended if the input value and table number are correct, suggesting that the problem occurred after the execution of the search routine. Several data flow measurements, with similar conclusions drawn from each, greatly enhance engineering productivity by rapidly eliminating false leads in tracking down software malfunctions.

Both the HP 64330 and 64340 analyzers perform the Trace Data Flow measurements using breakpoints. The HP 64330 analyzer uses software breakpoints, and the HP 64340 analyzer uses hardware (emulator) breaks.

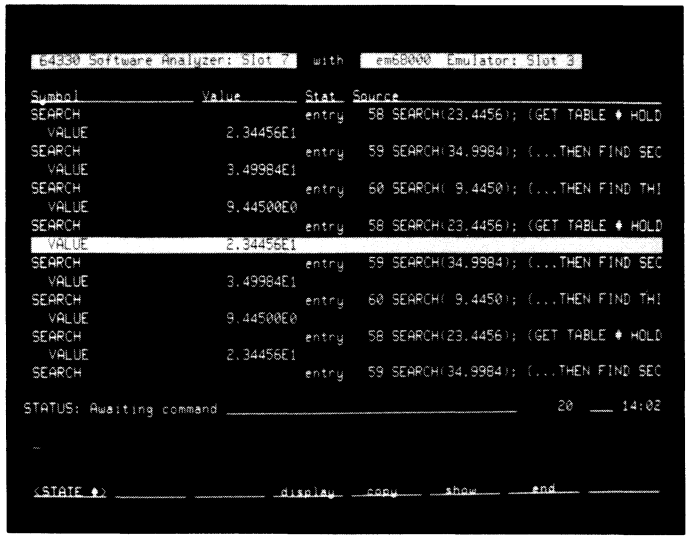

Figure 3. *The Trace Data Flow measurement monitors the input value (VALUE) on entry to the SEARCH routine.*

Trace Statements

As software debugging focuses on particular areas of difficulty, a Trace Statements measurement displays program execution in the same high-level language in which it was written. A measurement may be set up to trace the execution of a range of high-level source lines, or all statements within a procedure, function, or main program. High-level statement tracing (figure 4) is a very important feature for efficiently analyzing programs written in high-level languages. These capabilities sharply reduce software debugging time by eliminating the need to translate from assembly-level listings of high-level programs.

Displays include not only the sequence of executed high-level statements, but also the values of all variables referenced by these statements. As with the Data Flow measurement, the Trace Statements measurement helps to determine whether incorrect variable values are causing erroneous software activity. Alternatively, the conclusion may be that a programming or conceptual error within a statement is responsible. A designer can trace a complex section of code, view the executed statements, and watch variable values as they are changed from statement to statement. In this way, a software error can be located at the statement level. It is also possible to determine that a malfunction is not a result of the code as written. If input and output variables are valid, it is a basis for selecting the next code segment to be examined.

Trace Statements measurement also determines whether a statement actually does what was intended by the programmer. In many cases, a statement may assign a value to a variable, based on the values of many other variables and mathematical or logical operations. To verify that such statements perform the correct operation, the High-level Software Analyzers display the values of all variables read from and written to by each executed source line.

The Trace Statements measurement in the HP 64340 analyzer is executed in nonreal time when both local and global variables are captured. If a real-time mode is used, only the global variables are displayed. The HP 64330 does not have a real-time mode for Trace Statements; both local and global variables are displayed. An unlimited number of recursive calls may be traced with the HP 64340 analyzer, while the HP 64330 analyzer traces a maximum of 128 recursive calls.

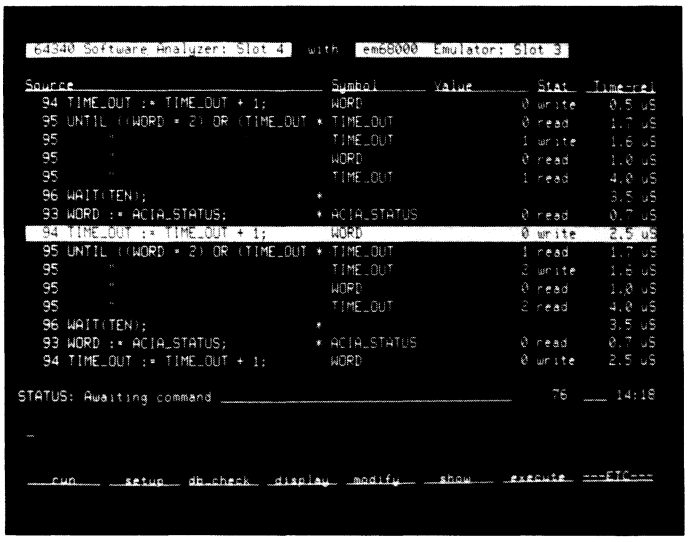

Figure 4. *The Trace Statements measurement shows executed source lines along with the values of all variables referenced.*

Trace Variables

Incorrect variable values are responsible for many software malfunctions. Knowing that a variable is incorrect is useful, but knowing how and where the variable was improperly set further simplifies error isolation. The Trace Variables measurement provides these capabilities in the high-level software analyzers.

With a Trace Data Flow measurement, it is possible to determine that a subroutine failure is due to an incorrect parameter value being passed to that routine. A follow-up Trace Variables measurement would then identify the source line assigning the spurious value. Subsequently, it may also be determined that the parameter was set incorrectly, not due to the logic of a particular statement, but rather, as a result of faulty variable values upon which the parameter depends.

Trace Variables measurement traces both static and dynamic variables. Analysis may be defined to trace variables only when read, written, or both. When a selected data element is accessed, results are displayed with file and procedure names, along with the line number and the high-level source statement responsible for the access (figure 5). The variable name, file and procedure names, and type of access (read or write) are indicated. Values are displayed in the appropriate type (i.e., Boolean, integer, real, etc.). With the HP 64340 analyzer, this measurement can be made in real time.

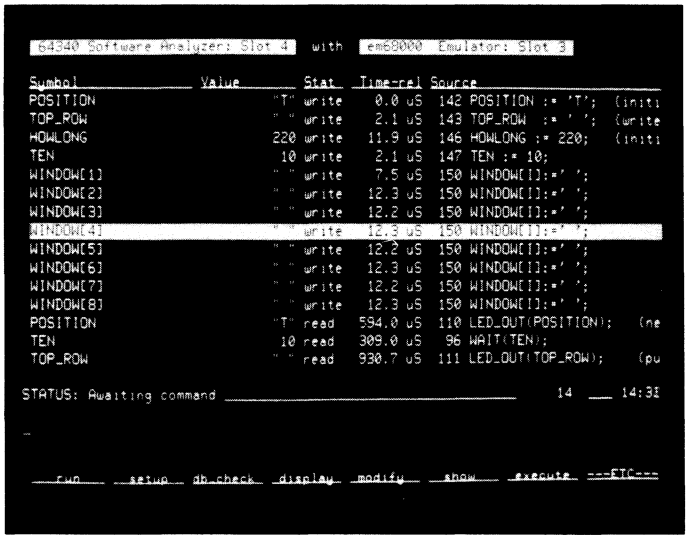

Figure 5. *This Trace Variables measurement, showing both variable assignments and reads, points out source statements responsible for the variable assignments.*

Count Statements (HP 64340 only)

Software coverage testing is an integral part of an overall software testing environment. Knowing the number of times particular source statements are executed, or whether a conditional branch is taken, yields valuable information regarding how extensively a software module has been tested. The Count Statements measurement is an additional real-time capability of the HP 64340 Real-time High-level Software Analyzer.

With the Count Statements measurement, it is possible to determine the number of times, if any, that a selected line of source code is executed. Using this data, a designer can evaluate the degree of testing done on a software module, and whether a different test approach is needed for greater testing coverage.

The software engineer may specify a range of up to 255 source statements to be counted within a single software module. The Count Statements measurement is made on software executing at standard operational speeds. Displays show the source lines specified along with the number of times each source line was executed (figure 6).

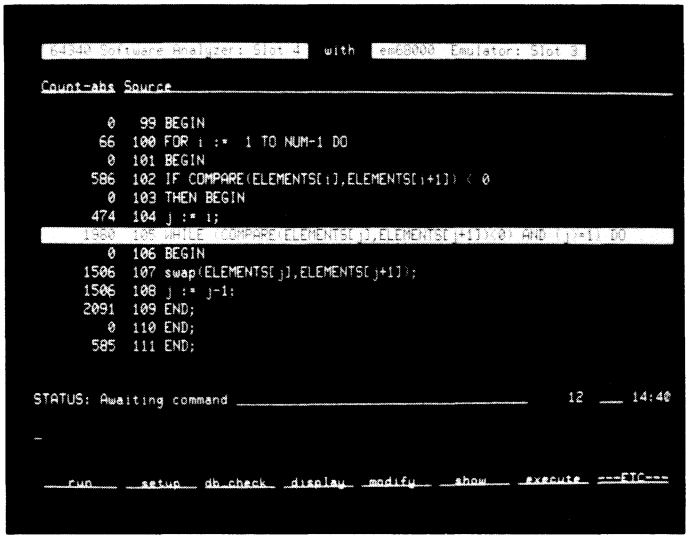

Figure 6. *A quick method of checking software coverage is using the Count Statements measurement (HP 64340) to see how many times, if any, selected lines of source code are executed.*

Time Modules (HP 64340 only)

Logical and functional software debugging does not always insure bug-free, operational code. Many times, software that is otherwise satisfactory fails to perform properly under real-time, full-load conditions. The Time Modules measurement shows the time required for software modules to execute in real time, at full speed.

Using the Time Modules measurement aids software engineers in finding software bottlenecks and bugs that are due to slow execution. Another application is checking a software module that generates a programmable time delay. If the delay needs adjustment, a "modify variable" command can be used to change the delay variable.

The Time Modules measurement times up to four modules simultaneously. Both single and multiple measurements may be made. If a module occurs more than once during a measurement, the statistics (minimum, maximum, mean, and number of occurrences) are displayed automatically (figure 7).

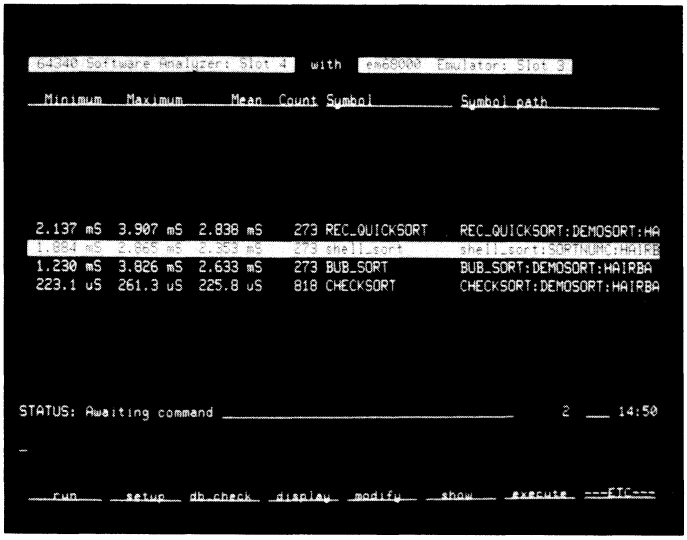

Figure 7. *When monitoring software performance under real-time full-load conditions, the Time Modules measurement (HP 64340) shows the execution time of up to four modules, with minimum, maximum, and mean times and the number of module occurrences.*

Hardware Break (HP 64340 only)

A hardware break can be set in the executing program, in either ROM or RAM. The break can be controlled by the enable/disable sequence, assuring that the break occurs in the proper context. Once the break occurs, you can specify the next measurement. Unlike the software breakpoints used in the HP 64330 analyzer, the hardware break causes a full break to the monitor, and the preceding measurement will not be executed again. In addition, the break vector can be directed to branch the program to a user routine and continue real-time execution, without returning control to the emulator.

Trace Qualification (HP 64340 only)

The HP 64340 Real-time High-level Software Analyzer provides software engineers with powerful trace qualification resources. Sequencing, windowing, and measurement enable/disable features are used to restrict analysis to the specific software area of interest.

Tracing a global variable within a very small section of code is a difficult task, under typical conditions. But, the measurement enable/disable qualifiers can define a context for initiating a measurement; enable/disable conditions are implemented in hardware. Windows create a repetitive enable/disable function, and the analyzer only captures data when the window conditions are true. Up to six levels of sequencing can be defined to establish a series of events that must be found in the specified order before beginning a measurement or defining an enable or disable condition.

Emulation Control Functions

Analyzer functions are available for controlling program execution in the emulation subsystem (figure 8). The load function loads programs into the emulator for execution. This is particularly convenient when analyzing two versions of the same software. For example, measurements may be made on software version A, then version B may be loaded and executed, and the same set of measurements made. Comparisons can be made quickly, to verify functional equivalence, point out critical differences, and identify any advantages of one version over the other.

The emulator can be set up to stop execution of the user program when a breakpoint occurs. Breakpoints are established using executable addresses (HP 64330), source line numbers, or module entry/exit points. In some applications, errant software execution might destroy data that is difficult or impossible to recreate, such as a disc directory. In this case, a breakpoint set at the beginning of the disc write routine could stop program execution before valuable information is lost.

After a breakpoint is encountered, it is often necessary to determine values of selected variables to decide whether it is "safe" to continue program execution. For example, incorrect variable values in a disc write routine could overwrite an important area of the disc. The high-level software analyzers allow the user to display variables, and a modify function can be used to change any improper variables, without returning to the editor to change source code, and then recompile and relink.

Another control function, run, initiates program execution in the emulator from the starting point, a specified address, a selected line number, a module entry point, or from the current value of the processor's program counter. Sections of code (that may require a long time to execute) may be bypassed or started at a point other than the normal starting point. In conjunction with breakpoints, the run function allows the software engineer to simulate interrupts or a task switch occurring virtually anywhere within a program. The run control can also be used to synchronize the analyzer with the initial program start-up or a restart after a breakpoint.

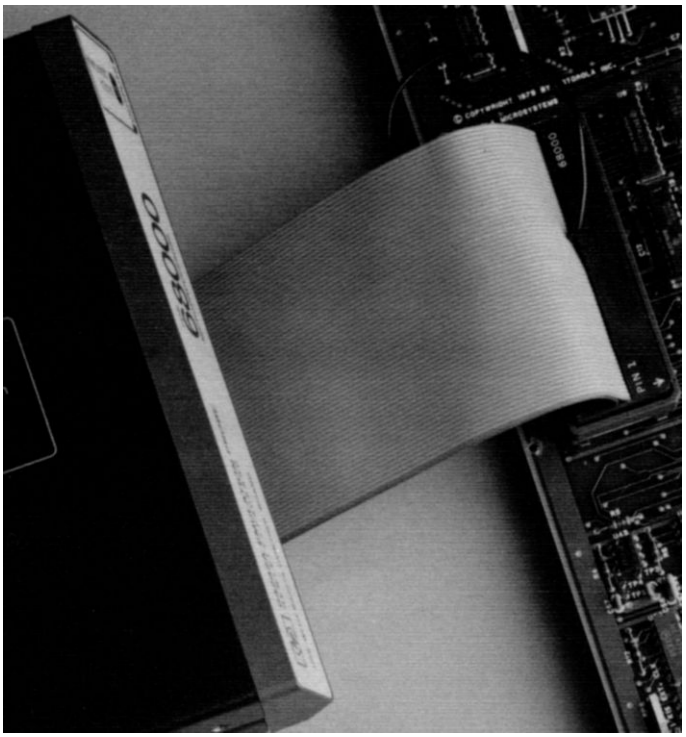

Figure 8. *Both high-level software analyzers are operated with an appropriate emulator. A major advantage of this configuration is that program execution can be controlled from the analyzers through the emulator controls.*

Comparison Chart
High-level Software Analyzers

Feature	HP 64330	HP 64340
Trace Modules	sw breakpoint	real time
Trace Data Flow	sw breakpoint	hw breakpoint
Trace Statements	sw breakpoint	real time and hw breakpoint*
Trace Variable	sw breakpoint	real time
Count Statements	no	real time
Time Modules	no	real time
Time/Count Tagging	no	yes
Emulator Control	yes	yes
Trace Qualification	no	yes
Display of Source File	no	yes
Display/Modify Variables	yes	yes
IMB Cross-triggering/Arming	no	yes
Change Variable Base	no	yes
Development Station Slots Used	0	3
Breakpoints	16 sw	9 hw

*Measurement made in real time for static data, and with hardware breakpoints when static and local data are included.

Characteristics, HP 64330

Measurement Modes

Trace Modules: displays the sequence of procedure calls and execution nesting levels; max of modules traced, 128.
Trace Data Flow: displays values of selected parameters into or out of specified modules; max of 10 module and variable names.
Trace Statements: displays high-level source lines with comment fields and values of variables referenced; may include all statements from a single module or range of source lines.
Trace Variables: displays source lines that assign and/or read specified variables; max of variable names, 10.

A single variable name will include many variables if the name represents an array, structure, or record.

Tracing

Trace depth: from 1 to 8 kbytes.
Breakpoints: max, 16 software breakpoints.
Language: Pascal and C.

Accessories Supplied

Model 64330 High-level Software Analyzer: Each high-level software analyzer consists of operating software on flexible disc and an operating manual.

Accessories Required

Model 64302A 48-channel Emulation Bus Analyzer: (required with HP 64330 analyzers; ordered separately) Each 48-channel emulation bus analyzer consists of a control card. Service manual is included. For operating characteristics, refer to emulator operating manual supplied with emulation subsystem. An HP 64032A Memory Extender Board must be added when the analyzer is installed in an HP 64100A Development Station with serial number prefix below 2309A.

Characteristics, HP 64340

Measurement Modes

Trace Modules: displays the sequence of procedure calls and execution nesting levels; max of modules traced, 288 in a single file; guaranteed minimum of 128 modules using up to 10 symbols (procedures, functions, or files) that lie within four contiguous ranges, with the enable/disable sequence off.
Trace Data Flow: displays values of selected parameters into or out of specified modules; max of 10 symbols (procedures, functions, variable names) for up to 3 modules.
Trace Statements: displays high-level source lines with comment fields and values of variables referenced; may include all statements from a single module or range of source lines within a module. Real-time measurement traces only static variables.
Trace Variables: displays source lines that write and/or read specified variables; in real-time mode, up to 10 dynamic variables in a single module, up to 9 nonadjacent static variables, or up to 10 static variables in a single module.
Static variables have a fixed location in memory.
Dynamic variables are stack-based variables.
Count Statements: displays a range of source statements along with the number of times each statement was executed; real-time measurement; up to 255 statements residing within a 4k address range and within a single module.
Time Modules: displays time a module uses in a complete execution; up to 4 modules may be timed simultaneously; max number of executed nesting levels, 255.

A single variable name will include many variables if the name represents an array, structure, or record. A C variable whose declaration is between statements will not be traced; variable declaration must be at the beginning of the function.

Tracing

Memory Size: 4k deep by 96 bits wide.
Sequences: up to six levels.
Windowing: one window with recursive activation; not a real-time measurement.
Enable/Disable: initiates/terminates analyzer execution; either or both conditions may be set by defining single events, or one of the conditions may be qualified with up to 9 ORed terms. Using enable/disable condition reduces the number of ranges available for tracing variables and modules.
Breakpoints: max, 9 hardware breakpoints.
Language: Pascal and C.

Electrical

Power consumption: typical, 12.4 A at +5 Vdc and 1.2 A at −5.2 Vdc; power supplied by development station.

Intermodule Bus

HP 64340 both drives and receives the trigger-enable signal across the Intermodule Bus (IMB).

Accessories Supplied

Model 64340 Real-time High-level Software Analyzer: Each real-time high-level software analyzer consists of three circuit cards, microprocessor-specific operating software on flexible disc (HP 64341XA), and an operators manual. An HP 64032A Memory Extender Board must be added when the analyzer is installed in an HP 64100A with serial number prefix below 2309A.

Ordering Information

HP 64330 High-level Software Analyzers

Model	Description
64331A	**High-level Software Analyzer for 68000 microprocessors** (use with HP 64242S 68000 Emulator)
64331B	**High-level Software Analyzer for 68000 microprocessors** (use with HP 64243AA/AB 68000 Emulators)
64332A	**High-level Software Analyzer for 8086 microprocessors** (use with HP 64222S 8086 Emulator)
64332B	**High-level Software Analyzer for 8086 and 80C86 microprocessors** (use with HP 64220S 8086/8087 Emulator or HP 64220S Opt 001 80C86 Emulator)
64333A	**High-level Software Analyzer for 8088 microprocessors** (use with HP 64226S 8088 Emulator)
64333B	**High-level Software Analyzer for 8088 and 80C88 microprocessors** (use with HP 64221S 8088/8087 Emulator or HP 64221S Opt 001 80C88 Emulator)
64334A	**High-level Software Analyzer for 68010 microprocessors** (use with HP 64249S 68010 Emulator)
64334B	**High-level Software Analyzer for 68010 microprocessors** (use with HP 64245AA/AB 68010 Emulators)
64335A	**High-level Software Analyzer for 80186 microprocessors** (use with HP 64224S 80186 Emulator)
64336A	**High-level Software Analyzer for 80188 microprocessors** (use with HP 64225S 80188 Emulator)
64337A	**High-level Software Analyzer for 68008 microprocessors** (use with HP 64244AA 68008 Emulator)
64338A	**High-level Software Analyzer for NEC 70116 microprocessors** (use with HP 64294S 70116 Emulator)
64339A	**High-level Software Analyzer for NEC 70108 microprocessors** (use with HP 64295S 70108 Emulator)

HP 64340 Real-time High-level Software Analyzers

Model	Description
64340A	**Real-time High-level Software Analyzer** (required software is ordered separately)
64341AA	**Real-time High-level Software Analyzer software for 8086 and 80C86 microprocessors** (use with HP 64220S 8086/8087 Emulator or HP 64220S Opt 001 80C86 Emulator)
64341BA	**Real-time High-level Software Analyzer software for 68000 microprocessors** (use with HP 64242S 68000 Emulator)
64341CA	**Real-time High-level Software Analyzer software for 8088 and 80C88 microprocessors** (use with HP 64221S 8088/8087 Emulator or HP 64221S Opt 001 80C88 Emulator)
64341DA	**Real-time High-level Software Analyzer software for 68010 microprocessors** (use with HP 64249S 68010 Emulator)
64341EA	**Real-time High-level Software Analyzer software for 80186 microprocessors** (use with HP 64224S 80186 Emulator)
64341FA	**Real-time High-level Software Analyzer software for 80188 microprocessors** (use with HP 64225S 80188 Emulator)
64341GA	**Real-time High-level Software Analyzer software for 68000 microprocessors** (use with HP 64243AA/AB 68000 Emulators)
64341IA	**Real-time High-level Software Analyzer software for 68010 microprocessors** (use with HP 64245AA/AB 68010 Emulators)
64342AA	**Real-time High-level Software Analyzer software for NEC 70116 microprocessors** (use with HP 64294S 70116 Emulator)
64342BA	**Real-time High-level Software Analyzer software for NEC 70108 microprocessors** (use with HP 64295S 70108 Emulator)

Accessories

64032A **Memory Expansion Module**
(Required for HP 64100A Development Station with serial number prefix below 2309A; occupies one slot.)

64960A **2-position Emulation Bus Cable** (two required)
Opt 010 through Opt 017 cables are special cables for installing the HP 64340A Real-time High-level Software Analyzer. Refer to the HP 64000 Configuration Guide to select the correct option for the combination of installed subsystems.

Opt 010	Emulation Bus Cable (1 plus 1)
Opt 011	Emulation Bus Cable (2 plus 1)
Opt 012	Emulation Bus Cable (3 plus 1)
Opt 013	Emulation Bus Cable (1 plus 2)
Opt 014	Emulation Bus Cable (2 plus 2)
Opt 015	Emulation Bus Cable (2 plus 3)
Opt 016	Emulation Bus Cable (1 plus 3)
Opt 017	Emulation Bus Cable (2 plus 3)

HP 6433XA High-level Software Analyzer must be used with an HP 64100A or 64110A Development Station with an HP 64000 Emulation subsystem that includes an HP 64302A Emulation Bus Analyzer. HP 64340A Real-time High-level Software Analyzer must be used with an HP 64100 or 64110A Development Station with an HP 64000 Emulation subsystem and the appropriate HP 64341XA software package. Both high-level software analyzers require an appropriate HP 64000 compiler.

Note: A linear address space is assumed by both High-level Software Analyzer series. Limited analysis of memory swapping systems can be accomplished using the sequencer capabilities of the HP 64340A.

Printed in U.S.A.
5953-9297

Data subject to change.

HEWLETT PACKARD

For more information, call your local HP Sales Office or nearest Regional Office: Eastern (301) 258-2000; Midwestern (312) 255-9800; Southern (404) 955-1550; Western (818) 509-2319; Canadian (416) 678-9430. Ask the operator for Instrument Sales. Or, write: Hewlett-Packard, 1501 Page Mill Road, Palo Alto, CA 94304. In Europe: Hewlett-Packard S.A., 7, rue du Bois-du-Lan, P.O. Box CH-1217 Meyrin 2, Geneva, Switzerland. In Japan: Yokogawa-Hewlett-Packard Ltd., 29-21, Takaido-Higashi 3-chome, Suginami-ku, Tokyo, 168.