DECEMBER 1975

HEWLETT-PACKARD JOURNAL

A 100-MHz Analog Oscilloscope for Digital Measurements

A new general-purpose oscilloscope has features such as dual-channel magnification and third-channel trigger display that enhance its versatility, particularly with respect to measurements in digital systems.

by Allan I. Best

DESPITE CONTINUING ADVANCES in circuit speeds, a great amount of digital design work continues to be carried on at clock rates below 100 MHz.

This is particularly true for digital designs involving TTL and CMOS devices where clock rates below 50 MHz predominate. Hence, the growing need in digital test instrumentation is not so much for the ability to work at the highest possible speed as it is for means of coping with the complexity of digital circuit operation, a need that is becoming more and more acute as the applications of microprocessors continue to expand.

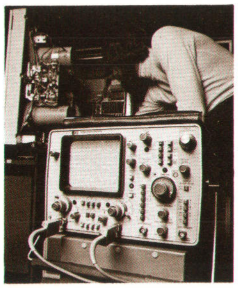
In assessing the oscilloscope needs of digital designers, it became clear that the requirements of a large segment of users could be met by a general-purpose, dual-channel scope that had a bandwidth of 100 MHz, a wide range of vertical deflection factors, a bright CRT capable of finely-drawn traces, a precision time base, sensitive, stable triggering and a delaying sweep with low inherent jitter that would enable timing measurements with less than 1% error.

In particular, for the debugging and field maintenance of digital systems—especially those based on microprocessors— it was expected that users would want to team such a scope with a logic state analyzer,[1] the logic state analyzer to locate problems in the data domain and the oscilloscope to work in the time domain finding the electrical malfunctions that cause the data-domain problems (see box, page 5).

A Well Fitted Oscilloscope

It was with this background in mind that a new oscilloscope was developed. The primary goal was to provide lab-quality performance and versatility in an easily-maintained instrument at an economical price. The result is shown in Fig. 1.

Although this instrument, the HP Model 1740A, has the compactness, ruggedness, and ease of maintenance required of an instrument for the field, it has all the attributes of a high-quality, dual-channel, 100-MHz laboratory oscilloscope. It is well suited for digital work with its bright CRT display, precision time bases, stable triggering, and a third channel that enables the timing of an external sweep trigger signal



Cover: *Display of waveforms fulfills an important function in the world of 1's and 0's just as it always has in the world of analog signals. The oscilloscope pictured here displays waveforms in the traditional manner but it can also be adapted to display 1's and 0's in a data format, as explained in the article beginning on this page.*

In this Issue:

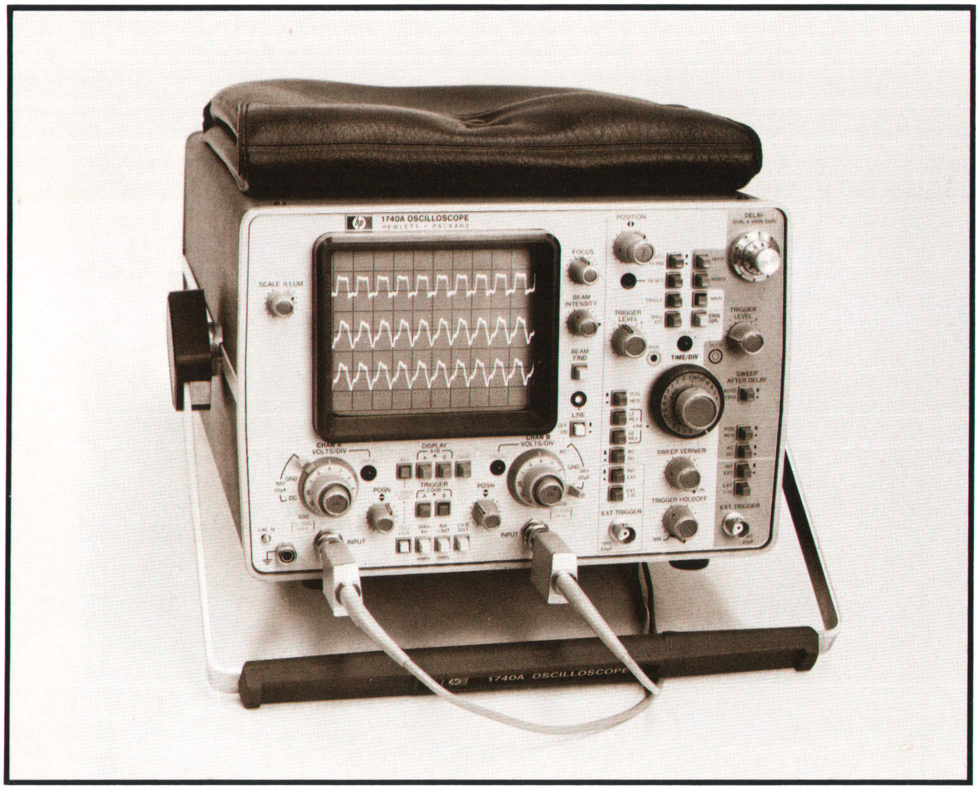


Fig. 1. *Model 1740A Oscilloscope can display the sweep trigger waveform as a third trace simultaneously with the waveforms in channels A and B. The new oscilloscope has dc-to-100-MHz bandwidth, selectable input impedance (high impedance or a well-matched 50Ω), precision delayed sweep, a bright, finely-focussed trace, and all the other characteristics of a high-quality laboratory oscilloscope.*

to be compared to the signals in both vertical channels. Of particular interest to digital designers is an option that enables the new scope to serve as the digital display for a logic state analyzer, with instant pushbutton restoration of the analog display whenever desired (Fig. 2).

×5 Vertical Magnifier

In earlier high-frequency oscilloscopes, increased sensitivity at reduced bandwidth in the vertical channel was obtained by cascading the two vertical channels into one channel, thus sacrificing the dual-channel display. In the new Model 1740A Oscilloscope, the ×5 magnifier operates on both vertical channels, increasing the sensitivity from a vertical deflection factor of 5mV/cm to 1 mV/cm, at a bandwidth of 40 MHz, while retaining dual-channel operation. In this mode the two channels may display signals separately in either the alternate or chopped display mode, or they may be combined (A−B) for single-channel display of differential signals.

Trigger Display

When two signals are being displayed on the Model 1740A in either the alternate or chopped modes, pressing the TRIG VIEW pushbutton adds a third trace, giving a three-channel display (Fig. 2b). The third trace displays the sweep trigger signal, thus enabling the user to make timing measurements between an external trigger signal and the signals in channels A and B. The propagation delay through the trigger-view channel matches those of the vertical channels within 2.5 ns ± 1 ns, thus assuring integrity in timing comparisons between the external trigger signal and the signals in channels A and B.

In effect, the trigger-view mode provides a third, 80-MHz channel for viewing a signal applied at the trigger input. The deflection factor is nominally 100 mV/div, compatible with ECL logic levels, or 1 V/div with the ×10 attenuator, compatible with TTL and CMOS levels. These are changed to 20 mV/div and 200 mV/div when the ×5 magnifier is used.

When the sweep is triggered by the signal in channel A or B, the trigger-view channel displays the same signal with approximately the same deflection factor and, as with an external trigger, it may be positioned by the TRIGGER LEVEL control to show the triggering point. The dc levels of the trigger amplifier are set so the sweep trigger level corresponds to the center horizontal graticule line on the CRT, thus the operator can see which point on the waveform initiates the sweep.

The displayed waveform is also processed through the trigger input filtering (HF REJECT, LF REJECT) so the operator sees the waveform exactly as the trigger-recognition circuit sees it. The trigger-level control functions like a positioning control, displacing the waveform vertically so the operator can choose a trigger level that avoids the likelihood of triggering on noise or other waveform anomalies

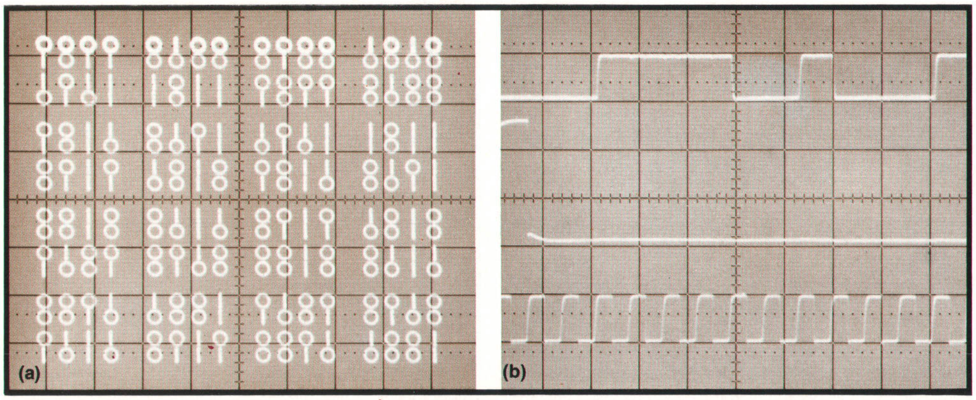


Fig. 2. *When equipped with the logic-state option Model 1740A can display either the data domain outputs of a logic state analyzer (a) or time-domain waveforms applied through its own inputs (b). The upper trace in (b) is the digital waveform corresponding to the right-hand column of the table display in (a), delayed one clock period. The scope is in the* TRIG VIEW *mode, displaying the logic state analyzer's trigger output on the middle trace.*

Design Approach

Although the design of the new oscilloscope covered ground already traversed by other HP oscilloscopes with respect to performance, it was decided not to retain elements of earlier designs if advancing technology made it possible to improve the design with respect to maintainability, reliability, and manufacturing cost.

One element of earlier designs that was retained, however, was the cathode-ray tube. This is the same tube used in the 180-series oscilloscopes.[2] Noted for its small spot size and bright traces, it has the writing speed needed for displaying low rep-rate, fast transitions at the 5-ns/cm sweep speed. An advanced design to begin with, it has been improved over the years and in the course of building 40,000 or so, HP production engineers have refined the manufacturing process such that a long, trouble-free life can be expected from these CRT's.

The highly-integrated vertical amplifier, on the other hand, is entirely new and contributes to the stable performance and manufacturability of the new oscilloscope. It is discussed in detail in the article beginning on page 8.

Horizontal System

Another technique retained from earlier designs is the trigger recognition circuits. For both the main and delayed sweeps, the new scope uses the same HP monolithic integrated circuits as the 275-MHz Model 1720A Oscilloscope.[3] They provide stable triggering on signals above 100 MHz but require an amplitude equivalent to only 1 cm of deflection at 100 MHz to do so.

A variety of trigger modes gives the flexibility needed for lab applications. The new scope triggers repetitively or singly on externally supplied or internal signals. Trigger slope and amplitude are selectable. The trigger input coupling can be dc or ac and it can be filtered to remove noise above 4 kHz (HF REJECT) or remove powerline and other interference below 4 kHz (LF REJECT).

The main sweep circuit has controllable trigger holdoff time, as used for the past eight years on HP high-performance scopes. This inhibits triggering for a selected time interval after a sweep terminates and is useful when examining complicated waveforms that have more than one trigger point.

The sweep circuits use the familiar Miller integrator. The well-regulated supply voltages and high-gain amplifiers for the integrators assure sweep accuracy well within 2% on the fast sweeps (3% with the horizontal ×10 magnifier). A full complement of sweep modes is provided, including main sweep, main intensified, calibrated delayed sweep, and calibrated mixed sweep.

The comparator that selects the point on the main sweep where the delayed sweep is to start has a stable trigger level such that delay jitter is less than 0.002% of the maximum delay on each range. This plus the precision 10-turn delay control and the sweep accuracy enables time intervals to be measured by the differential time measurement technique* over most of the range with an accuracy of ±(0.5% + 0.1% of full scale).

The new scope also has an A versus B mode for high-speed X-Y plotting. In this mode, the A channel signal drives the CRT in the vertical direction and the signal in the B channel drives it in the horizontal direction. The bandwidth of the horizontal channel in this case is 5 MHz. The A versus B capability is replaced by the logic-state option, however, when that option is installed.

The Logic-State Option

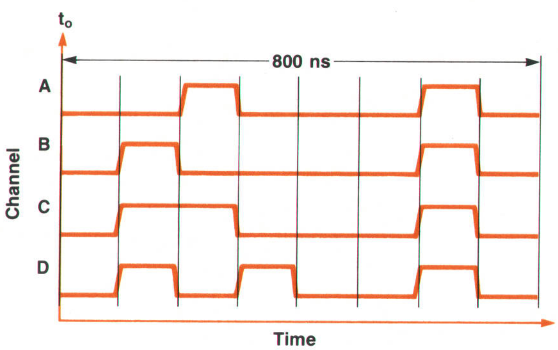
When equipped with the logic-state option, the Model 1740A can work with the Model 1607A Logic State Analyzer[1] to provide a measurement tool of singular usefulness for the digital designer and troubleshooter (Fig. 3). This option equips the 1740A with internal switching and rear-panel inputs for the logic state analyzer outputs. A front-panel pushbutton enables the user to switch back and forth between the

*To make this measurement, the delayed sweep is used and the "start" point is positioned at center screen with the delay control. The delay setting is noted and the "stop" point is then positioned at center screen, again with the delay control. The difference between the new delay-control setting and the previous one is the time interval between start and stop points.

Working in the Data Domain—Logic State Analyzers and Oscilloscopes

The on-going diffusion of digital techniques into all branches of electronic design has radically changed the nature of many—if not most—design, production test, and field maintenance tasks. Electronic engineers, who have all learned to use the underlying mathematics and analytical instrumentation for designing in the frequency and time domains, must now become familiar with the data domain.

A simple example will illustrate what one faces when dealing with the data domain. The drawing shows four waveforms. Whether we think of these as being generated in a series of combinatorial logic gates or from instructions in a microprocessor or a computer is irrelevant—simultaneous waveforms like these occur on a one-shot basis throughout all digital equipment, usually on a much grander scale, from 16 to 128 simultaneous signals.



The question is, if you are tracking down a system malfunction where these waveforms are involved, what do you measure and how? If the clock rate is too fast for a chopped oscilloscope display, you can't capture the four waveforms on a storage oscilloscope for analysis. Electronic counters could tell you how many logic highs occurred on each channel, but would not give timing relationships. You could also derive the number of logic highs from voltmeter measurements of the average value of the waveforms but what would this tell you? Oscilloscopes, voltmeters, and counters are familiar to all, but none of these classic instruments does a very good job on digital problems because none was designed to solve them.

Such a set of signals can be examined meaningfully with the aid of a logic state analyzer. These instruments sample all channels on every clock edge, detect the logic levels on all channels simultaneously, and store them for display and study.

The more important aspect of the problem, however, is what do we need to know about these signals once they have been captured? Here is where the concept of the data domain comes in. These waveforms can be control signals or they can be instructions, memory addresses, or data. Whatever they are, they can have varied meanings. If, for example, they represent bit-serial ASCII symbols with even parity, as may be found on an I/O bus, then the 8-bit frames here represent the letters D, B, F, and J. It might not be obvious from examining the waveforms that a parity error occurred with the letter J nor what caused the error, yet in terms of the data being transmitted, an error most certainly did occur. Before it can be corrected, its existence must be recognized.

These waveforms could also be bit-serial, least-significant bit and least significant digit first, hexadecimal code (essentially the data format of HP's pocket calculators). They would then be interpreted as 44, 42, 46, and 4A. Or, if they were word-serial hexadecimal, as in HP's 21MX Computers, they would mean something else.

Obviously, there are a host of choices in terms of data format, data code, and logic conventions that must be taken into account when dealing with the data domain. For the first generation of logic state analyzers, the choice was made to use single-level threshold (hi or lo, up or down, on or off), indexing by recognizing binary statements (Boolean triggering), and portrayal of the data as 1's and 0's. This machine-language presentation does not restrict the data format but leaves it to the user to interpret the display in terms of the code used.

When considering where and for what tasks a logic state analyzer may be used, the question invariably arises, "don't you ultimately have to see the waveforms to fix the problem?" It is worth trying to put this into perspective.

What logic state analyzers can do is to aid in the debugging of complex digital systems, particularly between the time that the computer simulation of the design is complete and the working hardware is operational. Because of the long data stream sequences typically used in most algorithmic design, particularly when looping or nested sub-routines are involved, locating the problem is more critical than analyzing why the problem occurred. It may simply be a software problem, such as accessing the wrong instruction in memory. This can be readily identified by a logic state analyzer.

However, when an electrical malfunction is the culprit, an oscilloscope is needed but it can't find which electrical parameters are at fault unless it gets a trigger from the vicinity of the bad data. This can be located and provided by a logic state analyzer.

The logic state analyzer is not about to displace the oscilloscope as a troubleshooting tool for digital systems, but it does add a dimension to test instrumentation that until now had not been adequately provided. The logic state analyzer can capture a segment of a rapidly executing digital sequence for analysis just as the oscilloscope can capture a waveform for examination.

Charles H. House

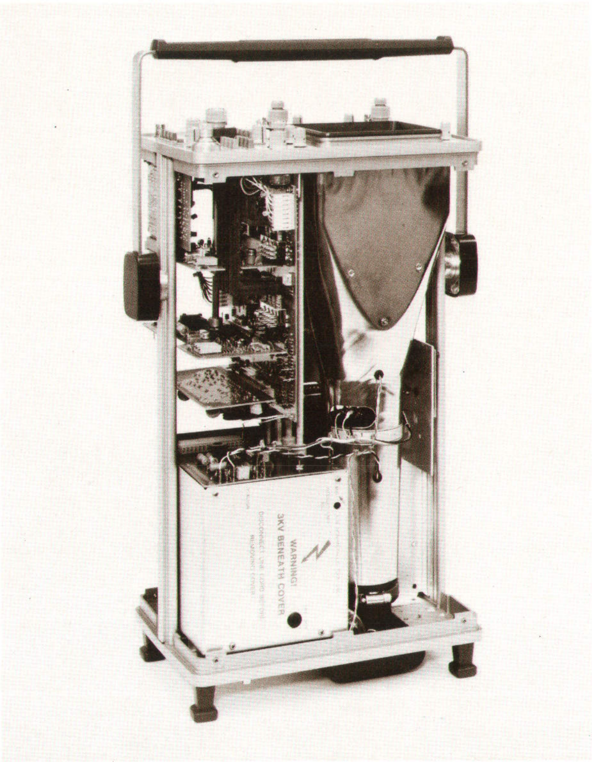
logic state display, as generated by the analyzer, and the analog display of signals detected by the scope's own probes (Fig. 2). There is no need to reconnect cables or reset controls when switching displays.

The logic state analyzer monitors data flow clocked in on up to 16 lines simultaneously. It generates the deflection voltages necessary for the oscilloscope so the clocked-in data can be displayed as a machine-language table of 1's and 0's, enabling the user to see the data flow on the monitored lines. A front-panel switch register can be set to any digital word up to 16 bits wide and when that word occurs, a pulse is generated that can be used to trigger the scope.

The user can page through an executing program with the logic state analyzer and once a problem area has been identified, the trigger word can be reset to a

A "Visible" Mechanical Design

The photo below, showing the Model 1740A Oscilloscope with both covers removed, illustrates the openness of the mechanical design. The byword during the design phase was "visibility"—visibility in this case meaning a high degree of order in the internal layout and ready accessibility to all test points and components.

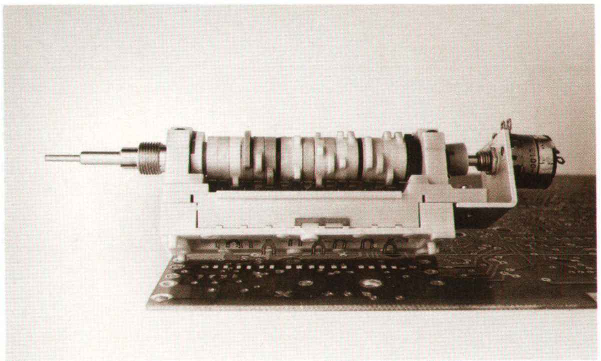


The primary element in the "visible" design was the reduction in wiring and cabling. The circuits are grouped functionally on eleven plug-together boards: three for the power supply, three for the vertical section, and five for the horizontal section. The three sections are interconnected by the interconnect board, a twelfth circuit board that satisfies the requirements of a cable that would have had perhaps some 26 wires. Thus, 52 cable solder and/or crimped connections were eliminated. A further benefit of the arrangement is that the vertical and horizontal sections can be disconnected from the power supplies and from each other to aid in troubleshooting.

Front-panel wiring was reduced substantially by mounting controls on the circuit boards wherever possible and using extension shafts from the front-panel. In many cases this also obtained an electrical advantage by placing the controls close to the circuits they control.

The powerline switch, fuse, and line-voltage-select switches are mounted on one of the power-supply circuit boards rather than the rear panel. This enabled effective isolation of the powerline primary circuit from the rest of the instrument, and it further simplified wiring.

A new approach to attenuator design eliminated much of the production time formerly required for assembling a complex attenuator. No electronics are contained within the switch mechanism itself. Instead, actuating cams press spring-finger shorting contacts down on pads in the printed-circuit board to switch gain and/or select input coupling modes, as shown by the wide-angle photo below where the switch has been raised off the board to disclose details. All the switched circuits can thus be incorporated on the printed-circuit board. This arrangement reduced assembly costs significantly.



Besides contributing to a more visible mechanical layout, the plug-together design also simplified some of the circuits. By eliminating the cable-to-cable variations in adjacent lead capacitance, the plug-together construction permitted a reduction in the number of adjustments that would otherwise be required to normalize performance. The plug-together-by-function design also permits thorough testing of the individual circuit functions before final assembly.

In the interest of reducing assembly costs, the mechanical parts, such as brackets, were standardized or eliminated as far as possible. For example, the usual practice of selecting the length of a screw to be just long enough to protrude 1/32 inch beyond its fastener was abandoned in the interest of reducing the number of different screws. This reduction in the number of screw types will be especially appreciated by service personnel who may have an occasion to disassemble and reassemble the instrument.

Circuits were designed not only for performance but also for minimum power consumption. As a result, the oscilloscope's total power consumption is less than 100 VA. Thus, no fan is required nor are vent holes required, thereby obtaining an extra degree of protection against dust and other contaminants.

John W. Campbell

word near the problem area. Then by switching the scope to the analog display, bus and control lines can be monitored to locate glitches, race problems, insufficient amplitude and other electrical problems that may be the cause of the digital problems.

Acknowledgments

The 1740A design group was led by Stan Lang until the start of pilot production when he transferred to another project. In addition to those mentioned elsewhere in these articles, the design team included Jim Carner, mechanical design including the vertical attenuator switch, Eldon Cornish, who designed the horizontal section, and Van Harrison who designed the CRT circuits, power supplies, and gate amplifier. Special thanks are due John Riggen and John

Cardon who provided valuable suggestions and support as section managers, and to Dick Stone who as product manager provided inputs anticipating customer requirements.

References

1. C.T. Small and J.S. Morrill, Jr., "The Logic State Analyzer, a Viewing Port for the Data Domain," Hewlett-Packard Journal, August 1975.
2. F.G. Siegel, "A New DC-50+ MHz Transistorized Oscilloscope of Basic Instrumentation Character," Hewlett-Packard Journal, August 1966.
3. P.K. Hardage, S.R. Kushnir, and T.J. Zamborelli, "Optimizing the Design of a High-Performance Oscilloscope," Hewlett-Packard Journal, September 1974.

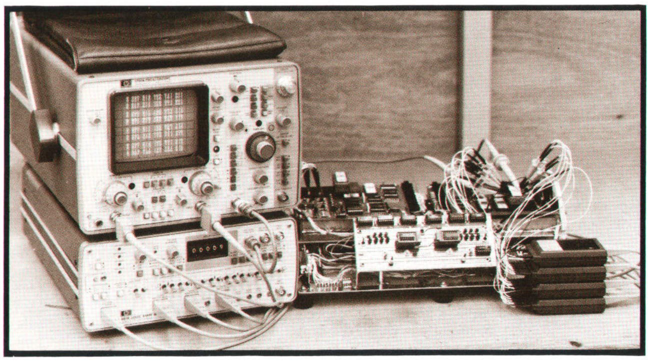


Fig. 3. *Model 1740A Oscilloscope equipped with the logic-state option (opt 101) is available with Model 1607A Logic State Analyzer (lower unit) in a package known as Model 1740S.*

Allan I. Best

A San Franciscan by birth, Al Best joined Hewlett-Packard's Oscilloscope Division in Palo Alto upon getting his BSEE degree from the University of California at Berkeley (1960), and moved with the division to Colorado Springs in 1964. Over the years he has contributed to a wide range of oscilloscope products as a circuit designer (185B Sampling Oscilloscope), project leader (1410A and related sampling plug-ins) and group manager (1740A). In his spare time, Al enjoys skiing in the winter and high-country trout fishing, back-packing, and 4-wheel driving during the rest of the year. He is married and has four children.

SPECIFICATIONS
Model 1740A Oscilloscope

Vertical Display Modes

Channel A; channel B; channels A and B displayed alternately on successive sweeps (ALT) or by switching between channels at 250 kHz rate with blanking during switching (CHOP); channel A plus channel B (algebraic addition); and trigger view.

Vertical Amplifiers (2)

Bandwidth and Rise Time at all deflection factors over temperature range of 0°C to +55°C.

BANDWIDTH (3 dB down from 8-div reference signal):
- DC-COUPLED: dc to 100 MHz in both 50Ω and 1 MΩ input modes.
- AC-COUPLED: approx 10 Hz to 100 MHz; 1 Hz with 10:1 divider probes.

BANDWIDTH LIMIT: limits upper bandwidth to 20 MHz.

RISE TIME: ≤3.5 ns, measured from 10% to 90% points of a 6-div input step.

DEFLECTION FACTOR
- RANGES: 5 mV/div to 20 V/div (12 calibrated positions) in 1, 2, 5 sequence, accurate within 3%.
- VERNIER: continuously variable between all ranges, extends maximum deflection factor to at least 50 V/div.

POLARITY: channel B may be inverted, front-panel pushbutton.

DELAY LINE: input signals are delayed sufficiently to view triggering edge.

INPUT COUPLING: selectable ac or dc, 50Ω (dc), or ground. Ground position disconnects input connector and grounds amplifier input.

INPUT RC (selectable):
- AC OR DC: 1 MΩ ±2% shunted by approx 20 pF.
- 50 OHM: 50Ω ±3%, SWR ≤1.4 at 100 MHz on all ranges.

MAXIMUM INPUT
- AC OR DC: 250 V (dc + peak ac) or 500 V p-p at 1 kHz or less.
- 50 OHM: 5 V rms.

A+B OPERATION
- AMPLIFIER: bandwidth and deflection factors are unchanged, channel B may be inverted for A−B operation.
- DIFFERENTIAL (A−B) COMMON MODE: CMRR is at least 20 dB from dc to 20 MHz. Common mode signal amplitude equivalent to 8 divisions with one vernier adjusted for optimum rejection.

VERTICAL MAGNIFICATION (×5)
- BANDWIDTH: 3 dB down from 8-div reference signal.
  - DC-COUPLED: dc to approx 40 MHz.
  - AC-COUPLED: approx 10 Hz to 40 MHz.
- RISETIME: ≤9 ns(measured from 10% to 90% points of 8-div input step).
- DEFLECTION FACTOR: increases sensitivity of each deflection factor setting by factor of 5 with maximum sensitivity of 1 mV on channels A and B.

TRIGGER SOURCE
- CHANNEL A: all display modes triggered by channel A signal.
- CHANNEL B: all display modes triggered by channel B signal.
- COMPOSITE: all display modes triggered by displayed signal except in Chop. In Chop mode, trigger signal is derived from channel A.
- LINE FREQUENCY: trigger signal is derived from power line frequency.

TRIGGER VIEW

Displays internal or external trigger signal. In Alternate or Chop mode, channel A, channel B, and trigger signals are displayed. In channel A or B mode, Trigger View overrides that channel. Internal trigger signal amplitude approximates vertical signal amplitude. Ext trigger signal deflection factor is approx 100 mV/div, or 1 V/div in EXT ÷ 10. Triggering point is approximately center screen. With identically timed signals to a vertical input and the Ext trigger input, trigger signal delay is 2.5 ns ± 1 ns.

Horizontal Display Modes

Main, main intensified, mixed, delayed, mag ×10, and A vs B.

MAIN AND DELAYED TIME BASE RANGES
- MAIN: 50 ns/div to 2 s/div (24 ranges) in 1,2,5 sequence.
- DELAYED: 50 ns/div to 20 ms/div (18 ranges) in 1,2,5 sequence.

TIME BASE ACCURACY

Sweep Time/Div	Accuracy* ×1	Accuracy* ×10	Temp Range
50 ns to 20 ms	±3%	±4%	0°C to +15°C
	±2%	±3%	+15°C to +35°C
	±3%	±4%	+35°C to +55°C

*Add 1% for 50 ms to 2 s ranges.

MAIN SWEEP VERNIER: continuously variable between all ranges, extends slowest sweep to at least 5 s/div.

MAGNIFIER (×10): expands all sweeps by factor of 10, extends fastest sweep to 5 ns/div.

CALIBRATED SWEEP DELAY
- DELAY TIME RANGE: 0.5 to 10 × Main Time/Div settings of 100 ns to 2 s (minimum delay 150 ns).
- DIFFERENTIAL TIME MEASUREMENT ACCURACY

Main Time Base Setting	Accuracy (+15°C to +35°C)*
100 ns/div to 20 ms/div	±(0.5%+0.1% of full scale)
50 ms/div to 2 s/div	±(1%+0.1% of full scale)

*Add 1% for temperatures from 0°C to +15°C and +35°C to +55°C.

DELAY JITTER: <0.002% (1 part in 50 000) of maximum delay in each step from +15°C to +35°C; <0.005% (1 part in 20 000) from 0°C to +15°C and +35°C to +55°C.

CALIBRATED MIXED TIME BASE

Dual time base in which main time base drives first portion of sweep and delayed time base completes sweep at faster rate. Also operates in single sweep mode. Accuracy, add 2% to main time base accuracy.

Triggering

MAIN SWEEP
- NORMAL: Sweep is triggered by internal or external signal.
- AUTOMATIC: bright baseline displayed in absence of input signal. Triggering is same as Normal above 40 Hz.
- SINGLE: sweep occurs once with same triggering as Normal; reset pushbutton arms sweep and lights indicator.

DELAYED SWEEP (SWEEP AFTER DELAY)
- AUTO: delayed sweep automatically starts at end of delay.
- TRIG: delayed sweep is armed and triggerable at end of delay period.

INTERNAL: dc to 25 MHz on signals causing 0.3 divisions or more vertical deflection, increasing to 1 division of vertical deflection at 100 MHz in all display modes (required signal level is increased by 2 when in Chop mode and by 5 when ×5 vertical magnifier is used). Triggering on Line frequency is also selectable.

EXTERNAL: dc to 50 MHz on signals of 50 mV p-p or more increasing to 100 mV p-p at 100 MHz (required signal level is increased by 2 when in Chop mode).

EXTERNAL INPUT RC: approx 1 MΩ shunted by approx 20 pF.

MAXIMUM EXTERNAL INPUT: 250 V (dc + peak ac) or 500 V p-p ac at 1 kHz or less.

LEVEL AND SLOPE
- INTERNAL: at any point on positive or negative slope of displayed waveform.
- EXTERNAL: continuously variable from +1.5 V to −1.5 V on either slope of trigger signal, +15 V to −15 V in divide-by-10 mode (÷10).

COUPLING:
- DC: full range
- AC: attenuates signals below approx 20 Hz.
- LF REJECT (MAIN SWEEP): attenuates signals below approx 4 kHz.
- HF REJECT (MAIN SWEEP): attenuates signals above approx 4 kHz.

TRIGGER HOLDOFF (MAIN SWEEP): increases sweep holdoff time.

A vs B Operation

BANDWIDTH
- CHANNEL A (Y AXIS): same as channel A.
- CHANNEL B (X-AXIS): dc to 5 MHz.

DEFLECTION FACTOR: 5 mV/div to 20 V/div (12 calibrated positions) in 1,2,5 sequence.

PHASE DIFFERENCE BETWEEN CHANNELS: <3°, dc to 100 kHz.

Cathode-Ray Tube and Controls

TYPE: Hewlett-Packard 12.7 cm (5 in) rectangular CRT, post accelerator approx 15-kV accelerating potential, aluminized P31 phosphor.

GRATICULE: 8×10 div (1 div = 1 cm) internal, non-parallax graticule with 0.2 subdivision markings on major horizontal and vertical axes and markings for rise time measurements. Internal floodgun graticule illumination.

BEAM FINDER: returns trace to CRT screen regardless of setting of horizontal, vertical or intensity controls.

Z-AXIS INPUT: +4V, ≥50 ns width pulse blanks trace of any intensity, usable up to 10 MHz for normal intensity. Input R, 1 kΩ ± 10%. Maximum input ±20 V (dc + peak ac).

REAR PANEL CONTROLS: astigmatism and trace align.

General

REAR PANEL OUTPUTS: main and delayed gates, 0.8 V to > + 2.5 V capable of supplying approx 5 mA.

AMPLITUDE CALIBRATOR (0°C to +55°C)
- OUTPUT VOLTAGE: 1 V p-p ±1% into ≥1 MΩ, 0.1 V p-p ±1% into 50Ω.
- RISE TIME: ≤0.1 μs
- FREQUENCY: approximately 1.4 kHz

POWER: 100, 120, 220, 240 Vac. ±10%, 48 to 440 Hz, 100 VA max.

WEIGHT: 13 kg (28.6 lb).

OPERATING ENVIRONMENT
- TEMPERATURE: 0°C to +55°C
- HUMIDITY: to 95% relative humidity at +40°C.
- ALTITUDE: to 4600 m (15,000 ft).
- VIBRATION: vibrated in three planes for 15 min each with 0.254 mm (0.010 in) excursion, 10 to 55 Hz.

DIMENSIONS: 335 mm W × 197 mm H × 492 mm D (l3.l9 × 7.75 × 19.38 in)

ACCESSORIES FURNISHED: blue light filter, front panel cover, power cord, vinyl accessory storage pouch, operators guide and service manual, two Model 10006D 10:1 divider probes.

OPTIONS
- 001: fixed power cord in lieu of detachable power cord.
- 101: LOGIC STATE DISPLAY single pushbutton (Gold Button) interface Option for operation with HP Model 1607A Logic State Analyzer.

PRICES IN U.S.A.:
- MODEL 1740A 100 MHz Oscilloscope, $1995.
  - OPTION 001: Add $15.
  - OPTION 101: Add $105.
- MODEL 1740S includes 1740A with Option 101, Model 1607A Logic State Analyzer, four interconnecting cables, and bracket and strap for combining into a single package, $4935.

MANUFACTURING DIVISION: COLORADO SPRINGS DIVISION
1900 Garden of the Gods Road
Colorado Springs, Colorado 80907 U.S.A.

An Oscilloscope Vertical-Channel Amplifier that Combines Monolithic, Thick-Film Hybrid, and Discrete Technologies

To minimize maintenance and calibration times by minimizing the number of parts and the number of adjustments, a high degree of integration was incorporated in the vertical amplifier system of the Model 1740A Oscilloscope.

by Joe K. Millard

HYBRID THICK-FILM TECHNOLOGY using HP-manufactured monolithic chips enables the vertical channel of the Model 1740A Oscilloscope to meet its bandwidth specifications without time-consuming adjustment of many trimmers. Furthermore, the specified bandwidth is maintained throughout an operating temperature range of 0 to 55°C.

Signal conditioning is accomplished primarily by two hybrid thick-film integrated circuits, shown as U1 and U2 in the block diagram of Fig. 1. The only other active components are the discrete FET impedance converters at the input, and the circuits involving transistors Q1-Q4.

Discrete components are used for attenuation only in the ×100 section preceding the FET impedance converter in each channel. The preamplifier IC (U1), besides carrying out the necessary control functions, performs six dc-actuated attenuation ranges per channel. With the ×100 attenuator, this realizes twelve calibrated deflection-factor ranges, from 5 mV/cm to 20 V/cm.

Range selection is accomplished by the switch assembly described on page 6 of the preceding article. The spring-finger contacts of this switch complete circuit paths through appropriate pads on the circuit board. Only the first five contacts, controlling

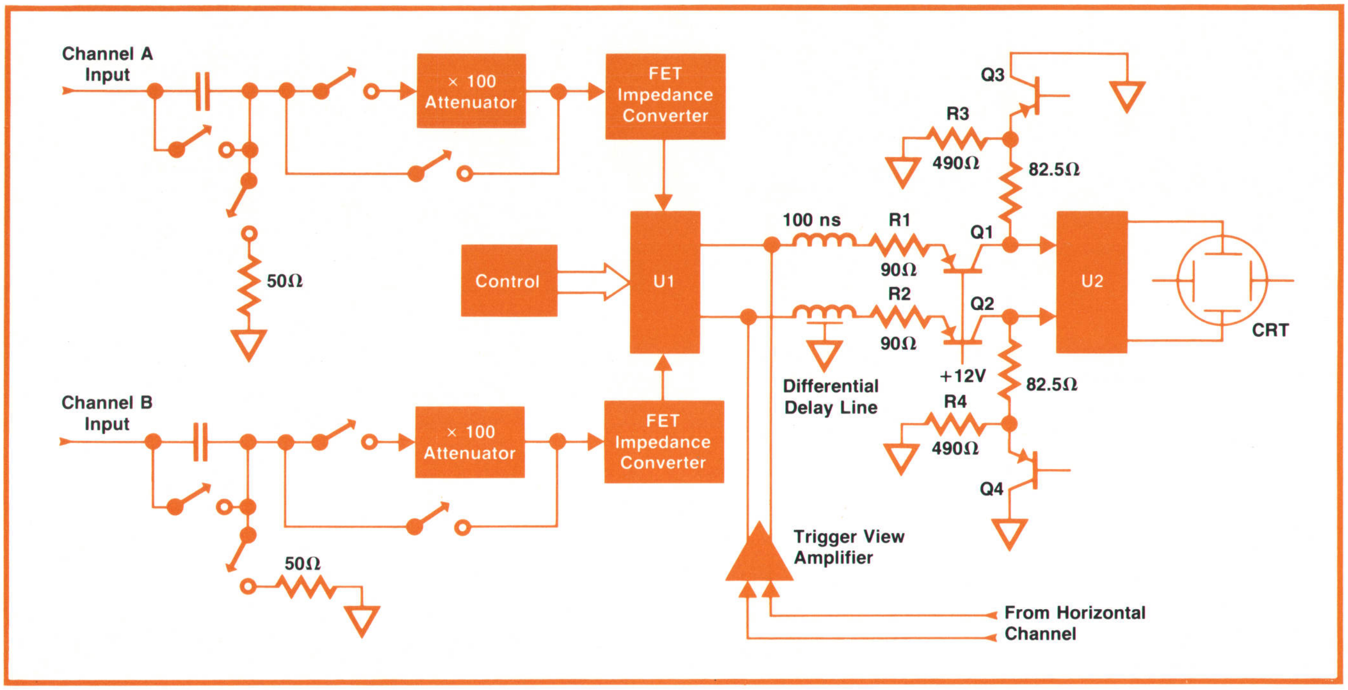
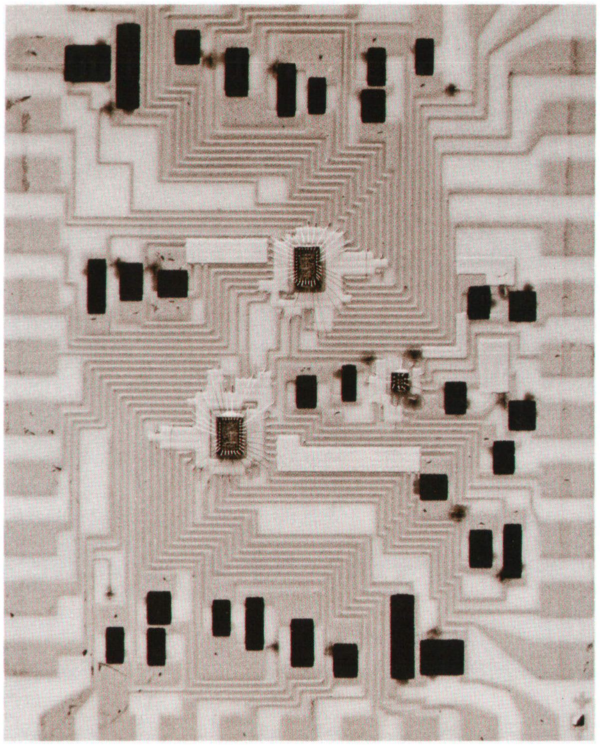


Fig. 1. *Block diagram of the vertical channel in the Model 1740A Oscilloscope. Most of the signal conditioning occurs within the two hybrid thick-film circuits, U1 and U2.*

Designing a High-Density Thick-Film Hybrid Integrated Circuit

Placing three monolithic chips, thirty-one resistors, four capacitors, and seventy-six wire bonds on a standard 25×35-mm substrate for the Model 1740A Oscilloscope preamp challenged the limits of thick-film technology (thick-films are used rather than thin films to minimize costs). The component density dictated the use of 0.15-mm conductors, which is definitely fine-line geometry by thick-film standards. In addition, the resistors and capacitors would have to be smaller than those used in present practice. The design would also have to eliminate many of the proposed probing pads needed for resistor trimming. These are space hogs that are better done without.

How were all these requirements met with a 210-nanoacre substrate? The fine-line geometry was achieved by pre-treating the substrate surface with a chemical agent that lowers the surface energy, preventing the screened paste from running in much the same way that the freshly-waxed surface of an automobile doesn't allow water beads to spread out.



The small-sized precision resistors were realized by refining laser trimming techniques to work with smaller geometries. The number of probing pads was reduced by connecting selected resistors to common nodes with shorting tabs and opening the tabs with the laser after the resistors have been trimmed. The need for discrete chip capacitors was eliminated by using thick-film capacitors constructed with an interdigitated structure coated with a glass frit that has a high dielectric constant.

A four-day burn-in of each completed hybrid, plus several quality-assurance gates along the way, assures high reliability. Finished circuits are thoroughly tested in only twenty seconds using an automatic test system designed for that purpose.

Richard D. Tabbutt

the coupling modes and the ×100 attenuator, carry signal currents while the other five simply switch dc control voltages to integrated circuit U1. This arrangement, besides minimizing the number of components and simplifying assembly, also improves performance by shortening the signal paths.

The preamp circuit U1 performs the conventional control functions of signal polarity selection, gain vernier control, channel switching and sync extraction, in addition to the six ranges of signal attenuation.

The trigger-view amplifier routes the trigger signal into the vertical channel at the output of U1, as shown in Fig. 1. It is electronically switchable, so it can be sequenced with channels A and B to derive a three-channel display showing the time relationship between the sweep trigger and the signals in channels A and B.

The output of U1 drives the delay line. Resistors R1 and R2 terminate the delay line to prevent reflections. Transistors Q1 and Q2 are impedance converters that also function as dc level shifters.

Deflection factors to 1 mV/cm for both channels are provided by the ×5 magnifier controlled by transistors Q3 and Q4. These transistors are normally saturated, shorting out R3 and R4 to provide a low RC time constant at the input to U2. When transistors Q3 and Q4 are switched off, the system gain is increased by a factor of five with a bandwidth of 40 MHz. At the same time, the positioning voltages are reduced by the same factor to maintain constant positioning.

Hybrid integrated circuit U2 provides a voltage gain of 50 for driving the CRT.

Preamplifier IC

Further simplification of the overall vertical assembly was achieved by placing most of the preamplifier circuits for both channels on a single hybrid integrated circuit (U1). The 25.4 × 34.9-mm ceramic substrate (see box at left) has 31 thick-film resistors, 4 capacitors, and 3 monolithic chips. The two large chips are the channels A and B preamp circuits, each consisting of 27 transistors, 23 diodes, and 34 monolithic resistors. The third chip is a four-transistor differential shunt-feedback amplifier that drives the balanced delay line.

An abridged schematic of one of the preamp chips is shown in Fig. 2. Following the signal path starting at the input to the chip, transistors Q1-Q3 along with diodes D1-D4 form a dc-controlled ×10 attenuator in conjunction with laser-trimmed resistors RT1 and RT2 on the hybrid substrate. The attenuator is actuated by biasing the lower end of resistor R1 to the appropriate negative voltage and allowing the lower end of R2 to float.

The ×10 attenuator is followed by triple-emitter transistors Q4 and Q5 and thick-film resistors RT3-

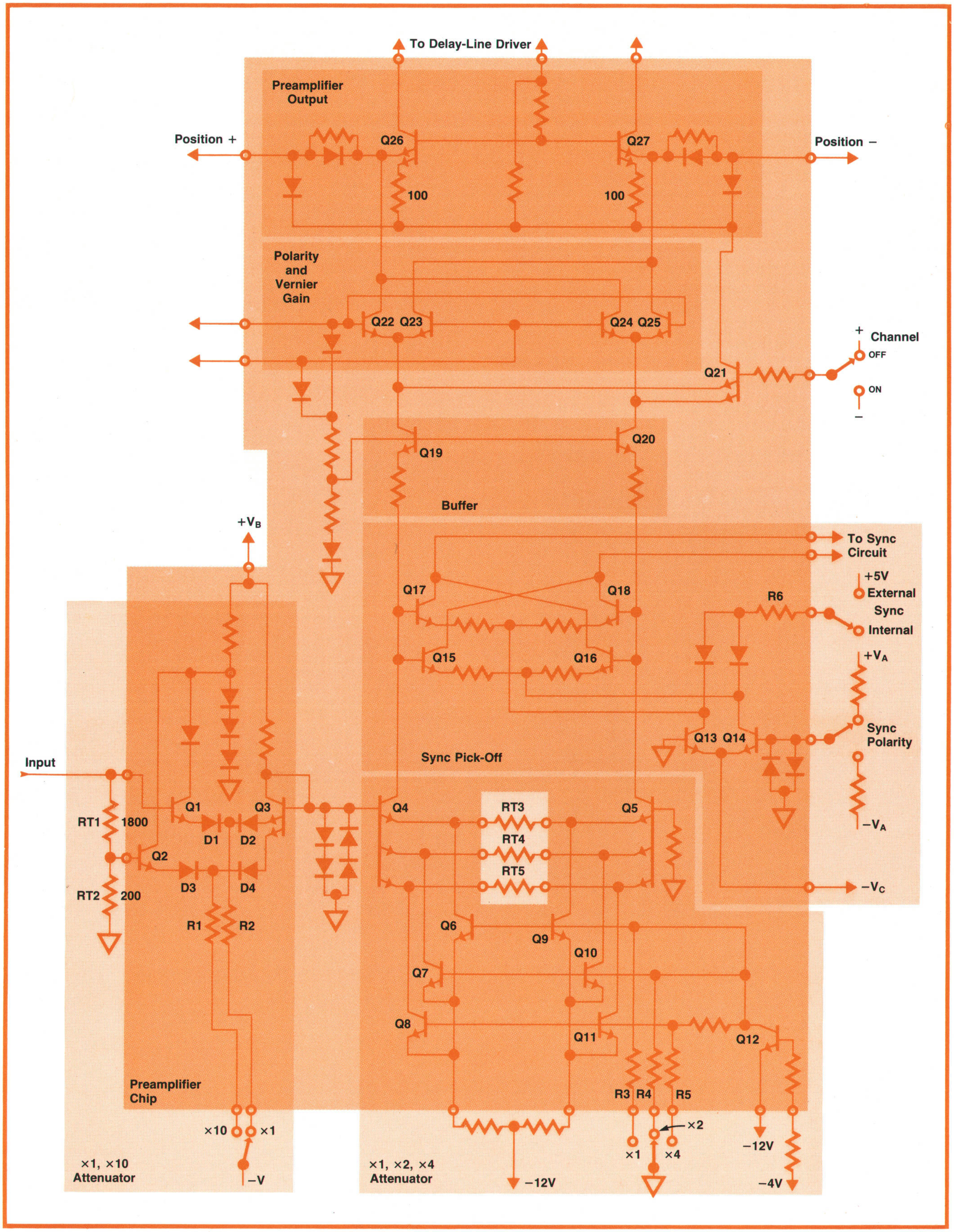


Fig. 2. *Abridged schematic of one of the two preamplifier monolithic integrated circuits.*

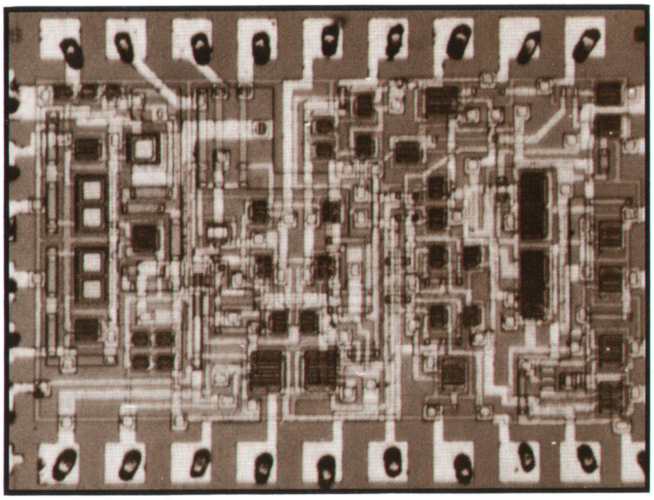


Fig. 3. *Preamp chip has twenty-seven 2-GHz transistors.*

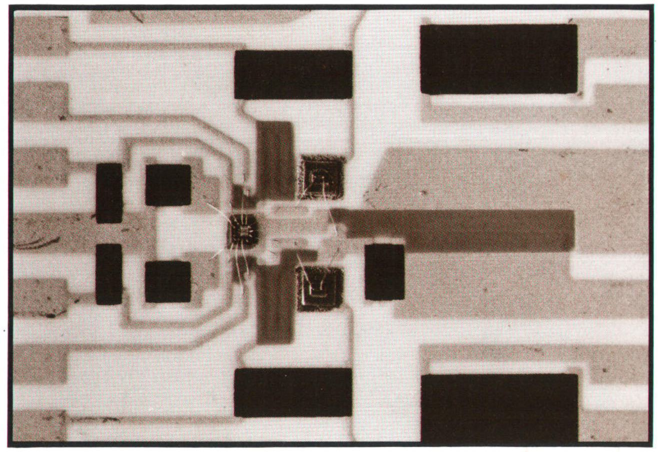


Fig. 4. *Hybrid output stage uses discrete chips for drivers.*

RT5 which constitute an attenuator with a 1-2-4 attenuation sequence. Range selection in this section is accomplished by grounding the lower end of resistor R3, R4, or R5 to actuate the appropriate set of current-source transistors (Q6-Q11).

During range selection, this section cycles four times while the ×10 section cycles twice and the external ×100 section once to give twelve attenuation ranges.

Sync Extraction

The sync signal is extracted at the outputs of transistors Q4 and Q5. As with other recent HP oscilloscopes, sync extraction precedes the polarity and gain vernier controls to prevent loss of triggering when these controls are adjusted. Transistors Q17 and Q18 invert the sync signal when they are turned on (and transistors Q15 and Q16 are turned off) by transistors Q13 and Q14. To switch the sync signal channel completely off, +5V is applied to resistor R6.

Proceeding towards the output through buffer amplifier Q19-Q20, the next control functions are the gain vernier control and channel polarity. These two functions are accomplished by a four-quadrant multiplier configuration (Q22-Q25) that provides continuously adjustable gain over a 2.5:1 range while maintaining a constant dc bias current.

Channel switching is accomplished by double-emitter transistor Q21. When the base potential on this device exceeds the base voltages on transistors Q22-Q25, it extracts and sums the currents that would flow to transistors Q22-Q25. The collector current of Q21 divides equally into the lower emitters of Q26 and Q27 so the channel bias current remains constant, maintaining the dc output level constant, but all signal information is lost.

Position modulation is accomplished by differentially varying the bias currents injected into the upper emitters of Q26 and Q27.

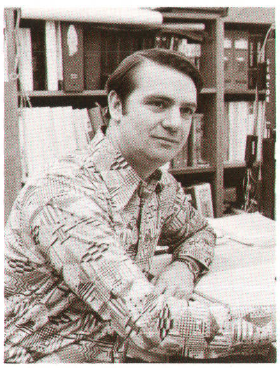
The collectors of Q26 and Q27 are connected to the corresponding collectors of the other preamp chip and to the input of a four-transistor delay-line driver. This stage provides a current gain of 8 when driving the 180Ω differential delay line.

Output IC

The output amplifier (U2 in Fig. 1) consists of a 25-mm square ceramic substrate with nine thick-film resistors, one high-frequency monolithic chip containing six transistors, and two discrete transistor chips for the final drive. The short signal paths afforded by the thick-film hybrid technology plus the performance of the HP transistors enabled these eight transistors to achieve a differential voltage gain in excess of 50 at a bandwidth of 150 MHz and with differential drive capability of 70 mA.

Acknowledgments

Ruth Buss, Gina Anderson, and Rose Stamps spent many hours developing prototypes of the hybrid circuits. Ken Fulton contributed to the special hybrid processing procedures and Joe Cochran developed the hybrid testing procedures.



Joe K. Millard

A native of Maryville, Tennessee, Joe Millard was involved with the design and development of nuclear instrumentation for seven years at the Oak Ridge National Laboratory (Tennessee) before joining Hewlett-Packard in 1972. He has BS, MS, and PhD degrees from the University of Tennessee. Married, and with two children, Joe golfs, skis, and hikes during leisure hours.

A Real-Time Operating System with Multi-Terminal and Batch/Spool Capabilities

RTE-II, an advanced version of HP's real-time executive system for 2100 Series Computers, has several new features that aid both real-time measurement and control and concurrent background activities such as program development.

by George A. Anzinger and Adele M. Gadol

ONE OF THE FIRST REAL-TIME operating systems to run on a 16-bit computer was Hewlett-Packard's disc-based, multiprogramming Real-Time Executive (RTE) system, introduced in 1968. Key features of this system were a priority scheme for concurrent execution of multiple programs and a foreground/background partition separating real-time tasks from non-real-time tasks. A powerful file management package was added later.

Experience gained in hundreds of RTE applications has now led to the development of RTE-II, an advanced version of this operating system. Major new capabilities are multi-terminal access to system resources and an optional batch-spool monitor that supplements the file manager. Multi-terminal operation is aided by buffering of input as well as output, background swapping, resource locking, and class input/output, a system of buffering and queuing I/O requests according to class numbers. The batch-spool monitor supervises program development and other background jobs, using spooling, or buffering of input and output job streams, to maximize throughput.

The principal hardware environment for RTE-II is the HP 9600 Series of real-time measurement and control systems.[1] RTE-II is also the operating system for central stations in HP 9700 Series Distributed Systems.[2] Central processors in these systems are HP 2100 or 21MX Computers.[3,4]

Multi-Terminal Operation

One of the requirements for RTE-II was that the system be able to handle multiple users at terminals, engaged either in program development or in use of the system for its real-time function, which might be anything from controlling a test to entering star charts in an observatory system. The central problems that were solved are common to many such uses.

The first of these problems was **buffer management**. Each terminal must be able to send data to the program or programs controlling it without locking any program into main memory so that it cannot be moved to the disc; this occurs, of course, if the area of memory we wish to move to the disc is being used in part as an input buffer. It is also desirable to have the input in the program's memory, so that it can be protected from other users and may be moved to the disc when input is not going on. RTE has always used buffered output. The output buffer and control information for it are moved to a block of system memory reserved for buffering; the actual output then takes place from this system memory, freeing the requesting program's buffer for further processing without waiting for the I/O device. In RTE-II we have provided for input buffering as well, by doing I/O from a reentrant subroutine, that is, a subroutine that can be shared by many programs. In the RTE system, reentrant subroutines contain a work space that the system moves to system available memory prior to reentering the subroutine (giving control of the subroutine to another program or process). The system restores this work space before it returns control to the interrupted process (see Fig. 1). Thus a program that has an active I/O request in progress may be moved to the disc in favor of a higher-priority program, which may also use the same I/O routine. When such an I/O request is completed, the I/O buffer is in system memory and is moved back to the user program's memory (as a side effect of restoring the work space) before it continues. By keeping the I/O buffer outside the user program while I/O is in progress but inside at other times the system minimizes its need for buffer memory and simplifies the protection of the system while allowing the program to be swapped. (Swapping, as defined in the HP RTE systems, consists of saving an executing program in its current state on the disc and replacing it in main memory

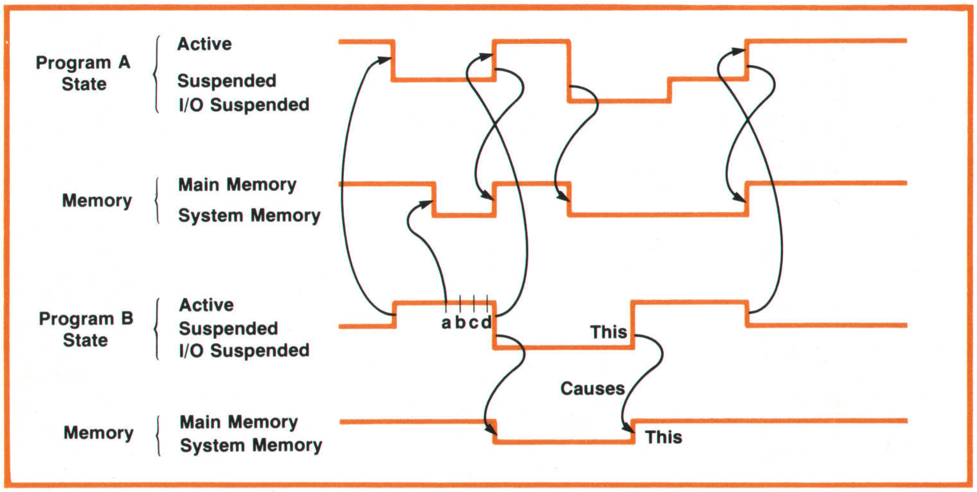


Fig. 1. *RTE-II provides for input buffering as well as output buffering by doing I/O from a reentrant subroutine. Here program A is processing in the reentrant subroutine when it is interrupted and control is given to program B. At point 'a' program B calls the same reentrant subroutine, causing program A's work space in memory to be moved to system memory. Program B makes repeated calls at 'b', 'c', and 'd', but memory is moved only once. When B is suspended, program A's work space is moved back and it continues in the subroutine from the point of suspension.*

with a program that was previously saved in the same manner or with a new program that is to be run from its primary entry point.)

A second problem was the need for **background swapping**. The background in an RTE system is an area of memory usually dedicated to running non-real-time tasks such as languages, editors, loaders, and other support programs. In HP RTE systems before RTE-II, background programs could not be swapped. This was consistent with the primary function of the system being real-time activities, which usually run in the foreground, and not terminal activity. For RTE-II, we wanted to add terminal activity and batch processing capability, which implies multiple editors, a batch monitor, and other non-real-time tasks that should not interfere with the foreground real-time activity. Therefore, we have provided the ability to swap out a terminal program or the batch monitor while waiting for an event to occur, such as completion of I/O or a subordinate program.

Third, provision had to be made for **resource control**. To allow several users at different terminals to access resources without interfering with each other, we have provided a locking mechanism. It is controlled by the system, so if a program is aborted the lock will be removed. There are two types of locks.

In resource number (RN) locking, two or more cooperating users assign a number to a resource, such as a section of code, that is to be used by their programs, but by only one at a time (Fig. 2). The operating system is restricted to allowing only one program to lock a given resource at a time and to queueing other requesting programs on the RN unlock. In logical unit (LU) locking, a program can lock an I/O device. (A logical unit in the RTE system is a number assigned to some I/O device.) The program has exclusive control of the device until it either unlocks the device or terminates. This type of locking is very useful if the I/O device is a line printer while it is not very useful for discs.

To access the multi-terminal capabilities of the system, the user needs to be able to initiate a dialogue from any one of the terminals. This is accomplished by the **multi-terminal monitor** (MTM). MTM consists of two very short programs which, when any key is struck on the terminal:

- Identify the terminal and send a prompt to the terminal, which identifies to the user the system address of that terminal
- Accept and execute any system command from the terminal
- If the command is a program invocation, supply to the program the address of the terminal
- Send any message resulting from the execution of the command back to the terminal.

To allow one program to handle more than one terminal or device, it is necessary that it continue processing while waiting for input/output. This was made possible by the **Class I/O** system (Fig. 3). In

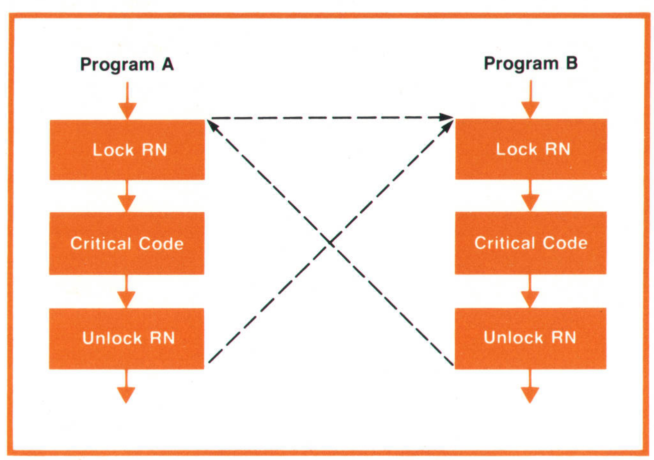


Fig. 2. *Resource number (RN) locking allows two or more cooperating programs to access sensitive areas of their code on a one-at-a-time-only basis. If program A gets to the lock first, B will be suspended until A unlocks that RN, at which time B is reactivated. B may then lock the RN, causing A to be suspended if it requests a lock on the same RN.*

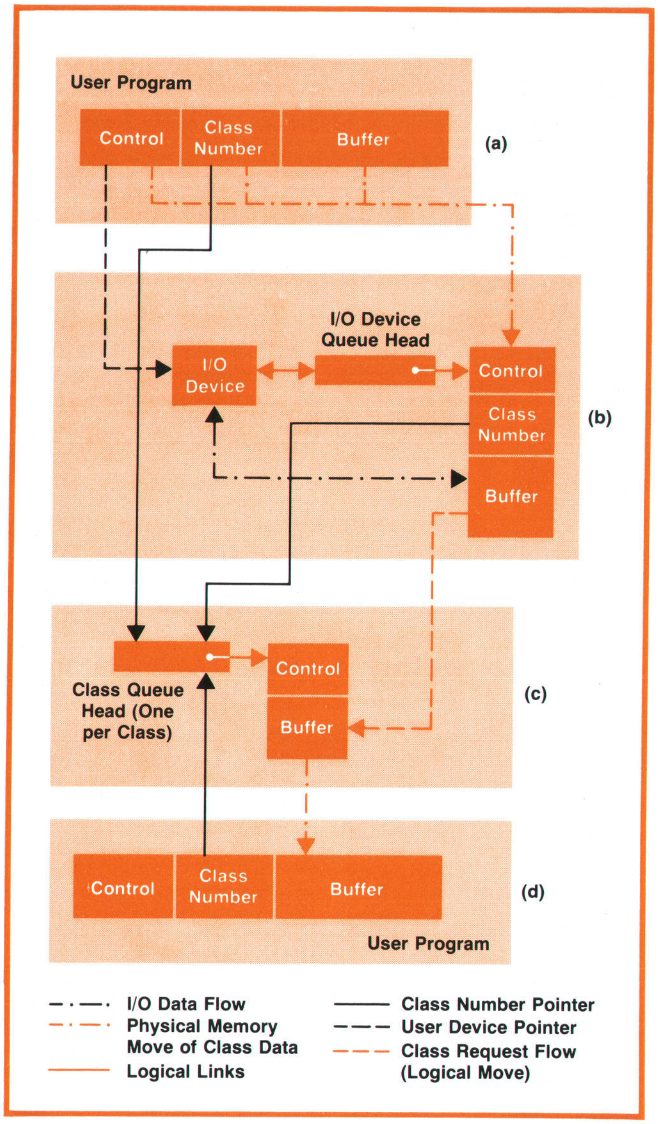


Fig. 3. *The Class I/O system makes it possible for a program to continue processing while waiting for input/output. When a Class I/O request is made (a), the requesting user program specifies a class number and an I/O device. The system moves this information to its memory and queues it on the specified I/O device (b). The I/O device driver then moves information to/from the buffer from/to the device. When the I/O is complete the driver signals the system which, by altering queue pointers, logically moves the completed request to the proper class queue (c). A user program, which may be the requesting program or another program, may now request information from this class queue (d). The system then moves the control and buffer information to the program's memory. A program requesting class information that has not yet reached the class queue is suspended until the information is available.*

the Class I/O system we have:

- Separated the I/O initiation and completion indications that a program makes and receives.
- Fully buffered I/O requests so the user need not worry about memory management or swappability.
- Allowed a user other than the initiator to receive I/O completion information, provided he knows the security code for the request.
- Provided a built-in dummy I/O device for program-to-program communication so that a program can control several I/O devices while also receiving data from another program.

The class I/O system has been used in HP distributed system software,[2] in the spool system, and in the multi-terminal monitor. It has proved flexible enough to handle tasks not even remotely related to its originally intended functions.

The maximum number of classes is established at system generation time. Once the class numbers are established the system keeps track of them and assigns them (if available) to any program making a Class I/O call with the class number parameter set to zero. Once the number has been allocated, the user can keep it as long as desired and use it to make multiple Class I/O calls. When the user is finished with the number it can be returned to the system for use by some other class user.

When the class user issues a Class I/O call the system allocates a buffer from system available memory and puts the call parameters in the header of this buffer. If the request is a WRITE or WRITE/READ* the rest of the buffer is filled with the caller's data. If the request is a READ the buffer will be filled when the I/O takes place. The buffer is then queued on the specified logical unit. Since the system forms a direct relationship between logical unit numbers and I/O devices, the buffer is actually queued on an I/O device. If this is the only call pending on that device the device driver is called immediately. Otherwise the system calls the driver according to program priority. In any case the program continues immediately without waiting for I/O completion.

After the driver completes its task the system queues the buffer in the completed class queue. If the request was a WRITE only the header is queued.

The system then waits for a GET call to that class number. The header (and data, if any) are then returned to the program that issued the GET call. Notice that it may or may not be the same program that issued the original Class I/O request. The GET issuer has the option of leaving the buffer in the completed class queue so as not to lose the data, or dequeuing it and releasing the class number. Completed requests for a given class number are queued on a first-in/first-out basis.

An example of the use of Class I/O for program-to-program communication is as follows:

- User program PROGA issues a Class WRITE/READ call with the class number parameter set to zero and the logical unit number set to zero. This causes the

*A class WRITE/READ call is treated by the system as a class WRITE in that the buffer space in system available memory is allocated and filled before the I/O driver is called, and as a class READ in that the entire buffer (and not just the header as for a WRITE call) is queued after the driver completes its task.

Introduction to Real-Time Operating Systems

An operating system is an organized collection of programs that increases the productivity of a computer by providing common functions for user programs. Examples of operating systems for specific purposes are:

- Timesharing (HP 2000)
- Disc Operating Systems (HP DOS-III)
- Real-Time Executive Systems (HP RTE-II/III)

A real-time computer system may be defined as one that "controls the environment by reviewing data, processing (it), and taking action or returning results sufficiently quickly to affect the functioning of the environment at that time."[1] The first applications of real-time measurement and control by computer occurred in the late 1950's and early 1960's. These pioneer applications were in the chemical and power industries and in command and control in the military. Their basic functions are still the basic functions of today's industrial computer systems, such as monitoring of sensors (analog and digital), periodic logging, scientific calculation, generation of management reports, and process control. The software of these early systems was tailored to each application; there was no distinction between what is today called the operating system software and the specific application software. All software development at that time was done in assembly language or machine language, and because of the high price of the computer hardware, a system could be justified economically only by having it perform many different functions. The result was very high system development costs that were not spread over many systems, but were repeated for every new application. Only in the middle 1960's did the real-time operating system appear as a separate entity that could be used as a building block for every application, with considerable savings in development cost.

The operating system software is part of the system software supplied with a computer system. System software includes assemblers, compilers, operating systems, loaders, libraries, and utilities (such as editors, debuggers, simulators, and diagnostics). These are the software tools needed for the development of applications programs required in a particular system. The operating system is in fact an extension of the computer system hardware; it helps the applications programmer use the computer system resources without detailed knowledge of the internal operation of I/O drivers, schedulers, file managers, and so on.

Some of the important functions of real-time operating systems are task management (program scheduling, resource allocation), memory management, input/output services, data management (file management, batch processing, I/O spooling, language processors, loaders, editors, debugging tools), and system integrity (power fail protection, memory protection, file security, error detection, etc.).

Many of the characteristics of real-time operating systems that boost speed and throughput, such as multiprogramming, concurrent I/O operations, system integrity features, and so on, are of a very general nature and are part of most commercial operating systems today. Early objections to such a generalized approach in non-real-time applications, such as larger core requirements, have mostly disappeared because of the dramatic lowering of memory prices.

HP RTE Operating System Family

The operating system of HP's first computer, the 2116A, was the Basic Control System (BCS), which was essentially an I/O monitor. Programming was done in HP assembly language or HP FORTRAN in a memory-based environment called System Input/Output (SIO). Since then the operating system software offered with 2100 Series Computers has evolved along several lines:

- DOS (Disc Operating System) for single user programming applications
- TODS (Test-Oriented Disc System) for automatic test applications
- Timeshared BASIC for multiple users programming in BASIC
- RTE (Real-Time Executive) for real-time multiprogramming.

RTE was initially developed for data acquisition, measurement, and control. It provides two environments for the user, physically separated in memory. *Background* is for program development tasks such as running a compiler or an editor. As the term suggests, a program running in the background is allowed to run when nothing more important needs to be run. *Foreground* is for time-critical or real-time applications. Foreground is protected from background by a hardware memory-protect fence, which prevents background programs from modifying the contents of any foreground memory location, transferring control to the foreground, or performing I/O. Any such attempts are intercepted by the system and examined for legitimacy, providing a high level of integrity for the foreground area. Programs not currently running may be swapped to disc. Time or event scheduling of programs is provided. A priority structure is provided and the system is optimized for response to the needs of real-time tasks. To further improve interrupt response where necessary, a privileged interrupt capability was added. With this capability the user can bypass the system entirely to service interrupts from devices chosen to be privileged.

RTE-C, a core-based version ("core" is what we called memory in the old days) is a later member of the RTE family, intended for applications where the environment will not tolerate a disc, or where the added cost of the disc is prohibitive. As in RTE, background and foreground areas are provided. Primary differences from RTE are that there is no disc for mass storage, and program preparation cannot be performed concurrently with real-time tasks.

Still later, to provide a simpler, more interactive facility for programming real-time tasks, RTE-B was created, offering real-time BASIC as a programming language in a very simple memory-based operating system.

To satisfy users' data handling requirements and to provide an improved interface to the system, a general-purpose file manager was added to the RTE system.[2] A powerful distributed systems capability was added to permit the user to create networks of systems with an RTE system functioning as the central station.[2]

The RTE-II system (article, page 12) was developed to improve RTE's performance in its primary applications of measurement and control as well as enhancing its usefulness as a general-purpose computational system by addition of a batch capability, input and output spooling, a multi-terminal monitor, and a new editor. RTE-III (extended memory) and multi-user real-time BASIC represent the latest additions to the RTE family. RTE-III is described in the article on page 21. Multi-user real-time BASIC will be described in a later issue of the Hewlett-Packard Journal.

References

1. J. Martin, "Design of Real-Time Computer Systems," Prentice-Hall, Englewood Cliffs, N.J., 1967, p.5.
2. S. Dickey, "Distributed Computer Systems," Hewlett-Packard Journal, November 1974.

Van Diehl *Kenneth A. Fox*

system to allocate a class number, if available, and the request to complete immediately. Logical unit zero is a dummy I/O device.

- When the WRITE/READ call completes, PROGA's data will have been placed in the buffer and this fact recorded in the completed class queue for this class.
- PROGA then schedules PROGB, the program receiving the data and passes to PROGB, as a parameter, the class number it obtained.
- When PROGB executes it picks up the class number and issues a Class I/O GET call to the class. PROGA's data is then passed from the system buffer to PROGB's buffer.

Another application of Class I/O is in the operation of the SPOUT program (see below).

Program Development

Before a newly delivered system can become useful (unless its intended use is program development) programs must be developed to solve the users' problems. The development of programs will, in most cases, continue for the life of the system as the user expands or changes his processes.

Program development proceeds as shown in Fig. 4. The program is written and translated into a machine-readable format. It enters the language processor (compiler or interpreter); if errors are found the source code is edited. The code from the language processor is combined with library subroutines and

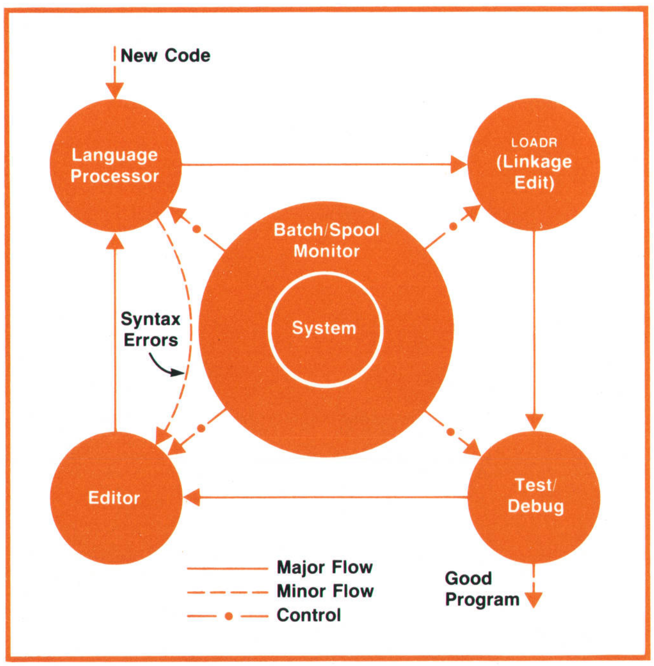


Fig. 4. *A model of the program development cycle, showing the utility programs that play roles in it. For RTE-II, major improvements have been made in the editor and loader, and a new batch/spool monitor has been developed.*

linked to the system. The resultant program is tested for correct function. In the rare case where it passes all tests, the program is "developed" and activity on it stops here until a failure is discovered by unsatisfied users or a logic error appears. Fixes are made to the source program to correct logic or design errors or to add features. The development loop now closes by going back through the language processor.

While traversing this loop we invoked a language processor, a loader, the user's program, and an editor. To ease the path around the loop in RTE-II we provide a high-level set of control programs, the batch/spool monitor. In the RTE-II development project considerable effort went into enhancing the editor, the loader, and the batch control capability.

The RTE-II system has a new *program* editor, designed to make it easy to edit programs (it comes in second best on text). The editor is inherently string and line oriented. It can find, replace, and delete strings. It can easily insert, replace, or delete characters in a line. It talks to the file system and it is *fast*.

The loader was enhanced in control capability, but the primary effort was aimed at improving its speed. To this end the system generator now provides a dictionary for all library entry points, and a study of where the loader spent its time led to faster symbol table search routines.

System enhancements for batch/spool consist of an LU switch capability, a batch clock and a break request. LU switch is a mechanism that allows programs running under control of the batch monitor to talk to a given logical unit while the actual LU is some other device. The batch monitor sets up the switches in a table that is accessable by the I/O system. Only programs running under the batch monitor are switched. This allows the batch monitor to switch output for the printer, for example, to a spool file from which it will be printed at a later time. The batch clock is an execution-time clock that is advanced every time the system clock is advanced (each 10 milliseconds), but only if a batch program is running. If the batch clock goes to zero it indicates a run-time limit has been exceeded and the offending program is aborted by the system. Batch elapsed time is not kept, since it is meaningless in a multiprogramming system. The break request is a system request which sets a flag for the specified program. The program may examine this flag and take any action it deems appropriate. When the batch monitor sees this flag it will abort any job it is running, or, if not in a job, will stop whatever it is doing and go back to the terminal for commands.

Batch/Spool Capability

The RTE-II batch and spooling capability is an extension of the file management package of RTE. The

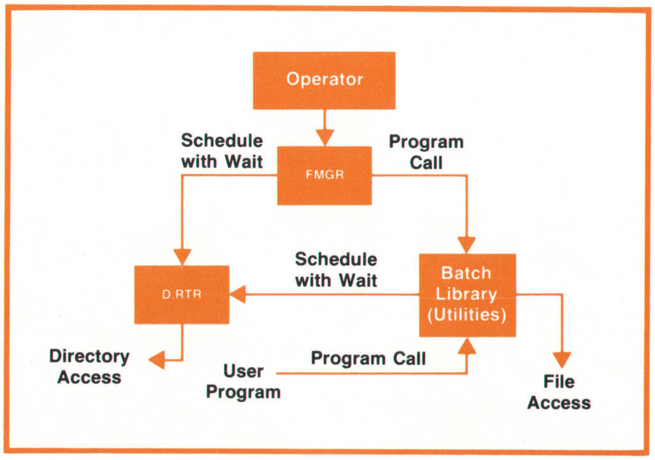


Fig. 5. *RTE-II batch and spooling capabilities are an extension of the RTE file management package, an optional set of programs and utility subroutines. FMGR is the interface between the user and the file system. D.RTR manages the file directory. The batch library handles program calls to the file system.*

file management package consists of a set of programs and utility subroutines that are physically optional and independent from the RTE operating system (see Fig. 5). The utilities are callable from user programs. The background program FMGR provides the interactive and/or batch interface between the user and the file system. The program D.RTR manages the file directory.

To add more extensive batch capability and a spooling function the interactive capability of FMGR was extended to provide such features as global parameters, which give the user the ability to write command procedures. Second, the FMGR command set was enlarged to provide commands for specific control of spooled operation. Finally, programming was added to effect input and output spooling for increased throughput.

Global parameters may be substituted for parameters in any of the FMGR commands. When the system encounters a global parameter it goes to a lookup table to get the current value of that parameter. Some global parameters may be set by the user and others are used by the system.

A typical transfer file, or command procedure, using global parameters might look like this:

```
:ST,1G::2G, 1G::3G
:PU,1G::2G
:TR
```

This set of commands could be placed in a file named MOVE. Then the command :TR, MOVE, TEST, 2, 10 will cause the file named TEST to be moved from cartridge number 2 to cartridge number 10. The user has supplied the values TEST, 2 and 10 for global parameters 1G, 2G, and 3G, and these values are put in the lookup table upon execution of the TR command.

Commands added to FMGR allow the user to set global parameters and do arithmetic and logical operations on them, to do conditional branching, and to print messages on various devices.

Spooling, or buffering of input and output job streams on the disc, was developed to increase throughput of the system while running tasks in batch mode. The spooling package is an option to the file management package, which itself is an option to RTE.

Input spooling in the RTE system is the reading of jobs from low-speed I/O devices to the disc, from which they are executed. Output spooling is the writing of job output to the disc and from there to the I/O devices. Spooling allows jobs to run at disc I/O speed instead of slower card reader or line printer speeds.

Tracing the progress of a batch job through the system makes clearer the interaction between the various pieces. Batch operation without spooling is quite simple and can be represented as shown in Fig. 6a. Note that job commands are read by FMGR directly from the input device and output is done directly to the output devices. This ties up the devices during processing and limits the job to the I/O speeds of these devices.

The addition of spooling to batch operation complicates the picture. Fig. 6b represents batch operation with the addition of spooling.

The important feature represented by Fig. 6b is that the operations of inspooling, batch processing, and outspooling take place in parallel. Note that input is now read directly by the inspooler JOB and written to spool files, one job per file. The operator runs JOB rather than FMGR. First JOB calls SMP to assign a spool access information table and associated unit number to the file and open it for I/O. Thereafter, JOB writes to this assigned unit as if it were a standard I/O device, and the writes are translated to the spool destination. When a job is completely read in, JOB puts a notation of this job on the job queue (in JOBFIL) and stores its location information in JOBFIL. JOB schedules FMGR to start processing (unless it is already executing) and then continues to inspool other jobs. When FMGR is ready to process a job, it searches JOBFIL for the highest-priority job and prepares it for processing. It sets up spool files for standard input and output units and puts the spool unit numbers into the batch LU switch table, which equates two units for the duration of the batch job. Thereafter, requests to these standard units will be translated to spool units and ultimately spool files. The program SMP monitors the created files, maintaining an outspool queue of files (in SPLCON) to be dumped for each device. It sends instructions to SPOUT, which runs continuously, by means of Class I/O telling it when to start files or try to lock a device in preparation for outspooling.

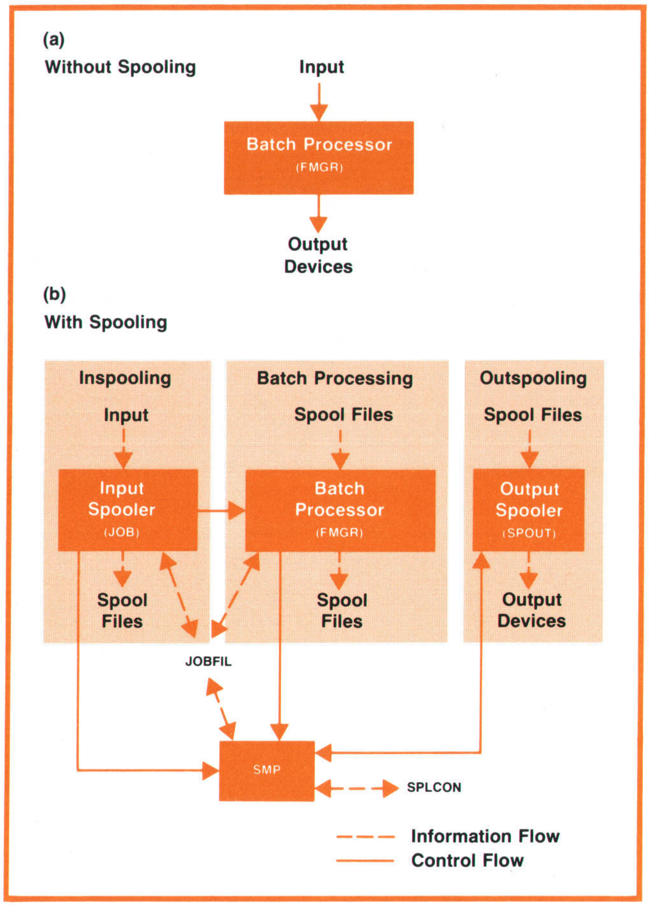


Fig. 6. *Input/output with and without spooling. To use spooling, the user runs* JOB *instead of* FMGR. JOB *writes input to spool files on the disc and stores their locations and priorities in* JOBFIL. FMGR *then processes the jobs according to priority.* SMP *monitors the spool files, maintains an outspool queue (in* SPLCON*) for each device, and sends instructions to* SPOUT*, the output program.*

The spooling capability may be tapped from outside the batch stream, although not automatically. A program may spool its output by following very much the same procedure that FMGR must follow to prepare a job to run under spooling.

Several of the new system features of RTE-II have been instrumental in the implementation of spooling. The resource number capability is used to control access to the spool control files. The LU locking capability allows the outspooler (SPOUT) to lock the devices it is dumping to for the duration of time it takes to output a single file. When spooling is used, the output from each job is guaranteed to be dumped in one piece.

Class I/O enables a particular implementation of SPOUT, which handles simultaneous outspooling to several devices and keeps several I/O requests pending for each device. Output to devices is written using class write and control requests; completion of these requests is indicated by a successful class GET. Identification of the file SPOUT is dumping and of the destination device is carried in the extra parameters of the class request. Stored in the access table of the file being dumped is the number of I/O requests pending. When SPOUT starts dumping a file, it reads and writes (using Class I/O) four records, increasing the pending count each time a record is written. Thereafter, the count is decreased each time a successful completion is indicated and increased (up to 4) each time a record is written. The count determines the program flow between the GET requests and the read/write loop.

Passage of blocks of information is also carried out through use of class write/read requests to LU#0 (dummy). SPOUT, in addition to detecting completion of writes, receives all its operating information through the same GET request. SMP write/reads the SPOUT control information on the same class that SPOUT uses to control the I/O devices. SMP also receives spool file information for spool setup using a class write/read on a different class.

The batch timer allows FMGR to keep track of the amount of time a job or program takes by sampling the timer contents at the beginning and end of a job. The user may also set time limits on jobs and programs running under the jobs so that these will be terminated if still running at the end of their limit.

Background swapping is necessary for batch operation, since FMGR must run user programs which are most likely background disc resident. This implies that FMGR must be swapped out.

It is batch LU switching that attends to translating I/O requests generated by batch processing from the "normal" LU to the spool LU corresponding to the appropriate spool file. This feature allows transparency of spooled operation to the programs running under batch.

General Enhancements

Besides its multi-terminal and batch/spool capabilities, RTE-II embodies a number of general and performance enhancements. General enhancements were made in the areas of memory management, swap control, power-fail/auto-restart, and microcode subroutine replacement.

When doing output to a buffered device the previous RTE system would allow all of memory to be used by that one device. This meant for example, that if a file was being punched all free memory would be filled with punch data. Furthermore, each time memory became available all contending users would be reactivated regardless of whether there was enough memory to satisfy any of the users. This allowed a low-priority program to lock out a higher-priority program requiring a larger block of memory. The low-priority program would use all the short blocks of memory and thus not allow any larger block to accumulate. To solve this problem the system now keeps track of the amount of memory each waiting program needs and reactivates all programs waiting for memory only when it has enough memory to satisfy the highest-priority waiting program. This allows high-priority programs to bid successfully for large blocks of memory. The system also enforces upper and lower buffer limits on memory queued on any I/O device. When a program makes an I/O request to a device which already has more than the upper-limit number of words of buffer memory queued on it the program is put in a buffer limit suspension. When an I/O device completes a request and causes memory to be returned, a check is made to see if the number of words of buffer memory in the device's queue is less than the lower limit. If it is, all programs in buffer limit suspension on this device are reactivated. This results in a kind of hysteresis that allows lower-priority programs enough time to do useful work before they are swapped out, while still keeping the I/O device busy (see Fig. 7).

We have also optimized the memory management routine to cut down system overhead. This was done by minimizing code within loops (usually at the expense of extra code outside the loops), and by keeping track of the largest block of memory available to allow rejection of requests for unavailable memory without an exhaustive search. Because constantly keeping track of the largest block could become time-consuming, a modified algorithm is used. Whenever memory is returned a check is made to see if the resulting block, after mergers with any contiguous memory, is larger than the largest known block. If so, the block is the new largest block. We don't change this value when memory is allocated, however. This means the system may have less memory than it thinks it has and therefore it will attempt to find memory for a request it cannot satisfy. But it can update the current largest-block information at the end of an unsuccessful allocation attempt and thus prevent any further fruitless searches. This turns out to be more efficient than searching for a new maximum block after each allocation.

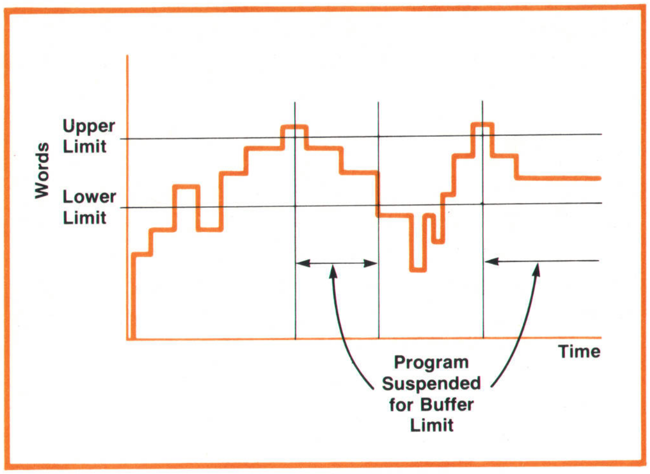


Fig. 7. *Improved memory management is a feature of RTE-II. The system enforces upper and lower buffer limits on memory queued on any I/O device. A program making an I/O request will be suspended if the buffer memory queued on the requested device exceeds the upper limit. Suspended programs are not reactivated until the queued memory drops below the lower limit. This helps give low-priority programs time to do useful work before being swapped out.*

When background swapping was implemented it became clear that some programs would not run if they were swapped, usually because of timing considerations. To solve this problem a memory lock request was added. This allows a program to request of the system that it not be swapped out of memory. In some installations this could prove undesirable, so a switch must be set at generation time to allow the system to service the memory lock request. We also found that most background programs used undeclared memory (memory between the last word used by program code and the last word in the program's area) for such things as symbol tables. For this reason a swap option has been included to swap all of the area or only the declared memory. This option is defaulted to all of memory for background programs and to only declared memory for foreground programs, but a system request is provided to alter the option.

Power-fail/auto-restart routines were developed which, while independent of specific I/O devices, yet restart all restartable devices. Also, a program is run at power-up which sends a power-failed message to all the terminals. This program is written in FORTRAN and its source code is provided with the system so the user may modify it to do special things for his installation.

The proliferation of microcode subroutine replacements had gotten to the point where a fair amount of time and memory was spent just calling and executing dummy subroutines to replace the invocation with an op-code. For example, when a multiply subroutine was replaced by microcode, the multiply software would be replaced by a dummy subroutine consisting principally of the op-code corresponding to the new microcode. The RTE-II system solved this problem by having the generators and the on-line loader replace the invocations at generation or load time. The user need only type in the entry point and its microcode replacement op-code and the system takes care of the rest.

Performance Enhancements

Several changes were made to the system to improve performance and reduce system overhead.

A more efficient time-keeping routine keeps the time of day in tens of milliseconds. Time is kept in double-word integer format, cutting the memory required from four words to two words. More importantly, it puts time in one base (hours was kept in base 24, minutes and seconds in base 60, and tens of milliseconds in base 100). This makes it twice as fast to test programs in the time list and makes updating their next run-time considerably easier and faster.

The dispatching algorithm was modified to correct false starts. The system may be loading a program when a higher-priority program is scheduled. In this case the previous system would finish the load and then swap the program without running it. RTE-II will either abort the load or, if the program is already in core, simply overlay it—it wasn't run, so the copy on the disc is still intact.

The dispatching algorithm was also modified in several areas to eliminate redundant processing. The system no longer checks for a possible content switch (i.e., changing the executing program) unless a change in some program's status occurs. Also, once a decision has been made that a given program is not swappable, no further swap checks are made for that area on the current pass through the dispatcher. Previously all programs contending for an area would force a swap check.

The dispatching algorithm was also modified to detect when a program being swapped has priority over the contender and, even though not currently executing, is scheduled to run within a short (user-selectable) time. In this case the swap is not done since to do so would likely result in a reswap to get the program back into memory before the contender has any cpu time.

Having done all these things we were worried about the amount of memory used in the system. To address this problem we optimized the operating system code so that, in most cases, it uses less memory than the previous system. We also shortened some of the memory-resident tables, thus freeing considerable memory. The result is that the system is only a little larger than the previous system and does more things faster and better than before.

Acknowledgments

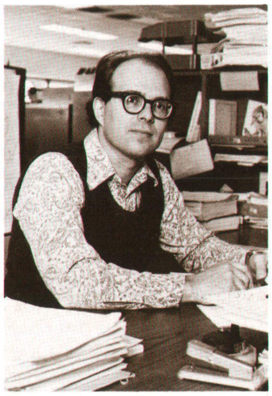
We are indebted to many people who provided guidance and help during the project. In particular, we wish to acknowledge Prem Kapoor for the work on the loader; Gene Wong for work on memory management and other loose ends; Ray Brubaker, Linda Averett, Marge Dunckle, Gil Seymour, and Dave Snow, who all helped translate the results into a useful product; Van Diehl for his capable product management; Steve Stark, Christopher Clare, Pete Lindes, Joe Schoendorf, and others, who provided ideas that shaped the final product; Shane Dickey and Earl Stutes for their help defining and using Class I/O; Tom Sapones and Dick Cook for work on the editor; Larry Pomatto for his help and support in providing hardware; Mike Chambreau, Ken Fox, and Gary Smith for their management; Joe Bailey, John Trudeau, Doug Baskins, and the rest of the support group for their ideas and prerelease control and testing efforts.

References

1. "Modular Systems for Sensor-Based Data Acquisition and Control," Hewlett-Packard Journal, August 1972, page 15.
2. S. Dickey, "Distributed Computer Systems," Hewlett-Packard Journal, November 1974.
3. Hewlett-Packard Journal, October 1971.
4. Hewlett-Packard Journal, October 1974.

Adele M. Gadol

Adele Gadol was responsible for the batch/spool portion of RTE-II. Born in New York City, she attended the University of Massachusetts and the University of Michigan, graduating from the latter in 1969 with a BS degree in mathematics. During the next three years she worked as a programmer and continued her studies at the University of Michigan, receiving her MS degree in computer, information, and control engineering in 1972. She joined HP the same year. Adele and her husband, an HP software designer, live in San Jose, California. She's an active member of the local chapter of ACM, and enjoys music (she plays flute), tennis, and swimming.



George A. Anzinger

George Anzinger has been improving and expanding the HP RTE system since 1971. He developed the moving-head system software and the file management package for RTE, and did most of the system modifications for RTE-II. George spent four years in the U.S. Navy before enrolling at the University of Wisconsin, where he earned his BSEE degree in 1968. He received his MSEE from Stanford University in 1969 and joined HP the same year. The Anzingers—George and his wife and their two small daughters—make their home in the Santa Cruz Mountains, a few miles from George's office at the HP Data Systems Division in Cupertino, California.

Real-Time Executive System Manages Large Memories

RTE-III does everything other HP real-time executive systems do, and adds large-memory management (up to 256K words) using HP's dynamic mapping system.

by Linda W. Averett

RTE-III IS A MULTI-PARTITION, real-time, multiprogramming operating system that supports up to 256K words of main memory. The latest in a series of upward-compatible, field-proven RTE's for HP 21MX Computers, RTE-III provides the user with all the features of RTE-II (see article, page 12) plus the following additional features:

- Increased system buffer area
- Increased program area
- More program linkage area
- Greater multiprogramming throughput by allowing up to 64 disc-resident programs to be simultaneously resident in memory
- Greater user protection via the use of a hardware fence and a memory protect feature.

Uses Dynamic Mapping System

RTE-III uses the dynamic mapping system,[1] a hardware option for HP 21MX Computers, to perform the logical-to-physical mapping necessary to use more than 32K words of physical memory. The dynamic mapping system has a set of four maps, each of which consists of 32 hardware registers and describes a 32K address space in memory. The four maps are the system map, which is automatically enabled on interrupt, the user map, which is enabled by the system before passing control to a user program, and the port A and port B maps, which are automatically enabled during a memory transfer involving the dual-channel port controller (DCPC).

A 15-bit address, sufficient to address 32K words of memory, is used in HP 21MX Computers. When the dynamic mapping system is enabled, this 15-bit address is split into two parts. The lower ten bits of the address become a relative displacement in a page in memory. The upper five bits of the address specify one of the hardware registers in the map that is currently enabled. The address of the physical page in memory is picked up from the indicated map register, and the page displacement is appended to it. Thus, the target address is derived by a mapping from a 32K logical memory space to a physical memory as large as 256K words. This mapping process does not slow down memory accesses.

Memory Organization

Physical memory is organized into building blocks (see Fig. 1). The base of the building block structure, beginning at physical page zero in memory, consists of the system links and communication area, the operating system, and the resident library. The first building block is the common area, followed by the memory-resident program area and the system available memory area. The remaining memory is divided into partitions that are used for executing

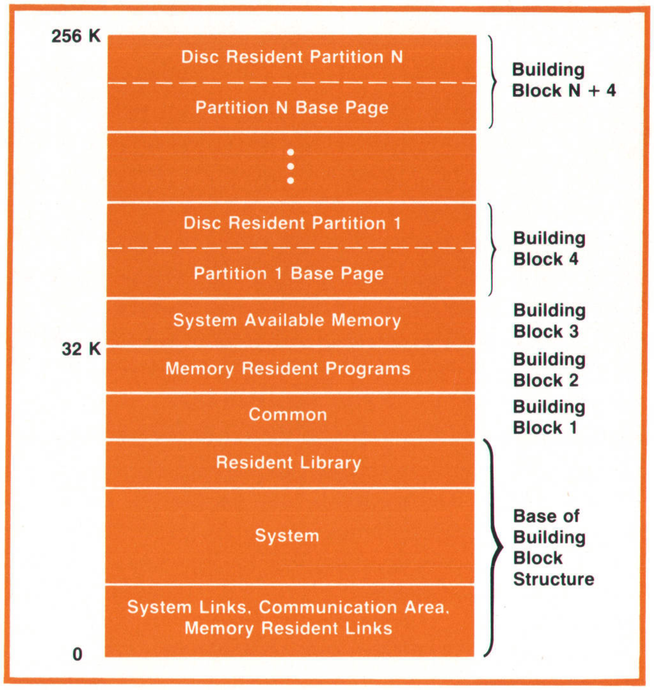


Fig. 1. *The RTE-III real-time executive operating system manages up to 256K words of physical memory, arranged into building blocks as shown. Sizes of the building blocks are determined by the user at system generation time within certain minimum and maximum limits.*

RTE-III Definitions

Map—a set of 32 hardware registers in the dynamic mapping system. Used to translate a logical 32K address space to a 32K segment of a 256K physical address space.
Page—1K (1024 decimal) words of memory.
DCPC—Dual-channel port controller with two assignable channels for performing direct memory accesses.
Partition—a fixed area in memory consisting of a user-determined number of pages. It is used for executing disc-resident programs.
Disc-Resident Program—a program that resides on the disc and must be loaded into memory to be executed.
Memory-Resident Program—a program that is always resident in main memory.
Swap—the action of writing a disc-resident program that is executing in memory onto the disc so another disc-resident program can be loaded and execute in that same area.

disc-resident programs. The user may determine the size of all these building blocks at system generation time within certain minimum and maximum limits.

The building blocks of physical memory can be arranged into different structures within the logical address space by use of the dynamic mapping system. At any instant a 32K logical address space is described by the map that is enabled. A key benefit is that the building blocks do not all have to fit into this 32K space at the same time. Thus, the individual blocks do not detract from the address space of the other blocks.

Fig. 2 indicates what the 32K address space may look like when the system map or the user map is enabled. All execution of code takes place under one of these two maps. The two DCPC maps are used only during a high-speed direct memory access.

Multi-Partition System

While RTE-III provides larger user areas by means of the dynamic mapping system, its major benefit is increased multiprogramming throughput. RTE-III can have up to 64 partitions. Thus at any instant, 64 disc-resident programs, in addition to the memory-resident programs, can be resident in memory. Being able to have more than two disc-resident program execution areas (partitions) decreases the probability of having to do program swapping. It is approximately 100 times faster to switch between two programs that are resident in memory than it is to swap using a 7900A Disc Drive (50 times faster for a 7905A Disc Drive). Thus in a multiprogramming environment, multiple partitions can greatly decrease the amount of time necessary to switch between programs.

The multiple partitions also improve the response of the RTE multi-terminal monitor (see article, page 12), because it is more likely that there will be memory available for the monitor when it is required.

Memory Management

The memory available for program execution is divided into two areas. One is the memory-resident program area, which is established at system generation time and does not change, and the other contains up to 64 partitions for program execution.

Any disc-resident program may be assigned to run in any partition that is large enough. If a disc-resident program is not assigned to a partition, it will be dispatched into any partition that is available and is big enough. If a partition is not available, then the allocated partitions will be examined to determine if one is swappable.

To give the user more control over which programs compete for memory, RTE-III provides for defining two types of partitions, real-time and background. There is no functional difference between these partition types, but unless a program is assigned to a specific partition, it will run in a partition of the same type. In other words, by default, real-time programs will run in real-time partitions, and background programs will run in background partitions.

Thus the user has the following capabilities for controlling partitions:

- Up to 64 partitions of varying lengths can be defined
- Partitions can be separated into two types
- Programs may be assigned to a specific partition
- Programs may be locked into a partition
- Partitions may be reserved for assigned programs.

Dispatching

RTE-III keeps track of the type and size, the allocation status, and the priority and status of the resident of each partition. When a disc-resident program is ready to be executed, the system checks first to see if the program is already resident in a partition. If it is, the hardware user map registers are loaded with the addresses of the physical memory pages that make up that partition, and the program is given control. If it is not resident, the system checks to see if the program is assigned to a partition. If so, and if that partition is free, the program is loaded into it and dispatched. If the partition is not free, the system will determine if a swap is possible.

If the program is not assigned to a partition the system will find the smallest free partition that is long enough for the program. If a free partition does not exist, the system looks for the partition that is long enough and contains the lowest-priority resident that qualifies for a swap. If a suitable partition is found, the user map is loaded with the addresses of the memory pages in that partition, the swap (or load if the partition was free) is performed, and the program is given control.

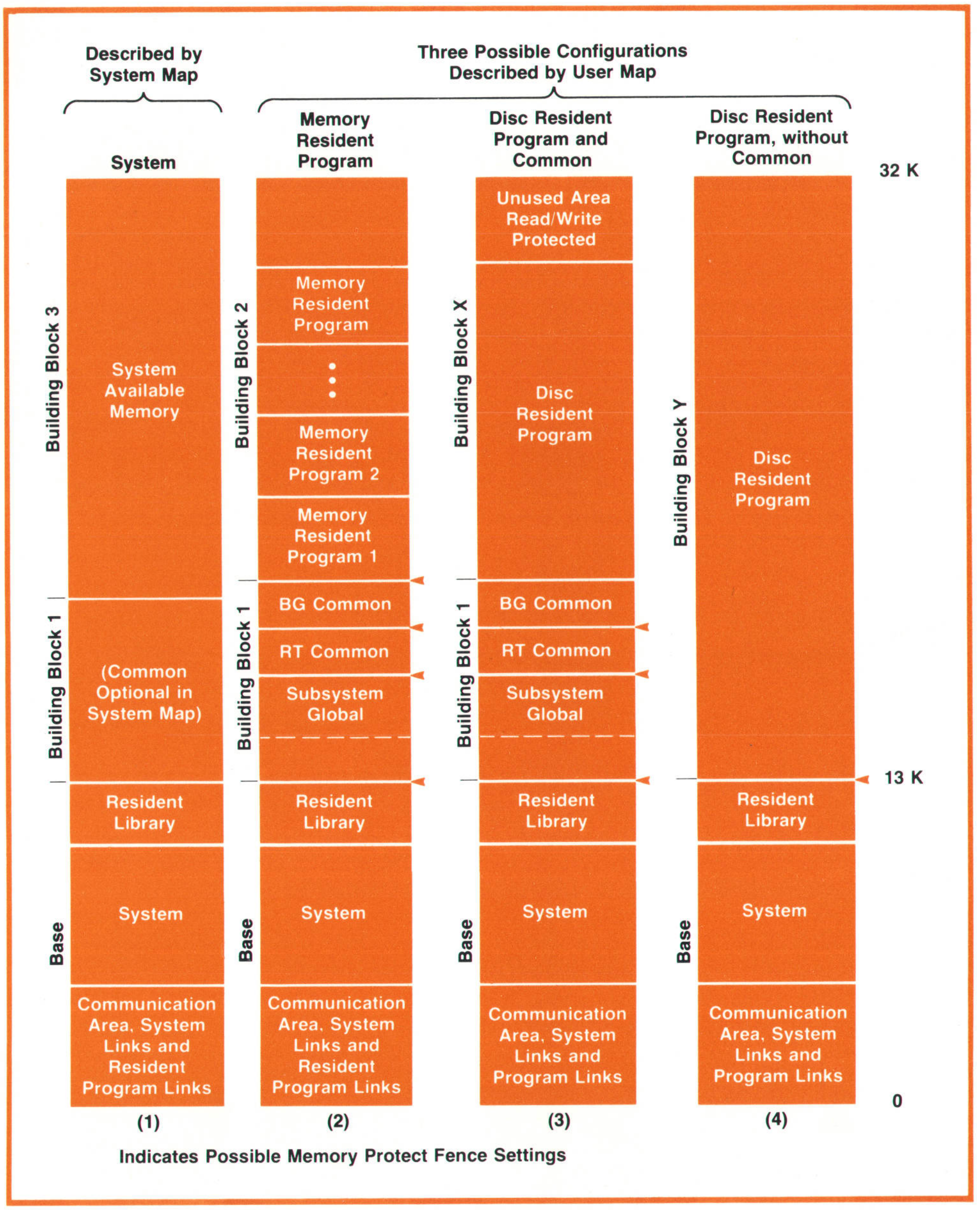


Fig. 2. *RTE-III uses the dynamic mapping system, a hardware option for HP 21MX Computers, to configure each program's 32K logical memory space using the building blocks of physical memory. All execution of code takes place under either the system map or the user map. Each map is a set of hardware registers whose contents tell how to translate logical memory addresses to physical memory addresses.*

When a suitable partition cannot be found, the program will remain scheduled and the system will try to dispatch it the next time it scans the scheduled list. This is why multiple partitions speed up multiprogramming throughput. The more partitions the system has, the less the probability that a swap will be necessary to execute the program, and the less the probability that the program will have to wait on memory.

Because memory-resident programs are always in memory, the system does not have to locate a partition to dispatch these programs. When a memory-resident program is ready to execute, the system loads the user map and gives control to the program.

Fig. 2 shows three possible configurations of the 32K logical address space that can be described by the user map for memory-resident and disc-resident programs.

Program Protection

RTE-III provides greater program protection than the other RTE systems. The memory protect fence is used, as it is in RTE-II, to provide protection on the lower boundary of the program. This hardware fence is set each time a program is dispatched; it prevents a a user program from writing into any memory location below the fence.

In addition to the memory protect fence, RTE-III protects all pages of memory that a program does not use in the 32K address space described by the user map while the program is executing. This prevents a user program in one partition from destroying a program in another partition.

Input/Output System

Before entry into any driver, RTE-III will determine which map, user or system, is necessary to process the I/O. Then the system will load the proper map and enable it. If the device requires a DCPC channel, the system will load the proper DCPC map. Thus the standard I/O drivers are not required to do any mapping and therefore are compatible across the entire RTE line of systems.

The fact that DCPC transfers occur under their own map enhances multiprogramming throughput. While a program in one partition is I/O suspended during a DCPC transfer, the user map can be set up to describe another program executing in another partition. If the DCPC transfer had to take place under the user map, no other program could execute during the transfer. Thus having a map for each DCPC channel in addition to a map for the user program and one for the system increases the efficiency of computer use.

Acknowledgments

RTE-III is a product of the combined efforts of many people. Special thanks go to Ray Brubaker and Eugene Wong, development engineers, who contributed greatly to the design and development of the product. Also, Jim Bridges, production engineer, Van Diehl, product manager, Carl Davidson, quality assurance, Joe Bailey, systems engineer, Joan Martin, technical writer, and Jim Bechtold, technical writer, contributed much time and good work toward making RTE-III a reality.

Much appreciation goes to Jack Elward, designer of the dynamic mapping system, and to Cle Riggins, 21MX project manager, for their technical assistance. Also, Carl Ubis put in a lot of hard work maintaining the equipment and setting it up.

Reference

1. J.S. Elward, "The Million-Word Minicomputer Main Memory," Hewlett-Packard Journal, October 1974.

Linda W. Averett

Linda Averett was project manager for RTE-III. She came to HP in 1974 with four years' experience in the design of real-time operating systems. A native of Knoxville, Tennessee, she graduated from the University of Tennessee in 1970 with a BS degree in engineering physics. She's married, lives in Sunnyvale, California, and enjoys tennis, swimming, and scuba diving.

HP 92001A Real-Time Executive System II (RTE-II)

FEATURES

Foreground and background multi-user swapping partitions.
Operation in as little as 16K of CPU memory, or up to 32K for user's real-time applications and RTE-II supported capabilities.
Supports cartridge disc subsystems providing 4.9 to 118 Mbytes of on-line storage with optional file management to provide ample capacity for programs and a fast-access data base.
Concurrent processing and program development in FORTRAN II/IV; Conversational Multi-User Real-Time BASIC (optional), ALGOL, and HP Assembly language.
Multi-terminal access to all system resources, serving multiple users concurrently.
Optional input/output spooling to disc to speed throughput without excessive use of CPU memory for buffering.
Powerful interactive editor to aid program development.
Supports coordination of distributed multiprocessor communication networks.
Supports data communication with IBM 360/370 or HP 3000.

ORDERING INFORMATION

RTE-II is offered as a choice of A-series operating system options for 9600 systems. RTE-II is also available as follows:
92001A RTE-II Software Package
92001A-Y13 Batch-Spool Monitor
92001A-Y15 Multi-User Real-Time BASIC

PRICE IN U.S.A.:

92001A RTE-II, $4000
92001A-Y13 Batch Spool Monitor, $1000.
92001A-Y15 Multi-User Real-Time BASIC, $1000.

HP 92060A Real-Time Executive System III (RTE-III)

FEATURES

Up to 64 separate multi-user swapping partitions, up to 19K words per partition for fast response to needs of many multiple users.
Manages 32 to 256K of CPU memory for user's real-time applications and RTE-III supported capabilities.
Supports cartridge disc subsystems providing 4.9 to 118 Mbytes of on-line storage, with file management to provide ample capacity for programs and a fast-access data base.
Concurrent processing and program development in FORTRAN II/IV, Conversational Multi-User Real-Time BASIC (optional), ALGOL, and HP Assembly language.
Multi-terminal access to all system resources, serving multiple users concurrently.
Input/output spooling to disc to speed throughput without excessive use of CPU memory for buffering.
Powerful interactive editor to aid program development.
Supports coordination of distributed multiprocessor communication networks.
Supports data communication with IBM 360/370 or HP 3000.

ORDERING INFORMATION

RTE-III is offered as a choice of A-series operating system options for 9600 systems. RTE-III is also available as follows:
92060A RTE-III Software Package
92060A-Y15 Multi-User Real-Time BASIC

PRICE IN U.S.A.:

92060A RTE-III, $6000. Includes Batch Spool Monitor.
92060A-Y15 Multi-User Real-Time BASIC, $1000.

MANUFACTURING DIVISION: DATA SYSTEMS DIVISION
11000 Wolfe Road
Cupertino, California 95014 U.S.A.

Hewlett-Packard Company, 1501 Page Mill Road, Palo Alto, California 94304

Bulk Rate
U.S. Postage
Paid
Hewlett-Packard
Company

HEWLETT-PACKARD JOURNAL
DECEMBER 1975 Volume 27 • Number 4

Technical Information from the Laboratories of Hewlett-Packard Company

Hewlett-Packard S.A., CH-1217 Meyrin 2
Geneva, Switzerland
Yokogawa-Hewlett-Packard Ltd., Shibuya-Ku
Tokyo 151 Japan

Editorial Director • Howard L. Roberts
Managing Editor • Richard P. Dolan
Art Director, Photographer • Arvid A. Danielson
Illustrator • Sue M. Perez
Administrative Services, Typography • Anne S. LoPresti
European Production Manager • Michel Foglia

CHANGE OF ADDRESS: To change your address or delete your name from our mailing list please send us your old address label (it peels off). Send changes to Hewlett-Packard Journal, 1501 Page Mill Road, Palo Alto, California 94304 U.S.A. Allow 60 days.