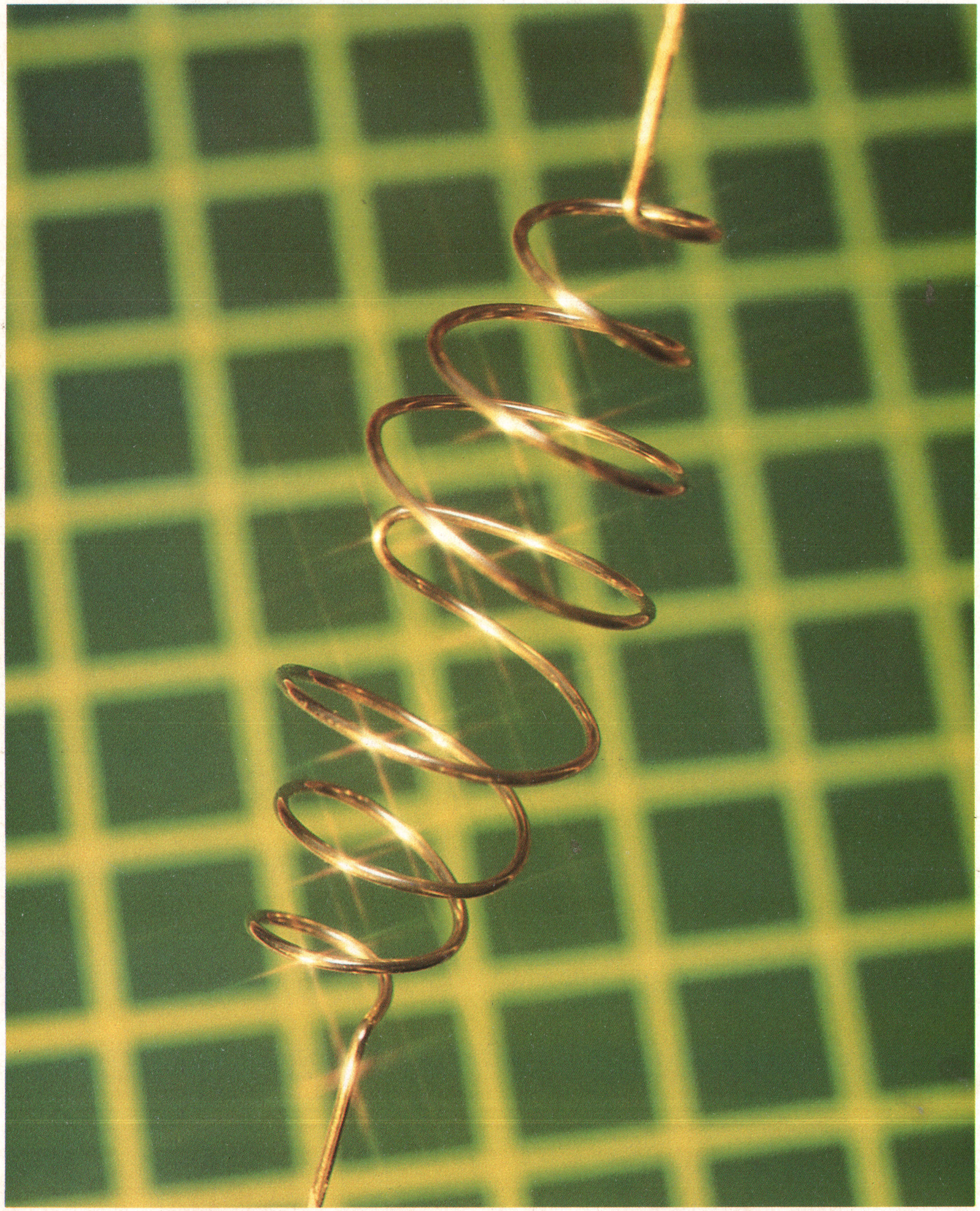
HEWLETT-PACKARD JOURNAL

DECEMBER 1987



HEWLETT-PACKARD JOURNAL

December 1987 Volume 38 • Number 11

Articles

4 **Vector Signal Generation and Analysis,** *by Allen P. Edwards*

6 **Hardware System Design for a Vector Analyzer,** *by Andrew H. Naegeli and Juan Grau*

15 **Quadrature and Phase Errors in Vector Demodulation**

17 **Firmware System Design for a Vector Analyzer,** *by Brian S. Messenger, Peter H. Fisher, and Stanley P. Woods*

25 **Vector Modulation in a Signal Generator,** *by David L. Gildea and Donald R. Chambers*

30 **Firmware for a Vector Signal Generator,** *by James E. Jensen and Eric D. McHenry*

34 **Low-Noise Synthesizer Design,** *by Thomas J. Carey, John C. Lovell, and Thomas L. Grisell*

39 **Digital and Vector Baseband Circuits for a Vector Signal Generator,** *by Chung Y. Lau*

41 **A GaAs IC Current Switch**

42 **Describing Signals in the I-Q Domain**

45 **A Wideband FM Subsystem for a Low-Noise Synthesizer Module,** *by Eric D. McHenry*

48 **Vector Modulator, Output Amplifier, and Multiplier Chain Assemblies for a Vector Signal Generator,** *by Wayne M. Kelly, Mark J. Woodward, Eric B. Rodal, Pedro A. Szente, and James D. McVey*

50 **Baseband Calibration**

53 **A Combinational Board Test System,** *by Michael E. Gravitz*

58 **Interactive Graphical Debugging Package**

68 **MPE XL: The Operating System for HP's Next Generation of Commercial Computer Systems,** *by John R. Busch, Alan J. Kondoff, and Darryl Ouye*

87 **HP 3000 Emulation on HP Precision Architecture Computers,** *by Arndt B. Bergh, Keith Keilman, Daniel J. Magenheimer, and James A. Miller*

Departments

3 **In this Issue**

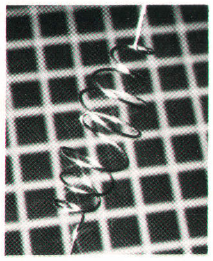
3 **What's Ahead**

65 **Authors**

90 **Annual Index**

Editor, Richard P. Dolan • Associate Editor, Business Manager, Kenneth A. Shaw • Art Director, Photographer, Arvid A. Danielson • Support Supervisor, Susan E. Wright
Administrative Services, Typography, Anne S. LoPresti • European Production Supervisor, Michael Zandwijken

In this Issue



This is our first bimonthly issue. From now on, the Hewlett-Packard Journal will be published six times a year, in December, February, April, June, August, and October. Your comments on this change are welcome. December will continue to be the annual index issue; the 1987 index is on page 90.

The cover photograph is of a three-dimensional model of a type of signal used in modern radar systems—the frequency chirp. The model is patterned after Fig. 12 on page 13, which shows HP 8980A Vector Analyzer displays of a rotated frequency chirp. The HP 8980A and the HP 8780A Vector Signal Generator are a pair of new instruments for testing digital radio, radar, and other systems that use vector modulation, a technique in which a carrier signal is modulated by one signal (I) and the same carrier shifted in phase by 90° is modulated by another signal (Q). The transmitted signal is the sum of the two modulated carriers. Vector modulation concepts are introduced in the article on page 4, and the designs of the HP 8780A and HP 8980A are described on pages 6 through 52. Among the design issues for the signal generator were the design of a low-noise carrier oscillator and the development of performance specifications and measurement methods in the absence of standards. The analyzer design provides two 350-MHz display channels, one for I and one for Q, along with several kinds of markers and measurement functions.

The Multiprogramming Executive, or MPE, is the operating system for the HP 3000 family of business computer systems. For the new HP Precision Architecture computers—the HP 3000 Series 930 and 950—MPE has been redesigned. MPE XL is a state-of-the-art operating system designed to realize the benefits of the architecture and provide a basis for future performance increases. Compatibility with previous MPEs and ease of migration to the new system for current users were major issues in MPE XL's development. In the file and transaction management systems, one of the key technologies is highly efficient support for mapped files, which allows megabytes of data to be accessed at main-memory speeds. The design of MPE XL is described in the paper on page 68. Compatibility features are discussed in more detail on page 87.

Very large-scale integrated (VLSI) circuits, application specific integrated circuits (ASICs), mixed analog/digital devices, and surface mount components are increasingly common in electronic manufacturing. To address the formidable testing problems posed by these devices and technologies, the HP 3065AT Board Test System has been developed. To the in-circuit tests of the earlier HP 3065 system, the HP 3065AT adds functional testing and new features to provide test engineers with new strategies and capabilities appropriate to the challenges they face. The HP 3065AT's design is the subject of the article on page 53.

-R.P. Dolan

What's Ahead

In the February issue, we'll have several articles on precision digitizing oscilloscope design based on waveform recorder technology. The products covered are the HP 5180T/U, 5183T/U, and 5185T Precision Digitizing Oscilloscopes. Also featured will be the HP Printed Circuit Design System software, and silicon-on-insulator (SOI) technology.

The HP Journal encourages technical discussion of the topics presented in recent articles and will publish letters expected to be of interest to our readers. Letters must be brief and are subject to editing. Letters should be addressed to: Editor, Hewlett-Packard Journal, 3200 Hillview Avenue, Palo Alto, CA 94304, U.S.A.

Vector Signal Generation and Analysis

This issue contains several articles describing the development of the HP 8780A Vector Signal Generator and the HP 8980A Vector Analyzer. To appreciate the nature of these products it is useful to discuss some of the concepts of vector modulation and its applications.

by Allen P. Edwards

MODULATION refers to the modification over time of some property of a signal. The modification can be of the signal's amplitude, as in AM or amplitude modulation. It also can be the change of some other property as in frequency modulation (FM) or phase modulation (PM). Signals are typically modulated for the purpose of encoding some information on the signal. This information may be in the form of voice, text, or in the case of radar signals, time (which is interpreted as distance or range).

In the last few years some modulation schemes particularly suited to digital communications have become increasingly important. One of the first of these is phase-shift keying (PSK). PSK can be thought of as digital phase modulation in which there is a fixed set of valid phases. In the case of 4PSK (QPSK) there are four phases spaced 90 degrees apart. Each valid signal location is called a signal state and in the case of QPSK, there are four states. The signal changes state in response to digital inputs. The signal will transition between the states in the shortest path, which is not the path of constant amplitude. Thus, PSK causes both amplitude and phase variation in the signal unlike conventional phase modulation which is confined to constant amplitude.

Another popular form of modulation is quadrature amplitude modulation (QAM). In this form of modulation two orthogonal carriers (sine and cosine components of a reference phase) are independently modulated to form n valid amplitude levels or states. The two components are considered the I (in-phase) and Q (quadrature) channels. The composite signal has n^2 states that form a rectangular pattern of dots when viewed in a graph of I versus Q. (This pattern is called a constellation diagram.) This modulation has multiple levels and phases, but just n I levels and n Q levels. Some examples are 16QAM (4×4) and 64QAM (8×8).

Vector Modulation

Vector modulation is simply the description of a modulation in terms of its I and Q components versus time. It can be used to describe all types of modulation, but is most useful in describing the rectangular modulations, which include AM, PSK, QAM, and pulse. It also can be extremely useful for analyzing FM and PM as well.

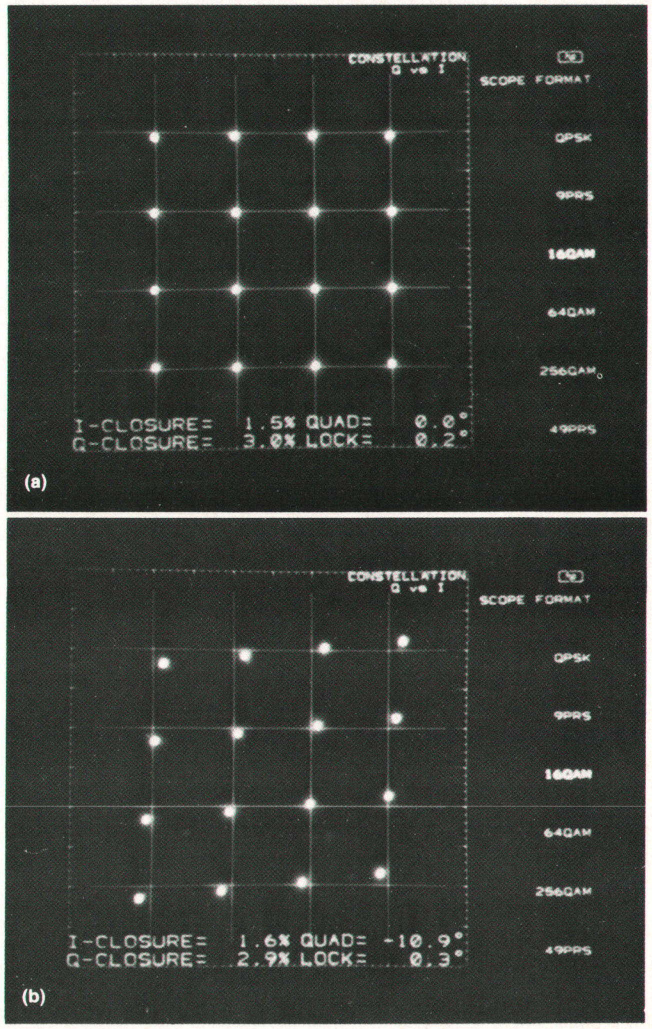
Vector modulation can be described by an equation with the form:

$$s(t) = i(t)\sin(\omega t) + q(t)\cos(\omega t)$$

This equation describes QAM modulation exactly. Let's see how it can also describe other types of modulation as well.

Amplitude Modulation: $s(t) = a(t)\sin(\omega t)$. Modulate the in-phase component with the desired modulation $i(t) = a(t)$ and set the quadrature term $q(t)$ to zero. Note that this equation also applies to pulse modulation.

Phase Modulation: $s(t) = \sin(\omega t + \phi(t))$. Represent s(t) as



```
I-CLOSURE=  1.5% QUAD=   0.0°
Q-CLOSURE=  3.0% LOCK=   0.2°
```

```
I-CLOSURE=  1.6% QUAD= -10.9°
Q-CLOSURE=  2.9% LOCK=   0.3°
```

Fig. 1. *Constellation diagram of a 16QAM radio (a) using the radio as a reference and (b) as it actually should look.*

the sum of its two rectangular components using $\sin(a+b) = \cos(b)\sin(a) + \sin(b)\cos(a)$. Therefore, $i(t) = \cos(\phi(t))$ and $q(t) = \sin(\phi(t))$.

Vector representation has a distinct advantage in describing modulation in that the representation is linear. Phase modulation has the problem that it is nonlinear (this is why Bessel functions often come up in its description). Vector representation is particularly useful in describing what happens when the modulation is passed through a filter. In the case of the traditional phase modulation description, RF filtering adds harmonic terms to the description. In the I-Q representation of phase modulation, RF filtering removes terms. Also, the effects of filtering on RF and baseband are equivalent in I-Q modulation where they are not in phase modulation.

The HP 8780A Vector Signal Generator is an IF and RF signal generator that covers an output frequency range of 10 MHz to 3 GHz. It provides a rich assortment of modulation types to the user as described in the articles on pages 25, 39, and 45. They include:

- FM: both wideband (FDM/FM and FMTV) and narrowband (low phase noise)
- Digital modulation: pulse, BPSK, QPSK, 8PSK, 16QAM, 64QAM, and others
- Analog vector modulation: I and Q inputs to 350-MHz bandwidth
- AM available either as a vector modulation input (350-MHz bandwidth) or on top of the digital modulation formats (to simulate fade). This AM has low phase noise consistent with the application's requirements.

Vector Analysis

The HP 8980A Vector Analyzer is basically a two-channel sampling oscilloscope with features designed for analysis of the I and Q outputs of a radio or demodulator. It has 350 MHz of X-Y (called I-Q) bandwidth. This allows the user to view repetitive events in the I-Q plane in real time. The HP 8980A is fully programmable. It has many measurements, markers, and features designed to make it easy to analyze modulated signals.

The HP 8780A and HP 8980A represent a new generation of signal generation and analysis products. They are required by the reality that users are no longer working with voice and simple pulse in their microwave applications. Digital modulation is a requirement in communications and radar applications of signal generators. Higher-bandwidth FM also is a requirement for communications and radar (wideband chirps are possible over more than 100 MHz with the HP 8780A). The analysis of these signals requires a fully programmable HP-IB (IEEE 488/IEC 625), high-bandwidth (350 MHz), high-resolution (up to 12-bit resolution) capability. The HP 8980A offers this.

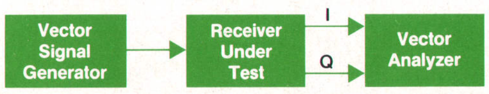


Fig. 2. *Recommended test setup for obtaining the correct constellation diagram of a 16QAM radio.*

Applications

In the area of digital radio (both satellite and terrestrial), the two products work together to test receivers. The HP 8780A serves as a calibrated transmitter with known and specified accuracy. This avoids the problem with home-built solutions created by unknown errors in the transmitter that mask and cancel the errors in the receiver. Fig. 1 shows the constellation diagram of a 16QAM radio as seen by a receiver that was adjusted with the radio as a reference and the constellation as it actually should look. A recommended test setup is shown in Fig. 2. As can be seen, a markedly out of adjustment radio may look good if not measured with a calibrated transmitter. In addition, home-built solutions have the hidden cost of developing NBS traceability, documentation, and supportability that always makes them more expensive than they are expected to be.

Another application uses the HP 8780A Vector Signal Generator as a calibrated standard to calibrate a demodulator for use as a test demodulator. The HP 8980A Vector Analyzer has built-in features that allow the compensation of external demodulator quadrature and phase errors as well as the more obvious offsets and gains.

Some other applications in radar testing include module testing, transient testing, time delay testing, and image rejection. These applications require coded pulses and the formation of extremely fast pulses or shaped pulses. With a minimum rise time of 1 ns and the ability to have baseband pulse-shaping filters, a wide variety of pulses can be formed by the HP 8780A. The FM deviations available in the generator allow chirp FM deviations. The combination of a calibrated demodulator, the HP 8980A, and a calculator allow the analysis of instantaneous frequency versus time in a chirp pulse.

Hardware System Design for a Vector Analyzer

by Andrew H. Naegeli and Juan Grau

TO MEASURE THE PERFORMANCE of many modern microwave systems, it is necessary to measure the instantaneous amplitude and phase of an RF signal. Digital radio systems, for example, use complex modulation signals that convey information in both amplitude and phase, and at very fast rates. Modulation rates of 60 megabaud and higher are common.

Most systems of this type use vector demodulation (also called I-Q demodulation, coherent detection, or quadrature detection) to recover the in-phase (I) and quadrature (Q) components of the signal. These in turn are used to detect the items of interest in the system. For example, the data decisions are made from the I and Q signals in QPSK (quadrature phase-shift keying) digital radios.

The HP 8980A Vector Analyzer (Fig. 1) is designed to measure instantaneous amplitude and phase of wideband RF and microwave signals by analyzing their I and Q components in the time domain. It is a fully programmable HP-IB (IEEE 488/IEC 625) instrument designed for R&D and production applications.

The amplitude and phase of a signal can be analyzed visually by plotting the Q component versus the I component graphically in Cartesian coordinates. The amplitude is then represented by the distance from the center of the plot and the phase is represented simply by the angle thereon. The polar displays of vector network analyzers work on this same principle.

Accurately displaying the I and Q signals from wideband systems is not so easy, however, because of the large bandwidth of the signals. Most analog display instruments have a limited bandwidth in the X axis. Typically the 3-dB bandwidth of conventional oscilloscopes is about 5 MHz in the X-axis, even if the Y-axis bandwidth is 200 MHz or more. This results in distortion of the vector display when fast signals are analyzed, as shown in Fig. 2. To overcome this difficulty the HP 8980A Vector Analyzer uses a sampling display system, with identical samplers in the I and Q channels. This technique provides identical wideband performance in both channels, where the bandwidth is set by the 1-ns rise time of the sampler circuits.

Using a sampled system also allows display of the constellation of a digital modulation signal. In a constellation display, the amplitude and phase states are viewed at a single time instant, which corresponds to the data decision sampling instant in a digital radio.

In addition to vector and constellation displays, the I or Q components can be displayed versus time for pulse timing measurements or for viewing eye patterns in digital radios. The timing is set like an oscilloscope, with time per division from 0.5 ns/div to 2 ms/div and delays up to 20 ms. Fig. 3 shows the vector and constellation displays of a 64QAM digital radio, as well as the I-versus-time display, which shows the eight-level eye pattern characteristic of this format.

Block Diagram

A block diagram of the HP 8980A is shown in Fig. 4. The instrument uses an analog CRT display to provide higher-quality vector and eye displays than a digital display system would give. The analog system can process data at a much faster rate, up to one million samples per second in this case, compared to about 20 thousand in a digital display. In addition, the analog display provides information in the intensity of various parts of the displayed wave, so the relative density of different transitions in a complex waveform can be determined visually.

Many digital display features are included also, like a selection of grids, markers, and text. These are made possible by the digital stroke generator circuit, which is part of a specially modified version of the HP 1345A Digital

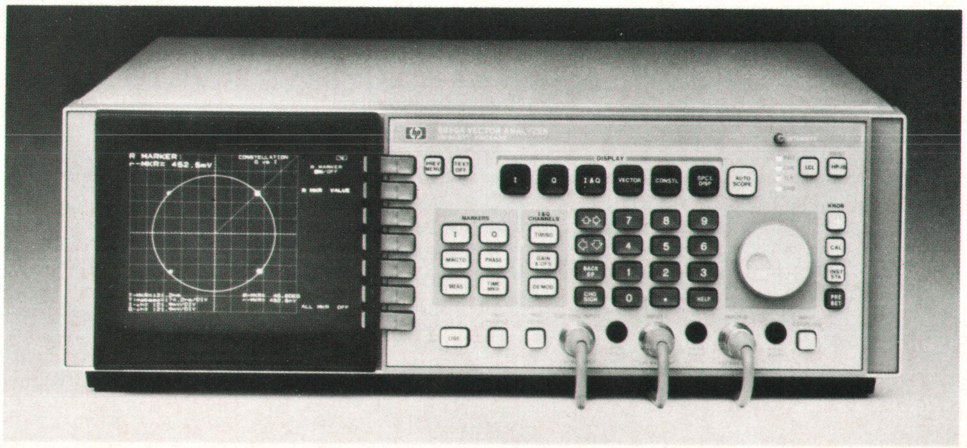


Fig. 1. *The HP 8980A Vector Analyzer is a 350-MHz two-channel X-Y sampling oscilloscope designed to characterize the in-phase and quadrature components of modern communications and radar system signals.*

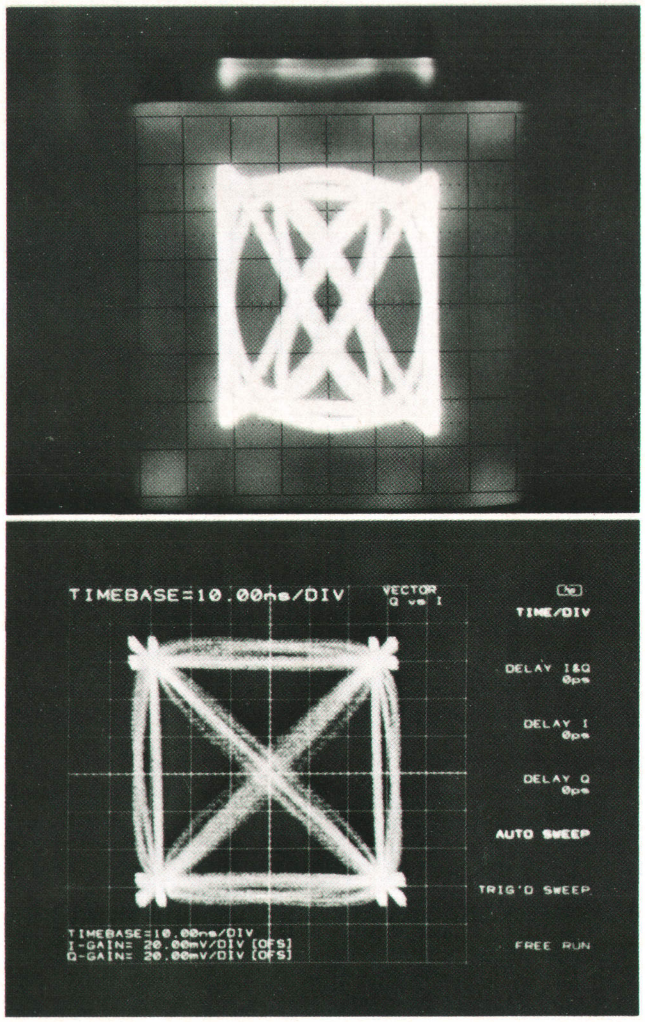


Fig. 2. *(Top) The narrower bandwidth of the X axis in conventional oscilloscopes distorts the X-Y display of wideband data. (Bottom) HP 8980A display.*

Display.[1] This CRT display system is included in the HP 8980A as a replaceable module. Digital inputs to the HP 1345A are sent on a 26-pin cable from the microprocessor controller. This system provides very powerful display control for displaying menus, markers, help text, and various user messages.

The X, Y, and Z outputs of the digital stroke generator are multiplexed with the X, Y, and Z outputs of the analog display system to display the analog and digital information together. The multiplexing is done on a printed circuit board outside the display module and the signal is sent back into the display module just before the deflection plate amplifiers. The relative timing of the refreshing of the digital information and the display of the analog signals is a complex process. This process is discussed in the article on the firmware design for the HP 8980A on page 17.

I and Q Input System

The I and Q inputs are processed before sampling by the system shown in Fig. 5. The input impedance is selected by using interchangeable BNC connector and series resistor

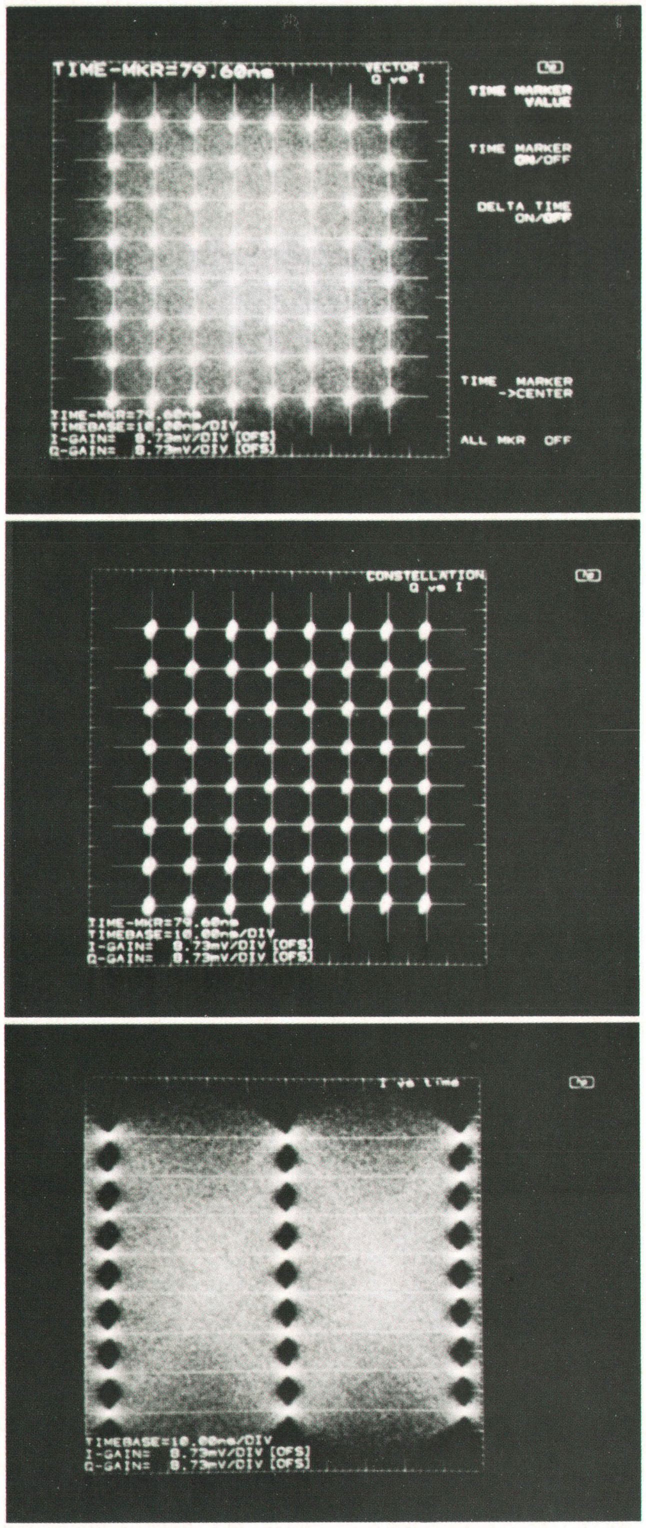


Fig. 3. *Vector (top), constellation (middle), and eye (bottom) displays for 64QAM radio.*

assemblies on the front panel. The 75-ohm connector uses a 25-ohm series resistor to bring the 50-ohm impedance of the rest of the instrument up to the 75-ohm level. The 50-ohm input connector uses a conductor in place of the series resistor. These assemblies are screwed into a socket

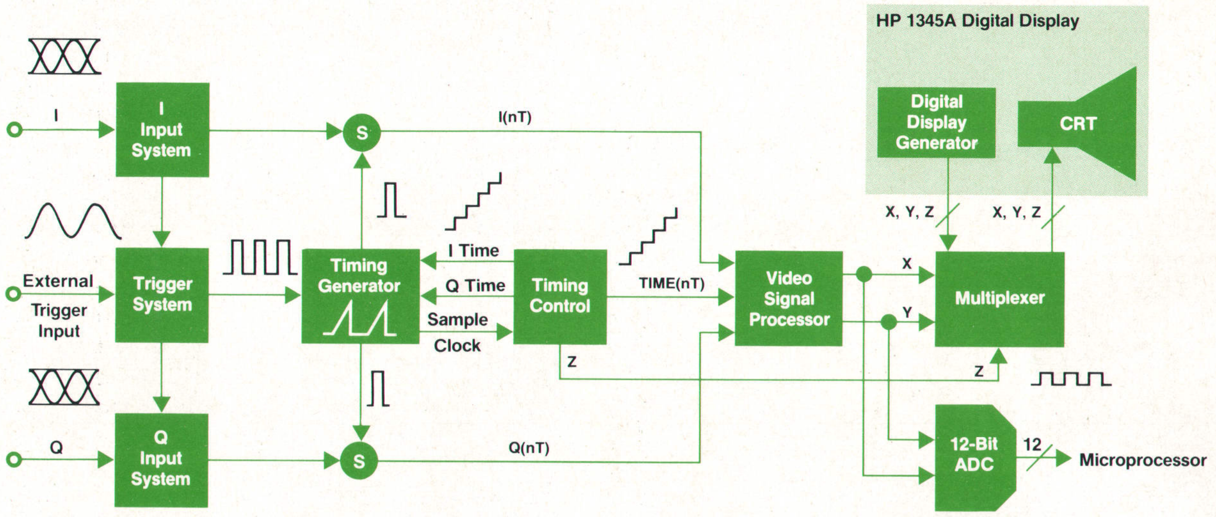


Fig. 4. *System block diagram.*

on the front panel for easy reconfiguration. The 75-ohm assembly has an attenuation factor of 1.5 because of the resistive divider, and this reduces the sensitivity of the instrument from 5 mV/div to 7.5 mV/div. This division factor can be accounted for in the gain and offset setup functions by selecting 75 ohms in an instrument state configuration menu.

The first input relay is used for switching in calibration signals and for opening the inputs when an overpower signal is detected by the detector. Ac coupling is selected by switching in a capacitor. The switched attenuator is used to scale the signal into the amplifier and sampler in coarse steps for optimum distortion and noise performance. The dc-to-500-MHz amplifier is built in surface mount technology for better RF performance than is possible with a standard printed circuit design. The amplifier has an offset control input for up to one full scale of offsets and a trigger output that is isolated from the signal output. The equalizer corrects the pulse distortion and poor frequency response of the coaxial delay line before the samplers. The delay line is required to view the signal at the trigger instant, since the trigger and timing systems take several nanoseconds to generate a sampling pulse.

Samplers

The samplers perform the function of compressing the 350-MHz input bandwidth into a video bandwidth of about 3 MHz. This allows the display unit with its limited bandwidth in the X and Y axis to handle the kind of signals that are likely to be encountered in modern communications and radar systems. To reproduce the input signals accurately in the video range (the postsampler signals are referred to as video signals), the samplers are driven by the timing generation block and the timing control block (Fig. 4). They use a variation of a timing technique called sequential sampling which takes advantage of the fact that the signals under observation are repetitive in nature (more on this in the timing section).

To have a total system bandwidth of 350 MHz, each band-limiting element in the chain must be able to accommodate bandwidths greater than this. In this case the samplers are required to provide a bandwidth greater than 500 MHz. In addition to bandwidth goals, the samplers are required to hold samples for very long periods of time.

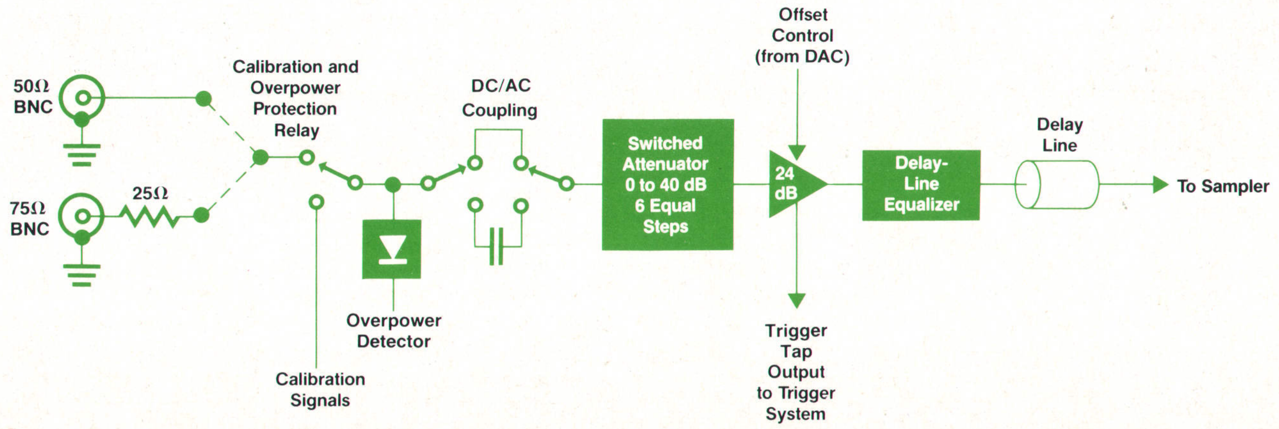

50Ω BNC
75Ω BNC
25Ω
Calibration and Overpower Protection Relay
DC/AC Coupling
Overpower Detector
Calibration Signals
Switched Attenuator 0 to 40 dB 6 Equal Steps
Offset Control (from DAC)
24 dB
Trigger Tap Output to Trigger System
Delay-Line Equalizer
Delay Line
To Sampler

Fig. 5. *Block diagram of I (or Q) input system.*

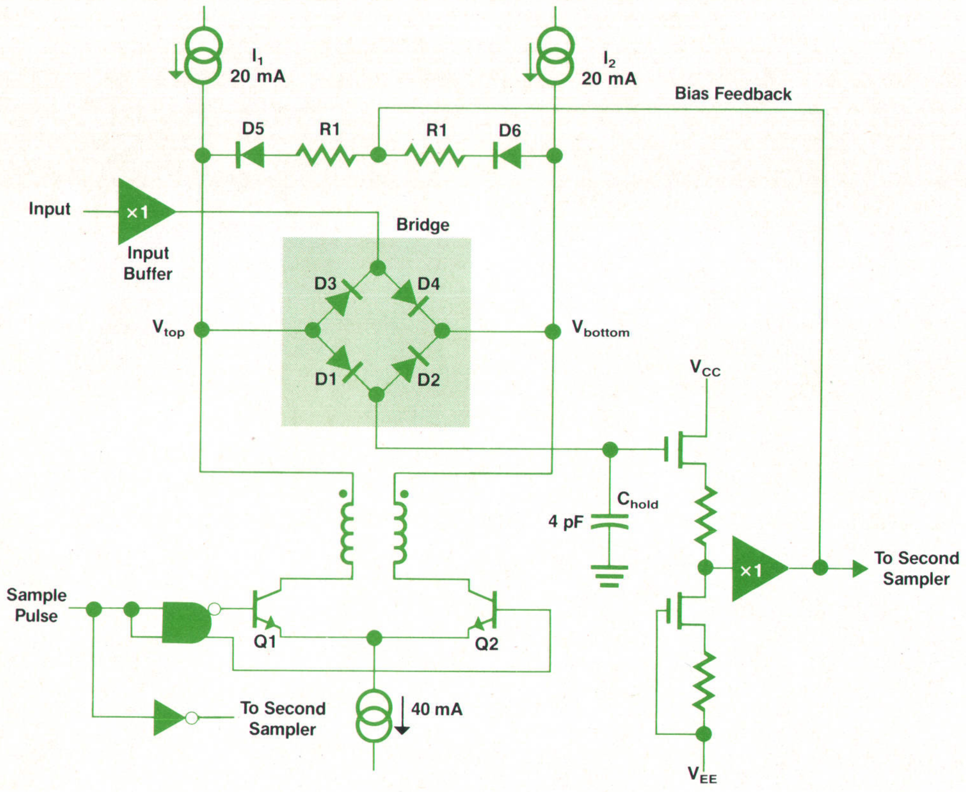


Fig. 6. *Sampler schematic.*

This allows the internal voltmeter enough time to complete the analog-to-digital conversion required in several of the internal measurement functions (calibration, print, HP-IB data query, constellation measurements, etc). It is very difficult to build a sampler that satisfies both of the above requirements in a single stage, that is, one with very wide input bandwidth and long hold time simultaneously. Therefore, we opted to split the task in two separate sampler circuits. The first sampler circuit provides the bandwidth while the second circuit handles the long hold times. The conversion time of the analog-to-digital converter (ADC) is approximately 25 μs per point, which for two channels plus overhead translates into hold times of over 100 μs.

The design of the second sampler is rather straightforward since it deals only with signals within the system's video bandwidth. In deciding the topology for the first sampler a few more issues had to be considered. Two sampling technologies are widely used at the present time: sample-and-hold or track-and-hold. We chose the track-and-hold technique because of better noise and distortion performance while providing sufficient bandwidth performance to satisfy our requirements. Using printed circuit board technology and state-of-the-art components, this circuit pushes to the limit the bandwidth performance of a track-and-hold design. Wide bandwidths are much more easily attained in sample-and-hold designs but noise, distortion, and sampling efficiency issues steered us away from using sample-and-hold techniques.

Fig. 6 shows a simplified schematic diagram of the first sampler. During track mode, transistor Q2 is on and Q1 is off. Since Q1 is off, the current from current source I1 only has one way to go—through the diode bridge, turning it on. While in this state the bridge behaves as a resistor equal to the dynamic resistance of one diode at the particular bias current. Since the load is capacitive, only at high frequencies is there any significant signal current flowing through the bridge. This means lower distortion at the lower frequencies. The voltage on C_{hold} ideally follows the input voltage within the bandwidth of the low-pass filter created by the bridge resistance (R_{bridge}) and C_{hold}. In reality parasitic and various FET-related capacitances in combination with trace inductances contribute as much to limiting the bandwidth as R_{bridge} and C_{hold} do. The input buffer serves two purposes: to provide reverse isolation to keep current spikes related to the sampling pulses from going out of the circuit board and to present a low-impedance drive for the bridge.

When the sampler goes into hold mode, transistor Q1 turns on and Q2 turns off. I1 flows directly into the collector of Q1 while I2 finds its way through D5 and D6. This current results in a voltage drop from the cathode of D2 to the anode of D1, reverse-biasing the diode bridge. During this transition the current in the diode bridge goes from 20 mA in its on state to zero current in the off state in a few hundred picoseconds, the time that the bridge takes to go from its low-impedance state to its high-impedance state. While the charge is being held in the first sampler, the second sampler acquires this voltage (see Fig. 7). When the first sampler returns to the tracking mode, the second sampler in turn goes into its hold mode, retaining the sampled voltage for further video processing.

The secret to achieving good distortion performance in this circuit is balance. Several things are done to ensure balance in the bridge's driving signals. To begin with, the

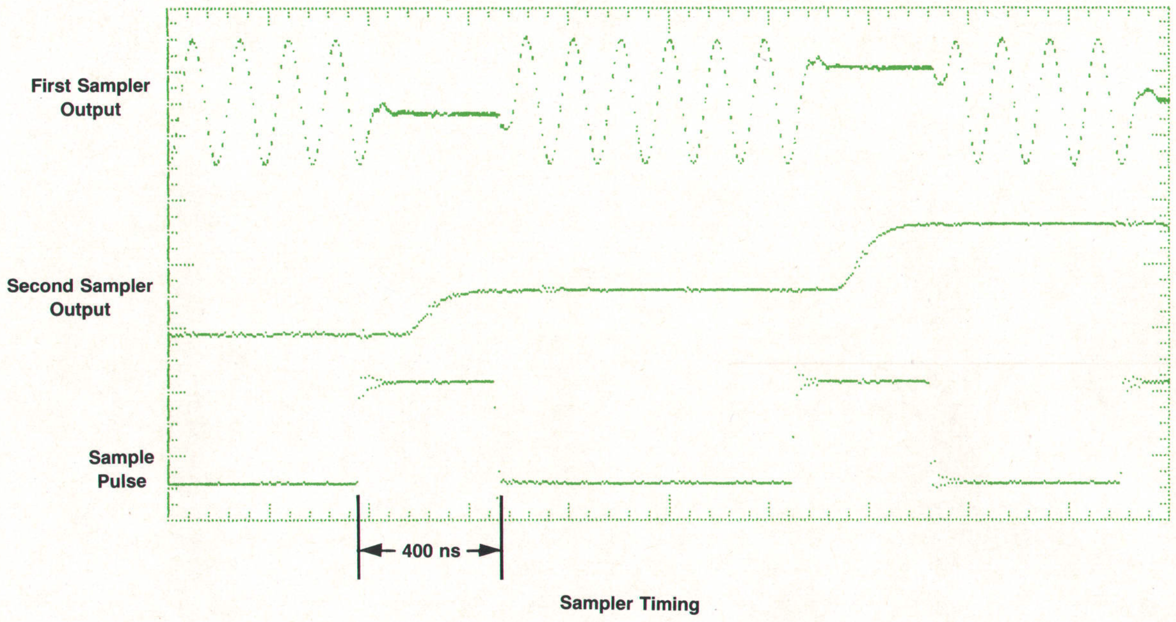


Fig. 7. *Sampler waveforms.*

diode bridge design is monolithic, which provides a good match between all four diode characteristics as well as their parasitic capacitances. The balun transformer at the collectors of the switching transistors helps ensure that the drive signals at the top and bottom of the bridge are complementary by presenting a high impedance to common mode signals from Q1 and Q2, but a low impedance to the differential mode signals. If symmetry in these waveforms were not maintained, the balance in the currents through the individual bridge diodes would be disrupted, resulting in distortion.

Another distortion-fighting mechanism in the circuit is the bias feedback loop. When the bridge makes the transition from on to off, a reverse voltage equal to two diode drops (D5 and D6) plus the IR drop in R1, appears across the bridge. The simple approach would be to ground the bias feedback node. This would be enough to ensure that the bridge is turned off over a range of C_{hold} voltages of about ±1.5V. In this case, the voltages across the bridge diodes D1 and D2 would be:

$$V_{D1} = V_{C_{hold}} - V_{top}$$

$$V_{D2} = V_{C_{hold}} + V_{bottom}$$

where V_{top} would be about 1.5V below ground and V_{bottom} the same amount above ground (R1 = 50Ω). Hence, the voltage across D1 and D2 would be a function of the sampled voltage $V_{C_{hold}}$. However, the diode bridge has parasitic capacitances that are significant compared to the hold capacitor and that are nonlinear with voltage. These capacitances suck up a proportion of the charge originally trapped in C_{hold} as reverse voltage develops across them, changing the voltage held by C_{hold}. Since the voltage across the bridge diodes is in this case a function of the sampled voltage itself, the nonlinearities in the parasitic capacitances would result in nonlinear redistributions of charge among the three capacitances in question. This would show up as distortion. To get around this problem, the feedback node is tied to the buffered version of the C_{hold} voltage. This ensures that the potential developed across the parasitic capacitances during the hold period is constant and not dependent on the sampled voltage. This greatly reduces distortion effects caused by the nonlinear capacitances of the sampling bridge.

The most significant distortion mechanism left in the circuit is that caused by slew rate limiting. The 20-mA maximum current that the bridge can supply acts as a limit on dV/dt in C_{hold}. With the bias currents chosen, this sampler is able to keep distortion terms less than −40 dB at

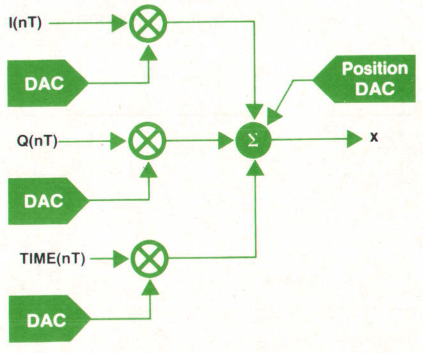


Fig. 8. *Video signal processing block.*

frequencies greater than 150 MHz. At low frequencies the distortion is less than −60 dB (for a 1V signal), essential in meeting the system's dc accuracy goals.

Video Signal Processing

After the samplers capture the data, some video processing is needed to condition the signal before it can be shown on the display unit. Some of the functions required are fine gain adjustment, position control, and signal routing to produce the several types of displays the HP 8980A is capable of displaying. For example, it may be necessary to route the Q-channel information to the Y axis of the CRT and the sweep information to the X axis to display Q versus time (this is the regular oscilloscope mode). On the other hand, for a vector display (the equivalent of X versus Y mode on scopes) the I-channel samples should be sent to the X axis instead.

In addition to the functions mentioned above, the video processing block performs certain kinds of linear operations needed to implement the quadrature and phase correction features of the instrument (see box on page 15). An extension of this capability is the 3-D display which will be discussed later.

The basic function of the video block can be described by the following matrix operation:*

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ x & x & x \end{bmatrix} \begin{bmatrix} I(nT) \\ Q(nT) \\ \mathrm{TIME}(nT) \end{bmatrix}$$

The last row of the matrix is filled with xs (don't cares) because only the X and Y information is ever needed for display (in this case, the Z axis of the CRT is driven by a separate circuit). Fig. 8 shows a simplified block diagram of this circuit. When a user chooses a setting, the microprocessor calculates the matrix coefficients corresponding to it. The video block then performs the remaining multiplications and additions needed to complete the matrix operation. The multiplications and additions are done in the analog domain to achieve the data processing speed required by the system, which can be as fast as one megasample per second in some instances. The fast data throughput is needed to display the large amounts of information contained in some signals used in the digital communication field (e.g., eye and vector diagrams of higher-order modulation formats). To try to achieve this data update rate digitally would have resulted in a much more expensive and complex circuit. This way the task is split between the microprocessor and the analog circuits.

*The timing signal TIME is the sweep control voltage generated in the timing control block. In addition to telling the timing generation circuits when to sample with respect to the trigger event, the TIME signal also tells the CRT display where along the X axis that sample should be positioned.

The processor calculates the matrix coefficients, which depend on the setup of the instrument. This computation can be time-consuming because it involves complex trigonometric functions. Fortunately, users do not require the ability to change setups (like sensitivity or display mode) very fast. Update rates of three or four times per second will satisfy most needs. This provides the software with enough time to do the necessary math. Once the coefficients are programmed, the analog circuits can do the remaining processing of the samples at blazing speeds (relatively speaking).

Consider a simple case—the vector display, where the Q-channel data is displayed on the Y axis and the I-channel data is displayed on the X axis of the CRT. The processor sets the matrix coefficients as follows:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} I(nT) \\ Q(nT) \\ \mathrm{TIME}(nT) \end{bmatrix}$$

If the fine gain needs to be adjusted, the firmware changes the ones to fractions of one, resulting in a sensitivity change.

The several kinds of displays that have been described so far are two-dimensional in nature. The Q and I channels are displayed versus each other or versus time. Fig. 9 shows a display of both channels versus time for a signal typically used in modern radar systems, the frequency chirp. A chirp

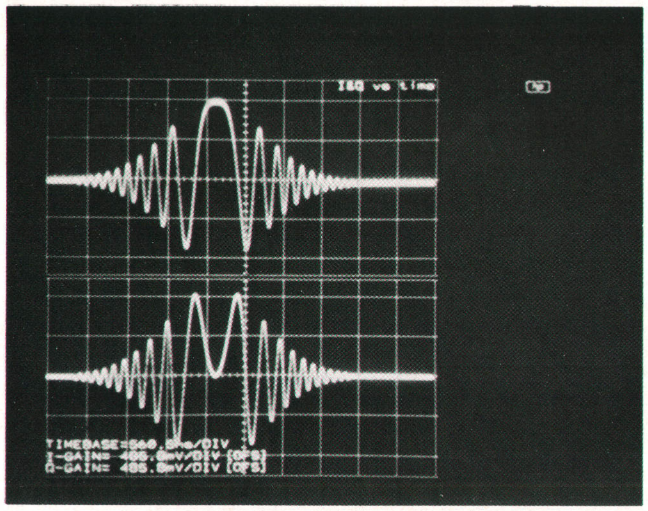


Fig. 9. *Chirp Q and I components versus* TIME.

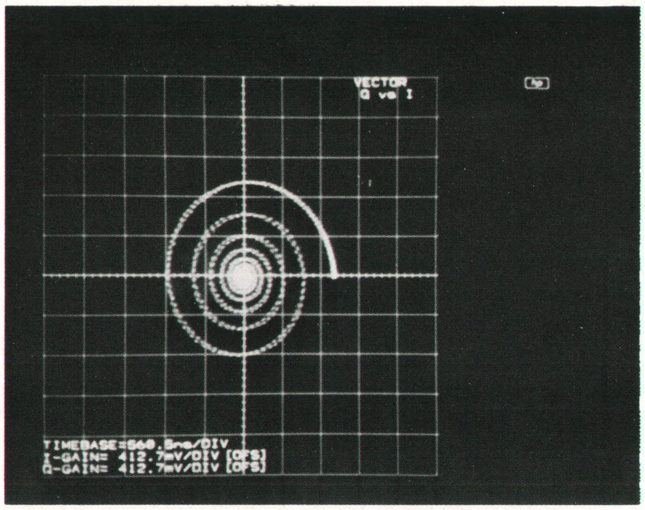


Fig. 10. *Chirp Q versus I display.*

is a signal whose frequency varies linearly with time. In most practical cases its amplitude is also modulated to provide timing references from which to measure radar returns. Fig. 10 shows the same chirp but this time one channel is plotted versus the other one. In this vector display phase and amplitude information can be extracted from the polar interpretation of the screen. The spiral trajectory tells us that as the phase changes with time so does the amplitude. The exact time dependency of this change is not as evident from this display as it is when plotted versus time.

By viewing the operation performed by the matrix as merely a mapping from one space (call it the signal space) to another (the display space), and recognizing that the video processing circuit is capable of setting the matrix coefficients arbitrarily (within the 12 bits of the digital-to-analog converters), one can consider a whole new set of interesting possibilities. Two HP 8980A functions that make use of this capability are the phase and quadrature correction algorithms (see box on page 15). The phase correction is a mapping that rotates the I-Q signal space about the origin. This operation is described by the matrix:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \end{bmatrix} \begin{bmatrix} I(nT) \\ Q(nT) \\ \mathrm{TIME}(nT) \end{bmatrix}$$

Quadrature errors are deviations from a perfect 90-degree relationship between the I and Q channels, causing portions of one channel to leak into the other one. The quadrature correction algorithm can undo this effect by choosing the matrix coefficients as a function of the amount of quadrature error q present:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos(q/2) & \sin(q/2) & 0 \\ \sin(q/2) & \cos(q/2) & 0 \end{bmatrix} \begin{bmatrix} I(nT) \\ Q(nT) \\ \mathrm{TIME}(nT) \end{bmatrix}$$

A signal showing both kinds of errors can be corrected by using a matrix that is the product of both these matrices (since matrix multiplication is not commutative in general, it is important to do it in the correct order). The resulting correction matrix becomes a modifier that can operate on any of the original matrices corresponding to the basic display types. This allows the correction to appear in the display versus time as well as the vector display, or any of the other display types available. An example of a QPSK signal before and after errors were removed by the HP 8980A Vector Analyzer is shown in Fig. 11. It is interesting to note that these errors are removed at baseband even though they can be introduced in the system's IF or carrier frequencies.

A further extension of the mapping concept results in the 3-D display mode. At any point in time, the I, Q and TIME signals are related to each other in a certain way. The display modes above have focused so far on two-dimensional cross sections of what is in fact a three-dimensional space with the I channel in one axis, the Q channel on another axis, and TIME on the third axis. There are some cases where being able to capture the three-dimensional essence of these signals adds insight to their true nature. Fig. 12 shows one such display where the same chirp of Figs. 9 and 10 is shown from two arbitrary viewing angles.

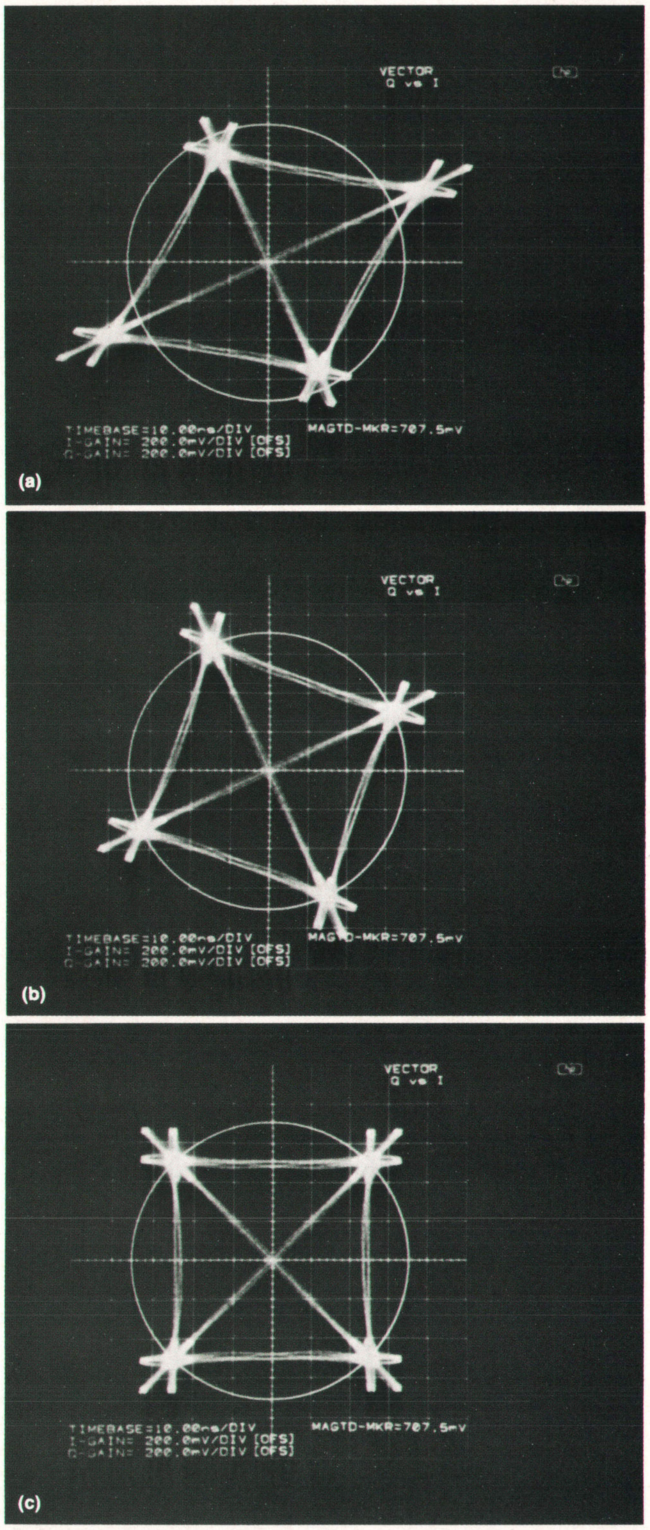
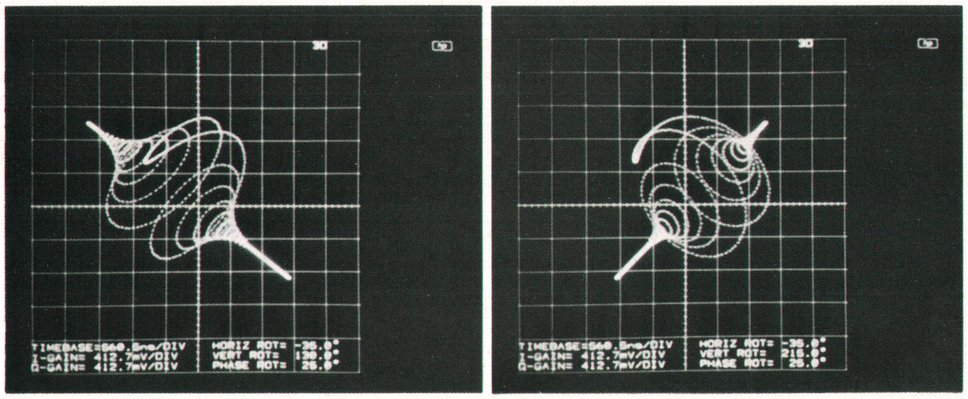


Fig. 11. *(a) QPSK signal corrupted by phase and quadrature errors. (b) After quadrature correction. (c) After quadrature and phase correction.*



```
TIMEBASE=560.8ns/DIV    HORIZ ROT= -35.0°
I-GAIN= 412.7mV/DIV     VERT ROT=  130.0°
Q-GAIN= 412.7mV/DIV     PHASE ROT= 25.0°
```

```
TIMEBASE=560.8ns/DIV    HORIZ ROT= -35.0°
I-GAIN= 412.7mV/DIV     VERT ROT=  215.0°
Q-GAIN= 412.7mV/DIV     PHASE ROT= 25.0°
```

Fig. 12. *Three-dimensional rotations of chirp displayed in Figs. 9 and 10.*

Features like the phase reversal near the center of the display can be appreciated much more readily now than on the previous pictures. Mathematically, this 3-D display is a simple extension of the 2-D version. In this case, however, all of the matrix coefficients will be nonzero and functions of the three rotation angles selected. As before, the instrument's firmware incorporates quadrature and phase correction effects into the 3-D display by simply multiplying corresponding matrices together.

Triggering

The HP 8980A is normally used with external triggering when measuring digital radios because the I and Q signals are filtered random data signals. The filtering of the data corrupts the timing for displaying eye patterns or constellations if the I or Q signal is used for triggering. The clock signal from the radio is used as the trigger input to avoid this problem. For other types of measurements, however, the I or Q signals can be used as the trigger source.

The function of the trigger system is to create an ECL-level digital signal with a precise time relationship to the selected trigger input. Basically, the signal is an ECL pulse train at the trigger frequency. This signal starts the timing generation circuit whenever the timing system is ready to take another sample. The trigger system selects the trigger source and sets the trigger level based on the requested value and the attenuation in front of the trigger output of the input amplifier. It uses fast voltage comparators and sets the trigger thresholds with digital-to-analog converters (DACs). Either dc or ac coupling is available in internal triggering, independent of what is selected for the display.

External triggering is always dc-coupled. Preset levels for TTL and ECL are available in external trigger, as well as variable levels with 40-mV resolution. The most-often-used trigger threshold setting is **AUTO**, which continuously adjusts the trigger to halfway between the positive and negative peaks of the external input signal. This usually works well for all but the slowest trigger inputs. Another function of the trigger system is to select either ground or ECL terminations (50Ω to −2V).

Timing

The HP 8980A uses a modification of the sequential repetitive sampling technique. In sequential sampling a single sample is taken as a result of a trigger event. By slightly incrementing the delay from trigger every time a new sample is taken, the original waveform can be reconstructed as long as it is repetitive (and stationary). The timing control block of the system's overall block diagram (Fig. 4) generates the sweep control voltage TIME. When the timing generation block receives a trigger, it starts a ramp which is then compared to the TIME voltage. When these two values become equal, a sample pulse is generated and sent to the samplers. The video signal processor uses the TIME signal to position the samples correctly in the display using the fact that this voltage represents time-axis informa-

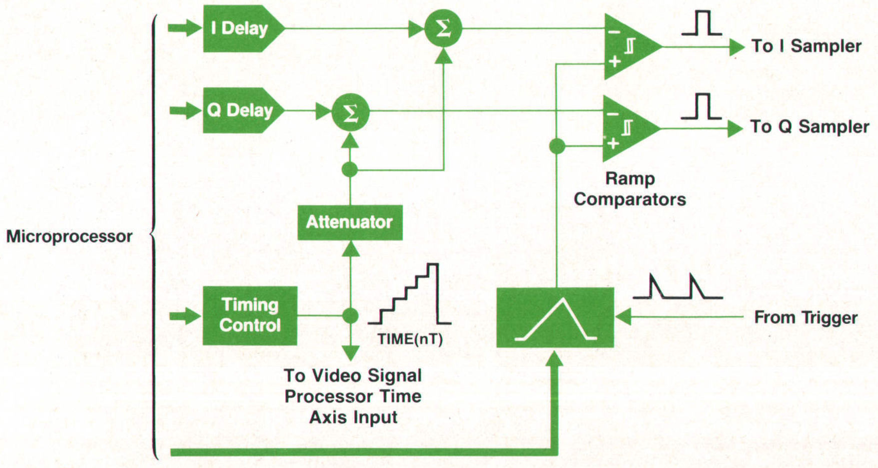


Fig. 13. *Timing system block diagram.*

tion of the sample in hand.

To accomplish some of the delayed sweep functions of the instrument, the TIME signal is first attenuated and then a dc offset is added to it before it is sent to the ramp comparators (see Fig. 13). The dc offset becomes a time delay after going through the ramp comparator. Since the ramp is designed to span a constant range of voltages, the control signal must be attenuated if a dc offset is added. The instrument's processor, armed with information on what sweep and how much delay is required, decides how much dc offset it needs to add as well as the attenuation factor and the ramp slope (the slope of the ramp can be continuously adjusted with 10 bits of resolution over six decades, starting at 10 ns/V). Fig. 14 shows a timing diagram of the system in which seven subsequent ramps have been superimposed to depict the reconstruction process.

Traditional sequential sampling has the drawback that since only one sample is taken per trigger event, as trigger rates get slower the number of samples taken per second (sample throughput) is reduced significantly. This effect is intolerable in the HP 8980A, which requires a high sample throughput to achieve the display quality needed. To get around this problem the user can enable the HP 8980A's multisample mode.

In the multisample mode the first point on the block is taken at the same time as it would have been during regular sequential sampling. In this case, however, the timing circuit reuses the same timing ramp to generate a sequence of sampling pulses spaced 1.5 μs apart until the display window is filled. If the instrument setting is such that the timing ramp is long compared to 1.5 μs, a lot of samples can be taken every time a ramp is launched by a trigger event. Contrasting this with the single-sample-per-trigger case found in normal sequential sampling, one can easily see the improvements in data throughput. Now instead of taking one sample per trigger, the timing system takes a block of samples each time a trigger event is received (the size of the block will depend on the instrument setting). Subsequent blocks are shifted in time by one resolution interval from the previous block until the waveform is completely reconstructed. Fig. 15 illustrates this process.

The multisample feature allows the HP 8980A to maintain over a wide range of conditions the fastest sample throughput allowed by the bandwidth of the display unit. Throughput limitations still occur for certain combinations of slow trigger rates and small time windows, such as would be needed to look at low-duty-cycle pulse trains.

To meet the instrument's timing specifications, all of the control voltages used in the timing circuits need to be generated very accurately and the ramp slope precisely known. Calibration plays a major role to ensure that this happens over the six-decade range of specified performance as well as over environmental conditions. The calibration section in the firmware article on page 17 describes in more detail how this is accomplished.

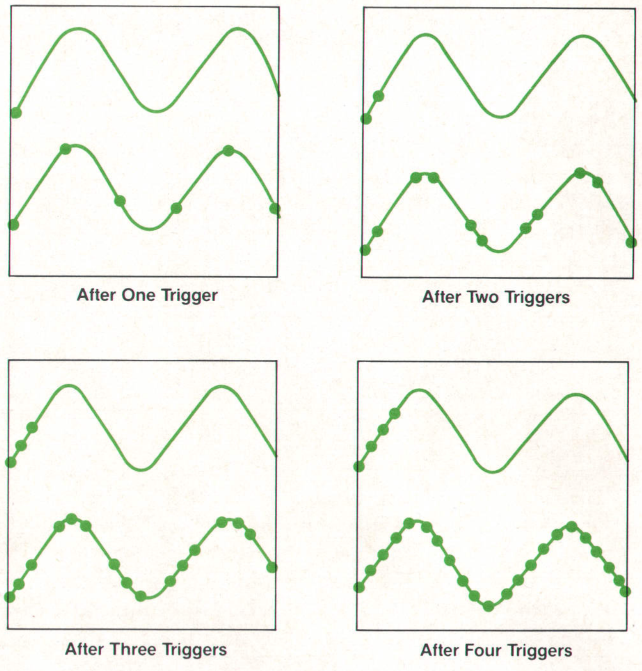


Fig. 15. *Signal reconstruction using multisampling. The increased data throughput of multisample mode (bottom trace in each block) is contrasted with normal sequential sampling (upper traces).*

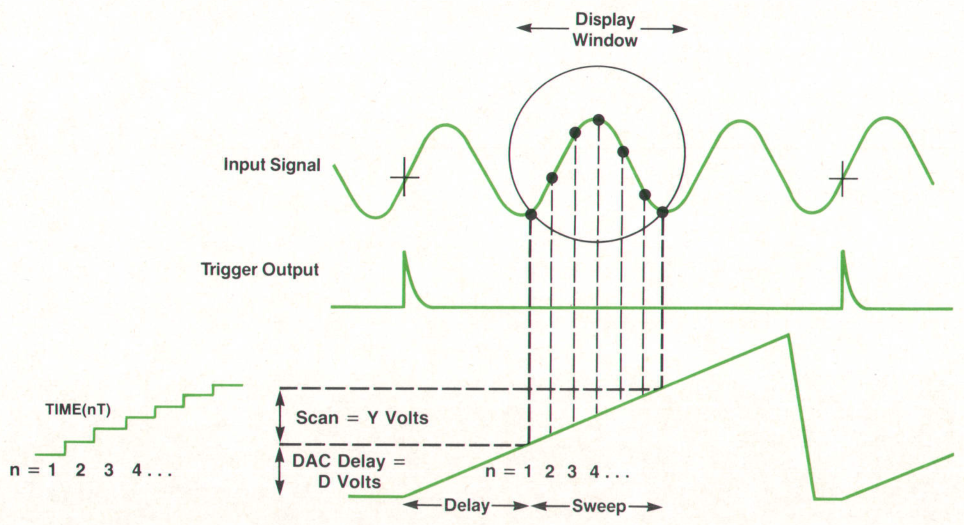


Fig. 14. *Signal reconstruction using sequential sampling. The events following seven consecutive trigger pulses (n=1 to 7) are superimposed in a single picture. This highlights the relationships between the voltage scanned by the* TIME *signal and the sweep as well as the relationship between the I (or Q) DAC delay voltage and the I (or Q) delay.*

Quadrature and Phase Errors in Vector Demodulation

Any practical implementation of vector modulation techniques invariably results in the appearance of phase and quadrature errors. To understand their causes, let's examine the block diagram of a nonideal vector demodulator such as the one shown in Fig. 1.

The RF signal splits into the in-phase (I) and the quadrature phase (Q) paths. The in-phase path is demodulated with a signal that ideally has the same frequency and phase as the carrier. In digital communications systems this unmodulated carrier is in most cases not available to the receiver because the transmitter and receiver are in different physical locations. Therefore, the demodulator must be able to reconstruct the unmodulated carrier from the modulated carrier. This is accomplished in carrier recovery and regeneration circuits, which lock onto the frequency of the modulated carrier but have a difficult time extracting the phase information accurately.

Another major use of vector modulation techniques is found in radar systems. Here the transmitter and receiver share the same physical location (e.g., the nose of an airplane or a mast of a ship). Therefore, the unmodulated carrier reference is available to the receiver for use in the demodulation process. In this case, differences in path lengths to the mixers between the input signal and the oscillator contribute to the phase error term.

The quadrature phase path is demodulated with the same regenerated carrier as the in-phase path, but it is shifted by 90 degrees, resulting in a cosine term. The argument has the same phase error value as the in-phase channel, but the quadrature term has the opposite sign. This quadrature figure describes inaccuracies in the 90-degree phase shifter. The quadrature error is split equally between the I and Q channels to simplify calculations. What is important to remember is that the in-phase error represents the difference between the modulated carrier and the recovered one while the quadrature error represents deviations from the 90-degree relationship between the drive to the I-channel mixer and the drive to the Q-channel mixer.

The results of these errors are shown mathematically in the following equations describing the demodulated I and Q signals:

$$I'(t) = I(t)\cos(\phi_\epsilon - q_\epsilon/2) + Q(t)\sin(\phi_\epsilon - q_\epsilon/2)$$

$$Q'(t) = Q(t)\cos(\phi_\epsilon + q_\epsilon/2) - I(t)\sin(\phi_\epsilon + q_\epsilon/2)$$

If no errors are present, I'(t) and Q'(t) are equal to the original I and Q modulations. The presence of phase and quadrature errors introduces leakage between the two channels which in turn deteriorates the performance of the system.

A significant advantage of vector modulation is that it provides two independent channels of information in a bandwidth that a single channel would normally occupy. Sent in quadrature, they can be uniquely recovered at the receiver. To take full advantage of this fact, receivers of this kind strive to minimize the effects of quadrature and phase errors.

Juan Grau
Development Engineer
Stanford Park Division

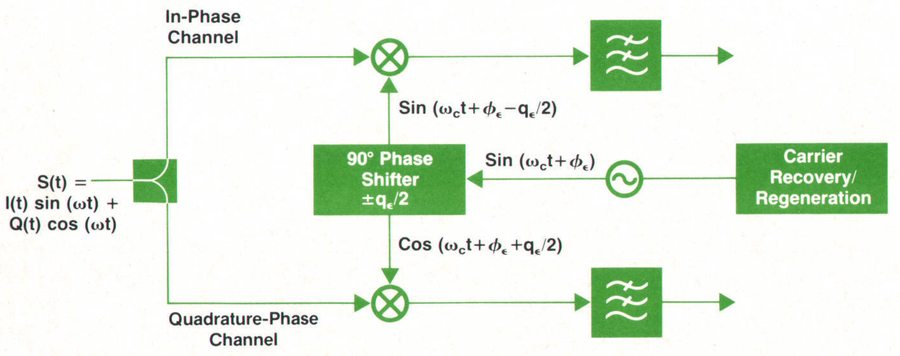

In-Phase Channel
Sin ($\omega_c t + \phi_\epsilon - q_\epsilon/2$)
S(t) = I(t) sin (ωt) + Q(t) cos (ωt)
90° Phase Shifter ±$q_\epsilon/2$
Sin ($\omega_c t + \phi_\epsilon$)
Carrier Recovery/ Regeneration
Cos ($\omega_c t + \phi_\epsilon + q_\epsilon/2$)
Quadrature-Phase Channel

Fig. 1. *Vector demodulator block diagram.*

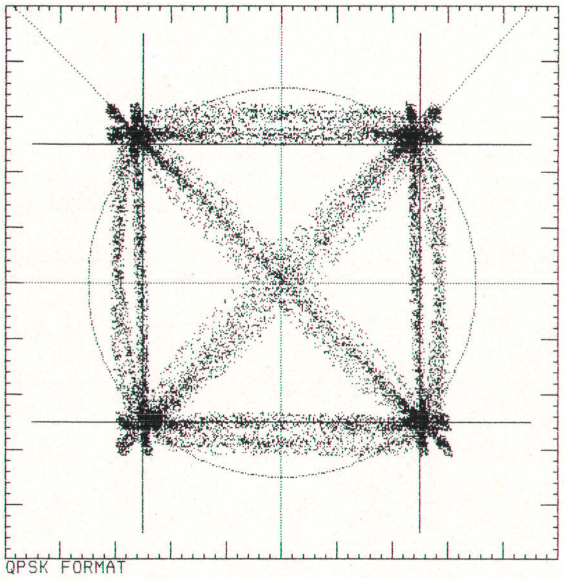
ADC and Measurements

The ADC measures the X and Y signals from the video signal processor to give quantitative voltage measurements. Voltages of I and Q can be displayed at a given time instant by setting the time marker to the desired instant and selecting the **CONTINUOUS I,Q** measurement function. The microprocessor controller sets the timing system for fixed time while it digitizes the X and Y signals. In a vector display, the I and Q voltages are then just the X and Y values, scaled and shifted by the gain and offset settings.

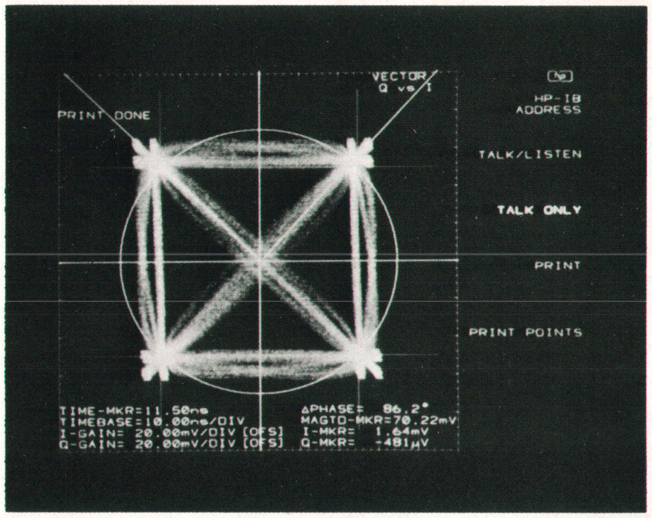
In a similar way, data can be acquired for HP-IB measurements in blocks of up to 1024 pairs. Measurements can be made at a fixed time or across an entire time sweep. I or Q can be measured separately, or together in pairs. Since the measurements are made at the X and Y outputs of the video signal processor, any phase or quadrature corrections are included before digitizing. These corrections could be done in software in the HP-IB controller that receives the data, but this would take valuable processing time. Making the corrections in hardware is much faster.

Although not designed to be a high-performance digitizing instrument, the HP 8980A's analog-to-digital resolution is higher than many instruments in its bandwidth range. The ADC has 12-bit resolution, but since amplifier noise is about four counts rms, the effective resolution for a single measurement is about nine bits. Most instruments in the 300-MHz-to-1-GHz range have seven or less effective bits because of noise. Finer resolution is possible for repetitive waveforms (i.e., not random data) by calculating the average of a number of measurements in the HP-IB controller.

```
HP 8980A              VECTOR
```



```
QPSK FORMAT

----- I Channel ---------------------------------------------------
I-GAIN= 20.00mV/DIV     DC-COUPLED              I OFFSET=  1.64mV
I-MKR=  1.64mV

----- Q Channel ---------------------------------------------------
Q-GAIN= 20.00mV/DIV     DC-COUPLED              Q OFFSET=  -481uV
Q-MKR=  -481uV

----- Phase-Magnitude ---------------------------------------------
PHASE OFS=   0.0DEG    QUAD OFS=   0.0DEG
PHASE-MKR= 134.2DEG    DELTA PHASE=    86.2DEG
MAGTD-MKR=70.22mV

----- Timebase ----------------------------------------------------
TIMEBASE=10.00ns/DIV    TIME-MKR=11.50ns
DELAY I&Q=    0ps       DELAY I=    0ps          DELAY Q=    0ps
AUTO SWEEP

----- Trigger -----------------------------------------------------
SOURCE= EXTERNAL   TERMINATION= GND   + SLOPE        LEVEL= AUTO
```



```
TIME-MKR=11.50ns                ΔPHASE=  86.2°
TIMEBASE=10.00ns/DIV            MAGTD-MKR=70.22mV
I-GAIN= 20.00mV/DIV [OFS]       I-MKR=  1.64mV
Q-GAIN= 20.00mV/DIV [OFS]       Q-MKR=  -481μV
```

Fig. 16. *Printed copy and photograph of corresponding display for QPSK vector analysis.*

The digitizing rate is dependent on the timing setup and other factors, but can be as high as 3000 sample pairs per second for fixed time measurements. This is quite slow compared to instruments like waveform recorders and digitizing oscilloscopes, but the time resolution is excellent, down to 50 ps or less. In addition, the HP 8980A digitizes at a fixed delay time relative to the trigger event, which can be an advantage in many situations over the random repetitive digitizing technique used in digitizing oscilloscopes.

The ADC also collects data for constellation analysis measurements, such as those made by the HP 3709A Constellation Display.[2,3] This function computes the quadrature, lock angle, and closure in I and Q for digital modulation signals. It is a quantitative measure of the radio's performance that can be used to measure a radio carrying live traffic without taking it off-line.

Many users asked for printing or plotting of the screen during our market research. Plotting the screen is impractical because of the unconnected nature of a sampled display of random data. It would be very difficult for the instrument to figure out how to connect the dots. Printing is far easier, however, since a raster pattern is an ordered set of dots already. Since the display is analog and the printed copy is digital, there is a difference between the appearance of the two. Fig. 16 compares a printed copy with a photo of the display for a complex vector diagram of a QPSK signal.

Finally, the analog-to-digital system is used for servicing the instrument by measuring voltages at various points throughout the instrument. Power supplies and crucial signal lines are measured as part of the self-test and diagnostic procedures. Some of these signals are also needed for performing the extensive self-calibration function, which assures specified accuracy in displayed voltages, timing, and triggering. This calibration is discussed in the following article.

Acknowledgments

Many others contributed to the hardware design of the HP 8980A Vector Analyzer. Kyle Karnosh designed the CRT display interface and the power supply circuits. Brian Messenger and Paul Lingane designed the trigger and input systems. Stan Little worked on the timing generation circuits, and John Hasen helped by getting our test systems in shape. Product design was by Charles Cook, Barry Rose, and Trevor Burward-Hoy. Ramani Pichumani designed the digital control system. Cindy Jensen worked on various

circuits and helped manage the introduction, along with production engineer Hilda Fong. Our circuit design gurus, Russ Riley and Tim Carey, contributed in many areas to make everything work right together.

References

1. K. Hasebe, W.R. Mason, and T.J. Zamborelli, "A Fast, Compact High-Quality Digital Display for Instrumentation Applications," *Hewlett-Packard Journal*, Vol. 33, no. 1, January 1982.
2. D.J. Haworth, J.R. Pottinger, and M.J. McKissock, "Dedicated Display Monitors Digital Radio Patterns," *Hewlett-Packard Journal*, Vol. 38, no. 7, July 1987.
3. M.J. McKissock, "Constellation Measurement: A Tool for Evaluating Digital Radio," *ibid*.

Firmware System Design for a Vector Analyzer

by Brian S. Messenger, Peter H. Fisher, and Stanley P. Woods

THE TREND IN THE DESIGN of electronic test instrumentation has been to provide the user with greater feature sets and performance while maintaining a consistent and easy-to-use human interface. This has resulted in the need for new methods to allow the user to exercise capabilities both on a lab bench and in an automatic test system environment. The advent of smaller and faster microprocessor systems that can be used in instrument design has opened a new realm of instrument control and calibration possibilities. It has also become possible to add calculation-intensive statistical capability to stand-alone instruments to allow them to perform functions that would previously have required an additional computer system. The common linkage between all of these advances is the need for writing and integrating large firmware modules for electronic instrumentation.

The HP 8980A Vector Analyzer is an example of this substantial change in the importance of software in electronic instrumentation. In the late 1970s, typical instruments used finite state machines or 8-bit microprocessors and perhaps 1K to 2K bytes of internal firmware. The HP 8980A uses a 32-bit 68000 microprocessor for overall control, in addition to a keyboard processor, an HP-IB (IEEE 488/IEC 625) interface processor, and a CRT display processor. A half megabyte of software is used by the HP 8980A. This software is written in C and, where optimization for speed was critical, in assembly language. In all, the software development task for the HP 8980A required approximately 90 engineer months of development.

The software task for the HP 8980A can be broken into five major blocks. The display software involves controlling grids, softkey menus, and measurement markers on the HP 1345A Display. The interactive help function gives the user information about all instrument functions with fingertip control. The hardware control software is responsible for setting up the state of approximately 250 bits that control everything from switches and digital-to-analog converters (DACs) to measurements and display modes. The internal calibration software for the HP 8980A allows the instrument's dc accuracy, triggering, and timing circuits to be calibrated with a single keystroke at any time. Finally, for the benefit of automatic test system users, the latest IEEE 488.2 standard is provided for HP-IB measurements and control.

Visual Markers

The HP 8980A provides the user with a variety of visual markers for quantitative visual measurements on the CRT. All markers have four core functions which perform consistently across all marker menus:
1. MKR VALUE, which is used to change marker value and invokes the marker on function below.
2. MARKER ON/OFF, which activates and deactivates a marker. This toggle function also invokes the marker value entry function above when the marker on function is

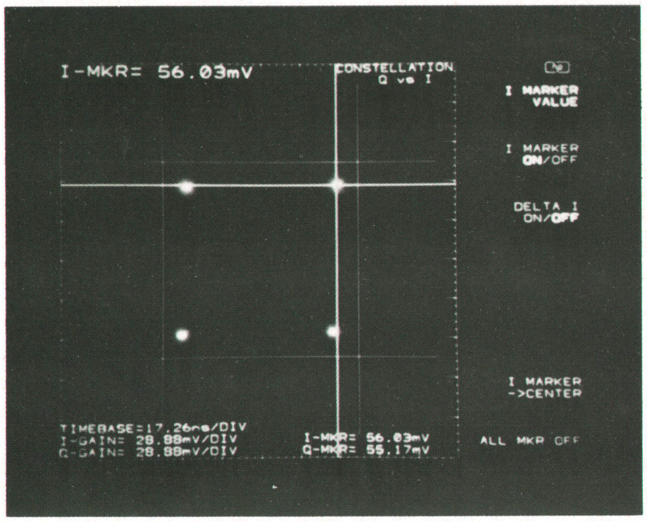


Fig. 1. *Using the HP 8980A's I and Q markers to determine the I and Q voltage components of one state of a QPSK signal.*

selected.

3. DELTA MARKER ON/OFF toggle function.
4. ALL MKR OFF, which turns all active markers off.

An additional function, MKR CENTER, appears on the menu for three types of markers (the I, Q, and TIME markers).

To measure either the I or Q voltage components of a signal visually, the instrument features a pair of I and Q markers. These markers are implemented as straight lines across the grid indicating the voltage at their position relative to the voltage represented at the center of the screen. Fig. 1 shows an example of how the I and Q markers are used to determine the I and Q voltage components of one state of a QPSK signal.

Because the center position of the CRT represents the user offset voltages (offsets dialed in from the front panel), the I or Q marker center function can be used to indicate the offset voltages. This function moves the markers to the CRT center position and displays the marker voltage at that position. This is particularly useful because only an offset annunciator is shown if the offsets are not zero. The actual offset value is not displayed unless the user selects the offset function in the gain and offset menu.

To invoke the I or Q delta marker function, the user first must position the I (Q) marker and then press the DELTA MARKER ON softkey to anchor a marker line at the current I (Q) marker position. The marker readout changes to delta marker values, which represent the voltage difference between the anchored marker line and the current I (Q) marker. Fig. 2 shows an example of how both I and Q delta marker functions are used in an amplitude measurement of the I and Q components of a QPSK modulated signal displayed in the time domain in split-screen display mode.

Because of the instrument's special ability to display the phase and amplitude of a signal at the same time, we developed two new marker types to simplify visual measurements of phase and amplitude in the I,Q plane: magnitude markers and phase markers.

The first new marker type, a pair of circular magnitude markers, indicates the magnitude of the vector sum of the I and Q signal components in volts. The magnitude markers are visible only in vector display modes (not time-domain display modes). Fig. 3 shows an example of a magnitude and phase measurement of one state of a QPSK modulated signal. The marker circle maximum extends to the four corners of the grid. The maximum display range for a given gain setting is equivalent to $5 \times$ volts/div $\times \sqrt{2}$ (the division units in volts/div refer to the general-purpose 10×10-division grid). Fig. 3 shows the special QPSK grid; only the tick marks of the grid frame indicate the 10×10-division raster.

The delta marker function associated with the magnitude marker function anchors a circular marker at the current magnitude marker position and the marker readout changes to delta magnitude marker in dB. This dB value represents the voltage difference between the magnitude marker anchor and the current magnitude marker position. Upon invoking the delta magnitude function, the readout starts with a dB value of zero since the anchor marker and magnitude marker are superimposed.

The second new marker type, phase markers, was developed to make visual phase measurements. The phase marker is implemented as a straight line originating from the center of the CRT and extending to the extremities of the grid. The marker moves clockwise or counterclockwise around the display like clock hands. The phase is displayed in degrees, starting with zero degrees at the right-hand part of the horizontal scope grid axis. Fig. 3 shows an example of measuring the phase and magnitude of one of the QPSK states.

Like the other markers, the phase marker has a delta marker associated with it. The user invokes the delta phase marker by first positioning the phase marker and then turning the delta marker function on. This anchors a phase marker line at the current phase marker position and changes the marker value readouts to the degree value measured between the anchored marker and the current phase marker position.

Fig. 4 shows an example of how magnitude and phase

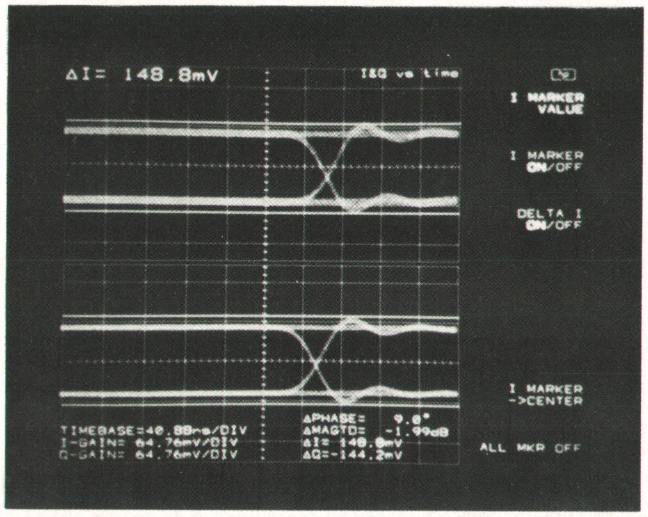


Fig. 2. *Using the I and Q delta marker functions to measure the amplitude of a QPSK signal displayed in the time domain in split-screen display mode.*

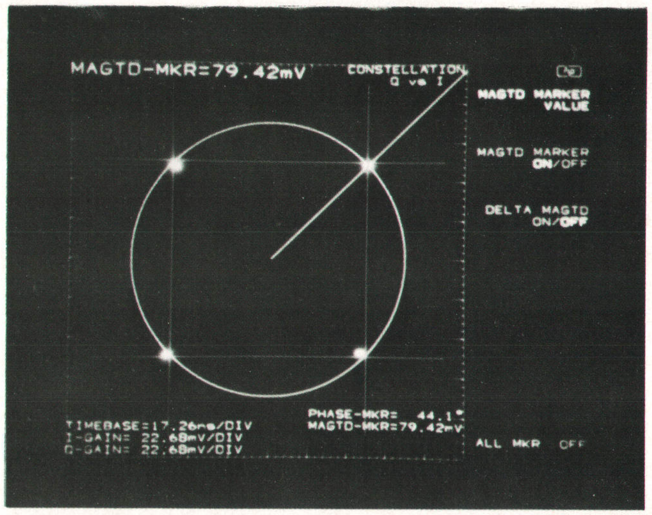


Fig. 3. *Measuring the amplitude and phase of a QPSK signal using the HP 8980A's amplitude and phase markers.*

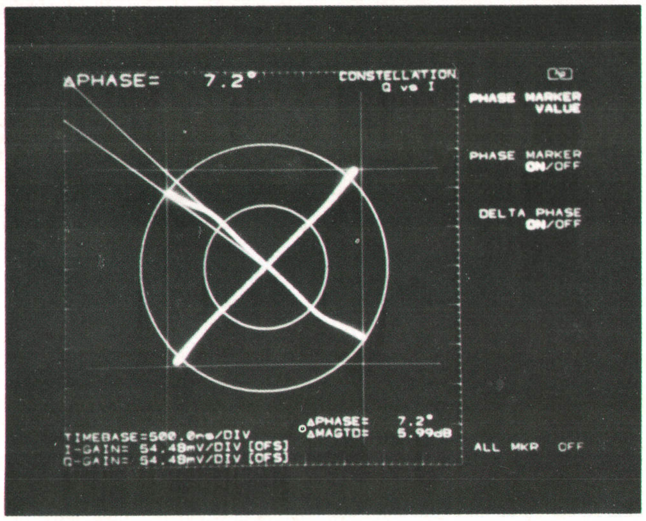


Fig. 4. *Use of amplitude and phase marker functions to determine AM-to-PM conversion and AM-to-AM compression (see text).*

markers help in determining amplitude modulation to phase modulation (AM-PM) conversion and amplitude modulation to amplitude modulation (AM-AM) compression. In this example, the HP 8780A Vector Signal Generator is used to generate the QPSK modulated signal. The signal is simultaneously amplitude modulated with a triangular signal using the SCALAR function of the HP 8780A. The output of the HP 8780A is fed into an amplifier/demodulator setup. The demodulated I and Q output signals of this setup are then input into the HP 8980A and displayed in constellation display mode.

The continuous amplitude modulation of the QPSK signal transforms the usual four dots of the QPSK state diagram into four lines forming a cross-like display as shown in Fig. 4.

AM-PM conversion and AM-AM compression occurs because of nonlinearities in the system. In the example illustrated by Fig. 4, these nonlinearities are produced in the amplifier/demodulator setup which introduces PM and amplitude distortion on the measured signal. The phase modulation introduced can be directly seen on the display which shows the signal slightly phase-shifted at the end (ideally there would be a perfectly straight line).

To measure the AM-AM compression, the user sets the HP 8780A output to the level at which the amplifier/demodulator setup shows linear behavior. The user moves the magnitude marker to that position and turns the delta magnitude marker function on. Next, the user increases the output level of the HP 8780A signal appropriately, which in the example shown in Fig. 4 causes AM-PM conversion and AM-AM compression. The user then sets the magnitude marker to the end point of the signal on the CRT. By comparing the output level increase with the delta magnitude reading on the display, the user can easily determine the AM-AM compression.

To measure the AM-PM conversion, the user first moves the phase marker to the linear part of the signal and then turns the delta phase marker function on. In the next step the user moves the phase marker to the end of the signal. In Fig. 4, the delta phase marker reading shows a deviation of 7.2 degrees between the linear portion of the signal and the distorted part, which indicates the AM-PM conversion for the particular level setting.

For time-related visual measurements, the HP 8980A is equipped with a pair of time markers. These markers are implemented as straight vertical lines across the grid on the CRT in all time-domain display modes. In addition, the analog samples on the screen at the time marker instant (value) are intensified. This feature serves two purposes. One is to allow the user to see the time marker position in vector display modes. The other is to indicate the sampling instant while measuring I and Q samples. The instrument measurement functions always measure the signal at the time marker instant. Because the time marker lines show up only in time-domain displays, the intensified samples are the only visual indicator in the vector display modes.

The time marker value is also an important parameter in constellation display mode. Unlike vector display mode, in which samples are taken across the entire time axis, the constellation display mode displays the signal only at the time marker instant.

If I and Q delay is added from the front panel, the time marker value at the left side of the grid is equal to the user I and Q delay. The time marker, like the other markers, has a delta marker function, which is visible only in time-domain-related display modes. The delta time marker function is invoked the same way as the delta I or Q markers. Fig. 5 shows a rise time measurement using the delta time marker function.

On-Line Help Feature

The HP 8980A on-line help feature describes the instrument's front-panel functions. The user invokes the function by pressing the **HELP** key, followed by the front-panel key about which the user wants advice. The instrument responds by writing the help text on the CRT. Along with a description of the key's function, the help feature gives

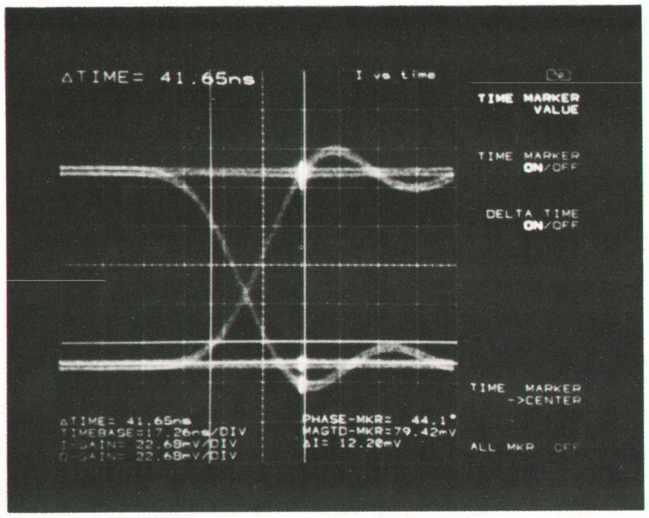
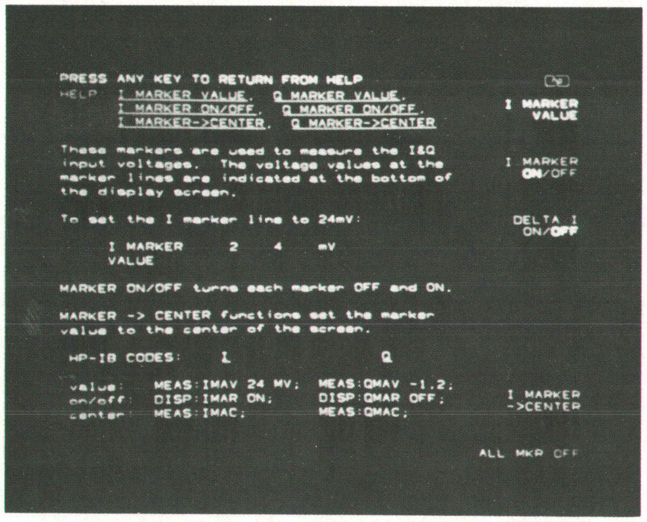


Fig. 5. *A rise time measurement using the delta tiıne marker function.*



```
PRESS ANY KEY TO RETURN FROM HELP
HELP  I MARKER VALUE, Q MARKER VALUE,                 I MARKER
      I MARKER ON/OFF, Q MARKER ON/OFF,                  VALUE
      I MARKER->CENTER, Q MARKER->CENTER

These markers are used to measure the I&Q
input voltages.  The voltage values at the          I MARKER
marker lines are indicated at the bottom of           ON/OFF
the display screen.

To set the I marker line to 24mV:                    DELTA I
                                                      ON/OFF
     I MARKER      2    4     mV
     VALUE

MARKER ON/OFF turns each marker OFF and ON.

MARKER -> CENTER functions set the marker
value to the center of the screen.

 HP-IB CODES:     I               Q

 value:   MEAS:IMAV 24 MV;  MEAS:QMAV -1.2;
 on/off:  DISP:IMAR ON;     DISP:QMAR OFF;          I MARKER
 center:  MEAS:IMAC;        MEAS:QMAC;              ->CENTER


                                                  ALL MKR OFF
```

Fig. 6. *Help screen for* I MARKER VALUE *softkey.*

additional information about any HP-IB mnemonics associated with that key.

To help users avoid feeling lost in the menus when they select a menu softkey, the menu text associated with the softkey is displayed simultaneously with the help text. Fig. 6 shows an example help screen for the I MARKER VALUE softkey.

The help text for keys with associated softkey menus is structured in the form of menu trees. These trees visually represent the menu displayed by the softkey plus all lower-level menus under it. This allows the user to scan all softkeys in the instrument without paging through all menus to find a particular function. We found this menu "road map" especially helpful to users searching for less frequently used functions located multiple levels deep in the menu structure. Fig. 7 shows the menu tree invoked by the **INST**rument **STA**te key, the most complex tree in the instrument.

Originally the help text occupied approximately 40K bytes of storage. Because of its size, we decided to reuse some of the text from other help screens and compress the text using the Huffman compression algorithm, resulting in about a 37% space reduction. We decided to use this algorithm because it is easy to implement and because the programs for compression and decompression are so general that they can easily be used in other instruments for similar applications.

In the compression process, the text compression routine generates one dictionary for the entire help text. In the next step, Huffman codes are generated for each paragraph and stored in individual blocks of memory (a paragraph corresponds to one help text screen for a particular key.)

Calibration

The HP 8980A calibration involves measurement and modeling of all major aspects of the instrument operation. For simplicity, it can be broken down into three sections: dc voltage calibration, triggering calibration, and timing calibration. Over 350 calibration factors are used in setting up the instrument for the entire spectrum of user-selected possibilities. These calibration factors may be results of direct measurements such as digital-to-analog converter (DAC) settings that produced a desired result, or they may be calculated using a measurement value and previous calibration factors.

During the initial development of the HP 8980A, two fundamental goals shaped the calibration development. The first goal was to eliminate potentiometers wherever possible. This reduces costly adjustments and allows tighter specifications since temperature drifts can be accounted for by calibrating at the operating temperature. The second goal was to take full advantage of the high-resolution DACs available to provide better accuracy and resolution for the user. This requires more elaborate calibration algorithms, averaging to reduce noise where needed, and verification to guarantee that the calibration results are accurate.

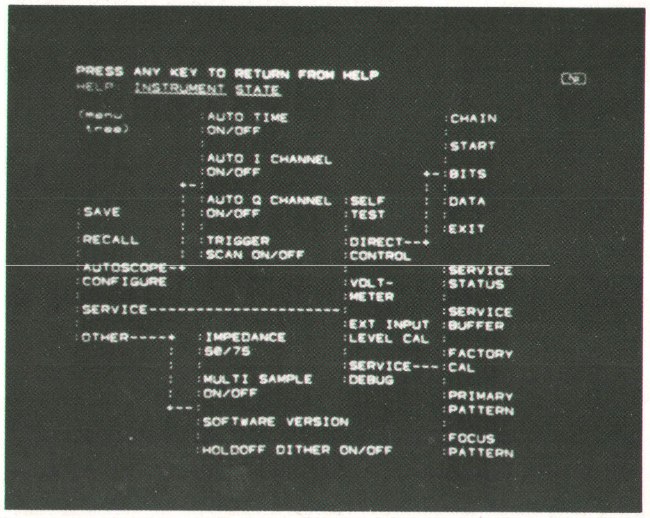
Display Setup. A wide variety of display modes are allowed by the HP 8980A, and the internal dc voltage calibration is intricately tied to their functionality. Considering the three variables I, Q, and TIME, the basic display modes of I versus TIME, Q versus TIME, vector, and constellation all involve displaying one variable along the X axis and one variable along the Y axis. The I and Q versus TIME, vector align, and constellation align display modes all involve swapping back and forth between two basic displays on a time division basis. Finally, the three-dimensional display mode allows the user total flexibility for viewing standard diagrams rotated in any direction. This display mode requires that any combination of I, Q, or TIME can be displayed along the X and Y axes.

Setting up any of the basic display modes is a simplified case of the three-dimensional case. In terms of instrument setup, the X direction and Y direction on the display are described as sums of weighted I, Q, and TIME values:

$$\text{X Rotated} = W_{IX}I + W_{QX}Q + W_{TIME\text{-}X}\text{TIME} \qquad (1a)$$

$$\text{Y Rotated} = W_{IY}I + W_{QY}Q + W_{TIME\text{-}Y}\text{TIME} \qquad (1b)$$

X Rotated and Y Rotated are the names of the analog signals



```
PRESS ANY KEY TO RETURN FROM HELP
HELP: INSTRUMENT STATE

(menu          :AUTO TIME                               :CHAIN
 tree)         :ON/OFF
               :                                        :START
               :AUTO I CHANNEL                          :
               :ON/OFF                              +--:BITS
             +-:                                    :  :
             : :AUTO Q CHANNEL  :SELF               :  :DATA
:SAVE        : :ON/OFF          :TEST               :  :
:            : :                :                   :  :EXIT
:RECALL      : :TRIGGER         :DIRECT--+
:            : :SCAN ON/OFF     :CONTROL
:AUTOSCOPE-+                    :                      :SERVICE
:CONFIGURE                      :VOLT-                 :STATUS
:                               :METER                 :
:SERVICE-------------------------:                      :SERVICE
:                               :EXT INPUT             :BUFFER
:OTHER----+   :IMPEDANCE        :LEVEL CAL             :
          :   :50/75            :                      :FACTORY
          :   :                 :SERVICE---            :CAL
          :   :MULTI SAMPLE     :DEBUG                 :
          :   :ON/OFF                                  :PRIMARY
          +---:                                        :PATTERN
              :SOFTWARE VERSION                        :
              :                                        :FOCUS
              :HOLDOFF DITHER ON/OFF                   :PATTERN
```

Fig. 7. *Menu tree invoked by* **INST STA** *(instrument state) key.*

that are switched in to drive the CRT deflection amplifiers directly. In the case of a vector diagram, I is displayed along the X axis while Q is displayed along the Y axis, resulting in W_{IX} and W_{QY} weights equal to 1.0 and the remaining weights equal to 0.0. In addition to three-dimensional rotation, the features of phase adjust and quadrature adjust are also implemented by adjusting these weight terms based on the angles desired. All of the weight terms translate directly to coefficients for analog multipliers in the instrument. The purpose of the dc voltage calibration is to determine how to set up switched attenuators, analog multipliers, offset controls, and position adjusts to produce any general-purpose display mode.

DC Voltage Calibration. The dc voltage calibration can be broken down into two successive sections: the RF gain path and the video gain and three-dimensional rotation circuit path. All of the voltage calibration is performed using a precision 0.0V or 0.5V voltage source with an accurate 50.0Ω source resistance. This calibration signal is switched into the instrument's effective input in place of the user's I or Q channel signal. Calibration of the RF gain path consists of measuring and calculating the relative gains for the range of possible switched attenuator combinations (Fig. 8). This involves switching the 0.0V and 0.5V sources into both channels for the attenuator combinations that will not overrange the input amplifier, and measuring the I-VIDEO and Q-VIDEO signals that are tapped off following the samplers. Up to 1000 analog-to-digital converter (ADC) readings are averaged for the cases where the signal amplitude differences are small and any residual noise is significant. By making four switched attenuator range measurements, it is possible to calculate the relative gains of the remaining three ranges accurately.

The three-dimensional rotation circuits and the position circuits require the greatest number of instrument calibration factors. The first task is to align the CRT digital grid and text information to the analog information in the form of the X Rotated and Y Rotated signals. This is done by writing successive dots at the two diagonal corners of the CRT display grid and measuring the values of the voltages driving the CRT deflection amplifiers. Once these opposite corner voltages are known, the voltages that correspond to any point on the display grid can be calculated. The subsequent calibration is used to determine how to set up various DACs in the instrument so that X Rotated and Y Rotated will produce proper voltages at the CRT deflection amplifiers to line up the user's signals with the CRT grid.

The multipliers used to implement equation set 1 each require three calibration factors for proper setup. The first factor is the null for the signal line. This is defined as the value of the offset DAC that produces an effective input of zero to the signal terminal of the multiplier (Fig. 9). This value is measured by varying the multiplier coefficient with a square-wave input while adjusting the offset DAC until the output of the multiplier remains constant. The value of the offset DAC is the null calibration factor for the specific attenuator range and input coupling. To handle offsets in the RF path accurately, the instrument calibration measures the signal line null for both input couplings with all seven switched attenuator possibilities.

The second multiplier factor is the coefficient null. This factor is measured by using the offset DAC to generate a square-wave input to the multiplier while adjusting the multiplier coefficient until the output of the multiplier remains constant. Finally, the multiplier gain is calibrated by using the 0.5V calibration voltage and adjusting the multiplier coefficient DAC until the proper deflection on the screen is obtained. This result is used to relate the multiplier coefficient to a specific user volts/division setting on a given range. Using the RF gain calibration factor ratios, this one reading can be used to set the volts per division across all attenuator ranges accurately.

Trigger Calibration. The internal trigger calibration for the HP 8980A is necessary to calibrate the gains and offsets in the trigger signal path. The internal trigger signals are produced by tapping the input signals, buffering and level shifting them, selecting the proper channel, and then driving a high-frequency trigger level comparator and Schmitt trigger (Fig. 10). The instrument allows the user to set up various conditions of input coupling, trigger coupling, switched attenuator settings, trigger slope, and internal trigger level.

A sequence of only eight different conditions must be measured per channel to model all of the possible user internal trigger setups accurately. During calibration, the instrument input is held at a precise 0V or 0.5V level. The trigger DAC which is used to set the variable trigger level can be stepped to create an artificial square-wave input to the trigger level comparator. The trigger level comparator may or may not generate a trigger signal that will initiate a timing ramp. Using a special setup, a single-shot trigger event can be detected in the timing system of the instrument. For each of the trigger calibration factor setups, a binary search algorithm is used to determine the trigger

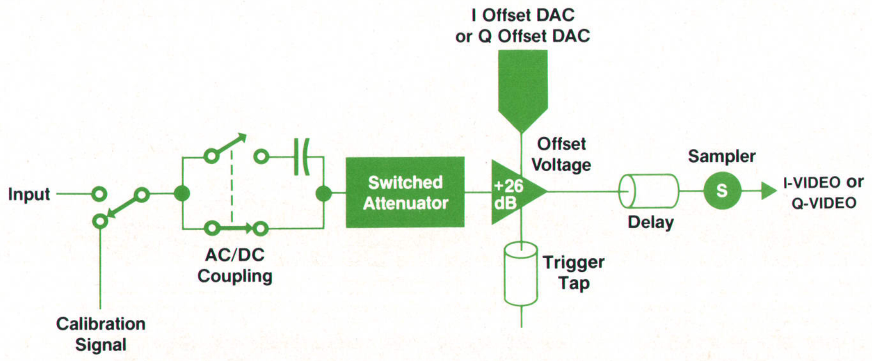


Fig. 8. *Calibration of RF gain path requires measuring the relative gains for the various possible switched attenuator combinations.*

DAC value that corresponds to the dc input voltage used. To test each value in the binary search, the trigger DAC is reset to one extreme or the other based on the trigger slope that is being calibrated. This allows a new positive-going trigger edge to be generated if the trigger DAC level is stepped across the input voltage level. When a trigger DAC calibration factor is determined, it is checked to guarantee that it is accurate and not a result of a stuck bit affecting the binary search.

Timing Calibration. The timing system of the HP 8980A requires calibration of the timing sweep and of the various possible delays. Accurately setting up the timing sweep requires that the slope of the timing ramp be known in volts/second. This value is measured for specific setups on each of the six timing ramps. Internal to the instrument, a variable-frequency, crystal-based timing signal is generated and can be switched into the I and Q inputs. This signal covers a range of 100 Hz to 16 MHz, which allows two cycles of the calibration timing signal to be displayed for each ramp range under test. The fine adjustment of the ramp slope is used on each ramp range to bring the first and second falling edges of the positive-edge-triggered square wave to positions at 25 and 75 percent of full screen.

The timing sweep calibration requires that sharp edges with virtually no timing jitter be analyzed to determine their position on the screen. The instrument's time marker has an internal resolution of 10 bits, which translates to 1024 distinct settings across the display screen. The timing sweep calibration uses steps of 100 time marker counts to window in on the falling edges under test. It then steps 10 counts at a time to fall in close proximity to the edge. Finally it is moved along one count at a time until the edge is crossed. When single-count steps are used, 25 measurements of the signal are averaged to reduce any error caused by jitter on the calibration signal. This averaging allows an accurate determination of the closest time marker value to the actual threshold. This can result in a half count of time marker accuracy in this measurement.

The time marker values corresponding to two adjacent falling edges of the calibrated timing signal are determined (Fig. 11). The time marker is frozen at each of these edges, and the ramp comparator voltages for these two points are measured. The ramp comparator voltage difference and the precise period of the input signal in seconds are then used to calculate the ramp slope in seconds/volt. Knowing the six different ramp slopes from calibration, the time base can be set up accurately from 500 ps/div to 2 ms/div.

Display Refresh Control

The HP 8980A uses a modified high-resolution HP 1345A Display. The modifications allow the HP 1345A Digital Display information to be brought out of the display and multiplexed with the analog X Rotation and Y Rotation signals at the output of the three-dimensional rotation circuitry. This multiplexed signal is then used to drive the CRT deflection amplifiers directly. The major advantage of this method compared with using a purely digital display is the significantly faster rate that sampled points can be written to the screen. The HP 1345A Display requires a SYNC pulse to produce a rewrite of the digital grid and text that is stored in CRT memory. Generally, about 20 percent of the time is spent writing the digital grid and text information at a rate sufficient to avoid flickering. This leaves approximately 80 percent of the time for switching to displaying the user's analog input signals.

The challenge in using a system such as this is that synchronization could occur between the user trigger signal and the digital signal refresh. As an example, if this system used a fixed 60-Hz digital refresh rate and the user analyzed a signal with a 60-Hz trigger rate, it could be possible that the user's signal would not be seen. The I and Q versus TIME display mode will also run into difficulties if a fixed refresh rate is used. In this case, sections out of the alternating sweeps might be missed at lower trigger rates. These considerations resulted in the need for an adaptive system to handle digital refresh and alternating sweeps in the I and Q mode.

The inputs to the system are a pulse from an internal 120-Hz timer, an end-of-timing ramp pulse, and an end-of-timing sweep pulse. At low sweep and trigger rates, it is necessary to monitor the end-of-timing ramp pulse to avoid starting a slow timing ramp and then interrupting it before sampling and displaying a point. The end-of-sweep pulse is used to guarantee that the same number of traces are drawn for each channel in the I and Q alternating display mode. This avoids problems where one channel would be brighter, then dimmer, then brighter, etc. The end-of-sweep pulse is also used to guarantee that switchovers are done after a sweep is complete.

The 120-Hz timer is used to guarantee that the digital information will always be displayed after various timeouts based on the user's sweep mode selection. When users are in the AUTO SWEEP or FREE RUN modes, the system will

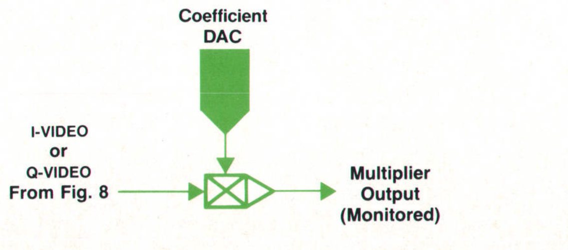


Fig. 9. *Multiplier setup for measuring gain of circuit in Fig. 8.*

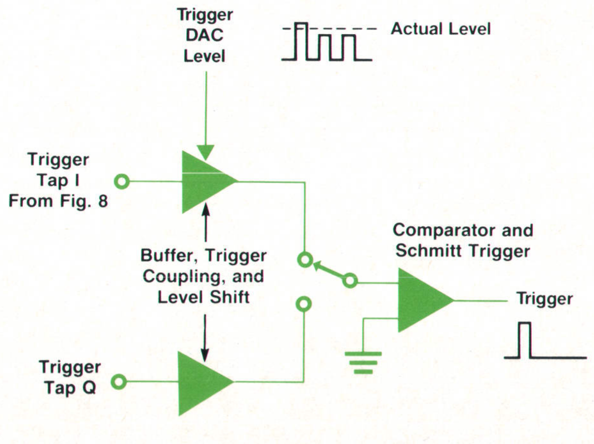

Trigger DAC Level
Actual Level
Trigger Tap I From Fig. 8
Buffer, Trigger Coupling, and Level Shift
Comparator and Schmitt Trigger
Trigger
Trigger Tap Q

Fig. 10. *Internal trigger generation circuitry.*

automatically generate a trigger signal if the trigger rate is too low. This simplifies the digital refresh and alternate functions since predictable timing sweeps are generated and can be interrupted at specific times to display the digital information. In the TRIGGERED SWEEP mode, the signal integrity on screen is maintained down to a trigger rate of less than 5 Hz. For this reason, the digital information is allowed to flicker to allow clean viewing of the user's signal in the TRIGGERED SWEEP mode. At trigger rates below 5 Hz, the signal may appear broken up, but it can still be measured.

HP-IB Firmware

The HP 8980A's HP-IB firmware consists of three separate blocks of code:

- HP-IB driver
- Interface routines
- Command routines.

The first block is written in 68000 assembly language and handles the major portion of the IEEE 488.2 standard implementation. In addition, it handles many of the IEEE 488.2 common commands. This powerful module is reusable.

The second block of code is a set of interface routines that provides input and output data formating, error checking and reporting, and calls to the third block of code which implements the commands. As described later in this article, a large portion of the code in the second and third blocks was generated automatically from a single data base. Both the second and third modules are coded in C.

IEEE 488.2 Standard. The HP 8980A is one of the first Hewlett-Packard instruments to implement the IEEE 488.2 standard. This standard can be thought of as a shell around the older IEEE 488.1 standard. The IEEE 488.2 standard provides five major contributions to help solve the problem of interfacing with programmable instrumentation:

1. Required interface capabilities. A minimum set of capabilities is defined that all instruments must implement. All instruments must be able to receive commands, send back data, respond to a device clear command, and request service. In addition, a minimum set of capabilities is defined if the instrument implements the controller, remote/local, or parallel poll functions. This helps simplify system design.
2. Common commands. Ten different sets of common commands are defined. These sets are autoconfigure, system data, internal operations, synchronization, macros, parallel poll, status and event, trigger, controller, and stored settings. One advantage of having common commands is that software modules can be written and used with all IEEE 488.2 instruments.
3. Data formats and communication protocol. The standard specifies how data is sent and received and when it can be sent and received. It defines states and state transitions for the device. These states are idle, read, query, send, response, done, deadlock, unterminated, and interrupted. This eliminates the past problem where not all IEEE 488 compatible devices could communicate with each other.
4. Status reporting. A status reporting model is defined to provide interrogation of the device's current state. The model includes a structure to report command syntax, protocol, and device dependent errors. It also standardizes the use of the status byte register and the output queue. This gives programmers a consistent way to obtain the status from all devices.
5. Data formats and syntax. The IEEE 488.2 standard specifies formats for decimal and nondecimal numeric data, character data, string data, expression data and block data. It also defines the syntax for encoding and response data. The formats and syntax for the response data are strict, allowing predictable data to be transmitted. At the same time the formats and syntax for the input data are more varied and allow flexible listening capability. This makes it easier to write programs to control HP-IB instruments. It also provides compatibility with older devices. In addition the 488.2 standard permits the use of compound or hierarchical commands. This limits the number of unique commands required for an instrument and helps organize commands into functional groups.

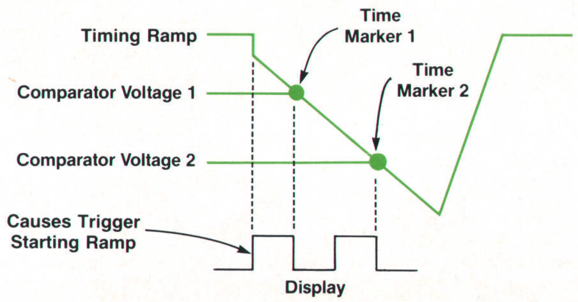


Fig. 11. *Timing sweep calibration first requires determining the two time marker values, after which the two dc ramp comparator voltages can be measured to calculate the ramp slope.*

Mnemonics. The HP 8980A has over 150 HP-IB commands separated into four main groups:

- IEEE 488.2 common commands
- HP 8980A device common commands
- HP 3709A compatibility commands
- HP 8980A device hierarchical commands.

The hierarchical commands are divided into nine hierarchical subsystems: Channel I, Channel Q, Demod, Display, Measure, Service, Timing, Trigger, and Waveform. Each subsystem is used to control or acquire data from the instrument. To execute a hierarchical command from a BASIC program, the hierarchical subsystem and the command must be sent. For example:

```
OUTPUT 709;"TRIG:COUP AC"
```

Suffixes are permitted with the IEEE 488.2 standard. They serve two functions: to allow numeric data multipliers and to make the statements more readable. For example:

```
OUTPUT 709;"CHANI:SENS 50MV"
```

Some mnemonics have alternate forms to allow more flexibility. For example, to change the sensitivity of channel two, we could send either:

```
OUTPUT 709;"CHANQ:SENS 1V"
```

or

```
OUTPUT 709;"CHAN2:SENSITIVITY 1V"
```

Both commands will change the sensitivity to 1 V/div.

Alternate commands are permitted where applicable to make the HP 8980A compatible with other Hewlett-Packard oscilloscopes and with the HP 3709A, a lower-cost constellation display.

Software Development Techniques

This section describes techniques used during the development of the HP-IB and help text firmware.

HP-IB Interface Code. During the development of the HP-IB firmware, we were faced with the problem of efficiently tracking the details of over 150 HP-IB functions being implemented by two different software teams. In addition, we needed to cope with constant changes to the External Reference Specifications. Our goal was to create a document describing each HP-IB mnemonic and its software interface in such a way that a computer could generate code directly from that document. The challenge was to get maximum code out of the document and yet retain a readable document. We achieved this by prescribing a consistent structure for the document, and introducing a simple grammar easily interpreted by the computer.

The document is organized in sections, one for each HP-IB mnemonic. Since almost every HP-IB command has a query associated with it, both command and query reside in each mnemonic section of the document. Each section is divided into three parts. The first part consists of documentation about the mnemonic itself and its use in the instrument. This part can be extracted to generate a comprehensive HP-IB mnemonic document for use by technical writers. The next part contains information about the mnemonic name and parameters. This information is used to generate input for a parser table generator for the IEEE 488.2 mnemonic parser. The third part is the programming section, which contains tables describing the mnemonic in terms of the software interface. The tables include information such as the internal mnemonic token name, numerical parameter variable names, scaling factors, toggle bit polarities, and user interface routine names.

Because the query part of the mnemonics and the HP-IB mnemonic parser were handled by a different software group than the command part of the mnemonics, each group had its own code generator routines to scan only the relevant part of the document. This way each group could make changes to their parts in the document without affecting the other group's part.

We wrote the code-generating routines on the HP 9000 Series 500 under the UNIX™ operating system, using the UNIX lex and yacc facilities for parsing. The output of these code generators is C language code that links the HP-IB mnemonic dependent code with the mnemonic independent code.

The advantage of this scheme is that we had only one document to maintain, and all information about an HP-IB mnemonic was kept in one place. Since the document formalizes the software interface between the two software groups, it greatly reduced misunderstandings. To change the mnemonics, an engineer need only modify this document and the interface code falls out free.

Unfortunately, not all mnemonic interface descriptions conformed to the prescribed format. About 30% of the mnemonics deviate from it, either partially or completely. We also discovered that the document grew too big to be maintained easily. Our revised approach uses one file per mnemonic for descriptions, and the scripts and programs that generate code will take groups of files as input rather than a single file as input.

Help Function Code. The help function code is completely generated automatically using UNIX scripts and some C programs. The help text source document resides in one file organized by sections, one for each help screen. The help text is entered exactly as it is later seen on the screen. Each section is preceded by a header line containing information for the code-generating scripts. This information indicates whether text should be reused from other sections, or if the softkey menu belonging to the key is to be displayed along with the help text.

The code-generating scripts scan this document and a C include file that contains all key codes for the instrument. The scripts ensure that each key has help text associated with it, and then generate three output source files:

- A Huffman code file for the help text
- A dictionary file for the Huffman codes
- A file that contains C code to decode the keys and calls to the appropriate display routines.

This scheme is invaluable in maintaining an up-to-date help function for the instrument. When a change needs to be made to the text, only one document needs to be updated and one script invoked to get a completely new help function.

Acknowledgments

Many more people made significant contributions to the HP 8980A software. Ramani Pichumani and Bill Wendin designed the low-level hardware interface and low-level measurement system. Jim Jensen provided a reusable numerical parameter processing module and helped out in difficult debugging sessions. Paul Stafford wrote code for the help text compression. Rafael Miranda contributed a major portion of the HP-IB firmware. The low-level HP-IB driver and IEEE 488.2 parser software package was developed by Mike Murphy who also consulted in HP-IB related issues. Kyle Karnosh significantly contributed in software testing and test strategies. Many other engineers in the lab also spent hundreds of hours testing the HP 8980A's firmware.

UNIX is a registered trademark of AT&T in the U.S.A. and other countries.

Vector Modulation in a Signal Generator

The HP 8780A offers a wide variety of modulation in both analog and digital formats. By combining the different modulation types, diverse signals such as Doppler-shifted QPSK for satellite communication can be simulated.

by David R. Gildea and Donald R. Chambers

THE HP 8780A VECTOR SIGNAL GENERATOR (Fig. 1) is a 10-MHz-to-3-GHz synthesized signal generator with special wideband modulation capabilities. It has calibrated scalar, digital, burst, vector, and frequency modulation capabilities not previously available in a Hewlett-Packard instrument. In addition, the HP 8780A has the excellent signal purity, frequency resolution, and amplitude level control characteristics of high-quality synthesized signal generators.

Different Modulation Capabilities

Vector Modulation. Sometimes called I-Q modulation, vector modulation refers to the independent modulation of the I (in-phase) and Q (quadrature phase) components of a carrier signal. Vector modulation is a very powerful scheme because it can be used to generate any arbitrary carrier phase and magnitude. The HP 8780A Vector Signal Generator has two analog baseband vector inputs so that the I and Q components of its synthesized carrier can be modulated individually.

Scalar Modulation. Scalar modulation refers to modulation of the carrier signal envelope or amplitude. A separate scalar input is provided so that both I and Q components can be modulated simultaneously.

Digital Modulation. Digital modulation is a term used in terrestrial and satellite communications to refer to modulation in which digital states are represented by the relative phase and amplitude of the carrier. Digital modulation is a special case of vector modulation because each modulation state can be decomposed into I and Q components. The HP 8780A can be used to produce any digital modulation using the two analog signal inputs. In addition, circuitry is built-in so that the most common digital formats (BPSK, QPSK, 8PSK, 16QAM, and 64QAM shown in Fig. 2) can be produced directly from digital input signals.

Burst Modulation. Burst modulation of the carrier is similar to pulse modulation and can be used in the HP 8780A to gate the carrier on and off using the burst modulation digital input. Burst modulation can be superimposed on any other type of modulation except modulations applied through

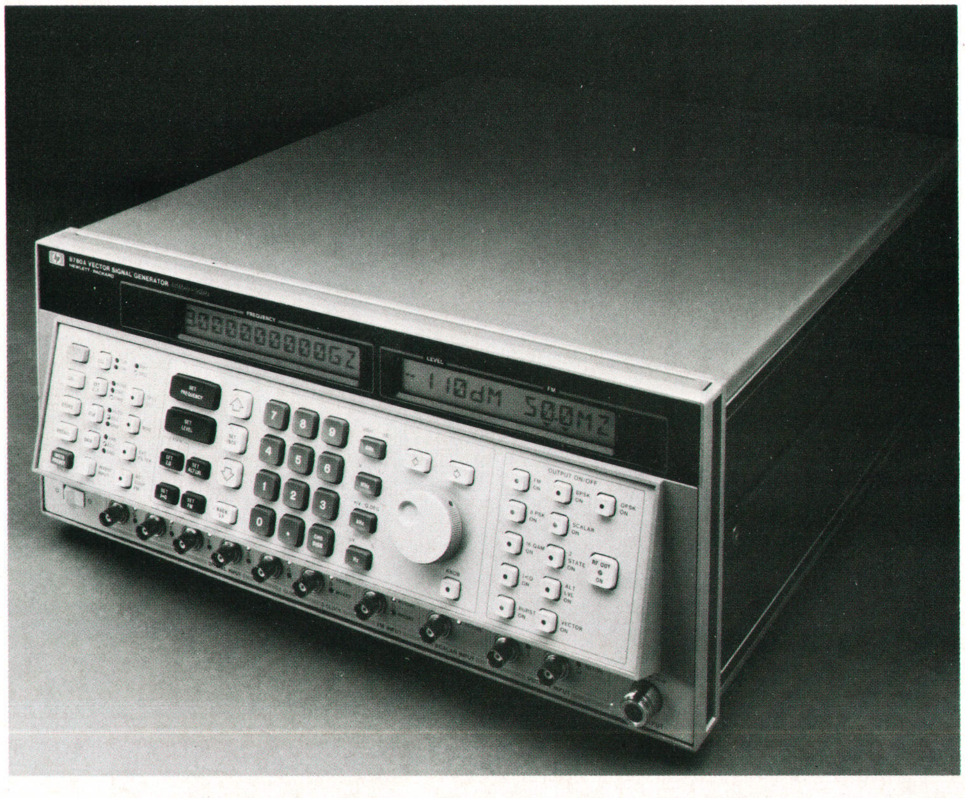


Fig. 1. *HP 8780A Vector Signal Generator is a synthesized IF source with exceptional modulation capabilities for modern receiver and component testing. It generates many standard digital modulations like QPSK and 16QAM and traditional modulations like AM, FM, and pulse.*

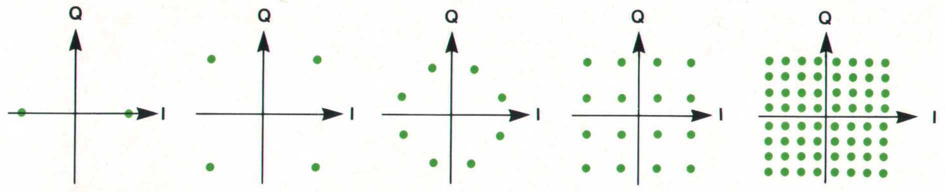


Fig. 2. *From left to right: BPSK, QPSK, 8PSK, 16QAM, and 64QAM in I-Q space.*

the HP 8780A's vector inputs.

Frequency Modulation. Frequency modulation of the output and reference signals can be done simultaneously with any of the other types of modulations.

Block Diagram

A conceptual diagram of the HP 8780A Vector Signal Generator is shown in Fig. 3. The output frequency is derived from an internal synthesizer referenced to an oven-stabilized time base. The synthesized frequency source contains provisions for frequency modulation. Both the coherent output and the RF output have identical frequencies and frequency modulation.

The key to the operation of the HP 8780A is the I-Q vector modulator discussed in detail in the article on page 48. When the carrier signal first enters the vector modulator it is split into I and Q components. The level and phase polarity of each carrier component are controlled in the two vector modulator mixers by analog baseband signals. The I and Q outputs are then recombined. Analog baseband input signals modulate the I and Q components directly in the vector modulation mode. Digital baseband input signals must first be mapped into the desired format and then converted by a high-speed DAC (digital-to-analog converter) into analog I and Q signals. The map can be configured by front-panel keys or through the HP-IB (IEEE 488/IEC 625). Digital, scalar, burst, and frequency modulation, or vector and frequency modulation, can be applied simultaneously.

The complete block diagram of the HP 8780A Vector Signal Generator is shown in Fig. 4. Vector modulation is done at 8 GHz and then converted down to the 10-MHz-to-3-GHz range for the RF output. The 8.01-GHz-to-11-GHz down-conversion signal is generated by multiplying the internal low-noise synthesizer output of 1.00125 to 1.37500 GHz by eight. The 8-GHz IF signal used in the vector modulator is generated by multiplying the 1-GHz output of the FM section by eight. A separate coherent carrier is generated by down-converting the 8-GHz signal before it goes into the vector modulator with the 8.01-GHz-to-11-GHz signal. The 1-GHz signal from the FM section is phase locked to the tenth harmonic of the 100-MHz reference signal from the low-noise synthesizer.

The I-Q vector modulator type was selected because the baseband circuitry necessary to drive it is much simpler to implement than a phase/magnitude modulator. In addition, the I-Q modulator mixers are inherently very linear, broadband, amplitude control elements. The 8-GHz IF for the vector modulation was chosen to obtain the very wide modulation bandwidth. This IF, greater than the output frequency, also avoids carrier feedthrough to the output and most spurious output frequencies.

Vector and Digital Modulation Performance

AM (amplitude modulation), FM (frequency modulation), and PM (phase modulation) all have well understood performance specifications and established measurement techniques. No such performance specifications and measurement techniques exist for vector or digital modulation.

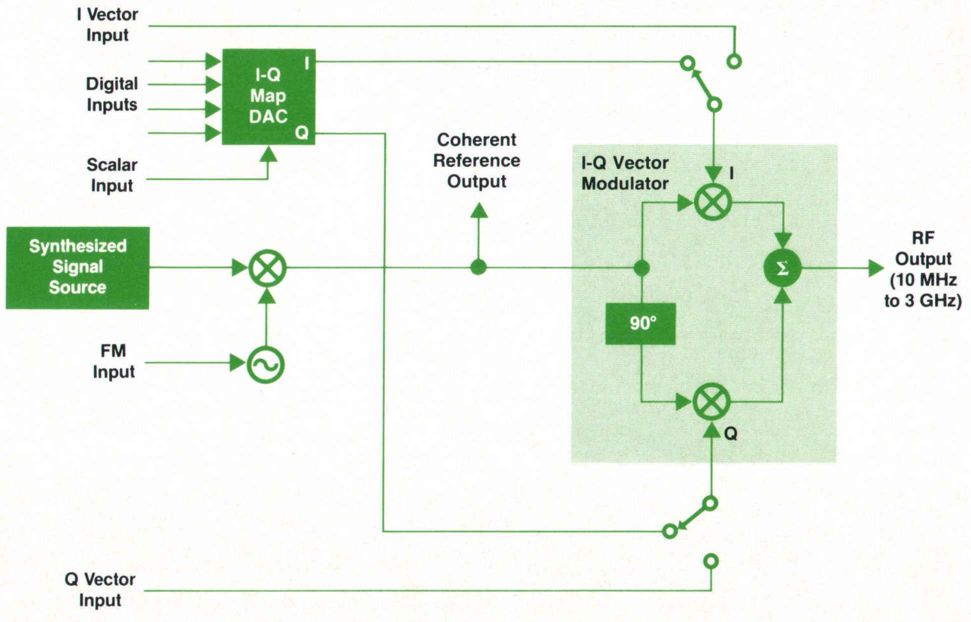


Fig. 3. *Functional block diagram of the HP 8780A.*

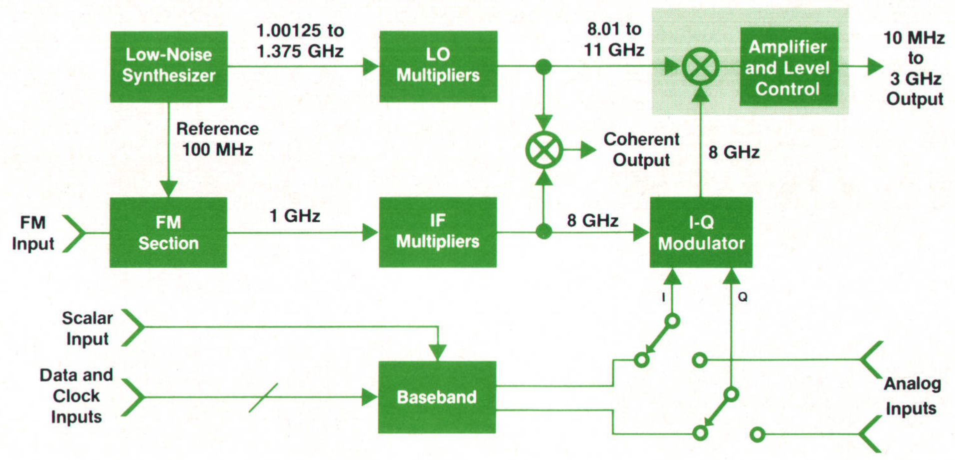


Fig. 4. *System block diagram of the HP 8780A.*

Before the HP 8780A Vector Signal Generator performance could be specified, it was necessary to establish what performance was most useful to customers and how it should be measured. Many engineers working with complex modulation systems were interviewed and the error mechanisms that degrade such systems were analyzed to arrive at the final specification set.

The HP 8780A data sheet specifies vector and digital performance in terms of I and Q accuracy, I and Q frequency response, and I-to-Q crosstalk. The I and Q accuracies compare the actual I and Q modulation levels produced by the generator with the theoretical levels as shown in Fig. 5. For modulation rates near dc, the HP 8510 Network Analyzer is the best test instrument for this measurement. Measured values are typically better than the data sheet specifications (see Table I) by a significant margin.

The I or Q frequency response specification is the flatness of the I or Q modulation level. Since I and Q are measured one at a time in this measurement, a power meter is used. A typical measurement result is shown in Fig. 6.

Crosstalk is the modulation that occurs on I when only Q is modulated or vice versa. Crosstalk at modulation rates close to dc is measured as part of the I and Q accuracies as depicted in Fig. 5. Unfortunately, no off-the-shelf test equipment is yet available to measure ac crosstalk with the necessary accuracy. The test setup shown in Fig. 7 consists of components that have been very carefully selected for exceptional frequency response and impedance match. Line lengths are all kept as short as possible. A typical measurement result is shown in Fig. 8.

Vector Calibration

The three principal errors in the vector modulator are:
1. Carrier leakage, the residual RF output when I and Q modulation levels are both set to zero.
2. I-Q imbalance, the difference between I and Q modulation levels with identical input amplitudes applied to the I and Q channels.
3. I-Q quadrature error, the deviation of the I and Q modulation phases from 90 degrees.

Table I

Digital Vector Modulation Accuracy

State	Specification	State	Specification
BPSK		16QAM	
I	1%	I, Q	2%
−I	1%	−I, Q	2%
QPSK		−I, −Q	2%
I,Q	1%	I, −Q	2%
−I,Q	1%	I, 0.33Q	2%
−I, −Q	1%	0.33I, Q	2%
I, −Q	1%	−0.33I, Q	2%
8PSK		−I,0.33Q	2%
22.5	1.2%	−I, −0.33Q	2%
67.5	1.2%	−0.33I, −Q	2%
112.5	1.2%	0.33I, −Q	2%
157.5	1.2%	I, −0.33Q	2%
−157.5	1.2%	0.33I,0.33Q	2%
−112.5	1.2%	−0.33I,0.33Q	2%
−67.5	1.2%	−0.33I, −0.33Q	2%
−22.5	1.2%	0.33I, −0.33Q	2%

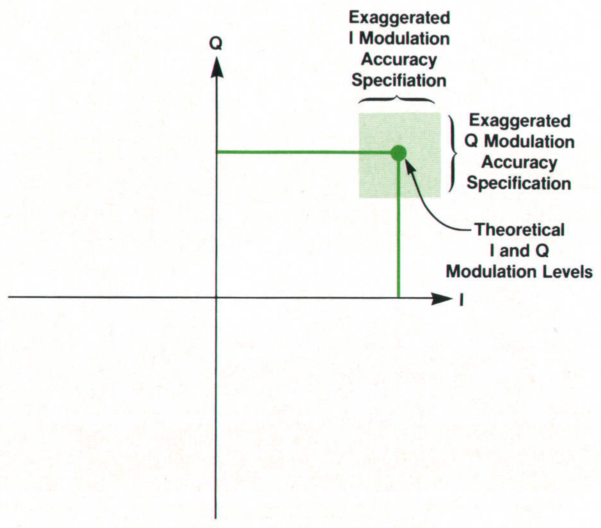

Q
Exaggerated I Modulation Accuracy Specifiation
Exaggerated Q Modulation Accuracy Specification
Theoretical I and Q Modulation Levels
I

Fig. 5. *I-Q accuracy definition shown in I-Q space.*

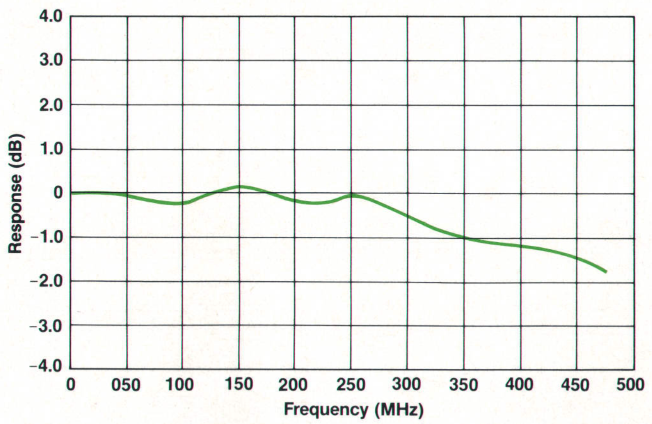


Fig. 6. *Frequency response of vector modulation. Carrier frequency is 1000 MHz, I channel, 7 dBm. The response drop from 0 to 350 MHz is 0.92 dB, while the maximum permissible drop is 2 dB.*

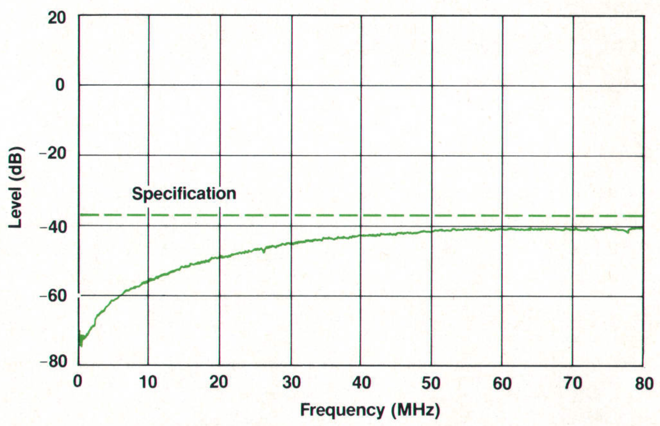


Fig. 8. *Dynamic crosstalk measurement, I channel, versus baseband frequency. Carrier frequency is 500 MHz. The Q-to-I-channel crosstalk is −40 dB at a baseband frequency of 80 MHz.*

To achieve the maximum vector signal accuracy, the I-Q modulator is constructed with compensating adjustments. The internal microcomputer is programmed to adjust the circuits to reduce these errors to a minimum.

Carrier leakage occurs when the vector modulator local oscillator signals leak through the mixers. These leakage signals are compensated for by adding small internal offsets to the modulation levels of the I and Q channels during calibration by the microprocessor. An improvement of typically 40 dB over the unadjusted hardware is obtained.

I and Q amplitude imbalance is equalized in a similar manner using adjustable analog attenuators controlled by the microprocessor. Microprocessor-controlled microwave phase shifters adjust for any quadrature error caused by differences in I and Q signal path lengths in the vector modulator. Fig. 9 shows the adjustment techniques used for the HP 8780A vector modulator.

Since both phase and magnitude adjustments are required to calibrate the modulator, a network analyzer might be considered necessary to achieve calibration. However, an algorithm was developed that uses an amplitude-only detector. Since the carrier leakage, I-Q imbalance, and I-Q quadrature are interactive, the algorithm iterates the adjustments until no further changes are observed. First, the I and Q leakages are alternately adjusted until the total leakage is minimized. Then the I-Q amplitude balance is adjusted until the detector measures equal levels for the conditions of unit signals applied to I only and Q only. I-Q quadrature is adjusted until the equation for the magnitudes of the modulation states:

$$|(I,Q)| + |(-I,-Q)| = |(-I,Q)| + |(I,-Q)|$$

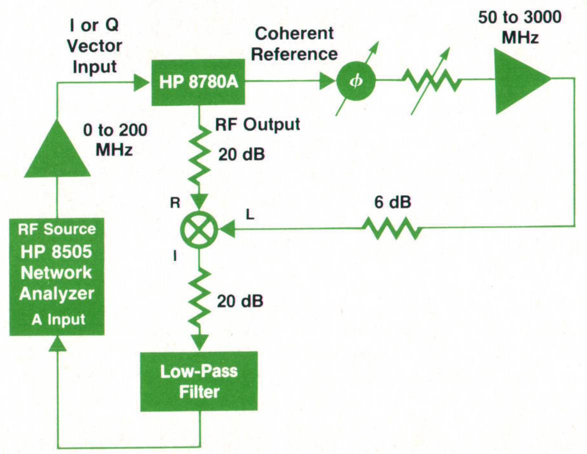


Fig. 7. *Dynamic crosstalk test setup. By varying the phase of the coherent reference, and selecting either the I or Q vector input, one can measure I-to-Q or Q-to-I crosstalk.*

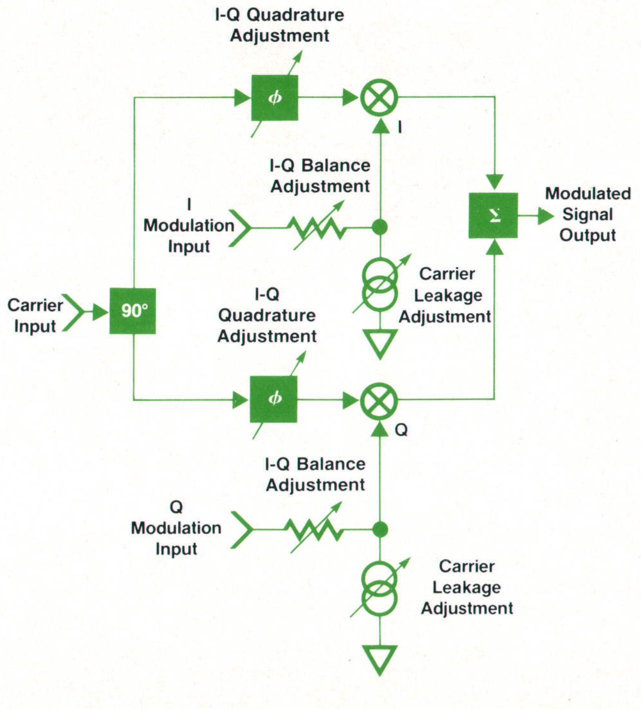


Fig. 9. *Improved vector modulator with microcomputer-controlled adjustments.*

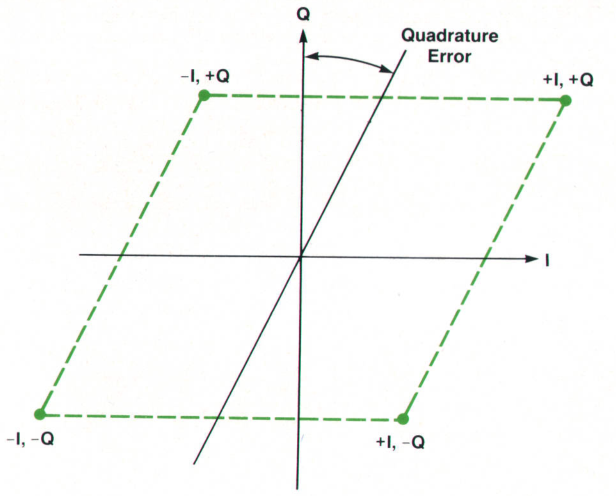


Fig. 10. *The effect of I-Q quadrature error drawn in I-Q space.*

is satisfied. The ordered pair (I,Q) represents the vector output in the first quadrant of I-Q space. Fig. 10 shows the four calibration states in I-Q space for a condition of exaggerated quadrature error.

Spectral Purity

A disadvantage of the high-IF block diagram is that all of the phase noise of the microwave 8-GHz and 8.01-to-11-GHz signals is converted to the output frequency. To get spectral purity that is acceptable to the 70-MHz user, the microwave signals must be exceptionally low-noise. Because both microwave signals are derived from the same 100-MHz reference, the multiplied reference noise within about 100 Hz of the carrier is partially canceled in the output down-conversion. At offsets greater than 100 Hz from the carrier, the low-frequency synthesizer is the key to the spectral purity of the HP 8780A. Because the synthesizer's phase noise and spurious outputs are multiplied by eight during conversion to the output frequency, they must be 18 dB better than required at the output to achieve the desired performance specification.

Acknowledgments

Phil Foster led the mechanical design effort and developed the modular shielding concept used throughout the HP 8780A. Charles Cook, Barry Rose, and Jamie Wandler contributed to the mechanical design of internal packages and the overall instrument. Les Brubaker worked very closely as part of the introduction team to set up the manufacturing facility and contributed significantly to the manufacturability of the final design. George Pontis developed the GaAs integrated output amplifier design. Special thanks are due the IC group at Microwave Technology Division, especially Joe Christiano, Val Peterson, Don D'Avanzo, and Don Estreich for their untiring work on the IC manufacturing processes.

Firmware for a Vector Signal Generator

by James E. Jensen and Eric D. McHenry

THE FIRMWARE USED to control the HP 8780A Vector Signal Generator is written in both C and assembly code and uses approximately 180K bytes of ROM. The majority of the code is written in C, chosen for its speed, efficiency, and flexibility. When the compiled C code for a particular function did not meet our speed requirements, the function was rewritten in assembly language. Such assembly language routines account for 10% of the code in the HP 8780A. Metrics show that 25% of the code is related to calibration, 18% to instrument hardware setup, and the remainder to the HP-IB (IEEE 488/IEC 625) and user interfaces.

Power-Up and Self-Test

On power-up, the HP 8780A firmware (Fig. 1) first runs a self-test program. The self-test checks ROM checksums, checks for RAM functionality, and measures several voltages with an analog-to-digital converter (ADC). If any errors are found, an attempt is made to display an error message on the instrument's display. Of course, the failure can affect the self-test program itself, making the display of the error message impossible. However, there is an LED indicator inside the HP 8780A that will also display an error number. This LED requires a minimum of hardware to function and may allow an indication of an error when front-panel display is not possible.

When the self-test is complete, the firmware checks the battery-backed RAM to verify that its contents are valid and then initializes the instrument hardware to the settings and calibration data stored in the RAM. If the RAM contents are not valid, due possibly to the battery needing replacement, a warning message is generated and the instrument is initialized to the preset state. At this point the firmware is in a tight polling loop waiting for some command from the keyboard or the HP-IB. It is also monitoring the HP 8780A hardware for error states that may occur.

Error Processing

Errors may occur in the HP 8780A because of several sources. The hardware may fail, the instrument may need more warm-up time, the user may have connected a high-voltage source to an instrument input, etc. These types of errors are looked for in the main polling loop of the instrument. When an error is detected, the **MSG** key will light up and an error message will be available to the user. When the cause of the error is removed, the error condition is cleared and the **MSG** key light goes out. All of these errors are watched for in the polling loop with one exception—the low-noise synthesizer module is controlled via a serial data bus. The module sends information to the HP 8780A processor at 19.2K baud and can send error information at any time. This high transmission speed requires an interrupt routine to receive the data. The low-noise synthesizer interrupt routine simply buffers the message sent. The main polling loop then processes messages from the buffer.

Front-Panel Operation

When a key is pressed, the firmware leaves the polling loop and runs a routine corresponding to the key pressed. Some keys, such as **FM ON**, cause the immediate change of a hardware setting. Others such as **RECALL** require more information from the user before any action is taken. In these cases the entry of numeric data is required.

The keys used to generate the numeric data for an instrument parameter (such as the number keys, the rotary pulse generator or knob, and the up and down arrow keys) are processed by a routine called enter. This general-purpose numeric entry routine handles number entry for all instrument parameters. enter knows about each type of parameter and its attributes through the use of a table. The table contains information such as which terminators are allowed, the appropriate increment value, where this parameter is displayed, and so on. The table also contains the name of

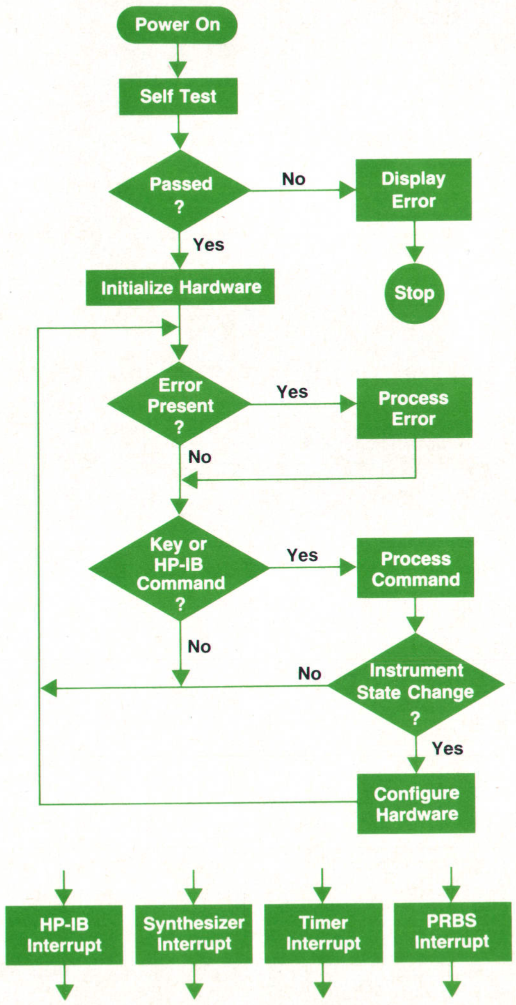


Fig. 1. *HP 8780A firmware flowchart.*

a function to run when a new value is entered. This function checks the numeric data received from enter and then either configures the hardware to that value or generates an error message that an illegal value has been entered. If the value entered is valid, it is saved in RAM and the display is updated with a call to a routine named dspfmt.

The dspfmt routine uses the same table as the enter routine to format the binary data into a string of characters to be displayed on the front-panel display. With these two routines, enter and dspfmt, all instrument parameters are entered, checked, formatted, and displayed. The tables defining each parameter make it very easy to change parameter attributes or add new parameters. The enter and dspfmt routines are also used in the HP 8980A Vector Analyzer firmware for parameter entry. Even though the two products use different microprocessors, the use of the C programming language made it possible to recompile the firmware for different microprocessor environments.

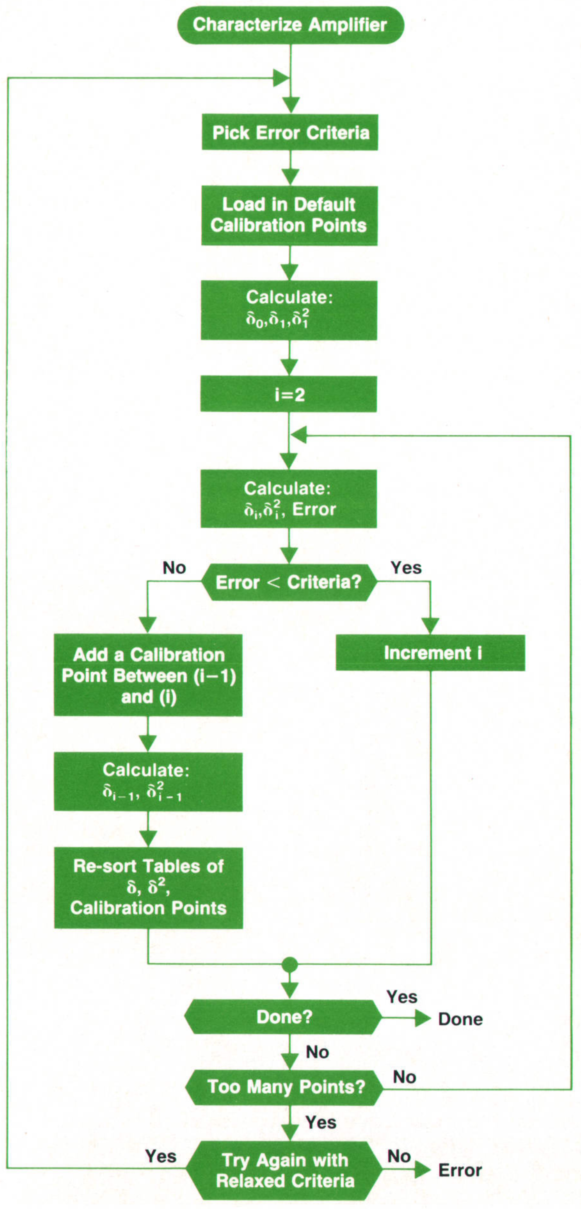

Characterize Amplifier
Pick Error Criteria
Load in Default Calibration Points
Calculate: $\delta_0, \delta_1, \delta_1^2$
i=2
Calculate: δ_i, δ_i^2, Error
No
Error < Criteria?
Yes
Add a Calibration Point Between (i−1) and (i)
Increment i
Calculate: $\delta_{i-1}, \delta_{i-1}^2$
Re-sort Tables of δ, δ^2, Calibration Points
Done?
Yes
Done
No
Too Many Points?
No
Yes
Yes
Try Again with Relaxed Criteria
No
Error

Fig. 2. *Calibration firmware flowchart.*

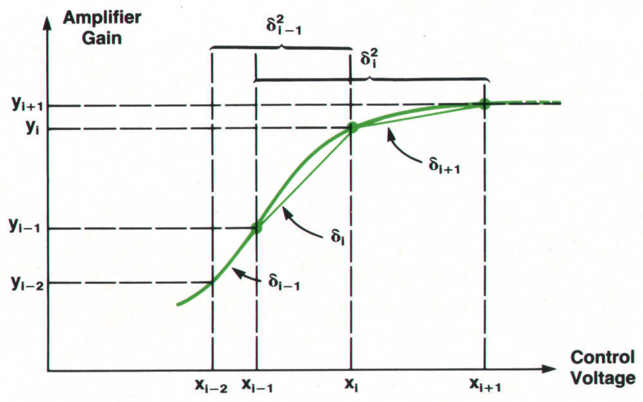
HP-IB Operation

The user can also configure the instrument over the HP-IB. The HP-IB firmware is interrupt driven. Using interrupts it is possible to process some HP-IB commands while the instrument is performing other functions such as calibration. In most cases the HP-IB commands are received from the bus by the interrupt routine and buffered for the main polling loop to process later. These commands are then processed the same as keyboard commands. An exception is when numeric data is received from the bus. The complete number is passed to the enter routine rather than having the enter routine gather each digit. Processing HP-IB numbers in this way is necessary since the HP-IB allows a larger variety of numeric data formats. For example, the value 31.000E−3 and the value 0.031 are both allowed from the HP-IB and recognized as the same value. The front-panel entry does not allow exponential input.

HP-IB commands that are processed completely during the interrupt routine are those commands used for querying the state of the instrument. For example, the HP 8780A is executing its calibration routine and an HP-IB controller would like to know if the calibration has been completed. The HP 8780A allows this query to happen without affecting the calibration operation. Without interrupt capability it would not be possible to acquire some status information during a long instrument operation.

Interrupt Processing

There are two other sources of interrupts. A timer generates an interrupt every 50 ms which is used to update several software timers. The software timers are used by routines to measure time intervals. For example, during calibration the calibration routine waits for an FM loop to lock. A software timer is used to signal when too much



$$\left.\begin{aligned}\delta_{i+1} &= \frac{y_{i+1} - y_i}{x_{i+1} - x_i}\\ \delta_i &= \frac{y_i - y_{i-1}}{x_i - x_{i-1}}\\ \delta_{i-1} &= \frac{y_{i-1} - y_{i-2}}{x_{i-1} - x_{i-2}}\end{aligned}\right\} \left.\begin{aligned}\delta_i^2 &= \frac{\delta_{i+1} - \delta_i}{x_{i+1} - x_i}\\ \delta_{i-1}^2 &= \frac{\delta_i - \delta_{i-1}}{x_i - x_{i-1}}\end{aligned}\right\} \text{Error} < \frac{(y_i - y_{i-1})^2}{4} \times \text{Max}\,(\delta_{i-1}^2, \delta_i^2)$$

Fig. 3. *Linear interpolation error.*

time has passed, that is, the loop should have locked by this time and therefore an error is generated. Other software timers are used for similar functions. The final interrupt source is used for PRBS (pseudorandom binary sequence) generation. PRBS is a special function which generates digital modulation patterns without the use of an external data generator. When PRBS is enabled, an interrupt occurs at about a 5-kHz rate. This interrupt sequences the hardware to the next state in the pseudorandom binary sequence. When PRBS mode is off, this interrupt does not occur.

Calibration

The calibration firmware is a very important part of the HP 8780A. The analog circuits are designed to have their inaccuracies removed through firmware calibration. This allows the HP 8780A to be very accurate while using fewer and lower-precision parts, which reduces the manufacturing cost of the instrument. A large part of the firmware development was devoted to the calibration algorithms.

Output Section Calibration. The calibration of the output section is conceptually easy—a variable-gain amplifier must be characterized by measuring its gain as a function of control voltage and frequency. However, the implementation of the characterization was not easy. The difficult part of the characterization is determining at which of the 4096 possible gain settings to measure the amplifier. Internal RAM is limited, and it had already been determined that the amplifier needs to be measured at eight different frequencies with single-precision (32-bit) floating-point numbers. To measure only half of the 4096 gain settings would have required 65K bytes of RAM. Predetermined calibration points will not yield accurate results because of unit-to-unit variations of the gallium-arsenide (GaAs) amplifier. Evenly spaced calibration points will not work because of varying inflection points in the gain-versus-control-setting curve. What is required is a routine that will determine at what points the calibration data should be taken. The routine needs to be able to cluster calibration points around inflection points and scatter them elsewhere while only selecting enough points to meet the final amplitude accuracy specifications.

The solution to this problem was derived with the help of our statistician. We are able to estimate the error in linear interpolation between two points as a function of the second derivative around those two points. This allows the software to:

1. Initially pick ten calibration points near where inflection points normally occur.
2. Start at one end of the control range and measure the gain at two adjacent calibration points (initially, two of the ten predetermined points).
3. Calculate the first and second divided differences (similar to first and second derivatives) at the two points and calculate the error in dB that would result if a linear interpolation were done between the two points i and i−1:

$$\text{error (dB)} < 0.25(y_i - y_{i-1})^2 \text{Max}(\delta_{i-1}^2, \delta_i^2)$$

where y_i and y_{i-1} are the measured gains at control settings i and i − 1 and δ_i and δ_{i-1} are the second divided differences around i and i−1. See Fig. 2 for a flowchart and Fig. 3 for error equations.

4. Compare the error to the specified error (in our case, 0.01 dB) and if the error is less than the specified value, continue and evaluate the next two calibration points at i+1 and i. If the error is greater than the specified value, the routine must create another point between i and i−1, resort the table of calibration points and repeat the calculation until the error between adjacent points is less than the specified error.
5. If the algorithm completes the calibration with a total number of points less than the predetermined maximum, the calibration points are stored and the characterization routine is finished. If the routine runs out of available RAM, it increases the specified error (for example, 0.05 dB instead of 0.01 dB) and retries the characterization routine to see if it can comply with the relaxed error criteria with less than the maximum number of points. This process will repeat up to four times with increasingly relaxed error criteria before the routine quits with an error and flags the user.

After the calibration points are selected by the characterization routine, the measurement routine measures the gain at eight frequencies at each calibration point and puts the measured points into RAM for a two-dimensional linear interpolation later. Fig. 4 shows typical calibration point distributions for four instruments.

FM Calibration. The calibration algorithm for the FM circuits is designed to measure several variables for the baseband signal processing section (dc to 12 MHz) and for the 1-GHz FM oscillator. After these variables are measured, the firmware generates correction coefficients that are used for each FM sensitivity that is selected. The measured variables are:

- Dc offsets and dc gain through the baseband section
- Gain versus control voltage for the variable-gain baseband amplifier that provides the fine FM sensitivity adjustment
- Attenuation table for the five switchable pi attenuators that provide the coarse FM sensitivity adjustment
- VCO sensitivity of the 1-GHz FM oscillator in MHz/V.

To measure gain versus control voltage for the variable-gain FET amplifier, a routine is required that is tolerant of part and temperature variations of the FET. The routine

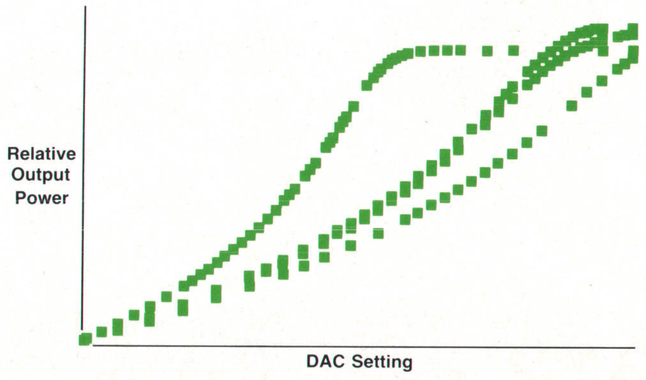


Fig. 4. *Output power versus digital-to-analog converter settings for four instruments.*

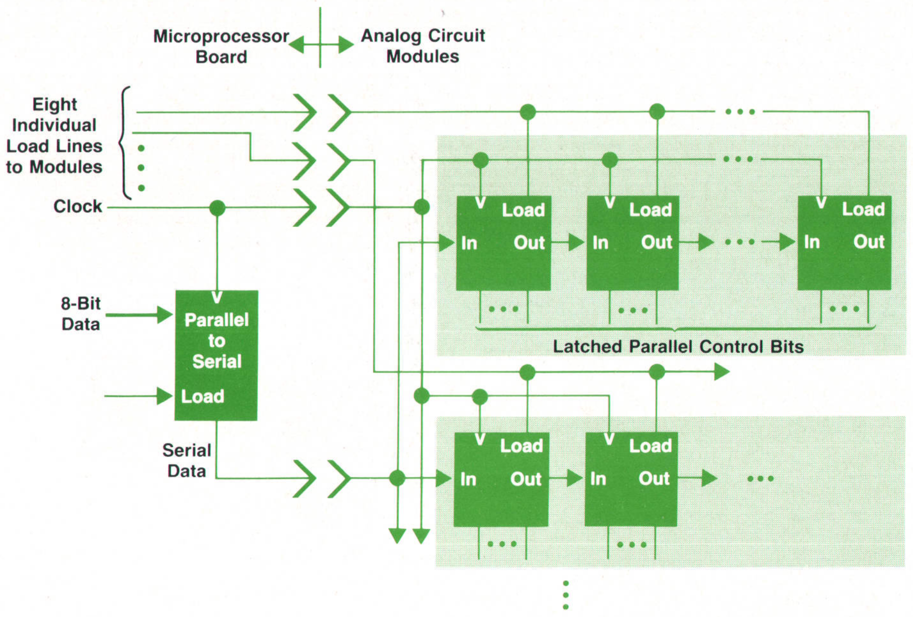


Fig. 5. *Analog circuit interface.*

first applies a control voltage that is increased until the FET goes into its pinchoff region. Then the control voltage is lowered until the FET is operating at a point 1 dB below pinchoff. This creates the upper limit for the FET. The control voltage is then decreased until the FET has a gain 3 dB less than the upper limit; this is the lower limit. Then ten control voltages between the lower and upper limits are selected and the FET gain is measured at these voltages. The measured gains are stored in a calibration table for later interpolation. If there is an error in the routine (e.g., because the FET does not have enough range), the error is logged and the user is notified.

The VCO sensitivity is measured by applying a voltage step to the VCO modulation input and reading a hardware counter designed to count the instantaneous frequency difference between 1 GHz and the actual VCO frequency. This count (measured in MHz) is then divided by the measured voltage step to determine the sensitivity of the VCO in MHz/V. Since the VCO is ac-coupled via a phase-locked loop, it complicates this section of the calibration routine. Applying a dc voltage generates first a step in frequency, then a slow decay in frequency back to 1 GHz. Therefore, the calibration routine must disable time-consuming interrupts during this process to get the correct count before the decay becomes appreciable.

Baseband Calibration. The calibration algorithm for the baseband circuits is described in the article on page 50.

Analog Circuit Interface

The analog circuits are controlled by a serial bit stream from the microprocessor to each of the eight analog modules. The modules contain latching serial-to-parallel registers and receive the clocked serial data common to all modules. The data is latched by a particular module only when the corresponding module load line is pulsed (see Fig. 5).

This interface technique allows some modules to have a high number of control bits by cascading the serial-to-parallel registers while other modules have only a few control bits. The range in the HP 8780A is from one bit per module to 104 bits per module. During the development cycle, the flexibility of this interfacing scheme was important, since individual analog designers frequently increased or decreased the number of bits per module. These changes required only firmware changes, rather than hardware modifications to the I/O section of the microprocessor board. Since writing to analog hardware is only initiated by a user input, such as changing frequency or modulation, the speed reduction imposed by serial data transmission compared with parallel transmission is not a problem.

Acknowledgments

The authors wish to thank Mike Murphy for his work on the HP-IB driver software and several assembly language math routines, Stan Woods for his work on the instrument self-test and diagnostics, Doug Anderson for his work on the FM baseband calibration algorithm, and Karen Kafadar who generated the math for the output section calibration routine.

Low-Noise Synthesizer Design

by Thomas J. Carey, John C. Lovell, and Thomas L. Grisell

THE SPECTRAL PURITY of a signal source is one of its most important characteristics. If the output frequency of such a source is multiplied, then the importance of high spectral purity is magnified. The internal frequency synthesizer of the HP 8780A Vector Signal Generator is designed to achieve a combination of state-of-the-art phase noise and spurious output performance.

Fig. 1 shows the block diagram of the low-noise synthesizer. A synthesis technique based on three phase-locked loops is used. An IF reference loop provides frequency resolution in millihertz steps. This phase-locked loop uses the fractional-N frequency synthesis technique, which allows the output of such a loop to be incremented in frequency steps smaller than the loop's reference frequency. An RF reference loop provides large-step resolution in 20-MHz increments. The outputs of these loops are combined in the output sum loop to provide coverage of the desired 980-MHz-to-1520-MHz output range with millihertz resolution.

Each phase-locked loop is in a separate housing with internal shielding as required. These modular housings ensure a high degree of isolation between the individual loops, which facilitates meeting the −90 dBc spurious performance specification.

The time base and controller sections, housed in a fourth module, consist of a 100-MHz voltage-controlled crystal oscillator (VCXO) locked to the HP 8780A high-stability time base and various multipliers and dividers to generate the necessary signals for locking the main phase-locked loops.

Conventional RF technologies are used in the synthesizer, but the block diagram and circuitry are optimized for the highest performance consistent with the size, cost, and producibility objectives. The range of the variable-modulus divider (P divider in Fig. 1), the VCO frequency ranges, and the phase detector reference frequencies are determined from an examination of the requirements for continuous coverage and an understanding of the relationship between oscillator tuning range and phase noise. For example, the number of steps in the output of the RF reference loop was halved by using both sidebands of the phase detectors in the sum and RF reference loops. The use of a variable-modulus divider at the output of the IF reference

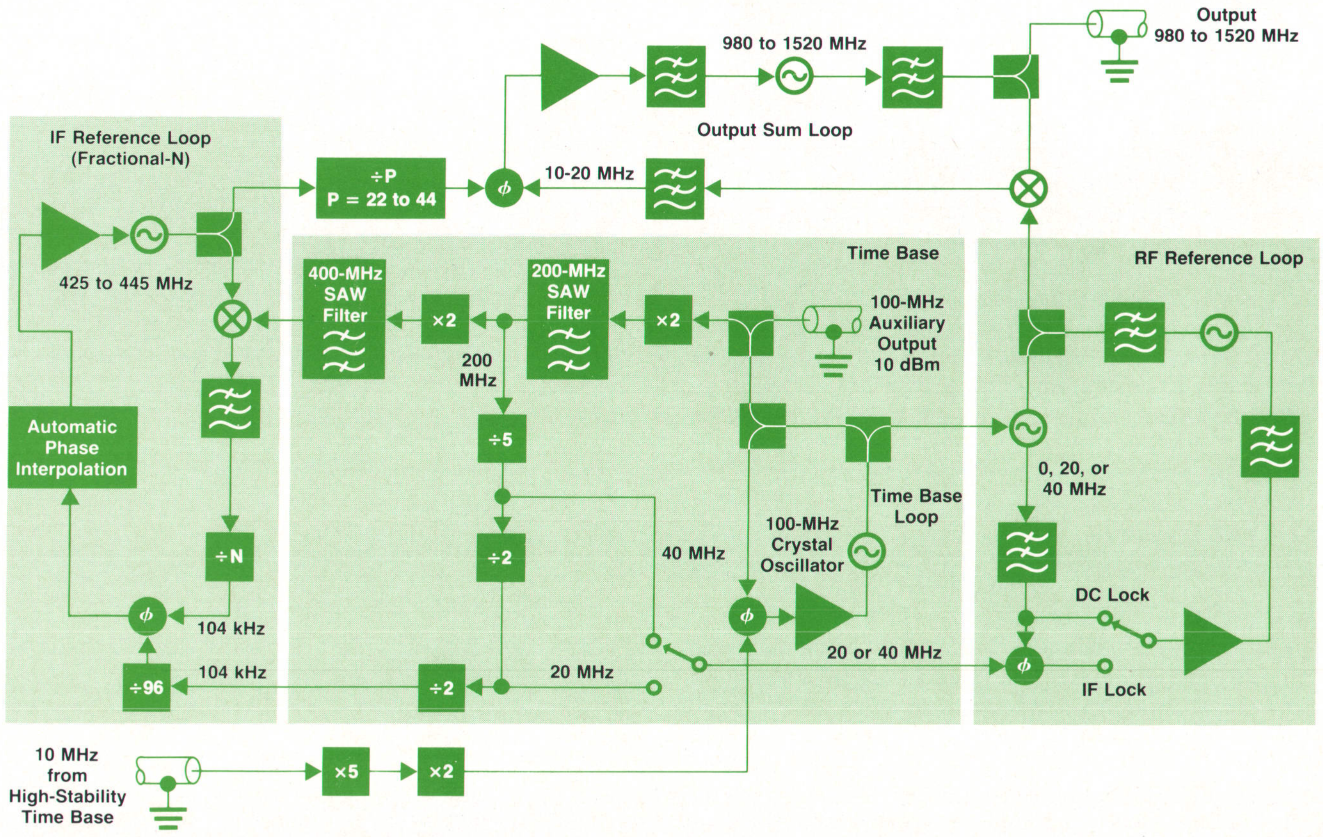


Fig. 1. *Block diagram of low-noise synthesizer used in the HP 8780A Vector Signal Generator. The HP 8780A only uses 1001.25 to 1375 MHz out of the full 980-to-1520-MHz frequency range.*

loop allows this loop's VCO to tune only 25 MHz, thereby preserving good phase noise while producing a 10-MHz-wide IF range for the sum loop phase detector. The maximum sum loop IF frequency (20 MHz) is a compromise between the IF reference loop phase noise performance and the number of steps required from the RF reference loop for continuous coverage. Fig. 2 shows the phase noise of a typical production unit.

Sum Loop

The sum loop takes the outputs from the IF reference and the RF reference loops and combines them algebraically to form the 1001.25-to-1375-MHz local oscillator reference for the HP 8780A. This combining of the two loops' outputs transfers each loop's stability, resolution, phase noise, and close-in (less than 1-MHz offset) spurious signals to the low-noise synthesizer output. A major sum loop design goal was to achieve this combination without adding contaminating spurs and noise.

Any discussion of the many trade-offs in indirect synthesizer design must include the performance of the loop's voltage-controlled oscillator (VCO). The VCO in this block diagram covers 1001.25 to 1375 MHz to cover the HP 8780A frequency range. With a loop bandwidth of 1 MHz and a noise floor goal of approximately −140 dBc/Hz, the open-loop oscillator noise goal would be −150 dBc/Hz at an offset of 1 MHz. The actual oscillator design achieves typical noise performance of −147 dBc/Hz at the 1-MHz offset. This performance is achieved by using two separate oscillator circuits to cover the frequency range. The range is divided into two 270-MHz segments so that the two oscillators' tuning curves have similar slopes, which helps in minimizing loop gain variation versus output frequency. The oscillators share a printed circuit board, along with buffer amplifiers and interface circuitry.

A partial circuit of the oscillator for the lower portion of the band is shown in Fig. 3. The resonator consists of L_t, C1, C2, C_{trim}, and varactor diodes D1 and D2, in a center-tapped, or balanced, arrangement to allow the −10V varactor bias offset to be connected without reducing the Q of the resonator (and increasing the noise). Surface mount components are used for the 2-pF capacitors and the varactors because of the advantage of small size at these frequencies. Q1 is a microwave device which, along with base inductor L_{base}, can be shown to present negative resistance to the resonator to negate its losses and enable the circuit to oscillate. By properly choosing the point where Q1's emitter is tapped on L_t and the signal level at the emitter of Q1, the phase noise can be optimized.

The VCO frequency is controlled by the sum loop board. Fig. 4 shows a simplified diagram of the circuits used to phase-lock the VCO. The sideband switch is four ECL gates connected as a DPDT switch to interchange the signals to the phase detector so the VCO can lock either above or below the RF reference loop frequency.

A familiar form of phase/frequency detector follows, but one in which problems of dead zones and gain variations are not present. This is important, for the phase/frequency detector must be operated at zero phase offset to achieve the required spur and noise performance. Output switches Q1 and Q2 remove noise appearing at the logic levels (important for the phase detector) and also give clean, closely matched output pulses to the loop amplifier. A combination of a low-noise operational amplifier and a feed-forward path using a miniature wideband transformer give accurate dc control to hold a zero phase relationship at the phase detector, as well as wideband performance to avoid introducing unwanted phase shift into the loop, which would cause peaking in the noise floor. Because the phase detector is operating at zero phase shift, its outputs are identical pulses. Since the loop amplifier is a balanced-to-unbalanced converter, its output ideally should have no pulse energy at the phase detector rate. This is not quite true, so some filtering is provided by the tune line filter, which provides 30-dB rejection of tune line signals above 9.5 MHz. This circuitry gives the necessary noise performance while keeping tune line spurs more than 95 dB below the LO synthesizer output. Acquisition is controlled on-board. When the sum loop unlocks because of a reference moving

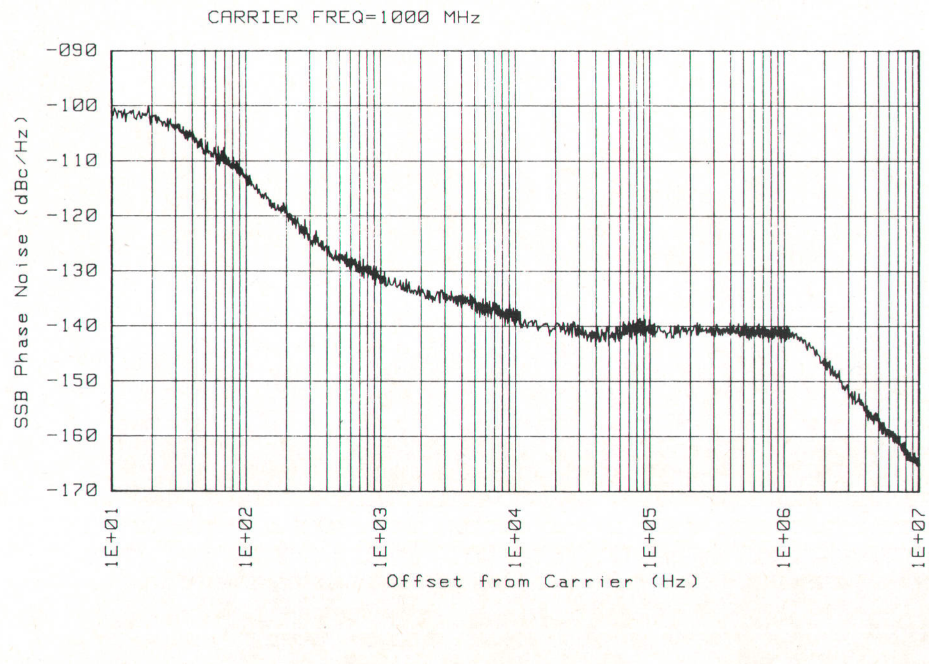


Fig. 2. *Typical phase noise performance for the HP 8780A's low-noise synthesizer.*

out of its IF passband or being removed, a search voltage is applied to the tune line. This sweeps the VCO through its range until it passes through a correct lock point. The lock detector circuitry then disables the sweep, leaving the loop locked.

RF Reference Loop

The RF reference loop (Fig. 5) provides the 20-MHz coarse frequency steps and the low phase noise required by the sum loop to cover the range of 980 to 1520 MHz. The phase noise of the RF reference is determined by the 100-MHz crystal oscillator below 5-kHz offsets, and above this it is limited by the combination of frequency divider noise floors and the multiplication of the crystal noise floor. The loop bandwidth is nominally 1 MHz. The VCO is two oscillators that cover 980 to 1250 MHz and 1250 to 1520 MHz and uses a printed circuit board identical to that used by the sum loop oscillator to cover the output frequency range.

The reference loop uses a 100-MHz sampler to achieve wide frequency coverage from a single low-frequency reference. The sampler acts as a harmonic mixer/phase detector by mixing harmonics of the 100-MHz signal against the VCO to generate an IF signal ranging from dc to 50 MHz. The reference loop operates in two different modes called dc and IF locks. Dc locks occur when the VCO frequency is exactly an integer multiple of the 1-MHz reference frequency and the sampler acts as the loop phase detector. This lock mode requires low input offset voltage and drift from the sampler IF amplifier, contrasted with IF locks, which use the sampler as a harmonic mixer to produce either 20 or 40 MHz for the IF phase detector. This dc-to-40-MHz amplifier is a relatively low-gain, wide-bandwidth, discrete amplifier whose dc operating point is controlled by a conventional operational amplifier.[1] The outputs of the sampler and IF phase detector are switched to drive the loop amplifier which integrates the phase error to produce the VCO tune voltage.

The reference loop presents special locking problems because of the nature of the sampling process. Since the VCO frequency is always within 50 MHz of a harmonic of the 100-MHz sampling rate, the output of the sampler is always between dc and 50 MHz. This is a problem because the VCO can lock at any harmonic of the 100-MHz reference that produces the correct output from the sampler. This possibility is circumvented by pretuning the VCO close to the desired lock point, and then only allowing the loop to search a narrow range around the pretune point. This would be fine if all of the possible lock points were 100 MHz apart, but this is not the case for the 40-MHz IF lock points. For example, 1040 MHz is 40 MHz above the tenth harmonic of 100 MHz, but 1060 MHz is 40 MHz below the eleventh harmonic. These two frequencies are only 20 MHz apart. Because the difference in tune voltage for frequencies only 20 MHz apart is less than the variation in absolute tune voltage, the VCO cannot reliably tune to one frequency instead of the other without additional help.

This help is provided by a quadrature phase detector for the IF lock points. The switched 20/40-MHz reference signal is passed through a 90-degree power splitter to produce coherent quadrature outputs for the main and quadrature phase detectors. When the loop is locked, the output of the main phase detector is zero volts (for zero phase error), but the 90-degree phase difference at the quadrature phase detector causes the output voltage to be positive or negative, depending on which side of the 100-MHz harmonic, or comb line, the VCO is tuned to. By monitoring the quadrature phase detector output it is possible to determine whether the IF lock point is upper sideband or lower sideband and thus gain the desired 100-MHz separation between possible lock points.

IF Reference Loop

A fractional-N synthesis technique is used to produce the millihertz frequency steps of the low-noise synthesizer.[2] This technique is distinguished from more-conventional phase-locked loops by the use of a noninteger frequency divider created by alternating the integer divide number between N and N+1 at a duty cycle determined by the offset from N. Changing the divider modulus this way introduces phase offsets at the phase detector output which are canceled by automatic phase interpolation (API), reducing spurious outputs to less than −70 dBc. A proprietary IC set and a precision resistor array achieve this performance with only one adjustment.

The VCO uses a shorted coaxial transmission line as a resonator and is tuned over the 425-to-445-MHz range with varactor diodes. The transmission line resonator has a Q high enough to use a 5-kHz loop bandwidth so that the close-in output phase noise is not degraded by the IF reference. The 425-to-445-MHz output is divided down by the divide-by-P counter to provide the 10-to-20-MHz reference

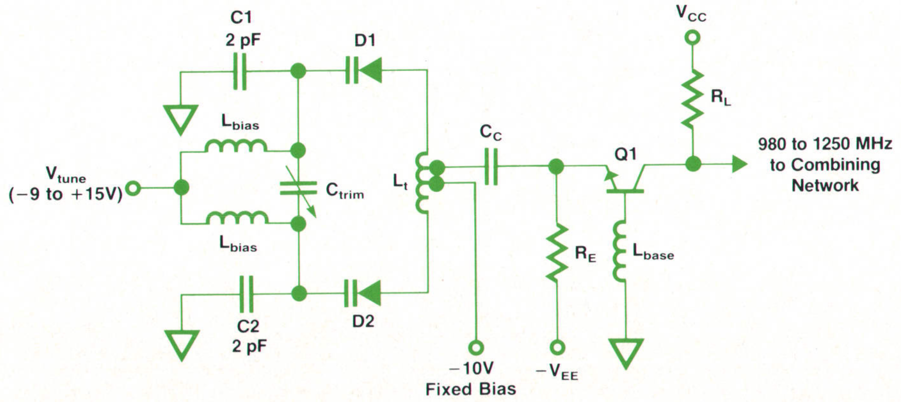


Fig. 3. *One of two VCOs in the sum loop for the low-noise synthesizer.*

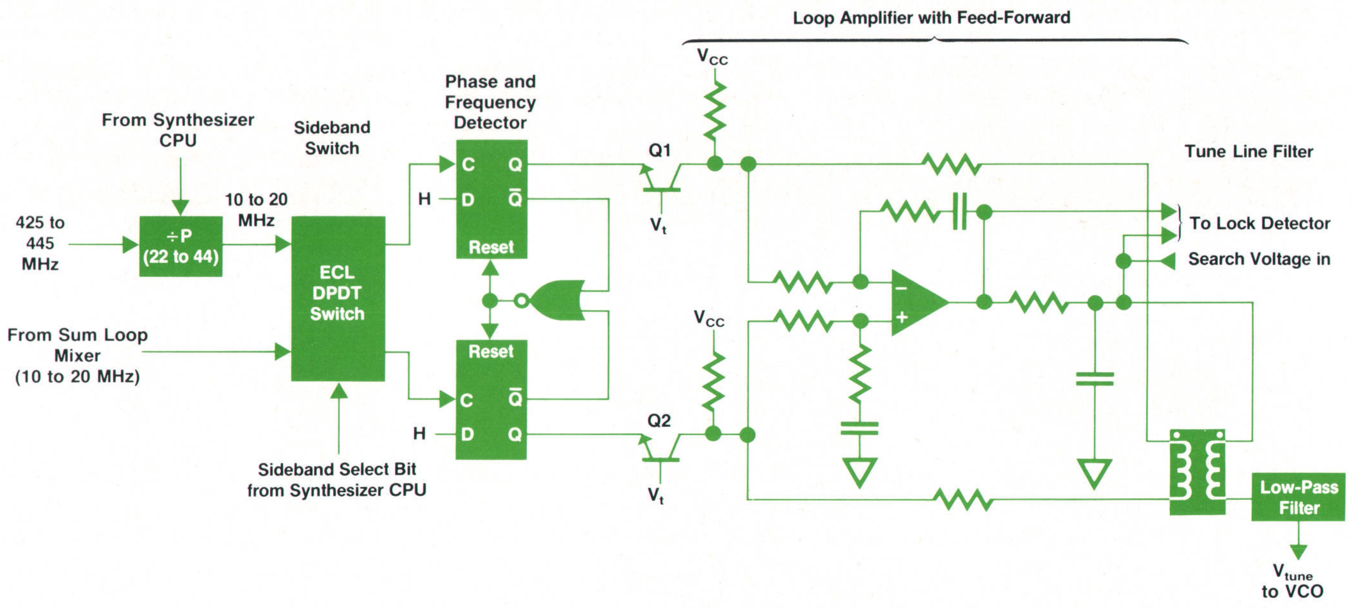


Fig. 4. *Sum loop phase-lock circuits.*

for the sum loop. This division reduces the spurs and phase noise by a minimum of 27 dB to comply with the −90 dBc spur specification.

Time Base

The time base is the heart of the synthesizer. The 100-MHz VCXO controls the stability, accuracy, and close-in phase noise of the frequency synthesizer. This board occupies the center third section of the time base module (the CPU and the divider board occupy the other two thirds). The crystal is shock mounted for reduced vibration sensitivity. The shock-mount consists of a floating board held by a rubber O-ring at each corner, with a brass block on top to give the board sufficient mass. This assembly acts as a 50-Hz mechanical low-pass filter to any vibrations that exist on the main board. The electrical connections to

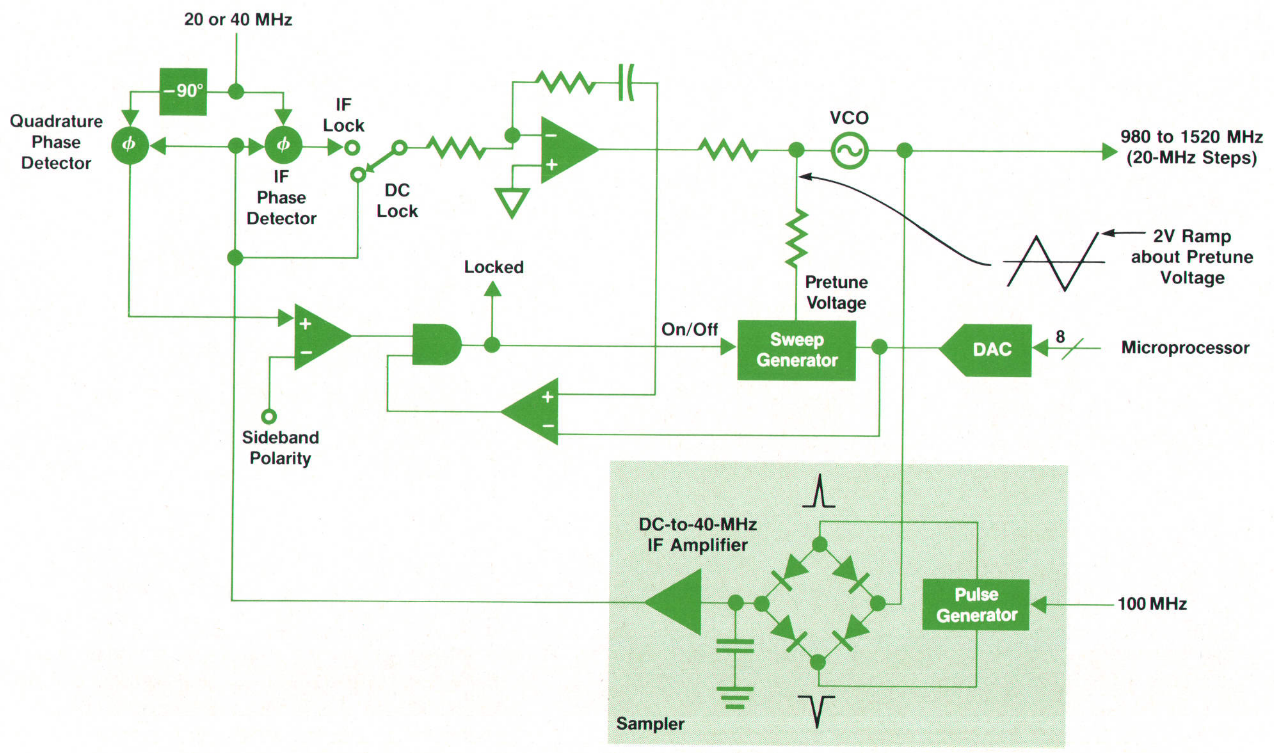


Fig. 5. *RF reference loop block diagram.*

the crystal are brought through special silicone-insulated wires which are flexible enough not to transmit any vibrations to the crystal. Extensive power supply filtering is necessary to provide the very low noise and ripple required for the subsequent multiplication of the 100-MHz signal. Power-line related sidebands have to be 110 dB below the signal level to meet the −90 dB spurious specification.

The time base divider board generates several reference signals for use by the IF reference and RF reference loops. The 400-MHz output used by the IF reference loop is a filtered harmonic of the 100-MHz VCXO. A surface-acoustic-wave (SAW) filter is used to eliminate unwanted harmonics and sidebands.

Synthesizer CPU

The low-noise synthesizer is controlled by a Z8 single-chip microprocessor containing 4K of ROM, 128 bytes of RAM, a clock generator, and 32 bits of programmable I/O. All but 13 bytes of the ROM is used in the synthesizer firmware. The board occupies the left third section of the time base frame, with connections via a 37-pin ribbon cable to the other frames. External communications with the CPU are implemented with a two-wire serial path identical to RS-232-C/V.24 except that it uses TTL levels instead of ±12V. With its internal microprocessor, the synthesizer is an independent module within the HP 8780A that relieves the instrument processor of the calculations of hardware settings and phase-lock monitoring.

Acknowledgments

The high performance of this synthesizer would not have been possible without the contributions of many people. We would like to thank Randal Burnette for his excellent fractional-N design, Eric McHenry for the initial development of the time base and microprocessor sections, Phil Foster for the modular product design, Allen Edwards for the original concept and block diagram, and Andy Naegeli, Dave Gildea, and Don Chambers for their management skills during development.

References

1. K.W. Renner, M.O. Bedwell, and J.F. Pierce, "A Wideband Direct Coupled Amplifier Utilizing a Fast/Slow Loop Concept," *IEEE Transactions on Nuclear Science*, Vol. NS-28, no. 1, February 1981.
2. D. Scherer, et al, "Low-Noise RF Signal Generator Design," *Hewlett-Packard Journal*, Vol. 32, no. 2, February 1981.

Digital and Vector Baseband Circuits for a Vector Signal Generator

By using a vector modulator instead of conventional amplitude and frequency modulators, the HP 8780A has the ability to generate complex digital and vector modulations. The baseband system is vastly different from those found in conventional signal generators. This system also provides the instrument with some special features.

by Chung Y. Lau

OVER THE LAST DECADE, there has been a steady growth in the complexity of modulation employed in RF and microwave systems. Some of the driving forces behind this growth are advances in IC technology, a need for more efficient use of bandwidth, availability of wideband and high-frequency devices, and the quest for superior system performance in electronic warfare and radar applications. Accompanying this growth is the need to generate or simulate these complex modulation signals. Traditional signal generators with their simple AM and FM modulation capabilities and limited modulation bandwidths do not typically meet this need. What is needed is an RF synthesizer with a very wide modulation bandwidth and very flexible phase and amplitude modulation functions. Up to now the approach taken by test engineers when faced with the task of generating complex modulation is to build their own equipment. This usually results in something that is very dedicated, works only over a small range of carrier frequencies and/or data rates, and is difficult to document and maintain. At HP we became aware of the need for such an instrument and the HP 8780A was developed in response to this need. To make the most use of all the modulation capabilities in the HP 8780A, one should be familiar with the description of signals in the I-Q domain. The box on page 42 goes over this concept.

Block Diagram and Features

The key contribution of the HP 8780A is the ability to generate wideband and precise modulation. The main components responsible for this performance are the HP 8780A's flexible baseband circuitry and its wideband linear vector (or I-Q) modulator. A block diagram of the modulation system is shown in Fig. 1. The main function of the baseband circuits is to provide the proper I and Q drive levels to the vector modulator. The two modulations provided are digital modulation and vector modulation. (A third modulation, FM, is provided by another block of circuits and is discussed by the article on page 45.)

Digital Modulation

When digital modulation is selected, the user will specify a modulation format. The following digital formats are built-in:

- BPSK (binary phase-shift keying)
- QPSK (quadriphase-shift keying)
- 8PSK (octaphase-shift keying)
- 16QAM (quadrature amplitude modulation)
- 64QAM (available only with option 064).

In addition, burst modulation can be selected simultaneously with the first three formats. In burst mode, the user can turn off the signal by applying a high logic level to the burst input, independent of the states of the other data

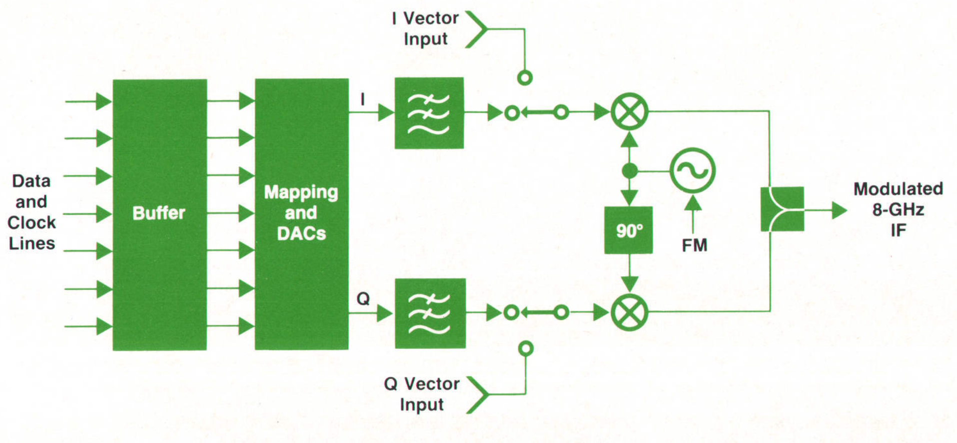


Fig. 1. *Simplified block diagram of modulation system for the HP 8780A.*

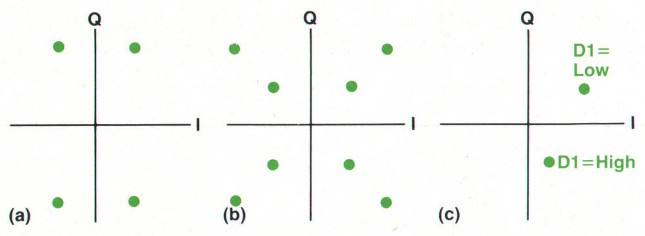


Fig. 2. *Some novel digital modulation features. (a) QPSK with 50% I-Q ratio selected. (b) QPSK with alternate level selected. Inner states correspond to D1 = high. Ratio entered is 50%. (c) Two-state selected. The two states can be arbitrarily positioned in the plane.*

lines. This is useful for those who wish to simulate TDMA (time-domain multiple access) signals where transmission of data occurs in bursts, or for those interested in generating coded radar pulses.

One can also choose scalar modulation in conjunction with any of the above formats. This allows linear control of output amplitude, that is, the output amplitude is proportional to the analog voltage applied to the scalar modulation input. When that voltage is zero, there is no output, and when it is one volt, the full-scale level is present at the output. This feature is useful for those interested in testing ALC (automatic level control) circuits in receivers, or for doing amplitude-fade simulations.

Besides the standard digital modulation formats, there are some special functions:

- **I<Q**, which allows the user to change the gain of the I channel relative to that of the Q channel (Fig. 2a).
- **ALT LVL** (alternate level), which allows the user to specify a second power level, and the output level can be switched from one level to the other at high speed with one data line (Fig. 2b).
- **2-STATE**, which allows the user to specify two arbitrary states in the I-Q plane, and the output can be switched from one state to the other at high speed with one data line (Fig. 2c).
- ADD CARRIER LEAKAGE, which allows the user to add a controlled amount of carrier to the modulated signal. This is useful for testing carrier recovery circuits in radios.
- ADD QUADRATURE ERROR, which allows the user to introduce quadrature errors in the vector modulator. This is useful for simulating nonideal modulators.
- Built-in PRBS generator. An internal PRBS (pseudorandom bit sequence) generator is available, although the symbol rate is fixed at only 5 kHz. This allows the user to generate digitally modulated signals (at a low rate) without an external data generator.

In digital modulation mode, the data rate allowed is dc to 150 MHz in the synchronous (clocked) mode, and dc to 50 MHz in the asynchronous mode. By synchronous we mean that the data lines are clocked by a user-provided clock signal; this gives simultaneous symbol transitions in both I and Q channels. The input data threshold can have a range of −2.5V to 2.5V, with a minimum peak-to-peak level of 300 mV, all measured into 50 ohms, which is the input impedance of the data lines. The clock input level requirements are similar. In the standard (non-64QAM) instrument, the user can select the 2-clock synchronous mode. In this mode the I and the Q channels can have different clocking signals; this makes it easy to simulate offset QPSK signals, or other modulations where the I and Q channels operate at different clock rates.

(continued on page 42)

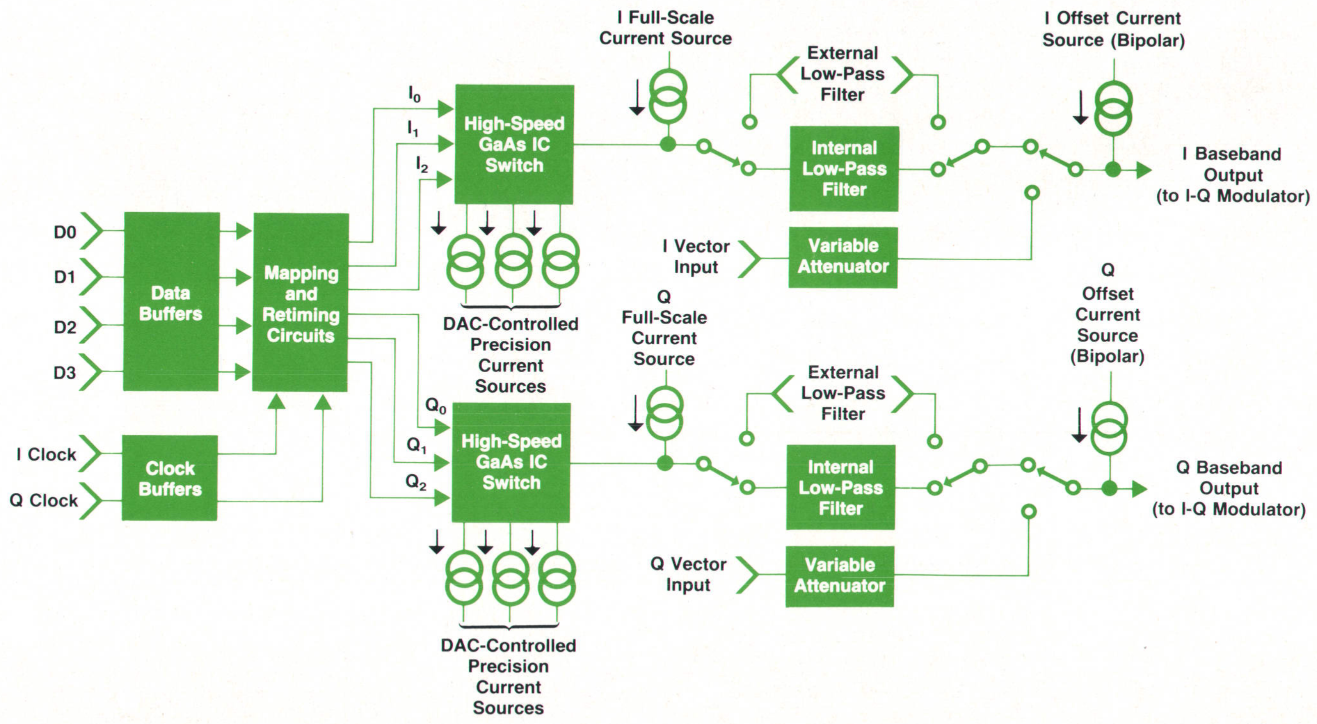

I Full-Scale Current Source
External Low-Pass Filter
I Offset Current Source (Bipolar)
I_0
I_1
I_2
High-Speed GaAs IC Switch
Internal Low-Pass Filter
I Baseband Output (to I-Q Modulator)
D0
D1
D2
D3
Data Buffers
Mapping and Retiming Circuits
I Vector Input
Variable Attenuator
DAC-Controlled Precision Current Sources
Q Full-Scale Current Source
Q Offset Current Source (Bipolar)
External Low-Pass Filter
Q_0
Q_1
Q_2
I Clock
Q Clock
Clock Buffers
High-Speed GaAs IC Switch
Internal Low-Pass Filter
Q Baseband Output (to I-Q Modulator)
Q Vector Input
Variable Attenuator
DAC-Controlled Precision Current Sources

Fig. 3. *Detailed baseband block diagram.*

A GaAs IC Current Switch

One of the most important requirements of the digital baseband system in the HP 8780A is to provide accurate levels to drive the vector modulator at high data rates. Early in the project we decided that we should use precision digital-to-analog converters (DACs) to provide the level accuracy, and a fast current-switching device to allow rapid transition from level to level by switching the DAC-generated currents on and off. For instance, to generate BPSK (binary phase-shift keying) signals, the I-channel baseband outputs two levels, corresponding to whether a current is switched on or off into the 25-ohm load. An additional programmable full-scale current source provides a bias so that the two levels are symmetrical about ground (Fig. 1). By connecting more than one switch in parallel, it is possible to generate more than two levels, and therefore generate more complex modulation. For example, 16QAM requires four levels per channel, and 64QAM requires eight.

When we started looking at current-switching devices, the GaAs IC process at HP's Microwave Technology Center was just about ready to go into production. There are many advantages in using the GaAs ICs. More than one current switch can fit on a chip so that interconnection parasitics are low, the devices have intrinsically fast switching times, and the current transfer ratio is unity. A driver had to be designed because the GaAs FET needs about a 2V swing at the gate to switch it. We experimented with several driver designs and finally chose a Schmitt trigger to drive the differential current switch. At first there was a big risk in going with GaAs ICs because the technology was unproven and we didn't really know whether GaAs ICs were producible or not. But the advantages outweighed the risks, and we made the decision to use the GaAs IC process.

Fig. 2 shows a block diagram of the IC. There are three Schmitt triggers driving three differential current switches. The Schmitt triggers require about a 0.5V swing at the input, and they provide a 3V swing to drive the differential current switches. The range of currents for each switch is zero to 20 mA. Fig. 3 shows the packaged GaAs IC current switch.

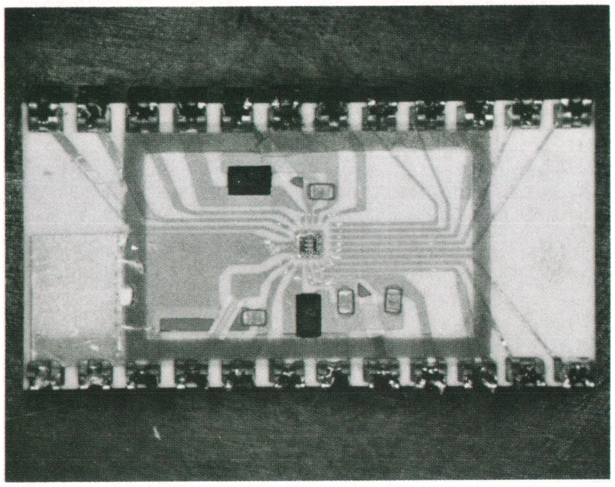


Fig. 3. *The GaAs current switching IC is bonded to a thick-film circuit and packaged as shown.*

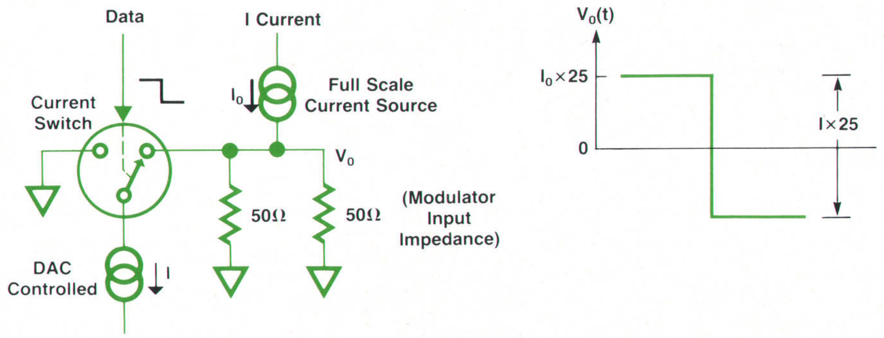


Fig. 1. *Programmable current source.*

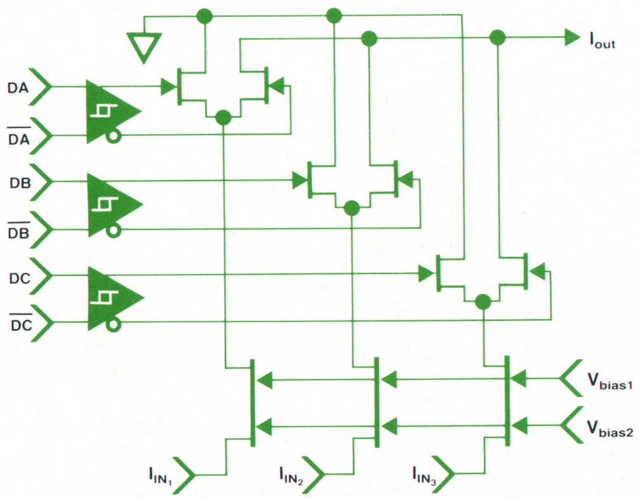


Fig. 2. *Block diagram of GaAs current switching IC.*

Describing Signals in the I-Q Domain

To make full use of the features of the HP 8780A Vector Signal Generator, one should be familiar with the description of modulated signals in the I-Q domain. Mathematically one can describe any signal modulated at a carrier frequency of ω_c radians/s by:

$$s(t) = A(t)\sin(\omega_c t + \phi(t))$$

where A(t) and ϕ(t) denote the amplitude and phase variations, respectively, of the signal s(t). The instantaneous frequency of the signal in radians/s is:

$$\omega(t) = \omega_c + d\phi(t)/dt$$

Another way of describing the same signal is in terms of the amplitude modulations of the in-phase component sin ω_ct and the quadrature component cos ω_ct. That is:

$$s(t) = i(t)\sin(\omega_c t) + q(t)\cos(\omega_c t)$$

The two descriptions are the same, of course, and with the notations we are using; $i(t) = A(t)\cos\phi(t)$, $q(t) = A(t)\sin\phi(t)$ and

$$A(t) = [i^2(t) + q^2(t)]^{1/2}$$

$$\phi(t) = \tan^{-1}[q(t)/i(t)]$$

The relationships between (i,q) and (A,ϕ) can be most conveniently seen by plotting s(t) in the I-Q plane, with i and q being the horizontal and vertical axes, respectively (see Fig. 1).

When viewing signals in the I-Q plane, one should keep in mind that one is looking at the signal's amplitude and phase relative to the carrier, and that the whole I-Q plane is actually spinning at the carrier frequency. The signals i(t) and q(t) are called baseband signals because they modulate the carrier and its quadrature. In the I-Q plane, a CW signal is simply a stationary

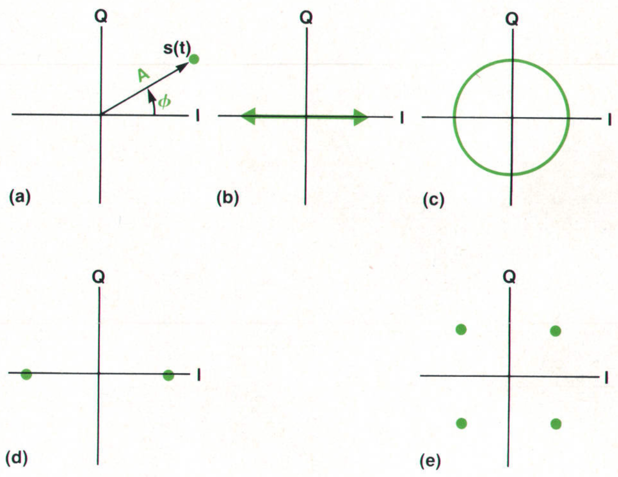


Fig. 1. *Examples of signals plotted in I-Q plane. (a) CW signal of amplitude A and phase ϕ. (b) AM signal tracing along I axis. (c) FM signal. (d) Ideal BPSK signal. (e) Ideal QPSK signal.*

(continued from page 40)

In instruments equipped with the 64QAM option, the user can select the serial clock mode in conjunction with 64QAM. This allows the user to generate 64QAM signals with just one data input line and a clock input. An internal serial-to-parallel converter converts the single data line into six parallel data lines to drive the baseband mapping circuits. In this mode the user can also provide a framing bit input so that the serial-to-parallel converter can frame six bits into a word, or symbol, at the proper times.

When digital modulation is selected, an internal baseband filter is switched in (one for each channel) to limit the spectrum of the baseband signals. This is necessary because the baseband signals have very fast switching times (less than 1 ns) and therefore occupy very wide bandwidths. Which one of the four filters gets switched in depends on the carrier frequency chosen. At low carrier frequencies, the filter selected will have a narrower bandwidth so that aliasing is not a problem at the instrument output. The users can also provide their own baseband filters so that they can shape the spectrum to exactly what they need.

Vector Modulation

In vector modulation, the user provides analog I and Q baseband signals to the vector inputs and the instrument acts like an ideal vector modulator at the selected carrier frequency. Internal calibration routines ensure that the modulator has low offsets, that the two channels are balanced (equal gain), and that quadrature is set correctly. In this mode the modulation bandwidth specified is dc to 350 MHz, resulting in a double-sided RF bandwidth of up to 700 MHz. This permits very wideband complex modulation and makes it possible to simulate signals like radar chirps, MSK (minimum-shift keying) signals, etc.

When used with two HP 8770A Arbitrary Waveform Synthesizers, the HP 8780A can provide exceptional signal simulation capabilities. Any signal within the specified modulation bandwidth can be simulated as long as the signal can be broken down into its I and Q components. Or, if one wishes to duplicate an RF signal, one needs only to vector demodulate it, capture the I and Q components with a waveform recorder, modify the data, if necessary, on a computer, and send the data to the HP 8770A Synthesizers so that they can vector modulate the HP 8780A.

Circuitry

The baseband circuits consist of three 13-by-5-inch printed circuit boards and contain about 1000 electrical parts. A more detailed block diagram of the baseband system is shown in Fig. 3 on page 40.

When digital modulation is selected, the user connects digital data lines to the front panel of the instrument, and point with constant I and Q components. A conventional AM signal is one that has its trajectory along a fixed line through the origin (the I-axis for example). An FM signal will have a circle centered at the origin as its trajectory, with the instantaneous frequency deviation given by the rate of change of its phase. An ideal BPSK (binary phase-shift keying) signal with no filtering appears as two points on the I-axis, equidistant from the origin. An ideal QPSK (quadriphase-shift keying) signal appears as four points that are the corners of a square centered at the origin. The display of digital modulation states on the I-Q plane is often called a constellation diagram.

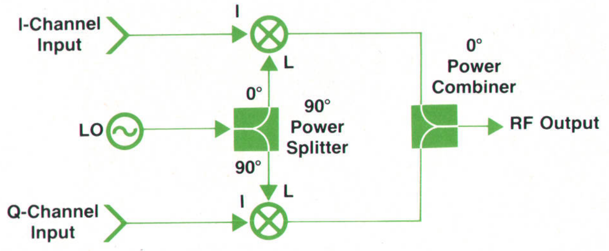


Fig. 2. *Simplified block diagram of a vector modulator.*

While going from an amplitude-phase description to an I-Q plane description is mathematically trivial, there are some powerful insights about the signal that one can gather:

1. The I-Q description gives a way of synthesizing the signal. Instead of phase modulation, which is nonlinear and difficult to do well, one can simply modulate the amplitude of the carrier and its quadrature in a linear manner. Mixers with wide modulation bandwidths and good linearity are readily available. This method of modulation is called vector modulation, and the block diagram of a vector modulator (or I-Q modulator) is shown in Fig. 2. To synthesize a complex modulated signal, one needs only to generate the baseband I and Q components of the signal. For BPSK and QPSK signals, the baseband signals are almost trivial, and even higher-order digital formats usually have fairly simple baseband components. One key advantage of vector modulators is that the same modulator can be used to generate a variety of modulations from digital formats to RF pulses, or even radar chirps, for instance.
2. Demodulating the signal is also straightforward. Using a vector demodulator, it is easy, at least in principle, to recover the baseband signals. A block diagram of a vector demodulator (or I-Q demodulator) is shown in Fig. 3.
3. Looking at a signal in the I-Q plane often gives good insights about the signal. Effects like crosstalk, data skew, compression, and AM-to-PM distortion, which are hard to visualize otherwise, are easy to see (Fig. 4).

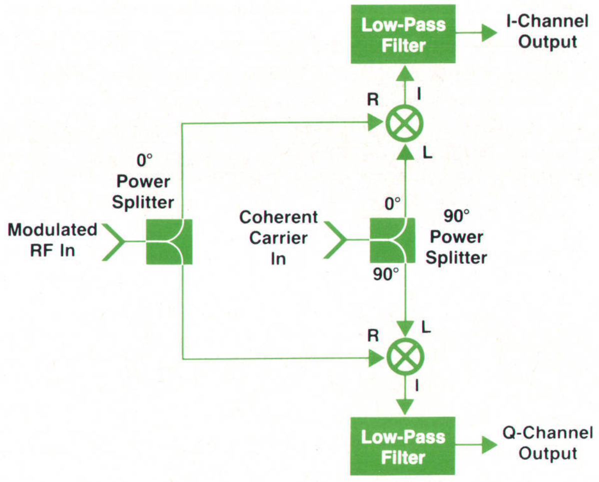


Fig. 3. *Simplified block diagram of a vector demodulator.*

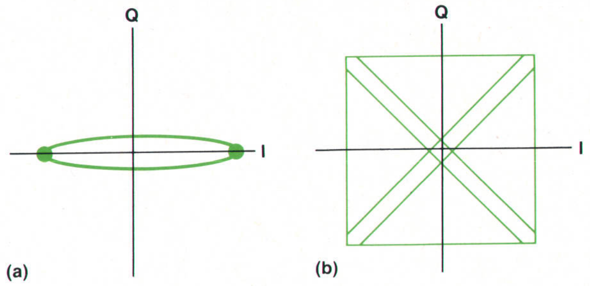


Fig. 4. *I-Q plane plot showing effects of (a) crosstalk and (b) data skew and very little filtering.*

these signals are routed to the buffer board. The function of the data buffers is to convert each data signal into an ECL signal to drive the mapping circuitry. The user must enter the data threshold using the front-panel keys: **GND**, **ECL**, or **VAR**iable (user enters threshold voltage). If an ECL data level is selected, the termination voltage is automatically set to −2V. The input impedance for each line is 50 ohms. If a clocked mode is selected, the clock buffer converts the incoming clock signal into an ECL signal to drive a set of retiming flip-flops. The clock buffers are a little different from the data buffers in that the user can select **AUTO** level for the clock signals. In that case an internal threshold detector finds the correct switching threshold for the clock signal, independent of clock duty cycles.

The main function of the mapping circuitry is to map the input data lines into the proper signals to drive the high-speed GaAs IC switches. The four incoming data lines are converted into six lines, three for each channel, to drive the GaAs switches. The exact mapping depends on the modulation format chosen, and the mapping logic is done by 100K ECL gates and relays. Electrical delays of the data and clock signals are carefully controlled so as not to introduce data skew between them.

There is one GaAs IC switch for each channel. The switches control the baseband output voltage to drive the vector modulator in CW and digital modulation modes. The IC contains three Schmitt triggers driving three current switches. The outputs of the current switches are connected in parallel to a 25-ohm load resistance. The amount of current being switched in each current switch is accurately generated by 12-bit digital-to-analog converters (DACs). A full-scale current source is connected to the output of the IC to provide bias to the output. The three current switches in each channel allow eight possible output baseband levels. The transition from level to level is triggered by input data transitions. The baseband output rise and fall times are largely independent of the data transitions because of the Schmitt triggers' regenerative action. In synchronous modes the data lines are also sharpened by the retiming flip-flops. The resulting transition time at the out-

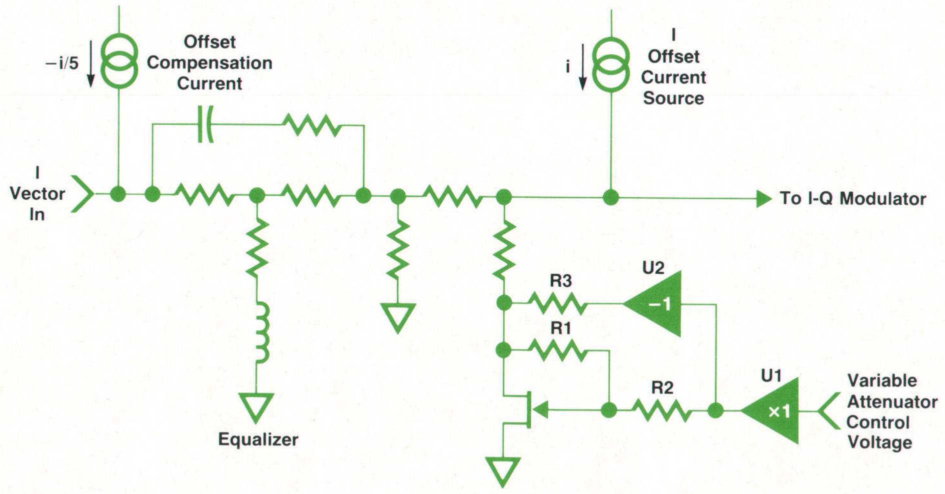


Fig. 4. *FET variable attenuator.*

put of the GaAs IC is around 500 ps. To have specified performance up to 150-MHz clock rates, the mapping circuits are carefully designed and laid out with the minimum number of components in the signal paths, and with line lengths carefully matched. The specified data skew and data asymmetry are both 1 ns. The box on page 41 describes the GaAs IC in more detail.

The two baseband voltages from the GaAs ICs are routed to the filter board, where they are switched to one of a set of low-pass filters. These filters are of the LC type, and there is a passive equalizer to flatten the passband response. This is necessary because of the skin losses in cables and relays and the finite Qs of the inductors. There is a provision for users to connect their own baseband filters—this is important for those users who want to simulate other modulators.

During baseband calibration, the internal microcomputer adjusts the vector modulator for minimum quadrature error. It calculates the DAC settings for the precision current sources to give the correct output levels for any modulation format. It also computes the settings of the offset DACs, one DAC for each channel, so that the inherent offsets of the vector modulator are canceled. The result is a static modulation accuracy specification of between one and two percent, depending on format.

In vector modulation mode, the user provides analog signals to the vector I and Q inputs. These signals are sent to the filter board, and each passes through an equalizer and variable attenuator. During calibration, the I and Q channel conversion losses are matched by adjusting these variable attenuators until the same magnitude signal applied to either the I or the Q vector input gives the same RF output level. Carrier leakage is canceled by switching in zero voltage to the vector inputs and adjusting the offset current sources. The variable attenuator (Fig. 4) consists mainly of a FET, and the circuit is designed for minimum distortion and maximum bandwidth. It is typically flat to within ±0.3 dB up to 100 MHz, and has a −3-dB frequency of more than 400 MHz. The two resistors, R1 and R2, keep distortion caused by the FET low. U2 and R3 remove the control voltage feedthrough to the output. The I offset current source nulls out the I component of carrier leakage. The offset compensation current tracks the offset current source to provide a zero termination voltage at the vector input, regardless of offset current magnitude. The nominal attenuation is 14 dB, or a factor of 5.

The switching between digital and vector modulations is done by RF relays. Before the baseband outputs leave the filter board, they are filtered by a pair of 800-MHz low-pass filters to remove any high-frequency crosstalk between the channels. The filter board's two outputs then drive the mixer in the vector modulator directly.

Acknowledgments

Kyle Karnosh and Dave Gildea contributed to the baseband block diagram, and Kyle also did investigations on the buffers and mapping circuitry. Lewis Dove of the Colorado Springs Technology Center did the packaging for the GaAs IC. I also thank him for the very smooth introduction of the hybrid package. Thanks are also due the GaAs IC group in our Microwave Technology Division, especially Rory van Tuyl, Val Peterson, and Andy Teetzel for the many suggestions and ideas on the design of the switch IC, and Don D'Avanzo and Don Estreich on the many improvements made to the RF IC process. Tim Chan from our reliability and materials department was instrumental in putting the IC through rigorous qualification testing. V.J. Bonnard provided excellent production support for the digital and vector sections of the HP 8780A. John Hasen wrote all the baseband testing software and saw the instrument through Class B environmental testing. Finally, thanks to Nick Lyons and Paul Stafford for writing the baseband calibration routines that make it possible to achieve this very high level of performance.

A Wideband FM Subsystem for a Low-Noise Synthesizer Module

by Eric D. McHenry

THE FM SUBSYSTEM of the HP 8780A Vector Signal Generator is responsible for supplying the 1-GHz fixed frequency reference for the generator during CW or vector modulation and a frequency-modulated 1-GHz signal for FM modes. The challenges in implementing this design were making an oscillator and related phase-locked loops that have state-of-the-art phase noise for CW applications, ultralinear wideband FM for ac-coupled FM applications, and low-noise narrowband dc-coupled FM.

Although the HP 8780A is called a vector signal generator, its phase noise performance needs to be very good not only for vector specifications (phase noise produces arcs on vector diagrams, see Fig. 1) but also for traditional Doppler radar applications where phase noise on a local oscillator reduces the sensitivity of the radar (see Fig. 2). The phase noise performance is determined equally by the low-noise local oscillator (see article on page 34) and the FM subsystem.

The FM subsystem can be functionally divided into two sections: baseband (dc-to-12-MHz bandwidth) and RF (producing a modulated 1-GHz signal). The modulated 1-GHz signal is multiplied up to 8 GHz by the IF multipliers before being mixed down to the HP 8780A's output frequency band. This multiplication directly affects the FM gain of the RF section, since all deviations are eight times larger at the HP 8780A output. Therefore, while the HP 8780A has an FM sensitivity of 50 MHz/V at the output, the modulation section has a sensitivity of only 6.25 MHz/V.

Although the multiplication makes the design of a linear FM oscillator easier (the linear range is eight times smaller), it creates many other problems, the most difficult being related to phase noise and microphonics. Much effort was put into the design to keep phase noise, microphonics, and spurious signals low (all of these get multiplied by a factor of 8, or 18 dB).

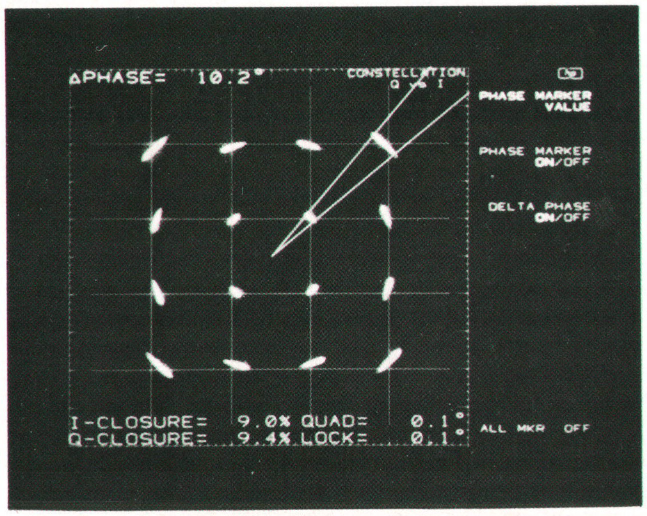

```
ΔPHASE=  10.2°                 CONSTELLATION
                                   Q vs I

I-CLOSURE=   9.0% QUAD=        0.1°
Q-CLOSURE=   9.4% LOCK=        0.1°
```

Fig. 1. *Severe phase noise on a 16QAM signal.*

FM Baseband Circuitry

The function of the FM baseband circuitry is to vary the amplitude of the baseband signal before it is applied to the RF modulation circuits. This is necessary because the sensitivities of the 1-GHz FM voltage-controlled oscillator (VCO) and the 100-MHz dc FM voltage-controlled crystal oscillator (VCXO) are fixed. To get the range of deviations required at the HP 8780A's output, the amplitude of the modulating signal must vary over a 60-dB range.

There are four blocks in the baseband chain (see Fig. 3):

- Overpower protect. The overpower protect circuit immediately latches open the FM input relay upon detection of a voltage exceeding the maximum input level and signals the microprocessor that an overpower condition exists at the FM input. The instrument's **MSG** (message) key light is lit and the overpower protect relay remains open until the user presses the **MSG** key or executes a similar HP-IB (IEEE 488/IEC 625) command.
- Calibration signals. These ±0.3V and ground signals are switched into the baseband input during calibration. They provide stimulus signals that allow measurement of baseband offsets, gain, and attenuation coefficients. Firmware calibration allows the use of simple voltage dividers to generate the calibration voltages. These voltages are measured during the calibration process and their measured values are used for the calculations.
- Variable-gain amplifier. This 11-dB amplifier with a bandwidth of dc to 12 MHz is capable of both inverting and noninverting amplification. The amplifier gain varies over a 3-dB range to provide continuous sensitivity coverage between step attenuator ranges. The design

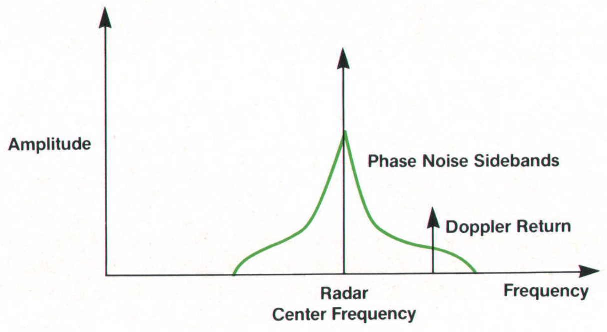


Fig. 2. *Phase noise limiting radar sensitivity.*

is implemented using a FET variable attenuator followed by a fixed-gain discrete amplifier with switchable polarity. During the FM calibration routine, the variable-gain amplifier is measured for dc offset, gain in both polarities, and gain versus control voltage.

- Step attenuators. Five fixed-value step attenuators are organized in a binary progression (2, 4, 8, 16, and 32 dB) and are switched to provide the coarse sensitivity range. The minimum resolution is 2 dB and the maximum attenuation is 66 dB. The design uses standard 1% resistors and a calibration algorithm that measures the actual attenuation of each section for higher accuracy.

FM RF Circuitry

The FM circuitry contains three phase-locked loops for the four types of frequency modulation done by the HP 8780A: CW, ac-coupled FM, dc-coupled FM, and wideband dc-coupled FM.

CW Mode. In CW (no FM) mode (Fig. 4a), the 1-GHz FM oscillator is locked to the 100-MHz auxiliary output of the low-noise local oscillator (LO) in a 500-kHz phase-locked loop using a sampling phase detector. This provides good phase noise and allows cancellation of close-in noise from the LO. Low-noise design was necessary for these circuits, including heavy filtering of the power supply lines to reject both power-line related and active device noise.

DC-Coupled FM Mode. In this mode (Fig. 4b), the 1-GHz FM oscillator is locked to an internal 100-MHz VCXO in a 500-kHz phase-locked loop. The baseband FM is applied to the 100-MHz VCXO. Since the phase-locked loop bandwidth is greater than the maximum modulation rate in dc-coupled FM, the baseband FM is transferred onto the 1-GHz VCO.

The phase noise in this mode is similar to that in CW, with the except of offsets less than 300 Hz. In this case, the cancellation effect present in CW mode is not available because the two 100-MHz oscillators that dominate the phase noise spectrum in this region are not phase coherent.

Some applications using dc-coupled FM mode include phase locking the HP 8780A to a external source for phase noise measurements, making low-noise narrowband FM tests, and simulating the Doppler shift on vector modulation transmitted from a moving satellite.

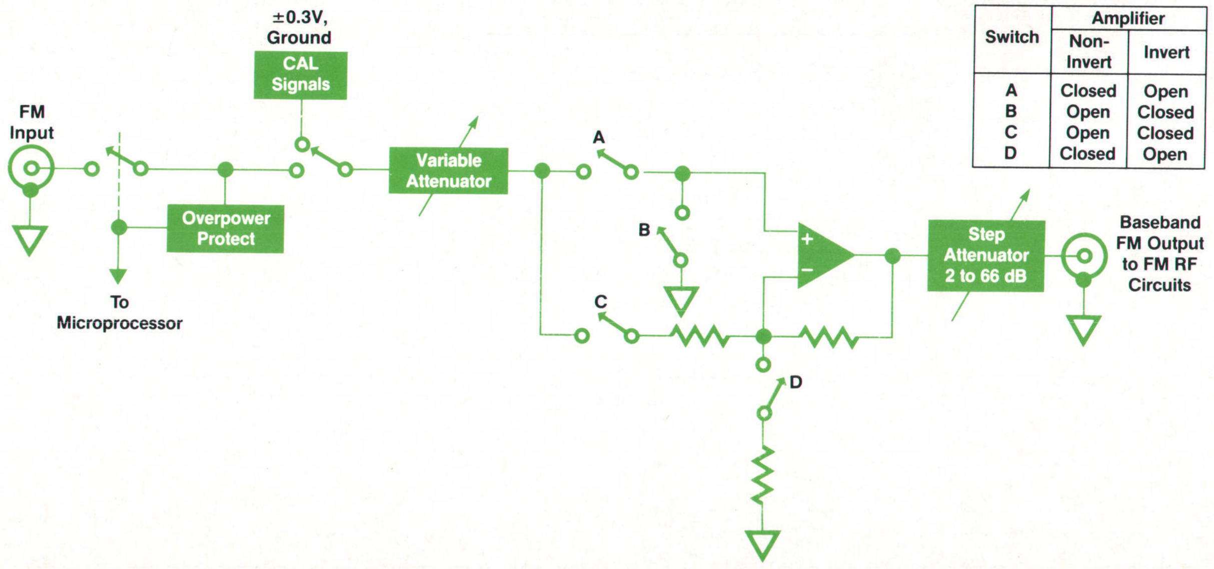
AC-Coupled FM Mode. This mode (Fig. 4c) is designed to cover the needs of satellite TV and chirped radar users. It provides wideband, ultralinear, high-index frequency modulation at standard video (1V peak to peak) levels. To provide high-index, phase-locked FM, it is necessary to restrict all phase deviations at the phase detector to within the phase detector's linear range. This is done by using an ECL divider in the FM path to divide down the phase deviations.

There are three conventional ways to modulate an oscillator that needs to be phase locked. They are modulating inside the loop bandwidth (as in dc-coupled FM), modulating both inside and outside the loop bandwidth, and modulating completely outside the loop bandwidth. The first method has the restriction that the maximum modulation rate must be less than the loop bandwidth (this is okay for narrowband dc-coupled FM). The second method allows higher modulation rates, but creates phase nonlinearity at the loop bandwidth crossover point.

The ac-coupled FM loop uses the third method of modulating completely outside of the phase-locked loop bandwidth. This technique allows high modulation rates with very linear phase. To meet a particular TV video specification (field time distortion), it was necessary to select a loop bandwidth of 20 Hz.

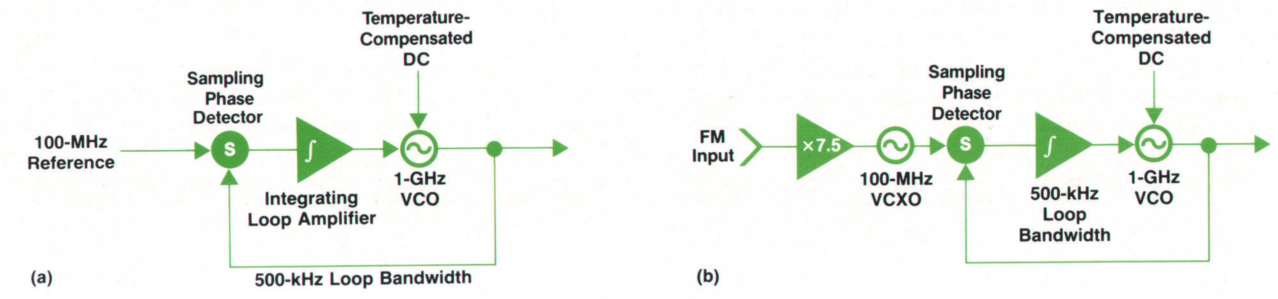
Wideband DC-Coupled FM. In this mode, one side of the 1-GHz FM VCO varactors is set to a temperature-compensated dc voltage while the other side of the varactors is directly fed by the baseband FM signal. This special function allows dc-to-12-MHz rates and deviations greater than 31.25 MHz peak to peak at the VCO, which corresponds to 250 MHz peak to peak at the HP 8780A output.

Wideband dc-coupled FM mode is useful for sweeping



Switch	Amplifier	
	Non-Invert	Invert
A	Closed	Open
B	Open	Closed
C	Open	Closed
D	Closed	Open

Fig. 3. *FM baseband circuits.*



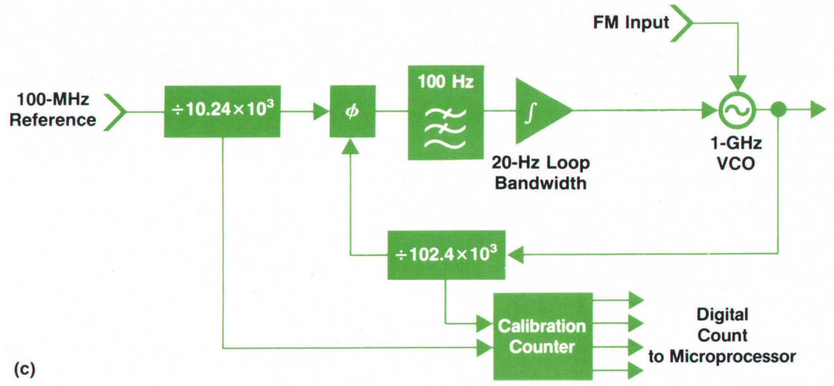


Fig. 4. *Block diagrams for three FM modes: (a) CW, (b) dc-coupled FM, and (c) ac-coupled FM.*

the HP 8780A or locking the HP 8780A to noisy oscillators.

Calibration Counter

The calibration counter (Fig. 5) for the ac-coupled FM sensitivity measurements works by counting the instantaneous difference between the 100-MHz reference and the 1-GHz VCO. Both signals are divided by ECL circuits before being applied to a D flip-flop. The flip-flop operates as a sampler and produces a digital signal at a frequency equal to the difference between the two inputs. This frequency is converted into an 8-bit byte by a gated counter. Associated one-shot circuits generate a pulsed reset signal for the counter and a count valid signal for the microprocessor. The microprocessor reads the count byte during calibration and divides the count by the applied voltage to determine the sensitivity of the 1-GHz VCO.

Acknowledgments

The author would like to thank Dan Bower for his early work on the FM circuits, Doug Anderson for his design of the FM baseband circuits, Kaaren Marquez for her invaluable work at the end of the project to get the FM circuits

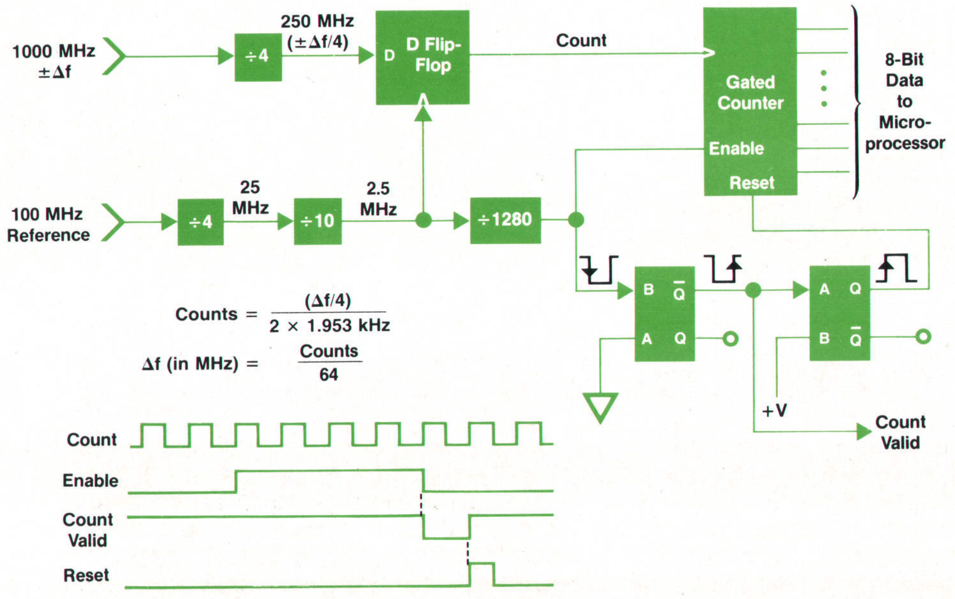


Fig. 5. *Ac-coupled FM calibration counter.*

through environmental testing and into production, and Mike Loushine and the production engineering staff for smoothing the transition of the FM circuits from the lab into production.

Vector Modulator, Output Amplifier, and Multiplier Chain Assemblies for a Vector Signal Generator

by Wayne M. Kelly, Mark J. Woodward, Eric B. Rodal, Pedro A. Szente, and James D. McVey

OBTAINING an accurate modulated signal from a synthesized low-frequency source requires several carefully designed assemblies to modulate and amplify the signal and to multiply its frequency to the range desired. The diverse types of modulation and wideband performance required of the HP 8780A Vector Signal Generator made the design of such assemblies for it particularly challenging.

Vector Modulator

The vector modulator in the HP 8780A consists of three microcircuits interconnected by isolators. The block diagram is shown in Fig. 1. The modulator generates very accurate I and Q signals when driven by the digital baseband circuits and the isolators used between the individual microcircuit assemblies ensure extremely low crosstalk between the I and Q channels. The three microcircuits and their functions are described below.

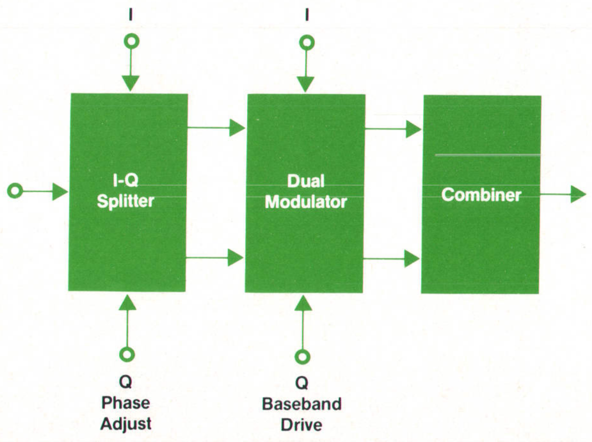


Fig. 1. *Vector modulator block diagram.*

I-Q Splitter

The I-Q splitter takes the HP 8780A's 8-GHz IF signal and generates two phase quadrature signals of equal amplitude. These high-level (12 dBm) signals drive the LO ports of the I and Q modulators. Included in the hermetic package are voltage-adjustable phase-shift networks in each path. Approximately 25 degrees of phase adjustment range is obtained for a tuning voltage range of −4 to −12V. During calibration these phase shifters permit exact quadrature to be obtained even though the individual isolators or modulators are not perfectly phase matched. The quadrature hybrid is a four-wire interdigitated line type built on a 0.025-inch-thick sapphire substrate. The individual phase shifters are built using the same type of hybrid construction, but with GaAs tuning varactors connected to ground on the coupled and direct output ports. This way, the resultant signal from the isolated port of the hybrid circuit is simply equal to the reflection coefficient (plus a fixed phase offset) of the varactor. As the dc voltage on the varactor is changed, a corresponding change in capacitance occurs. This results in a change in the angle of the reflection coefficient ρ. Since the magnitude of ρ is unity, the result is a phase change in the output signal.

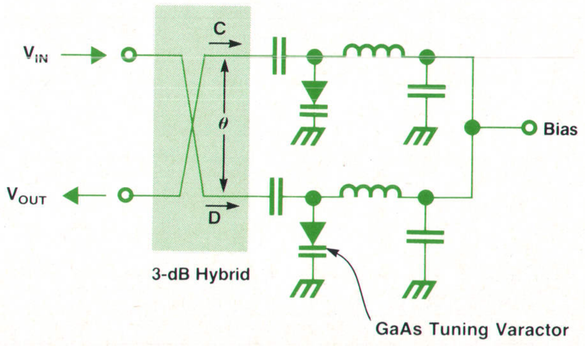


Fig. 2. *Schematic of phase shifter.*

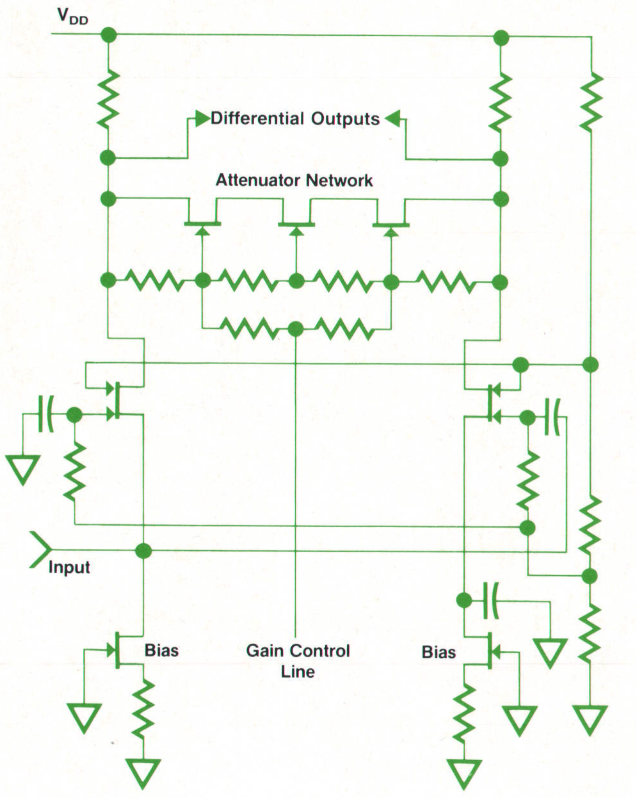


Fig. 3. *Preamplifier section of the HP 8780A output amplifier.*

Fig. 2 shows a schematic of the phase-shift circuit. $V_{out} = 2CD\rho V_{in}$ where C is the coupled voltage and D is the direct voltage. θ is the electrical length of the hybrid and is 90 degrees at 8 GHz. C and D are functions of θ according to:

$$C = \frac{jk\sin\theta}{\sqrt{1-k^2}\cos\theta + j\sin\theta}$$

$$D = \frac{\sqrt{1-k^2}}{\sqrt{1-k^2}\cos\theta + j\sin\theta}$$

where k is the voltage coupling coefficient and is equal to 0.707 for a −3 dB coupler.

Dual Modulator

The dual-modulator microcircuit provides wideband (0 to 350 MHz) modulation to the two quadrature signals from the splitter. Also included in the package are low-pass filters on the modulator's IF ports. These filters reject any 8.0-GHz IF signal at the baseband input ports and provide a 50Ω matched termination to the modulator I port (at 8 GHz). This matching prevents abnormal modulator behavior over the full modulation bandwidth at 8.0 GHz. At a carrier frequency of 8 GHz the 350-MHz modulation rate represents less than a 10% bandwidth. This makes it possible to produce very linear modulation with low crosstalk.

The modulator uses double balanced mixers that were carefully chosen to have the following attributes:

- A wideband dc-coupled I port with flat frequency response
- Excellent (40 dB) L-to-R isolation and low dc offset
- Excellent linearity for IF drive levels up to ±0.1V
- Low VSWR (i.e., 50Ω) on the mixer I port from dc to 350 MHz
- Low AM to PM conversion.

The filter circuit was designed using a standard lumped-element low-pass filter prototype.[1] The series inductors are realized by equating the equivalent inductance of a high-impedance (120Ω) transmission line at cutoff to the lumped-element values. In a similar manner, the capacitors are realized by equating the capacitance of low-impedance (18Ω) transmission lines to the lumped-element values. To provide the 50-ohm matched termination to the modulator's I port at 8 GHz, a resistively loaded open-circuit shunt stub is connected at the appropriate distance from the end element of the filter. This point is where the impedance locus of the filter is an open circuit.

Two double balanced mixer modules and the two filters are assembled in a single machined aluminum housing.

Combiner

The combiner microcircuit takes the two modulated signals and adds them together to produce a single vector. Low-pass filters in each path prevent harmonics (16, 24, 32 GHz, etc.) generated by the modulators from appearing at the output port or coupling between modulators. If harmonics were permitted to enter the output mixer they would appear as spurs in the output spectrum of the instrument. Harmonics coupling between modulators can cause crosstalk between the I and Q signals.

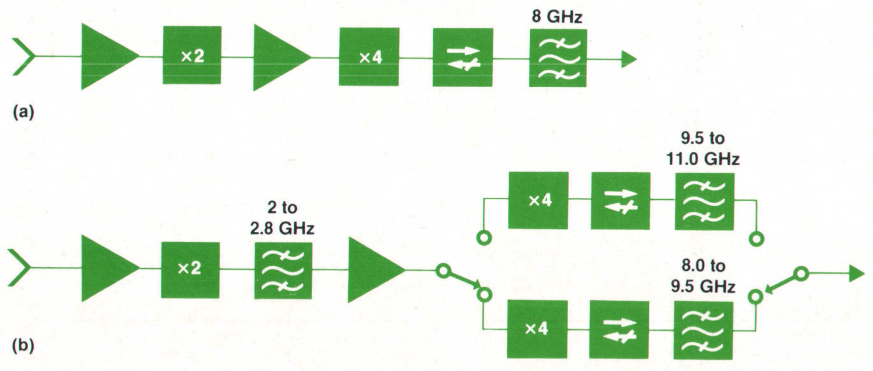


Fig. 4. *IF (a) and LO (b) multiplier chain block diagrams.*

Baseband Calibration

The calibration method for the baseband section of the HP 8780A must calibrate the following variables, using the RF detector for both amplitude and quadrature information:

- Digital-to-analog converters (DACs) for precision current sources (drive DACs and burst DACs)
- DACs for I and Q offset current sources (offset DACs)
- Vector input attenuators
- Quadrature DAC
- Baseband filter factors.

Calibration of the amplitude variables (the drive DACs), vector attenuators, and filter factors is done in a straightforward manner using binary search methods. Changing the attenuators changes the offset null settings, so an iterative approach is used until both parameters are set properly.

The more sensitive offset and burst null DACs require another approach. Since during null calibration the detector operates at a low signal level, a bucket search method is used (Fig. 1). To find the null setting for the DAC, the range is divided into ten discrete settings. Measurements are taken at each setting and the lowest and next lowest settings are used for the search. The lowest and next lowest settings are guaranteed to be on opposite sides of the true null. The DAC is set to search on the side of the lowest setting for another reading equal to the next lowest setting (indicated by the goal in Fig. 1). This gives two settings equally spaced from the true null, which then lies exactly halfway between the goal and next lowest settings.

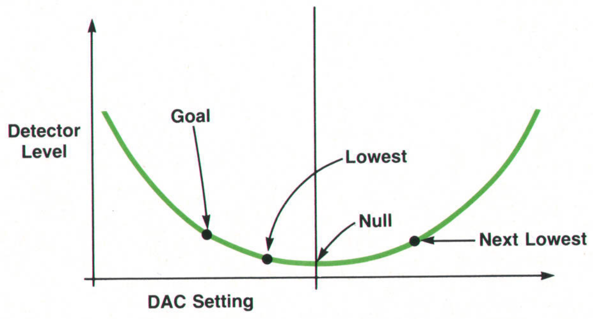


Fig. 1. *Bucket search method for calibrating offset and burst null DACs.*

The quadrature adjustment requires another indirect method of measurement, because there is no way of measuring quadrature directly. The quadrature can be assumed to be 90 degrees if and only if the sum of the first and third quadrant QPSK amplitudes is equal to the sum of the second and fourth quadrant amplitudes. The difference between the sums must change sign as the quadrature changes from less than 90 degrees to more than 90 degrees. Thus, a test for adequate range can be made before trying to calibrate. A binary search is then made on the quadrature DAC setting to minimize the differences between the sums of the states.

The result of the calibration is a table of DAC settings, which are then multiplied by the appropriate factors to create the currents applied to the I-Q modulator as required by the different modulation formats.

Paul S. Stafford
Development Engineer
Stanford Park Division

In-phase (zero degree) signal addition is done with a resistively loaded reactive power combiner. This type of combiner provides minimum loss (3 dB), high isolation, and excellent input/output VSWR. These attributes are essential for flat amplitude/phase response and low crosstalk.

The circuit was very carefully designed to yield this flat response. To ensure excellent repeatability and uniformity, the circuit is photoetched on a 0.01-inch-thick board. The etched circuit is then soldered into a gold-plated aluminum housing. This provides a rugged, environmentally sound microcircuit.

Output Mixer

The output mixer takes the 8-GHz modulated signal from the vector modulator and down-converts it to the 10-to-3000-MHz range. The local oscillator covers the frequency range of 8.01 to 11.0 GHz.

The output mixer uses a double balanced mixer that was carefully chosen to have a wideband IF response (10 to 3000 MHz), excellent amplitude and phase flatness, and low distortion. A diplexer circuit on the mixer's I port provides three important functions:

1. Rejection of signals above 5 GHz to the output port
2. A 50-ohm termination to the mixer I port at 8 GHz
3. Reactive matching at 3 GHz to improve conversion loss flatness and I-port match.

The diplexer was designed using a standard lumped element, low-pass/high-pass design method.[1] The low-pass filter (n=12) is realized by using high-impedance and low-impedance transmission lines to simulate the lumped inductors and capacitors. The high-pass filter uses lumped capacitors for the series elements and short lengths of shorted high-impedance lines for the shunt elements. The high-pass filter (n=4) is terminated by a 50-ohm chip resistor.

A matching stub at the diplexer common junction is used to improve the I-port return loss from 8 to 14 dB at 3 GHz. A perfect match at 3 GHz is not practical since the mixer module's I-port impedance is a complex function of frequency. Attempting to improve the match at 3 GHz further would degrade it at 1 GHz.

The physical realization of the diplexer is similar to that in the dual-modulator microcircuit.

Output Amplifier

The specifications for the HP 8780A require its output

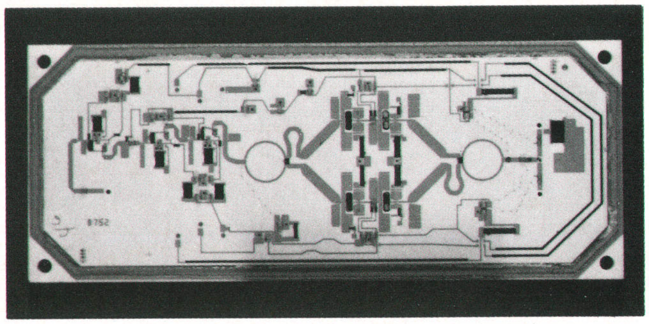


Fig. 5. *Photograph of high-power output amplifier thick-film circuit.*

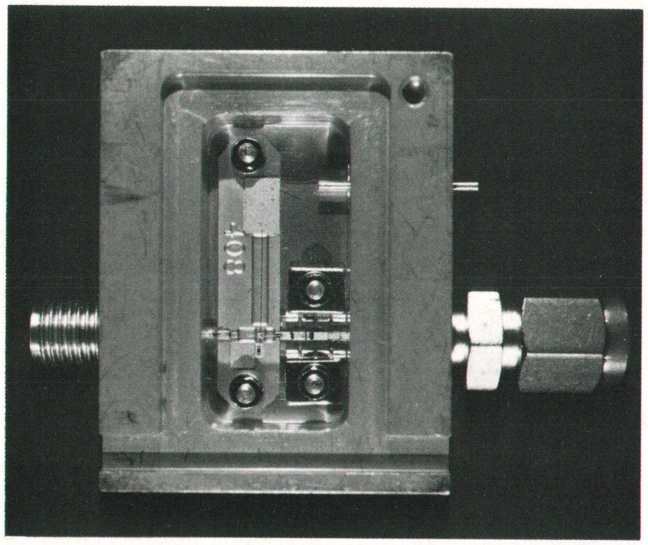


Fig. 6. *Photograph of multiplier module.*

amplifier to be wideband and ultralinear and to exhibit low noise with flat frequency response and good input and output matching. The amplifier linearity affects the vector accuracy of the instrument. Poor amplifier frequency response and matching can cause dynamic crosstalk or AM-to-PM conversion. The noise figure of the amplifier sets the instrument noise floor. A GaAs integrated circuit amplifier was chosen to meet these requirements. During the amplifier development, it was necessary to make many process and design refinements to achieve all the performance goals while maintaining good amplifier yields.

Typical amplifier performance parameters are 33 dB of gain with a gain adjustment range of 20 dB, 12-dBm output power, third-harmonic distortion of −41 dBc at 7-dBm output power level, 8-dB noise figure, and input and output return losses greater than 20 dB from 10 MHz to 3 GHz. Total power dissipation is about four watts on two one-millimeter-square chips. This requires good thermal design in the package to keep chip temperatures low for high reliability and performance.

The output amplifier is made up of a preamplifier and a power amplifier. The preamplifier consists of two gain stages, two buffer stages between the gain stages, and two attenuator networks. The first stage of the preamplifier converts the single-ended input to a differential signal using a common-gate and common-source stage (Fig. 3). The input match is set by adjusting the bias current in the common-gate section. After the input stage, the amplifier is differential to minimize sensitivity to power-supply noise and poor RF grounds. The preamplifier attenuator networks provide 20 dB of gain adjustment with 0.01-dB resolution. This acts as the instrument's output level vernier. The attenuators are realized using FETs operating in their linear region to provide a variable load for the differential gain stages (Fig. 3). The attenuation is controlled by changing the gate-to-source voltage on the attenuator FETs using a digital-to-analog converter (DAC).

The power amplifier uses two gain stages to provide approximately 10 dB of gain to boost the output level to at least 10 dBm at maximum preamplifier gain. The differential outputs allow one side to be used for the instrument output and the other for a diode detector circuit. The diode detector circuit is used to calibrate the instrument output level versus frequency. The detector is also used to calibrate the digital modulation formats. The effects of temperature on the detector are compensated in firmware using a temperature sensor mounted in the amplifier package.

Multiplier Chains

Fig. 4 shows the LO and IF multiplier chain block diagrams. The first sections of both multiplier chains are similar and consist of a driver amplifier, doubler, and high-power amplifier. The LO chain adds another bandpass filter. Both multiplier chains include frequency quadruplers, but each is of slightly different design to obtain the necessary performance. The IF multiplier operates at a fixed frequency output of 8 GHz while the LO multiplier operates at an output frequency that varies from 8 to 11 GHz. Two quadruplers are used in the LO multiplier chain, each covering approximately half the frequency range.

Driver Amplifier and Frequency Doubler

The buffer amplifier provides a minimum output of 12 dBm in the 1-to-1.4-GHz range and is similar to the one used in the low-noise synthesizer, but has higher bias current to provide higher output power. This amplifier design

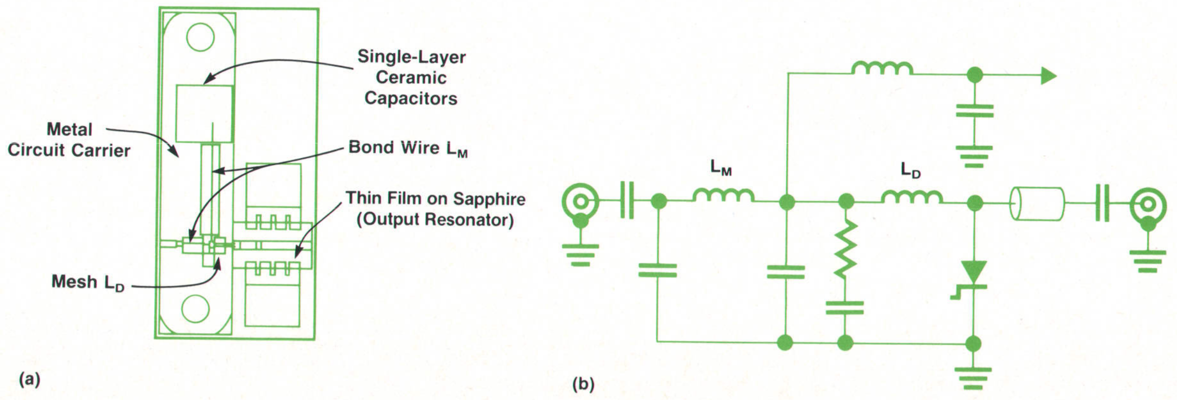

Single-Layer Ceramic Capacitors
Metal Circuit Carrier
Bond Wire L_M
Thin Film on Sapphire (Output Resonator)
Mesh L_D
(a)
L_M
L_D
(b)

Fig. 7. *(a) Multiplier layout. (b) Schematic.*

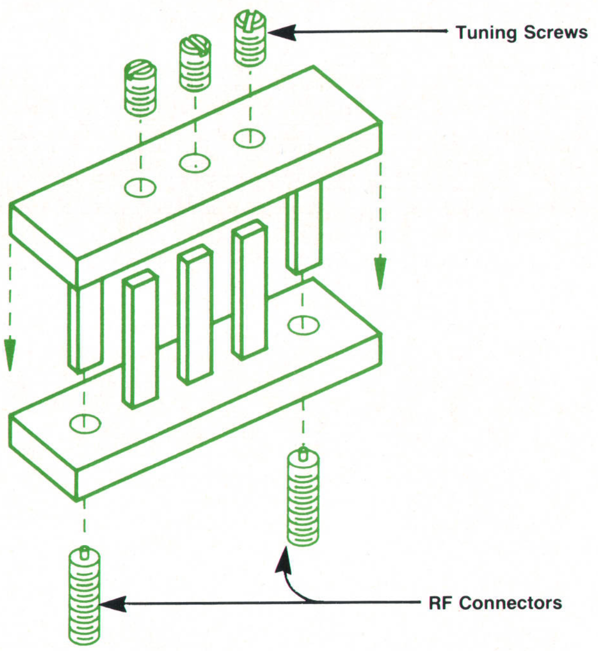


Fig. 8. *Exploded view of comb line filter.*

is discussed in more detail in the low-noise synthesizer design article on page 34.

The doubler consists of two coaxial baluns that are hand-soldered to the board and four hand-loaded axially packaged matched Schottky diodes arranged in a bridge configuration.

High-Power Amplifier

The high-power amplifier boosts the filtered outputs of the frequency doublers to a power level of one watt to drive the frequency quadruplers. This amplifier appears twice in the HP 8780A, once to amplify the 2-GHz signal, and again to amplify the 2-to-2.75-GHz signal. The high-power amplifier consists of three preamplifier stages, a 90-degree power splitter, four power stages (two in each leg of the power splitter), a 90-degree power combiner, and a pin diode switch. The 90-degree power splitter and combiner allow the use of two devices to get twice the power output and help provide a good output match. The pin diode switch is used to switch the power between the two frequency quadruplers in the 8-to-11-GHz channel and to provide the RF on/off features of the instrument in the 8-GHz channel.

The most distinctive feature of the amplifier is the ceramic package in which it is housed. The thick-film circuit (Fig. 5) is made on an alumina substrate with silver metallization on the back. Affixed to this substrate is a 0.06-inch-thick alumina backplate that is partially metallized, three RF connectors, four heat studs, and six feedthrough capacitors. After assembling the substrate to the backplate, a stamped metal lid is attached to complete the RF seal. All assembly is done with conductive epoxy.

Frequency Quadruplers

Step-recovery diode multiplier circuits are used to quadruple the frequency of high-level synthesized signals to the frequency bands of 8 GHz and 8 to 11 GHz, respectively. With one-watt input power, the nominal multiplier output levels of these signals are 18 and 16 dBm, respectively. (The information used to design the quadruplers can be found in *Hewlett-Packard Application Note 920*.) The features of the multipliers built for the HP 8780A are high power output (about 16 to 20 dBm), good efficiency (about −10 to −14 dB), very low phase noise, good signal purity, and absence of parametric oscillations.

The high power output is achieved by driving the step-recovery diode at a high power level and mounting it on a beryllium-copper heat sink (Fig. 6). The high efficiency of the quadruplers is obtained by using high-Q components in the input matching network. The reactive components are made of ceramic capacitors and gold bonding wires. Also, the diode is operated at optimum bias through a self-bias network.

Stability is achieved by loading the circuit near the diode with a resistance low enough to stabilize the multiplier, yet not so low as to degrade its efficiency significantly. Fig. 7 shows the general layout and circuit diagram of the multipliers.

In the LO multiplier chain the 8-to-11-GHz band is covered by two multipliers for two reasons: one, to make it possible to filter out potentially troublesome harmonic products and two, to maintain high quadrupler efficiency and maximum power output as the frequency is varied. One quadrupler is optimized for an output frequency from 8 to 9.5 GHz, and the second is optimized for 9.5 to 11 GHz.

Multiplier Chain Filters

The frequency doubler and multiplier output signals contain many harmonics of the input signals. Only the second and fourth harmonics, respectively, are desired. Other frequency components must be filtered out to minimize spurious outputs from the HP 8780A. Four microwave bandpass filters and two pin diode switches are used to select the desired multiplier output frequencies.

A comb line structure (Fig. 8) was selected which allowed similar design and fabrication techniques to be used for all filters. The comb line structure also allows relatively broad stopbands to be realized as required by the wide modulation bandwidth for the IF multiplier chain and the tunable frequency coverage of the LO multiplier chain. Spurious passbands at least four times the filter center frequency, high-power handling, and low loss are additional important goals achieved with the comb line structure.

In the LO multiplier chain, the doubler is followed by a bandpass filter to prevent fundamental and third-harmonic energy from reaching the quadrupler and experiencing odd-order multiplication which would produce frequencies within the quadrupler output filter passband. The filters following the LO multiplier quadruplers pass only the 4× output frequencies and eliminate other multiplier outputs. Since a single wideband filter cannot do this function, one of two narrower-band filters is selected using pin diode switches. A single filter is used at the output of the IF multiplier quadrupler.

Reference

1. G.L. Matthaei, et al, *Microwave Filters, Impedance-Matching Networks, and Coupling Structures*, McGraw-Hill, 1964, p. 83.

A Combinational Board Test System

The HP 3065AT Tester provides a completely integrated set of resources for testing analog, hybrid, and digital circuits incorporating surface mounted devices, application specific ICs, and VLSI circuits.

by Michael E. Gravitz

MANUFACTURERS OF TODAY'S sophisticated electronic equipment are increasing their use of complex VLSI, ASIC, and mixed-signal devices. These devices, especially in surface mount form, pose new and formidable testing problems. To test these advanced technologies effectively, HP has developed the HP 3065AT Combinational Board Test System (Fig. 1). This new product, using the HP 3065 Board Test System[1] as a platform, retains all of the capabilities of the HP 3065 while providing additional capabilities to address the testing needs of newer devices and printed circuit board assemblies.

The HP 3065AT is a combinational tester in the sense that it supports both in-circuit and functional testing methods. This is of great value to the user because it gives the option of tailoring a test strategy to a particular need. Typically, the user can begin testing using in-circuit techniques (the quickest and easiest form of testing to develop) and then add functional tests as needed for yield enhancement. This is a powerful strategy for the user, because risk is reduced and the usefulness of the tester investment is increased.

As with the original HP 3065, the development of the HP 3065AT proceeded with parallel hardware and software efforts. The primary goal of each of these efforts was to retain complete compatibility with the original product while implementing feature sets that support testing of newer technologies.

Hardware Enhancements

The hardware developed to implement the HP 3065AT feature set complements and enhances the features already possessed by the HP 3065. The added features are:

- High-speed clock generators
- High-speed programmable drivers with complex waveform generation capability
- More precise control of existing hybrid card drivers and receivers
- 10-million-patterns-per-second data rate sequencer
- 64K-bit-deep high-resolution error logger, backtracer, and pattern capture circuitry

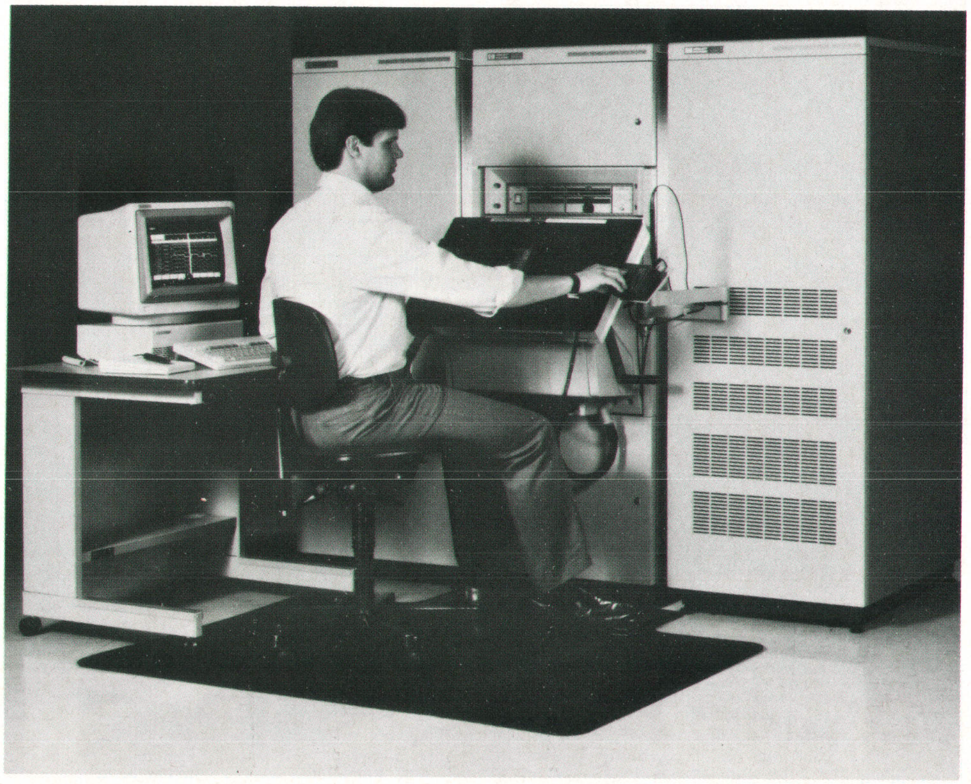


Fig. 1. *The HP 3065AT Advanced Technologies Combinational Board Test System is the most advanced board test system of the HP 3065 family of testers. The HP 3065AT has been specifically designed to test the most advanced technologies: surface mount devices, application specific ICs, and VLSI circuits like 1M-byte memories and 32-bit microprocessors.*

- Trigger receivers
- Sync output to clock external receivers
- Backtrace probe
- Higher-bandwidth, general-purpose I/O ports.

These features can best be understood in the context of the existing HP 3065 architecture and the additions made to it to produce the HP 3065AT. Fig. 2 is a block diagram of the HP 3065AT, including the scanner structure of the test head and the buses that interconnect the cards. The heart of the HP 3065AT resides on the board in the foreground and communicates with the hybrid cards and system controller via the backplane bus. This board, known as the functional controller card (FCC), consists of two separate printed circuit assemblies which, when mated, have the same form factor as an HP 3065 hybrid card. The rear printed circuit assembly of the FCC contains the digital subsystem of the HP 3065AT and mates with the backplane bus. The front printed circuit assembly contains the drivers, receivers, and associated relay structures of the HP 3065AT. In addition, the front printed circuit assembly provides interconnect signal routing to the HP 3065AT's analog bus. Finally, the front printed circuit assembly provides the connection points to the test fixture where the unit under test (UUT) is placed.

Digital Subsystem

The HP 3065AT's digital subsystem consists of three major functional elements: the timing and formatting circuits, the vector address sequencer, and the pattern capture and error logging circuits.

Timing and Formatting Circuits. The timing and formatting circuits generate the fundamental timing for all aspects of FCC operation. A signal called Reference Clock is the master clock on the FCC; all other clocks and timing signals are derived from it. Reference Clock can be generated either by taking a signal directly from the UUT or by using an on-board, rate multiplying phase-locked loop. The phase-locked loop (PLL) can be driven by one of two sources. First, an on-board crystal oscillator can be used. Second, a signal from the UUT can be presented to the PLL. When the PLL is used, Reference Clock may be an integer or fractional multiple of the input clock.

Each rising edge of Reference Clock is referred to as an event marker. This concept, reflected directly in the software, is used in the creation of test vectors. During a single test vector several things usually occur: the test vector from the hybrid cards is driven to the UUT, the response data from the UUT is compared to the expected response contained on the hybrid cards, and the high-speed lines (clocks and formattable outputs) transition at selected times to produce control signals at the UUT. For a given test vector cycle the programmer may wish to drive the hybrid card data at time x, receive data from the UUT at time y, have the formattable drivers transition at still other times, and end the vector at time z. All of the above transition times may be different for different test vectors. Combinations of drive time (x), receive time (y), period length (z), and formats can be programmed into the FCC and then called on the fly on a test-vector-by-test-vector basis. Each of these combinations is referred to as a timing set. Two typical

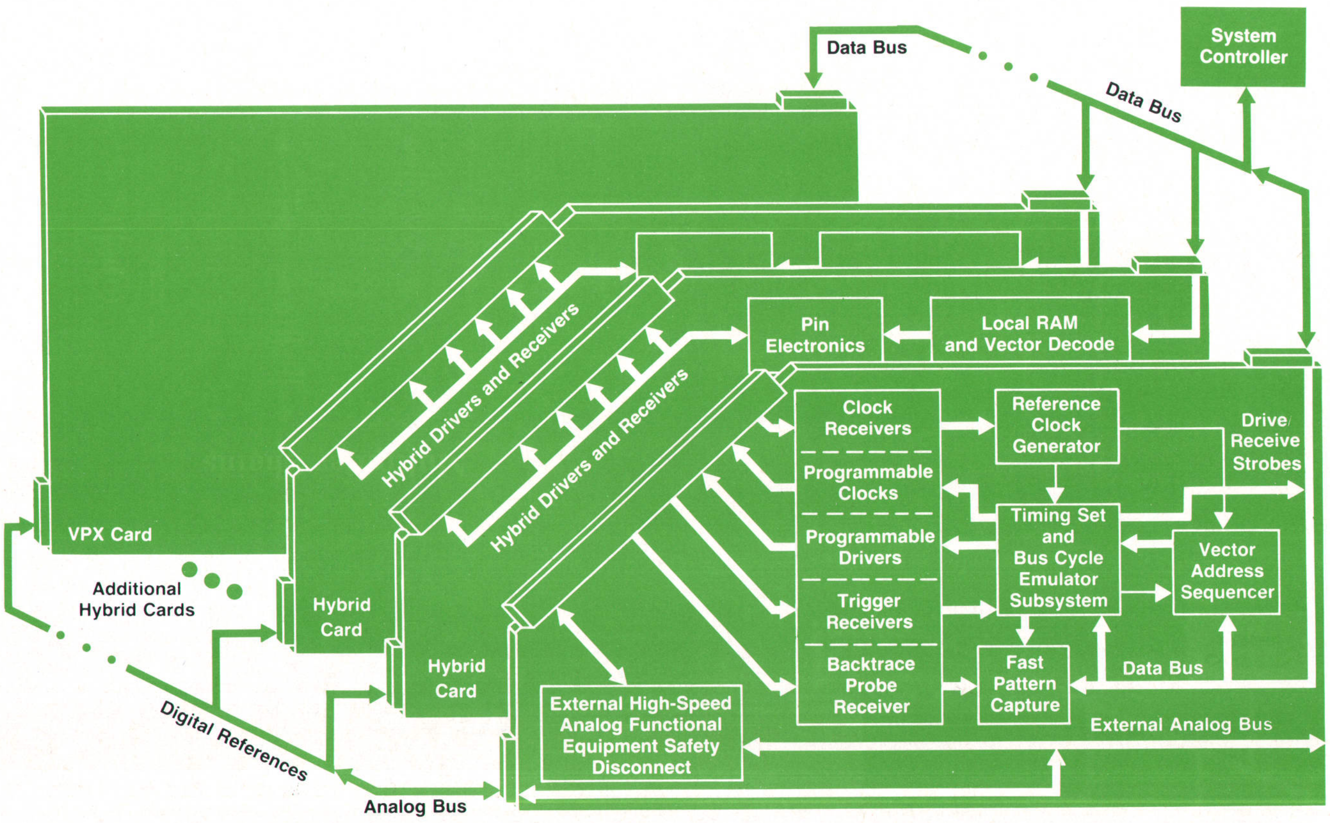


Fig. 2. *Block diagram of digital subsystem for the HP 3065AT.*

timing sets appear in Fig. 3.

There are 240 event markers available to the user. They can be used in blocks of 16 to form up to 15 timing sets. This arrangement gives the user the flexibility to form timing sets ranging from 15 sets that consist of 16 events each to one very large timing set consisting of 240 events. Combinations of timing sets between these two limits are possible as long as the block of 16 event markers rule is observed.

Six formattable drivers and two clock generators form the kernel of the enhanced UUT excitation capability of the HP 3065AT. The formattable drivers, programmed via the event marker/timing set approach just described, are intended for use as control lines in the test environment. The clock generators, as their name implies, are for use in driving clock lines on the UUT. Although similar, there are several characteristics that differentiate the operation of formattable drivers from clock generators.

Formattable drivers have the ability to trade off between events and timing sets while clock generators have only one timing set of 240 events.

To facilitate the handling of wait states, the clock generators are free-running with respect to the rest of the timing system. That is, if a wait condition is encountered (a feature described below in the sequencer section), the portion of the timing system controlling the formattable drivers and hybrid cards halts at a particular event, holding their lines static. However, the clock generators continue operating, resulting in an uninterrupted stream of clock pulses for the logic on the UUT. When the cause of the wait condition is removed, the formattable driver/hybrid card portion of the timing system synchronously restarts at the next event that has the proper edge relationship with the clock generator. This feature is a key element in the HP 3065AT's ability to deal with logical structures that have variable response times. Examples of these structures abound in 16/32-bit microprocessor bus environments.

To make maximum use of these resources, the formattable drivers and the clock generators are relay switched to one of two pins per driver/generator.

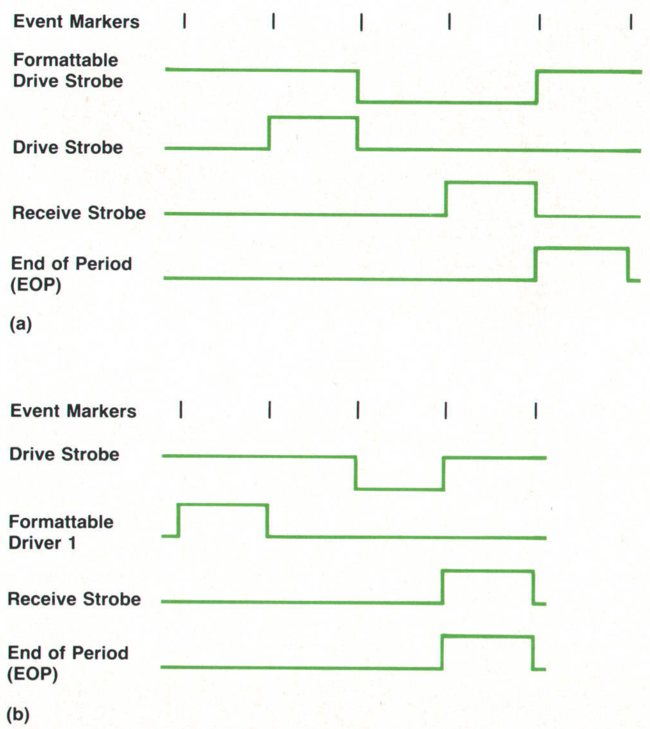


Fig. 3. *Two typical timing sets of waveforms for the HP 3065AT.*

Vector Address Sequencer. The vector address sequencer (VAS) of the HP 3065AT sends vector addresses to the hybrid cards. This sequencer, like the vector processors used by the HP 3065, is programmed in a high-level language called VCL (vector control language). While retaining language compatibility among the processors, the VAS has operational characteristics tailored to functional testing. Higher (10 MHz) operating rates, the elimination of dead cycles, and the ability to handle nondeterministic UUT response timing are examples of features added for functional testing.

The vector address sequencer has three types of sequencing instructions available for controlling test execution. They are:

1. An instruction that sends one vector address to the hybrid cards.
2. An instruction that sends a stream of vector addresses to the hybrid cards.
3. An instruction that sends a single vector address and branches to any location in the VAS program space. The branch can be either conditional or unconditional. A conditional branch can be qualified by the UUT trigger lines, the hybrid card's master pass/fail line, the system controller, or an external condition (accessible via the external bus). In all cases the source of the qualifier can be selected on the fly on a per vector basis.

This instruction set, which adheres to the notion of a reduced instruction set architecture, is not reflected directly in VCL. Instead, VCL presents the user with traditional constructs (loops, branches, subroutines, etc.) and leaves the task of conversion to the VAS constructs to the VCL compiler.

Internally, the major sections of the VAS include the directory RAM, the directory RAM pointer, the sequence RAM pointer, and the sequence RAM. These components and their relationship to the hybrid cards are shown in Fig. 4.

The directory RAM holds the sequencing instructions to be executed during the test. The VAS can perform two distinct types of sequencing instructions: execute the current instruction and fetch the succeeding instruction from the directory RAM, or execute the current instruction and fetch the instruction at the directory RAM address specified in the branch field of the current instruction.

Each sequence RAM word is 40 bits long and consists of three fields: the 15-bit sequence RAM address field (S field), the 11-bit branch address/sequence count field (B field), and the 13-bit control bit field (C field). At the beginning of each instruction the contents of these fields are used to determine the type of instruction the VAS is to execute and which qualifiers are to be active for the instruction. Various pointers and counters loaded from these fields are used to establish control over instruction execution:

- The sequence length counter is a 15-bit-wide counter

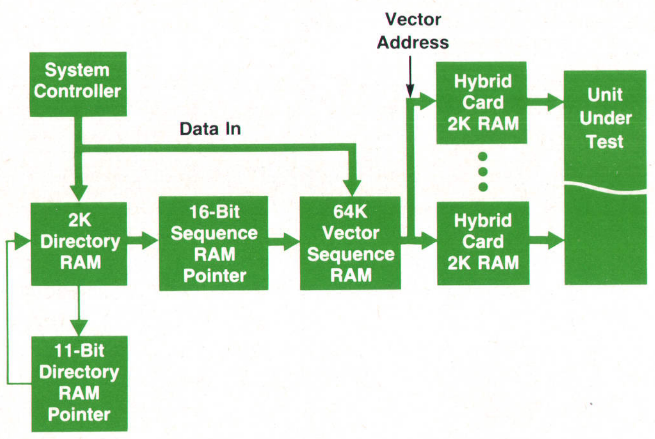


Fig. 4. *Vector address sequencer circuit.*

that can be loaded from a directory RAM field. The terminal count is used to control the sequencing of directory instructions; the next directory instruction will not be executed until the sequence length counter has reached terminal count. This capability is used to implement a class of instructions that cause a linear sequence of sequence RAM locations to be emitted as vector addresses.

- The directory RAM pointer is an 11-bit-wide counter used to access the directory RAM. The counter can be incremented, held, or loaded from a directory RAM field. In addition, the counter can be preloaded directly by the test system controller.
- The sequence RAM pointer is a 16-bit-wide counter used to access the sequence RAM. The counter can be incremented or loaded from a directory RAM field.

The three fields of the directory RAM and the counters and pointers just described form the control structure for the VAS. By having them control the way in which the sequence RAM is accessed, traditional flow structures such as loops, branches, and subroutines can be readily implemented. This structure retains the HP 3065's need to store only unique vectors in the hybrid card RAMs. That, coupled with the HP 3065's keep/toggle capabilities described in reference 1, gives this architecture a very powerful vector generation capability.

Pattern Data Capture and Error Logging Circuits. The pattern data capture and error logging circuits perform the functions of counting the number of test vectors applied to the UUT, logging the response from any hybrid card receiver, and comparing the results received from any hybrid card receiver against an expected response.

The pattern data capture and error logging functions occur on a single channel at a time, with a maximum single-pass storage depth of 64K bits. When used in comparison mode, the controller loads data into the response RAM which is later compared to a set of responses from a UUT node. In storage mode, the data from a UUT node is stored in the response RAM and collected after the test by the controller. There are two distinct storage modes: normal and fast pattern capture. In normal mode the response from the node of interest is sampled at the hybrid card receive strobe time, resulting in one sample stored per test vector. In fast pattern capture mode a node is sampled at a rate set by the reference clock. This results in a peak sample rate of 33.3 MHz. These three modes of operation—comparing and normal and fast pattern capture—are used by the HP 3065AT software to support the graphical debug, autolearn, and backtrace functions.

The other major portion of the error logging and pattern capture circuitry is the vector counter. This 24-bit counter is used by the HP 3065AT software to relate the linear count generated by the counter to the user's vector number which appears in the source code. In addition, the counter is used to address the response RAM. Since the counter is 24 bits wide, it can count up to 16,770,000 vectors. Its relation to the response RAM is such that when the counter is within 64K of terminal count the RAM will store vectors. Since the vector counter can be preloaded by the system controller, an offset can be loaded to indicate when logging and comparing should begin. Finally, control bits are emitted by the VAS to control the counter and prevent it from advancing during UUT initialization sequences.

To ensure that the data compared and captured by the FCC error logging and pattern capture circuitry is the same as that obtained by the hybrid card receiver, it is necessary to delay the strobe used to capture the data on the FCC with respect to the strobe sent to the hybrid card. This is done using a programmable delay line. By implementing the delay in this fashion it was possible to develop an autocalibration procedure that adjusts the delay for each receiver path to the response RAM. This procedure is transparent to the user and results in enhanced overall product performance.

Analog Subsystem

The analog subsystem resides on the front portion of the FCC. The capability provided by the components residing on this portion of the FCC are:

- 200 V/μs overdriving tristate drivers for applying clock and formattable output waveforms to the board under test. Each driver is relay switched between two UUT paddle pins.*
- Four threshold comparators used to generate VAS branch qualifiers and timing system waits. Each comparator is relay switched between two UUT paddle pins.
- A threshold comparator used to buffer and level shift a UUT clock input into the phase-locked loop in the timing subsystem. Relays multiplex this comparator from one of four paddle pins.
- A threshold comparator used to buffer and level shift the rear-panel external qualifier input to the VAS.
- A dual-threshold comparator for use as a buffer and level shifter for the handheld backtracing probe.
- A 50Ω pass-through to the paddle pin field for four external signal buses from the rear portion of the FCC.
- All hardware required to support earlier HP 3065 confirmation and diagnostic (C/D) testing plus the additional hardware to support HP 3065AT C/D testing.

To fit this circuitry in a space already being used by current HP 3065 C/D electronics, it was necessary to use printed circuit board space more efficiently. Savings came in three areas: the placement of two single-threshold comparators on the same hybrid module footprint (1 inch by

*These UUT pins are called paddle pins because they are shaped like tiny paddles.

1.5 inches) where there was originally a single dual-threshold comparator, the replacement of mercury-wetted relays with dry-reed relays where reliability would allow it, and the use of integrated relay drivers. It was possible to substitute dry-reed relays for 96 of the 119 mercury-wetted relays and to use dry-reed relays exclusively for the 42 relays needed by the HP 3065AT portion of the electronics. In addition, three of the four modules needed by the HP 3065AT can be of the single-threshold variety.

System Software

The software for the HP 3065AT is an integral part of the product. It is a logical extension of current HP 3065 software and is designed to be fully compatible with it. The software package for the HP 3065AT can be considered to operate in one of seven distinct areas. They are board preparation, functional test preparation, in-circuit test generation, in-circuit test debug, functional test generation, functional test integration (debugging), and functional test execution. Some of these areas are the same as for the HP 3065 while others are entirely new. Each activity is described briefly in the sequence in which it generally occurs, with a more detailed description following.

Board Preparation. Board preparation consists of those activities required to get the HP 3065AT ready to generate in-circuit tests. This consists of describing the devices used on the board (usually by manufacturer part number) and the connections among them. The HP 3065AT required some extensions in this area, mostly in the generation of custom part libraries.

Functional Test Preparation. The functional test preparation phase, special to the HP 3065AT, consists of three tasks: defining the functional test resource requirements, defining any nodes where bed-of-nails access is not desired, and adding any special backtrace requirements to the backtrace file.

In-Circuit Test Generation. The in-circuit test generation phase is a carryover from the HP 3065. It consists of running IPG-II, the in-circuit program generator. IPG-II analyzes the board data and prepares the test plan, fixture wire list, shorts test, and analog and digital in-circuit tests. Extensive modifications to IPG-II were required to have it operate with the HP 3065AT hardware.

In-Circuit Test Debug. The in-circuit test debug phase is also a carryover from the HP 3065. In it, the analog and digital in-circuit tests generated by IPG-II are debugged to ensure that the UUT is being tested properly. Typically, users begin production testing of boards at this time while concurrently developing functional tests.

Functional Test Generation. Another phase added to support the HP 3065AT hardware, the functional test generation phase, involves defining test vectors used to stimulate the UUT. These vectors can either be from simulators (via CAE links) or manually generated. UUT responses can be obtained from simulator output or learned by the HP 3065AT from a known good board.

Functional Test Integration. The test integration phase, also special to the HP 3065AT, allows the user to run the functional test against a known good board. This is an interactive mode, and the user is provided with tools to analyze failures. Techniques such as a nodal graphical debugger, test looping, and sync signals for external instrumentation are provided.

Functional Test Execution. Functional test execution is the final phase of operation and it is also special to the HP 3065AT. This is the production test environment. The major functions performed are go/no-go testing, fault diagnosis, and the logging of data. Fault diagnosis involves the use of an algorithm that automatically traces back to the first failure.

These seven sections define the general operating environment the user sees when operating the HP 3065AT. A more detailed description of this environment along with some of the implementation details follows.

Board Preparation

The HP 3065AT uses the same methods as the HP 3065 to enter board topological information—the board description editor or an HP CAD-Vantage link. However, three extensions of the board preparation process were added for the HP 3065AT:

1. Use of the FCC to create higher-performance in-circuit tests.
2. Downloading of test vectors from external sources to a newly created pattern capture format (PCF).
3. Delivery of backtracing information to the device libraries.

By having the FCC available for in-circuit testing the user can more effectively test those devices that require capabilities such as clock synchronization, formatted waveforms, higher clock rates, no dead cycles, and wait conditions. The language controlling the FCC and the target for PCF information is VCL. VCL is an extension of the language used to control the processors in the HP 3065. VCL for the HP 3065 has three sections: declaration, vector definition, and vector execution. These sections have been expanded with the addition of FCC and PCF specific instructions which allow control of FCC attributes and PCF syntax. In addition, two new sections have been added to give control of the testing environment and the timing and formatting capabilities of the FCC. In all, 27 entirely new and seven modified VCL statements have been added to the existing 49-statement VCL instruction set.

The PCF data format can be used for design system and logic simulator outputs or by users who are manually writing test vectors and wish to use a compact vector data format. Its general form is:

```
  D D D D B B R R R R
“ 0 0 0 0 0 1 H H H H ”
“ 0 1 1 1 H L L L H L ”
“ 0 0 . . . . . . L L ”
“ 1 Z Z 0 0 0 H L X X ”
```

This example shows four vectors of data for ten I/O pins. The first row indicates whether a node is a driver (D), a receiver (R), or bidirectional (B). The next row down begins state definitions. The definitions for drive and receive lines are:

(continued on page 59)

Interactive Graphical Debugging Package

Given the complexity of writing functional tests for modern printed circuit assemblies and in-circuit tests for VLSI parts, it was obvious that a comprehensive debugging package would be a critical part of the HP 3065AT. Digital status is useful for debugging tests for simple parts, but can produce pages of data for a large board test, obscuring the data the test engineer really needs. The HP 3065AT uses an enhanced version of digital status to give the test programmer information about the test failure, but augments this with a highly flexible virtual logic analyzer to allow a more detailed analysis of the cause of the failure. While recompiling a test to implement debugging features is not too bothersome when working with simple parts, it would have a major impact on test development time for the lengthy and complex tests for which the HP 3065AT was designed. Hence, the HP 3065AT debugging package provides test execution control, timing control, and interactive sync pulse features that reduce the need to recompile to a minimum. A major part of writing large tests is predicting the response of the device under test. The debugging package also provides several autolearning features that allow the test engineer to use a proven good device to provide expected responses.

To increase the utility of the graphical debugging package, the user interface was made extremely flexible. There are over 65 new commands associated with the package, most supported by softkeys. The softkeys are arranged in a three-level menu structure which can be completely customized by the test programmer to suit personal styles. All of the commands can also be entered from the command line, and since they are all BT BASIC extensions, can be put into BASIC programs to allow their use in specialized production test procedures.

Virtual Logic Analyzer

The virtual logic analyzer is the centerpiece of the HP 3065AT's graphical debugging package. This analyzer allows the user to collect and display nodal responses for the device under test (DUT) without the need to connect an external instrument. The virtual logic analyzer provides the following benefits:

- Existing system connections are used to collect data.
- It is synchronous with the tester so the displayed data is what the system actually uses for pass/fail decisions.
- Expected data can be displayed for comparison with actual data and failure information can be highlighted.
- Displayed data can be collected at vector rates (on response strobe) or at event marker rates down to 30-ns resolution.
- Data can be collected at the tester, saved in a file, and recalled later for analysis.
- The interface is consistent with the rest of the system (node names or device and pin names).

The virtual logic analyzer acts like a test engineer's spreadsheet in that it uses the terminal as a 56-vector-by-8-node (columns by rows) window into a data space of up to 67 million vectors by 99 nodes. Interactive controls are provided to move the window around this space or to dump a selected range of vectors to a hard-copy device. The software uses the 64K response RAM of the FCC (functional controller card) to collect data from the test head and gets expected data from the state file.

The expected data can be added to the state file by three mechanisms: using the node state format to transfer data from a logic simulator on a CAE system, using the autolearn features to capture the expected data from a known good device, or using the test-head simulator to generate expected data from the compiled test. The test-head simulator generates the programmed I/O states of the test being debugged, which can then be displayed to help debug the test even before using the tester hardware.

An added feature of the virtual logic analyzer is that it can display graphical data on an inexpensive text terminal by using the HP line drawing characters that are standard on most HP terminals. Color is also used if a color terminal is available, greatly enhancing the readability of the displayed data.

Interactive Test Control

To aid the test engineer further, the HP 3065AT graphical debugging package provides a very useful set of test control commands. The support for these commands was designed into the FCC hardware during the initial definition phase of the HP 3065AT. The software takes full advantage of the FCC response RAM and vector counter to implement these features and, as it turned out, the code had to be tediously handcrafted since these hardware resources are used in different ways simultaneously.

The user is able to control the execution of the test by specifying the mode of execution (single execution or recycle continuously) and the halt criteria (first failure, specified vector, or end of test). The programmer also can run to a vector, pause with the test active, and single step from that point. These features are very useful when using an oscilloscope or external logic analyzer to resolve time related problems.

The overall timing of the test can be altered interactively by changing the reference clock that determines the event markers (FCC) or by changing the vector cycle and receive delay (VPC/VPX). This feature can be used to determine if the DUT has timing problems, to debug fixture related failures, or to verify that the test itself has enough timing margin to account for part variances.

To help control external instruments, the debugging package allows the user to define and manipulate up to 1048 sync pulses on selected vectors of the test. The sync pulses can be used to trigger an oscilloscope or clock an external logic analyzer and to mark the drive and receive points of each vector (the sync pulse goes low at the drive strobe and high at the receive strobe). The current sync pulse definition can be saved for future use.

Autolearn

The autolearn commands of the debugging package allow the user to collect and save nodal responses from a known good device for three purposes: to generate expected data for backtracing functional tests, to generate expected data for the virtual logic analyzer, and to create sets of PCF vectors for in-circuit tests of ROMs, ASICs, and other parts. Autolearn also allows the test programmer to characterize the running time of nondeterministic tests for HP's Safeguard™ analysis.

When collecting expected data, the programmer must worry about whether or not the test can produce consistent results. If parts are not initialized properly or race conditions or hazards exist in the circuit, the response of the board or device can vary from test to test. After the data has been learned, the debugging package provides commands to confirm the data and display discrepancies, or to mask the discrepancies by changing the response from a high or a low to an unknown state (X). This can be done automatically or interactively. The package also allows cyclic redundancy check (CRC) values to be generated in lieu of a state response. This is usually done in the case of a very long test where stored response data would be unwieldy. The debugging package allows the programmer to alter the learned data further manually.

The software maintains the response data in the form of a linked list of transitions (equivalent to the node state format). This has been shown experimentally to optimize both the data space required to hold the data and the time complexity required to operate on it. Some very interesting algorithms were developed to transform these linked lists into hardware download data, to compare two lists for consistency, and to perform the masking operation between two lists.

The autolearn features are also used to assist in generating a test for difficult parts like ROMs and application specific ICs. An interesting characteristic of these parts is that the inputs are usually very easy to generate (a count sequence for addresses or other algorithmically generated sequences) while the outputs are very difficult or tedious to predict. If the programmer can create the inputs, autolearn can be used to determine the output responses and automatically produce PCF data files for the test. These files can then be merged into the original test, producing a complete VCL test for the part. As with node data, the PCF data can be confirmed and masked to checked for inconsistent results. An added benefit of this procedure is that if a production change occurs that alters the device's responses, the autolearn process can be repeated easily to update the test.

Acknowledgments

The HP 3065AT's graphical debugging package is special in that almost all of the people on the HP 3065AT team have had a role to play in its definition or development. Special credit should go to the following people: Chris Cain and Mike Wahl for the long, late hours spent developing nearly 7000 lines of hardware interfacing code, John Heumann for writing the test-head simulator software package, Elaine May for her work on upgrading digital status to support the FCC and writing the autolearn outputs code, and Kevin Keirn for his transition list algorithms. I would especially like to acknowledge Bill Groves for his pioneering work on functional test debugging in the late 1970s. The DTS-70 Probe program was the inspiration for the debugging and backtracing packages for the HP 3065AT Test System.

George Booth
Development Engineer
Manufacturing Test Division

Drive Line:		Receive Line:	
0	= low	L	= low
1	= high	H	= high
Z	= high impedance	X	= don't care
.	= repeat from column directly above	.	= repeat from column directly above

The time sequence of application of the vectors is read from top to bottom. The actual nodes assigned to the PCF data are defined in the declaration section of VCL. The PCF statements appear in the vector execution section of VCL either directly or through the use of the include statement.

The mechanics of board preparation enable the user either to develop setup-only in-circuit tests or to generate complete, ready-to-run tests. In developing setup-only tests the user need only specify resource information, not detailed waveform and test vector data. This allows the test programmer an earlier start on fixture building. While the fixture is being built, the details of the nonlibrary tests can be filled in during the functional test generation process.

Backtracing is a significant part of the HP 3065AT's capabilities and is used to diagnose failed devices during functional testing. To support backtracing, additional device library information is required relating outputs to inputs and the states of bidirectional lines. This information is included as part of the HP 3065AT device library, but for those devices not in the library, two statements can be used to generate it:

```
trace ... to ... (declaration)
when (declaration)
```

The trace ... to ... statement informs the backtracer which output and bidirectional pins of a device are affected by which of its input and bidirectional pins. For example,

```
trace Bus_Req, Wait_Line, Address_Lines to Chip_Select, Enable
```

indicates that the outputs, the first three pin groups, are traceable back to the inputs, pin groups Chip_Select and Enable.

The when statement gives the backtracer information about the device's activities. Given certain input states, when tells the backtracer whether bidirectional pins are inputs or outputs. It also tells the backtracer when the bidirectional and output pins are inactive. For example:

```
when In_Enable is "10" inputs Data_Bus
```

indicates that the bidirectional pins in pin group Data_Bus will be inputs when pin group In_Enable is set to "10." Similarly:

```
when In_Enable is "01" outputs Data_Bus
```

indicates when the Data_Bus pins are outputs. Finally, the following statement indicates when they are turned off:

```
when In_Enable is "00" inactive Data_Bus
```

Functional Test Preparation

This section is special to the HP 3065AT and exists for the sole purpose of functional testing. Because of this, it can be eliminated if only in-circuit testing is anticipated.

The functional testing performed on a board can consist of one or more tests. Each test may test the entire board (using all I/O pins), a cluster of the board (using a subset of the I/O pins), or a group of parts (driving and receiving from internal nodes). Each of the tests needs to be defined and the I/O nodes for that test declared. The specific I/O requirements of each test must be defined so that IPG-II can subsequently optimize pin assignments.

To define a functional test, the HP 3065AT user writes a VCL source file that is analogous to the in-circuit setup-only test. This file describes the processor desired (VPX or VAS), the node assignments and groupings, input and output pins, power pins, logic family types, and FCC resources (timing sets, clocks, triggers, and wait lines). At this point the functional test is setup only, meaning the resources for the test have been specified but not the detailed vector and timing information. The specifications come later during the functional test generation phase.

The HP 3065AT continues to use the hierarchical file system developed for the HP 3065. As a result, extensions to that system have been put into place to support the functional test environment. For example, all functional test related files exist under a directory named functional, which in turn exists under the main board directory. Each functional test then has a directory under functional. The name of each directory can be specified by the user but the use of a name indicative of the test being performed will result in a more meaningful path name. Finally, the VCL test itself must be named test. This results in a path name that might appear as:

```
/⟨board⟩/functional/video_ram/test
```

After creating a source file for each of the planned functional tests, the user creates a file named func_tests under the functional directory that lists all the directory names of the functional tests desired. The functional program generator (FPG) and IPG-II use information from this file to create test information. The ordering of tests within this file also establishes the order of tests generated in the test plan.

Functional testing, because of its performance orientation, imposes some constraints on the bed-of-nails fixture. There will be nodes that for reasons of loading or access should not or cannot be probed. To support this constraint, nodes and devices not to be probed can be listed in a file called nonail. Listing a node or device in this file causes two things to happen. First, IPG-II will not assign tester resources to the nodes and devices and no in-circuit test will be generated for them. Second, FPG will generate a manual (handheld) probe test for the listed nodes and devices.

After the func_tests and nonail files are created, the user is ready to run a utility called func_prep. This utility runs the appropriate compiler on each file. When this is complete the functional tests will have object files and a backtrace source file (it is a source file so the user can edit it if necessary). In addition, there will be a nonail object file and a state file to hold board state information (yet to be learned).

Finally, after editing the backtrace source file to add comments or to modify the flow of the backtrace tree (optional), the user can run the backtrace compiler to generate the backtrace object file.

In-Circuit Test Generation

From the user's standpoint, the process of in-circuit test generation for the HP 3065AT is no different than that for the HP 3065. Internally, however, there are some significant differences. To retain compatibility between in-circuit and functional testing the algorithms used to assign tester resources are altered. For functional testing the main change is that all tester resources being used for excitation must be available for the duration of a given test. This means that they cannot be multiplexed with either board outputs or internal nodes. Of course, for other functional tests, excitation pins can be multiplexed to other nodes. The information needed to assign tester resources appropriately is available to IPG-II in the compiled files: backtrace.o, nonail.o, and func_tests.

When writing the test plan, IPG-II adds a functional test section that consists of two parts. The first part is a subroutine containing the functional go/no-go tests. Initially these tests include comments to explain each test. This is done on the premise that the user will first debug the in-circuit tests. Each functional test has the general form:

```
test 'functional/⟨directory⟩/test'
  if dutfailed then
  call backtrace 'functional/⟨directory⟩/test'
endif
```

The second part of the functional test section is the backtrace subroutine. This subroutine, using five new BT BASIC statements as building blocks, controls the application of tests used during backtracing and the reporting of results to the test plan. As written by IPG-II, it is a complete subroutine that can be custom tailored by the user. The need for custom tailoring is in part driven by the need to exploit the customized backtracing paths defined during the editing of the backtracing source. Recall that this code was originally emitted by the functional program generator.

In addition, the backtracer is capable of invoking tests that are external to the test that originally caused the UUT failure. This capability, supported by the external_test (btnode$) statement, helps to give the HP 3065AT the ability to backtrace through analog circuitry.

In-Circuit Test Debug

This step is the same as with the HP 3065 except for the fact that the graphical debug module (GDM, described later in the function test integration section) of the HP 3065AT can be used to aid in debugging the in-circuit digital tests. This step assumes that a fixture has been built and verified for wiring correctness. Given a fixture, the main tasks completed in this section are:

- Complete short-circuit tests by running them with autolearn on.
- Verify proper operation of UUT analog tests.
- Verify proper operation of in-circuit digital tests. The HP 3065AT graphical debug module can be used as a debugging aid.

Functional Test Generation

The functional test generation phase is a new phase of test development developed expressly for the HP 3065AT. In this phase, the detailed test vector, waveform, and internal state information for functional testing are added to the setup-only functional and in-circuit tests which were developed during the functional test or board preparation phases.

As discussed in the board preparation phase, VCL and PCF enable the user to deliver waveform and timing information efficiently to the compilers. After the sources for the enhanced library and functional tests are completed, they can be compiled. The statements accomplishing these tasks are compile digital/Uxx; debug, list for each in-circuit test and compile functional/⟨test name⟩/test; debug, list for each functional test. The list option places a compiled listing of the user's source file in a file name digital/Uxx.l or functional/⟨test name⟩/test.l. The debug option generates two files:

```
functional/⟨test name⟩/test.d
functional/⟨test name⟩/test.v
```

The .d file is the same as the one generated by the HP 3065. It contains the debug data used by the digital status statement and generates a comprehensive debug list if a failure occurs in the test. The .v file is used by the HP 3065AT and contains vector information for the graphical debug module.

The other major activity in this phase is to generate the node state information required for backtracing if a failure occurs during a functional test. By definition, internal nodes are in the circuit being tested but are not driven or received during the test.

Node state data can be downloaded from another system to a target format called node state format (NSF) or learned by the graphical debug module from a known good board. If the learning approach is used, it will be done during the test integration phase.

The NSF, like the PCF, is deliberately compressed. For each node, the format shows only the states that change and the number of the vectors where the states change. This format is sometimes referred to as a first-derivative format, since only the changes of state are shown, rather than all states.

A transition consists of a vector number followed by an equals sign and the changed state for that node. For example, 28="1" indicates that the node changes to a high state at vector number 28. The possible states are "0" (low), "1" (high), and "X" (don't care). An example of a portion of a node state file is shown below.

```
node "c2-1"
  1="0" 4="1" 6="X" 7="0" 8="X" 9="1"
  14="X" 15="0" 20="1"

node "c2-2"
  1="0" 3="X" 10="1" 15="0" 25="1" 32="X"
```

Once the NSF has been generated, a standard compile statement will convert it into object code in the state file. The statement appears as:

```
compile "functional/⟨test name⟩/node_states;states
```

Test Integration Phase

In this phase, the compiled tests are run and debugged. For the HP 3065, this task historically has been fairly straightforward; automatically generated in-circuit tests are inherently easy to debug. However, for the HP 3065AT, debug is potentially more difficult. This is because of the nature of the testing being done—nonlibrary in-circuit testing of increasingly complex VLSI devices and functional testing in general. To help satisfy the debugging requirements of these testing methods, a graphical debug module (GDM) has been developed. This module, while developed as part of the HP 3065AT, is also compatible with the HP 3065, with some restrictions.

The GDM is controlled by a two-level softkey structure which is loaded onto the softkey pad by executing the command debug ⟨test name⟩. Then the three main sections of the GDM appear. They are the test control module, the logic analyzer module and the autolearn module. While softkey control was envisioned as the primary means of GDM control, GDM can also be controlled from the command line or a BT BASIC program.

Test Control Module. The test control module provides statements that control the execution of a digital test, change its timing parameters, and add synchronization pointers to it. All of these can be done temporarily during a debugging session without recompiling the test.

An aid in debugging a digital test is the ability to control how it is executed. The test control module offers a number of options. They are:

- Loop on test ignoring errors.
- Run test until first failure, display it, and repeat from beginning.
- Loop on test to specified vector.
- Run test to first failure; halt and tristate drivers.
- Run test to specified vector; halt with drivers and clocks active.
- Single-step through test.

The execution control statements appear as one of three second-level softkey menus under the execute control softkey. This softkey menu is shown in Fig. 5a. Softkeys labeled

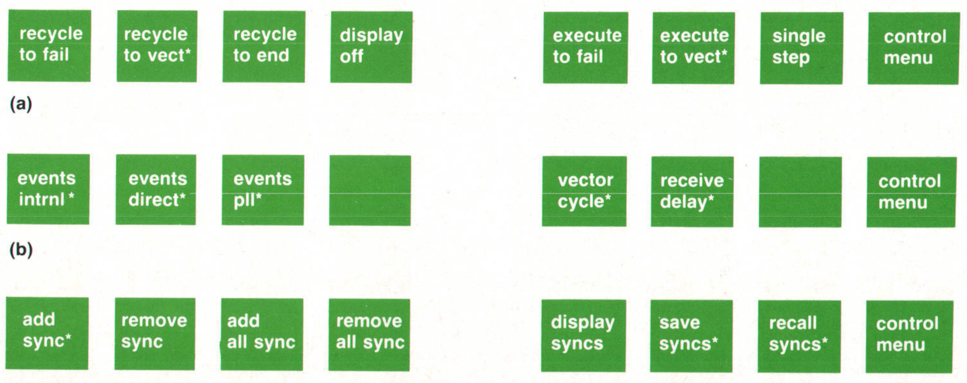


Fig. 5. *(a) Execution control statement softkey menu. (b) Timing control softkey menu. (c) Synchronization control softkey menu.*

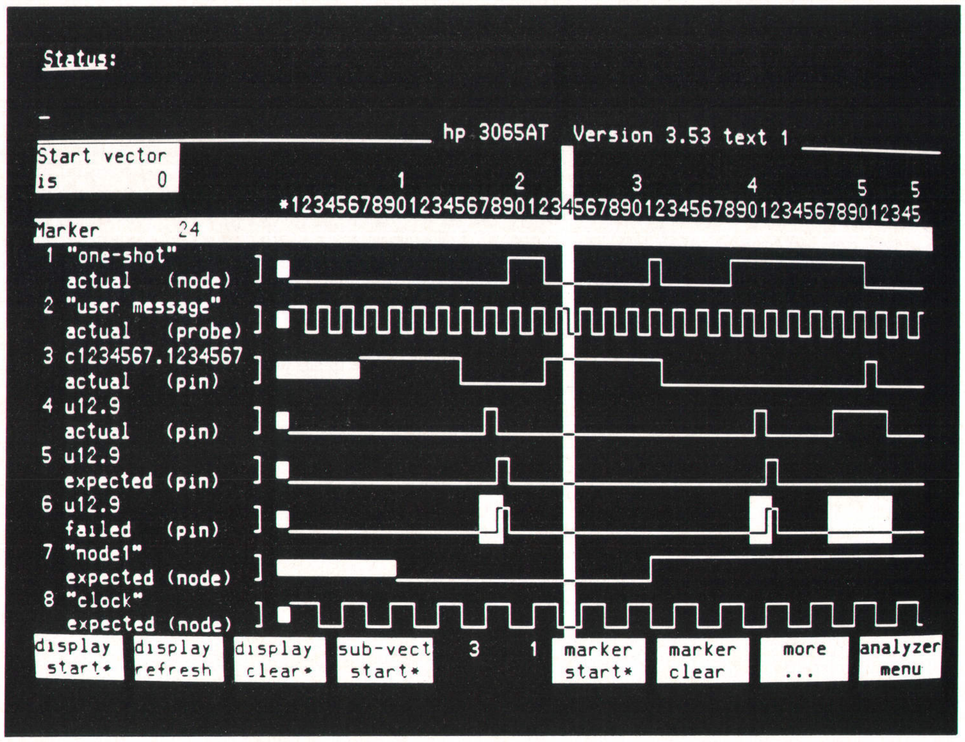


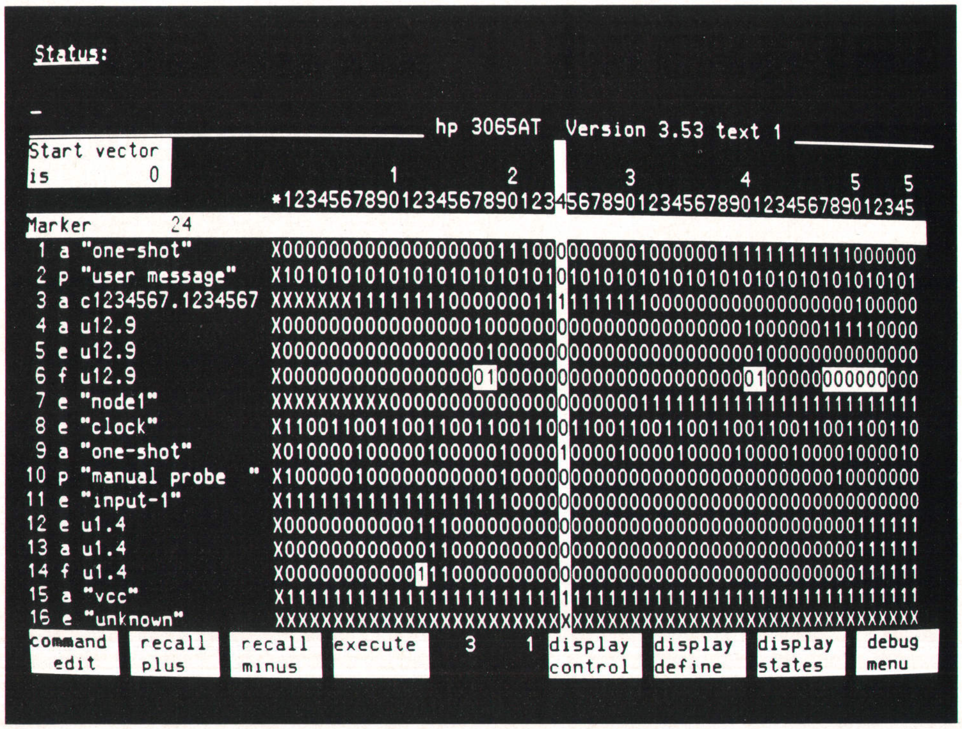
Fig. 6. *Example of graphical waveform display format.*

with an asterisk place the BT BASIC command on the command line and require information as to vector number, etc. to be added before execution. Since these instructions can also be executed from the command line, it is possible to chain them together. For example:

```
execute to 9767 | single step
```

causes the hardware to execute the current test to vector 9767 and then execute one more vector (9768).

The test timing control submodule gives the user control over some aspects of test timing. If the test specifies processor VAS, the event marker frequency can be modified; for the other two processors it is possible to modify vector cycle and receive delay times. The softkey menu appears as shown in Fig. 5b.



```
Status:

_
________________________ hp 3065AT  Version 3.53 text 1 ________
Start vector
is          0
                                  1         2         3         4         5   5
                        *1234567890123456789012345678901234567890123456789012345
Marker        24
 1 a "one-shot"          X000000000000000000111000000000100000011111111111000000
 2 p "user message"      X101010101010101010101010101010101010101010101010101010
 3 a c1234567.1234567    XXXXXXX111111110000000111111111000000000000000000100000
 4 a u12.9               X000000000000000001000000000000000000000001000000111100000
 5 e u12.9               X000000000000000001000000000000000000000001000000000000000
 6 f u12.9               X000000000000000001000000000000000000000010000000000000000
 7 e "node1"             XXXXXXXXXX00000000000000000000001111111111111111111111111
 8 e "clock"             X110011001100110011001100110011001100110011001100110011 0
 9 a "one-shot"          X010000100001000010000110000100001000010000100001000010
10 p "manual probe   "   X100000100000000000100000000000000000000000000010000000
11 e "input-1"           X111111111111111111100000000000000000000000000000000000
12 e u1.4                X000000000001110000000000000000000000000000000000111111
13 a u1.4                X000000000000110000000000000000000000000000000000111111
14 f u1.4                X000000000001110000000000000000000000000000000000111111
15 a "vcc"               X111111111111111111111111111111111111111111111111111111
16 e "unknown"           XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
command  recall  recall  execute   3   1 display  display  display  debug
 edit     plus   minus                   control  define   states   menu
```

Fig. 7. *Example of state display format.*

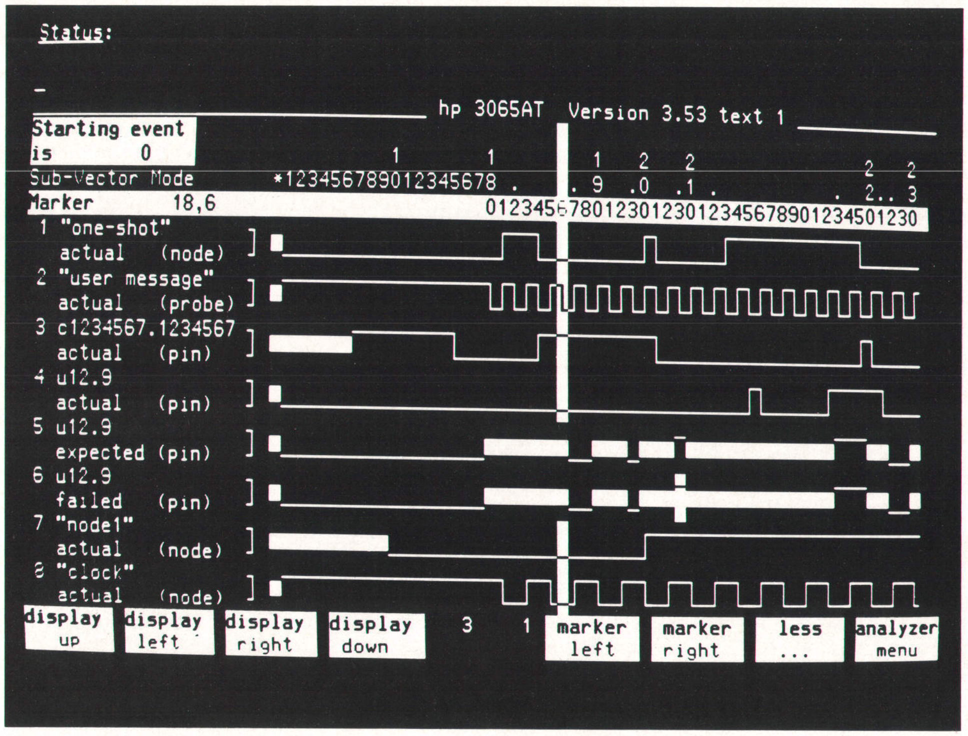


Fig. 8. *Example of mixed vector and event display.*

The synchronization control submodule works only with the VAS. It provides commands that give the user the ability to manipulate synchronization signals without recompilation. The synchronization control softkey menu is shown in Fig. 5c.

Virtual Logic Analyzer. The virtual logic analyzer is a highly interactive tool that gives the user the ability to display either the waveforms or the logic states of the circuit under test. This feature operates with either the HP 3065 or the HP 3065AT, but the full range of its capabilities is available only when running on the HP 3065AT.

A tool such as this has some distinct advantages over a logic analyzer that is not part of the test system. Some of the advantages are:

- Convenience. No need to connect an external analyzer to the UUT.
- Triggering at a given test vector is easily accomplished.
- Data correlation between analyzer and test system is guaranteed by design.
- Waveform display can include combinations of expected and actual responses.
- Display uses the node names assigned by the user to the UUT.

The virtual logic analyzer offers two waveform display formats: graphical waveforms (Fig. 6) or logic states (Fig. 7). Both of these waveform display formats show three logic states: low, high, and undefined. The undefined state, shown as single-height, blacked-out areas or by the symbol X, means that the state at that point is either not specified or is between the low and high voltage levels. For either of these two modes the user has the option of displaying either the actual UUT response or failed information. In the failed option, the display shows the expected UUT nodal response and highlights where the received state differs from the expected state. This is shown as double-height, blacked-out areas on the waveform display and inverse video on the state display.

All of the above modes of operation work with either the HP 3065 or the HP 3065AT. There is a mode, however, that is exclusive to the HP 3065AT. This mode involves the use of the fast pattern capture capability of the FCC. Basically, the mode allows the user to observe nodes of interest at the event marker rate. This will provide time interval resolution as fine as 30 ns, the minimum event marker spacing. This mode of operation is invoked at a particular vector number. Once invoked, all subsequent vectors are expanded by the number of event markers contained by that vector (see Fig. 8).

The on-screen display capacity of the analyzer depends on the display mode selected. For the waveform mode, up to eight nodes appear on the screen at one time; for the state mode, 16. For either mode, information for up to 99 nodes can be collected at one time. Access to the nodes is controlled by display up and display down softkeys. The depth of data generated at any one time is 1024 vectors/event. It is possible to scroll through this information by using the display left and display right softkeys. When the user ventures outside of the generated window of information, a new window is generated automatically. Information is generated at a rate of 3 nodes/second, which results in a worst-case response time of 33 seconds.

Control of the analyzer is via a hierarchical command set that appears as a set of softkeys at the bottom of the display screen. These softkeys, like the test control softkeys, can also be executed from the display's command line or from a BT BASIC program.

Autolearn Module. Autolearn is the third major module of the GDM. Once a test has been debugged and is operational, the user can, as an alternative to the PCF, learn nodal states of the UUT and place them in that particular test's state file. In the most general sense, autolearn runs the functional test, collects responses (either patterns, cyclic redundancy check (CRC) words, or a combination of both) for each node and then stores those responses in the test's state file. These stored responses can then be used by the backtracer and virtual logic analyzer. While this is conceptually simple, there are two complicating factors—operational inconsistencies and uninitialized logic states. To help deal with these difficulties, the autolearn module includes a function that learns UUT nodal data a specified number of times, automatically compares previous data to new data, and then displays inconsistent nodal data. This function is called "confirm all nodes." While confirming nodal data, it does not update the state file. This gives the user the opportunity to decide if the operation is to be expected or if there is some problem with the test. A variation on the confirm all nodes function is the function called "mask all nodes." This function performs the same operation as the confirm all nodes function, but does update the state file. Any data that is inconsistent will be crossed out and, if CRC words are being learned, all unique CRC words for the inconsistent nodes will be added to the state file as valid responses.

The autolearn module has other commands that help reduce the time required to complete the UUT learning process. For nodes that continue to have inconsistent responses, single-node learning is available. This allows the user to debug the test and then have the system learn only those nodes that are required. Supporting this process are single-node learn and store operators.

Control of the autolearn module is via the softkey structure common to the rest of the GDM. It is a two-level structure with three second-level structures under the main autolearn control menu.

Test Execution

Test execution, the final phase of the board test process, is where production testing occurs. Typically, large quantities of boards are tested. Faults are detected, diagnosed, and then repaired. The main difference between the HP 3065 and HP 3065AT at this point is that the HP 3065AT will use the backtracer to diagnose failures detected during functional tests. Operation of the backtracer can be fully automatic or involve additional manual probing. The degree of manual interaction will depend on the coverage of the UUT's fixture and how the backtracing tree was structured during the functional test preparation phase.

Confirmation/Diagnostics

The confirmation/diagnostics (C/D) software for the HP 3065AT, like all of the HP 3065AT software, is a logical extension of the standard HP 3065 confirmation/diagnostic module. It is controlled via a menu with new routines added to verify the functionality of the FCC. In addition, two forms of autocalibration have been added as part of the C/D package. For users not requiring ultimate system timing accuracy, an automatic calibration of hybrid card receivers and manual backtracing probe can be performed using the confirmation/diagnostic fixture. However, maximum system performance is obtained through the use of a cable assembly that connects a reference driver directly to receivers. This eliminates the loading effects of the C/D fixture. Manual probing is aided by a template placed over the scanner field and software prompts of where to probe next. The total time required to calibrate a full scanner is on the order of 10 minutes; this needs to be done semiannually or whenever a scanner card is replaced.

Acknowledgments

The HP 3056AT is the result of a great deal of hard work by many individuals. On the hardware side, Bob McAuliffe and Dave Prater designed the vector address sequencer and timing/formatting system. John Siefers designed the hybrid circuit modules used for the formattable outputs and clock drivers. He also labored many long hours to reduce jitter in the phase-locked loop to acceptable levels. Finally, in his spare time, John also developed the fixture wiring termination schemes used by the HP 3065AT. Eddie Williamson designed the front printed circuit board and supervised the environmental test process. John Algiere developed the calibration models and specifications for the HP 3065AT and also performed RFI testing to be sure that the product complies with VDE level A standards. Jim Hayes did the mechanical design of the HP 3065AT and never once was in the critical path. Bill Kamagaki, our technician, produced reams of computerized schematics and had the pleasure of going into the environmental test chamber in his snowmobile suit to attend balky equipment.

On the software side, Mike Lassner, Kevin Keirn, and Ken Posse developed the extensions to the VCL compiler. Elaine May modified IPG-II to account for the nodal requirements of functional test. Rod Browen and Ken Parker developed the additional BT BASIC statements needed in a functional test environment. Mike Wahl developed the backtracing control software and Rich Marsh wrote the compiler for it. George Booth masterminded the graphical debug module with John Heumann developing its interface to the remainder of the software system. Chris Cain, the hardware group's window into the the land of software, wrote the low-level control routines and worked very hard to modify, extend, and create the confirmation/diagnostic and autocalibration routines used by the HP 3065AT. Finally, John Keller, the software group's project manager, provided many of the initial concepts for the HP 3065AT's architecture and also kept his group pointed in the right direction.

Reid Trimble and Mike Redig of production engineering helped to make the product more producible and Sam Cox labored long and hard to provide timely production test software for exercising the hardware in environmental test.

John Ferguson, Mike Dapron, Stig Oresjo, and Mike Horton of product marketing provided timely commentary on marketing issues and were our first users and product debuggers. Their patience and thoroughness in using the product's features in its early stages has added much to its overall friendliness and versatility.

Dick Barney, the program manager, smoothed the way as the project advanced through its development stages.

Reference

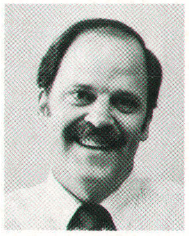
1. *Hewlett-Packard Journal*, Vol. 35, no. 10, October 1984, complete issue.

Authors

December 1987

4 Vector Signals

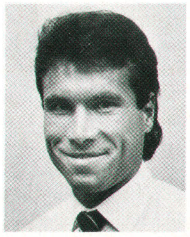
Allen P. Edwards



A Stanford University graduate (BSEE, MSEE 1971), Allen Edwards was born in Los Angeles, California. He started at HP in 1971 and has worked on the HP 8558B Spectrum Analyzer, the HP 436A and HP 438 Power Meters, the HP 8901A Modualtion Analyzer, the HP 8903A Audio Analyzer, the HP 8780A Vector Generator, and the HP 8980A Vector Analyzer. Currently an R&D section manager, his work has resulted in four patents related to power meters and frequency synthesizers. A member of the IEEE, Allen is married, has two children, and lives in Palo Alto, California. He has an Extra Class rating as an amateur radio operator and is interested in photography and home remodeling.

6 Vector Analyzer Hardware

Juan Grau



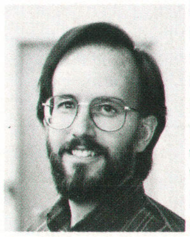
A native of Mexico, Juan Grau studied electrical engineering at Rice University (BSEE 1980) and Stanford University (MSEE 1983). He began work at HP in 1980 and has contributed to the HP 8901B modulation Analyzer, the HP 8902A Measuring Receiver, and the HP 8980A. He is named coinventor on a pending patent related to phase and quadrature correction. Juan lives in San Mateo, California, enjoys music, and has the reputation among his colleagues of being an excellent soccer player.

Andrew H. Naegeli

Andy Naegeli's work in the area of RF modulation analysis techniques has resulted in two patents (one pending) and two papers presented at HP's RF and microwave measurement symposiums. He started at HP in 1975 and worked on the design of the HP 8901A Modulation Analyzer. He then became project manager for the HP 8901B and the HP 8902A Measuring Receiver, and subsequently the HP 8980A. Andy is a member of the IEEE and holds the BSEE (1975) and MSEE (1979) degrees from Stanford University. Born in Pasadena, California, he is married, has two daughters, and lives in Moraga, California. He is very active in local church activities and enjoys camping with his family, hiking, bicycling, jogging, and music—especially singing and playing string bass.

17 Vector Analyzer Firmware

Stanley P. Woods

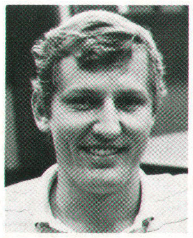


Stan Woods was born in Golfito, Costa Rica and attended the University of Washington, earning a BSEE degree in 1983. He then joined HP and has worked on the HP 8780A Vector Generator and the HP 8980A. Stan lives in Cupertino, California, is married, and has an infant daughter. Outside of work he enjoys fossil collecting, woodworking, hiking, geology, science fiction, and electronic projects.

Brian S. Messenger

Brian Messenger joined HP in 1981 and has worked on IC R&D and the RF circuits and firmware design for the HP 8980A. He is a member of the IEEE and studied electrical engineering at the University of California at Berkeley (BS 1982, MS 1984). Born in Sacramento, California, he now lives in Cupertino, California where he is an adult scout leader. His interests include backpacking, tennis, skiing, and sailing.

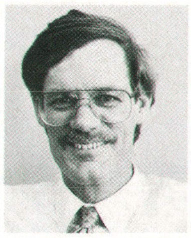
Peter H. Fisher



A native of Wiesbaden, Federal Republic of Germany, Peter Fisher received the Diplom Ingenieur in electrical engineering in 1982 from FH-Dieburg. He then studied digital electronics and software at the University of Iowa on a Fulbright scholarship in 1983. Peter joined HP in 1984 and contributed to the software design of the HP 8980A. Married, he lives in Cupertino, California and enjoys travelling, hang gliding, and windsurfing.

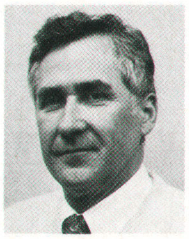
25 Vector Modulation

David R. Gildea



Dave Gildea joined HP in 1978 as an R&D engineer and was a project manager before he recently left the company to pursue his current interest in using global positioning satellites for surveying. He earned a BSEE degree in 1966 and an MSEE degree in 1971 from Stanford University. He has authored two papers on vector and digital modulation and is coinventor of a patent pending on the same subject. Dave is married, has two daughters, lives in Menlo Park, California, and is interested in marine navigation.

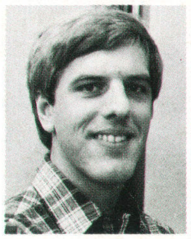
Donald R. Chambers



Don Chambers is an R&D project manager and has worked on microwave components, noise figure products, several microwave signal generators and synthesizers, and vector generators. Before joining HP in 1974, he worked for SRI performing microwave component and system research. He also served in the U.S. Air Force as a captain. An author of several articles related to his work at SRI, Don's work has resulted in a patent on couplers. Don was born in Portland, Oregon and is a graduate of Oregon State University (BSEE 1954 and MSEE 1959). He is a member of the IEEE, has a married son, and lives in Palo Alto, California. His current outside interest is renovating his house.

30 Vector Generator Firmware

James E. Jensen



Currently a firmware project manager, Jim Jensen has contributed to the firmware and digital design of the HP 438 Power Meter and the HP 8780A. He joined HP in 1980 after receiving a BS degree in electrical engineering and computer science from the University of California at Berkeley in 1979. Born in San Pablo, California, he now lives in San Jose, California. Jim is married and enjoys sailing and working on model airplanes.

Eric D. McHenry

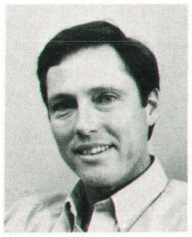
Author's biography appears elsewhere in this section.

34 Low-Noise Synthesizer

John C. Lovell

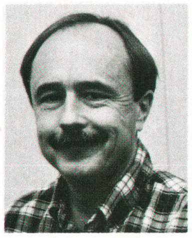
John Lovell designed the RF reference loop and time base sections of the low-noise synthesizer used in the HP 8780A. Born in Boulder, Colorado, he studied electrical engineering at the University of Tennessee (BSEE 1982) and then joined HP. Interested in feedback control systems and precision analog instrumentation, he is married and lives in Sunnyvale, California. John enjoys backpacking, reading, and playing the drums.

Thomas J. Carey



After receiving the BSEE (1971) and MSEE (1972) degrees from the University of Florida, Tim Carey served two years in the U.S. Army Signals Corps as a first lieutenant and worked as an R&D engineer for Motorola before joining HP in 1977. At HP he has worked on the HP 8642A and HP 8656B Signal Generators and was project manager for the low-noise synthesizer used in the HP 8780A. He is currently a product marketing engineer concerned with synthesizer market development. Born in Nashville, Tennessee, Tim lives in Mountain View, California and spends his leisure time breeding horses for dressage and show jumping, taking photographs, and endurance bicycling.

Thomas L. Grisell



Tom Grisell studied electrical engineering at San Jose State College (BSEE 1965) and Stanford University (MSEE 1969). He joined HP in 1965 and has contributed to the HP 140 Series of plug-in spectrum analyzers, the HP 8656A Signal Generator, and the low-noise synthesizer used in the HP 8780A. He also was project manager for the HP 8444A Tracking Generator. His work has resulted in one patent related to reference frequency generation for synthesizers. A native of San Francisco, California, he now lives in Palo Alto, California. Tom's interests include old radios, amateur radio, home computing, and fixing his old Datsun 280Z.

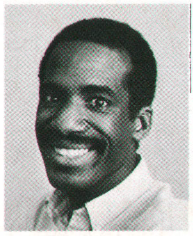
39 Baseband Circuits

Chung Y. Lau

Born in Hong Kong, Chung Lau studied electrical engineering at the University of California at Berkeley (BSEE 1975 and MSEE 1976). He joined HP in 1976 and besides his work on the HP 8780A, has also contributed to the design of the HP 8901 Modulation Analyzer and the HP 8903B Audio Analyzer, about which he wrote a article for the HP Journal in 1980. Chung is married, has two sons, and lives in Sunnyvale, California. He enjoys family activities and is a skilled bridge player (life master).

45 Wideband FM

Eric D. McHenry



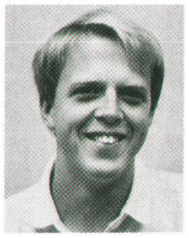
Eric McHenry worked on the firmware for the HP 8901B Modulation Analyzer and the firmware and FM circuits for the HP 8780A. He came to HP in 1980 after receiving a BS degree in electrical engineering from the Massachusetts Institute of Technology. Born in Seattle, Washington, he now lives in Sunnyvale, California with his wife and child. During his leisure time, he enjoys woodworking, camping, and hiking.

48 Modulator, Amplifier, Multiplier

Pedro A. Szente

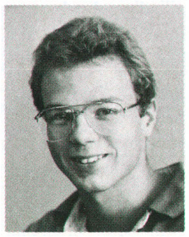
A native of Sao Paulo, Brazil, Pete Szente received a diploma in mechanical and electrical engineering from Escola Politécnica, Universidad de Sao Paulo and the MS and PhD degrees in electrical engineering from Stanford University. After serving as an associate professor at the Instituto Tecnológico de Aeronáutica in Brazil, he came to HP in 1967 and has contributed to the design of the HP 8445A Preselector, the HP 84801 Fiber Optics Power Sensor, the first converter for the HP 8558 Spectrum Analyzer, the HP 8470B series of detectors, and the multipliers for the HP 8780A. His work has resulted in three patents and he has coauthored an article on low-barrier Schottky-diode detectors. Pete lives in Los Altos, California and enjoys classical music and photography.

Eric B. Rodal



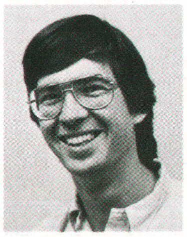
Eric Rodal was born in Baltimore, Maryland and went west to study electrical engineering at the University of California at San Diego (BSEE 1983). He then began work at HP and contributed to the design of the multiplier amplifier and the Option 64 output amplifier for the HP 8780A. Eric lives in Cupertino, California and enjoys skiing and racing Hobie catamarans.

Mark J. Woodward



Mark Woodward worked on the GaAs IC output amplifier for the HP 8780A. He is interested in RF and microwave analog design and studied at the University of Wisconsin at Madison (BSEE, BS-Math 1984) and Stanford University (MSEE 1987). Born in Columbus, Wisconsin, he now lives in Palo Alto, California and enjoys windsurfing, skiing, and traveling.

James D. McVey



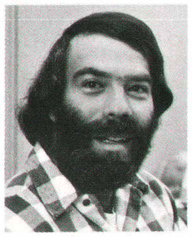
A graduate of Stanford University (BSEE 1984, MSEE 1985), Jim McVey started at HP in 1985 working on the HP 8780A and Option 64 for the HP 8780A. Now living in Palo Alto, California, he was born in Berkeley, California. His interests include microwave design, skiing, and windsurfing.

Wayne M. Kelly

A native of Oakland, California, Wayne Kelly attended San Jose State University, earning a BSEE degree in 1963 and an MSEE degree in 1965. He then worked for GTE Sylvania as a microwave design engineer before joining HP in 1975. At HP he designed a radar preamplifier used in F16 aircraft and the I-Q modulator and output mixer for the HP 8780A. He is coauthor of an article on GaAs FET amplifier compression measurements and coinventor of a patent related to balanced mixer design. He is a member of the IEEE, married, and the father of two daughters. Wayne lives in Los Altos, California where he indulges his passion for building massive concrete retaining walls and wood decks when not skiing.

53 Board Test System

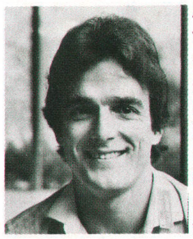
Michael E. Gravitz



The hardware project manager for the HP 3065AT, Mike Gravitz also served as project manager for the HP 3065's VPX processor. Mike has been with HP since 1974 and earned a BSEE degree from the University of Missouri at Rolla in 1969 and an MSEE degree from the University of Missouri at Columbia in 1973. In addition to his board test work, Mike designed in-house IC test equipment and contributed to the development of the HP 3456 Digital Voltmeter. Outside of work Mike enjoys bicycling, eating too much homemade ice cream, building high-performance motorcycles, whitewater kayaking, and refining his elaborate Lotus 1-2-3™ retirement model (retirement always seems to be five years away).

68 MPE XL

Alan J. Kondoff

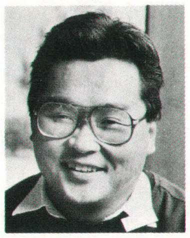


Alan Kondoff completed work for his BS degree in electrical and computer engineering from the University of Michigan in 1977 and has been with HP since 1976. He started as a field systems engineer and moved to R&D systems development in 1981. He contributed to the software I/O architecture for HP Precision Architecture and is now responsible for disc storage management, integrated transaction management, and concurrent backup. Alan is married, has two daughters, and likes bicycling, water skiing, and alpine skiing.

John R. Busch

John Busch has been with HP since 1976 and has been involved in commercial computer system research and development. He's currently a section manager with HP's Information Technology Group and is responsible for advanced operating system architecture, performance, and design. An alumnus of the University of California at Los Angeles, John holds BA and MA degrees in mathematics and a 1985 PhD degree in computer science. He is married, has two sons, and enjoys tennis and skiing.

Darryl Ouye



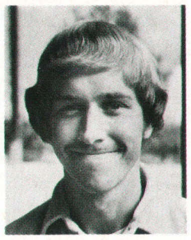
Since joining HP in 1979, Darryl Ouye has worked on commercial operating system design and development. A project manager in the Information Technology Group, he's responsible for operating system kernel design and development. He's a graduate of California State University at Chico and holds BS and MS degrees. His outside interests include wind surfing and guitar.

87 HP 3000 Emulation

Keith Keilman

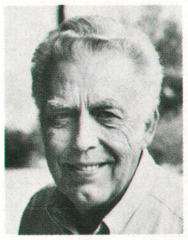
Keith Keilman has been with HP since 1979 and has contributed to the development of the HP 300 and other computers. He also helped develop the HP 3000 Emulator and Object Code Translator for HP Precision Architecture computers. A graduate of the University of California at Berkeley, he received his BSEE degree in 1978 and worked for IBM before joining HP.

Daniel J. Magenheimer



Dan Magenheimer has worked on the HP Precision Architecture project since its inception. He has contributed to the design of the instruction set, the simulator and remote debugger, an object code emulator/compiler, the HP-UX system linker, and millicode. He's now a project manager for FORTRAN/HP-UX in HP's Computer Language Lab. Born in Milwaukee, Wisconsin, he attended the University of California at Santa Barbara, completing work for a BA degree in computer science in 1981. He earned his MSEE degree from Stanford University in 1985. Dan and his wife and daughter live in Union City, California. He sings in the choir at his church and enjoys playing volleyball and softball.

Arndt B. Bergh



The emulator and translator for HP Precision Architecture are the latest in a long line of responsibilities for Arne Bergh, who came to HP in 1956. His projects have included instruments, magnetic devices, memories, and computers, notably the hardware design of the first HP 3000 Computer. He's named as inventor on six patents related to computer architecture, memories, and magnetic devices. A native of Canada, he received his AB degree in chemistry from St. Olaf College in 1947 and his MS degree in physics from the University of Minnesota in 1950. He also served for four years in the U.S. Army. A resident of Los Altos Hills, California, Arne is married and has three children. His favorite recreational activity is sailing on San Francisco Bay.

James A. Miller

Jim Miller received a BS degree in physics from Santa Clara University in 1970 and joined HP the same year. He has contributed to many HP 1000, HP 300, and HP 3000 software projects, most recently serving as technical lead and then project manager for the HP 3000 Emulator and Object Code Translator for HP Precision Architecture. He interrupted his HP career twice to study social psychology at New York University and received an MS degree in 1974. Born in Burbank, California, he is married, has two children, lives in San Jose, and enjoys backpacking and cross-country skiing.

MPE XL: The Operating System for HP's Next Generation of Commercial Computer Systems

MPE XL is a new commercial operating system developed for HP Precision Architecture computer systems. It provides fundamental advances in operating system technology and helps users migrate to the new systems by providing maximum compatibility with existing systems.

by John R. Busch, Alan J. Kondoff, and Darryl Ouye

THE HP PRECISION ARCHITECTURE development program has resulted in a new base for HP's computer systems for the following decades, providing advances in technology and capability along with a smooth migration path for current customers. The foundations of this program resulted from significant technological advances and symbiotic relationships among processor architecture, compiler, operating system, data base, and data communications components. These basic components provide a high degree of compatibility with existing commercial system interfaces, thereby permitting high leverage of existing commercial subsystems, applications, and utilities, which results in complete and compatible system offerings at first release.

The role of the MPE XL commercial operating system in achieving the objectives of the HP Precision Architecture program falls into three primary areas: providing fundamental advances in operating system technology, participating in a synergistic manner with the advances in the other underlying technologies so that the entire product realizes the benefits, and participating in customer base migration by providing maximum compatibility in all areas relating to operating system services for supported environments.

Currently, two HP Precision Architecture implementations run the MPE XL operating system. These are the HP 3000 Series 930[1] and 950[2] Computers.

In terms of fundamental technology advances, MPE XL is designed to provide a base for evolutionary increases in performance, availability, and features. Performance is intended to scale directly with respect to processor speed, main memory availability, and workload. Highly efficient support for mapped disc files, allowing megabytes of user data to be accessed at memory access speeds, is one of the key technologies employed.

For increased system availability, support of the atomic transaction model, with its inherent recovery properties and minimized disc posting (write) requirements, is integrated into the design of MPE XL. This provides logical data consistency and recoverability in highly concurrent workloads, resulting in increased data integrity and system availability with increased overall performance.

With respect to system synergism, the component technologies provide services that exploit one another in desirable ways. The leading-edge processor performance for a given implementation technology is provided by HP Precision Architecture.[3,4,5] The compiler's optimization exploits the architecture to maximize the effective processor speed for a given code sequence.[6] The operating system provides data access that scales with processor speed. The ALLBASE/XL data base provides highly concurrent access to shared data.[7] The cooperation of component technologies translates the processor price/performance leadership to the system level.

Special use of the architecture's addressing and protection mechanisms are made by MPE XL so that data access speed is maximized using the hardware translation and protection mechanisms. Compiler services are provided so that the specialized instructions can be accessed and frequently used code sequences entered without losing optimization context. System services are provided by MPE XL to data bases for transaction management and control over disc/main memory traffic so that minimal paging is incurred and minimal waiting is required for the paging that does occur. Standard building blocks such as port (interprocess communication), table, and semaphore management are provided for structuring data communications and other subsystems in a manner that is independent of processor configurations.

With respect to migrating the existing installed base to the new architecture, MPE XL provides full compatibility at several levels. The syntax and semantics of the programmatic (intrinsic) and command interfaces of MPE V are supported and extended. Not only are programs compiled with the MPE XL native compilers supported, but object level programs compiled with MPE V-based compilers are also supported via emulation of the MPE V instruction set or translation of the MPE V-based instruction set into the native HP Precision Architecture instruction set.[8] This level of compatibility allows the full range of familiar subsystems, tools, and applications to be available through the MPE XL environment, even if the source code is unavailable.

The following sections describe MPE XL's use of HP

Precision Architecture and its compilers, the structure of the operating system, the design of the kernel, the I/O and data management portions of the operating system, the tools and utilities provided, and the support of the user environment.

HP Precision Architecture and MPE XL

As described in reference 4, HP Precision Architecture defines 32 general-purpose registers and eight space registers, and assigns special purposes for some of these registers. MPE XL also assigns special uses for some of the registers. These conventions are to support an architected program structure. The space register assignment for MPE XL is as follows:

SR 0. Procedure call linkage (hardware architecture specified).
SR 1 to SR 3. Mapped file addressing.
SR 4. Access to program literals.
SR 5. Process local data (stack, heap, etc.).
SR 6. System code and data.
SR 7. System code and data.

SR 1 to SR 3 are used primarily for accessing permanent disc files mapped into virtual space, and provide 4G bytes of addressability. In MPE XL, all open disc files are mapped into virtual memory. The user has a choice between accessing files through the MPE XL file system interfaces or directly accessing them (using pointer or array type access) as regular data. Since these registers are used for long-pointer addressing,[4] initial releases of MPE XL will provide 2G bytes (out of 4G bytes possible) of direct addressing to each file. User mapped files will be discussed in a following section.

SR 1 is also used to return values from a function call. It is used in conjunction with GR 28 and GR 29, as discussed later.

HP Precision Architecture does not provide addressing relative to the program counter. Therefore, a space register, SR 4, is assigned to track the currently executing code space for access to literals. A single program object has a size limit of 1G bytes because of short-pointer addressing rules.[4] The value in this register is changed whenever an external procedure call (intermodule) is made. There is no practical limit on the number of modules within a program unit, and therefore, no practical limit on program size.

SR 5 is used to point to the process local data area. Included in this area are the stack, heap, process local global data, library globals, etc. The total size of this data area is 1G bytes. The value in this space register is changed on each process switch and external interrupt.

SR 6 and SR 7 are used as system global addressing registers. Contained in this space are all of the operating system code and data. The size of the system space is limited to 2G bytes. The values of these registers never change once the system has been booted.

In addition to space register assignments, the program architecture also designates certain general registers for specific uses:

GR 2. Return pointer. This register contains the code offset portion of the return address when a procedure call is executed.
GR 23 to GR 26. Argument registers. These registers are used to cache the first four parameters when calling procedures.
GR 27. Data pointer. This register is used to point to the process global data area in SR 5 space.
GR 28 and GR 29. Function return. These registers are used to return values from functions.
GR 30. Stack pointer. This register is used to point to the top of the process stack in SR 5 space.

Mapped Files

A special type of virtual memory object is the mapped file. A mapped file is a disc file that is mapped directly into the virtual address space. All disc files in MPE XL are mapped, and are managed by the same standard kernel mechanisms as regular data and code objects.

MPE XL exploits HP Precision Architecture's large address space, large physical memories, and protection mechanisms to manage large volumes of disc file data efficiently in main memory. Megabytes of user disc file data can be present in main memory and accessed at hardware speeds through standard virtual-to-real translation mechanisms. This eliminates buffer location and a substantial number of extra disc I/O operations for repeated file reference. This method of caching disc files in primary memory is similar in principle to disc caching in MPE V,[9] but the overhead is significantly lower for mapped files since the translation hardware allows location of and access to pages of mapped files with no incremental performance penalty.

MPE XL's memory manager can now arbitrate main memory use based on total system loading, and not be constrained by an application's local view of buffering or data demand. When only one application is active on the system, all main memory can contain pages of the disc files or data bases it is accessing. As main memory contention occurs, the memory manager determines the most equitable allocation of memory as determined by global metrics. This yields the highest system-level throughput possible, and responds dynamically to varying configurations of main memory size and CPU speed.

User mapped files is a file access mode whereby disc file access via system intrinsics can be bypassed. Users can access disc file data via normal machine LOAD and STORE instructions. This is discussed further in the "MPE XL Data Management" section of this article.

In both file system and user mapped access, disc file buffering is performed by memory management and is integrated with normal page fault handling. Optimizations are performed to speed up sequential access, such as fetching or posting multiple pages at a time.

Protection

Protection in HP Precision Architecture is implemented by a combination of privilege levels, access rights, and protection identifiers. These facilities are used by MPE XL to provide a secure system that prevents unwarranted access to user and system objects.

Privilege Levels. Programs execute at one of four privilege levels. Privilege levels 0 and 1 are reserved for system code. Level 2 is reserved for privileged subsystems and privileged third-party software. User code executes at level 3 (least privileged).

Each procedure in MPE XL specifies the privilege level required to call it and the level at which it is to execute. Changes in privilege levels are implemented with promotion stubs, which are short code sequences created during link time. The linker inserts a branch to the stub, which causes a privilege level change, and then branches to the target procedure.

Data pages are also assigned protection levels. Process local data is assigned privilege level 3. Files accessed through the MPE XL file system (not user mapped) are assigned level 1 privilege. All system and subsystem data is assigned privilege level 2.

Access Rights. Each physical page has a set of access rights that specify the type of access allowed at each privilege level. These may be a combination of read, write, execute, and gateway (privilege level change) rights.

Protection Identifiers. To protect access to addresses and allow flexible, low-overhead data sharing by callers of the same privilege level, HP Precision Architecture provides a protection identifier (PID). Each physical page may have an associated PID, which a process must have to access that page. MPE XL maintains a list of PIDs for each process, allowing them access to different types of memory objects.

Four special hardware registers act as a PID cache. When memory is accessed, the hardware compares the value of the page's assigned PID with the values contained in the four PID registers. If a match is found, the access is allowed. If the page's PID is not found in one of the PID registers, a trap is generated. The PID trap handler consults the PID list maintained for the process. If the page's PID is contained in the list, the PID is loaded into one of the PID registers and the process continues. Otherwise, the process is aborted.

Primitives

Because HP Precision Architecture is a RISC-type architecture,[4] some of the functions implemented in the instruction set of MPE V-based systems are not included in the HP Precision Architecture instruction set. They have been implemented by MPE XL software and compilers. The advantage of implementing these infrequently used but widely leveraged functions in software is that it allows the hardware to optimize its path lengths (providing a higher processing rate) and allows the functions to be modified easily.

This scheme does require that the overhead in calling these types of routines be small in comparison with "normal" procedures. To reduce overhead, primitive functions in MPE XL are implemented by a combination of assembly coded routines, predefined compiler functions, in-line routines, and routines with fixed addresses.

Assembly Routines. High-level language compilers are used to implement modern operating systems because of the ease of developing and maintaining code. However, certain operating system functions cannot be supported in a high-level language unless machine related functions are placed into the compiler or an escape to assembly code is allowed. The former has a heavy implementation cost for the compiler and the latter can result in unsupportable code if abused.

The policy in MPE XL has been to support machine related functions in assembly code without support from the compiler, that is, there is no escape to assembly. However, assembly routines may be declared as externals in a high-level module and calls to assembler routines are handled in the same way as calls to high-level language procedures.

MPE XL use of assembly coded routines is minimized to very low-level functions which, in general, provide the interface to the hardware resources. Such things as first-level interrupt handling, cache manipulation, interrupt environment setup, etc., are handled by assembly procedures.

Predefined Functions. While the compiler does not directly support assembly, it does support several ways in which use of high-level language procedures can be optimized. The predefined function is a routine that has been implemented by the compiler as part of its feature set, that is, it has become a part of the language (although it may require special capabilities to use these functions).

The reason for supporting predefines is to minimize procedure calling overhead for system functions that are performance sensitive. In certain critical paths, the overhead of calling procedures can be a major factor. Predefined functions are treated like other types of statements supported by the compiler; their code sequences are simply inserted into the current instruction path. There is no procedure call overhead, and the inserted instructions are within the scope of the compiler optimizer.

Care is taken to ensure that the functions implemented as predefines are truly required for performance and that they will remain stable (i.e., they will not change over the life of the compiler).

In-Line Code. Since predefined functions are limited in scope, the HP Pascal/XL compiler supports an in-line procedure option, which provides the same level of performance as predefines. However, since an in-line procedure is not specified as a part of the language, it can be changed more easily. The operating system developers can implement as many or as few in-line procedures as required. A procedure can be implemented as an in-line routine or a normally called procedure as performance tuning continues over the life of the system. This gives developers the benefit of a minimal call overhead with the flexibility to change what is implemented in-line.

In-line procedures are also within the scope of the compiler optimizer, giving further efficiencies over abbreviated calling sequences.

Millicode. One problem with predefines and in-line procedures is that since they are replicated in each instruction path in which they are used, they take up more space than a regular procedure, which exists only in one place.

MPE XL has solved this problem by supporting millicode routines.[6,10] This type of routine is implemented in a high-level language and is treated by the compilers as a special type of called routine. It is special in that most of the procedure call overhead is eliminated by restrictions on how millicode may be used. For example, most of the operations performed in millicode are register intensive, using no storage (main memory) for work areas. In effect, the operation of millicode routines is handled much like in-line code.

Stubs. In MPE XL, code stubs (short instruction sequences) are used for various purposes. HP Precision Architecture provides two types of branch instructions, one with a short addressing range and another with a long addressing range. If the target of a short branch cannot be reached by the instruction (because of placement by the linker of the caller and callee in the memory image) the linker creates a long-branch stub, which performs the actual branch to the target address. The linker inserts a branch to the stub in the caller's instruction stream.

Other types of stubs are provided for privilege promotion, parameter relocation, procedure import and export, etc.

Procedure Calls

In the MPE V-based architecture, a procedure call (PCAL) instruction performs stack and register manipulation on procedure calls. In HP Precision Architecture systems, this function is provided by four compiler generated code sequences: call, entry, exit, and return.[6]

All of these code sequences are automatically generated by the compilers when procedures are invoked. There is no explicit control required by the programmer. Also, compilers can abbreviate these sequences in special cases, such as leaf procedures (no external calls), to optimize the calling sequence further through elimination of unnecessary state saves.

System Software Architecture

The MPE XL commercial system software architecture refers to the functional partition, standards, and durable interfaces of operating system components, subsystems, tools, and applications executing on MPE XL HP Precision Architecture implementations.

Fig. 1 depicts the application, subsystem, and operating system structure in an MPE XL commercial system. Fig. 2 displays the functional partitioning of the service layers. Operating system programmatic and command interfaces are implemented directly on MPE XL common services, providing efficient support of the user environment.

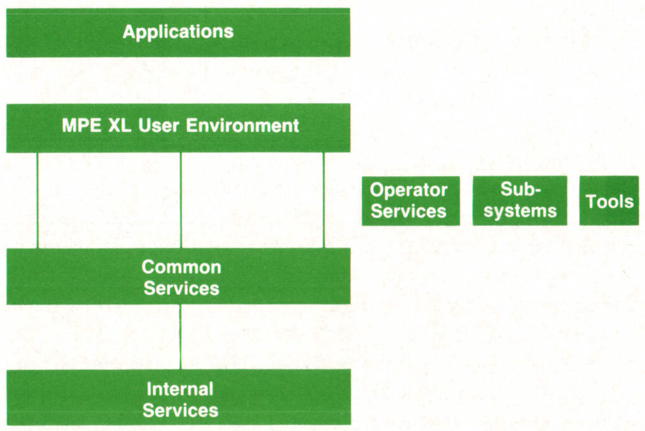


Fig. 1. *MPE XL commercial operating system structure.*

The MPE XL Kernel

The MPE XL kernel supports the sharing of the basic hardware resources such as main and secondary storage, central processor, and virtual space. It also provides support for interprocess communications, program loading, logical devices, traps, system tables, environment switch, timers, etc.

The MPE XL kernel was designed with three main objectives: performance, structure, and extensibility.

High Performance. System performance must scale with processor performance. By designing integrated components that cooperate in managing a global performance strategy, MPE XL is able to give optimized performance across diverse hardware configurations and workloads.

Flexible Structure. The structure of the operating system must allow for performance, flexibility, and maintainability. These objectives are achieved by means of standard mechanisms, system primitives, and modular component design.

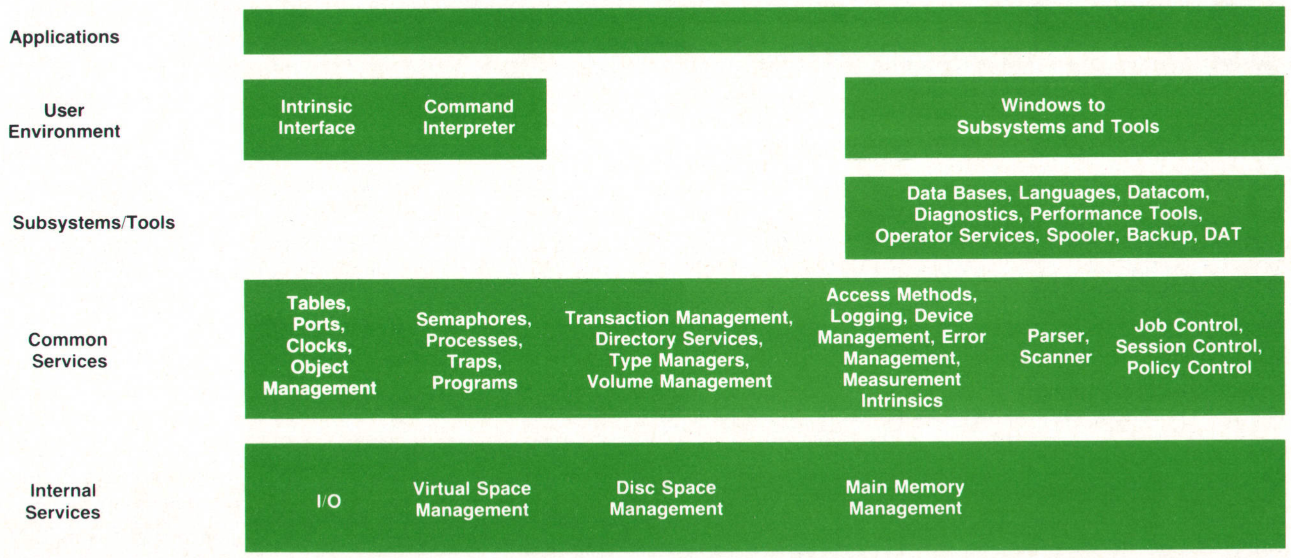

Applications
User Environment
Intrinsic Interface
Command Interpreter
Windows to Subsystems and Tools
Subsystems/Tools
Data Bases, Languages, Datacom, Diagnostics, Performance Tools, Operator Services, Spooler, Backup, DAT
Common Services
Tables, Ports, Clocks, Object Management
Semaphores, Processes, Traps, Programs
Transaction Management, Directory Services, Type Managers, Volume Management
Access Methods, Logging, Device Management, Error Management, Measurement Intrinsics
Parser, Scanner
Job Control, Session Control, Policy Control
Internal Services
I/O
Virtual Space Management
Disc Space Management
Main Memory Management

Fig. 2. *Functional partitioning of MPE XL service layers.*

Heavy use of standard mechanisms such as semaphore locking and interprocess communication considerably reduced the development and maintenance cost of MPE XL. This is because each mechanism was written by a single engineer, instead of each engineer having to write a specialized, ad hoc version. This also results in higher reliability of the mechanism and allows any performance enhancements to be shared easily among all components of the system. Standard mechanisms also provide uniform data structures, which are exploited by dump analysis and debug utilities.

Through procedures called *system primitives*, the kernel isolates architectural dependencies from the rest of the operating system. These procedures create an environment for handling traps, I/O completion interrupts, and the dispatching of processes. Other examples of system primitives are semaphore locking, dispatcher entry and exit, TLB (translation lookaside buffer) fault handling, and external interrupt disable/enable.

Although the external interfaces of the various parts of the kernel have been carefully integrated, their internal designs are independent of one another. Thus it is easy to change, for example, the memory manager's algorithm without having to modify other parts of the system. Also, because of a careful separation between modules that implement a specific policy and those that provide basic mechanisms, it is a relatively simple task to change the behavior of the system without changing basic services.

Extensibility. MPE XL is designed so that extensions to the basic function set can be added easily. This is a result of its use of standard mechanisms and modular design. By taking advantage of the features of HP Precision Architecture, MPE XL will be able to support a wide range of configurations.

New software functions are also easily supportable. For example, new file access methods can be provided by implementing a new file type manager (see the "MPE XL Data Management: The File System" section of this paper). The basic file manipulation services remain unchanged.

Standard Tables

Because much of system programming involves getting and returning table entries, standard tables provide the largest gain in leveraging code. Tables have been designed to meet the requirements of data communications as well as the internal requirements of ports, virtual space management, the memory manager, and other MPE XL components.

Tables can be created with any number of entries. The entries can either have the same fixed size or the size of each entry can be set each time one is obtained. The table can be main memory resident or, more likely, swappable. In either case the table can be initially small and be allowed to grow as demand for its entries grows. This eliminates the need for system managers to configure table sizes in MPE XL. Tables will scale their sizes to system configuration and demand.

Usually, obtaining a table entry is simply a matter of locking out other accesses to the table, taking the first entry off the table's free list, and unlocking the table. However when the table's free list is empty, one of four things will happen. If possible, an attempt is made to expand the table. If it is a critical request and the table has a pool of emergency entries, then the requester is given an emergency entry. If the caller can wait, then the process is suspended until an entry is returned by another process. Otherwise, the caller is returned "table empty" status.

Ports

Ports provide a means for processes to send and receive messages, which is classically referred to as interprocess communication (IPC). MPE XL's contribution above and beyond this classic method of synchronization is the ability to send messages at (approximately) procedure call speeds without the overhead of a process context switch. The basic mechanism is as follows. A process calls the system primitive SEND MSG, specifying a message and a target port. The message is copied into system memory (really a system table entry) and then queued at the end of the port's message queue. If the port was empty and the port's process is waiting on a message from this port, the port's process is awakened and given the message. Otherwise the message is left queued on the port, to be read later by the port process.

Processes can create any number of ports and selectively wait for a message on any subset of them.

The I/O system has special requirements for ports. The I/O software mirrors the I/O hardware in that there is a software manager for each hardware component: the I/O channel, the device adapter, and the I/O device itself. Each I/O request must pass through these three managers for request initiation and for the completion interrupt.

Because of the complex, asynchronous nature of I/O requests, the ports system is a natural means of communicating between managers. However, the overhead of six process dispatches per I/O request would be intolerable, so instead of processes, the managers are implemented as HP Pascal/XL procedures. Each time a message is sent to an empty port, the port's manager is called and the message is passed to it as a parameter.

When the manager has finished with the message it simply returns control to the port software. If there is another message queued on the port, it is dequeued and the manager is called again. This process is repeated until there are no more messages queued on the manager's port.

The procedure-like nature of this type of port server provides asynchronous communications common to message-based systems with the performance of procedure calling. When a server is inactive and there are no other messages pending, the next message is passed to the server procedure in a manner similar to the normal procedure call mechanism. This eliminates the overhead of queuing and dequeuing messages when there is no need to wait. Ports also provide a mechanism for implementing critical sections of code. Port servers are not activated if they are already busy.

Virtual Space Management

Virtual space management (VSM) provides a fundamental mechanism for translating virtual addresses to disc addresses. This mechanism is used for disc file, transient, program, and operating system objects. Virtual memory is allocated by calling the VSM procedure, CREATE OBJECT.

Callers can specify whether the object is to be allocated its own space or to be packed with other affiliated objects, such as those associated with a particular process. Packing of objects allows the memory manager to bring in a process's context with a few disc reads of contiguous disc regions, instead of trapping on each code and data page read.

Although disc space can be reserved when the object is created, such as the initial allocation for a new file, a virtual space does not have to be bound to disc space until a page is referenced. This makes it possible to create large, sparsely populated objects that occupy a minimum amount of disc space. As in MPE V, disc space is allocated in multipage chunks called extents. In MPE V, each extent of a file is the same size, and there is a limit of 32 extents per file. These restrictions have been eliminated in MPE XL. The size of each extent is independently calculated based on the object's use, current size, and growth rate, and on the pool of available disc space. VSM also places no practical limit on the number of extents an object may have.

VSM provides two basic levels of translation structures, one that is transient and one that is memory resident. The transient structures are swappable, that is, they do not have to remain in memory. Only the most recently used portions of the transient translation structures are in memory at any given time. These swappable structures consist of virtual space object descriptors (VSOD), which are kept in a set of b-trees, along with their associated disc extent b-trees (Fig. 3).

VSM takes advantage of automatic table expansion so that the number of b-trees maintained by VSM is scalable. An object can be found by hashing to its appropriate b-tree and traversing until the appropriate node is found. Each object descriptor has another b-tree associated with it, called an extent b-tree, which is independently swappable from the VSOD to which it belongs. The extent b-tree maintains virtual-page-to-disc translation information, along with information about each page's state (virgin, access rights, etc.).

The virtual page cache, VSM's memory resident translation structure, provides the memory manager a fast virtual-to-disc address mapping, along with information about an object's locality, initialization character, class (type of object), and access rights.

VSM also provides information to the rest of MPE XL about the virtual location of certain known structures used by MPE XL subsystems. Each structure is identified by a fixed number, called a *known system object*. This allows MPE XL subsystems to locate the requested structure's virtual address quickly by means of a constant identifier.

Memory Manager

The MPE XL memory manager works to support the highest levels of multiprogramming by implementing concurrent memory fetch and replacement activities.

It cooperates with processor and secondary storage managers to optimize system performance for the configuration's current workload. Mapped files alleviate the need for system buffers and effectively increase the file's buffering capacity to the size of available main memory when the system is lightly loaded, or to an equitable portion of

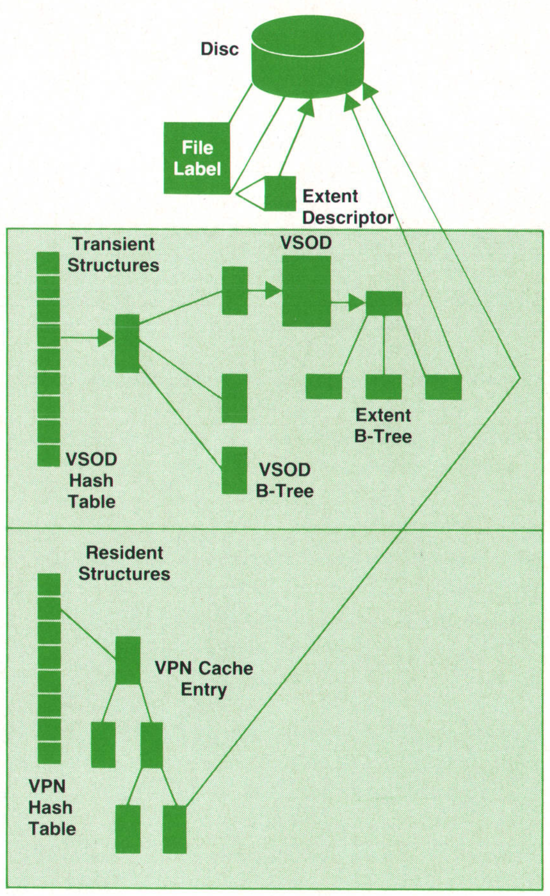


Fig. 3. *This is an example snapshot of an open file mapped into memory with virtual space management. The file label and its associated extent descriptor provide a permanent, disc resident image of a file. The virtual space object descriptors (VSODs) provide a set of swappable translation structures consisting of active spaces on the system (there are 2^{32} available spaces). These spaces include file, transient, program, and operating system spaces. The most active object descriptors remain in main memory and are subject to the same memory pressure algorithms as the objects they describe. An extent b-tree provides a swappable translation structure associated with each object descriptor to map disc addresses into virtual page numbers. The virtual page (VPN) cache provides a quick disc address translation for ranges of virtual pages (virtual addresses) that are in memory or have recently been there.*

memory when the load is increased. A disc file's occupancy of memory is not managed by the file system, because it has no knowledge of the global state of main memory.

The memory manager also provides a number of services that allow subsystems limited control over paging traffic.

Locality control procedures are used by the file system and data base to instruct the memory manager to prefetch and flush portions of a file to disc. For example, when a file is opened as read-only with sequential access, the file system will issue prefetch requests for data just ahead of the user's current position in the file. Flush requests are issued after the data has been moved from the mapped file to the user's area. In the case of a read-only file, the flush does not have to write the pages back to disc, since they were not modified. Instead, it just frees the main memory pages.

The memory manager also provides a means to add and delete areas of virtual memory in a process's minimum locality. The process will not be given the CPU until all of its minimum locality has been swapped into main memory. This is done to avoid immediate page faulting when a process is given control of the processor.

Disc posting dependency queues are provided by the kernel, and are used by the integrated transaction manager and data base for explicit control of the order of posting (writing) pages to disc. Processes can continue execution in an uninterrupted manner while asynchronous disc posts occur, rendezvousing with the final disc post completion to commit the transaction. This enables them to recover from crashes with consistent data on disc.

Process Management

Like many other services provided by MPE XL, process management supports a generalized process structure while maintaining compatibility with MPE V.

In MPE XL, a process can obtain more information about other processes than was possible in MPE V. The set of processes about which a given process can obtain information is not limited to that process's parent and children. A process can obtain information about any process in its tree, even processes that have terminated. Processes (and process trees) can also be adopted by other process structures.

Process notification, based on process signals, is a process management facility that will notify interested processes when a particular process is created, activated, suspended, or terminated. Processes can declare themselves to be dependent on a set of other processes. This means that when a process terminates for any reason, all processes that are dependent on it are also terminated.

Switch

MPE XL supports a 16-bit MPE V compatible environment, called *compatibility mode*, concurrent with a native 32-bit environment called *native mode*. To provide access to functions in either environment, the *switch* mechanism allows procedures in native mode to call compatibility mode procedures and vice versa.

MPE XL operates partly in compatibility mode and partly in native mode. Calls to MPE XL services may result in switches. These switch calls have been highly optimized and do not contribute significantly to the path length of system services.

In each switch call, the target procedure must be specified. Within MPE XL, there is a large set of procedures with hard-coded procedure identifiers which are known to the switch facility. Switching to these procedures can be performed quickly by specifying the procedure's indentifier. However, it is not practical to assign procedure identifiers to target procedures that are outside of MPE XL's domain. In their case, the procedure's ASCII name and library search path are specified. When the switch facility is called for the first time for the target procedure, it performs a dynamic load of the procedure. Information about the procedure is then saved in a process local table so that subsequent switch calls to the same procedure can proceed with a quick table lookup.

Passing parameters from compatibility mode to native mode is efficient, since the 64-bit address space can completely envelop an MPE V 16-bit address. This is not true when switching from native to compatibility mode, since MPE V's 16-bit addressing cannot reach all HP Precision addresses. The technique employed by the switch mechanism depends on the type of parameter, its length, and the privilege level of the target procedure. Value parameters and small reference parameters are copied to the compatibility mode stack. If the target procedure is privileged and can run in split stack mode, and if the reference parameter length exceeds a threshold, then an MPE V extra data segment is created for the duration of the switch call. This data segment exactly encapsulates the native mode reference parameter. The target compatibility mode procedure is then called with its DB register pointing to the new (temporary) extra data segment.

Clocks

Reading system clocks merely means reading HP Precision Architecture's internal clock registers, scaled appropriately to provide the correct units and data representation.

Time-outs are provided for MPE XL internals, subsystems, and user intrinsics. The time-out interface is simple; the caller supplies the time-out interval, a port to send a message to when the time-out expires, and the contents of the message to send. As in MPE V, pending time-outs that are no longer needed can be aborted.

High-Level Input/Output

High-level I/O (HLIO) is a set of services that interface and schedule kernel and MPE XL data management I/O requests to the MPE XL LLIO (low-level I/O) system, discussed later. HLIO performs the necessary I/O preparation, request transformation, and request scheduling before handing a request to the LLIO system.

I/O preparation envolves allocation of resources required to perform the request, ensuring that user buffer heads and tails are aligned on architected cache line boundaries (HP Precision Architecture caches are managed by software), and requesting MPE XL's memory manager to *freeze* or pin down the pages in main memory for the following DMA (direct memory access) by I/O hardware, if necessary.

Request transformation involves the translation of the external environment's transfer request into corresponding MPE XL LLIO transfer functions. For example, a device specific I/O request from the MPE XL environment may not correspond directly to a function on the new I/O hardware attached to HP Precision Architecture. It is HLIO's responsibility to transform the external environment's request into MPE XL LLIO requests that satisfy the semantics of the original request.

Request scheduling by HLIO envolves the application of system policies on arriving requests to best satisfy global system needs before releasing them to the MPE XL LLIO system. Disc request prioritization and queue depth (number of outstanding requests) on active channel programs for a device are among the policy decisions applied.

MPE XL Data Management: The File System

Two primary objectives needed to be satisfied through the implementation of a new, high-level data management subsystem in MPE XL. These were to exploit performance advantages available through MPE XL on HP Precision Architecture, and to provide an extensible base that can track the functional and availability evolution of MPE XL.

The most significant advantage of HP Precision Architecture exploited by the data management subsystem is the architecture's extremely large virtual address space and main memories. All accesses to secondary store (disc) are performed via machine LOAD and STORE instructions to regions (disc files) mapped into the machine's virtual address space. This eliminates the need for explicit disc file buffer management and location. Buffer location is performed by HP Precision Architecture hardware, while management functions are handled by standard MPE XL memory management facilities.

By providing a byte-array abstraction of secondary store to the disc access methods, a simple, focused code algorithm can be designed to access each type of disc file. To an access method, a disc file appears as a 2G-byte (in the initial MPE XL release) array of bytes onto which it can impose its specific organizational and semantic rules.

A fallout of the single-level store nature of disc file access in MPE XL is the notion of a user mapped file (Fig. 4). Certain access methods allow users to modify the contents of a disc file directly without going through the access methods. A virtual address can be passed back to the application (user). Using this address, accesses to the disc file can be performed with a simple, dereferenced pointer. The application now has access to a disc file at LOAD/STORE machine instruction speed without incurring the additional overhead of file system access methods.

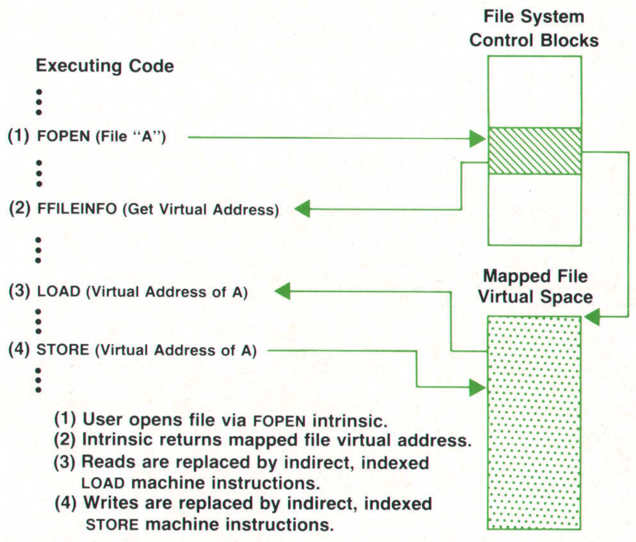
Since user mapped files are under control of the MPE XL file system, all security, protection, and rendezvous mechanisms are defined and enforced as for conventionally accessed files. This allows applications to make use of user mapped files as shared, named common storage between programs, as a basis for their own specific disc access method, or as retained working storage. Mapped files offer applications the opportunity to achieve high levels of performance through conventional programming practices.

Integrated Mechanisms

Key secondary storage management mechanisms have been integrated with disc data management to effect optimal system resource utilization under varying load conditions.

The behavior and reference pattern of disc accesses has been extensively measured and modeled for a wide range of application mixes on commercial HP systems. Through these analyses and experiences with internal disc caching on MPE V-based computer systems, a set of relevant metrics and mechanisms emerged. Disc prefetch, posting, extent allocation, and placement were among the mechanisms selected for integration.

When a page fault against a disc file is detected or projected to occur, a strategy routine is invoked to determine how much data should actually be brought in, or prefetched, from disc. The goal of this strategy routine is to provide enough data to keep the application running without unnecessarily overcommitting disc or main memory resources. In making its decision, the strategy routine consults and adjusts a set of metrics.



File System Control Blocks
Executing Code
(1) FOPEN (File "A")
(2) FFILEINFO (Get Virtual Address)
(3) LOAD (Virtual Address of A)
(4) STORE (Virtual Address of A)
Mapped File Virtual Space
(1) User opens file via FOPEN intrinsic.
(2) Intrinsic returns mapped file virtual address.
(3) Reads are replaced by indirect, indexed LOAD machine instructions.
(4) Writes are replaced by indirect, indexed STORE machine instructions.

Fig. 4. *User mapped file.*

The prefetch strategy routine views the pattern of both global and local references against a disc file, and makes a heuristic determination of how much data should be prefetched from disc. The algorithm used is deliberately simple. Disc file metrics consulted indicate file consumption rate, access pattern (sequential or random), fault rate, access method hints, and global main memory availability. Through this localized file view and global feedback metrics from the prefetch mechanism, the file system can adjust to varying application demand, CPU availability, and main memory availability to provide the best global result in a dynamic environment.

The posting (writing modified data to disc) strategy routine uses the same set of metrics as does the prefetch routine, but towards a slightly different set of goals. The posting routines judiciously manage main memory occupancy, file consistency, and disc utilization.

Main memory occupancy of file pages is primarily the responsibility of the memory manager. The file system posting routines assist the memory manager's job in several ways. When sequential access is performed to a file, the posting strategy routine explicitly requests the memory manager to post large, consecutive virtual address ranges. The memory manager is also informed that there is a low probability that these pages will be referenced in the short term, which makes them readily available for replacement if main memory is needed. This input to the memory manager eases its ability to claim needed main memory pages on demand while minimizing the number of accesses to disc to post modified pages (minimizing disc utilization).

Disc utilization is optimized through the previously mentioned prefetch and post strategies, along with extent management. When a page fault occurs on a portion of a disc file that does not have disc space allocated, a strategy routine is invoked that determines the size and placement of the file extent to be created. File extent sizes on MPE XL are variable, with no practical limit on the number of extents in a file. The metrics consulted when determining the new extent characteristics are dominant access mode (sequential or random), file capacity, access method hints, and extent fault frequency.

Sequentially accessed files tend to have large extents consecutively located on a disc drive. This allows the prefetch and post strategy routines to minimize accesses to the disc, increasing effective disc utilization. Random access files tend to have smaller extents spread across disc drives, allowing highly parallel access to data for both prefetching and posting. Extent allocation size may also be modified according to the remaining disc space on a drive, allowing full (100%) utilization of the media by the file system.

Specialized Mechanisms

The throughput of the MPE XL data management subsystem has been enhanced through the implementation of specialized mechanisms. These mechanisms address systematic problems in commercial computing systems. The areas can be loosely categorized into increased data concurrency and repetitive disc file reference.

MPE XL integrates a transaction management facility within its disc data management subsystem. This facility allows privileged subsystems of MPE XL data management services to make use of the integrated lock, log, and recovery facilities.

All permanent data structures managed by the MPE XL file system have the transaction management property enabled. This allows highly concurrent access to Turbo-Image/XL data bases, system directories, and disc file labels. MPE XL transaction management will be discussed later in this article.

Granular use of shared and exclusive semaphores in data management services has virtually eliminated artificial points of serialization in data management. Concurrent file opens, closes, page faults, disc file map-ins to virtual space, and name space resolution can occur. If contention does arise, automatic process priority elevation mechanisms alleviate convoy effects. In many instances, like exclusive file access, all locking is avoided by access methods to give optimal performance.

Repeated references to disc files are enhanced by a LRU (least recently used) list of closed files. All files that are closed (no users) are placed on the LRU list and remain mapped into virtual space with file pages remaining in main memory. When an initial file open operation locates a file on the LRU list, it is merely removed from the LRU list and reactivated. Thus the file open operation and subsequent accesses can potentially occur with no disc accesses. This is especially significant for operating system functions like directory scans, or job steps in user applications where the output of one program feeds the next.

Temporary or new files are also treated specially in MPE XL. Because of the transient nature of these files, the MPE XL file system avoids posting data to disc unless main memory is required by memory management for other tasks. It is possible for a temporary file to be created, accessed, and deleted without a single disc file access.

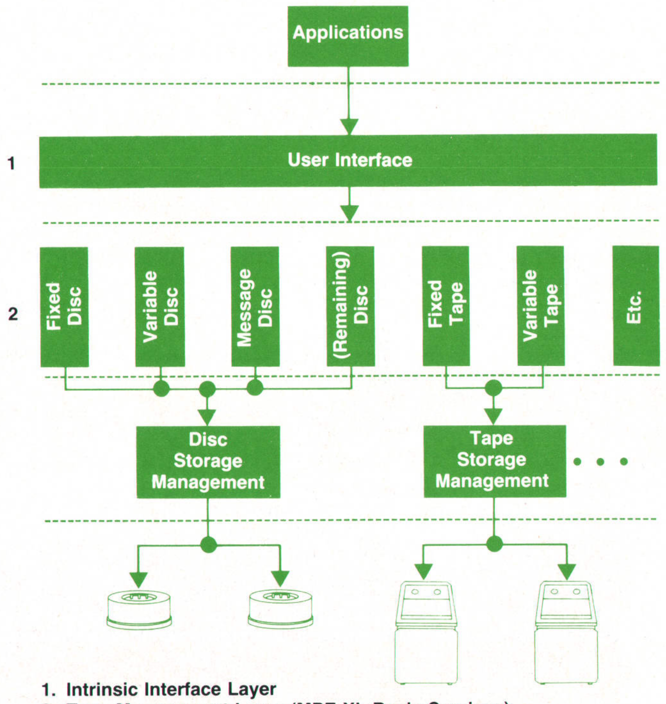
Accessing new portions of new disc files, or extending the end-of-file point on existing files, also have special optimizations. When a page fault is detected on one of these previously untouched or virgin pages while accessing a file, a disc transfer to read the page is avoided. The memory manager merely claims an available page in memory and initializes it to the file's fill character (usually blanks or zeros).

File System Architecture

The organization of the MPE XL file system is optimized for flexible extension of operating system externals via a common set of file system services. Three levels constitute the primary access method path in the file system. The levels are the storage management, type management, and intrinsic interface layers (Fig. 5).

The highest level of the file system hierarchy is the intrinsic interface. Each intrinsic interface module can efficiently support the procedural interface, semantics, and error conventions documented for the interface.

Type managers occupy the middle level of the file access hierarchy. They define a consistent set of interfaces and operators that can be applied to a file through the intrinsic interface. All intrinsic environments access a specific file type through the same type manager, ensuring integrity of the file. Access to a particular type manager, or type manager operator, may be restricted by an intrinsic interface, and is not enforced by the type manager. Internally, type managers provide a logical abstraction (fixed, variable, byte stream, keyed, etc.) of the specific storage management



1. Intrinsic Interface Layer
2. Type Management Layer (MPE XL Basic Services)
3. Storage Management Layer (MPE XL Basic Services)

Fig. 5. *File system structure.*

module being accessed.

Storage management is the lowest layer of the three abstract layers of the MPE XL file system. It defines the set of basic operators that can be applied to a specific class of devices (discs, tapes, terminals, etc.). Within the disc storage management module, additional subsystems such as transaction and disc volume management are also included.

Binding between the three levels is performed at file open time. The file open module determines the proper storage management and type management modules from the file label, and performs a one-time binding between the levels for all subsequent accesses. Establishing the access path early, through specific type and storage management levels, permits MPE XL data management to execute short, focused code sequences.

Namespace Resolution

The MPE XL file system maintains an internal file name, called a UFID (unique file identifier), for every file in the system. This identifier guarantees that an instance of a file is unique in time. The identifier is a combination of the unique media identification and a time stamp. This UFID provides a file handle that satisfies both network and transaction management requirements. MPE XL file system services use this handle internally as the rendezvous mechanism between users and files.

MPE XL Data Management: Transaction Management

The classic approach to providing transaction management functionality in a commercial computing system has been *ad hoc*, at best. Concurrency control mechanisms are either provided by file system access methods or data base management. Recovery, auditing, and backup and recovery mechanisms are provided by file system, data base, and application facilities. To complicate issues further, different file system access methods, data bases, and applications tend to manage each mechanism differently. The result is a complex solution requiring duplication of effort, high administrative and support overhead, and compromised performance.

By consolidating all these functions into a single module common to all disc access methods, MPE XL has achieved a consistent and efficient facility that all privileged subsystems and the MPE XL file system can use. Performance and efficiency gains are also realized over implementations of these facilities at higher levels of the system by tight coupling with memory management, I/O, and HP Precision Architecture protection hardware.

The three basic facilities provided by the MPE XL transaction management facility are the recovery manager, the log manager, and the lock manager.

Lock Manager

The concurrency control mechanism in a system can be viewed as a scheduler. It accepts begin, data access, and commit requests from transactions, and decides whether to allow, postpone, or reject these requests. To control concurrent transactions accessing shared data, a concurrency controller has been implemented in the MPE XL file system.

The unit of locking in MPE XL is a logical page (4096 bytes). Implicit locking (load/store access) locks single pages, while explicit locks by virtual address range may accumulate multiple page locks. When a transaction is granted access to a page, a protection identifier (PID) is placed on that page, allowing only those processes participating in that transaction to read or read/write that page. Any other process attempting access to that page is trapped by hardware to the lock manager.

The operating system and privileged subsystems are provided with procedures to define the boundaries of a transaction. Locks can be claimed for file system operations as they are executed. This is sometimes referred to as locking on the fly. Under this scheme, it is possible for deadlocks to occur.

The MPE XL lock manager detects a deadlock and resolves it by backing out one of the transactions and releasing its locks, thereby allowing the other transaction to proceed, and then notifying the program so that it can perform cleanup and optionally restart the transaction. A simple example of a deadlock is shown in Fig. 6. Extensive analysis has shown that deadlock probability is low, and that throughput advantages over classic logical locking schemes are substantial.

Log Manager

Logging and recovery provide both physical and logical disc file consistency. Physical consistency is provided to preserve structures within the files in case of a system crash. Logical consistency is provided in case of a user transaction abort, a user process abort, a system crash, or a hard crash.

Logging maintains enough information to preserve consistency at the transaction level and guarantee that the recovery manager can perform soft or hard crash recovery. To implement transaction abort, logging records contain enough information about each user operation on a set of files to undo the actions. This information, which includes the content of the before and after images of the modified virtual address range, is written in the recovery log.

Transaction 1	Transaction 2
Request Lock on Object A .....Lock Granted	(Inactive)
(Inactive)	Request Lock on Object B .....Lock Granted
Request Lock on Object B .....Lock Not Granted, Wait Until Free	(Inactive)
Waiting	Request Lock on Object A .....Lock Not Granted, Wait Until Free
Wait	Wait

Fig. 6. *Deadlock sequence.*

Recovery Manager

The MPE XL file system and the TurboImage/XL DBMS use logical and physical disc file recovery, relieving them of the burden of providing their own recovery. Application updates to files that have a recovery property are recorded in a recovery log.

Recovery handles several types of failures, including:

- Program transaction aborts occurring at run time: an implicit abort transaction (through program termination without logical transaction conclusion), or an explicit abort transaction issued by a program.
- Soft crash failure (hardware or operating system failures). It is assumed that all disc hardware and media are in a good state.

If a soft crash failure occurs, the recovery manager reads the log to restore the work of all committed transactions that are not reflected on disc and the user has seen as committed. It will also abort all uncommitted transactions.

Coupled Environment

The term "coupled environment" describes the MPE XL file system's use of the MPE V file system code for infrequent functions, thereby providing full MPE V file system compatibility. This dual (coupled) file system environment was developed to:

- Provide complete MPE V file system compatibility.
- Provide for future MPE XL file system functionality.
- Provide full MPE XL file system performance for paths not relying on the coupled environment.

No performance penalty is incurred because of the presence of the coupled environment.

All MPE V file system calls are intercepted by the MPE XL file system's intrinsic interface. The appropriate type manager is called, and if it is an MPE V file system type manager, the MPE V compatibility mode code will be invoked.

Fig. 7 describes the flow of an MPE intrinsic in the coupled environment.

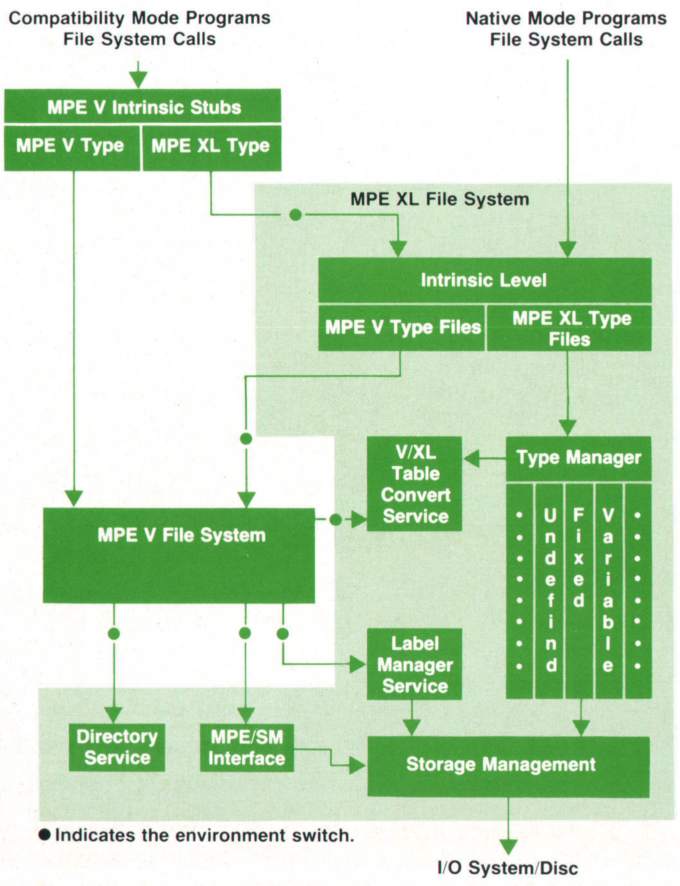


Fig. 7. *Coupled environment provides full file system compatibility.*

MPE XL I/O Software

This section describes the architectural aspects of the low-level MPE XL I/O software system, including its overall goals, framework, and external interfaces.

The low-level I/O software system (LLIO) is the part of the I/O system that is directly responsible for controlling the operation of the system's I/O hardware components. The high-level I/O (HLIO) system implements operating system I/O device control policies and drives the low-level I/O software system to implement device file I/O, virtual memory I/O (including mapped disc files), and other I/O operations.

External Interfaces

As shown in Fig. 8, the I/O software system is a low-level component of the operating system, responsible for the operation of the system's I/O hardware modules.

The external interfaces of the I/O software system are: a system I/O interface, a configuration interface, an I/O services interface, a diagnostics interface, an interrupts interface, and a hardware I/O interface (Fig. 9).

The interfaces to the I/O software system are message based, except for the interface to the I/O hardware and portions of the I/O services interface. Interactions with the I/O software system take place by transfers of data packets, called messages. I/O system code is activated by a scheduling mechanism.

The MPE XL message transmission system (ports) is a key component of the I/O software system. It is described fully from an external point of view later in this document,

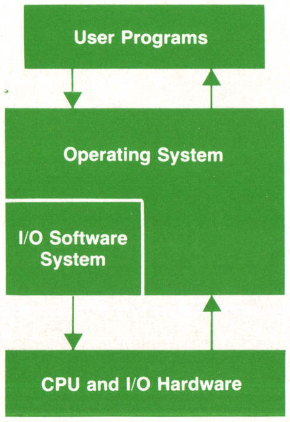


Fig. 8. *General system organization.*

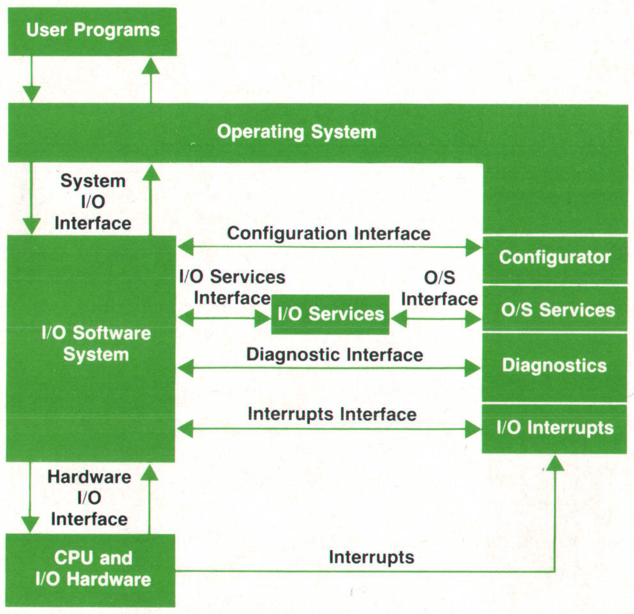


Fig. 9. *I/O software system interfaces.*

and is based upon the basic port mechanism previously described.

System I/O Interface

The system I/O interface is the interface used by the operating system to initiate I/O operations within the I/O software system. I/O request messages are sent from the operating system to the I/O system through this interface. I/O reply messages are returned through this interface to inform the operating system of the results of an I/O request.

Through the system I/O interface the operating system can abort I/O requests it has previously made, and obtain status information about the I/O hardware and I/O software. Also through this interface, the I/O software system notifies the operating system of the occurrence of asynchronous events. Asynchronous events are events that are neither solicited by nor associated with any particular I/O request made to the I/O system. Examples of asynchronous events are a disc coming on-line after an operator has mounted a disc pack, a printer experiencing a power failure, or a modem communication line on a switched telephone network dropping out.

Hardware I/O Interface

In HP Precision Architecture there are no specialized machine instructions for performing I/O operations. Instead, I/O software communicates with HP Precision Architecture I/O hardware through a memory mapped interface in which registers of the I/O hardware modules appear in processor-accessible memory address space.[5] Thus, HP Precision Architecture I/O software interfaces with the I/O hardware by storing information to and loading information from registers on the I/O hardware modules.

In addition to this direct form of I/O operation, HP Precision I/O Architecture provides a standardized mechanism for direct memory access (DMA) by the I/O hardware. I/O software participates in DMA operations by first constructing DMA control programs in main memory, then using a direct I/O register operation to instruct the I/O hardware to begin a DMA operation. After performing data transfers, the I/O hardware places status information back into main memory at locations prespecified by the I/O software, and notifies software via an I/O interrupt that the DMA is completed.

Diagnostic Interface

MPE XL systems are provided with an on-line diagnostics capability in the form of a diagnostics subsystem. The diagnostics interface to the I/O software system has been provided to connect I/O software to the diagnostics subsystem.

Through this interface, I/O software can notify diagnostics software of conditions that call for actions by diagnostics. The diagnostic system can gain temporary exclusive access through the diagnostics interface to any internal hardware-controlling component of the I/O software system for test purposes. This allows diagnostics to be run on-line. After testing is completed, the diagnostic system sends a message through the diagnostics interface returning control of the I/O hardware module to the I/O software system.

Managers

The fundamental operational components of the I/O software system are called *managers*. Manager is the MPE XL name for what has been called a driver in past systems. A manager is a system procedure that controls the operation of one or a collection of I/O hardware modules, or that performs an I/O related control or management function, such as the implementation of a control protocol for a communications link.

In HP Precision Architecture I/O software, there is a *type* of manager for each kind of I/O job to be done in the system, and an *instance* of that type of manager for each occurrence of that job, which must be kept separated from other jobs of the same type. Some examples of managers are:

- A device manager for a particular family of I/O devices, such as CS-80 discs, laser page printers, and streaming magnetic tapes
- A device adapter manager for a particular kind of HP-CIO channel's device adapter, such as the HP-IB or a local area network
- A channel adapter manager, also called a bus adapter manager, for a particular kind of I/O channel, such as HP-CIO
- A disc subsystem manager for control of a set of discs and the device adapter that connects the discs to the I/O channel
- An I/O protocol manager for a particular kind of I/O protocol.

Managers communicate with one another, with MPE XL, and with other parts of the system by messages. Requests for service from an I/O manager are transmitted by a message specific to that manager and that function. The manager is activated by the arrival of the message, reads and processes the message, takes whatever action it needs to perform the desired function, and then suspends itself (by exiting), and waits for another message. Often, when performing its function, a manager will need to send its own request messages to other managers that participate in the control of the I/O hardware path. When hardware I/O action completes, an interrupt is generated that results in the activation of a lowest-level I/O manager, typically the channel manager. From this point, reply messages are sent up the path from manager to manager and eventually back to the operating system, communicating the completion status of the I/O function originally requested.

Configurator

The function of the MPE XL configurator is to build a particular instance of the I/O software system to suit a particular hardware configuration. To do this, the configurator interacts with the I/O managers, one at a time. This interaction takes place for the purpose of obtaining configuration information from the managers, for example, the sizes of data structures required by the various types of managers. The configurator first creates instances of managers, and then directs the managers through a binding operation to link them together so that they can thereafter communicate with one another.

I/O Services

I/O services in MPE XL are a collection of code modules, typically callable procedures, that provide common service functions to the I/O managers.

Three main classes of services are provided: communication services for sending and manipulating messages, resource management services for working with system resources such as memory buffers and timers, and system management services, which include I/O transaction management, I/O software traps and measurement facilities, power-on notification, and an interface to the globally accessible I/O system ports used by I/O software.

Interrupt Handlers

Interrupt handlers are system code modules that execute in response to hardware interrupts. The system external interrupt handler (EIH) is the first software to execute in response to an interrupt.

MPE XL's approach is to make the interrupt handler essentially an interface mechanism to pass control and interrupt information on to the appropriate I/O manager. The I/O interrupt handler, called as a procedure by the system interrupt handler, does very little work. Information is passed to the I/O manager by sending a message from the I/O interrupt handler to the manager. The I/O manager then handles the hardware specific processing needed to service the I/O interrupt.

Through this implementation, MPE XL can quickly respond to classes of interrupts and most efficiently dispatch the proper handler. For disc I/O completions, special disc I/O completion queue handling is invoked to follow a fast path through the I/O system.

I/O System Tables

Although there is a need for tables of information within the MPE XL I/O software system, no particular table structures are architecturally specified. Instead, table structures are considered to be implementation dependent.

The system design uses the collection of self-contained managers, each with its own private data structures, to store most of the tabular information within the system. Each manager's data structures are known in detail only to itself. Access to a manager's data structures from outside of the manager is obtained only by sending messages to the manager requesting information and by the manager's sending back the information in a reply message. This approach keeps the managers independent of each other, and keeps other parts of the system independent of the internal structures of managers.

Some global information is necessary for system operation. This information, not the property of any particular manager, is kept in I/O system global tables managed by the I/O services.

Included in global tables are configuration information and global services information. Configuration information includes the associations of managers with their (message) port numbers. To provide powerfail recovery, part of this information, namely the port numbers for all of the lowest-level managers, must be kept in a memory resident table. Global services information also resides in a memory resident table. The port numbers of the globally accessible services ports are kept in this table. These global ports include the configurator port, the memory services port, the surrogate services port, and the diagnostics port.

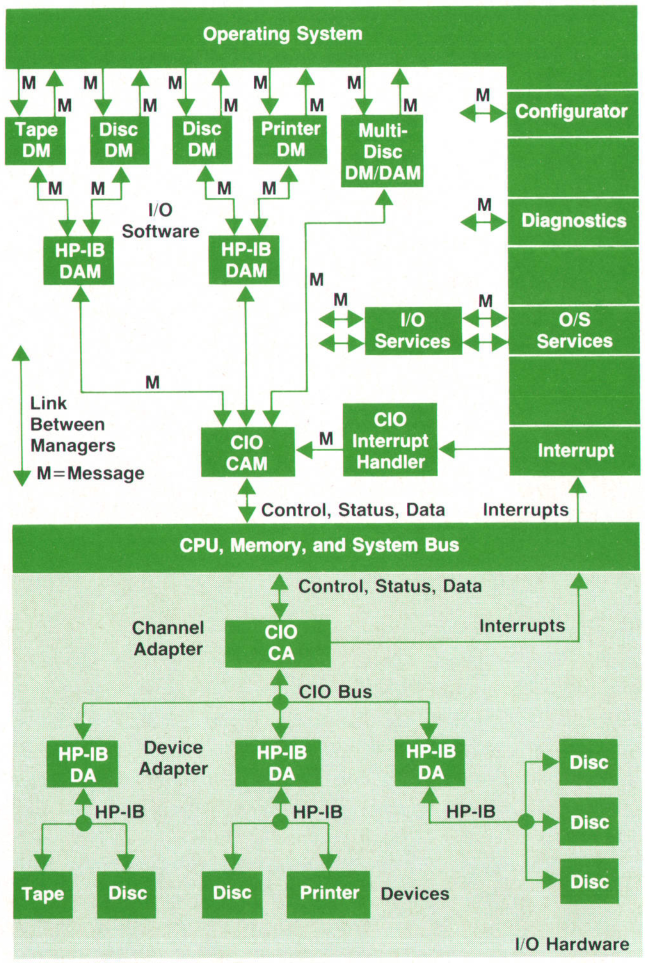


Fig. 10. *I/O software organization.*

Hierarchical Organization

The general design of MPE XL I/O software provides for a 1-to-n-level hierarchy of I/O managers. This generality of organization allows implementers great flexibility in choosing particular organizations to meet the needs of their systems. Fig. 10 shows an example organization. It is for example only, and not intended to represent any actual system organization.

The fundamental idea of the hierarchical organization is to have an I/O software structure that mirrors the I/O hardware structure, insofar as the I/O hardware has to be visible to software. That is, there is an I/O manager for each type of I/O hardware module, and an instance of the I/O manager for each instance of its type of hardware module that needs management.

An example of a three-level hierarchical organization for the control of a tape drive, two disc drives, and a printer is shown in the lower half of Fig. 10. In this example, there are three levels of I/O hardware: device, device adapter, and channel adapter. Hence, there are three levels of I/O software: device manager (DM), device adapter manager (DAM), and channel adapter manager (CAM). The CAM has an interrupt handler associated with it, because in most implementations, the channel adapter (also known as a bus adapter) is the hardware module that sends interrupts to the CPU.

Since there is one tape device and one printer device, there is one instance of a tape DM and one of a printer DM. However, since there are two disc devices of the same kind on this system, there are two instances of a disc DM to control them separately. There is only one copy of the disc DM code in the system. That code is shared between the two instances of the disc DM. Code sharing is explained further below.

The idea underlying the hierarchical organization is that each type of manager is responsible for the operation of a particular type of I/O hardware. The manager knows all about its associated hardware, but does not know the details of operation of other I/O hardware modules that lie in the path leading from the CPU to its associated hardware module. Instead, other managers encapsulate the knowledge of how to operate the other pieces of I/O hardware, and those managers present a software interface to make those hardware modules accessible to the higher-level managers.

The benefits of this form of organization are threefold. First, it provides maximum flexibility of system configuration. The I/O hardware can be configured as desired to satisfy customer needs, and can be expanded incrementally. The corresponding modular I/O software can be plugged together to match the hardware configuration.

Second, it provides maximum leverage in the I/O software, by encapsulating software changes (which may be made necessary by hardware changes) inside just those software modules that are associated with the altered hardware. For example, a new implementation of the HP-IB device adapter may require a new method of software control for the adapter. If the changes are limited to the internal controls of the device adapter, then a new device adapter manager will be created to implement the new form of control. The new device adapter manager can use the same software interface to the device managers above it as was used in the old device adapter manager. Thus, there need be no change whatever to the device managers.

Third, it provides reliable I/O software by having single-point control over each I/O hardware module and its associated software resources. With only a single software module manipulating the state of a hardware module and manipulating the system resources associated with that hardware module, conflicts and inconsistencies that can occur when multiple software modules share access to a resource are prevented.

Message System

The MPE XL ports facility provides the mechanism for message transmission between I/O managers, and between other parts of the operating system. Message transmission includes not only the passing of message data between a sending and a receiving code module, but also the scheduling and activation of the receiving code module. In the I/O software system, several kinds of code module scheduling and activation are provided. These range from the simplicity of a procedure call to the complexity of a process switch.

These functions provide a priority execution scheme for the I/O software system similar to that of the operating system's process scheduler. High priority is given to those modules that manage the most critical I/O resources, for example the channel. Lower priority is given to the managers of the end devices. This allows the I/O system to be tailored for fast response to events deemed important (such as a channel interrupt) through preemption of modules responding to lesser events.

An I/O manager sends a message by first building the message and then calling the I/O service procedure IO SEND, which is the interface between I/O managers and the internals of the message system. IO SEND accepts the sender's message and calls upon the message system to deliver the message to the receiver, which has been addressed by giving its port number and a subqueue number as parameters to IO SEND.

A port is a data structure within the message system that holds messages in transit between a sender and a receiver. When a message is sent to an I/O manager, the message is queued within the manager's port at the tail end of the queue specified by the sender. Senders, therefore, have to know the receiving manager's protocol for queue use. They learn about this during I/O configuration binding. The message will be delivered to the manager when it becomes the next message to be delivered.

Port Servers

A port server is a procedure that receives and acts upon a port's messages. When a message is sent to a port, the server is activated by means of a procedure call made to it by the message system. Thus, it is activated quickly, about as quickly as a procedure call. The actual activation time can be somewhat longer than a straight procedure call, however, because of possible message queuing and priority scheduling.

Activation of a server can involve a process context switch. This occurs when entering or leaving the interrupt control stack (ICS) environment, which is a special system environment, independent of user processes, established to implement priority scheduling of servers and other system events.

There are three types of servers: those specified to run only on the ICS, those specified to run only in a process, and a hybrid type that can run in either environment, ICS or process. ICS servers always run on the ICS, and gain the benefit of a priority dispatching mechanism, which orders their execution so that the highest-priority server with a message pending runs first. ICS servers are assigned their relative priorities during system configuration, allowing the system to be tuned for the best aggregate I/O throughput. Since ICS servers always have priority over user processes, they are not affected by the time-slicing quanta that normally interrupt user process execution. They are, however, interruptible by I/O interrupts and preemptable by higher-priority ICS servers.

Since the ICS is an environment that preempts normal process execution, it is crucial that ICS servers operate under a basic restriction: they must not block themselves, nor cause any other code to run on the ICS that could block because of the unavailability of a resource or wait for a long time for an event to occur. In cases where a server has a real need for a service that could block or wait, then either the server must not be an ICS server or it must be an ICS server that uses a surrogate server to handle the blocking or waiting.

Process-based servers always run in a process environment, never on the ICS. Thus, they run at the priority of the process in which they are activated. This allows the same server to run at different priorities at different times depending on the importance of the process within the system. Process-based servers are not subject to the restrictions of the ICS; hence they may block or wait for events.

Hybrid servers execute in either the ICS or the process environment. They are activated by the message system in the environment that is current when it is their turn to execute, as determined by ICS or user process priority. Hybrid servers must be designed to operate under ICS restrictions since these servers cannot assume that they are not on the ICS.

Activation of Servers

A server is activated by the message system by a procedure call. The message system establishes the proper environment for the server (ICS or process) and then calls the server procedure. Parameters passed to the server include a pointer to the next message to be processed and a pointer to the server's private data area associated with the server's port.

An important function of the message system for server activation is the enforcement of nonreentrancy or seriality of servers. Once a server has been activated by an incoming message, it will not be activated again (i.e., reentered) until it has finished its current operation and voluntarily exited its procedure. This feature of the message system, performed automatically together with message queuing, considerably simplifies the design and implementation of servers, since the servers themselves do not have to account for operating sequences in which they might be interrupted and reentered.

ICS servers are assigned execution priorities with respect to one another, and as a group they take priority over all other processes in the system. This means that all ICS servers with unmasked messages pending will be executed before any user process. In this way an operating system can give preference to I/O operations, and the system can be tuned for best overall I/O performance. Process-based servers, and hybrid servers being called in a process environment, are activated in the same manner as ICS servers, except that they are subject to the operating system's process priority scheme. These servers have no defined priority with respect to one another. Rather, they assume the priority of the process in which they are invoked.

MPE XL User Environment

The MPE XL user environment is generally a superset of the MPE V environment, with a few exceptions (for example, initial releases do not support CSTX). MPE XL extends the command and programmatic interfaces of MPE V, and supports execution of object code from MPE V systems as well as recompilation using native compilers. The following sections describe the emulation tools and structures used to support the MPE XL environment, as well as the extensions to the MPE V commands and programmatic interfaces available in the MPE XL environment.

MPE V Emulation

MPE V machine emulation is supported by the HP 3000 Emulator and the HP 3000 Object Code Translator (OCT).[8] The emulator is a program that reproduces the exact behavior of the HP 3000 instruction set (with the exception of a few privileged instructions). Users do not have to invoke special services to run the emulator. This is handled automatically by the operating system whenever a compatibility mode program is loaded.

OCT provides object code compatibility by translating MPE V-based HP 3000 machine instructions into HP Precision Architecture instructions. Essentially, it is an object code compiler. The advantage that it has over the emulator is in the execution speed of the compiled code. Translating CPU-intensive programs can yield up to three or four times the execution speed of emulation. While optimization techniques used by OCT eliminate unnecessary code sequences, the largest savings is in the elimination of the instruction fetch and decode cycles required by the emulator. OCT must be explicitly run against a compatibility mode program.

The emulation tools and compatibility mode system services execute in an environment that is similar to that provided by the MPE V-based HP 3000. This consists of a 16-bit addressable stack, code and data segments, and a bank 0 (absolute addressing) segment. The stack and segments are defined and treated like their MPE V-based HP 3000 counterparts. The compatibility mode tools support the MPE V hardware register use that defines these objects.

The MPE V-based HP 3000 architecture supports the notion of absolute addressing instructions and privileged operating system instructions. These are handled in MPE XL by supporting a data structure that the emulation tools treat as physical memory, but is actually a virtual object in MPE XL.

A 128K-byte area of virtual memory is carved out of the

1G-byte program stack to contain the MPE V compatibility mode program stack. This area contains the 64K-byte maximum MPE V program stack plus two 32K-byte guard bands. These guard bands are used to detect compatibility mode address violations via HP Precision Architecture protection mechanisms, rather than relying on explicit bounds checking code generated by the OCT which would increase code path lengths. The guard areas do not have any virtual (or disc) space allocated to them.

MPE V system data structures, such as the PCB (process control block) and XDDs (input/output device directories for spooling), are carried over on top of MPE XL by running Initial, MPE V's startup program. Initial creates and initializes all required MPE V operating system data structures in MPE XL objects. This provides the base through which MPE XL can selectively run compatibility mode MPE V system code and leverage its functionality for operations that are not performance sensitive.

MPE XL User Interface

The main objective of the MPE XL user interface is to provide a compatible environment for the current customer base while providing an extensible structure for future enhancements. This was done by developing the basic support functions in native mode while maintaining many of the MPE V command executors in compatibility mode. These support functions include job and session management, scanner/parser, variable management, symbol table management, and the command interpreter (CI). The most visible of these elements is the CI.

The CI is the part of the user interface responsible for reading user input, executing the command, and properly handling any errors. The new CI is a line-oriented processor that accepts interactive or batch input. Also, there are programmatic interfaces to the CI to permit MPE XL subsystems and user programs access to system services. The CI is designed to be friendly and consistent, compatible with MPE V, language localizable, user tailorable, structured, maintainable, and extensible.

A significant number of feature enhancements have been included in the CI, making it commensurate with the scope of offerings of MPE XL. Included in these features are executable history stacks, centralized scanner/parser (for language localization), system and user defined variables with expression evaluators, command files, and file search paths.

MPE XL Support Features

MPE XL provides support tools appropriate for the various levels of support required. High-level problem determination tools exist to allow personnel not expert in the internals of MPE XL to resolve problems. Expert-level tools also exist to support experts in the quick resolution of problems.

Symbolic Debugger and Dump Analysis Tool (DAT)

MPE XL contains an extremely powerful operating-system-level debugger. In addition to several features not found in most system debuggers, an interactive programming language-like environment is provided. Furthermore, off-line dump analysis is provided by the same program such that nearly all of the debugger functions are available when processing a memory dump.

Standard debugging features include program and data breakpoints, single-stepping through a program, modification of storage or registers in various addressing modes (native and compatability mode), and stack trace.

The debugger/DAT supports a rich set of commands, which includes support of variables (integer up to 64 bits, pointers, and strings), arithmetic and Boolean operations, indirect addressing, loop control (WHILE, FOR EACH), dynamic command expansion, standardized error handling, command stack history, a standard output filter that uses regular expressions, a powerful expression formatter, and the ability to do input and output from or to a file.

There are standard functions in the following categories: string manipulation, type coercion, address manipulation, process management, procedure identification, and symbolic access. A facility for writing user defined functions (macros) is also provided.

Full support is provided for both native and compatability mode. When in compatability mode, the debugger/DAT displays all programs, storage, and registers appropriate to compatability mode so that they are intuitively useful to an HP 3000 MPE V programmer.

Windows. The windows feature of the debugger allows the user to have several areas of storage on the screen at the same time, including the current section that the program is executing (displayed symbolically), the top of the stack, the registers, and areas of selected virtual storage (see Fig. 11). After the completion of each command, all of the windows are automatically updated to reflect the current state of the program execution and memory. In addition, values that have changed are highlighted.

Powerful commands allow establishing windows to point to a given address, jumping to an address in a window, displaying windows in any base, scrolling, and changing window size.

Symbolic Access and Formatting. Symbolic access and formatting are available for both program locations and data structures. Program locations are displayed in terms of symbolic procedure names and offsets rather than memory addresses.

A symbolic specification for accessing and formatting data structures consists of a virtual address and an HP Pascal/XL type definition. This specification is used to map the specified area in memory. Full support of HP Pascal/XL type definitions is provided, including formatting arrays, dereferencing pointers (to any depth), and case variants. An example of symbolic formatting is shown in Fig. 12.

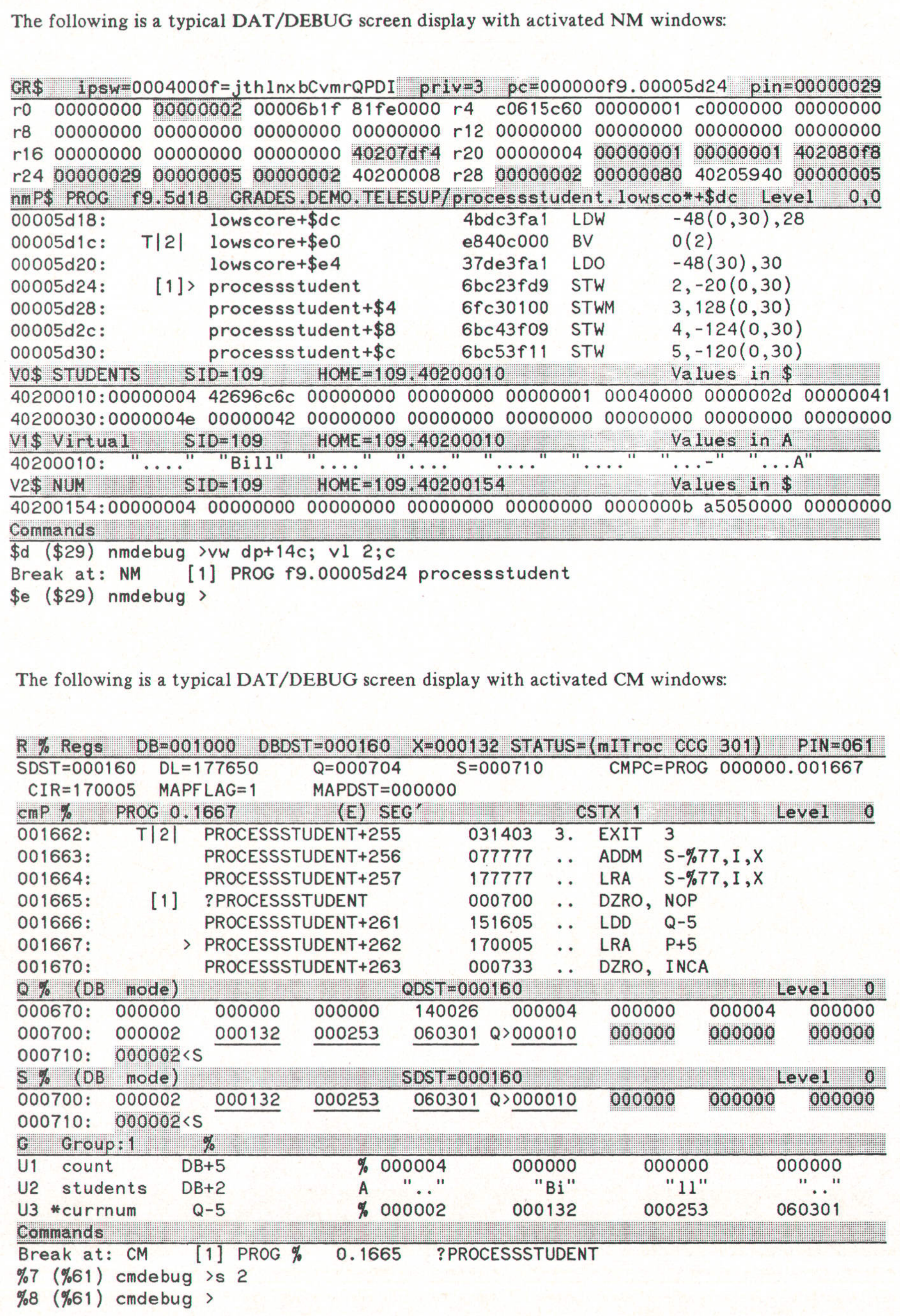
Symbolic access is provided by functions that return a value based on a symbolic specification, the length of a symbolic type, the offset of an element in a symbolic type, and the value of HP Pascal/XL constants.

Symbolic access to all operating system data structures is provided by referencing information contained in the Spectrum object module (SOM) files. This includes records of the following: constant and type definitions, variable definitions, and procedure definitions. SOM files with this debugging information are produced automatically in the process of building MPE XL. This ensures that the symbolic formatting information will always match the version of the operating system being debugged.

Macro Facility. Since the debugger/DAT is an interpreter, powerful user defined macros can be written that use all of the debugger functions. The macro facility includes local variables, an automatic help facility, and parameter passing. From the user perspective, macros can be invoked like any standard function that returns a single value. An example of a macro is shown in Fig. 13.

MPE XL Macro Package. A macro package is provided with MPE XL containing macros to answer frequently asked questions about a dump in such a way that Hewlett-Packard support personnel can provide first-level troubleshooting support without requiring user knowledge of the internal structure of MPE XL. These macros are designed to be used both interactively and in batch.

Three levels of macros are provided. High-level macros answer general questions, such as, "What is the current process doing?" (see Fig. 14), "What was the machine state?," or "What files are being accessed by whom?" Low-level macros allow the dump reader to home in on a particular area, such as, "Who is sharing this file?" or "What


The following is a typical DAT/DEBUG screen display with activated NM windows:

```
GR$   ipsw=0004000f=jthlnxbCvmrQPDI  priv=3  pc=000000f9.00005d24  pin=00000029
r0  00000000 00000002 00006b1f 81fe0000 r4  c0615c60 00000001 c0000000 00000000
r8  00000000 00000000 00000000 00000000 r12 00000000 00000000 00000000 00000000
r16 00000000 00000000 00000000 40207df4 r20 00000004 00000001 00000001 402080f8
r24 00000029 00000005 00000002 40200008 r28 00000002 00000080 40205940 00000005
nmP$ PROG  f9.5d18  GRADES.DEMO.TELESUP/processstudent.lowsco*+$dc  Level   0,0
00005d18:           lowscore+$dc             4bdc3fa1  LDW      -48(0,30),28
00005d1c:    T|2|   lowscore+$e0             e840c000  BV       0(2)
00005d20:           lowscore+$e4             37de3fa1  LDO      -48(30),30
00005d24:      [1]> processstudent           6bc23fd9  STW      2,-20(0,30)
00005d28:           processstudent+$4        6fc30100  STWM     3,128(0,30)
00005d2c:           processstudent+$8        6bc43f09  STW      4,-124(0,30)
00005d30:           processstudent+$c        6bc53f11  STW      5,-120(0,30)
V0$ STUDENTS    SID=109      HOME=109.40200010                Values in $
40200010:00000004 42696c6c 00000000 00000000 00000001 00040000 0000002d 00000041
40200030:0000004e 00000042 00000000 00000000 00000000 00000000 00000000 00000000
V1$ Virtual     SID=109      HOME=109.40200010                Values in A
40200010:  "...."  "Bill"  "...."  "...."  "...."  "...."  "...-"  "...A"
V2$ NUM         SID=109      HOME=109.40200154                Values in $
40200154:00000004 00000000 00000000 00000000 00000000 0000000b a5050000 00000000
Commands
$d ($29) nmdebug >vw dp+14c; vl 2;c
Break at: NM    [1] PROG f9.00005d24 processstudent
$e ($29) nmdebug >
```

The following is a typical DAT/DEBUG screen display with activated CM windows:

```
R % Regs    DB=001000  DBDST=000160  X=000132 STATUS=(mITroc CCG 301)    PIN=061
SDST=000160  DL=177650      Q=000704      S=000710       CMPC=PROG 000000.001667
 CIR=170005  MAPFLAG=1      MAPDST=000000
cmP %    PROG 0.1667           (E) SEG'            CSTX 1                Level   0
001662:     T|2|   PROCESSSTUDENT+255        031403  3.   EXIT  3
001663:            PROCESSSTUDENT+256        077777  ..   ADDM  S-%77,I,X
001664:            PROCESSSTUDENT+257        177777  ..   LRA   S-%77,I,X
001665:       [1]  ?PROCESSSTUDENT           000700  ..   DZRO, NOP
001666:            PROCESSSTUDENT+261        151605  ..   LDD   Q-5
001667:          > PROCESSSTUDENT+262        170005  ..   LRA   P+5
001670:            PROCESSSTUDENT+263        000733  ..   DZRO, INCA
Q %  (DB  mode)                      QDST=000160                     Level   0
000670:  000000   000000   000000   140026   000004   000000   000004   000000
000700:  000002   000132   000253   060301 Q>000010   000000   000000   000000
000710:  000002<S
S %  (DB  mode)                      SDST=000160                     Level   0
000700:  000002   000132   000253   060301 Q>000010   000000   000000   000000
000710:  000002<S
G   Group:1        %
U1  count        DB+5             % 000004        000000       000000       000000
U2  students     DB+2             A   ".."          "Bi"         "11"         ".."
U3 *currnum       Q-5             % 000002        000132       000253       060301
Commands
Break at: CM     [1] PROG %   0.1665   ?PROCESSSTUDENT
%7 (%61) cmdebug >s 2
%8 (%61) cmdebug >
```

Fig. 11. *Debugger windows show several areas of storage at the same time.*

```
($1) $ nmdat> fv a.c0083800 tm.tbl_hdr

PACKED RECORD

   HDR_ADDR              : a.c0083800
   HDR_SEMAPHORE         :
      SEM_STATE : 1
      SEM_CLASS : 0
      SEM_SPEC : 1
         SEM_WAIT_COUNT : 0
         SEM_HEAD_PIN   : 7ffd
         SEM_TAIL_PIN   : 7ffd

            SEM OWNER PIN : 7ffd
   HDR_OPTIONS           : [ 1 , 5 , 7 , 8 ]
   HDR_ENTRY_SIZE        : 10
   HDR_ALLOCATED_HEAD    : 0
   HDR_ALLOCATED_TAIL    : 0
   HDR_PRIM_SIZE         : 3000
   HDR_PRIM_HEAD         : de60
   HDR_PRIM_TAIL         : 35b0
   HDR_PRIM_INIT_SIZE    : 3000
   HDR_PRIM_INCR_SIZE    : 0
   HDR_PRIM_NEXT_BLOCK   : ffffffff
   HDR_SECN_SIZE         : 0
   HDR_SECN_HEAD         : 564e80
   HDR_SECN_TAIL         : 7c0
   HDR_SECN_INIT_SIZE    : 0
   HDR_SECN_INCR_SIZE    : 0
   HDR_SECN_NEXT_BLOCK   : 6e5000
   HDR_FREE_LIST_TYPE    : 0
   HDR_LINK_OFFSET       : 0
   HDR_INIT_CHAR         : 0
   HDR_TBL_NUM           : 0
   HDR_TBL_NAME          : ' vW@ u  I    y @            s @'
   HDR_BODY_PTR          : 0.6f6040
   HDR_OBJECT_PTR        : 0.6f6040
   HDR_PRIM_MAX          : 2ffff
   HDR_SECN_MAX          : ffffffff
   HDR_LAST_ENTRY        : 2fff
   HDR_SM_GUFD_HEAD      : 0
   HDR_SM_LOCK           : 1
   HDR_SM_POST           : 0
```

Fig. 12. *Symbolic formatting of a variable display.*

outstanding I/O exists on this device?" Expert macros are used by lab personnel to show operating system data structures.

Diagnostic and Repair Aids

Diagnostic Umbrella. The diagnostic umbrella consists of a standard user interface (including security checks), a diagnostic library, and a diagnostic message catalog. Diagnostics can be added to the library and their associated messages added to the message catalog through the user interface. Diagnostics in the library can be invoked to diagnose a system component while the system is in normal operation (for components that can be diagnosed without impairing system operation). For component diagnosis requiring a dedicated system, the system is brought into single-user, minimum-configuration mode, and the diagnostic is invoked through the diagnostic umbrella. The diagnostics package is intended to be used not only locally, but also from remote regional field support response centers. Autodiagnostic invocation is also provided.

Logging Facilities. MPE XL provides logging of I/O errors to a disc file, logging of ongoing system events to a memory resident circular buffer via FOOTPRINT, logging of major system events (logons, security violations, powerfails, etc.) to a system log file, and user logging facilities. Facilities are provided to access and format these event logs.

Dump. MPE XL supports the invocation of a dump, where main memory and relevant disc memory allocated to transient objects (e.g., stacks) are dumped to a tape or disc region.

PMEDAT. PMEDAT is a stand-alone utility that uses the primitive MPE XL startup and driver environment to allow the viewing and modification of disc resident structures without requiring that MPE XL boot from disc.

MPE XL Error Confinement, Notification, and Recovery

Error Convention and Reporting. MPE XL gives each error a unique identifier (subsystem identifier appended with subsystem error identifier). When an error is detected, recovery is attempted if it is possible, (e.g., retries on an I/O error, powerfail state save with restore on power-up). If the error cannot be recovered, the error identifier is returned to the invoking procedure. In this manner, error information is returned to the highest level, where partial degradation or system failure is invoked.

Partial degradations are attempted for file specific (e.g., bad track), process specific (e.g., protection violation), or device specific (e.g., unavailable disc volume) errors by returning error codes on the file access or launching the process into a user supplied error handler. If a system integrity error is encountered that cannot be recovered from,

```
 \*****************************\
 \***  PM_LOADED_FILE_TABLE  ***\
 \***************************JH\

{mac pm_loaded_file_table : any (
   ufid                   : str ='',
   detail                 : int = 1,
   field_name             : str = '',
   extension              : bool = FALSE)

   MACVER  = 'A.00.01'
   MACKEY  = 'MXPM HP Q_PM_X PM EL LOADED FILE TABLE'
   MACHELP = 'Prints the specified information in the Loaded File Table' +
             ' for the specified UFID.'
{

loc lft_pointer           : ptr = kso_pointer (kso_number ('kso_lft'));
loc entry_len             : int = symlen ('!pm_som:lft_entry', #8);

if ufid <> '' then {
   loc gufd_ptr = fs_ufid_to_gufd (ufid);
   loc ufid_ptr = symval (gufd_ptr, '!fs_som:gufd_record.ufid');
   loc ufid_str = strextract (ufid_ptr, symlen ('!fs_som:ufid_type', #8));
   loc lft_ent_ptr = streadkey (lft_pointer, ufid_str);
   if (extension) then {
      if detail <> 0 then
         pm_l_f_t_1 (lft_ent_ptr, field_name)
   }
   else {
      if field_name <> '' then {
         if detail <> 0 then
            fv lft_ent_ptr '!pm_som:lft_entry.!field_name';
         return symval (lft_ent_ptr, '!pm_som:lft_entry.!field_name')
      };
      if detail <> 0 then
         fv lft_ent_ptr '!pm_som:lft_entry'
   }
}
else {
   loc max_soms = symval (lft_ent_ptr, '!pm_som:lft_entry.max_soms');
   loc ext_length = symlen ('!pm_som:lft_extension', #8);
   loc lft_ent_ptr = lft_pointer + ???;
   loc end_ptr = vainfo (lft_ent_ptr, 'ending_vba');
   while (lft_ent_ptr < end_ptr) do {
      if detail <> 0 then
         if extension then {
            pm_l_f_t_1 (lft_ent_ptr, field_name)
         }
         else if field_name <> '' then {
            fv lft_ent_ptr '!pm_som:lft_entry.!field_name;'
         }
         else
            fv lft_ent_ptr '!pm_som:lft_entry';
      loc lft_ent_ptr = lft_ent_ptr + entry_len + max_soms * ext_length
   }
}
}}
```

Fig. 13. *Debugger macro.*

```
$309 ($3) nmdat > PM_FPIB(,2)
Formatted Process Information Blocks:


(part 2)  ===========  PORT INFO  ===========  =====  INTERRUPT INFO  =====

      P   P P     A P
      O   R R H L P P
      R   I I I O R R                                        PROCESS
      T P A D I I I I               NAME or NUM              INTRPT  DISP  DELAYED
      I F D E N I N N PRI    SEM    WAITED                   DISABLE INT   INT
PIN   N P J L T T T T BST    RANK   PORT                     COUNT   LEVL  LEVL
---   - - - - - - - - ---    ----   ----------------         -----   ----  ----
$1    T F T F F F T F $35    $0     PROGEN_GLOBAL_PORT       $0      $6    $6
$2    T F F F F F T F $0     $0     STD SIGNAL PORT          $0      $6    $6
$3    T F F F F F F F $1     $0     STD SIGNAL PORT          $0      $6    $6
$4    T T F F F F F F $0     $0      pfp_port0               $0      $6    $6
$5    T T F F F F F F $0     $0     $ffffffe5                $0      $6    $6
$6    T T F F F F F F $0     $0     $ffffffe4                $0      $6    $6
$7    T T F F F F F F $0     $0     $ffffffe3                $0      $6    $6
$8    T F F F F F F F $0     $0     $ffffffe1                $0      $6    $6
$9    T F F F F F T F $0     $0     $ffffffe0                $0      $6    $6
$a    T F F F F F F F $0     $0     $ffffffe2                $0      $6    $6
$b    T F F F F F F F $0     $0     STD SIGNAL PORT          $0      $6    $6
$c    T F F F F F T F $0     $0     STD SIGNAL PORT          $0      $6    $6
$d    T F F F F F F F $0     $0     STD SIGNAL PORT          $0      $6    $6
$e    T F F F F F F F $0     $0     STD SIGNAL PORT          $0      $6    $6
$f    T F F F F F F F $0     $0     STD SIGNAL PORT          $0      $6    $6
$10   T F F F F F F F $0     $0     $ffffff7a                $0      $6    $6
$11   T F F F F F F F $0     $0     $ffffff76                $0      $6    $6
$12   T F F F F F F F $0     $0     $ffffff70                $0      $6    $6
$13   T F F F F F F F $2     $0     $ffffff6c                $0      $6    $6
$14   T F F F F F F F $0     $0     STD MESSAGE PORT         $0      $6    $6
$15   T F F F F F F F $0     $0     SYSMAIN_PORT             $0      $6    $6
$16   T F T F F F F F $2     $0     JSMAIN_PORT              $0      $6    $6
$17   T F F F F F F F $0     $0     SESSIONMAIN_PORT         $0      $6    $6
$18   T F F F F F F F $0     $0     JOB_QUEUE_PORT           $0      $6    $6
$19   T F F F F F F F $1     $0     JSMAIN_PORT              $0      $6    $6
```

Fig. 14. *The MPE XL macro package contains macros to answer frequently asked questions about a dump, such as, "What is the current process doing?"*

a system failure is issued with the unique error number. Since control is always passed upwards on error detection, additional recovery and partial degradations can be incorporated over time.

MPE XL Resiliency. When unanticipated errors, such as arithmetic exceptions or integrity check failures occur, MPE XL has architected a standard recovery mechanism for system code. The system detects that a critical exception condition has occurred and system code is executing, and will issue an escape back to a prespecified control point in the system code (HP Pascal/XL recover block). System code can then attempt to recover logically by releasing all critical resources and returning an error condition back to the caller.

Minimum System Boot. MPE XL can come up into single-user mode with a minimum set of the hardware configuration operational (the single system disc, console, and cold load path).

Acknowledgments

MPE XL is the result of the effort of over 150 engineers and managers over more than a five-year period. The unswerving dedication of these individuals resulted in the creation of this new production operating system base, which will serve as the foundation for the next decades of HP's commercial computer systems. Each individual who contributed over these years to this operating system should feel a sense of accomplishment and deserves recognition.

References

1. D.A. Fotland, et al, "Hardware Design of the First HP Precision Architecture Computers," *Hewlett-Packard Journal*, Vol. 38, no. 3, March 1987, pp. 4-17.
2. G.R. Gassman, et al, "VLSI-Based High-Performance HP Precision Architecture Computers, *Hewlett-Packard Journal*, Vol. 38, no. 9, September 1987, pp. 38-48.
3. J.S. Birnbaum and W.S. Worley, Jr., "Beyond RISC: High-Precision Architecture," *Hewlett-Packard Journal*, Vol. 36, no. 8, August 1985, pp. 4-10.
4. M.J. Mahon, et al, "Hewlett-Packard Precision Architecture: The Processor," *Hewlett-Packard Journal*, Vol. 37, no. 8, August 1986, pp. 4-21.
5. D.V. James, et al, "Hewlett-Packard Precision Architecture: The Input/Output System," *ibid*, pp. 23-30.
6. D.S. Coutant, et al, "Compilers for the New Generation of Hewlett-Packard Computers," *Hewlett-Packard Journal*, Vol. 37, no. 1, January 1986, pp. 4-18.
7. A.S. Brown, et al, "Data Base Management for HP Precision Architecture Computers," *Hewlett-Packard Journal*, Vol. 37, no. 12, December 1986, pp. 33-48.
8. A.B. Bergh, Keith Keilman, D.J. Magenheimer, and James A. Miller, "HP 3000 Emulation on HP Precision Architecture Computers," *this issue*.
9. J.R. Busch and A.J. Kondoff, "Disc Caching in the System Processing Units of the HP 3000 Family of Computers," *Hewlett-Packard Journal*, Vol. 36, no. 2, February 1985, pp. 21-39.
10. K.W. Pettis and W.B. Buzbee, "Hewlett-Packard Precision Architecture Compiler Performance," *Hewlett-Packard Journal*, Vol. 38, no. 3, March 1987, pp. 29-35.

HP 3000 Emulation on HP Precision Architecture Computers

Two software subsystems for HP Precision Architecture machines provide program execution that duplicates that of HP 3000s running the MPE V operating system.

by Arndt B. Bergh, Keith Keilman, Daniel J. Magenheimer, and James A. Miller

ONE OF THE GOALS for the development of MPE XL, the commercial operating system for HP Precision Architecture computers such as the HP 3000 Series 930, was to provide an MPE environment that would enable an unmodified object code program for previous HP 3000 designs—those running MPE V—to be loaded and run successfully on the new HP Precision Architecture computer family. For this, a compatibility mode environment has been included in the MPE XL operating system. This environment uses two subsystems, the HP 3000 Emulator and the HP 3000 Object Code Translator (OCT), to provide program execution that duplicates that of MPE V on the non-HP-Precision-Architecture HP 3000s.

In this paper, we will refer to the pre-HP-Precision-Architecture HP 3000 as the *stack-3000*. Accurate reproduction of the stack-3000 behavior is accomplished by implementing a stack environment in MPE XL and having the stack-3000 object code operate in this compatibility mode environment. The architecture of the new computers is particularly well-suited to support this task and both the Emulator and the OCT operate on this stack-3000 data space.

The two software subsystems are provided to satisfy two different needs. The Emulator duplicates the behavior of the individual stack-3000 hardware instructions with all bounds tests, error handling, etc. implemented in software so that identical program behavior is established. The Translator creates new code modules that use hardware page protection and improved code generation to provide significantly better performance, but may have slightly different error handling characteristics.

The Emulator enables a user program to be loaded and run without preparation or modification on an HP Precision Architecture computer with the normal MPE commands. In the case of the OCT, specific action by the user is required to produce a translated program before it can be run in the improved-performance translated mode. This involves no code modification but only a compiler-like invocation of the OCT for translation, after which the target program will automatically execute translated.

HP 3000 Execution Validity

Confidence in the ability to duplicate the behavior of a stack-3000 on an HP Precision Architecture computer stems from three factors. First is the experience of the development team. The team developing the Emulator and OCT had written the microcode and diagnostics for a previous HP 3000 implementation, and had several years experience developing microcode and diagnostics for this complex instruction set type of architecture. In addition, the team leader was a principal in the original design of the stack-3000.

Second, both the Emulator and the OCT were tested with an extensive set of diagnostics. These diagnostics were assembled from diagnostics that had accumulated over the fourteen-year life of the HP 3000 and were used to verify consistency from model to model.

The third equally important factor relates to performance in actual use. The compatibility mode parts of the operating system have been using the Emulator for over two years, and have found very few bugs and no functional problems—a strong testimonial to the value of good diagnostics and experience.

Compatibility Mode Environment

The compatibility mode (CM) environment exists in the virtual address space of the HP Precision Architecture native mode (NM) environment. Each process has a CM stack and program object code, and a CM globals table defines the environment for each compatibility mode program. If the program has been translated, that native code also will be included in that program environment.

The CM operating system environment includes the Emulator for executing programs that have not been translated, operating system code that has been ported from MPE V, and the MPE V operating system structures required by the Emulator and OCT to execute stack-3000 object code.

These CM operating system structures are: a system code segment table (CST), which contains entries for all the MPE V operating system code and libraries, the user code segment tables (CSTX), which contain entries for the user program code segments, the data segment table (DST), which contains entries for all the stacks and extra data segments controlled by privileged code, and the absolute object used to simulate absolute memory on a stack-3000.[1]

Functional Differences

The actions of all user mode programs remain the same as on a stack-3000, but a number of privileged mode functions are available to software on the earlier machines that either are nonexistent in CM on HP Precision Architecture computers or must be provided by alternate means.

The major example is the I/O system, where all direct contact with the new physical I/O has been moved to native mode, making CM I/O instructions meaningless. Similarly, memory manager and dispatcher functions have been moved to native mode, so these instructions are not supported. Control of the interrupt system has been removed from CM and direct reading and setting of the DB and bank environment registers also have been removed since they now deal in HP Precision Architecture virtual space. All privileged mode instructions in these categories now trap to an unimplemented instruction trap.

Also in these categories are three user mode exceptions: read switches (RSW), read I/O mask (RMSK), and read clock (RCLK). On a stack-3000 computer RCLK reads process clock, not time. Because user mode programs must run without any unexpected traps, no user should be using these instructions since their use will cause an unimplemented trap.

A number of privileged mode instructions on a stack-3000 deal with manipulating data in physical memory. A special absolute object has been set up in MPE XL virtual space to represent this physical memory, and the data used by these instructions has been located in this space. This allows these instructions to continue to be functional and has avoided the need for rewriting considerable portions of CM operating system code.

Emulator

In MPE XL, the Emulator performs the functions served by microcode on a stack-3000. The basic operating environment of the Emulator, as shown in Fig. 1, includes the CM stack and code objects and the Emulator.

When operating in compatibility mode, some HP Precision Architecture hardware registers are reserved for the CM stack and code registers designated in Fig. 1 and the function of microcode is replaced by the Emulator. Direct control of the registers and the instruction execution sequence is required for valid CM program behavior, and for this reason the Emulator is written in HP Precision Architecture assembly code.

In the execution of a stack-3000 instruction, program flow starts in the fetch module. This module fetches the instruction from location P in the CM object code and extracts the high-order ten bits, which contain opcode and memory reference addressing information. It extracts the low-order six bits, saves them in a register, and increments P to the next CM instruction address. The fetch routine then uses the high-order ten bits to vector into one of 1024 locations in the mapping table module of the Emulator.

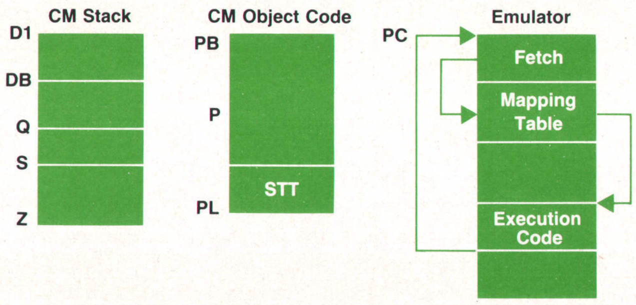


Fig. 1. *The basic operating environment of the Emulator includes the compatibility mode stack and code objects and the Emulator.* PC *is the native mode program counter.*

Each entry in the mapping table contains a branch instruction that branches to the execution module that implements the action called for in the high-order ten bits of the CM instruction. If, for example, this were a Q register relative load instruction, the branch target would be a routine that uses the previously saved low-order six bits of the instruction to assemble an address displacement. This displacement would be added to the address in the Q register to produce an address of the desired data. This data then would be fetched and pushed onto the stack. A push involves incrementing the S register to the next sixteen-bit data location and storing the fetched data at this address. The final action of this routine would be to branch back to the fetch module to complete the instruction cycle.

By following the P register instruction sequencing, which includes branches and PCALs to other code segments and returns, the desired CM program behavior is achieved.

As might be expected, some interesting problems were encountered. On a stack-3000, conditional branching is based on hardware controlled condition code, overflow, and carry bits that reside in a status register. HP Precision Architecture has no similar hardware indicators, but does have an instruction that branches on the results of a comparison of the contents of two registers, or of one register with zero. The problem of implementing condition codes was resolved by reserving two registers to contain data generated by all CM instructions that set condition codes. Branch on condition then can be emulated with a branch on register comparison. Similarly, a register is reserved for carry. A bit in a status register is used for overflow, where an HP Precision Architecture branch on bit instruction can be used.

Emulator Performance

Writing the Emulator in machine assembly code made it possible to take advantage of architectural performance features and minimize possible cache delays. HP Precision Architecture provides a delayed branch feature that allows the execution of an additional instruction following a branch instruction. An instruction can also be executed during a cache fetch cycle. Careful attention has been paid to these features in writing the Emulator.

Instruction frequency mix data taken on stack-3000 computers indicates that thirty-five of the nearly two hundred instructions account for eighty-five percent of the activity. Nearly twenty percent of the activity occurs in the load instruction alone. Such repetition drastically increases emulator locality and thus reduces the cache and TLB miss ratios, yielding enhanced performance.

At the cost of expanded code size, some heavily used stack-3000 instructions include the fetch routine in the body of their associated Emulator instruction execution routine. This allows the elimination of a branch instruction and permits the interleaving of the fetch function with execution code to produce an additional improvement in performance.

Use of the instruction frequency mix allows a first-level prediction of Emulator performance. By counting the instruction path lengths (in terms of machine clocks) of the top ninety-plus instructions and weighting these in terms of the mix percentages, an average instruction clock length

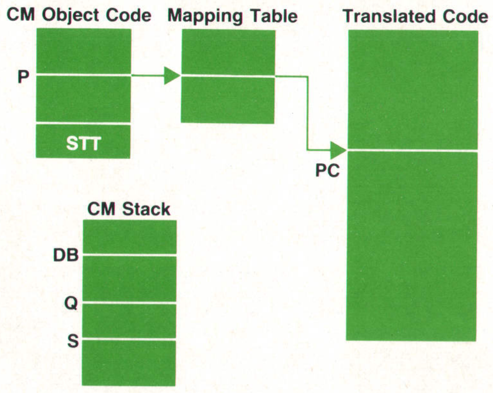


Fig. 2. *Operating environment with translated code.*

can be computed.

The cache and TLB miss penalties can be estimated by separating the sequence into three categories. The fetch module is used so often that its penalties can be assumed to be zero. Measurements of microcode behavior on a stack-3000 show that average miss penalties of the mapping and execution sections will be one tenth of the normal program miss penalties. The instruction fetch and stack data manipulation will encounter the normal program miss penalties. Weighting all these factors, an overall average instruction clock length can be calculated. Dividing this by the number of clocks per instruction in native mode programs provides the adjusted instructions per average stack-3000 instruction. The final number derived from this sequence of computations is about 12.

Object Code Translator

The Object Code Translator (OCT) essentially performs the role of a compiler whose source language is CM object code and whose output is HP Precision Architecture object code. The data manipulated by this code is still the CM stack and data structures, but the Emulator is replaced by the native code emitted by the OCT. The operating environment with translated code is as shown in Fig. 2.

A translated native HP Precision Architecture code module is generated for each CM code segment. Entry to this code is accomplished through the PCAL (procedure call) or EXIT mechanism. The execution of this native code manipulates the CM data space to produce the required stack-3000 behavior. Some complex instructions like PCAL, EXIT, and the floating-point instructions use code sequences that reside in the system and are shared between the Emulator and the OCT. These sequences have been assembly coded by hand to optimize execution speed.

Code sequences are occasionally encountered that use data dependent branching, and the target cannot be determined at translation time. For these cases, a node mapping table is generated which contains the translated code address corresponding to an address in the stack-3000 code segment. Not all locations in the stack-3000 code segment have corresponding locations in translated code. The translator attempts to identify all of the node points in the program, and inserts their addresses into the node mapping table.

It is possible that the target of a branch—an indexed branch, for example—may not be a node point. In this case, the valid stack-3000 state is reconstructed and control is transferred to the Emulator where processing continues until a convenient return point in the program is reached. To allow for this transfer to emulation mode, the OCT includes an image of the stack-3000 object file in the translated output. An indexed load from the program segment may also rely on the presence of this code object for its source of data. The XEQ instruction, which interprets the data on the top of the stack as an instruction, is another example of an instruction that often requires transfer from translated to emulated mode.

OCT Performance

Since the OCT is able to preprocess the object code, optimizations can be done that are not possible with the Emulator. One of the major optimizations is the elimination of the instruction fetch and mapping sequences of the Emulator. The software used in the emulator for environment protection is also eliminated by using hardware page protection. Another unnecessary overhead in most instructions is the storing of data required for condition codes. By finding the last instruction to set the condition code before a test, the setup in all other instructions is eliminated. Optimization techniques are applied to code between nodes such as using registers in place of the stack.

The OCT recognizes commonly executed code sequences and replaces these with HP Precision Architecture code sequences that perform the same operation more efficiently. In a few specific cases, entire procedures are replaced, thus eliminating the PCAL and EXIT overhead.

The overall effect of these techniques, plus others, is translated code performance that is about two to five times that of the Emulator. Instruction mix and code locality will affect this ratio.

Acknowledgments

The early prototypes of the Emulator and Object Code Translator were done at HP Laboratories with the aid of Carol Hammond Thompson. The project was subsequently transferred to the MPE XL operating system group and developed by a team that includes John Figueroa, Tom Graham, Dan Lee, Charlie Ehlman, and Tony Hunt.

Reference

1. B.E. Forbes and Michael D. Green, "An Economical Full-Scale Multipurpose Computer System," *Hewlett-Packard Journal*, Vol. 24, no. 5, January 1973, pp. 2-8.

Index

HEWLETT·PACKARD JOURNAL
Volume 38 January 1987 through December 1987

Hewlett-Packard Company, 3200 Hillview Avenue, Palo Alto, California 94304 U.S.A.
Hewlett-Packard Central Mailing Dept., P.O. Box 529, Startbaan 16, 1180 AM Amstelveen, The Netherlands
Hewlett-Packard (Canada) Ltd., 6877 Goreway Drive, Mississauga, Ontario L4V 1M8 Canada
Yokogawa-Hewlett-Packard Ltd., Suginami-ku, Tokyo 168 Japan

PART 1: Chronological Index

January 1987
Low-Frequency Analyzer Combines Measurement Capability with Modeling and Analysis Tools, *Edward S. Atkinson, Gaylord L. Wahl, Jr., Michael L. Hall, Eric J. Wicklund, and Steven K. Peterson*
Applications
Measurement Modes and Digital Demodulation for a Low-Frequency Analyzer, *Raymond C. Blackham, James A. Vasil, Edward S. Atkinson, and Ronald W. Potter*
Demodulation Example
Analyzer Synthesizes Frequency Response of Linear Systems, *James L. Adcock*
Curve Fitter for Pole-Zero Analysis, *James L. Adcock*
Performance Analysis of the HP 3000 Series 70 Hardware Cache, *James R. Callister and Craig W. Pampeyan*

February 1987
A New Family of Precise, Reliable, and Versatile Fiber Optic Measurement Instruments, *Michael Fleischer-Reumann*
A Color-Coding Scheme for Fiber Optic Instruments and Accessories
Stable LED Sources for a Wide Range of Applications, *Michael Fleischer-Reumann*
An Accurate Two-Channel Optical Average Power Meter, *Horst Schweikardt*
Optical Power Meter Firmware Development, *Bernhard Flade and Michael Goder*
Detectors for Optical Power Measurements, *Josef Becker*
Precision Optical Heads for 850 to 1700 and 450 to 1020 Nanometers, *Hans Huning, Emmerich Müller, Siegmar Schmidt, and Michael Fleischer-Reumann*
Optical Power Splitter
A High-Precision Optical Connector for Optical Test and Instrumentation, *Wilhelm Radermacher*
Design Approach for a Programmable Optical Attenuator, *Bernd Maisenbacher, Siegmar Schmidt, and Michael Schlicker*
A Programmable Fiber Optic Switch, *Michael Fleischer-Reumann*
Quality Microwave Measurement of Packaged Active Devices, *Glenn E. Elmore and Louis J. Salz*
HP 8510 Software Signal Processing

March 1987
Hardware Design of the First HP Precision Architecture Computers, *David A. Fotland, John F. Shelton, William R. Bryg, Ross V. La Fetra, Simin I. Boschma, Allan S. Yeh, and Edward M. Jacobs*
An Automated Test System for the First HP Precision Architecture Computers, *Thomas B. Wylegala, Long C. Chow, and Randy J. Teegarden*
A Distributed Terminal Controller for HP Precision Architecture Computers Running the MPE XL Operating System, *Gregory F. Buchanan, François Gaullier, Olivier Krumeich, Eric Lecesne, Jean-Pierre Picq, and Heng V. Te*
Hewlett-Packard Precision Architecture Compiler Performance, *Karl W. Pettis and William B. Buzbee*
Viewpoints—A Viewpoint on Calculus, *Zvonko Fazarinc*

April 1987
Digital Signal Generator Combines Digital and Analog Worlds, *Uwe Neumann, Michael Vogt, Friedhelm Brilhaus, and Frank Husfeld*
User Interface and Software Architecture for a Data and Arbitrary Waveform Generator, *Ulrich Hakenjos, Wolfgang Srok, and Rüdiger Kreiser*
A Planning Solution for the Semiconductor Industry, *Edward L. Wilson, Kelly A. Sznaider, and Clemen Jue*
A Study of Panel Deflection of Partially Routed Printed Circuit Boards, *George E. Barrett and John H. Lau*
Deflections, Forces, and Moments of a Printed Circuit Board
Reliability Theory Applied to Software Testing, *H. Dean Drake and Duane E. Wolting*
Appendix: Derivation of the Software Reliability Model

May 1987
State-of-the-Art CAD Workstations for Mechanical Design, *Wolfgang Kurz, Dieter Sommer, Karl-Heinz Werner, Dieter Deyke, and Heinz P. Arndt*
Example Macro
ME Series 10 Link to HP-FE
The ME Series 10 NC Links
ME CAD Geometry Construction, Dimensioning, Hatching, and Part Structuring, *Karl-Heinz Werner, Stephen Yie, Friedhelm Ottliczky, Harold B. Prince, and Heinz Diebel*
Alpha Site Evaluation of ME Series 5/10, *Paul Harmon*
Intrabuilding Data Transmission Using Power-Line Wiring, *Robert A. Piety*

June 1987
Permuted Trace Ordering Allows Low-Cost, High-Resolution Graphics Input, *Thomas Malzbender*
The Hewlett-Packard Human Interface Link, *Robert R. Starr*
HP-HIL Input Devices
Software Verification Using Branch Analysis, *Daniel E. Herington, Paul A. Nichols, and Roger D. Lipp*
Advantages of Code Inspections
Viewpoints—Direction of VLSI CMOS Technology, *Yoshio Nishi*
Software Architecture and the UNIX Operating System: An Introduction to Interprocess Communication, *Marvin L. Watkins*
Benchmarking UNIX IPC Facilities

July 1987
Dedicated Display Monitors Digital Radio Patterns, *David J. Haworth, John R. Pottinger, and Murdo J. McKissock*
Automated Timing Jitter Testing
Constellation Measurement: A Tool for Evaluating Digital Radio, *Murdo J. McKissock*
A Digital Radio Noise and Interference Test Set, *Geoffrey Waters*
Noise Crest Factor Enhancement
Noise Bandwidth Measurement
Microprocessor-Enhanced Performance in an Analog Power Meter, *Anthony Lymer*
An Accurate Wideband Noise Generator and a High-Stability Reference Source, *Dayananda K. Rasaratnam*

General-Purpose Wideband Thick-Film Hybrid Amplifier
Automated Radio Testing Shortens Test Time and Enhances Accuracy, *John A. Duff*
A Reusable Screen Forms Package

August 1987
A Handheld Business Consultant, *Susan L. Wechsler*
Cash Flow Analysis Using the HP-18C
The Equation Solver Menu in the HP-18C
History and Inspiration of the Solve Interface
An Evolutionary RPN Calculator for Technical Professionals, *William C. Wickes*
Example Problem
HP-28C Plotting
Mechanical Design of the HP-18C and HP-28C Handheld Calculators, *Judith A. Layman and Mark A. Smith*
Symbolic Computation for Handheld Calculators, *Charles M. Patton*
A Multichip Hybrid Printed Circuit Board for Advanced Handheld Calculators, *Bruce R. Hauge, Robert E. Dunlap, Cornelis D. Hoekstra, Chong Num Kwee, and Paul R. Van Loan*
An Equation Solver for a Handheld Calculator, *Paul J. McClellan*
Electronic Design of An Advanced Technical Handheld Calculator, *Preston D. Brown, Gregory J. May, and Megha Shyam*

September 1987
A VLSI Processor for HP Precision Architecture, *Steven T. Mangelsdorf, Darrell M. Burns, Paul K. French, Charles R. Headrick, and Darius F. Tanksalvala*
Pin-Grid Array VLSI Packaging
HP Precision Architecture NMOS-III Single-Chip CPU, *Jeffry D. Yetter, Jonathan P. Lotz, William S. Jaffe, Mark A. Forsyth, and Eric R. DeLano*
Execution Unit
A Precision Clocking System
Design, Verification, and Test Methodology for a VLSI Chip Set, *Charles Kohlhardt, Tony W. Gaddis, Daniel L. Halperin, Stephen R. Undy, and Robert A. Schuchard*
VLSI Test Methodology
A Midrange VLSI Hewlett-Packard Precision Architecture Computer, *Craig S. Robinson, Leith Johnson, Robert J. Horning, Russell W. Mason, Mark A. Ludwig, Howell R. Felsenthal, Thomas O. Meyer, and Thomas V. Spencer*
VLSI-Based High-Performance HP Precision Architecture Computers, *Gerald R. Gassman, Michael W. Schrempp, Ayee Goundan, Richard Chin, Robert D. Odineal, and Marlin Jones*

October 1987
In-Service Transmission Impairment Testing of Voice-Frequency Data Circuits, *Norman Carder, William I. Dunn, James H. Elliott, David W. Grieve, and W. Gordon Rhind*
Processing Passband Signals in Baseband
LMS Algorithm for Equalizer Update
Digital Phase-Locked Loops
An Infrared Link for Low-Cost Calculators and Printers, *Steven L. Harper, Robert S. Worsley, and Bruce A. Stephens*
A Low-Cost Wireless Portable Printer, *David L. Smith and Masahiko Muranami*
Manufacturing State-of-the-Art Handheld Calculators, *Richard W. Riper*
Information Technology and Medical Education, *G. Octo Barnett, M.D., Judith L. Piggins, Gordon T. Moore, M.D., and Ethan A. Foster*
A Framework for Program Development, *Derek Coleman and Robin M. Gallimore*

December 1987
Vector Signal Generation and Analysis, *Allen P. Edwards*
Hardware System Design for a Vector Analyzer, *Andrew H. Naegeli and Juan Grau*
Quadrature and Phase Errors in Vector Demodulation
Firmware System Design for a Vector Analyzer, *Brian S. Messenger, Peter H. Fisher, and Stanley P. Woods*
Vector Modulation in a Signal Generator, *David L. Gildea and Donald R. Chambers*
Firmware for a Vector Signal Generator, *James E. Jensen and Eric D. McHenry*
Low-Noise Synthesizer Design, *Thomas J. Carey, John C. Lovell, and Thomas L. Grisell*
Digital and Vector Baseband Circuits for a Vector Signal Generator, *Chung Y. Lau*
A GaAs IC Current Switch
Describing Signals in the I-Q Domain
A Wideband FM Subsystem for a Low-Noise Synthesizer Module, *Eric D. McHenry*
Vector Modulator, Output Amplifier, and Multiplier Chain Assemblies for a Vector Signal Generator, *Wayne M. Kelly, Mark J. Woodward, Eric B. Rodal, Pedro A. Szente, and James D. McVey*
Baseband Calibration
A Combinational Board Test System, *Michael E. Gravitz*
Interactive Graphical Debugging Package
MPE XL: The Operating System for HP's Next Generation of Commercial Computer Systems, *John R. Busch, Alan J. Kondoff, and Darryl Ouye*
HP 3000 Emulation on HP Precision Architecture Computers, *Arndt B. Bergh, Keith Keilman, Daniel J. Magenheimer, and James A. Miller*

PART 2: Subject Index

Subject	Month
A	
Abstraction function	Oct.
Accelerating factor, software test	Apr.
Access rights	Dec.
Active device measurements	Feb.
Adaptive enhancer	Oct.
Adaptive equalizer	Oct.
Algorithm, LMS	Oct.
Algorithm, motor control	Feb.
Algorithm, timing, infrared link	Oct.
Alignment, optical connector	Feb.
Alpha site testing, ME Series 5/10	May
Amplifier, output, vector signal generator	Dec.
Amplifier, transimpedance	Feb.
Amplifier, wideband, thick-film	July
Amplitude distortion measurements	Oct.
Analyzer, low-frequency	Jan.
Analyzer, vector	Dec.
Arbitrary waveform generator	Apr.
Architecture, HP-HIL	June
Architecture, HP Precision	Mar. Sept. Dec.
Architecture, UNIX	June
Associativity, cache	Mar.
Attenuation measurements	Oct.

Attenuators, optical .... Feb.
Autocalibration, low-frequency
analyzer .... Jan.
Autolearn module, board testing ...... Dec.
Automath processing .... Jan.
Autosequence, low-frequency
analysis .... Jan.

B

Baseband circuitry, vector signal
generator .... Dec.
Baseband processing .... Oct.
Benchmark testing .... Jan.
Benchmarking, UNIX .... June
Binning, multiple parts
manufacture .... Apr.
Bolometer .... Feb.
Branch analysis .... June
B-trees .... Dec.
Bus converter .... Sept.
Bypassing, cache .... Mar.

C

Cache bus .... Sept.
Cache controller .... Sept.
Cache design .... Sept.
Cache design and performance .... Mar.
Cache simulation .... Jan.
CAD, mechanical engineering
workstation .... May
CAD, portable printer .... Oct.
Calculator manufacturing .... Oct.
Calculus, Viewpoints .... Mar.
Calendars, planning data base .... Apr.
Calibration firmware .... Dec.
Calibration, optical power .... Feb.
Calibration plane .... Feb.
Cash flow analysis, HP-18C .... Aug.
Channel adapter .... Sept.
Channel model .... Oct.
Characterization, chips and boards . Sept.
CIO bus .... Sept.
Circuit board testing .... Dec.
Clock generator, 2× .... Sept.
Clocking system .... Sept.
Closed-loop system design .... Jan.
CMOS, VLSI, Viewpoints .... June
Code expansion .... Mar.
Code inspections .... June
Coding, infrared transmission .... Oct.
Color coding, fiber optic
instruments .... Feb.
Combinational board tester .... Dec.
Compatibility mode .... Dec.
Compiler performance, HP
Precision Architecture .... Mar.
Compile-time information .... Mar.
Complex operations .... Mar.
Component level testing .... June
Computers, HP Precision
Architecture .... Mar.
Sept.
Configurator, MPE XL .... Dec.
Connector, optical .... Feb.
Constellation diagram .... Oct.
Dec.
Constellation display .... July
Constellation measurement, digital
radio evaluation .... July
Context-sensitive keys .... Aug.
Conveyor, final assembly .... Oct.
Coprocessor, floating point .... Mar.
Sept.
Coupled environment .... Dec.
Coupling loss, optical fibers .... Feb.
CPU, NMOS-III VLSI .... Sept.
Crest factor, effect on symbol error .. July
Critical word first .... Mar.
Curve fitter, low-frequency analysis .. Jan.
Cycles per instruction (CPI) .... Mar.

D

Data cache and TLB .... Mar.
Data circuit testing .... Oct.
Data generator .... Apr.
Data logging, circuit board testing .... Dec.
Data logging, ITIMS .... Oct.
Data management, MPE XL .... Dec.
Data transmission, intrabuilding ...... May
Data type nodes .... Oct.
Debugger, symbolic .... Dec.
Debugging, interactive graphical
testing .... Dec.
De-embedding .... Feb.
Deflection, printed circuit board ..... Apr.
Delay distortion measurements .... Oct.
Delay element .... Sept.
Demodulation, digital .... Jan.
Demodulation, quadrature and
phase errors .... Dec.
Demodulation, vector signal .... Dec.
Design graph .... Oct.
Detectors, optical .... Feb.
Device control protocol .... Mar.
Diagnostics, MPE XL .... Dec.
Diamond® HMS-10/HP Connector .... Feb.
Digital modulation .... July
Dec.
Digital phase-locked loops .... Oct.
Dec.
Digital radio, instrumentation .... July
Oct.
Digital radio, vector generation
and analysis .... Dec.
Digital signal generator .... Apr.
Dimensioning, ME Series 5/10 .... May
Direct mapped cache .... Mar.
Direct solver .... Aug.
Display driver, handheld
calculator .... Aug.
Display refresh control .... Dec.
Display, vector graphics generator .. July
Distributed terminal controller .... Mar.
Drafting software, ME Series 5/10 ..... May
Dump analysis tool .... Dec.

E

Education, calculus .... Mar.
Education, medical .... Oct.
EEPROM, optical head .... Feb.
Emulator, MPE V .... Dec.
Enhancer, adaptive .... Oct.
Equalizer, adaptive .... Oct.
Equation solver, handheld
calculator .... Aug.
Error adapter .... Feb.
Error correcting memory .... Mar.
Error correction, cache .... Sept.
Error correction, network
analyzer .... Feb.
Error handling, MPE XL .... Dec.
Estimating software reliability .... Apr.
Execution-time theory .... Apr.
Execution unit .... Mar.
Sept.
Explode and roll forward planning . Apr.
Extension, software design .... Oct.

F

FFT analyzer, two-channel .... Jan.
Fiber optic test instruments .... Feb.
File system, MPE XL .... Dec.
Files, mapped .... Dec.
Filter design, examples .... Jan.
Financial calculator .... Aug.
Finite element analysis,
link, ME Series 5/10 .... May
Finite element analysis,
panel deflection .... Apr.
Firmware, vector analyzer .... Dec.
Firmware, vector signal
generator .... Dec.
Fixture, microwave transistor .... Feb.
Floating-point coprocessor .... Mar.
Sept.
Flow control protocol .... Mar.
FM, wideband subsystem .... Dec.
Formal methods, software
design .... Oct.
Forms, screen, reusable .... July
Framework for program
development .... Oct.
Frequency response synthesis .... Jan.
Frozen cycles .... Jan.
Functional testing, circuit boards ..... Dec.
Functional testing, software .... June

G

Geometry construction,
ME Series 5/10 .... May
Grant program, HP, medical
education .... Oct.
Graphics tablet .... June

H

Handheld calculators .... Aug.
Harvard New Pathway
curriculum .... Oct.
Hatching, ME Series 5/10 .... May
Hilbert transform filters .... Oct.
Hinge link, handheld calculator ...... Aug.
Hit rates, cache .... Jan.
Mar.
HP-18C calculator .... Aug.
HP-28C calculator .... Aug.
HP 3000 Series 930 Computer .... Mar.

Dec.
HP 3000 Series 950 Computer .... Sept.
Dec.
HP 3000 Series 825S Computer ...... Sept.
HP 9000 Model 840 Computer .... Mar.
HP 9000 Model 850S Computer ..... Sept.
HP DesignCenter, ME Series 5/10 ..... May
HP-FE, link with ME Series 10 .... May
HP-HIL .... June
HP-IB firmware, vector analyzer .... Dec.
HP Precision Architecture .... Mar.
Sept.
Dec.
Hybrid circuit, handheld
calculator .... Aug.
Hybrid circuit, wideband amplifier .. July

I

Identifiers, protection .... Dec.
IEEE 488.2, instrument .... Dec.
Impairments, data circuit .... Oct.
In-circuit testing .... Dec.
Infrared printer .... Oct.
Input devices, HP-HIL .... June
Input/output, MPE XL .... Dec.
In-service transmission impairment
measurements .... Oct.
Instantiation, software design .... Oct.
Instruction cache and TLB .... Mar.
Instruction gatherer .... Jan.
Instruction unit .... Mar.
Interface, user, CAD .... May
Interference testing, digital radio ...... July
Interprocess communication, UNIX . June
Interrupt processing .... Dec.
Invariant predicate .... Oct.
I-Q splitter .... Dec.
Iterative solver .... Aug.
ITIMS .... Oct.

J

Jitter, low-level one-shot circuit .... July
Jitter, automated testing .... July

K

Kernel, MPE XL .... Dec.

L

LAN, power-line .... May
Language, MUMPS .... Oct.
Language, RPL .... Aug.
Latency, terminal controller .... Mar.
Launch plane .... Feb.
LCD assembly, calculator .... Oct.
LCD interconnect .... Aug.
Oct.
LED sources .... Feb.
Line size, cache .... Mar.
Links, ME Series 5/10 .... May
LMS algorithm .... Oct.
Localization, calculator .... Aug.
Lock manager .... Dec.
Log manager .... Dec.
Logging, transmission impairments .. Oct.
Looping, program-controlled .... Apr.
Low-noise synthesizer .... Dec.

M

Macros, MPE XL .... Dec.
Mail, electronic, medical
education .... Oct.
Managers, MPE XL .... Dec.
Manufacturing, calculator .... Oct.
Manufacturing, semiconductor,
planning tool .... Apr.
Mapped files .... Dec.
Math interface unit .... Sept.
Math subsystem .... Sept.
Measurement modes, low-frequency
analysis .... Jan.
Measurement, noise bandwidth .... July
Measurement, phase and amplitude
distortion .... Oct.
Measurement plane .... Feb.
Measurement tools, computer
performance .... Jan.
Measurement, transient .... Oct.
Mechanical design, HP-18C and
HP-28C .... Aug.
Mechanical design software .... May
Medical education .... Oct.
Memory bus .... Sept.
Memory controller .... Sept.
Memory, data pattern .... Apr.
Memory management, ME CAD .... May
Memory management, MPE XL .... Dec.
Memory subsystem .... Sept.
Memory system .... Mar.
ME Series 5/10 .... May
Message system, MPE XL .... Dec.
Methodology, verification
and test .... Sept.
Metrics, software testing .... June
Microsampler .... Jan.
Microwave transistor
measurements .... Feb.
MidBus .... Sept.
Millicode .... Mar.
Dec.
MIPS .... Mar.
Miss, cache .... Sept.
Miss penalty, cache .... Mar.
Model filter .... Oct.
Model, printed circuit board .... Apr.
Model, transistor fixture .... Feb.
Modeling, computer .... Jan.
Modem receiver .... Oct.
Modulation, digital .... July
Oct.
Dec.
Modulation, vector .... July
Dec.
Monitor, system .... Mar.
Motor control algorithm .... Feb.
MPE XL operating system .... Dec.
MPE V emulation .... Dec.
Multiplier chain, vector signal
generator .... Dec.
MUMPS language .... Oct.

N

Namespace resolution .... Dec.
Native mode .... Dec.
NC links, ME Series 10 .... May
Network analyzer transistor
measurements .... Feb.
Network management protocol .... Mar.
New Pathway, medical curriculum .. Oct.
NMOS-III processor, HP Precision
Architecture .... Sept.
Noise bandwidth measurement .... July
Noise crest factor enhancement .... July
Noise generator, wideband .... July
Noise measurement, digital radio ..... July

O

Object code translator .... Dec.
Operating system, MPE XL .... Dec.
Operating system, UNIX IPC .... June
Optical attenuators .... Feb.
Optical connector .... Feb.
Optical filters, fixed and variable ..... Feb.
Optical heads .... Feb.
Optical interface .... Feb.
Optical power meter .... Feb.
Optical power splitter .... Feb.
Optical sources .... Feb.
Optical switch .... Feb.
Oscillator, delay line .... Apr.
Overlapped processing .... Mar.

P

Package design, handheld
calculator .... Aug.
Package design, portable printer .... Oct.
Packaged device measurements .... Feb.
Packaging, VLSI .... Sept.
Panel deflection, printed circuit ...... Apr.
Part structures, ME Series 5/10 .... May
Patient simulation .... Oct.
Pattern data capture .... Dec.
Pattern generator .... Apr.
Paused cycles .... Jan.
Performance, cache .... Jan.
Performance, compilers, HP Precision
Architecture .... Mar.
Performance, HP 9000 Model 840 ... Mar.
Performance, terminal controller ..... Mar.
Performance, UNIX IPC facilities ..... June
Permuted trace ordering,
graphics tablet .... June
Phase distortion measurements .... Oct.
Photodiodes .... Feb.
Physiology simulation .... Oct.
Pin-grid array .... Sept.
Pipeline, CPU .... Sept.
Pipelines, processor and TLB .... Mar.
PL-10 .... Apr.
Planning module, IC manufacturing . Apr.
Plotting, HP-28C .... Aug.
Pole-zero analysis .... Jan.
Ports, MPE XL .... Dec.
Posting, disc .... Dec.
Power-line characteristics,
data transmission .... May
Power meter, analog .... July
Power meter, optical .... Feb.
Power splitter, optical .... Feb.
Precharge/pulldown bus .... Sept.
Primitives, software design .... Dec.

Printed circuit board construction .... Sept.
Printed circuit board deflection .... Apr.
Printer, portable, infrared .... Oct.
Privilege levels .... Dec.
Procedural language, RPL .... Aug.
Procedure calling .... Mar.
Procedure calls .... Dec.
Procedure level testing .... June
Process management, MPE XL .... Dec.
Processor, VLSI, HP Precision Architecture .... Sept.
Program development framework .... Oct.
Program nodes .... Oct.
Program specifications .... Oct.
Programming, data generator .... Apr.
Property specifications .... Oct.
Protection, MPE XL .... Dec.
Protocol, HP-HIL .... June
Protocols, terminal control .... Mar.
Pyroelectric detector .... Feb.

Q

Quantum detectors .... Feb.
Queuing theory .... Mar.

R

Radio, digital, measurements .... July
Radio, digital, signal generation and analysis .... Dec.
Realizations .... Oct.
Recovery manager .... Dec.
Reference loops, RF and IF .... Dec.
Reference source, high-stability .... July
Register file board .... Mar.
Reliability, software .... Apr.
Remote maintenance protocol .... Mar.
Responsivity, optical detectors .... Feb.
Rigorous software development .... Oct.
RMS thermal converter .... July
Robotic loading .... Oct.
ROM, 512K-bit, handheld calculator .... Aug.
RPL, ROM-based procedural language .... Aug.
RPN calculator .... Aug.

S

Samplers, vector analyzer .... Dec.
Scan paths .... Sept.
Screen forms, reusable .... July
Servo design, example .... Jan.
Signal generator, digital .... Apr.
Signal generator, vector .... Dec.
Simulation, patient .... Oct.
Simulation, physiology .... Oct.
Simulator, cache .... Jan.
Size, cache .... Mar.
Snapshot, HP 3000 measurement tool . Jan.
Software, board testing .... Dec.
Software design .... Oct.
Software reliability .... Apr.
June
Software testing .... Apr.
Software, UNIX IPC .... June
Software verification .... June
Solve Interface, HP-18C .... Aug.
S-plane calculator .... Jan.
Splines .... Apr.
Splitter, optical power .... Feb.
SPN, Semiconductor Productivity Network .... Apr.
Spread spectrum, data transmission . May
SPU design, HP Precision Architecture .... Sept.
Structural testing, software .... June
Stubs, MPE XL .... Dec.
Stylus design, graphics tablet .... June
Sum loop, low-noise synthesizer .... Dec.
Support, MPE XL .... Dec.
Surface mount, deflection effects .... Apr.
Switch, MPE XL .... Dec.
Switch, optical .... Feb.
Symbolic computation .... Aug.
Symbolic debugger, MPE XL .... Dec.
Symbolic entry, handheld calculator .... Aug.
Synthesis, low-frequency response .... Jan.
Synthesizer, low-noise .... Dec.
System clock .... Sept.
System interface unit .... Sept.
System level testing .... June
System main bus (SMB) .... Sept.

T

Tables, standard .... Dec.
Team testing, software .... June
Temperature control, optical detector .... Feb.
Terminal controller, MPE XL .... Mar.
Test methodology, VLSI .... Sept.
Test system, HP Precision Architecture computers .... Mar.
Testing, additive interference .... July
Testing, calculator hybrid circuit .... Aug.
Testing, circuit boards .... Dec.
Testing, computer-based .... Oct.
Testing, computer system .... Jan.
Testing, in-service transmission impairment .... Oct.
Testing, radio, automated .... July
Testing, software .... June
Testing, software, ME Series 5/10 ..... May
Testing, timing jitter .... July
Thermal converter, RMS .... July
Thermal design, processor .... Sept.
Thermal detectors .... Feb.
Thermopile .... Feb.
Thick-film circuit, wideband amplifier .... July
3-D display, vector analysis .... Dec.
Time base, low-noise sysnthesizer ... Dec.
Timing algorithm, infrared link .... Oct.
Timing jitter testing .... July
Timing, variable .... Apr.
TIMS .... Oct.
TLB .... Mar.
TLB controller .... Sept.
Trace ordering, graphics tablet .... June
Transaction management, MPE XL .... Dec.
Transient measurements .... Oct.
Transimpedance amplifier .... Feb.
Transistor measurements, microwave .... Feb.
Transistor test fixture .... Feb.
Translation lookaside buffer .... Mar.
Sept.
Translator, MPE object code .... Dec.
Transmission, data, infrared .... Oct.
Transmission, data, intrabuilding ..... May
Transmission impairment measuring set .... Oct.
Triggering, vector analyzer .... Dec.

U

Unidirectional link .... Oct.
UNIX, interprocess communication .... June
User mapped files .... Dec.

V

Variable stack, HP-28C .... Aug.
Variable timing .... Apr.
Vector address sequencer .... Dec.
Vector analysis .... Dec.
Vector demodulation .... Dec.
Vector graphics generator, constellation display .... July
Vector modulator .... Dec.
Vector signal generation .... Dec.
Verification methodology, software .... June
Verification, VLSI chip set .... Sept.
Video signal processing .... Dec.
Viewpoints: calculus .... Mar.
Viewpoints: VLSI, CMOS .... June
Virtual logic analyzer .... Dec.
Virtual space management, MPE XL .... Dec.
VLSI, CMOS, Viewpoints .... June
VLSI packaging .... Sept.
VLSI processor, HP Precision Architecture .... Sept.
VLSI test methodology .... Sept.
Voice-frequency data circuit testing .... Oct.

W

Walkthroughs .... June
Waveform generator, arbitrary .... Apr.
Workstation, CAD, ME Series 5/10 ... May
Write-to vs. write-through caches .... Mar.

X

Y

Yield, VLSI chip set .... Sept.

Z

Zero-catching receiver .... Sept.

PART 3: Product Index

ME Series 5 ME Drafting Package .... May
ME Series 10 ME Design and Drafting Package .... May
MPE-XL Operating System .... Dec.
PL-10 Integrated Circuit Manufacturing Planning System .... Apr.
HP-18C Business Consultant .... Aug., Oct.
HP-28C Scientific Professional Calculator .... Aug., Oct.
1345A Digital Display .... Dec.
2345A Distributed Terminal Controller .... Mar.
HP 3000 Series 70 Computer .... Jan.
HP 3000 Series 930 Computer .... Mar., Dec.
HP 3000 Series 950 Computer .... Sept., Dec.
3065AT Combinational Board Test System .... Dec.
3562A Dynamic Signal Analyzer .... Jan.
3708A Noise and Interference Test Set .... July
3709A Constellation Display .... July
4948A In-Service Transmission Impairment Measuring Set . Oct.
8152A Optical Average Power Meter .... Feb.
8154B LED Sources .... Feb.
8158B Optical Attenuators .... Feb.
8159A Optical Switch .... Feb.
8175A Digital Signal Generator .... Apr.
8510 Microwave Network Analyzer .... Feb.
HP 9000 Model 825S Computer .... Sept.
HP 9000 Model 840 Computer .... Mar.
HP 9000 Model 850S Computer .... Sept.
35723A HP-HIL Touch Accessory .... June
45911A Graphics Tablet .... June
46021A HP-HIL Keyboard .... June
46030A HP-HIL Keyboard .... June
46060A HP-HIL Mouse .... June
46083A HP-HIL Rotary Control Knob .... June
46084A HP-HIL Security ID Module .... June
46085A HP-HIL Control Dial Module .... June
46086A HP-HIL 32-Button Box .... June
46087A HP-HIL Digitizer .... June
46088A HP-HIL Digitizer .... June
46094A HP-HIL Quadrature Port .... June
81000AS/BS Optical Power Splitter .... Feb.
81520A Optical Head .... Feb.
81521B Optical Head .... Feb.
82240A Infrared Printer .... Aug., Oct.
85014A Active Device Measurements Pac .... Feb.
85041A Transistor Test Fixture .... Feb.
8780A Vector Signal Generator .... Dec.
8980A Vector Analyzer .... Dec.
92916A HP-HIL Bar-Code Reader .... June

PART 4: Author Index

Adcock, James L. .... Jan.
Arndt, Heinz P. .... May
Atkinson, Edward S. .... Jan.
Aziz, Asad .... Sept.

Barnett, G. Octo .... Oct.
Barrett, George E. .... Apr.
Becker, Josef .... Feb.
Bergh, Arndt B. .... Dec.
Blackham, Raymond C. .... Jan.
Boschma, Simin I. .... Mar.
Booth, George .... Dec.
Brilhaus, Friedhelm .... Apr.
Brown, Preston D. .... Aug.
Bryg, William R. .... Mar.
Buchanan, Gregory F. .... Mar.
Bunsen, Chris M. .... Aug.
Burns, Darrell M. .... Sept.
Busch, John R. .... Dec.
Buzbee, William B. .... Mar.

Callister, James R. .... Jan.
Carder, Norman .... Oct.
Carey, Thomas J. .... Dec.
Chambers, Donald R. .... Dec.
Chin, Richard .... Sept.
Chow, Long C. .... Mar.
Coleman, Derek .... Oct.

DeLano, Eric R. .... Sept.
Deyke, Dieter .... May
Diebel, Heinz .... May
Drake, H. Dean .... Apr.
Duff, John A. .... July
Dunlap, Robert E. .... Aug.
Dunn, William I. .... Oct.

Edwards, Allen P. .... Dec.
Eisenstein, Gabe L. .... Aug.
Elliot, James H. .... Oct.
Elmore, Glenn E. .... Feb.

Fazarinc, Zvonko .... Mar.
Felsenthal, Howell R. .... Sept.
Fisher, Peter H. .... Dec.
Flade, Bernhard .... Feb.
Fleischer-Reumann, Michael .... Feb.
Forsyth, Mark A. .... Sept.
Foster, Ethan A. .... Oct.
Fotland, David A. .... Mar.
French, Paul K. .... Sept.

Gaddis, Tony W. .... Sept.
Gallimore, Robin M. .... Oct.
Gassman, Gerald R. .... Sept.
Gaullier, François .... Mar.
Gildea, David R. .... Dec.
Grieve, David W. .... Oct.
Goder, Michael .... Feb.
Goundan, Ayee .... Sept.
Grau, Juan .... Dec.
Gravitz, Michael E. .... Dec.
Grisell, Thomas L. .... Dec.

Hakenjos, Ulrich .... Apr.
Hall, Michael L. .... Jan.
Halperin, Daniel L. .... Sept.
Harmon, Paul .... May
Harper, Steven L. .... Oct.
Hauge, Bruce R. .... Aug.
Haworth, David J. .... July
Headrick, Charles R. .... Sept.
Herington, Daniel E. .... June
Hoekstra, Cornelis D. .... Aug.
Horning, Robert J. .... Sept.
Hug, Berthold .... May
Huning, Hans .... Feb.
Husfeld, Frank .... Apr.

Jacobs, Edward M.	Mar.
Jaffe, William S.	Sept.
Jensen, James E.	Dec.
Johnson, Leith	Sept.
Jones, Marlin	Sept.
Jue, Clemen	Apr.
Keilman, Keith	Dec.
Kelly, Wayne M.	Dec.
Kohlhardt, Charles	Sept.
Kondoff, Alan J.	Dec.
Kreiser, Rüdiger	Apr.
Krumeich, Olivier	Mar.
Kurz, Wolfgang	May
Kwee, Chong Num	Aug.
La Fetra, Ross V.	Mar.
Lau, Chung Y.	Dec.
Lau, John H.	Apr.
Layman, Judith A.	Aug.
Lecesne, Eric	Mar.
Lipp, Roger D.	June
Lotz, Jonathan P.	Sept.
Lovell, John C.	Dec.
Ludwig, Mark A.	Sept.
Lymer, Anthony	July
Magenheimer, Daniel J.	Dec.
Maisenbacher, Bernd	Feb.
Malzbender, Thomas	June
Mangelsdorf, Steven T.	Sept.
Mason, Russell W.	Sept.
Matthews, Ian M.	July
May, Gregory J.	Aug.
McClellan, Paul J.	Aug.
McHenry, Eric D.	Dec.
McKissock, Murdo J.	July
McVey, James D.	Dec.
Messenger, Brian S.	Dec.
Meyer, Thomas O.	Sept.
Miller, James A.	Dec.
Moffatt, John E.	Sept.
Moore, Gordon T.	Oct.
Müller, Emmerich	Feb.
Muranami, Masahiko	Oct.
Naegeli, Andrew H.	Dec.
Neering, Michael	Feb.
Neumann, Uwe	Apr.
Nichols, Paul A.	June
Nishi, Yoshio	June
Odineal, Robert D.	Sept.
Ottliczky, Friedhelm M.	May
Ouye, Darryl	Dec.
Pampeyan, Craig W.	Jan.
Patton, Charles M.	Aug.
Peterson, Steven K.	Jan.
Pettis, Karl W.	Mar.
Picq, Jean-Pierre	Mar.
Piety, Robert A.	May
Piggins, Judith L.	Oct.
Potter, Ronald W.	Jan.
Pottinger, John R.	July
Prince, Harold B.	May
Rhind, W. Gordon	Oct.
Riper, Richard W.	Oct.
Robertson, David	July
Robinson, Craig S.	Sept.
Radermacher, Wilhelm	Feb.
Rasaratnam, Dayananda K.	July
Rodal, Eric B.	Dec.
Rosenbladt, Peter	Sept.
Salz, Louis J.	Feb.
Schlicker, Michael	Feb.
Schmidt, Siegmar	Feb.
Schrempp, Michael W.	Sept.
Schuchard, Robert A.	Sept.
Schweikardt, Horst	Feb.
Shelton, John F.	Mar.
Shyam, Megha	Aug.
Smith, David L.	Oct.
Smith, Mark A.	Aug.
Sommer, Dieter	May
Spencer, Thomas V.	Sept.
Srok, Wolfgang	Apr.
Stafford, Paul S.	Dec.
Starr, Robert R.	June
Stephens, Bruce A.	Oct.
Swadener, Paul	Aug.
Szente, Pedro A.	Dec.
Sznaider, Kelly A.	Apr.
Tanksalvala, Darius F.	Sept.
Te, Heng V.	Mar.
Teegarden, Randy J.	Mar.
Undy, Stephen R.	Sept.
Van Loan, Paul R.	Aug.
Vasil, James A.	Jan.
Vogt, Michael	Apr.
Voss, Guenter	May
Wahl, Gaylord L., Jr.	Jan.
Waters, Geoffrey	July
Watkins, Marvin L.	June
Wechsler, Susan L.	Aug.
Weiss, Don	Sept.
Werner, Karl-Heinz	May
Wickes, William C.	Aug.
Wicklund, Eric J.	Jan.
Wilson, Edward L.	Apr.
Wolting, Duane E.	Apr.
Woods, Stanley P.	Dec.
Woodward, Mark J.	Dec.
Worsley, Robert S.	Oct.
Wylegala, Thomas B.	Mar.
Yeh, Allan S.	Mar.
Yetter, Jeffry D.	Sept.
Yie, Stephen	May

Hewlett-Packard Company, 3200 Hillview Avenue, Palo Alto, California 94304

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

HEWLETT·PACKARD JOURNAL

December 1987 Volume 38 • Number 11

Technical Information from the Laboratories of Hewlett-Packard Company
Hewlett-Packard Company, 3200 Hillview Avenue
Palo Alto, California 94304 U.S.A.
Hewlett-Packard Central Mailing Department
P.O. Box 529
1180 AM Amstelveen, The Netherlands
Yokogawa-Hewlett-Packard Ltd., Suginami-Ku Tokyo 168 Japan
Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive, Mississauga, Ontario L4V 1M8 Canada

```
00199127              HPJ 10/87
GEORGE PONTIS
SUITE 409
1742 SAND HILL RD
PALO ALTO, CA             94304
```

CHANGE OF ADDRESS: To subscribe, change your address, or delete your name from our mailing list, send your request to Hewlett-Packard Journal, 3200 Hillview Avenue, Palo Alto, CA 94304 U.S.A. Include your old address label, if any. Allow 60 days.

5953-8566