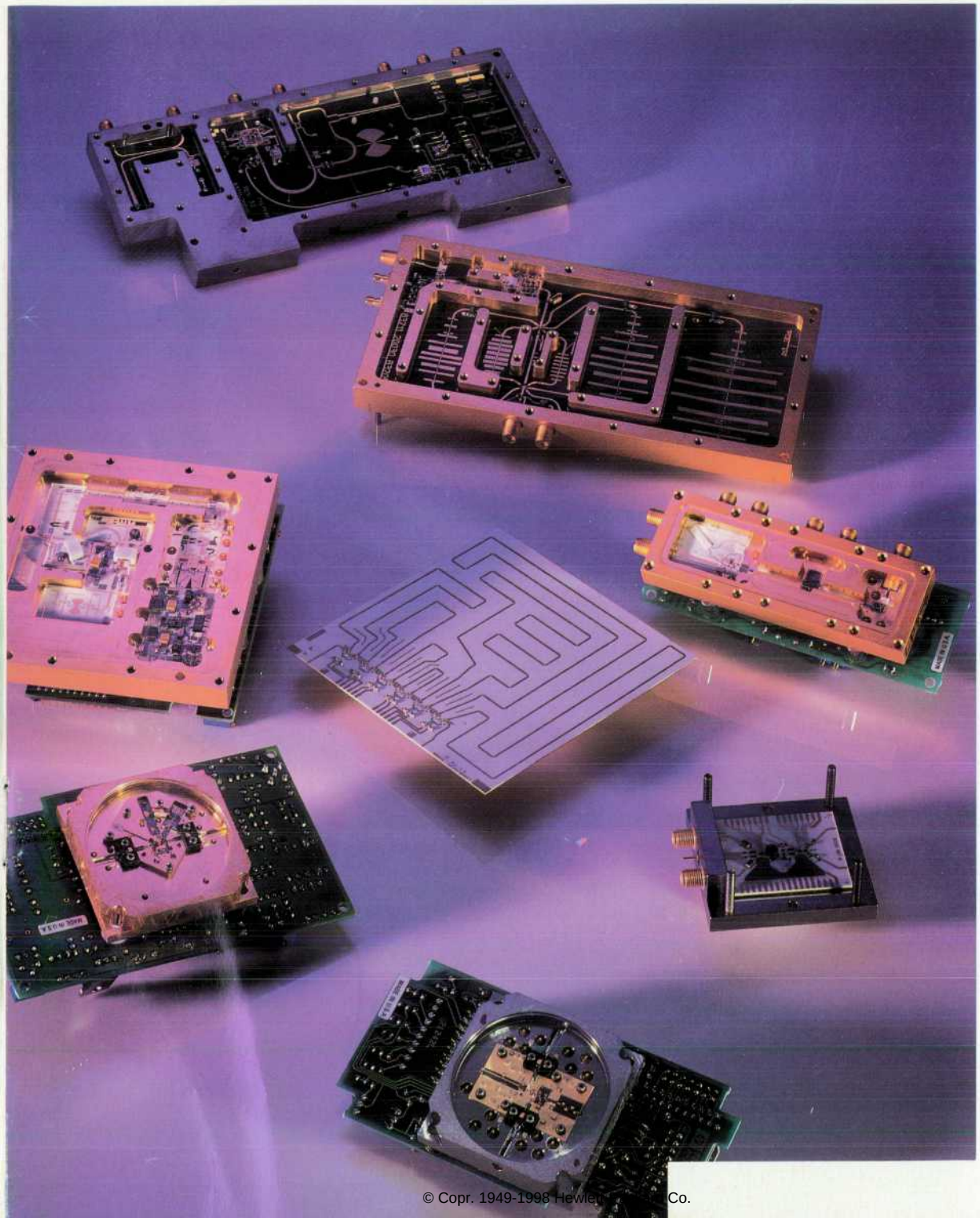
HEWLETT-PACKARD
JOURNAL

April 1993



HEWLETT-PACKARD
JOURNAL

April 1993 Volume 44 • Number 2

Articles

Departments

The Hewlett-Packard Journal is published bimonthly by the Hewlett-Packard Company to recognize technical contributions made by Hewlett-Packard (HP) personnel. While the information found in this publication is believed to be accurate, the Hewlett-Packard Company disclaims all warranties of merchantability and fitness for a particular purpose and all obligations and liabilities for damages, including but not limited to indirect, special, or consequential damages, attorney's and expert's fees, and court costs, arising out of or in connection with this publication.

Subscriptions: The Hewlett-Packard Journal is distributed free of charge to HP research, design and manufacturing engineering personnel, as well as to qualified non-HP individuals, libraries, and educational institutions. Please address subscription or change of address requests on printed letterhead (or include a business card) to the HP headquarters office in your country or to the HP address on the back cover. When submitting a change of address, please include your zip or postal code and a copy of your old label. Free subscriptions may not be available in all countries.

Submissions: Although articles in the Hewlett-Packard Journal are primarily authored by HP employees, articles from non-HP authors dealing with HP-related research or solutions to technical problems made possible by using HP equipment are also considered for publication. Please contact the Editor before submitting such articles. Also, the Hewlett-Packard Journal encourages technical discussions of the topics presented in recent articles and may publish letters expected to be of interest to readers. Letters should be brief, and are subject to editing by HP.

Please address inquiries, submissions, and requests to: Editor, Hewlett-Packard Journal, 3200 Hillview Avenue, Palo Alto, CA 94304 U.S.A.

In this Issue

State-of-the-art designs for three traditional types of microwave signal sources—the synthesized signal generator, the sweep oscillator or sweeper, and the pulse generator—make up the bulk of this issue. All three are considered basic test equipment, the synthesized signal generator for receiver testing, the sweeper for component testing and general-purpose applications, and the pulse generator for digital device testing.

A signal generator provides a stable, low-noise output signal at a precise frequency and power level and offers flexible modulation capabilities. The article on page 6 introduces the new HP 8370 Series and HP 70340 Series synthesized signal generators, which have state-of-the-art performance in the frequency range of 0.01 to 20 gigahertz. Key to their performance is a new architecture that uses two broadband fundamental oscillators instead of the oscillator and multipliers of previous designs, totally eliminating unwanted subharmonic frequencies. Details of the frequency synthesis section are in the article on page 12. The microwave chain, which divides the synthesized signal and provides for amplitude and pulse modulation, is described in the article on page 17. New processes developed for mixed-model, multiple-option production of these generators are detailed on page 30.

The HP 83750 Series sweepers (pages 38 and 46) produce swept-frequency output signals in the frequency range of 0.01 to 20 gigahertz. While previous designs locked the output frequency to a precise reference only at the beginning and end of each sweep in continuous-sweep mode, this new design ensures output frequency accuracy throughout fast analog sweeps. Like the signal generators with which they share this issue, these new sweepers use two broadband oscillators to generate their basic signal, eliminating multipliers and subharmonics.

The HP 8133A pulse generator (pages 52, 56, and 60) offers pulse repetition rates to 3 gigahertz and extremely precise placement of the pulse edges to help resolve subtle timing and signal integrity problems in fast digital devices. Available configurations include a single pulse channel, two pulse channels, or a pulse channel and a programmable channel that generates data patterns or pseudorandom binary sequences. The design makes extensive use of hybrid microcircuits and custom integrated circuits.

Fundamental to the measurement of acoustic noise is the 1/3-octave real-time frequency analyzer. FFT (fast Fourier transform) analyzers are also useful for certain measurements. Taking advantage of recent increases in the speed of off-the-shelf digital signal processing chips, the HP 3569A real-time frequency analyzer provides in a single handheld package a dual-channel FFT analyzer, a real-time 1/3-octave sound intensity filter analyzer, an integrating sound level meter, and a reverberation time processor. The all-digital processing results in very high precision. The article on page 73 describes the HP 3569A, focusing on the design of its multirate bank of digital bandpass filters.

The Simple Network Management Protocol, a computer network management standard promulgated by the Internet Activities Board, has been widely accepted in today's open systems environment. Network monitors complying with this standard must gather network statistics according to one of the many Management Information Bases, some vendor-specific, that have been registered on the Internet. The HP LanProbe II network monitor implements the Remote Monitoring Management Information Base, which was developed by HP, Novell, ProTools, and other vendors and is enjoying growing support. The article on page 82 describes this Management Information Base and its implementation in the HP LanProbe II.

Unseen microprocessor-based systems are fundamental to the operation of many products not thought of as computers, such as microwave ovens and modern automobiles. Developers of such "embedded" systems need an extensive set of hardware and software tools to help them debug, analyze, and integrate these systems. The HP 64700 embedded debug environment (page 90) is designed to provide access to these tools with a single, easy-to-use, graphical user interface that conforms to the OSF/Motif standard. The debug environment offers a high-level language debugger, a real-time emulation control and state analysis interface, a real-time software performance analyzer interface (see page 107 for a discussion of the new software performance analyzer), and other tools, including a dynamic real-time operating system analysis tool. The debug environment can be used alone or integrated into an even more extensive set of software development tools under the HP SoftBench environment.

R.P. Dolan
Editor

Cover

The advanced microwave synthesized signal generators, sweep oscillators, and pulse generator featured in this issue depend on state-of-the art hybrid microcircuit technology for performance, reliability, and economy. On the cover, some of the microcircuits developed for these products pose for a group photograph.

What's Ahead

The June issue will feature the HP ORCA system, a robot arm optimized for automating analytical laboratory tasks and other applications with similar needs. Also featured will be the HP Ultra VGA video adapter, the MPE/iX operating system, and HP's object-oriented database HP OpenODB. Finally, there will be six papers from the 1992 HP Software Engineering Productivity Conference—three are on practices for improving software quality, one on an error handling technique, one on configuration management for software tests, and one on a user interface development tool.

A New Family of Microwave Signal Generators for the 1990s

This family of generators includes both stand-alone and modular versions. A new architecture and state-of-the-art technologies result in advanced performance.

by William W. Heinz, Ronald E. Pratt, and Peter H. Fisher

Microwave signal generators have been major products of the Hewlett-Packard Company over a time frame spanning five decades. In the 1950s, klystron signal generators such as the HP 618 and the HP 620 were providing calibrated signals covering major bands with convenient single-knob tuning and basic modulation capability for testing communications and radar receivers. The advent of backward-wave oscillator tubes in the 1960s led to microwave swept frequency sources that could be tuned electronically over octave bandwidths and provided efficient component test capability.

During the 1970s, solid-state YIG-tuned (yttrium iron garnet) oscillators and multipliers replaced the tubes, and with the incorporation of frequency synthesis techniques, new dimensions in frequency stability, accuracy, and spectral purity were realized over very broad bandwidths. Along with improved modulation, sweep capability, and programmability, these were the contributions of the HP 8672, 8673, and 8340 families of microwave sources, which set the standards for the 1980s. Major enhancements in performance and convenience were added later in the decade with the introduction of the HP 8360 family of synthesized sources.

Over the years, the needs of customers who use microwave sources have become more specialized. Manufacturers of wideband receivers must have generators with more modulation capability and wider frequency coverage, together with excellent spectral purity and a low broadband noise floor. People testing radars need low phase noise, accurate power output over a wide dynamic range, and flexible, accurate pulse modulation capability. Customers working in the communications industry need high-index frequency modulation and low phase noise, residual FM, and spurious frequency emissions. Other customers desire lighter weight and more compact packaging.

New Signal Generators

The new HP 8370 and 70340 Series signal generators offer cost-effective solutions for these requirements as well as for other needs that can be anticipated for the 1990s. The HP 8370 Series includes not only signal generators for receiver test applications but also sweepers for component test and general-purpose applications. This article and the articles on pages 12, 17, and 30 discuss the design and manufacturing of the HP 8370 Series and HP 70340 Series signal generators. The sweeper members of the HP 8370 family are known as the HP 83750 Series, and are described in the articles on pages 38 and 46.

Six HP 8370 and 70340 signal generators have been introduced so far (see Table I). Four are in standard-rack-width, 5.25-inch-high packages and two are in Modular Measurement System (MMS) format.

Table I
Microwave Signal Generator Characteristics

Model Number	Frequency Range	Output Power	Modulation Capability	Physical
83731A	1 to 20 GHz	–90 to +10 dBm	Log AM, FM, Pulse	5.25 in high, 35 lb
83732A	0.01 to 20 GHz	–90 to +10 dBm	Log AM, FM, Pulse	5.25 in high, 35 lb
83711A	1 to 20 GHz	–90 to +10 dBm	CW Only	5.25 in high, 35 lb
83712A	0.01 to 20 GHz	–90 to +10 dBm	CW Only	5.25 in high, 35 lb
70340A	1 to 20 GHz	–90 to +10 dBm	Log AM, FM, Pulse	4/8-width MMS, 19 lb
70341A	0.01 to 1 GHz extension	–90 to +13 dBm	Log AM, FM, Pulse	1/8-width MMS, 5 lb

The HP 8370 Series represents the stand-alone members of the family. This series includes two signal generators provided with modulation capability optimized for receiver test applications. The HP 83731A (Fig. 1) covers the frequency range from 1 to 20 GHz, and the HP 83732A adds RF and IF test capability, covering the full 10-MHz-to-20-GHz range in a single, compact (17-inch-deep) package weighing less than 35 pounds, approximately half the weight of the previous generation of sources. Modulation capability includes high-performance pulse modulation (rise and fall times <10 ns and 80-dB on/off ratio). AM performance has been enhanced to provide logarithmic (–10 dB/volt) response down to –60 dB, suitable for antenna scan simulation and deep power sweeps for amplitude compression testing of receivers and subsystems. With the new wideband FM capability, modulation indexes exceeding 300 are possible.

The HP 83731A and 83732A feature a built-in pulse source that provides pulse rates from 3 Hz to 3 MHz and pulse widths from 25 ns to 419 ms. The pulse source can be selected from the front panel or an external controller. In triggered mode, delays of ±419 ms are available for pulse return

Fig. 1. The HP 83731A synthesized signal generator is representative of the stand-alone HP 8370 Series signal generators. This 1-to-20-GHz instrument has modulation capability optimized for receiver test applications.

simulation or other applications. In addition, there are pulse doublet and externally gated burst modes of operation, useful for receiver shadow-time measurements and simulating various real-world signals.

There are many measurement situations in which it is desirable to have the very accurate level calibration available at the front-panel RF connector of the signal generator transferred to a remote point in a system where lengths of cable or losses introduced after switch boxes have degraded the flatness of the signal. The user level correction feature allows this to be accomplished by connecting a power meter at the remote point with an HP-IB (IEEE 488) cable between it and the signal generator. The generator can control the power meter and store correction factors in memory (four tables can be stored with up to 401 points each). Calibrated, flat power levels then become available for accurate measurements at remote ports.

The other two signal generators of the HP 8370 Series are the HP 83711A and 83712A, which are CW sources designed for testing where an economical source without modulation capability is needed, such as in exciter and local oscillator applications. Sufficient power (+10 dBm) for mixer drive is provided with the same low harmonic levels (–55 dBc) as the versions with modulation.

Modular Versions

There is a market segment that has requirements for instrumentation that equals the best in the industry along with reduced size and increased flexibility. To meet this need the significant contributions of the HP 8370 signal generators have been leveraged into the Modular Measurement System (MMS) platform (see Fig. 2). Supporting an optional modular display and adding an MSIB* interface, the HP 70340A offers the same performance as its rack-and-stack counterpart, the HP 83731A, at half the size and two thirds the weight. A companion unit, the HP 70341A, is a one-slot MMS module that extends the HP 70340A's capability to cover the 0.01-to-20-GHz frequency range. The state-of-the-art accuracy of the HP 8370 Series signal generators is maintained in the modular versions by using the same overall architecture, which allows the modular products to be partitioned so that interchangeability of modules is achieved without any need for recalibration.

The HP 70341A frequency extension module contains 0.01-to-1-GHz dividers, amplifiers, modulators, and filters. Accurate power output is maintained by placing the low-frequency leveling detector inside the HP 70340A. This allows the frequency response characteristics of the 0.01-to-1-GHz path to be calibrated and stored in the nonvolatile memory of the HP 70340A. Correction factors for the nonlinearities of the pulse and scan modulators can be calibrated separately and stored in the nonvolatile memory provided in the HP 70341A.

* Measurement Systems Interface Bus, the internal bus of the open standard Modular Measurement System.

Fig. 2. The HP 70340A signal generator is the Modular Measurement System version of the HP 83731A. The HP 70341A frequency extension module extends the range of the HP 70340A down to 10 MHz.

Fig. 3. An example of a modular measurement system consisting of an HP 70340A signal generator (highlighted) and an HP 71500A microwave transition analyzer.

This allows complete interchangeability of modules while maintaining the state-of-the-art specifications.

Measurement Systems

In addition to reducing size and providing interchangeability of modules, the HP 70340A can work with other HP MMS products to create powerful, compact measurement systems. An example is illustrated in Fig. 3, which shows the HP 70340A and the HP 71500A microwave transition analyzer working together to obtain sophisticated data. Fig. 4 shows the HP 70340A being used with the HP 70841A pattern generator and the HP 70842A error performance analyzer to produce a compact, modular 3-gigabit bit error rate test system.

New Signal Generator Architecture

To provide the improvements required for this new family of signal generators, a new architecture was implemented and technologies were incorporated that had matured only after the previous generation of sources had been designed. A simplified block diagram is shown in Fig. 5. Two broadband YIG-tuned oscillators replace the YIG-tuned oscillator and multipliers employed in past designs and totally eliminate the generation of subharmonic frequencies. Broadband gallium arsenide (GaAs) monolithic microwave integrated circuits (MMICs) provide the gain, output power, and variable attenuation needed to develop the required signals. A bank of switched low-pass filters is used to attenuate unwanted harmonics without degrading the fidelity of the complex modulation placed on the carrier frequency.

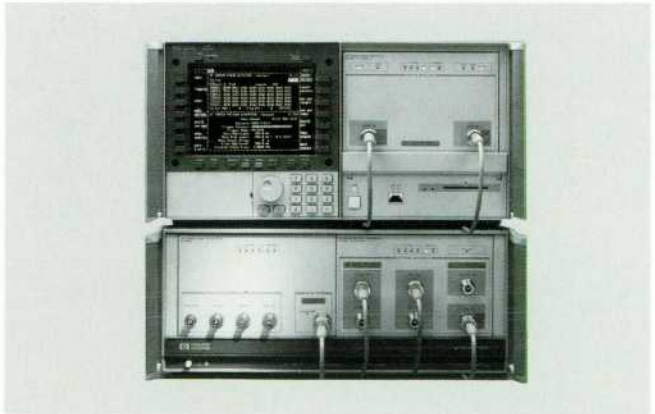


Fig. 4. A compact, modular 3-Gbit/s bit error rate test system consisting of the HP 70340A signal generator (highlighted), the HP 70841A pattern generator, and the HP 70842A error performance analyzer.

Frequency extension below the 2-GHz lower limit of the YTO (YIG-tuned oscillator) is accomplished with microwave and RF frequency dividers, which cost less and provide higher performance than the previously used heterodyne low-frequency coverage. Precision, low-cost RF modulators implemented with surface mount technology complement their microwave counterparts.

In addition to the use of GaAs MMICs, improvements in performance, cost, size, and weight have been attained through the high levels of integration achieved in the microwave chain by incorporating Duroid substrate technology to realize directional couplers, switches, filters, and interconnects. Previous reliance on multiple microcircuit packages connected together with coaxial cables has been supplanted by the use of just two integrated microcircuit subsystem packages.

The use of surface mount technology for the printed circuit boards provides a high degree of compactness and weight reduction compared to previous signal generators. In addition, the levels of performance required for the low-band (divider) frequency extension could not have been realized with the through-hole technology of the past.

Leverage and Reuse

A high level of leveraging and commonality went into the design of the new family of signal generator products described here. The benefits of this approach extend to manufacturing as well, since economies of scale in fabrication, assembly, and test provide efficiency and savings. The microwave chain subassemblies are identical in the HP 83731A and 70340A. A simple replacement of the modulators within these microcircuits with 50-ohm transmission lines is the only change required to produce one of the CW signal generators of the HP 8370 Series. Even the power supply in the HP 8370 Series signal generators is common to all versions and is a self-contained, line-driven, switching unit. Four of the six major surface mount printed circuit board subassemblies in the HP 70340A are identical to the HP 8370 Series signal generator boards. The low-band divider board in the HP 70341A is identical to the low-band board in the HP 8370 Series signal generators. The power supply in the MMS version is driven by the standard 40-kHz, 24-volt power bus of the MMS mainframe.

The Modular Measurement System standard has tight specifications for electromagnetic compatibility. Fig. 6 shows the initial results of emissions testing performed on the new family of sources. The modular source met specifications without any of the design modifications that were required to make the stand-alone products pass the test.

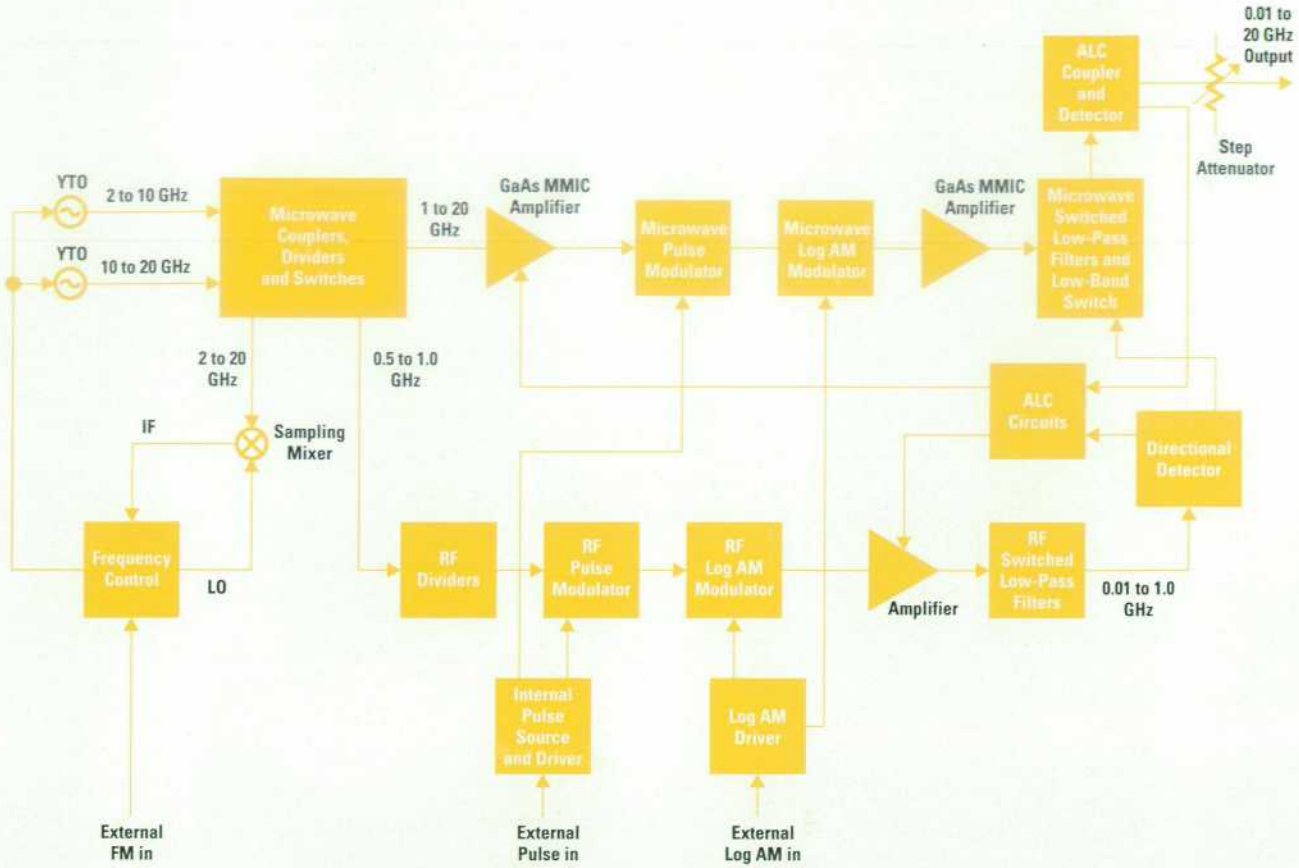


Fig. 5. Simplified block diagram of the new signal generator family. Two broadband oscillators replace the oscillator and multipliers of previous architectures.

The modular source can be placed in the same mainframe as the signal analyzer that was used to obtain the data presented in Fig. 6. This condition must not degrade the system performance. A test was devised that placed the HP 70340A in close proximity to a modular scalar network analyzer system that has a 140-dB dynamic range. Initial tests showed some degradation in network analyzer performance. This was eliminated by connecting a 50-ohm load on the 0.5-to-1-GHz output of the signal generator and repairing a cracked solder joint on a coaxial cable. Retesting showed that the presence of the HP 70340A could not be detected by the wide-dynamic-range tracking generator system. Passing this test successfully is evidence of the EMI performance of the MMS platform.

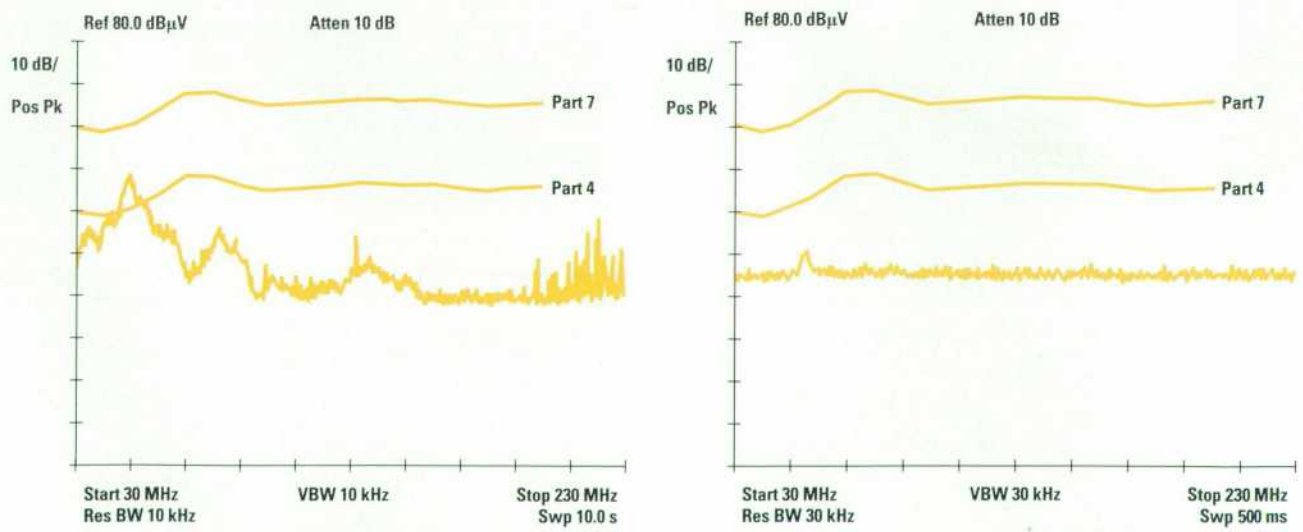


Fig. 6. Initial radiated emissions test results for stand-alone (left) and modular (right) prototype signal generators. The modular unit met the specification while the stand-alone unit required further engineering. The limits shown are from MIL-STD 461B, RE 02.

Because of space constraints, the modulator drive circuits had to be squeezed onto the power supply board. These circuits already existed on boards that are specific to the HP 8370 signal generators. Since they were designed to fit into a standard cell, it was possible to leverage them for reuse in the HP 70340A. The microprocessor board is also specific to the HP 70340A, but is essentially an HP 8370 CPU board with the space used for the front-panel interface circuits on the stand-alone board occupied by the MSIB interface.

Firmware

Another example of reuse is in the area of the firmware interfaces that exist between the hardware and the end user. Many instruments contain more computer power than was available in a small mainframe a decade ago. The firmware that controls this power has to be designed just as carefully as the microwave circuits. Reuse of firmware has played a key role in the development of this new family of signal generators.

A greatly simplified model of the signal generator firmware is shown in Fig. 7, which indicates the major processes that must be controlled by the resident operating system when the instrument is in its normal mode of operation. All but one of these processes were reused from past efforts, so we were able to concentrate on developing the instrument execution routines that are a major contributor to the high performance of the product family. The display handler, front-panel interface, and error handler processes were leveraged from the development efforts of several other products. A significant contribution was a hardware independent SCPI (Standard Commands for Programmable Instruments) parser supplied by HP's Instrument Controller Laboratory.

The front-panel processes share a common database that contains the requirements that are specific to each product. The token executor is a small process that receives and outputs standardized messages in the form of tokens. The operating system ensures that the tokens are processed in a sequential manner. A unique database for each instrument defines the actions of the token executor.

Low Cost of Ownership

Each signal generator application has specific needs. The basic architecture of this new family of sources addresses a variety of these needs with an implementation yielding cost-effective solutions. Total cost of ownership is minimized by powerful internal verification capabilities, which can quickly isolate a fault to a given subassembly and reduce the mean time to repair to about four hours. The calculated mean time

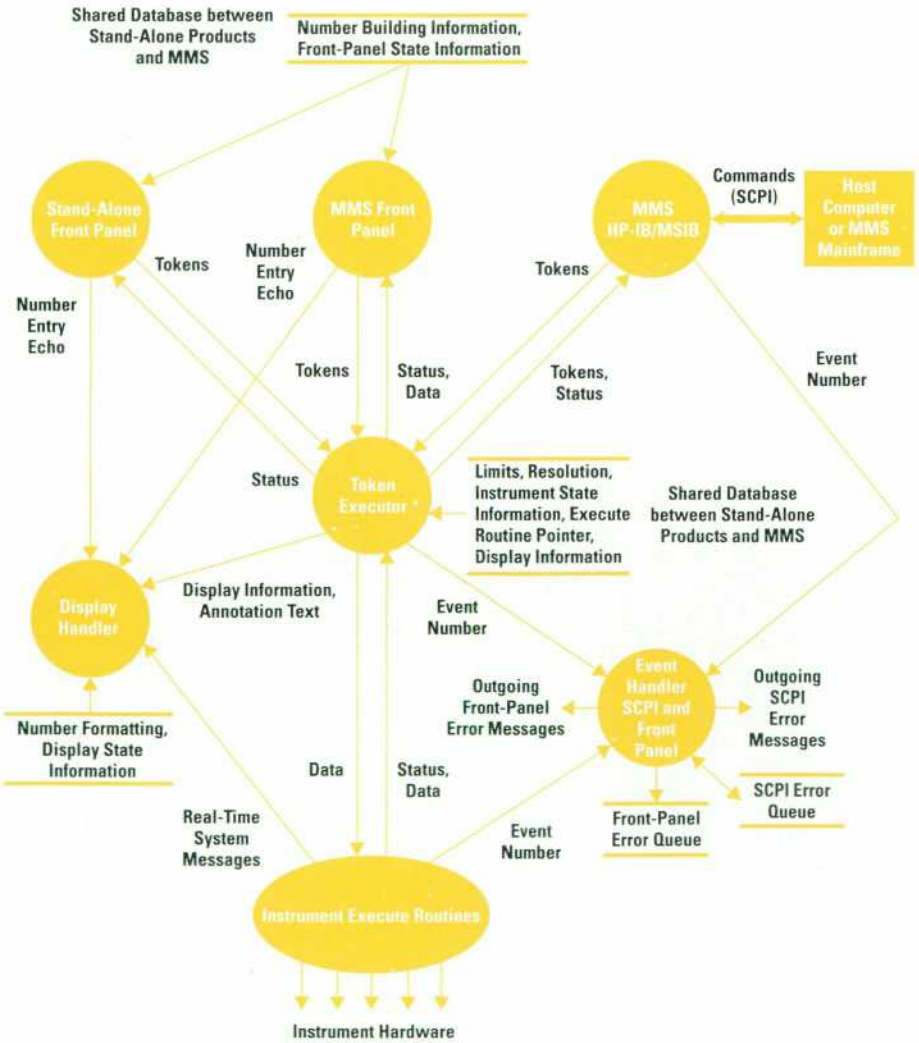


Fig. 7. Overview of the HP 8370 Series and 70340 Series signal generator firmware. Six major processes are controlled by the resident operating system.

before failure is greater than 20,000 hours (about 10 years of normal use) and the recommended recalibration cycle has been extended to two years.

Acknowledgments

Many of the people who contributed to the products described here are authors of other articles in this issue. Behind all of us is a large group of people who contributed to making the project a success. Tim Carey and Lynn Chroust provided key marketing support for the program and Al Barber, as the project manager at the beginning of the project, made contributions to the definition and to the design. Ron Larson, Gary Rosen, and Ed Cirimele did a great job making the modular products a reality. Bob Skinner played a key role helping us apply surface mount technology to new, high-performance designs. Bill Wendin, Mark Johnston, and Paul Zander were instrumental in providing the firmware. Special thanks go to Bob DeVries and Phil Foster, who developed the product design concepts that made the entire product family a reality.

Broadband Fundamental Frequency Synthesis from 2 to 20 GHz

A broadband fundamental YIG-tuned oscillator is locked to a stable reference and controlled by four phase-locked loops to produce the low-phase-noise output signal of the HP 8370 and 70340 signal generators.

by Brian R. Short, Thomas L. Grisell, and Edward G. Cristal

The frequency synthesis subsystem of the HP 8370 Series and HP 70340 Series synthesized signal generators produces a stable, accurate microwave carrier signal in the frequency range from 2 to 20 GHz and delivers it to the microwave subsystem, which is described in the article on page 17. Also, it is in the frequency synthesis subsection that frequency modulation is applied to the carrier.

Fig. 1 is a simplified diagram of the microwave oscillator phase-locked loop used in the HP 8370 and HP 70340 signal generator families. Both VCOs in the diagram (the reference VCO and the LO) are locked to a common stable frequency reference (not shown in the figure). The phase/frequency detector outputs a voltage proportional to the phase difference between the reference VCO and IF frequencies, which after filtering and integration tunes the microwave YIG-tuned oscillator (YTO) in a direction to drive the phase difference error to zero.

Frequency modulation, when used, is summed into the forward path of the loop and simultaneously modulates the microwave oscillator together with the control signal. In dcFM mode the phase-locked loop is opened (the integrator nulled) and only the FM signal modulates the microwave oscillator.

Detailed Description

A more detailed drawing of the synthesis subsystem is shown in Fig. 2. The microwave oscillator is a pair of broadband fundamental YIG-tuned oscillators covering 2 to 10 and 10 to 20 GHz. Each YTO relies on a high-Q yttrium iron garnet device whose resonant frequency is proportional to its internal magnetic field. The YTO tuning circuitry consists of two windings that control the magnetic field: a main coil, which carries the dc control current and sets the CW frequency, and a much smaller winding that has a broad frequency response, which is used for ac control and FM. The YTO system is locked to an internal 10-MHz frequency standard or to the user's external source by the phase-locked loop. Frequencies between 1 and 2 GHz are derived in the microwave subsystem by dividing the 2-to-4-GHz band. Frequencies between 10 MHz and 1 GHz are obtained by additional dividers located on the low-band subsystem printed circuit board. ALC (automatic level control), AM, and pulse modulation are added in the microwave subsystem.

The frequency synthesis subsystem consists of four phase-locked loops: the reference oscillator loop, the local oscillator loop (LO loop), the offset oscillator loop, and the microwave YIG oscillator loop (YO loop). The entire subsystem, excluding the sampler and the YIG oscillators, resides on three surface mount printed circuit boards. The main coil driver and its associated filters, amplifiers, and digital communication circuits reside on a 4-by-11-inch board. The YO loop, offset oscillator loop, FM driver, and associated digital communication circuitry reside on one 4-by-15-inch board, and the LO and reference oscillators and their associated

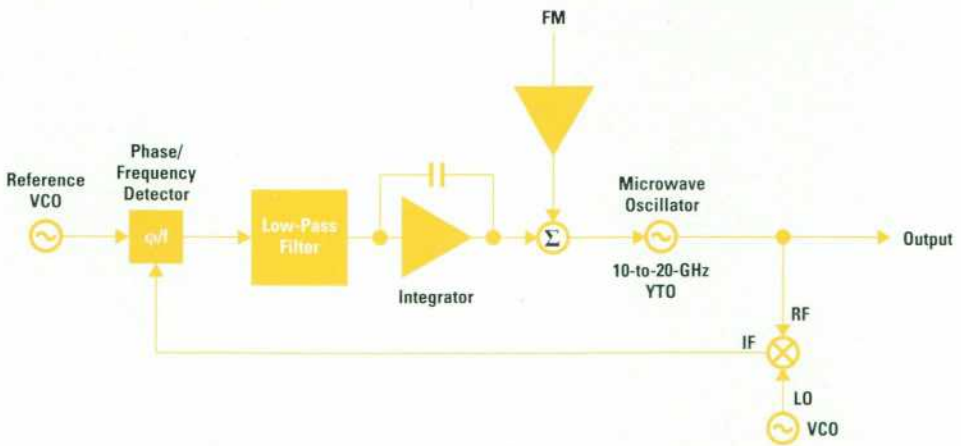


Fig. 1. Basic IF-type phase-locked loop used in the HP 8370 and 70340 Series signal generators.

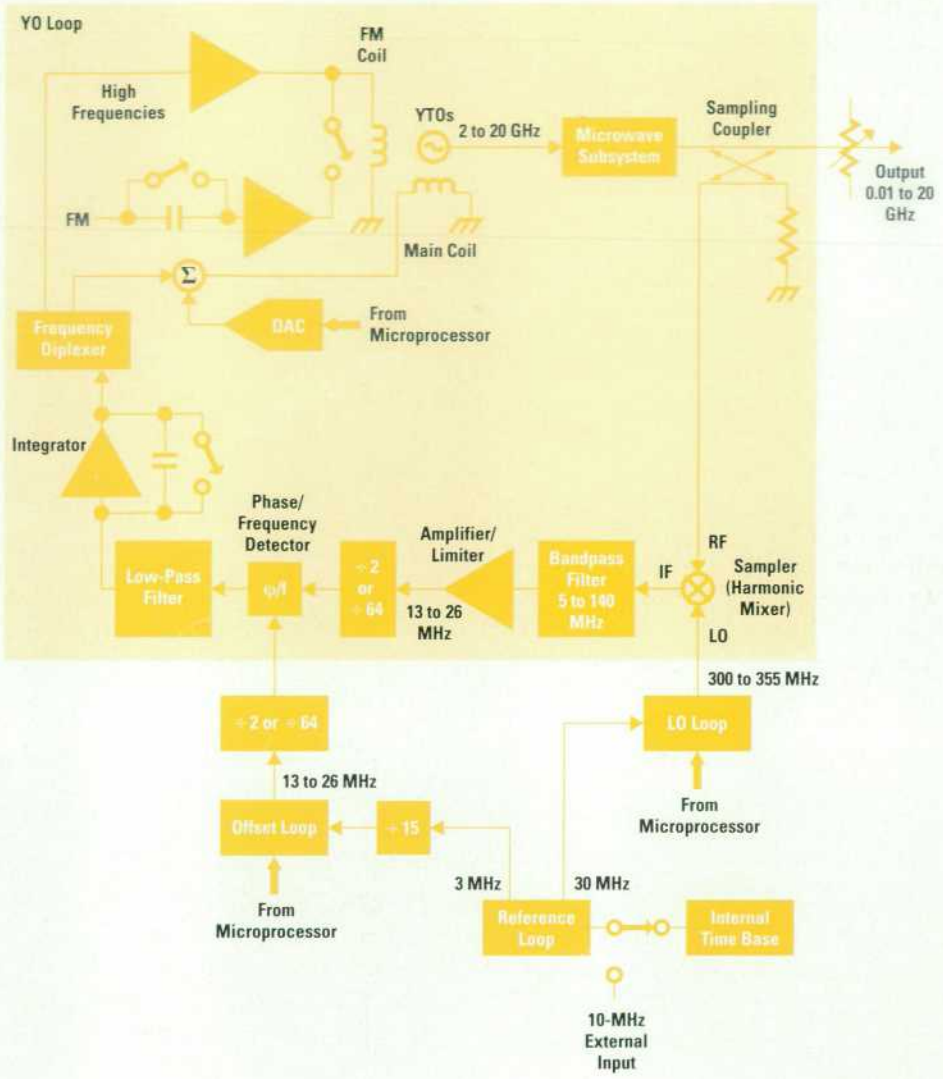


Fig. 2. Block diagram of the frequency synthesis section of the HP 8370 Series and 70340 Series synthesized signal generators.

digital communication circuits reside on a second 4-by-15-inch board.

A quick trip around the YO loop in Fig. 2 shows its basic operation. A small amount of the YTO signal near the output of the instrument is sampled by a broadband transmission-line directional coupler and routed to the sampler, which acts as a harmonic mixer. The sampler, driven by the LO loop, produces a comb of harmonics which down-converts the microwave signal to the intermediate frequency (IF). The IF and extraneous mixing products are heavily filtered by the 5-to-140-MHz bandpass filter, then amplified, hard limited, and converted to digital pulses.

The digital signal is divided by 2, increasing the effective dynamic range of the phase/frequency detector, and becomes one of the two control signals used for phase comparison. The other control signal to the phase/frequency detector is generated in the offset oscillator loop. This loop provides the fine frequency resolution for the synthesizer.

The phase/frequency detector provides a net positive or negative voltage when the IF and offset oscillator frequencies are different. When the IF and offset oscillator signals have the same frequency, the detector produces a voltage proportional to their phase difference. In both situations, the integrated error voltage tunes the YIG oscillator frequency in a direction to drive the error to zero. When the error voltage is zero, the YIG oscillator is phase-locked and remains locked because of the action of the integrator.

The error voltage from the integrator is frequency-diplexed so that higher-frequency components are routed to the low-inductance FM coil, while dc and lower-frequency components are transferred to the main coil. The frequency diplexer cutoff frequency is about 5 Hz.

Frequency Modulation

Frequencies within the loop bandwidth of the YO phase-locked loop are tracked out by the action of the loop. Consequently, FM rates must exceed the YO loop bandwidth to modulate the YTO. To achieve the required FM bandwidth (1 kHz to 1 MHz in the HP 8370 Series) the YO loop bandwidth is reduced to about 600 Hz from its nominal 30 kHz. The high end is limited by the characteristics of the FM coil. Some gain and frequency equalization are added in the FM drive circuitry to flatten the overall response and provide a nominal 5-MHz/volt sensitivity and 600-ohm impedance at the front panel.

Also in FM mode, the dividers in front of the phase/frequency detector are switched to divide by 64. This further increases the detector's effective dynamic range, thereby allowing a

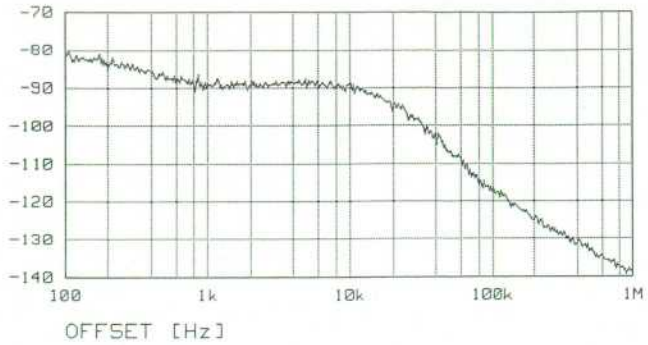


Fig. 3. Model 83731A phase noise performance at 9.999 GHz and +10 dBm.

large FM modulation index. The modulation index β is defined as the peak frequency deviation, Δf, divided by the modulation frequency, f_m. The phase/frequency detector is limited to ±2π radians, so we have the result:

$$\frac{\Delta f/64}{f_m} < 2\pi.$$

Rearranging factors gives:

$$\beta < 2\pi\,(64) = 402.$$

Because of practical limitations in the phase/frequency detector β is limited to around 300.

Phase Noise Performance

As previously mentioned, the YO loop is able to track frequency variations within its bandwidth. This applies to noise as well. The largest noise contributors are the YTO, the LO, and the offset oscillators. The LO noise is increased by the harmonic factor of the sampler. The total phase noise at the output of the YO loop is the YTO phase noise reduced by the open-loop gain, plus the other loops' phase noise contributions. The noise contributions of the LO and offset oscillators are more significant within the YO loop bandwidth, and the YTO phase noise is the most significant outside of the YO loop bandwidth. The YO loop bandwidth is selected to be at the frequency at which the frequency-multiplied noise of the LO synthesizer and the free-run noise of the YTO are about equal. The noise contribution of the offset synthesizer is generally insignificant. A typical phase noise plot showing single-sideband phase noise as a function of offset frequency at 9.999 GHz and +10 dBm is shown in Fig. 3.

Offset Loop

The offset loop is the phase-locked loop that sets the frequency resolution of the HP 8370 and 70340 Series signal generators. The offset loop output is phase-compared with the sampler IF output (see Fig. 2). Therefore, the output frequency of the instrument changes by the same amount as any frequency change in the offset loop. The offset loop has an output frequency tuning range of 13 MHz to 26 MHz. Depending on the instrument option ordered, the frequency resolution can be either 1 Hz or 1 kHz.

The offset loop output signal drives the divide-by-2 or divide-by-64 circuit ahead of the YO loop phase detector. Because the offset loop is phase-compared with the output of the sampler, the spurious and phase noise performance of the offset loop translates directly to the output of the signal generator. Stated another way, the equation relating the output frequency of the offset loop to the output frequency of the entire instrument has no factors of frequency multiplication or frequency division. Therefore, given a desired set of spurious and phase noise specifications for the signal generator output, the corresponding specifications are required of the offset loop.

To obtain the frequency resolution required, the offset loop uses a fractional division scheme. A proprietary IC chip applies a digital correction technique using sigma-delta modulator technology and interpolative analog-to-digital conversion to shape the frequency spectrum of the fractional error energy.[1] The shaping of the spectrum pushes the fractional error energy away from the carrier, where it can be filtered by manipulation of the loop response. The result is a fractional-N phase-locked loop that is not encumbered by the sophisticated error correction circuitry of past fractional-N phase-locked loops. The performance is high and the cost is low.

A basic block diagram of the offset loop is shown in Fig. 4. The VCO tunes from 520 MHz to 1040 MHz. This is a desirable frequency tuning range because the other reference loop, the LO loop, has a VCO that tunes from 300 MHz to 355 MHz. In any synthesizer, it is desirable not to have the internal phase-locked loops tuning over common frequency ranges. This is to minimize coupling between synthesizers, which could produce spurious responses. The output of the VCO is well-isolated from the dividers to prevent any spurious response from the VCO itself. The dividers include the output divider (divide by 40), and the prescaler ahead of the fractional synthesizer IC. The output of the fractional synthesizer IC is phase-compared with a 200-kHz reference signal. This reference is derived by dividing a 3-MHz external signal (traceable back to the 10-MHz crystal) by fifteen. The outputs of the phase detector are summed by a differential amplifier. At the output of the differential amplifier, a variable-gain stage keeps the loop gain relatively constant. The VCO tuning constant varies over the tuning range, and the frequency division number from the VCO to the phase detector varies by a factor of 2 to 1. Both of these factors contribute

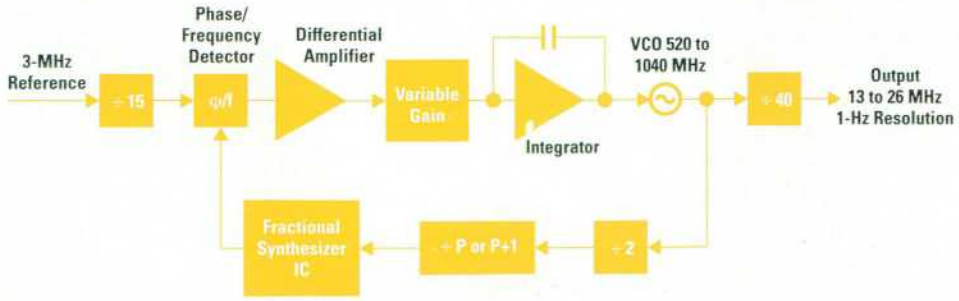


Fig. 4. Offset phase-locked loop.

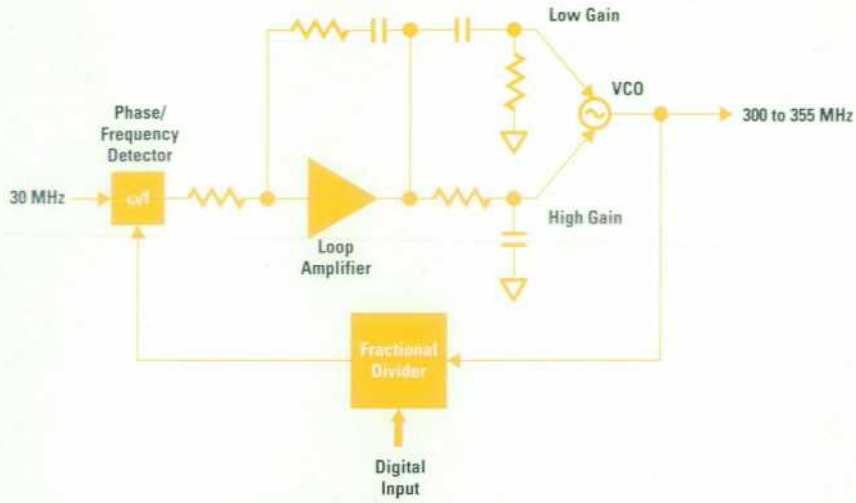


Fig. 5. LO loop.

to reducing the loop gain at the high end of the output frequency range. The variable-gain stage compensates for these variations. The result is that the phase noise response of the loop remains relatively constant over the output frequency range. The variable-gain stage is followed in the loop by the loop integrator.

As previously mentioned, the spurious performance of the offset loop must be as good as the desired spurious performance of the signal generator output. The actual offset loop spurious performance is over 25 dB better than required. The phase noise performance required of the offset loop is also set by the desired signal generator phase noise specifications. The offset loop exceeds the desired phase noise performance at all offsets from the carrier by at least 30 dB. The divider at the output is the key to having so much margin in the spurious and phase noise performance. With frequency division at the output, the spurious responses and phase noise are improved by 20 times the logarithm of the division factor with respect to the carrier signal. In the offset loop the frequency division factor is 40, and the resulting improvement in noise and spurious performance is 32 dB.

The only other performance specification that affected the offset loop design was the switching speed. It was determined that to meet the instrument switching speed specifications, the offset loop must switch in less than 10 ms. To obtain this specification, the loop bandwidth was adjusted and a discharge path for the loop integrator capacitor was provided.

Microwave Sampler

The HP 8370 Series uses a fundamental YIG oscillator to generate the output frequency directly, instead of multiplying the output from a lower-frequency oscillator. This requires that the microwave sampler be able to operate at the output frequency (as high as 20 GHz) to provide down-conversion for frequency control. The sampler used is borrowed from the HP 5350 Series microwave counters.[2] It is received assembled and tested, and used as-is.

There are distortion mechanisms in the sampler that can cause spurious signal products at its output. Some of these, in turn, can cause undesired mixing products in the synthesis section and appear at the instrument output as spurious signals. An algorithm was used to search for the best sampler LO frequency for each microwave frequency to minimize spurious signals. These frequencies are stored in ROM in the HP 8370 Series and are used to set the LO and offset synthesizers appropriately for each output frequency band.

LO Loop

As mentioned above, the LO loop output is one of the inputs to the microwave sampler. The LO loop, shown in Fig. 5, is a divide-by-N phase-locked loop with an output range of 300 to 355 MHz in 0.5-MHz steps. Phase noise performance of this loop is critical, since its output frequency is multiplied in the sampler to within an IF spacing of the YIG oscillator frequency. The multiplication can be as high as 66 times the LO frequency at 20 GHz, which increases the noise contribution by approximately 36 dB. Noise performance depends on careful VCO design, fractional division, and a low-noise reference.

The VCO is a cost-effective but high-performance printed circuit board varactor-tuned oscillator that uses a length of solid coaxial cable as the inductive element. This reduces the vibration sensitivity of the oscillator circuit because of the inherent self-shielding of the coax.

An important feature of this VCO is the presence of two tuning inputs: a low-sensitivity input and a high-sensitivity input. The phase-locked loop uses the high-sensitivity input to tune the VCO over the required frequency range. The low-sensitivity input has almost constant sensitivity in MHz/volt. This input is the one on which the loop bandwidth depends, so the bandwidth also remains constant. Constant bandwidth helps maintain optimum noise performance. The VCO phase noise is typically –122 dB/Hz at a 10-kHz offset.

Fractional division is used in the divide-by-N circuitry to aid low-noise performance. A straightforward way to achieve 0.5-MHz steps in frequency would be to use a 0.5-MHz reference and a divide-by-N divider. This, however, would cause a large multiplication of the reference noise. Instead, a 30-MHz reference is used, and the 0.5-MHz steps are developed by using a fractional divide-by-N circuit. Thus, noise is multiplied up by a much smaller factor: 0.5/30 or approximately 36 dB less than using a 0.5-MHz reference.

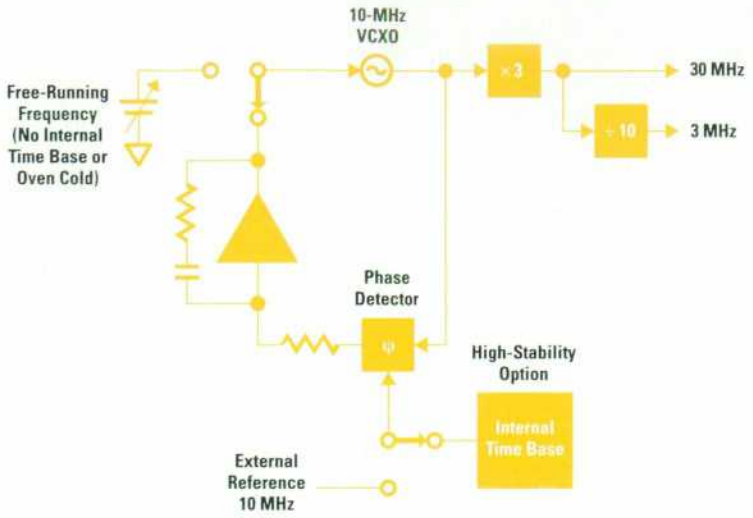


Fig. 6. Reference loop.

The fractional divider is realized using a combination of ECL digital circuits and TTL PAL (programmable array logic). It operates at 60 MHz.

The low-noise reference is provided by the reference loop, which is discussed below.

Much design work went into the whole instrument, and the LO loop in particular, to achieve excellent EMC (electromagnetic compatibility) performance. Again, one of the most difficult aspects was the large multiplication factor from the LO loop to the instrument output. Special internal shielding was developed for the VCO section on the printed circuit board to eliminate RF susceptibility of the VCO at its oscillation frequency, an important aspect to many users of microwave signal generators whose applications are in noisy RF environments that are rich in potentially interfering signals.

There are no adjustments in the LO loop.

Reference Loop

The reference loop, shown in Fig. 6, provides two references: 3 MHz for the offset loop and 30 MHz for the LO loop. It also allows the use of an external 10-MHz reference. The internal reference is a 10-MHz printed circuit board crystal oscillator, which is locked to either the external reference or the HP high-stability time base option if one is present. External references are automatically selected when present at the rear-panel connector, and the instrument microprocessor is notified for front-panel indication.

The printed circuit board crystal oscillator can lock over a range greater than ±500 Hz and maintains a noise level of −155 dBc/Hz at 1-kHz offset at the 30-MHz output to the LO loop. There is one adjustment to set the unlocked frequency of the oscillator to account for crystal variations.

The 30-MHz output is developed by frequency-tripling the 10-MHz signal, and the 3-MHz output is derived from the 30-MHz output by a divide-by-ten digital divider.

An on-board detector senses an external reference and on-board logic disables the internal high-stability reference while switching the reference phase-locked loop input to the external reference port at the rear-panel connector.

Digital and Analog I/O

Digital control and readback are done serially, using 16 bits. As a self-check, the serial bus is capable of reading back the data sent. It can also read status information from the LO loop. Unlock indications are handled by an interrupt system to avoid constant polling overhead for the instrument microprocessor.

Analog voltages and frequencies can be measured via a single-line analog bus that each board can drive using local analog multiplexers. Thus, power supplies, tune voltages, reference frequencies, and so on can be measured for troubleshooting and functional verification.

Acknowledgments

We would like to acknowledge the contributions of Sunia Yang, Bill Cornelius, David Eagleton, and Terry Noe to the development of the synthesizer subsystem. Sunia designed the YO loop, Bill designed the YIG low-frequency driver, David characterized the YTO performance, and Terry designed the FM circuitry.

References

1. B. Miller and R.J. Conley, "A Multiple Modulator Fractional Divider," *IEEE Transactions on Instrumentation and Measurement*, Vol. IM-40, no. 3, June 1991, pp. 578-583.
2. S.R. Gibson, "Gallium Arsenide Lowers Cost and Improves Performance of Microwave Counters," *Hewlett-Packard Journal*, Vol. 37, no. 2, February 1986, pp. 4-10.

A New High-Performance 0.01-to-20-GHz Synthesized Signal Generator Microwave Chain

Driven by a broadband YIG oscillator, the microwave chain only divides the oscillator output instead of multiplying and heterodyning like previous designs. The benefits include no subharmonics and higher-performance pulse and amplitude modulation. The major functions of the microwave chain are integrated on two microcircuits.

by William D. Baumgartner, John S. Brenneman, John L. Imperato, Douglas A. Larson, Ricardo de Mello Peregrino, and Gregory A. Taylor

The microwave chain of the HP 8370 and HP 70340 synthesized signal generator families receives the output of the frequency synthesis section (2 to 20 GHz) and creates the leveled signal generator output signal (0.01 to 20 GHz).

The evolution of the microwave chain was driven by the needs of receiver test and local oscillator customers for higher performance, and the desire to reduce size, weight, and cost. The key performance goals were +8-dBm output power, ±1-dB level accuracy, no subharmonics, low harmonics (–55 dBc), fast (10 ns) high-fidelity pulses, and the capability for simultaneous pulse and deep (–60 dBc) amplitude modulation to simulate rotating-antenna transmitters (Fig. 1).

Previous architectures start with a 2-to-6.6-GHz fundamental band, multiply to 26.5 GHz, and postfilter with a YIG (yttrium iron garnet) filter. The frequency control, ALC, AM, FM, and pulse modulation are situated in the fundamental band for lower cost. The disadvantages associated with this approach are subharmonics, low power, slow pulse rise time (25 ns) because of the narrow YIG filter, and slow AM response when the AM is simultaneous with pulse modulation.

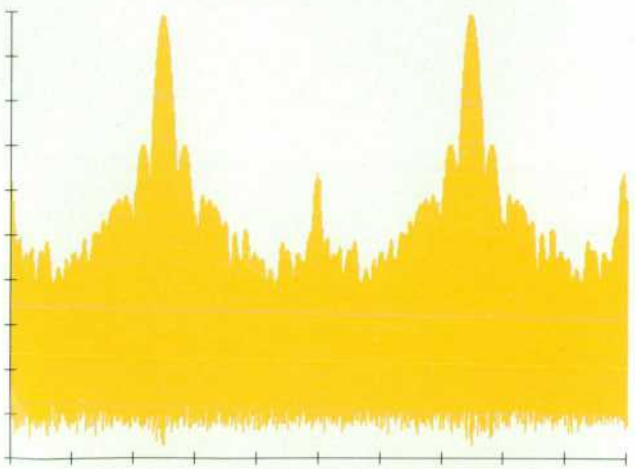


Fig. 1. Rotating antenna spectrum is an example of simultaneous pulse and amplitude modulation.

The HP 8370/70430 design takes advantage of new broadband gallium arsenide (GaAs) components to achieve the performance goals. Fig. 2 is a block diagram of the microwave chain. Subharmonics are eliminated by tuning a pair of broadband YIG-tuned oscillators (YTOs) over a 2-to-20-GHz range. Six low-pass filters reduce harmonics to less than –55 dBc. The low-pass filters have wide bandwidths, so they do not slow or distort pulses. Deep AM is linear in dB per volt and flatness with frequency is corrected to achieve –60-dBc depth. AM and pulse modulation can be used simultaneously.

Broadband noise in the output of a synthesized signal generator affects the sensitivity of a receiver under test and degrades noise figure meter accuracy when the signal generator is used as a local oscillator. In this design, AM noise is minimized by maintaining high power throughout the chain and not placing an amplifier at the output to boost power at the expense of noise. The YTO determines the AM noise at about 20 dB above the thermal noise level.

High Integration Level

To lower cost while increasing performance, the circuits after the YTO are integrated on two principal microcircuits, called the output module (Fig. 3) and the modulation module (Fig. 4). This maximizes performance by eliminating transitions, connectors, and cables that add loss and mismatch. The circuits are fabricated on PTFE*-based Duroid substrates and therefore can be relatively large without the circuit cracking encountered with both thick and thin films on hard substrates, which limits circuit sizes to about an inch or so. Both microcircuits are soldered into plated aluminum housings. To make it possible to use Duroid substrates with traditional chip and wire bonding, a bondable gold process was developed. Fabrication was simplified by careful tolerance analysis and the circuits are designed to eliminate RF adjustments, which can be costly. RF testing is done at the higher integration level. This improves yields over testing the lower-level circuits because a microcircuit can meet overall specifications and be accepted even

* PTFE stands for polytetrafluoroethylene.

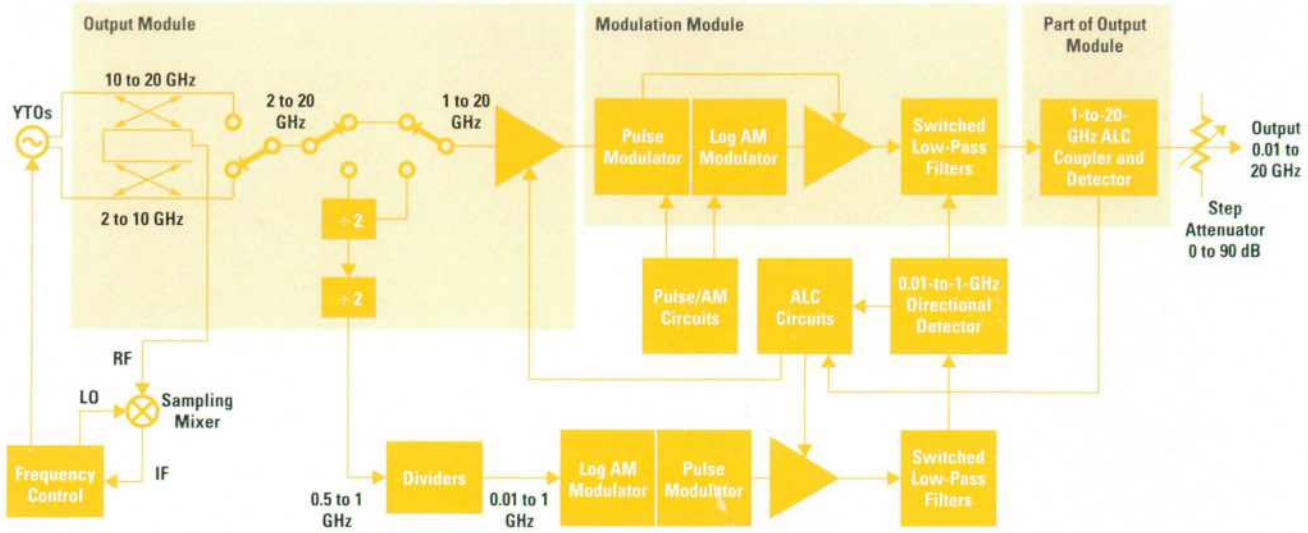


Fig. 2. Block diagram of the microwave chain of the HP 8370 and 70340 Series synthesized signal generators. Most of the microcircuits are integrated into two modules: the output module and the modulation module.

though one of its components may exhibit high loss and another low.

Further integration is achieved by using the amplifiers for more than one function. The first amplifier in Fig. 2, in addition to boosting power and buffering the YTO from AM frequency pulling, acts as the ALC modulator. The second amplifier is part of the pulse modulator.

Binary Dividers

The use of binary frequency dividers to generate the 10-MHz-to-2-GHz band departs from the heterodyne systems used in previous broadband signal sources. The decision to replace the microwave mixers, oscillators, and amplifiers used in heterodyne down-converters with high-speed digital integrated circuits was driven by receiver test requirements for reduced noise, harmonics, and spurious signals. Advances in frequency dividers, broadband components for modulation and leveling, and high-density surface mount circuits for harmonic filtering make it possible to produce the divider system at a much lower cost than heterodyne systems of equivalent performance.

Heterodyning a section of the microwave band to the 10-MHz-to-2-GHz spectrum involves design compromises. Mixers used in the frequency translation inherently generate in-band spurious signals. Drive levels to the mixer are reduced to minimize these spurs, decreasing the signal-to-noise ratio at the mixer output. Amplification used to bring the signal up to required levels raises the broadband noise floor as well. The mixing process duplicates the microwave phase noise characteristics in the RF band. Frequency division avoids these problems. Phase noise and spurious signals are reduced 6 dB with each octave of division. For a 70-MHz signal this results in a 36-dB improvement over heterodyne systems. The dividers handle moderate power levels and result in noise-floor improvements up to 20 dB.

Frequency division introduces some different design constraints. Output waveforms are square waves with a very rich harmonic spectrum. Third harmonics are only 9.5 dB below the fundamental and even-order harmonics, theoretically suppressed, require filtering. The harmonic specification of –55 dBc required filtering after the final amplification stages. Increased filtering to remove divider harmonics was added to the low pass-filter structures at minimal increase in product cost.

The modular nature of the cascaded dividers allowed low-cost frequency extension of the multifunction microcircuits from the 2-GHz low end of the YTO down to 1 GHz. Signal conditioning in the 1-to-2-GHz range is most economically

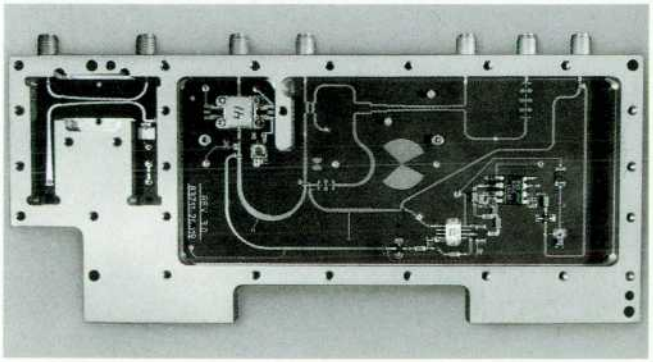


Fig. 3. Output module.

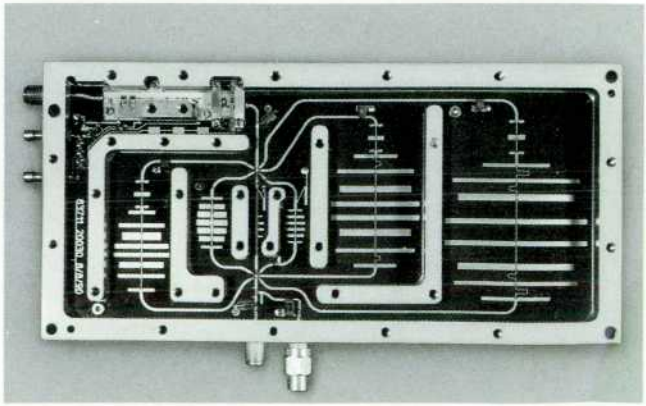


Fig. 4. Modulation module.

done with the distributed components in the microcircuits. The extra octave allows the base source to supply the bands in the 1-to-2-GHz range, which used to require several generators or a complete heterodyne system. Surface mount technology provides repeatable RF performance for lumped-element design of the 10-MHz-to-1-GHz band.

Level accuracy is achieved by a ratioing ALC loop that is accurate over level and temperature and is corrected for power flatness variation over frequency.

Output Module

The first of the two multifunction microcircuits, the output module (Fig. 5), performs four functions. The first function is to couple a portion of the oscillator output back to the frequency sampler and the frequency control subsystem. Two directional couplers, each covering a YIG-oscillator frequency band, were designed for this purpose.

The second function of the output module is to provide switching to route the microwave signal from the appropriate YIG oscillator band through the microwave chain. Thirdly, the output module divides the YIG oscillator output in the 2-to-4-GHz range by two or four to generate signals from 0.5 to 1.0 GHz and 1.0 to 2.0 GHz. The 0.5-to-1.0-GHz band is used to drive the low-band circuit board, which generates outputs in the 0.01-to-1.0-GHz range.

The fourth function of the output module is microwave signal processing for the automatic level control (ALC) subsystem. A broadband directional coupler detector samples the average power incident on the synthesizer load and feeds back the sample to the level control loop circuitry. The variable-gain traveling-wave amplifier acts as the leveling loop control element, modulating the microwave power to the desired level.

A PTFE-based material, RT/Duroid 5880, was chosen as the substrate for the output module. This material has a low loss tangent (0.0009 at 10 GHz), can be processed in a printed circuit board shop (resulting in low cost), and can be used for large circuits that have complex outlines (unlike a large piece of alumina, which will crack when subjected to thermal stresses after being been laser-routed into a complicated shape).

Extensive use was made of CAD tools in the development of the components that are integrated into the output module. These tools included linear circuit simulators, system simulators, and graphical layout tools. One extremely useful program, written by Jeff Meyer of HP's Systems Solutions Division, computes the solution of Laplace's equation in a two-dimensional region. The program uses the method of moments to solve for the Laplacian potential. Properties of unconventional transmission line structures can be analyzed

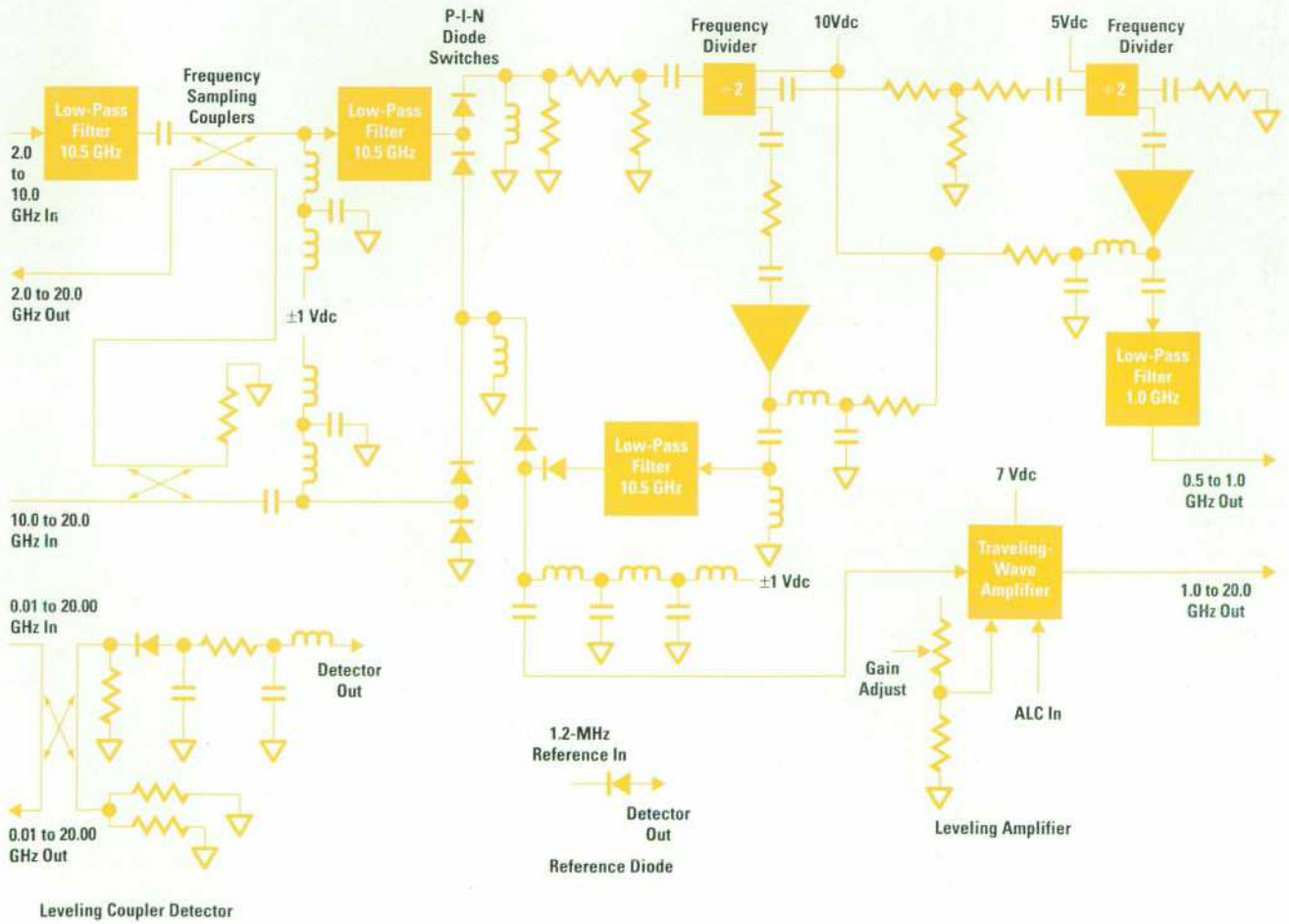


Fig. 5. Block diagram of the output module.

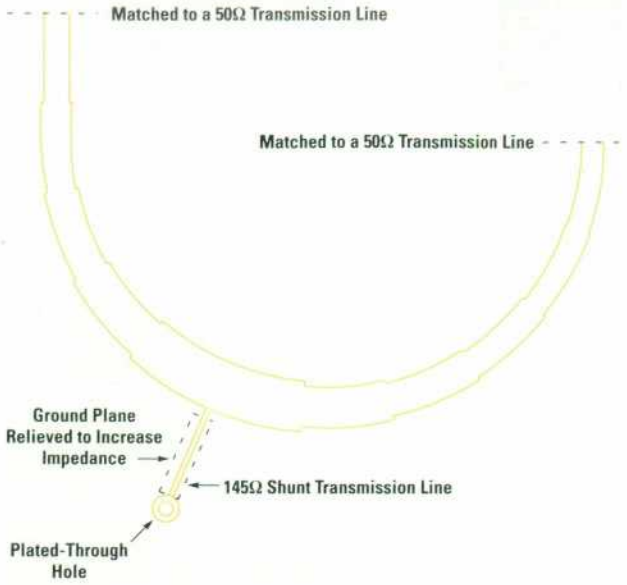


Fig. 6. Printed ground return with shunt stub in a diode switch circuit.

with the help of this program, and as a result, it was possible to design circuits not realizable with ordinary microstrip transmission line.

Switching in the output module is done with p-i-n beam-lead diodes. Two switches cover the 1-to-20-GHz frequency range. One is a double-pole, double-throw switch and the other is a single-pole, double-throw switch. It was a design goal to use as many printed components as possible to lower the assembly cost and ensure repeatable performance from unit to unit. The printed ground return shown in Fig. 6 provides a dc short while passing microwave signals from 2 to 20 GHz with a minimum of 20 dB return loss. The method of moments field solver program was used to design the 145-ohm characteristic impedance shunt stub in this circuit. Since the maximum impedance achievable with a microstrip transmission line on a 0.010-inch-thick RT/Duroid 5880 substrate is 110 ohms, it was necessary to modify the structure by etching the gap in the ground plane as shown in Fig. 7. The field solver was used to calculate the gap width required to achieve 145 ohms for a fixed top conductor width of 0.006 inch. By making H≫h, the transmission line impedance was

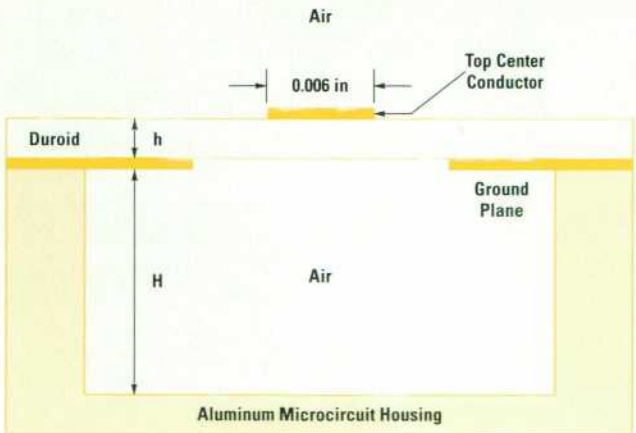


Fig. 7. Cross section of 145-ohm shunt stub.

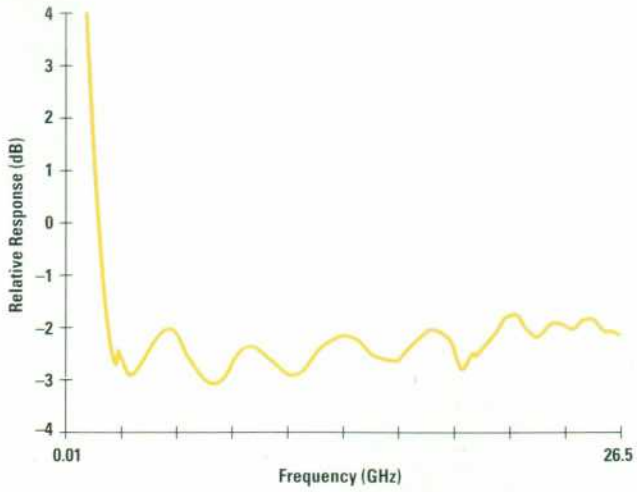


Fig. 8. Leveling coupler detector frequency response.

made insensitive to manufacturing variations in the housing dimensions and in the circuit attach process.

The leveling coupler is an asymmetric, continuously tapered, parallel-line coupler. The nominal coupling is 20 dB from 2 to 20 GHz. An RT/Duroid 5880 overlay, clamped in place with a silicone sponge-rubber pad, is used to equalize the even-mode and odd-mode velocities. The method of moments field solver was used to design this structure by generating a table of odd-mode impedance values versus coupling gap. This table was used to synthesize a design using ideal coupled transmission lines. The directivity of the coupler is typically better than 16 dB to 20 GHz. Good directivity in the leveling coupler results in a good source match and output level accuracy for the signal generator. The overlay and sponge clamp are thick enough so that currents are not induced in the package lid above the coupler. This allows a smooth transition to conventional microstrip. A broadband microstrip detector using planar doped barrier beam-lead diodes is therefore easily integrated with the coupler into a single housing. Integrating the leveling coupler detector with other circuits into one module yields substantial cost savings relative to previous signal generator block diagrams in which the coupler was a connectorized component purchased from an outside vendor and the detector was separately housed in an expensive package. The integrated design also allows the addition of a reference diode for temperature compensation. Figs. 8 and 9 show typical performance for the coupler detector.

Two additional microstrip couplers in the output module cover the 2-to-10-GHz and 10-to-20-GHz frequency bands. These couplers are used to feed back the signals from the frequency synthesis section to a frequency sampler. The construction of the 2-to-10-GHz coupler is similar to that of the leveling coupler. Its coupling ratio is 20 dB at 2 GHz and rolls off at 6 dB per octave to 10 GHz. The coupler was designed this way to reduce the level of harmonics incident on the frequency sampler. The 10-to-20-GHz coupler is entirely planar, making it very inexpensive to manufacture. It is a three-section, symmetric coupler with a nominal 33-dB coupling ratio. The even and odd modes are equalized by a printed capacitor at each end of the center section.

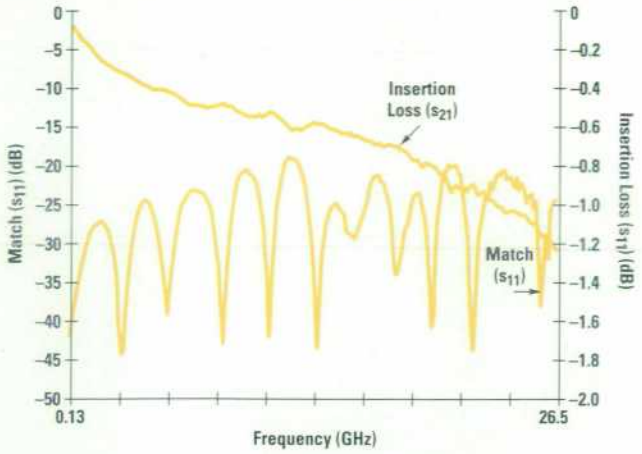


Fig. 9. Leveling coupler/detector insertion and return loss (match).

The broadband traveling-wave amplifier (described later) is housed in a hermetically sealed, ceramic package. This ensures reliable performance of this GaAs integrated circuit over varied environmental conditions such as high humidity and temperature. Careful engineering ensures that the ceramic package does not degrade the microwave performance of the traveling-wave amplifier chip to 20 GHz.

Two static frequency dividers generate the 1-to-2-GHz band, which is switched back into the 1-to-20-GHz path, and the 0.5-to-1-GHz output drive for the 0.01-to-1-GHz divider band.

Modulation Module

The functions of the second microcircuit, the modulation module, include pulse modulation, amplitude modulation, and harmonic filtering, all from 1 to 20 GHz. This module also contains a low-distortion combining switch for the 0.01-to-1-GHz band. The modulation is implemented with a 0.010-inch thin-film alumina circuit on a 0.015-inch molybdenum carrier for thermal expansion matching and a hermetic thick-film alumina-packaged traveling-wave amplifier. The harmonic filtering and broadband combining switch are implemented on a 0.010-inch Duroid substrate. Fig. 10 is a diagram of the modulation module.

AM for the instrument is provided by a modulator containing five shunt p-i-n diodes.[1] Originally designed for pulse modulation, in this case the modulator is used to provide 60 dB of logarithmic AM. The transfer function of the AM modulator is corrected as a function of modulation depth and frequency with gain and offset DAC (digital-to-analog converter) adjustments. Fig. 11 shows the deviation from linearity as a function of depth and frequency. The driving function departs

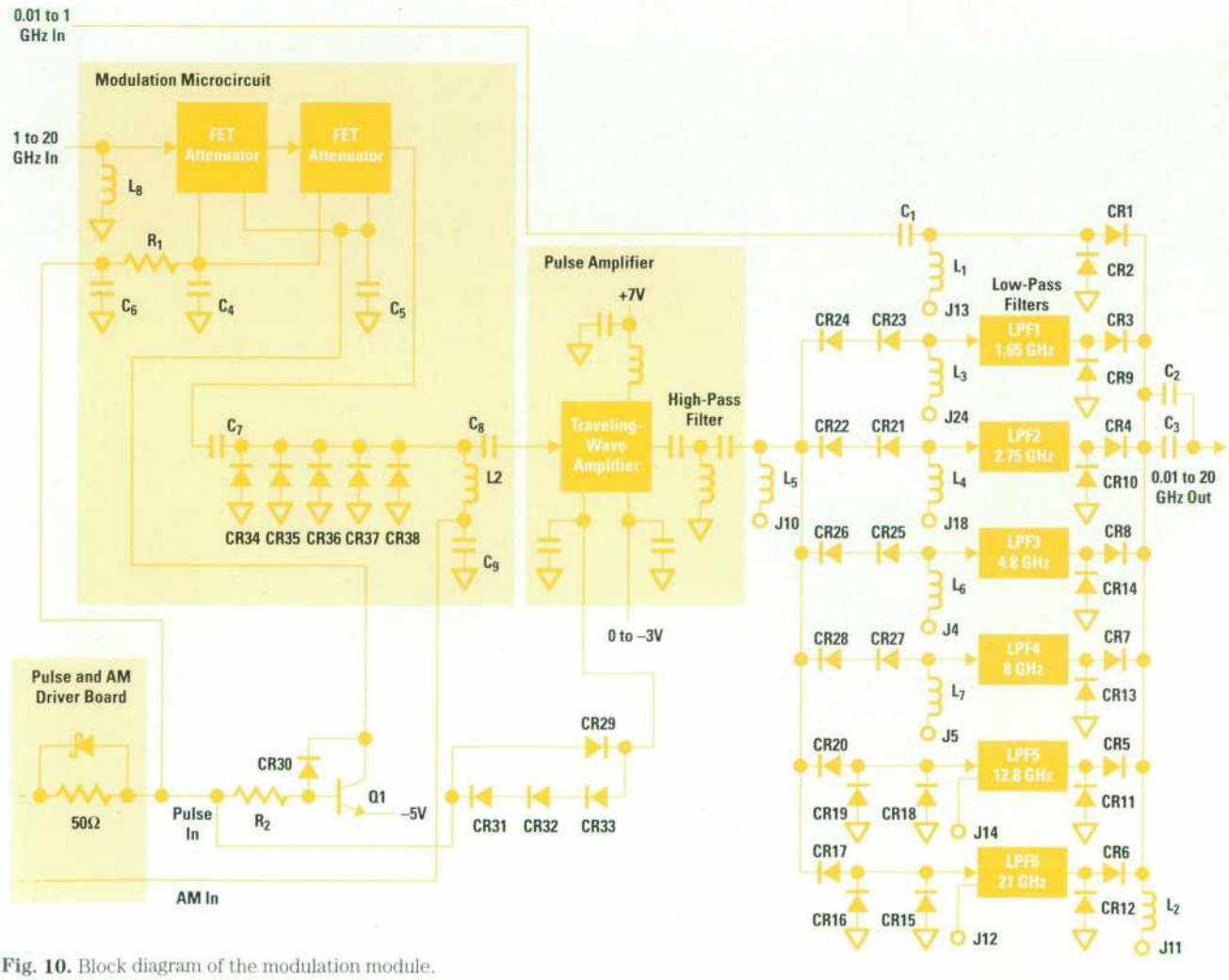


Fig. 10. Block diagram of the modulation module.

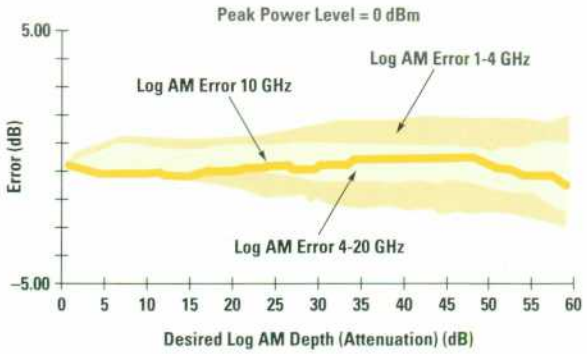
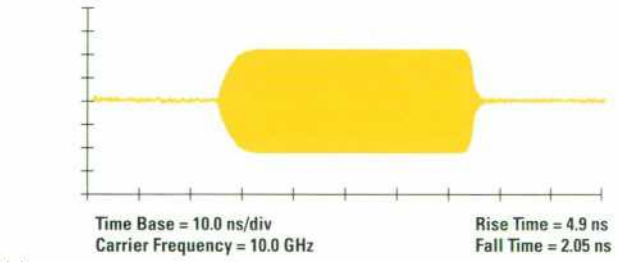


Fig. 11. Typical log AM error (deviation from desired depth) at 25°C for carrier frequencies between 1.0 and 20 GHz.

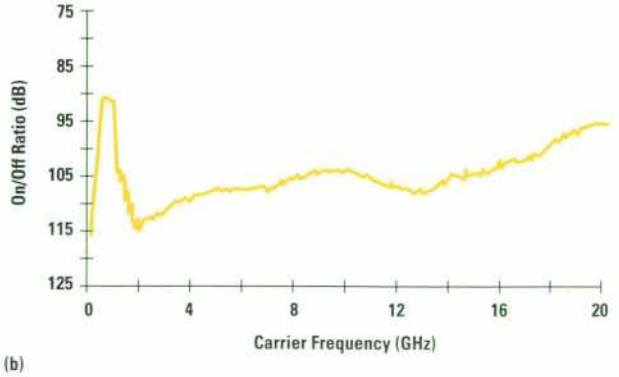
from traditional AM in that it is logarithmic (–10 dB/V). The log driving function is ideal for simulating large dynamic ranges like the deep nulls in an antenna scan pattern.

Pulse modulation for the instrument is implemented with three GaAs ICs: two attenuators and one traveling-wave amplifier. Specifications include 10%-to-90% pulse rise and fall times less than 10 ns, pulse on/off ratio >80 dB, <10% pulse overshoot, ±1-dB pulse level accuracy (relative to the CW level), and <20-mV peak-to-peak video feedthrough. Fig. 12 shows a typical pulse and a typical pulse on/off ratio plot.

Attenuators. The attenuators are series-shunt-series attenuators (Fig. 13). The series elements are common-gate FETs with 50-ohm resistors connected in parallel from drain to source. The shunt element is composed of four common-gate FETs connected in parallel with some transmission line spacing between them. This spacing contributes to a good



(a)



(b)

Fig. 12. (a) Typical pulse modulation envelope. (b) Typical pulse modulation on/off ratio at +8 dBm.

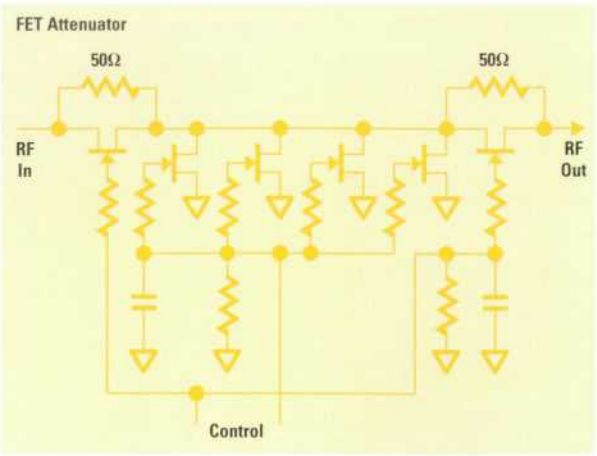


Fig. 13. FET attenuator.

match. It also results in a lower on/off ratio at low frequencies, where the spacing is negligible compared to a wavelength. The gates of the shunt FETs are driven by signals that are complementary to those that drive the gates of the series FETs. In the on state the series FETs are on and the shunt FETs are off, allowing the signal to pass. In the off state the series FETs are off and the shunt FETs are on, thus connecting the 50-ohm resistors that are in parallel with the series FETs to ground through the shunt FETs. This maintains a good match in the off state, unlike p-i-n diode modulators. This preservation of match in both states contributes to the good pulse envelope fidelity.

Amplifier. The traveling-wave amplifier is a seven-stage cascode design (Fig. 14). The gates of the common-gate FETs of the cascode are used to control the amplifier gain. In the on state, this control is high and the amplifier provides about 6 dB of gain. In the off state, this control is low, pinching off the FETs and turning the amplifier off. In this way the amplifier contributes to the pulse on/off ratio. The frequency response is dominated by the gate-to-source capacitance of the common-source FETs, so the on/off ratio is lower at higher frequencies. The traveling-wave amplifier thus complements the attenuators, compensating for loss and on/off ratio frequency response. Also, the match of the traveling-wave amplifier is maintained in both the on and off states.

Pulse Performance. Other factors that affect pulse on/off ratio are leakages and waveguide modes. The main leakage path is through the gate of the first series FET of the first attenuator, through the bias circuitry, and into the traveling-wave amplifier control. Bypassing is required to reduce this leakage to acceptable levels. Waveguide modes are suppressed with polyiron and narrow channels in the housing.

The pulse level accuracy is dominated by thermal effects in the traveling-wave amplifier. In the off condition, the amplifier cools down. When it is turned on, the gain is initially higher and drops as the traveling-wave amplifier heats up, with a time constant on the order of 10 μs. Pulse level changes are typically about +0.5 dB from CW levels.

The pulse rise and fall times are dominated by the traveling-wave amplifier switching time. The bypassing element

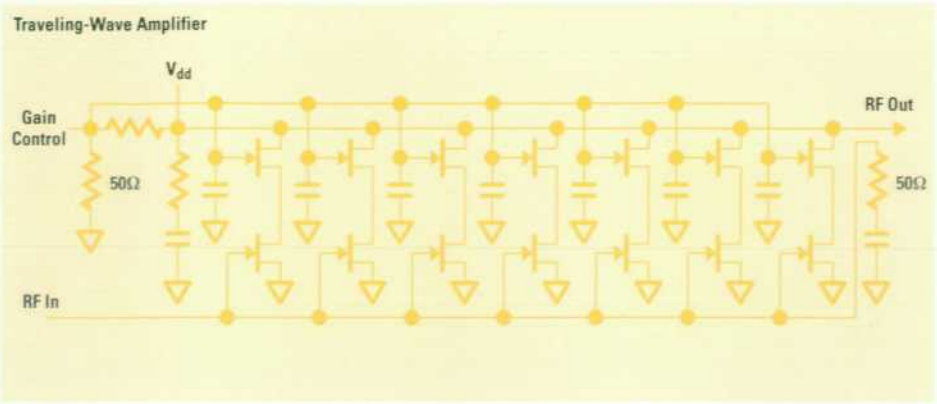


Fig. 14. Traveling-wave amplifier.

requirements impose a lower limit on capacitance at the traveling-wave amplifier control input. Also, there are internal capacitors on each of the seven stages' second gates. The drive source impedance is nonlinear: the off-state impedance is low, and the on-state impedance is 50 ohms. A low impedance for the off state provides quick discharging of the bypass capacitance, and the matched impedance for the on state maintains good pulse envelope fidelity (a low-impedance driver in both states would cause pulse overshoot and ringing). The driver used is simply a low-source-impedance circuit with a 50-ohm series output resistor and a Schottky diode in parallel with the 50 ohms (Fig. 10). For signal generator models without AM and pulse modulation the carrier circuit is replaced by a through line on a Duroid substrate.

Filters. The ≤55-dBc harmonic performance of the instrument is achieved with a switched low-pass filter assembly. Six filters connected between two 1P6T (one-pole, 6-throw) p-i-n diode switches strip the harmonics from signals between 1 GHz and 20 GHz. The output combining switch also includes an extra throw for 0.01 to 1 GHz, making it a 1P7T switch. Harmonic filtering is not done for this path. The p-i-n diode used for this throw is a 1-μs-lifetime diode, which maintains the harmonic performance down to 0.01 GHz. Fig. 15 shows typical second-harmonic performance; higher-harmonic performance is typically better.

The low-pass filters are distributed quasi-elliptic low-pass filters. They were computer optimized, emphasizing above-band spurious response. The lower three filters are asymmetric, which gives almost double the number of distinct transmission zeros in the stop band, and they are cascaded with five-element, higher-frequency, distributed low-pass filters to reduce spurious transmission bands.

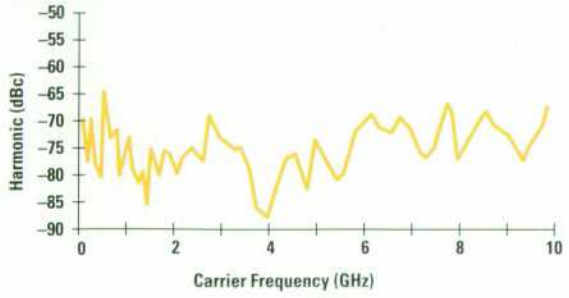


Fig. 15. Typical second-harmonic levels measured at an output power of +6 dBm.

The next two filters are optimized, symmetric, distributed Chebyshev low-pass filters. The last filter is only four shunt stubs on a 50-ohm line. This makes its loss lower, and it is easy to cut off the stubs to create a 50-ohm through line. The filter cutoff frequencies are 1.65 GHz, 2.75 GHz, 4.80 GHz, 8 GHz, 12.8 GHz, and 21 GHz.

Switches. The p-i-n switches are compensated to match the off diodes' capacitance. The input splitting switch has four series-series arms and two series-shunt-shunt arms, the latter for isolation and low loss at higher frequencies. The output combining switch has six series-shunt arms and a 0.01-to-1-GHz arm farther down the line. In the 0.01-to-1-GHz arm, the long-lifetime p-i-n diode's capacitance serves as the middle element of a five-element distributed low-pass filter.

The bias elements are distributed where possible for low cost, and all have capacitive feedthroughs to connect to a standard printed circuit board for bias switching circuitry.

An example will illustrate the operation of the switches (see Fig. 10). Consider the operation at 2 GHz. Only the LPF2 path is biased on and all other paths are biased off. This is accomplished by the application of +15V to J18 and a negative current for J4, J5, J12 through J14, and J24. J10 and J11 are each connected to 330 ohms to ground. The +15V on J18 forward biases CR4, CR21, and CR22, the current being set by the voltage and the 330-ohm resistors. CR10 is reverse biased by this same +15V, and the path through LPF2 is established. The other paths' negative currents forward bias CR2, CR9, CR11 through CR16, CR18, and CR19. This also sets up a reverse bias on CR1, CR3, CR5 through CR8, CR17, CR20, and CR23 through CR28. Thus the signal can only flow through LPF2. An important consideration is that the low-pass filters that are off do not disturb the match. For this reason series-only diode switches can't be used. For example, if CR9 were not present, LPF1's output reflection coefficient, which has magnitude unity and arbitrary phase because of the line lengths required to connect the circuit, could resonate with the capacitance of reverse biased CR3 to make a short circuit at the output switch's common pole. The input switch uses series-series diodes to combat the same effect, which works only because of the finite Q of the reverse biased diode's capacitance.

Low-Band Output Section

The low-band section (Fig. 16) provides the 10-MHz-to-1-GHz frequency band by dividing the 500-MHz-to-1024-GHz signal produced by the divide-by-four circuits in the output

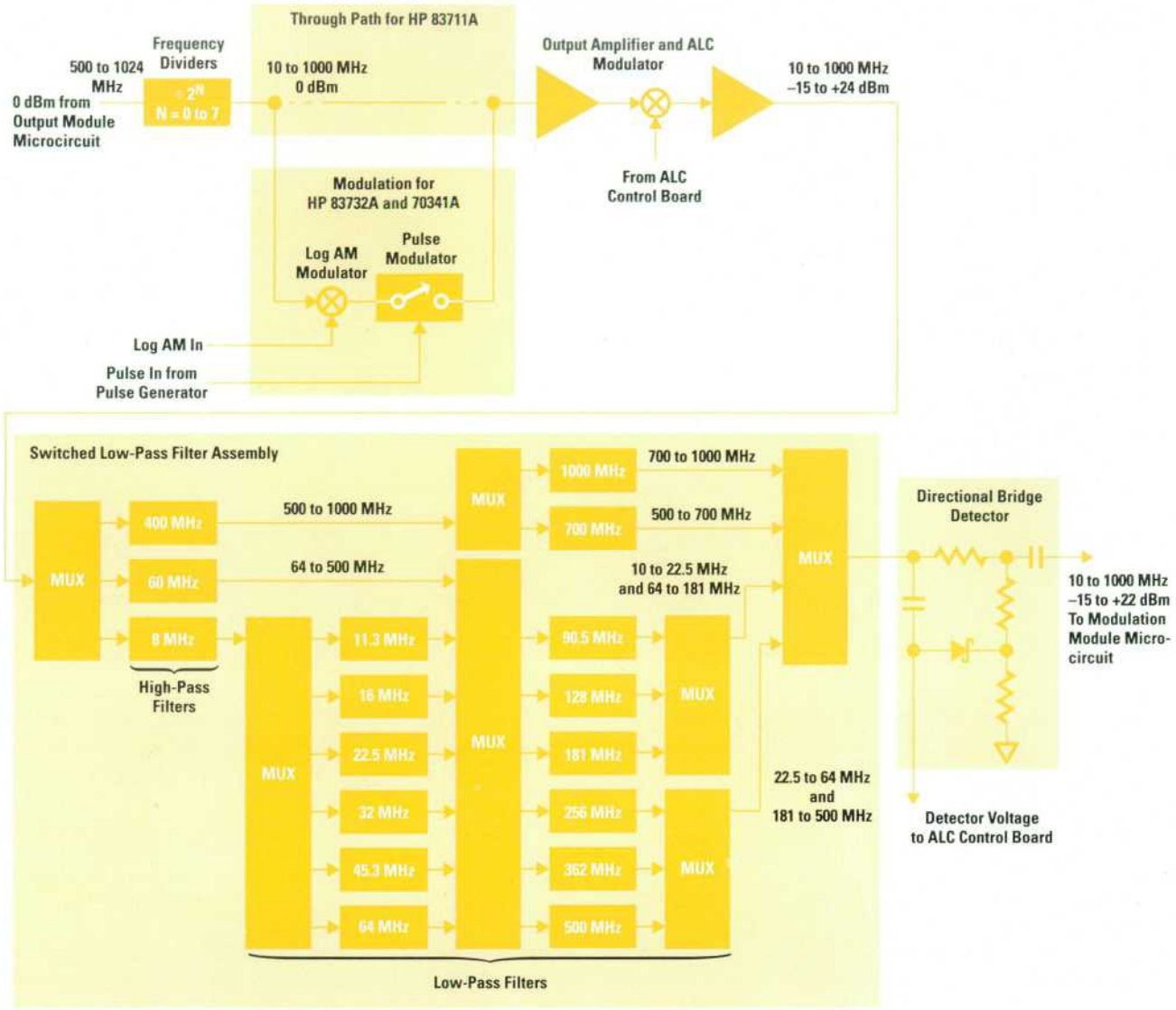


Fig. 16. Block diagram of the low-band output section, which provides 10-MHz-to-1 GHz frequency coverage.

module microcircuit. The low-band input signal is divided, amplified, filtered to reduce harmonics, and detected for automatic level control (ALC). For modulated signal generator models, logarithmic AM and pulse modulators are inserted between the divider output and the amplifier input. The –55-dBc harmonic requirement inspired placement of filters at the end of the signal chain, easing the linearity constraint on the circuit blocks preceding the filters.

To minimize the impact on the cost of the 1-GHz-to-20-GHz base model, a modular design approach was taken. The low-band system consists of self-contained CW, modulation, and directional bridge detector printed circuit boards. Leveling is accomplished by ALC detection and drive signals with characteristics similar to the microwave system, allowing common use of the ALC circuits. This results in a system to which either a CW or a modulated frequency extension can be added without affecting the 1-GHz-to-20-GHz band.

Dividers. The divider block consists of a through path for the 500-MHz-to-1-GHz signal from the microwave output module microcircuit and six cascaded divide-by-2 stages to develop the full 10-MHz-to-1-GHz band. A broadband limiting amplifier is used to sum all seven octaves of divider outputs. The limiter's output is a 0-dBm square wave with good match. Power level at this point is relatively independent of divider drive levels. Because of the performance limitations of the dividers above 300 MHz and the finite bandwidth of the limiter, second harmonics are as high as –12 dBc. Third harmonics reach the –9.5 dBc level predicted by analysis of a square wave.

Output Amplifier. One output amplifier covers the entire 10-MHz-to-1-GHz band. It consists of four cascaded gain stages with a p-i-n diode modulator between the first two stages. The 25-dB-gain amplifier provides a minimum power of +24 dBm. A 40-dB-dynamic-range p-i-n diode modulator is used for ALC. For a stable ALC loop bandwidth over all power levels, a constant dB/volt transfer function is desired. A shaped exponential current driver ensures a 3-dB/volt drive characteristic that closely matches that of the microwave system.

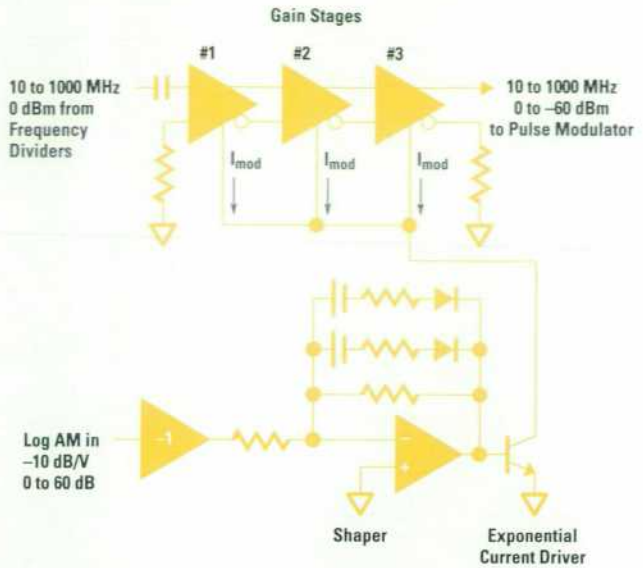


Fig. 17. Low-band logarithmic AM modulator.

Harmonic Filters. The low-band switched low-pass filter network covers two decades from 10 MHz to 1000 MHz. Implemented entirely as a lumped-element design with surface mount components for compact size, the network consists of 14 low-pass filters, three high-pass filters, the p-i-n diode switches, and drivers to select the switches. The low-pass filters are spaced two per octave, providing 45 dB of second-harmonic rejection at the low end of the band and greater than 55 dB of third-harmonic rejection. Stop-band spurious responses are lower than –50 dBc through 2 GHz to suppress the very rich harmonic spectrum of the dividers.

All filters are seventh-order elliptic designs. Elliptic filters have the very steep transition band required for second-harmonic rejection by virtue of transmission zeros in the stop band.

The network is separated into three frequency bands, each with an input high-pass filter to eliminate lower-frequency video components generated by the pulse modulator. Two filters covering 500 MHz to 1000 MHz are grouped together as one band to minimize insertion loss by keeping path lengths short. The six filters that span 64 MHz to 500 MHz serve double duty as harmonic filters for this range and as postfilters for the remaining six 10-MHz-to-64-MHz low-pass filters. The filters have stop-band spurious resonances primarily because of the self-resonance of the inductors used. For the lower-frequency bank of filters these occur below the 1000-MHz system bandwidth. High-order divider harmonics would degrade the –55-dBc harmonic performance should they fall on a stop-band spurious response. All signals in the 10-MHz-to-64-MHz filter bank are subsequently routed through a filter in the 64-MHz-to-500-MHz bank to suppress the spurious responses. For example, when the 32-MHz filter is selected, the signal is further filtered by the 256-MHz low-pass filter. Spurious resonances of the remainder of the filters are above 1000 MHz, where the gain roll-off of the output amplifier effectively reduces the high-frequency harmonic content of the dividers.

Long-lifetime surface mount p-i-n diodes are used in all low-frequency filter paths to prevent distortion at higher power levels. The insertion loss of the network is typically 3.5 dB. The network occupies a 3.5-by-4.5-inch area.

Amplitude Modulation. The low-band logarithmic amplitude modulator is a broadband 10-MHz-to-1000-MHz design. It is used with an open ALC loop, in the hold position of the track-and-hold mode. This avoids reduction of modulation bandwidth during simultaneous pulse modulation and AM, a limitation of previous ALC systems. Also, AM depth is not restricted to the floor of the level detector. Since there are separate modulators for AM and ALC, the full 60-dB AM range is available at all power levels. However, without ALC feedback to correct for inaccuracies or drift, the modulator must exhibit constant drive sensitivity and flat frequency response at all attenuation levels.

The modulator is a three-stage differential amplifier capable of 75-dB dynamic range at 1000 MHz (Fig. 17). Each stage consists of two cross-coupled differential pairs. Cross coupling results in a transfer characteristic that is the gain difference of each pair. Gain is controlled by varying the difference current in the emitters of the differential pairs. Minimum gain occurs when both pairs are driven at exactly the same current level, the signal at one collector canceling the signal from the other.

The first stage (Fig. 18) is always in the limiting mode, driven at a high RF level. The output signal is independent of the 10-MHz-to-1000-MHz level and is equal to $I_{mod}R_c$, the difference current of the two pairs times the collector resistance. At low attenuation levels this signal is enough to fully switch stages 2 and 3. The final output is again the difference current, I_{mod}, into the load resistance R_c. As attenuation is increased, I_{mod} is reduced. Eventually the signal from the first stage is too small to fully switch stage 2 and stage 2 cannot fully drive stage 3. The gain of these output stages is now dependent on the drive levels of the previous stage as well as the difference current, I_{mod}, at the differential emitters.

When the output signal is plotted as a function of I_{mod} on a logarithmic scale, gain is seen to vary from 20 dB per decade change of I_{mod} at low attenuation levels to 60 dB per decade change of I_{mod} at high attenuation levels. Generating I_{mod} from an exponential current source and then shaping the input to the exponentiator results in a –10-dB/volt modulation input characteristic (Fig. 19).

A minimum current, I_{bias}, is always run through both sides of the differential pairs to keep the transistors operating at a high gain-bandwidth point. This prevents the modulator frequency response from changing when the modulator drive is changed. Temperature stability is inherent because the stage gain is the difference between the two monolithic differential pairs, which tend to drift at the same rate.

Pulse Modulation. The 10-to-1000-MHz pulse modulator is composed of two single-pole, double-throw GaAs switches. On-off ratios greater than 90 dB are achieved with just two devices. An interesting feature of this modulator is the three-speed variable-rise-time control. This feature is necessary because the switched low-pass filter network follows all of the modulation. If the RF rise time is too fast going into a filter, the output of the filter shows an excessive amount of ringing and overshoot. The variable RF rise time is achieved by low-pass filtering the complementary control

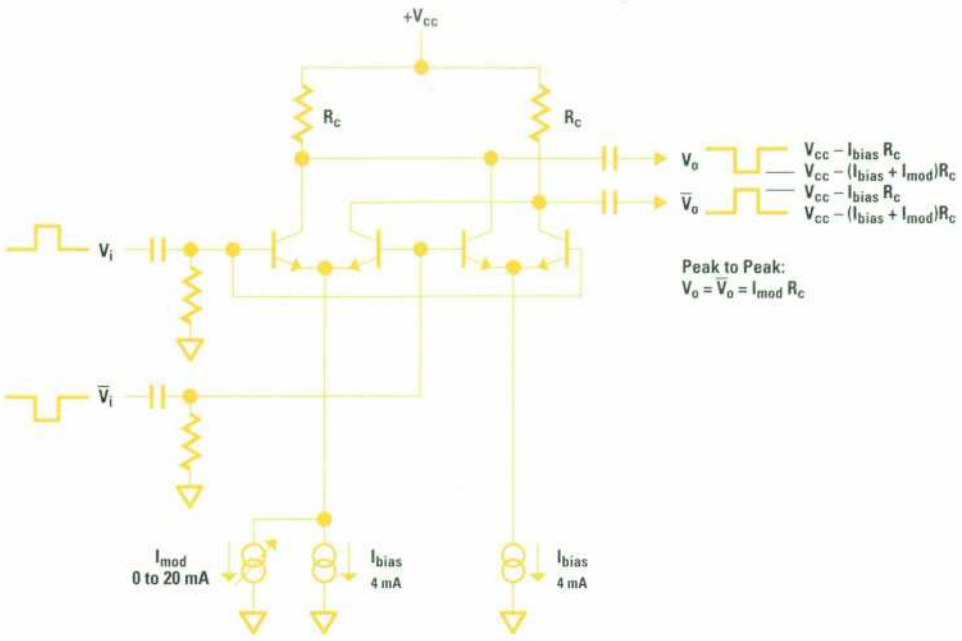


Fig. 18. Logarithmic AM modulator differential gain stage.

voltage inputs of the GaAs switches. Passive RC filters are used, with an analog switching network to select one of three sets of filters. The three rise times correspond to three frequency bands: 10 to 64 MHz, 64 to 500 MHz, and 500 to 1000 MHz.

Power Leveling

The automatic level control (ALC) system is a feedback loop that allows precise control of the microwave output power level over the signal generator's full output power range (from maximum power to –90 dBm) and frequency range (0.01 to 20 GHz). The main components of the ALC system are shown in Fig. 20. The microwave output power is sampled by the microwave coupler and detected by the diode sensing element. The diode transfer function is highly nonlinear and varies significantly with temperature. However, over a large part of its dynamic (operating) range, the detector's dc voltage output changes by approximately a decade for a 10-dB change in microwave input power. If this detector output voltage is logarithmically converted, the overall transfer function of the detector and the logarithmic amplifier is approximately a constant. At 25°C this constant varies from 4 to 6 mV per dB of input power change.

Traditionally, logarithmic amplifiers have been used at the outputs of microwave detectors to compensate for the nonlinear diode transfer function. The transfer function of the microwave amplifier gain as a function of gain control voltage complements that of the detector and log amplifier and is instrumental in the determination of the system loop transfer function and the control system bandwidth. The gain of the microwave amplifier can be adjusted over approximately a 35-dB range by applying a control voltage to the second-gate control line. This increases the harmonic distortion generated by the amplifier, especially near the device pinch-off voltage, but the harmonics are adequately filtered before entering the microwave coupler and therefore have minimal effect on the level accuracy.

(continued on page 28)

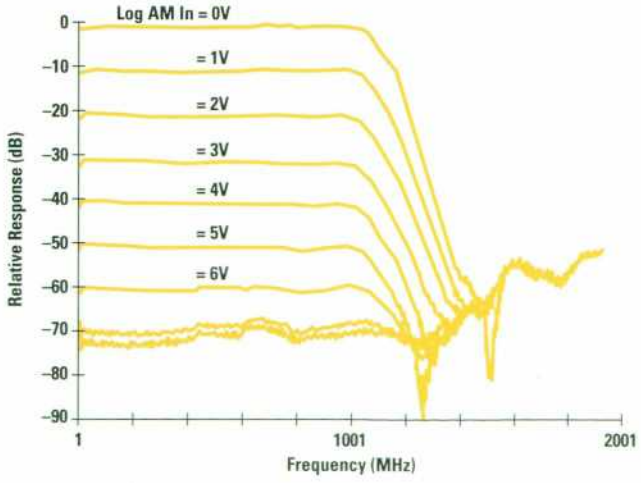


Fig. 19. Frequency response of the logarithmic AM modulator.

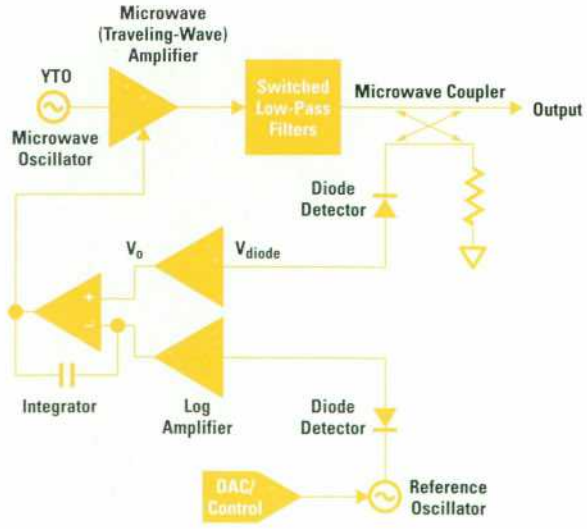


Fig. 20. Automatic level control loop.

Internal Pulse Generator

This section describes the internal pulse modulation source in the HP 83732A microwave signal generator. The pulse generator is digitally based, and is implemented in a programmable gate array. Pulse width, pulse repetition interval, and pulse delay can be independently controlled.

The pulse generator can operate in five different modes: external pulse mode, internal pulse mode, triggered pulse mode, pulse doublet mode, and gated pulse mode.

External Pulse Mode

In external pulse mode the pulse generator simply passes a TTL-level input signal from the pulse/trigger input to drive the pulse modulator. The input polarity can be inverted. The input signal is buffered and passed on to the video output.

Internal Pulse Mode

In internal pulse mode the pulse generator provides TTL-level signals to drive the pulse modulator of the HP 83732A and the sync and video outputs on the front panel. Three parameters can be controlled from the keyboard or through HP-IB (IEEE 488, IEC 625) commands. These are the pulse repetition interval, the pulse width, and the delay from sync to video. The sync output pulse width is a fixed 50 ns, while the video output pulse width is programmable. The RF output pulse width is the same as the video output minus RF pulse compression. Both the sync and video outputs have a nominal 50-ohm output impedance. They provide +5 volts into high impedance and greater than +2.5 volts into a 50-ohm load. In this mode, "negative delay" is possible (see below).

Triggered Pulse Mode

In triggered pulse mode the pulse generator provides the same TTL-level signals as in internal pulse mode. However, the pulse/trigger connector on the front panel is used as a trigger input. This externally supplied trigger signal provides the pulse repetition interval. It is rising-edge triggered from a pulse greater than 25 ns wide. Only the pulse width and delay parameters are variable. The sync output is generated at a fixed minimum time delay from the pulse/trigger input. The pulse generator rejects triggers during the middle of a cycle. Thus, the pulse generator can divide an external trigger source in this mode.

Pulse Doublet Mode

In pulse doublet mode the pulse generator operates in the same manner as in triggered pulse mode, but with additional capability. The external trigger signal is also passed through to the pulse modulator as in external pulse mode. Thus, it is possible to get two pulses out for each input pulse. The first is just the external pulse mode output, while the second is controlled by the delay and pulse width settings for triggered pulse mode. If the two pulses overlap, then just one large pulse results, with the leading edge determined by the external input signal and the trailing edge determined by the pulse generator.

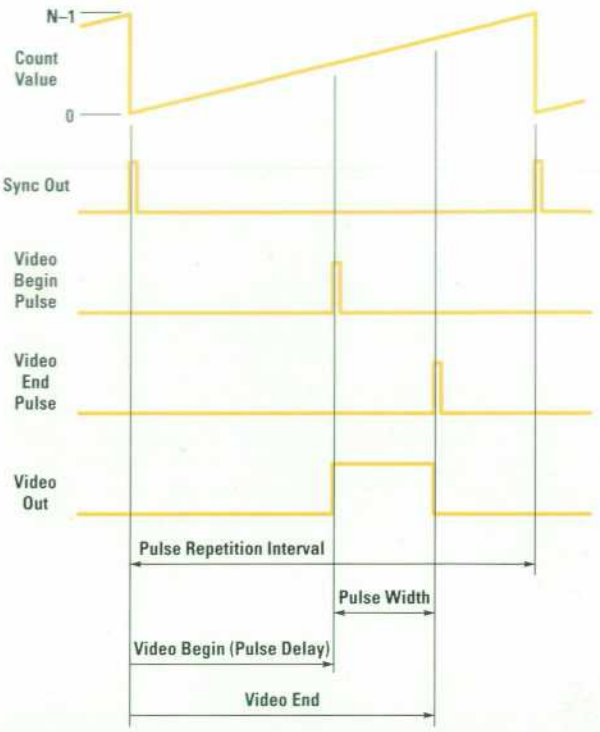


Fig. 2. Internal pulse generator timing relationships.

Gated Pulse Mode

In gated pulse mode the pulse generator responds to the external input in a level-sensitive manner. While the external input is high, the pulse generator free-runs as in internal pulse mode. If the external input is low, no pulses are generated. The pulse generator triggers on the rising edge of the external input, and always outputs complete pulses. Only the pulse width and the pulse repetition interval are variable.

(continued on page 28)

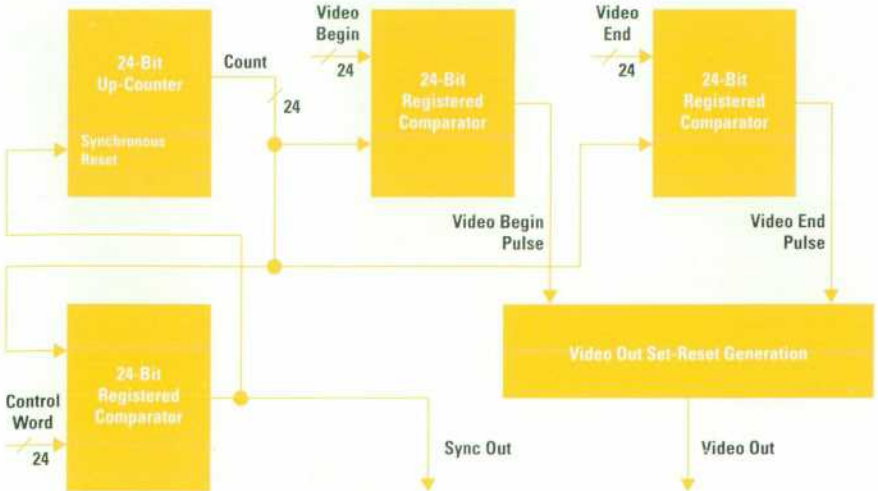


Fig. 1. HP 83732A signal generator internal pulse generator.

Implementation

The pulse generator is based on a single-counter design. Timing of width, rate, and delay are all determined from one 24-bit up-counter. Fig. 1 shows the basic architecture. The counter counts up from zero to (N–1), giving a total count length of N. When the count value matches the control word, a pulse is generated that synchronously resets the counter to zero and the cycle starts over. The reset pulse is also the sync output pulse, marking time zero of the cycle.

There are two other registered comparators. One marks the beginning of the video pulse and the other marks the end. The control words are 24-bit video begin and video end inputs. A small set-reset state machine converts these pulses into the actual video output pulse. Fig. 2 shows the details. Video begin and video end signals occurring simultaneously are defined as a reset signal, making a zero pulse width possible. Also, negative delay from the sync output to the video output is possible because of the arbitrary placement of the video begin and video end pulses.

The entire pulse generator is contained in one Xilinx XC3030 field-programmable gate array. An 88-bit serial interface is included in the gate array to allow loading all of the necessary pulse generator control data from the main instrument microprocessor. The pulse generator is clocked at 40 MHz, so all programmable parameters have 25-ns resolution.

This design is approximately equivalent to 25 to 30 SSI/MSI standard parts.

Douglas A. Larson
Development Engineer
Stanford Park Division

As shown in Fig. 21, the transfer function of the traveling-wave amplifier output power as a function of control voltage is approximately linear in dB/volt over most of the output adjustment range. The rest of the components in the feedback loop are linear in volts/volt (integrator, microwave filters, cables, etc.). Therefore, if the cascade of linear components has gain A, the loop gain will be approximately A(5 mV/dB)(10 dB/V).

Once the loop gain is set, the control loop bandwidth can be set by choosing the RC values of the integrator. Changes in loop gain will directly affect loop bandwidth. Since the detector transfer function changes as the diode is operated at higher power levels and the traveling-wave amplifier gain control saturates at higher control line voltages, there is a decrease in loop gain at high vernier levels. This in turn causes a corresponding decrease in loop bandwidth. The

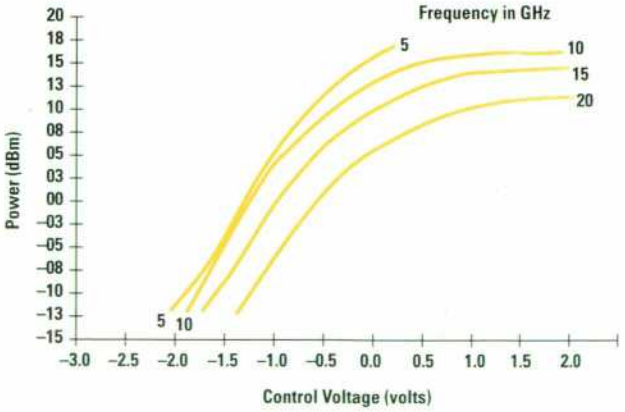


Fig. 21. Microwave (traveling-wave) amplifier output power as a function of control voltage.

loop bandwidth is least at high frequencies and high power levels. The loop bandwidth is greatest at lower frequencies and low vernier levels. The integrator pole was chosen so that amplitude level switching time is short enough at the minimum loop bandwidth, loop peaking is minimized, and loop stability is ensured at the maximum loop bandwidth.

The integrator provides the dominant pole in the system and the next closest pole is from the log amplifier. As the detector voltage decreases, the log amplifier gain increases and its pole decreases in frequency. A limiter on the log amplifier gain at low input levels establishes the minimum pole frequency. This prevents loop instabilities during transients in which no microwave signal is present at the detector input. The next pole is set at 100 kHz on the traveling-wave amplifier second gate control line. This reduces noise outside the ALC loop bandwidth.

The dynamic characteristics of the loop have been reviewed and the need for the log amplifier in the feedback path has been discussed. The log amplifier output voltage V_o is:

$$V_o = V_t \ln(V_{diode}/I_{sat}R).$$

There are two temperature dependent terms: V_t, which varies linearly with temperature, and a less predictable parameter, I_{sat}. I_{sat} and V_t are parameters of the logging transistor. Statically, incorporation of the log amplifier in the feedback path presents a problem and the nonlinear and thermal characteristics of the diode need to be compensated. Historically, these problems have been overcome through extensive characterization of diodes and log amplifier shaping techniques. Once data has been acquired on a diode's characteristics, temperature compensation is applied to the reference path of the integrator. This method is inherently less accurate because of the assumption that all diodes have identical characteristics. The dual log leveling loop circumvents both the thermal and the nonlinearity problems. The effect of the dual logging transistor is that the error terms associated with V_t and I_{sat} are ratioed out. The reference detector in thermal contact with the feedback diode has a similar effect: the nonlinearities and thermal variations of the diodes are also ratioed out.

The input to the reference detector is a 1-MHz sinusoidal signal. Its level can be precisely controlled with a DAC and it exhibits excellent temperature stability. Thus, given the symmetry of the reference path to the feedback path, the coupled microwave output power will be adjusted by the action of the servo loop to be identically equal to the power of the reference. All dynamic errors of the loop are calibrated out with the vernier calibration algorithm. In this algorithm, the DAC is stepped and the instrument output power is recorded. Then a curve is fit to the pairs of points, giving power out as a function of DAC number. Thus, for any desired output power, the required DAC value can be computed. This eliminates the need for adjusting operational amplifier offset voltages and compensates for slight differences in detector characteristics.

The most significant remaining sources of error within the feedback system are differential thermal characteristics in the detector and logging transistor pairs. The temperature drift caused by these effects is less than ±0.15 dB over the operating range of 0°C to 55°C. The largest sources of error

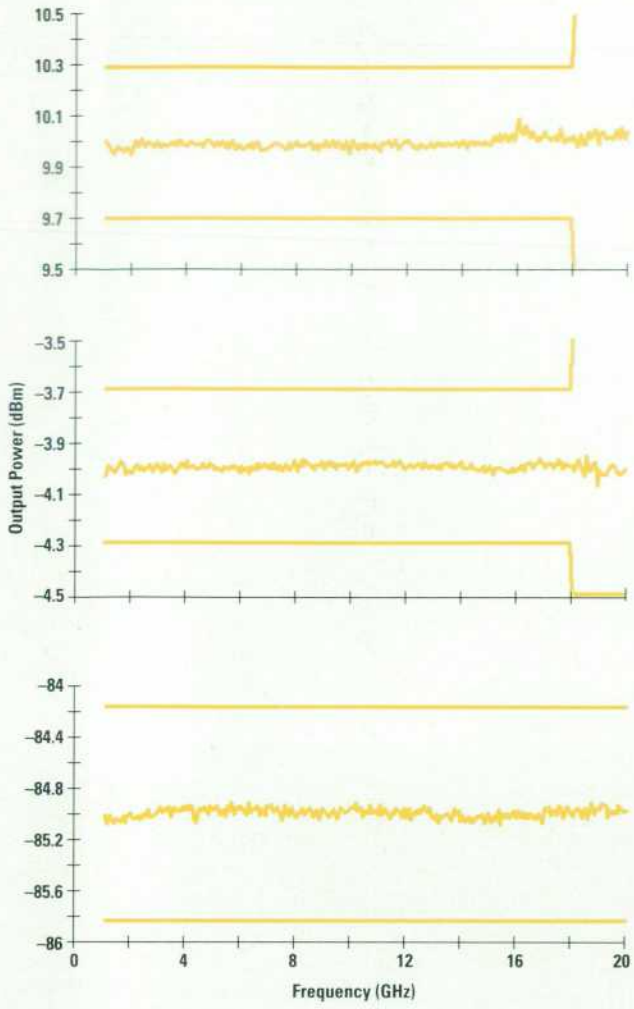


Fig. 22. Level accuracy of the HP 8370 and 70340 Series signal generators.

are changes in conductor losses of components outside the feedback loop, such as the cables and the step attenuator. This accounts for an additional ±0.25 dB of the level accuracy error budget.

A frequency flatness calibration digital correction table is generated to compensate for component losses outside the leveling loop and correct the frequency unflatness of the microwave coupler, thereby relaxing the design constraints on the coupler. HP's proprietary planar doped barrier diodes are instrumental in achieving the tight level accuracy specifications over the vernier operating range (see Fig. 22). Unlike their Schottky diode counterparts, planar doped barrier diodes have consistent frequency flatness independent of operating power level.

Acknowledgments

In any successful project there are many people who contribute to the overall success. Thanks to Soojin Choi for developing the bondable gold-on-Duroid-substrate process and ensuring the design was manufacturable during the design process, to Eddie Plantillas for development of the test system, to John Duran for endless hours of testing of the environmental and prototype units, to Harrell Huckeba for partial development of the modulation module, to Yvonne Vieira for her skill in building the breadboards and prototypes, to Ed Cirimelle for flawless housing designs, to Marlene Hartigan for keeping us organized in tracking our defects and solutions, to Jim Logie for setting up the fab parts and helping us out of the polyiron cracking problems, to the team at the Microwave Technology Division—Morgan Culver, Gene Burdick, Ted Shimkowski, and Sig Johnsen—for development of the ceramic hermetic traveling-wave amplifier package, to Orrin Baisden and Al Bates for keeping the material straight, and to Mark Johnston for coding all the algorithms and juggling an incredible amount of inputs.

Reference

1. M.K. Koenig, "A High-Speed Microwave Pulse Modulator," *Hewlett-Packard Journal*, Vol. 42, no. 2, April 1991, pp. 34-36.

Concurrent Signal Generator Engineering and Manufacturing

Production tests were developed early enough to be used for design characterization. Several new production processes were developed. The project had a design-for-assembly philosophy, an integrated assembly and pretest strategy, online video-image production procedures, and a networked computing test environment.

by Christopher J. Bostak, Camala S. Kolseth, and Kevin G. Smith

The project team for the HP 8370 and 70340 Series synthesized signal generators attempted to develop the requisite manufacturing processes in parallel with the product design. Key manufacturing and R&D personnel were collocated and functions were blurred as necessary to accomplish this objective. The lab pilot run used the planned manufacturing processes and personnel as soon as possible. By developing many of the production processes early, we were able to leverage the production tests to perform additional design characterization. Several new processes were developed to accomplish the overall objectives of quality, efficiency, and flexibility for a mixed-model, multiple-option production line. In this article, we describe the design-for-assembly philosophy, the integrated assembly and pretest strategy, the implementation of online video-image production procedures, and the networked computing test environment.

Design for Manufacturability

A number of design features facilitate the mixed-model production mode. The microcircuit chain is integrated and maintains the same form factor and connector locations in modulated and unmodulated instruments. The housings and covers are common parts. With the exception of the front panel, all the sheet-metal and chassis parts, including the subpanel and the rear panel, are identical across the family. The form factor of the circuit boards allows mounting in both the Modular Measurement System (MMS) and the HP System II enclosures. The interconnect strategy allows the simple addition of further enhancements by including excess capacity.

The design-for-assembly philosophy adopted at the beginning of the project is evident in the final design. There are several examples of part count reductions. The main deck assembly for the HP 8370 Series is a prime example. Instead of being screwed together on the assembly line, large portions of the chassis are riveted together at the sheet-metal facility. This reduces inventory and assembly costs. It also improves quality by guaranteeing that all the parts fit together. An error in fabrication is discovered sooner, and less scrap is produced.

The main deck assembly includes the main deck, the tang deck, the rear panel, two side struts, the fan cover, the oscillator bracket, the rear frame, and some hardware. Instead of pressing in captive inserts and screwing the parts together, these parts are riveted at the fabrication site, saving an estimated 44 parts in final assembly. Another part reduction is realized by attaching the display RF shield to the subpanel by a semipierce method. A hinged board service strategy was reengineered early in the development to eliminate a few dozen parts and replace them with one part.

The tang deck includes sheet-metal features that hold the boards in their service positions (see Fig. 1). This design change was directly driven by early production engineering involvement. In addition to part savings, the assembly is much simplified. To a great extent, the unit assembles from the top down and assembly reversals are minimized.

Assembly and Pretest

The HP 8370 and HP 70340 Series microwave signal generators are assembled and tested using a modern "layered" manufacturing approach combining video-image-aided instructions and in-situ testing by the assemblers. The traditional approach has been to assemble and test the various boards and modules separately with dedicated fixtures and then bring them all together to be tested as a final product. This

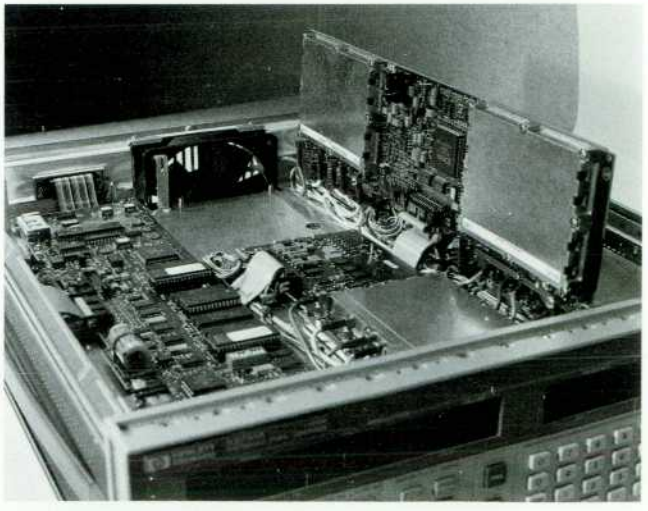


Fig. 1. HP 83731A showing the tang deck with board in service position.

Fig. 2. Manufacturing process flow, showing integrated assembly and pretest steps.

method can often lead to costly instrument-level troubleshooting and rework. Also, the assembly and test functions have historically been executed by separate personnel.

By testing the signal generator layer by layer (subassembly upon subassembly) we reveal as many potential failures as possible at their earliest and hence least costly points of opportunity. The term pretest, as used in this article, simply refers to testing an assembly short of the finished product and applying interim test criteria. We argue that the successful implementation of this approach, integrated with robust product design, will prove advantageous to companies with a highly diverse product mix and with medium to high production volumes.

In addition to early problem discovery, pretesting during assembly offers a number of other benefits. First, this method minimizes the cost of building, documenting, and handling test fixtures, jigs, adapters, and the like. Except for the test racks, any "fixturing" should be built into the product itself. Second, each assembly is tested in the actual instrument environment. Therefore, the assembly is subjected to more realistic component variations than by an ideal fixture. Finally, the discovery and resolution of failures early in the production process helps minimize costly cycle time resulting from failures in the final test stage of the process.

Approximately 23 principal assembly and pretest steps must be performed before the instrument is considered complete and begins final testing. The actual number of steps differs from model to model within the signal generator family. These steps are presently divided between two workstations or cells, Pretest 1 and Pretest 2, as shown in Fig. 2. Both are essentially identical and can handle all "pretest 1" and "pretest 2" procedures. This gives the line maximum flexibility. In Fig. 2, process steps that are pretests are indicated by open boxes. This partitioning largely follows the signal generator block diagram. Note that a number of items such as microcircuit modules, printed circuit boards, and power supplies are assembled elsewhere and may undergo additional prior screening. The assembly and pretest process described here is at the instrument level only.

Each assembly and pretest station consists of a well-equipped workbench with a computer terminal and an adjacent test equipment rack (see Fig. 3). A dedicated material handler ensures that kits are available to assemblers when

Fig. 3. Assembly/pretest workstation.

```
PRETEST STATION 1

================================================
>>==> 1. Assemble Front Panel   Pass
      2. Main Deck Assembly     Pass
      3. Powr,Fan,Cbls Install  Pass
      4. DC Cab/Microproc Assy  Pass
      5. ET Installation        Pass
      6. DC/Rpnl Cab, Osc Assy  Pass
      7. Power Supply Pretest   Pass
      8. uProcessor pretest     Pass
      9. Store S/N and Options  Pass
     10. Ref/Lo Assembly        Pass
     11. Ref/Lo funct pretest   Pass
     12. Ref syn pretest        Pass
     13. Lo syn pretest         Pass
================================================

23 Sep 1992  16:28:19   Serial Number: 3233A00131
_
                                          User 1   Caps   Running
SELECT  NextTest        Input #  Auto seq  LoopTest Utility EXITmenu
                                 uence
```

Fig. 4. Example of test executive menu for assembly and pretest steps.

needed. Hardware and other frequently used parts are stored in easy-to-reach bins on cantilevered arms at the fully adjustable, ergonomic workstations. Equipment commonality ensures that any pretest can be run at any pretest station.

A common operator interface is shared between the pretest station and other process workstations. This interface is provided by an HP 9000 Series 300 computer with a color graphics display running the HP-UX* operating system, the X Window System, HP BASIC/UX, and a test executive that generates menus. There is tremendous flexibility in this scheme. Assemblers and other test operators need very little additional training to become immediately productive on new workstations in the line.

Once an assembler logs onto the pretest station terminal, the video display presents the menus for product, station, and procedure selection. When the appropriate instrument type and station are selected, a menu listing the stepwise sequence is displayed (see Fig. 4). When a step is selected the test executive loads the proper assembly or test software and directs and prompts the operator by means of a series of images.

Types of Pretests

After an image-aided assembly procedure has been completed, the next step is often what is referred to as a functional verification pretest. This test leverages built-in test features and merely requires ac power and an HP-IB (IEEE 488, IEC 625) connection to the test rack. Functional verification tests use the digital and analog buses to determine power supply voltages, data readback capability, miscellaneous analog voltages, digital flags, and so on. In other words, is the DUT (device under test) minimally functional? The principal goal of this test is to determine, with a high degree of confidence, whether the DUT has been assembled correctly. A well-written explanation of a similar self-test and analog bus strategy can be found in reference 1.

The next step is the performance pretest, which uses equipment in the test rack to determine if the assembly is providing outputs within the specifications necessary for the next level of assembly. This step can provide information that the functional verification procedure cannot. Boards are tested for performance criteria such as frequency accuracy and RF power output. Each of the pretests takes approximately five minutes or less to execute. This is insignificant with respect to total process flow.

The system is intended to be simple. Each pretest provides pass/fail information to the operator. If the DUT fails, the assembler removes the last failed subassembly and passes it to the defect elimination station (Fig. 5) for troubleshooting. A known-good module from a pretest buffer is substituted for the failed module so that the unit can continue with minimal delay to the next assembly process. Consecutive failures on the retried pretest require the operator to ask the defect elimination technician for assistance. The defect elimination station has a more extensive diagnostic test station and a known-good instrument, which assist the technicians in isolating failures to the component level. Once the failure cause is determined, the module is reworked, if possible,

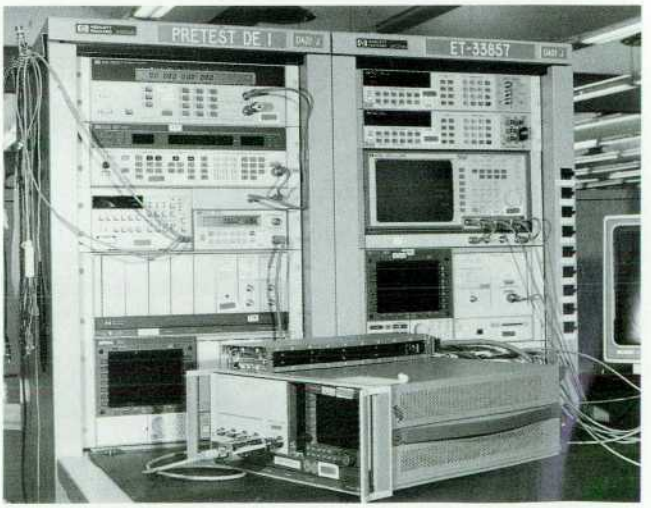


Fig. 5. Pretest/defect elimination rack.

and delivered to a buffer of known-good modules for future use in the assembly/pretest process.

Since in this scheme the assemblers are responsible for screening their own assemblies, the assemblers must be trained adequately to a relatively higher level of competence, that of assembler/pretester. They must be comfortable interacting with a video terminal and in handling live equipment. Additionally, there are some limitations on the pretests themselves. These are primarily related to speed and personnel skill set. First, the typical manufacturing process requires that board and module pretests should be short in duration for higher total throughput. Second, they should be high-value tests that do not require too much interpretation. Relatively high-skill-level operations such as adjusting potentiometers and interpreting oscilloscope traces generally do not lend themselves to this approach. This may exclude phase noise or switching speed tests, for example. Pretests also may not catch heat related problems or intermittent failures because of the relatively short periods of instrument operation.

Defect elimination troubleshooting tests are more detailed and extensive than pretests, and pretests in turn are a superset of the functional verification tests. Each board has a number of analog, digital, and RF test points available to the defect elimination technician for troubleshooting in addition to the outputs tested in pretest. The microprocessor board performs its own low-level self-test on power-up, checking power, RAMs, clock, and so on, and indicates test status by means of on-board LEDs, an especially useful feature.

Success Factors

It has been estimated that at least 70% of a product's life cycle cost is determined in the design stage. Hence the modern emphasis on design for manufacturability and design for testability as keys to competitive advantage.[2] The adoption of a test-as-you-build approach naturally imposes its own specific considerations (see "A Design for Manufacturability, Design for Testability Checklist," at right).

It should be noted that this approach benefits most from a project or systems engineer that acts as the system architect and oversees the partition of top-level (instrument-level) specifications into required traceable "hard" specifications at the module or printed circuit board assembly level. This activity will become even more important as firms attempt to leverage designs from product to product and consider make-or-buy and internal-or-external decisions. Early hard specifications under the control of a single chief engineer best allow for successful concurrent design, manufacturing, and test. Experience is unquestionably important in this activity.

Concurrent test development always involves hitting a moving target. This is particularly true in this scheme because specification and test scope changes below the final instrument level have a direct impact on the pretests. Board-level and module-level specifications may be inadequately derived or documented early in a program. There may be problem vendors. Special orders and options added later in program development will impact hardware and firmware designs.

For a discussion of design for testability and the importance of adequate partitioning see reference 3. A good argument for the general concepts of robust design and specification tolerances is presented in reference 4.

A Design for Manufacturability, Design for Testability Checklist

Some design for manufacturability and design for testability issues require special consideration when designing an instrument for the layered assembly and pretest technique described in the accompanying article. Here are a few examples based on our experience in working with this family of instruments:

- First, the most general concern: Is the instrument partitioned effectively, both functionally and physically, for layered assembly and pretest such that faults can be detected and isolated independently and unambiguously? For example, a single-board phase-locked loop is generally preferred over one distributed on several boards.
- If using a switching power supply, will the power supply start up with the minimum loading presented by the lowest testable instrument assembly? If not, then an interim load fixture will be required.
- Can the instrument be operated as necessary (e.g., via the HP-IB) if the front panel is not present? Some processes may find this desirable for assemblability or cosmetic reasons. Don't forget the on/off power switch.
- Are all shared utilities (digital buses, power, and analog buses) capable of operating with the instrument in all permutations of assembly? This is important not just for the standard assembly sequence, but for possible service, repair, and rework. Consider circuit board loading, failed component problems, possible signal sneak paths, cabling, and so on.
- Are test outputs designed, and the instrument partitioned, to be compatible with the typical 50-ohm input impedance of standard test equipment? Can oscilloscopes, analyzers, and other gear be set for high impedance if necessary (e.g., 1 megohm)? If not, test outputs should be adjusted for the correct impedance, polarity, coupling, power levels, and so on, or fixturing may be required.
- Has the impact of calibration procedures been included in the design of pretests and built-in test firmware code? For example, if the YIG oscillators are not calibrated, the YIG oscillator loop may not lock.
- If an assembler must operate flying-lead oscilloscope probes on the board, are there adjacent and obvious test points (both signal and ground)? As a matter of routine, assemblers should not have to probe components, attach ground clips to the chassis, or perform similar tasks.
- Does the instrument design allow for additional options without requiring access to previous subassemblies? For example, are microprocessor-board DIP switches readily accessible? In other words, consider the impact of likely future options on the instrument and its assembly and test early in the design. Don't forget the impact of options on the organization and design of the test software.
- Have all warmup-related items been eliminated or minimized? Assemblers must turn units off to work on them between tests. Therefore, tests sensitive to components or circuits requiring even ten minutes of warmup time are not good candidates for pretest.
- Does the functionality of a unit or assembly vary during the manufacturing process? For example, thermal and electromagnetic effects may be more variable with the covers off.
- Finally, since the instrument itself is the ultimate test bed, are one or more prototypes scheduled and budgeted in the project for concurrent test development?

Networked Computing and Test Environment

If pretests could accurately test all of the parameters that determine the performance of the instrument, no further testing would be necessary. However, in complex instruments like the HP 8370 and 70340 Series synthesizers, there are too many variables and board interactions to guarantee system operation based upon pretest results. Therefore, final test verifies that the complete unit complies with the warranted specifications.

For many years, the test strategy for manufacturing high-performance microwave frequency synthesizers has followed

a traditional methodology. Each instrument specification is tested. The tests are grouped by function or by use of common equipment in test stations. A test station contains several pieces of measuring equipment which are operated either manually or by a computer. Each computer and system is independent of the other computers and stations.

This approach, while conceptually simple, leads to several problems. Implicit in manual tests is operator intervention and sometimes interpretation of the data. Inconsistent results occur under these conditions. Even without operator intervention and interpretation, production line capacity needs can dictate multiple identical test stations. A typical production line might have ten or more different test stations, each operating independently. Problems can arise when it is necessary to update some of the test procedures or analyze data that is on several systems. There have been solutions in the past that have corrected some of these deficiencies. Older computer networking schemes like Hewlett-Packard's Shared Resource Management (SRM) software and hardware allowed computers running HP's BASIC/WS software to communicate on a rudimentary level and share files and printers. To analyze data, people wrote their own software to calculate means, standard deviations, and other statistical process control information. Until recently, there were insufficient products to allow enough control and integration of the test process to make it economically feasible to use it on a production line with a capacity on the order of 50 to 250 units per month.

In designing the test systems and the surrounding computer systems for the HP 8370 and 70340 Series synthesizers, we had several important goals. First, in keeping with our design team's goal of concurrent engineering, the test systems needed to aid the R&D engineers in the characterization of the product. Second, while aiding the R&D engineers was important, the systems needed to be easy to use for people with widely varying skill levels. Third, the test systems needed to be organized such that the test times at all of the stations were approximately equal for an efficient pipelining system. Fourth, all of the test systems needed to be integrated in a manner that allowed file and printer sharing, access to the data for analysis, and display of digitized images. Last, the cost for the integration of the process was to be a small fraction of the test equipment cost. A production system with these characteristics would yield an efficient and organized production line.

Given the above goals, there was one choice that seemed most appropriate. The production line for the HP 8370 and 70340 synthesizers uses a cluster of networked HP 9000 Series 300 computers running the HP-UX operating system. A diagram of the configuration is shown in Fig. 6. A central server stores the operating system and all of the production test software on a disk. Each test system has its own computer that boots from the server. As shown in the figure, at the physically remote location in the environmental test lab, another server acts as a repository for the operating system, while the test software is linked to the main production server. This allows one location for the storage of both tests and data. The main server also has access to the disk that the image capture PC uses to store the images. In this way, there is an easy transfer of the images from development to the production line.

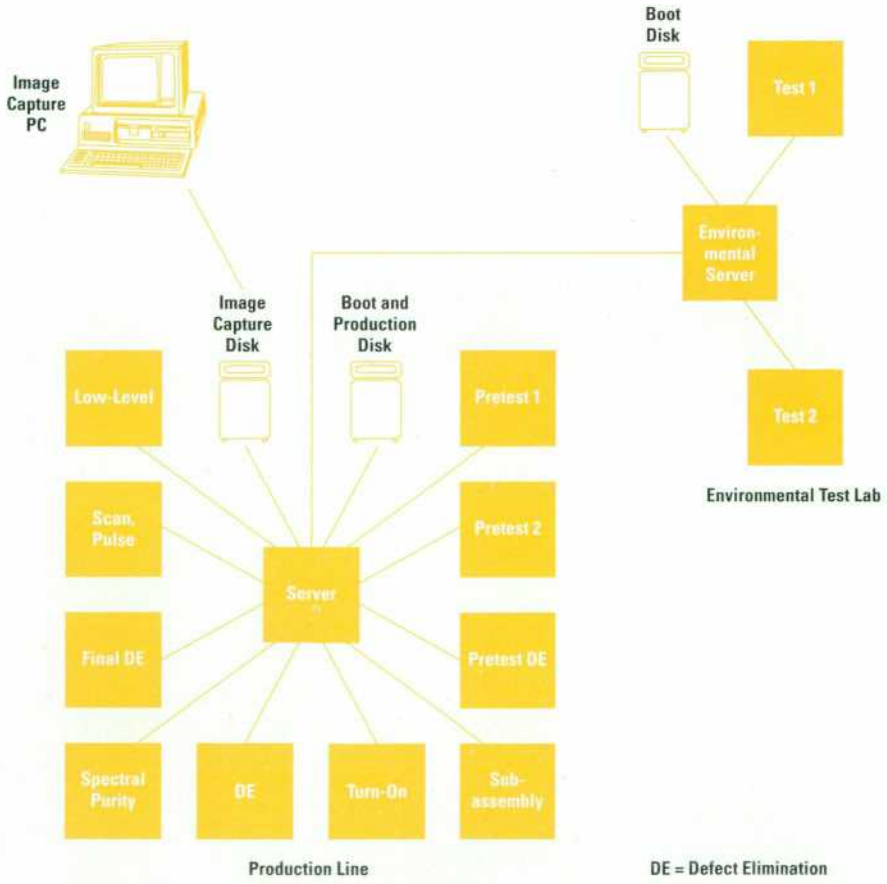


Fig. 6. Computer network configuration.

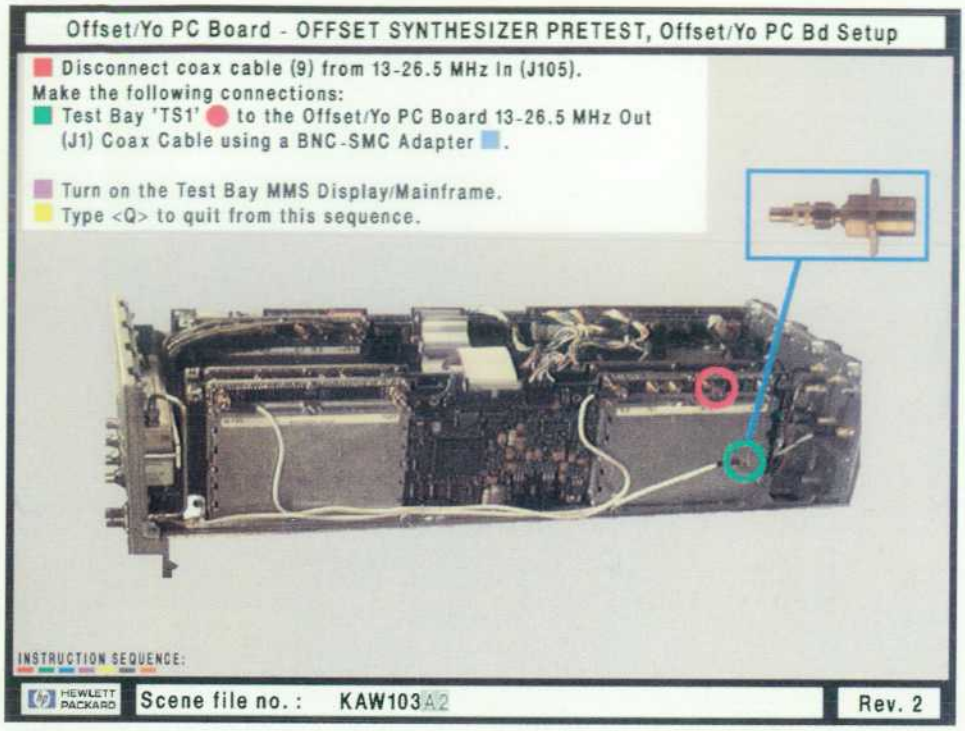


Fig. 7. Video image example from the offset synthesizer pretest.

The same test algorithms that were used for the development of the product are used on the manufacturing line. In many cases, this is not the ideal approach. A better way is to identify a small set of characteristics of the product that accurately represent its functionality, such that testing these parameters determines whether the product is constructed correctly. If the product design is originally well-characterized and proven, this simple functional test is sufficient to prove that it meets all specifications. In the case of the HP 8730 and 70340 Series synthesizers, too many parameters are not completely predictable or controllable. As a result, we test to the specifications directly. This leads to a large number of tests and a potentially complicated test system.

To combat this problem, several levels of software are used to achieve a simple user interface. Some people are afraid that the technicians and assemblers on a manufacturing line will not be able to use the HP-UX operating system. Using HP's implementation of the X Window System and running HP BASIC/UX, the interface is not much different from what the technicians and assemblers have experienced in the past in independent stations. The amount of effort required to implement this interface was minimal.

By setting a goal for the throughput of the line, it was possible to divide the test systems so that each system takes approximately the same amount of time to do its job. Using the clustered environment allows many other benefits. The need for a printer at each test station is obviated. As stated earlier, there is only one place for all test and data files, which eases modification, revision tracking, and data retrieval. Potentially, the HP-UX system allows much greater flexibility in the use of other tools, such as defect tracking systems, data analysis programs, and so on. Powerful statistical and mathematical software packages are available to analyze the data. Once familiar with the basics of the system, technicians have expanded their knowledge by using the multitasking capabilities of the computers to do several tasks at once.

The cost increase for this system over other solutions was minimal compared to the cost of the test equipment. The HP-UX-based system provides the greatest flexibility while sacrificing little of the original goals. Developing the software under HP-UX was a great help to the team because it allowed sharing the information and software at an early stage and an early trial of the use of the HP-UX system on the manufacturing line. Thus, the networked computing environment provides many advantages over the previous method and lays the foundation for continued process improvements.

Online Video Image Procedures

Online video image procedures for assembly and pretest provide quality documentation to the assemblers, help maintain the integrity of the procedures, and minimize paper drawings and documentation. The procedures combine text with graphics and are color-coded to correspond to the sequence of assembly and pretest (see Fig. 7). The procedures are presented to the assembler online by an HP 9000 Series 300 computer which also controls the test equipment for pretesting at each workstation. The video images are integrated into the test executive software environment, thereby providing a single user interface for both assembly and pretest operations. The online nature of the procedures ensures that all assemblers see the same version of the documentation. In addition, the procedures have proven to be a useful training aid and help to expedite the learning process for new assemblers.

The HP 9000 HP-UX platform was chosen because it meets the requirements for both online video image presentation and a well-managed distributed test strategy. Image files are transferred electronically from the image capture development platform (DOS-based) via the site LAN and a dedicated storage drive to the HP 9000 network server on the production line. The image files are received in TGA graphics format

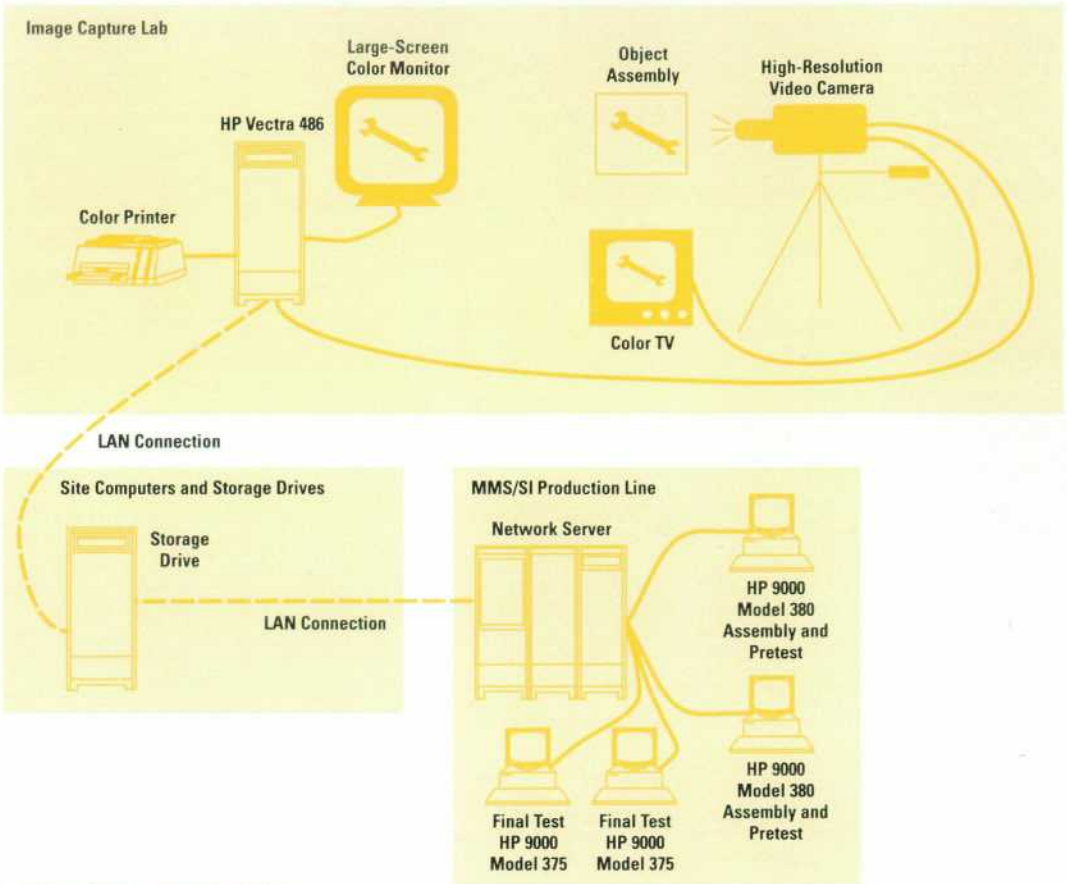


Fig. 8. Image capture lab hardware logistics.

from the development system. A conversion utility is used to transform the files to GIF graphics format for display on the HP-UX platform. The resulting GIF files are stored on the production network server. Since assembly and pretest steps are combined throughout the manufacturing process, display of all images is controlled through the menu-driven test executive that runs the pretests.

In establishing our video image capture process, we were able to leverage significantly some earlier work done at the HP Spokane Division. Our hardware platform and software tools mirror Spokane's implementation, with the addition of online image viewing.

The video image procedures are developed offline in an image capture lab. The lab is equipped with an HP Vectra 486 computer, a PostScript® color printer, a large-screen color monitor, a color television set, and a high-resolution video camera (see Fig. 8). Software tools running on the HP Vectra 486 PC are used in the process (see Fig. 9). Pictures of the assemblies are taken in the lab with the frame capture camera. A 4M-byte video graphics adapter board accepts the data from the camera and converts it to a TGA file. It is then necessary to correct the aspect ratio of the captured image for use in a paint program. A software tool that employs a linear interpolation technique is used to adjust the image to the required aspect ratio. The image is then called into the paint program where color modification, shading, and other techniques are used to touch up the picture. Once the pictures have been refined, they are accessed from an object-oriented development tool and combined with colored text and objects to form a scene, or procedure. The entire scene is rendered and saved by this software tool as a TGA graphics file (in effect, a snapshot of the scene). At this point, the TGA file is copied to the image capture storage drive for transfer to the HP 9000 server.

The image files that are referenced in the scenes are quite large (2 to 4 Mbytes each), and it takes many files to make a complete assembly and pretest procedure (the HP 70340A synthesizer takes approximately 70 scenes). Consequently, it is necessary to archive the files on some other storage mechanism. Again, we were able to use a software application and corresponding file-naming convention developed by the Spokane Division to archive the files in an organized manner. The software is a mouse-activated, menu-driven application that allows the user to specify scenes to archive or retrieve. The software automatically verifies the file name, compresses the file, and copies the file into the correct subdirectory in the archive database. The archive database resides on a dedicated storage drive and is accessed from the Vectra 486 via the site LAN.

The test executive gave us a vehicle for providing a consistent menu-driven user interface for both assembly and pretest procedures. Extensions to the test executive were developed to facilitate the display of graphics files. All images are stored in one image directory in the test executive's hierarchy. HP BASIC/UX programs were added for all assembly or pretest steps that invoke a utility for displaying the images. Image lookup tables provide a map for cross referencing model and option combinations to the correct sequence of images to be displayed. Assemblers simply enter the model, options, and serial number data at the test executive menu

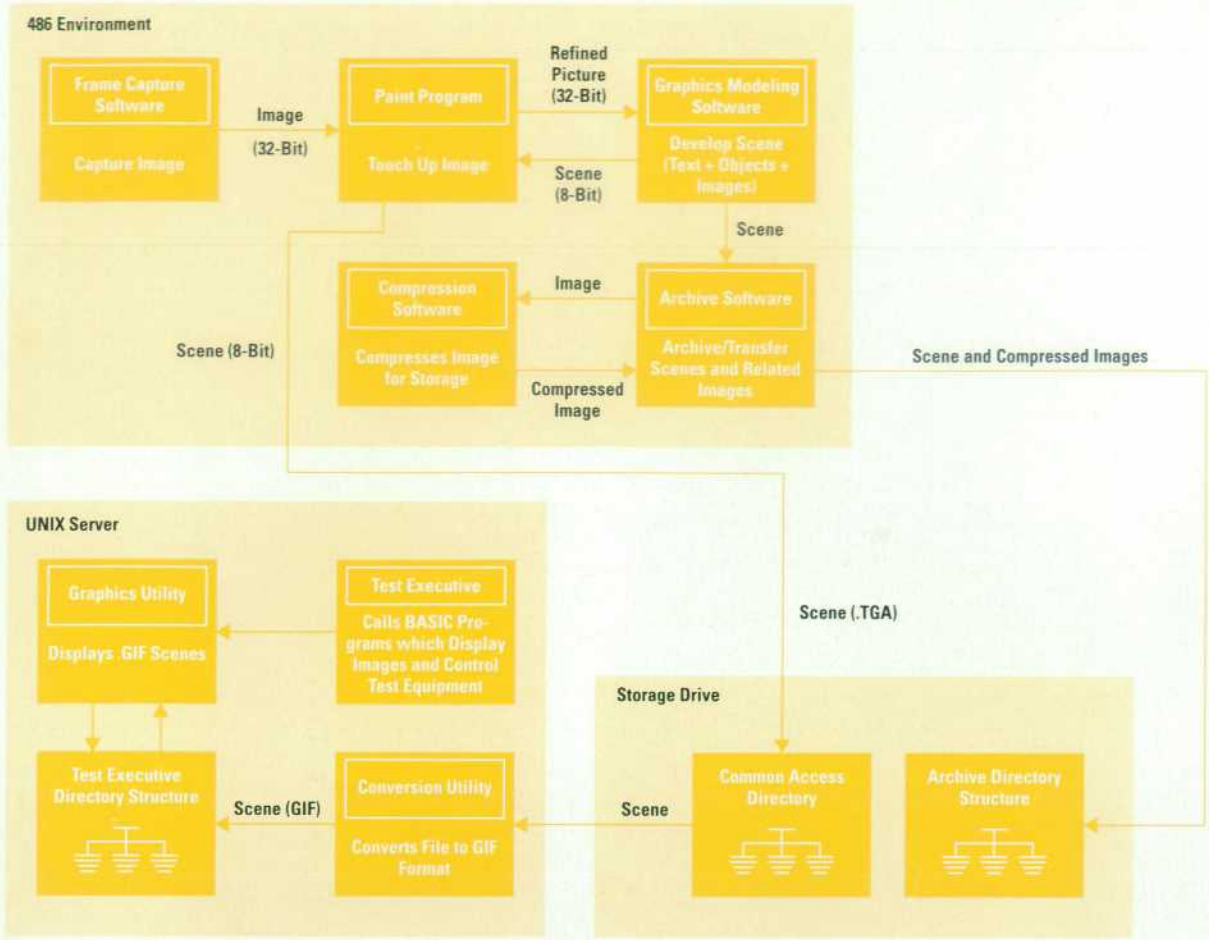


Fig. 9. Image capture lab software logistics.

prompts, and the correct sequence of procedures is displayed in the X Window environment. Hooks are available for entering data by means of a bar-code scanner from the serial label on the instrument's rear panel. The assembler advances from one image to the next for a particular assembly or pretest step by tapping the space bar. The backspace key is used to return to a previous image. In the case of a pretest step, instructions are displayed for the test setup and the test drivers are automatically invoked from the same procedure when the assembler advances past the setup.

Conclusions and Comments

The synthesizer manufacturing process continues to undergo incremental improvement. The integrated assembly and pretest process has already been successful in improving the quality of instruments presented to final test. Failures are being audited to see if the pretest strategy can be tightened reasonably to minimize such failures. Work remains to fine-tune the process and extend it to other instruments and options in the family.

The networked computing environment has proved to be as useful and flexible as forecast. The image capture procedures have been well-received by the assemblers and have already proved useful in shortening the learning period. Experience gained in this process should enable us to achieve greater flexibility and use the same line for extensions to the same basic instruments.

Acknowledgments

V.J. Bonnard organized and led the manufacturing new product introduction project team. Steve Cunningham contributed significantly to the product's design for manufacturability. Dave Milani was the original architect of final test. John Westerman at the HP Spokane Division established the image capture process that we were so successfully able to leverage. Thanks to Paul Arndt of the Spokane Division, who inherited the process from John and who provided us with invaluable technical support during our image capture implementation stage, to Jim Bertsch for his inputs to the design for manufacturability discussion in this article, and to Tim Chan for the photography for Figs. 1, 3, and 5.

References

1. M.J. Seibel, "Built-in Synthesized Sweeper Self-Test and Adjustments," *Hewlett-Packard Journal*, Vol. 42, no. 2, April 1991, pp. 17-23.
2. *Improving Engineering Design*, U.S. NRC, 1991, p.1.
3. J. Turino, *Design for Test*, Van Nostrand Reinhold, 1990.
4. G. Taguchi and D. Clausing, "Robust Quality," *Harvard Business Review*, January-February 1990.

A New Generation of Microwave Sweepers

The HP 83750 family of microwave sweepers achieves a new level of swept frequency accuracy by being fully synthesized in all sweep modes, including fast analog sweeps. It also uses fundamental oscillators for improved signal purity.

by Alan R. Bloom, Jason A. Chodora, and James R. Zellers

Swept-frequency microwave signal sources (sweepers), have long been considered basic equipment for component test applications. Sweeper capabilities also satisfy many general-purpose needs, both in the laboratory and in production.

The first HP microwave sweeper, the HP 670A, employed a klystron oscillator, which was mechanically swept by means of a motor drive. Sweep speed capability was improved considerably in the 1960s by the HP 8690 Series, which employed an electronically-tuned backward-wave oscillator (BWO). In the 1970s, the HP 8620 Series achieved new levels of reliability by replacing BWO tubes with solid-state YIG oscillators. In 1980, microprocessor control was added in the HP 8350 Series sweepers[1] to provide full programmability and many convenient user features.

The frequency accuracy of most sweepers, including those mentioned above, is limited by the microwave oscillator and its drive electronics. Newer synthesized sweepers, such as the HP 8360 Series,[2] achieve excellent frequency accuracy in CW and stepped-sweep modes. In continuous-sweep mode, however, they provide synthesizer correction only at the beginning and the end of each sweep band—there is no frequency correction during the actual sweep. The HP 83750 Series (Fig. 1) establishes a new standard of swept frequency accuracy by being fully synthesized in all sweep modes, including fast analog sweeps.

The HP 83750 Series of synthesized sweepers is part of the HP 8370 family of microwave sources. This family also includes CW generators and synthesized signal generators, which are discussed in the article on page 6. Four HP 83750 sweeper models are currently available:

- HP 83751A: 2 to 20 GHz, +10 dBm
- HP 83751B: 2 to 20 GHz, +17 dBm
- HP 83752A: 0.01 to 20 GHz, +10 dBm
- HP 83752B: 0.01 to 20 GHz, +16 dBm from 0.01 to 2 GHz, +17 dBm from 2 to 20 GHz.

Extended frequency coverage to 110 GHz is available with HP 83550 Series millimeter heads, using the optional source module interface. Other options include a 70-dB attenuator, a high-stability time base, and alternate output connector type and location. For field test applications, a portable package is also available, which adds a tilt-bail handle, rubber bumpers, rear feet, and a protective front-panel cover.

Fundamental Oscillator Technology

The RF block diagram of the HP 83752A sweeper appears in Fig. 2. A dual YIG oscillator (DYO) generates a microwave signal between 2 and 20 GHz using two separate oscillators mounted in a common magnet assembly (see article, page 46). One oscillator tunes from 2 to 11 GHz, and the other tunes from 11 to 20 GHz.

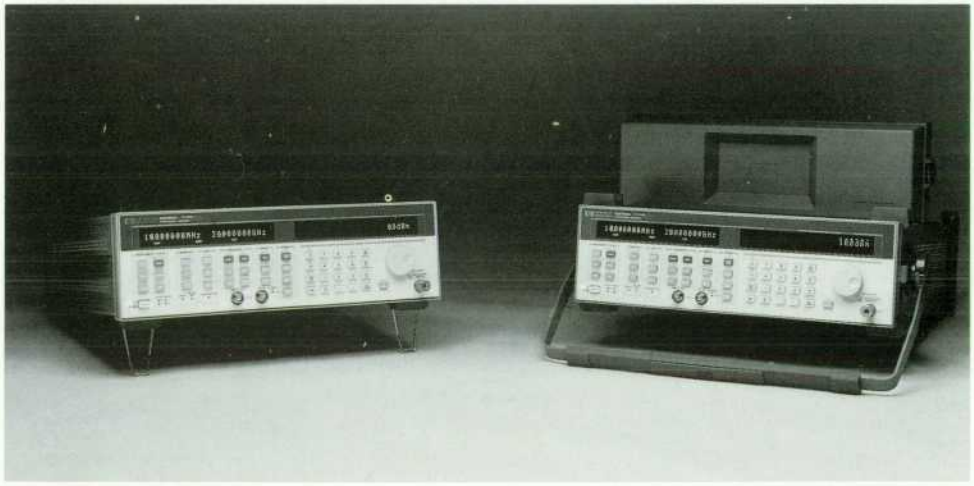


Fig. 1. The HP 83750 Series synthesized sweepers are packaged in a standard five-inch-high cabinet. They fit in the same rack panel space as the older HP 8350 Series nonsynthesized sweepers.

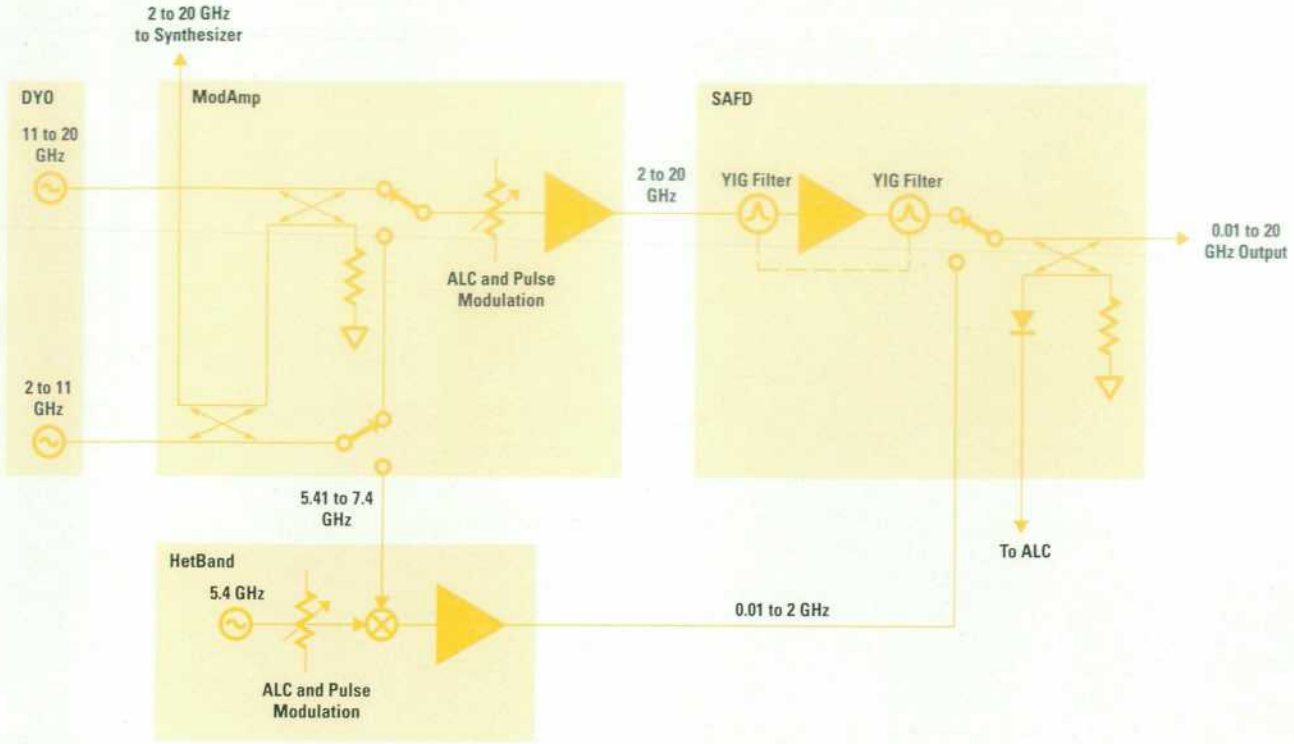


Fig. 2. RF block diagram of the HP 83750 Series sweepers. The RF chain includes four new microcircuit designs.

Previous HP sweepers used a lower-frequency oscillator multiplied up to the desired output frequency. The advantage of the fundamental oscillators in the HP 83750 is that they produce no subharmonics.

The two oscillator outputs feed into a modulator amplifier (ModAmp) microcircuit, which provides signal switching and distribution, amplification, and ALC and pulse modulation (see article, page 46). A low-level output goes to the sampler assembly for use by the synthesizer.

The main 2-to-20-GHz output signal from the ModAmp is routed to a switched amplifier filter detector (SAFD) microcircuit, which provides power amplification and two stages of YIG filtering (see article, page 46). The filtering reduces output harmonics to less than –45 dBc. A significant advantage of the YIG-filtered output is extremely low broadband noise, which is important for scalar network analysis applications.

The SAFD microcircuit also contains a broadband automatic level control (ALC) detector, which is used for power leveling over the entire 0.01-to-20-GHz frequency range. This technique avoids the power discontinuity commonly associated with sweepers that use separate leveling detectors for different frequency ranges.

To generate a signal below 2 GHz, the YIG oscillator is tuned between 5.41 and 7.4 GHz. This signal mixes with the output of a fixed 5.4-GHz synthesized oscillator to produce the 0.01-to-2-GHz output. An amplifier and filter follow. A major objective of this design was to reduce nonharmonic mixing spurious signals to less than –50 dBc for output levels less than +5 dBm, while minimizing broadband noise.

Frequency Control

The microwave oscillator and filter in the HP 83750 each contain a tiny sphere of crystalline yttrium iron garnet (YIG). When subjected to a magnetic field, a YIG sphere resonates at a microwave frequency that is directly proportional to the strength of the magnetic field. The sphere is mounted in the pole gap of an electromagnet. The field strength, and thus the YIG resonant frequency, are directly proportional to magnet current.

The job of the YIG driver, Fig. 3, is to generate a current proportional to the desired frequency. A sawtooth-shaped signal produced by the sweep generator board is scaled and offset by the YIG driver to sweep the YIG oscillator and filter over the desired frequency range. Smaller correction signals from the sweep generator compensate for nonlinearities and delays in the YIG frequency-versus-current response.

Even if the phase-locked loop synthesizer were not present, this architecture would produce a swept microwave signal with fair frequency accuracy. The synthesizer need only apply a small correction voltage to eliminate any remaining errors.

The function of the sweep generator board (Fig. 4) is to generate the main sweep ramp and several correction signals. Previous instruments used an integrator to form the sweep ramp. Such an integrator must be carefully designed to prevent its drift and nonlinearity from contributing to swept frequency error. In the HP 83750, a digital-to-analog converter (DAC) generates a stepped sweep ramp simultaneously with the analog ramp. Once per step, the two ramps are compared, and the sampled error voltage feeds back to correct

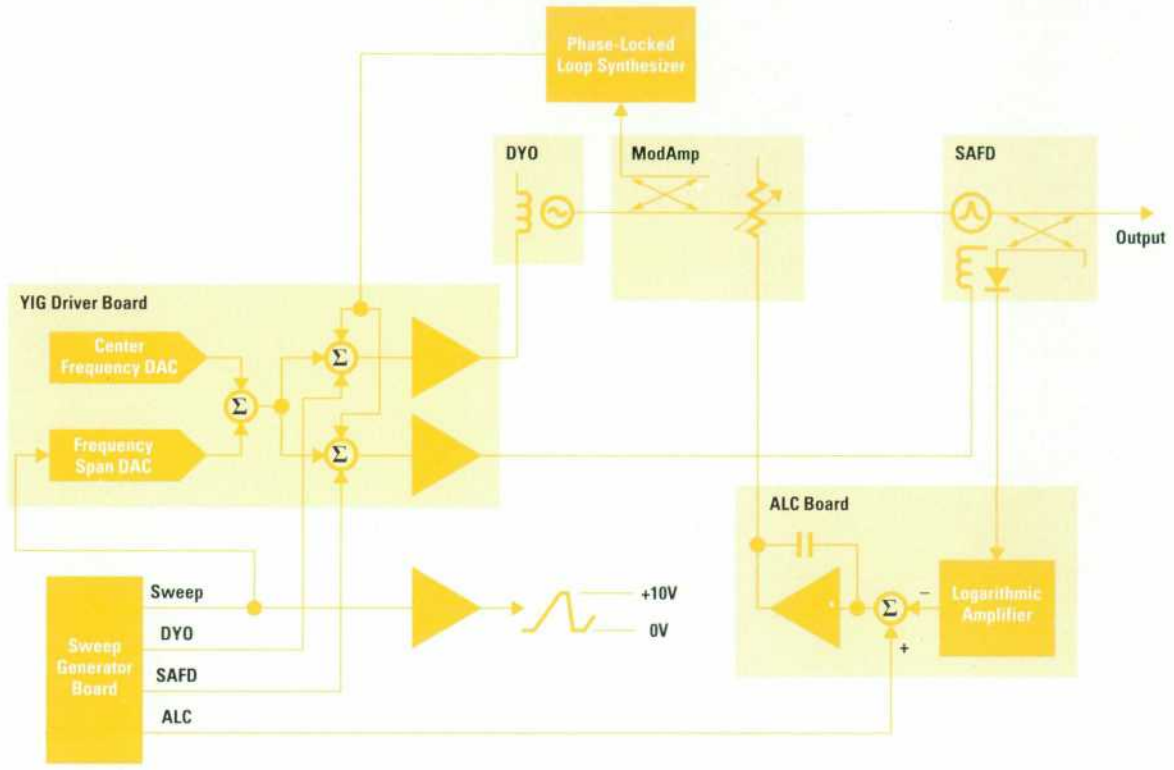


Fig. 3. HP 83750 frequency control block diagram. Unlike previous designs, the synthesizer in the HP 83750 remains locked throughout the sweep.

any integrator errors. The resulting sweep ramp reflects the accuracy and stability of the DAC output while retaining true analog sweep.

The analog sweep ramp is available to the user as a 0-to-10V ramp (**Sweep Out**) whose amplitude is independent of frequency span. Another output voltage (**V/GHz**) proportional to absolute frequency is also available. At the user's option, it can be scaled and offset to sweep between any two voltages between –10 and +10 volts.

The Digital Signal Processor

The digital sweep ramp and four correction voltages are generated by five DACs. A TMS 320C10 digital signal processor (DSP) calculates appropriate values in real time and loads

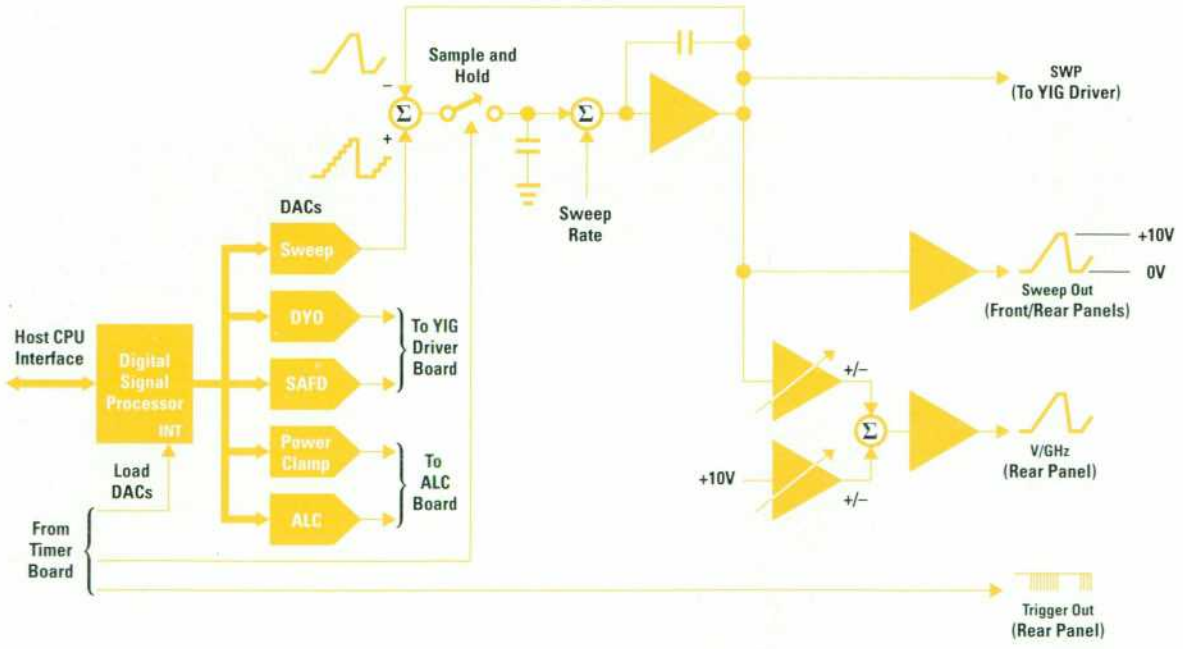


Fig. 4. The sweep generator assembly produces an analog ramp and four correction signals for use by other assemblies.

them into the DACs as the instrument sweeps. When the instrument is turned on, the host microprocessor (an MC68000) downloads calibration data to the DSP. After that, the host need only update the DSP with new start and stop frequencies, power, and sweep time whenever the user changes those values. The DSP automatically calculates the correction voltages based on that data.

The DACs are updated between 101 and 1601 times per sweep, depending on sweep time. A circuit on the timer board generates 101 to 1601 trigger pulses per sweep that are applied to the DSP interrupt input. The interrupt causes the DSP to update all five DAC outputs simultaneously.

The DSP has a minimum of about 100 microseconds between pulses to calculate upcoming DAC values. In this time, it must compute three third-order interpolations, one straight-line interpolation, two sweep ramps, and two complex delay compensation waveforms. A special third-order curve-fit algorithm is the key to achieving this performance using inexpensive DSP hardware (see below).

Two of the third-order interpolations are used to calculate linearity corrections for the DYO and SAFD. Actually, the synthesizer alone could correct for YIG oscillator nonlinearity, but there is no guarantee that the oscillator and filter are identical in this respect. The DSP correction ensures that the oscillator and filter track each other throughout the sweep.

Even more critical, for both tracking and swept frequency accuracy, is delay compensation. Whenever the current through a YIG magnet is changing, the YIG resonant frequency lags the current. The time lag is a function of sweep speed, start frequency, and previous history of magnet current. This delay causes fast transients to occur at the start of each band that are difficult for the synthesizer circuitry to correct for. The DSP uses a proprietary algorithm to calculate delay compensation, which is added to linearity correction and output to the appropriate DAC. Using the DSP to perform this function replaced a boardful of sensitive analog circuitry. It also made it easy to try many different compensation waveforms in our quest to achieve a new standard of swept frequency accuracy.

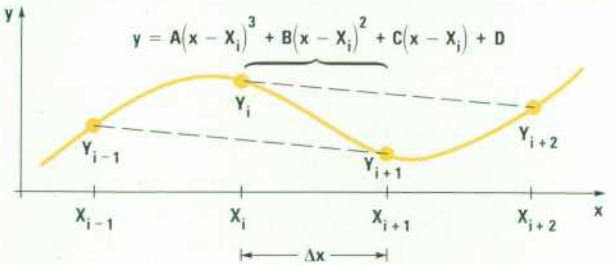
The third use of the third-order interpolation algorithm is for the power clamp output. The ALC board uses this signal to limit maximum RF drive level to the SAFD. Overdriving the YIG filter can cause squegging (low-frequency amplitude oscillations), which lowers the available output power.

The ALC output controls the instrument RF power level. The ALC board compares this signal to a logged version of the

Third-Order Curve-Fit Algorithm

The frequency-versus-current transfer function of a YIG oscillator or filter is only approximately linear. Production testing measures the deviation from linearity at a series of calibration frequencies and stores this information in the instrument's nonvolatile memory.

In Fig. 1, the Ys indicate the required correction at each calibration point. The Xs indicate the nominal frequency of each point. To interpolate between calibration points, we use a third-order curve y(x), defined by four parameters A, B, C, and D. A different set of parameters must be calculated for the interval between each pair of calibration points.



$$\left.\begin{aligned} y(X_i) &= Y_i \\ y(X_{i+1}) &= Y_{i+1} \\ \frac{d}{dx}y(X_i) &= \frac{Y_{i+1} - Y_{i-1}}{2\,(\Delta x)} \\ \frac{d}{dx}y(X_{i+1}) &= \frac{Y_{i+2} - Y_i}{2\,(\Delta x)} \end{aligned}\right\} \longrightarrow \left\{\begin{aligned} D &= Y_i \\ C &= (Y_{i+1} - Y_{i-1})/2 \\ A &= C + 2D + (Y_{i+2} - Y_i)/2 - 2(Y_{i+1}) \\ B &= -A - C - D + Y_{i+1} \\ &(\Delta x \text{ normalized to unity}) \end{aligned}\right.$$

Fig. 1. This third-order algorithm produces an interpolated curve that passes exactly through each calibration point and has a continuous first derivative. Only add and shift operations are required to compute the coefficients A, B, C, and D.

The most straightforward way to calculate the four parameters is to generate four equations by setting y(x) equal to the four closest calibration points, and then solve for the four unknowns A, B, C, and D. Besides being computationally intensive, this method results in sharp edges at calibration points, that is, it has a noncontinuous first derivative.

A well-known method called the cubic spline uses all N available calibration points to generate a curve with continuous Nth-order derivatives. Unfortunately, the calculations would take much longer than the available 100 μs. Other interpolation algorithms, such as the B spline, are faster, but the resulting curve does not pass exactly through the calibration points.

The method employed in the HP 83750 Series sweepers uses four equations to solve for the four unknowns. Two equations result from constraining the curve to pass exactly through the two nearest calibration points, Y_i and Y_{i+1}. The other two equations result from setting the slope of the curve at the segment endpoints equal to the slope of a line connecting the two points that bracket each endpoint. This guarantees that the slope of each segment will match the slope of the adjacent segment where the segments connect, so there will be no sharp edges.

Solving these four equations results in the equations for A, B, C, and D shown in Fig. 1. Note that all multiplications and divisions are by factors of two. This makes the computation very efficient since multiplying or dividing by 2 is equivalent to shifting the binary number left or right one bit. Shifting can occur automatically as the DSP loads the number into its accumulator.

Calculating A, B, C, and D requires 33 DSP cycles or 6.6 μs. The total time to do all computations required for the three separate interpolated outputs is 153 cycles or 30.6 μs.

Alan Bloom
Development Engineer
Microwave Instruments Division

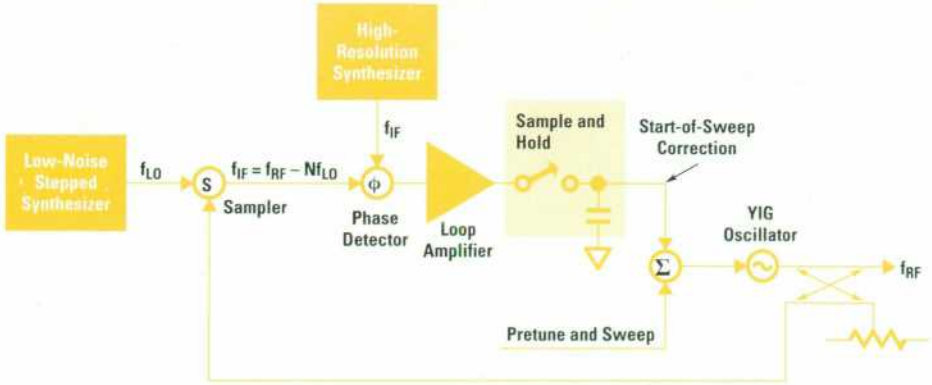


Fig. 5. Traditional synthesizer block diagram. In the traditional "lock and roll" synthesizer, the YIG oscillator is phase locked at the start of the sweep to a multiple of a low-noise, stepped CW synthesizer frequency. The analog correction voltage from the phase detector is sampled and held for the duration of the sweep.

detected RF output level and adjusts modulator current to make the two signals equal. Detector calibration and temperature compensation are handled by additional circuitry on the ALC board. The DSP ALC signal includes basic RF power level, flatness compensation, and power sweep.

"Flatness compensation" refers to the use of factory calibration arrays to correct for the frequency response of the coupler detector located in the SAFD. In addition, the user can enter arrays of up to 801 points to correct for the response of the user's test system. These user calibration points can be at any frequencies within the instrument's range. Since the third-order algorithm requires equally spaced calibration points, ALC flatness compensation must use linear interpolation.

Synthesizer

Hewlett-Packard microwave synthesizers (for example the HP 8360 Series) have traditionally used the block diagram shown in Fig. 5. The local oscillator (LO) drive for the sampler is a low-noise stepped synthesizer. Its output has coarse frequency resolution, but extremely low phase noise. It is not able to sweep its output frequency. The IF output of the sampler is then locked to a high-resolution synthesizer, often a fractional-N loop, which provides the output frequency resolution.

The only loop in the traditional block diagram with swept capability is the fractional-N loop. It is used as the reference for the sampler IF phase-locked loop. Therefore, its output is directly translated (mixed) to the RF output. This limits the width of a synthesized sweep to the width of a fractional-N sweep, which is typically tens of megahertz. For broader sweeps, the traditional synthesizer must sweep unlocked. The start frequency is phase locked, then the phase-locked loop error voltage is sampled and held while the frequency sweeps open-loop. This technique is called "lock and roll." Later instruments use the synthesizer to count the stop frequency and apply a correction to subsequent sweeps. This makes both the start and stop frequencies quite accurate, but the actual sweep is still performed open-loop with significant frequency errors between the endpoints.

In the HP 83750 (Fig. 6), the fractional-N loop is used as the LO for the sampler. The fractional-N loop output frequency is multiplied by the harmonic number and translated to the RF output. This gives the HP 83750 the ability to perform true synthesized broadband analog sweeps. If the fractional-N loop sweeps an octave, the RF output will sweep an octave. For example, the HP 83750 can sweep the full 11-to-20-GHz RF band in one continuous phase-locked sweep. This improves the swept frequency accuracy of the instrument by at least an order of magnitude (Fig. 7). Swept frequency errors are now limited to timing uncertainties and transients that cannot be completely removed because of limited phase-locked loop bandwidth. Both of these errors improve linearly with reduced sweep speed and span. This architecture gives the HP 83750 Series state-of-the-art swept frequency accuracy, allowing very precise, high-speed swept measurements.

The HP 83750 fractional-N assembly contains a voltage-controlled oscillator (VCO), a fractional divider, and a phase-locked loop. This circuitry can synthesize a 250-to-500-MHz signal with resolution better than 10 nHz and excellent swept frequency accuracy and phase noise (see page 44). The fractional-N phase-locked loop output is the LO drive for the microwave sampler. A coupler sends a portion of the YIG oscillator output signal to the RF port of the sampler. The action of the sampler in the frequency domain is similar to that of a harmonic mixer. The IF output frequency f_{IF} of the sampler is a function of the LO and RF frequencies f_{LO} and f_{RF} and the harmonic number N:

$$f_{IF} = f_{RF} - Nf_{LO}.$$

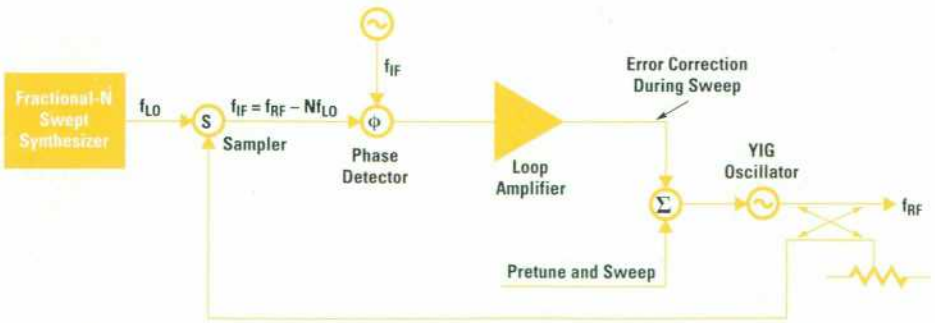


Fig. 6. HP 83750 synthesizer block diagram. In the HP 83750 synthesizer, the YIG oscillator is phase locked to a multiple of a fractional-N synthesizer frequency during the sweep.

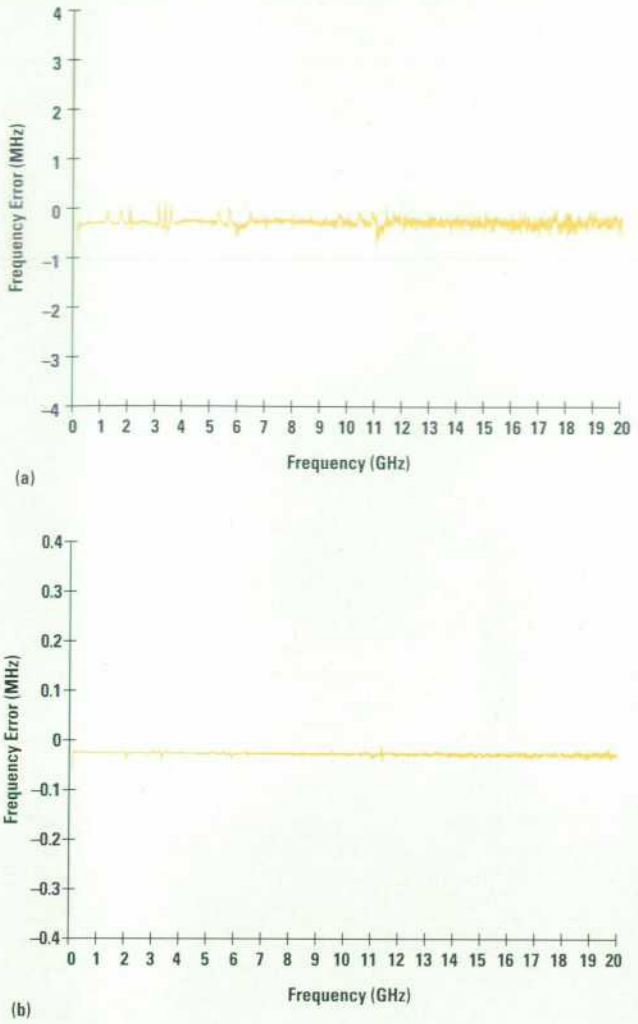


Fig. 7. (a) HP 83750 swept frequency accuracy with a 100-ms sweep. (b) Improvement obtained at a slower sweep speed, in this case a sweep time of 1 s.

The YO loop assembly is another phase-locked loop. It amplifies and limits the IF output of the sampler, divides it by 10, and then corrects the YIG oscillator frequency to phase lock the IF/10 signal to a 1-MHz reference. This constrains the IF output of the sampler to be 10 MHz. The output frequency of the YIG oscillator (f_{RF}) is then:

$$f_{RF} = Nf_{LO} + 10\ \text{MHz}.$$

The harmonic number N ranges from 7 to 41. This yields a range of 2 to 20 GHz for the synthesized YIG oscillator output frequency.

This synthesis structure places stringent performance requirements on the fractional-N assembly. The frequency accuracy, resolution, and phase noise of the fractional-N loop are all multiplied by the harmonic number. Analog techniques improve the swept frequency accuracy of the phase-locked loop enough to achieve our performance goals. Excellent frequency resolution is easily attained by using 48-bit accumulators in the fractional divider. The phase noise, however, is an inherent limitation of the HP 83750 architecture because fractional-N synthesizers have higher noise than stepped synthesizers. This causes more close-in phase noise than that exhibited by traditional HP synthesizers. The fractional-N phase-locked loop bandwidth and filtering are optimized for CW phase noise. The resulting performance is better than the nonsynthesized HP 8350/83592 at almost all offsets, and the residual FM is considerably improved. This performance should easily meet the requirements of most sweeper applications.

The HP 83750 synthesizer was designed for excellent swept frequency accuracy at low cost. A fixed frequency reference replaces the high-performance stepped synthesizer used in the traditional block diagram. The digitally corrected fractional-N synthesizer is much less expensive than the fractional-N synthesizers used in previous microwave instruments. This new architecture provides synthesized frequency accuracy and broadband phase-locked sweeps at a price usually associated with open-loop sweepers.

Self-Test

The HP 83750 includes an analog bus for self-test and calibration. The analog bus is a single wire that connects to most printed circuit assemblies. Each assembly includes an analog switch that can connect the bus to any of several test points. By properly setting these switches, firmware self-test routines can measure voltages throughout the instrument using a single 12-bit analog-to-digital converter (ADC) located on the CPU board.

Over 150 tests can be executed with a single key press. When a hardware problem occurs, it can have far-reaching effects, causing a handful of test routines to report failures. For example, an incorrect power supply voltage might cause all circuits that use that voltage to "fail." Finding the most independent failure and reporting that to the operator is the goal of the diagnostic feature.

For each test in the instrument, a list of its dependencies was created. For example, the list below suggests that the test of the **Sweep Out** signal requires that the digital sweep DAC, timer #2, the timer interrupt, the DAC trigger, and the sweep trigger tests all work properly. If any of these tests has failed, then it is inappropriate to suspect the **Sweep Out** circuits as the primary cause of failure.

Test	Dependencies
SWPOUT	DIGSWPDAC
	TIMER2
	LINT_TIMER
	LDAC_TRIG
	LSWP_TRIG

After building a comprehensive table of dependencies we used a computerized algorithm to sort all tests into a single list sorted in order of interdependence. Instrument firmware uses this list to report the primary failure—the one the service technician should investigate.

Test limits can be changed in the field to accommodate hardware upgrades. A test patch feature, similar to the one used in the HP 8360 Series, allows the customer service organization to alter test limits and store them into nonvolatile memory. (The original default limits are still safely retained as well.) Self-test algorithms look for test patch limits first before using the default limits. Both default and test patch limits, as well as measured test data, can be read via the HP-IB connector (IEEE 488, IEC 625) on the rear panel.

A Digitally Corrected Fractional-N Synthesizer

Fractional dividers are used to achieve arbitrarily fine frequency resolution in a phase-locked loop synthesizer. Normally, frequency dividers can only produce integer divide ratios. Fractional division is accomplished by alternating the instantaneous divide number between N and N+1. Accumulators control the number of cycles each divide number is used. The fractional divide number is the time average of the instantaneous divide number. If a fractional divider is used in a phase-locked loop, the output frequency (f_{out}) is:

$$f_{out} = (N.F)f_{ref},$$

where N.F is the fractional divide number and f_{ref} is the phase-locked loop reference frequency.

The phase-locked loop is in fact attempting to hop the VCO frequency between Nf_{ref} and $(N+1)f_{ref}$. This causes considerable phase modulation on the VCO. Most HP fractional-N synthesizers (Fig. 1) have used a technique called analog phase interpolation (API) to remove this phase modulation.[1] API uses the accumulators that control the instantaneous divide number to predict the phase error resulting from the fractional division. A DAC sums a canceling signal into the output of the phase detector. This analog correction must be incredibly precise to achieve the necessary spurious performance.

The HP 83750 fractional-N synthesizer (Fig. 2) uses a digitally-corrected fractional divider developed at HP's Spokane Division.[2] The divider uses the same concept as sigma-delta analog-to-digital converters (ADCs). These converters operate by greatly oversampling the analog input with a coarse (often one-bit) ADC. This output is then digitally filtered to eliminate out-of-band quantization noise. Sigma-delta modulator techniques[2] can then be applied to shape the quantization noise so that most of the noise is pushed outside the frequency band of interest. The quantization noise is removed by filtering.

In the fractional divider, the fractional divide number is analogous to the analog input to the interpolative ADC. The integer divider is analogous to the one-bit ADC.

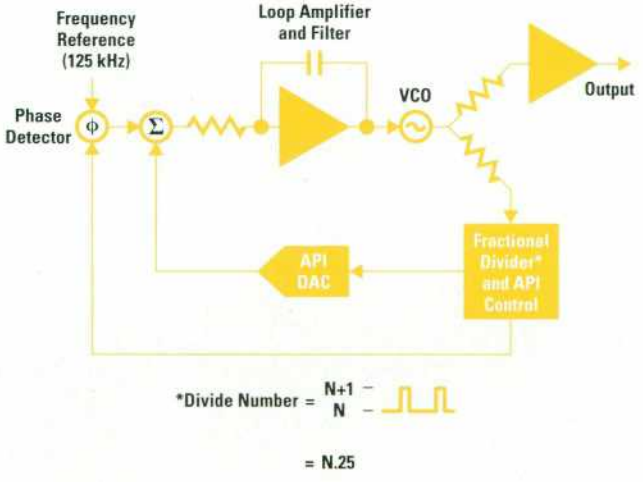


Fig. 1. The traditional fractional-N synthesizer uses an analog phase interpolator (API) to eliminate unwanted phase modulation of the VCO. The API needs 0.03% accuracy to reduce spurious sidebands to –70 dBc.

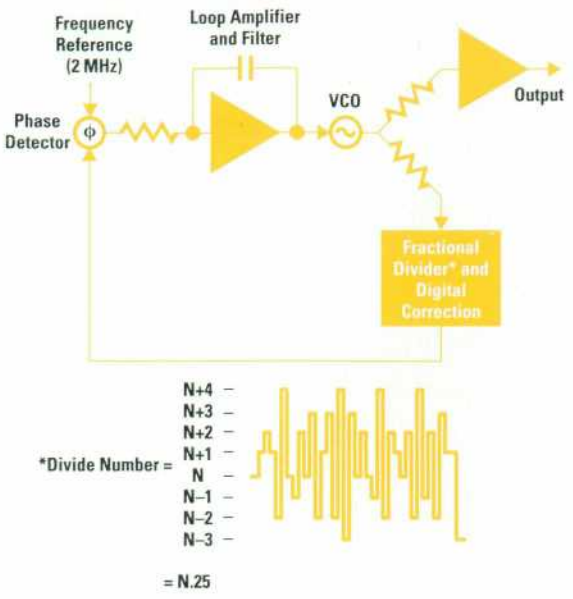


Fig. 2. The HP 83750 fractional-N synthesizer uses digital correction to shape the quantization noise. The phase-locked loop filters and removes the noise.

Digital techniques shape the fractional-division noise and push it well outside the bandwidth of the phase-locked loop. The phase-locked loop acts as a low-pass filter and removes the fractional division noise before it is applied to the VCO.

The digital correction changes the integer divide number every reference cycle, or every 500 ns in the HP 83750. The divide number is not toggled between N and N+1, but can take on any integer value between N–3 and N+4. The digital correction causes the divide number to vary in a random fashion, producing pure noise with no spurious content. This noise is shaped so that it increases at a rate of 40 dB per decade of offset frequency. The phase-locked loop low-pass filters this noise so that it never rises above the phase noise of the VCO. In this manner, the phase errors produced by fractional division are removed without the cost, size, and complexity of the API circuitry traditionally used in fractional-N synthesis.

Acknowledgments

The research and development of the fractional-N and prescaler ICs was done by Brian Miller and Bob Conley of HP's Spokane Division. They also provided consultation and support to the many users of these parts within HP.

References

1. D.D. Danielson and S.E. Froseth, "A Synthesized Signal Source with Function Generator Capabilities," *Hewlett-Packard Journal*, Vol. 30, no. 1, January 1979, pp. 18-26.
2. B. Miller and B. Conley, "A Multiple Modulator Fractional Divider," *Proceedings of the 44th Annual Symposium on Frequency Control*, 1990, pp. 559-568.

Jason Chodora
Development Engineer
Microwave Instruments Division

A significant new feature of the HP 83750 is that instrument firmware is stored in flash EPROM, a type of reprogrammable nonvolatile memory. This allows instrument firmware to be upgraded in the field to add new features or to fix bugs. Firmware upgrades are distributed on 3.5-inch disks, which can be read using an HP 9122D HP-IB disc drive. A smaller nonreprogrammable boot ROM contains the firmware loading routine and some basic self-test routines that execute whenever the instrument is turned on.

Self-Calibration

The HP 83750 uses large arrays of calibration constants to correct for YIG filter linearity, YIG oscillator linearity, and power flatness. The instrument can generate these arrays automatically. The only other equipment required is a power meter for power calibration.

A front-panel **Peak** button causes the instrument to align the YIG filter automatically at each of approximately 230

frequencies between 2 and 20 GHz. At each frequency, the analog bus is used to monitor ALC modulator drive voltage as the YIG filter frequency is varied. When a peak is found, the proper correction value is stored in memory. This function is also called *autotracking*.

YIG oscillator calibration is available via a Special Function menu. In this case, the analog bus monitors the YO loop correction voltage and adjusts the calibration constants until the voltage is zero.

A front-panel **Flatness Cal** button allows automatic power level calibration using an HP 437B, 438A, or 70100A power meter. The user can obtain calibrated power at any point in a test system by calibrating with the power meter located at that point.

Placing time-consuming calibration routines in firmware lowers instrument cost by reducing test time. For example, autotracking via an external controller running a BASIC program took 50 minutes. The firmware autotracking routine runs in less than one minute. Faster calibration also gives better accuracy, because there is less frequency drift between the beginning and the end of the procedure.

Product Design

Ruggedness, light weight and serviceability were the goals of the HP 83750 product design. Extensive shock and vibration testing was done to ensure that the instrument meets or exceeds HP standards for ruggedness. An optional portable package with tilt-bail handle is available that meets the requirements of MIL-T-28800E Type III Class 5 Style D.

The weight was reduced through the application of modern technology. A switching power supply eliminates the heavy power transformers found in earlier designs. The dual YIG oscillator combines two separate microwave oscillators within one magnet structure. Other sweepers that use fundamental oscillators require two or three devices to cover the same frequency range, which adds to cost, weight and power consumption. An HP 83752A sweeper weighs 16 kg (35 lb) compared with 22.5 kg (49.6 lb) for an HP 8350B with HP 83592A plug-in.

Servicing an HP 83750 is facilitated by ease of access (see Fig. 8). A central cardcage houses most of the printed circuit boards. Test points are located at the top of each board. A board can also be raised on an optional extender board for more extensive troubleshooting. The power supply, bolted to the left side of the chassis, is replaceable as a unit. All microcircuits are located on an RF deck at the right side of the chassis and can be accessed by removing the top and side instrument covers. The entire RF deck can be removed from the instrument without disrupting any RF connections.

Fig. 8. Internal view of the HP 83750 chassis.

Acknowledgments

The authors would like to thank the many other engineers who made valuable technical contributions to this product. Stewart Chaimson designed the YIG driver and RF interface electronics, and developed the algorithms and data structures for YIG oscillator and filter delay compensation and linearity. Lance Haag developed calibration and alignment procedures, evaluated RF system performance, and developed the sampler assembly. Steve Punte was the technical leader and Doug Bender provided management support for firmware development. Other members of the firmware team included Jim Grishaw (hardware and HP 8757 control), Thanh Heyman (user interface), Mike Seibel (SCPI coupling and flatness calibration), Sue Wood (self-test), and Phuoc Tran and Dan Podell (firmware QA). Roger Valentine and Andy Smith did the product design. Jon Jasper was responsible for new product introduction production engineering and test development. Jon Kiser coordinated the environmental testing. We would also like to thank Arlen Dethlefsen and Rolf Dalichow for their support.

References

1. R. Dalichow and D.E. Fullmer, "A Broadband, Fully Programmable Microwave Sweep Oscillator," *Hewlett-Packard Journal*, Vol. 33, no. 2, February 1982, pp. 3-10.
2. R.P. Oblad, J.R. Regazzi, and J.E. Bossaller, "A Family of High-Performance Synthesized Sweepers," *Hewlett-Packard Journal*, Vol. 42, no. 2, April 1991, pp. 6-16.

Microcircuits for the HP 83750 Series Sweepers

Four custom microcircuits provide the basic output signal, the RF band, signal switching and distribution, amplification, ALC and pulse modulation, power amplification, and two stages of YIG filtering.

by Eric V.V. Heyman, Rick R. James, and Roger R. Graeber

This article discusses the design of the four custom microcircuits designed for the HP 83750 Series sweep oscillators. The microcircuits are:

- The dual YIG oscillator (DYO)
- The switched amplifier filter detector (SAFD)
- The 0.01-to-2-GHz heterodyne band microcircuit (HetBand)
- The combiner modulator amplifier (ModAmp).

Dual YIG Oscillator

The signal for the HP 83750 Series sweepers is generated in the dual YIG oscillator microcircuit. The DYO is actually two YIG oscillators in one magnetic housing. One oscillator covers the span from 2 to 11 GHz and the other covers from 11 to 20 GHz. The output power exceeds 20 mW from separate outputs for each band.

The high-band 11-to-20-GHz oscillator consists of a YIG resonator and a single GaAs IC chip that contains both the oscillator and buffer stages. Fig. 1 is the schematic diagram. The IC is fabricated using an HEMT GaAs IC process with an f_T of 50 GHz and an f_{max} of 100 GHz. The chip (Fig. 2) measures only 960 by 960 μm.

The oscillator stage consists of a 200-μm FET in a source follower configuration. The feedback is generated by a 0.2-pF thin-film capacitor connected between the source and ground. This feedback generates an impedance looking into the gate of the device that has a negative real part and thus has a reflection coefficient greater than 1, which is a necessary condition for oscillation to begin.[1] The condition for oscillation to begin is:

$$\Gamma_{device}\Gamma_{resonator} > 1,$$

where Γ_{device} and $\Gamma_{resonator}$ are the reflection coefficients of the device and resonator, respectively. The condition at oscillation is:

$$\Gamma_{device}\Gamma_{resonator} = 1.$$

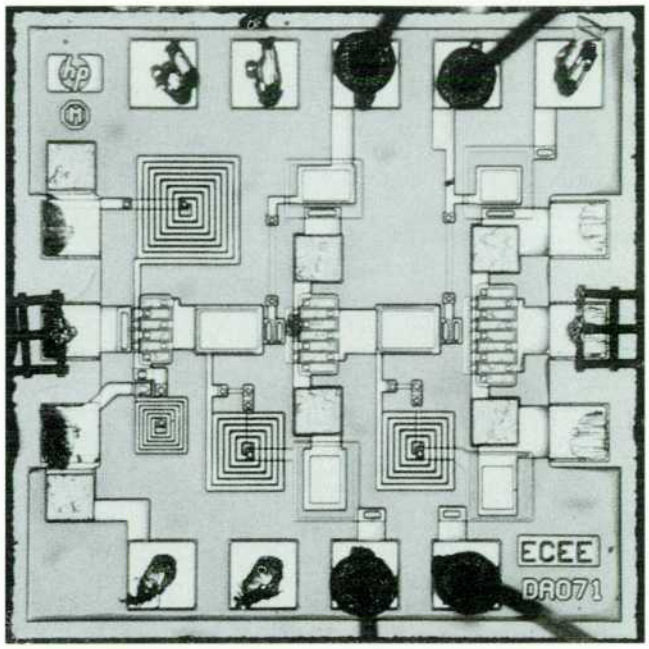


Fig. 2. Photograph of the DYO chip.

This relationship is achieved as Γ_{device} is reduced by limiting during the buildup of oscillation. The phase condition is satisfied by a shift along the resonator curve, possibly allowing the oscillation to occur somewhat off resonance.

The source follower configuration has the potential to oscillate at undesired frequencies above or below the desired band. These undesired oscillation conditions, called lockup modes, are a result of the interaction of the YIG coupling loop parasitics and the active device. The oscillator circuit must be designed so that there is insufficient reflection gain to support the lockup mode. At the low end of the band this is accomplished by placing an inductor in parallel with the

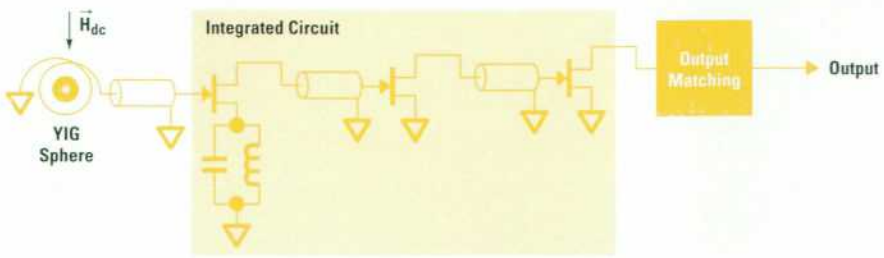


Fig. 1. DYO high-band schematic diagram.

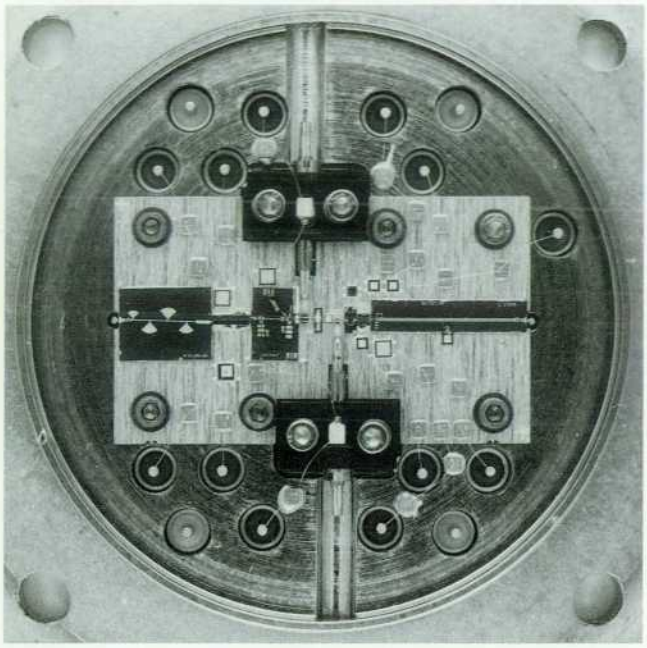


Fig. 3. Photograph of the DYO.

source feedback capacitor. This has the effect of reducing the capacitance on the source and thus the reflection gain at lower frequencies. In addition, a high-impedance coupling loop is used that does not provide the proper phase relationship for the lockup mode. To avoid lockup at the high end of the band the transmission line between the device and the resonator is kept short.

The buffer amplifier stages consist of a 300-μm FET followed by a 400-μm FET, both in the common source configuration. The oscillator stage is matched to the buffer amplifier using a short length of transmission line. The primary purpose of the buffer stages is to provide isolation and therefore a stable match to the oscillator stage, making the oscillator frequency independent of the load.

The YIG (yttrium iron garnet) resonator provides the high-Q tuning circuit for the oscillator. The high-band resonator is constructed of a 300-μm-diameter, undoped YIG sphere centered in a multiturn coupling wire. The ratio of sphere to loop diameters is a trade-off between suppression of spurious resonances and oscillation strength. The YIG resonator provides a resonance that tunes linearly with an applied magnetic field.[2]

The 11-to-20-GHz high-band oscillator is built on a 0.010-inch molybdenum carrier (Fig. 3). This carrier is held to the lid by studs inserted in the lid. The carrier's function is to provide a continuous ground plane. The YIG GaAs IC is soldered to a small heat spreader and then epoxy-attached to the carrier. A 0.010-inch fused silica microstrip circuit is epoxy-attached between the YIG resonator and the IC to provide the proper transmission line length. The output circuit is a 0.010-inch sapphire microstrip circuit which provides output matching and transition to a right-angle SMA connector.

The low-band 2-to-11-GHz oscillator consists of a YIG resonator, a bipolar transistor oscillator stage, a matching network, and a broadband buffer amplifier (Fig. 4). The oscillator stage uses a silicon bipolar transistor with an f_{max} of 22 GHz.

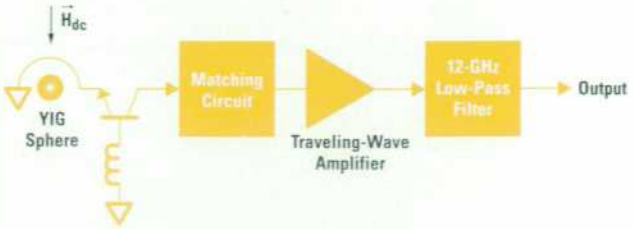


Fig. 4. DYO low-band schematic diagram.

The transistor is configured as a common base circuit with an inductor in series with the base terminal. The inductor is realized as a length of transmission line. This inductor transforms to a negative real impedance at the emitter port and thus meets the above criteria for oscillation. The collector port is terminated in a matching network that provides load conditions to optimize oscillation strength, harmonics, and linearity. The buffer stage consists of a 2-to-20-GHz traveling-wave GaAs IC amplifier. The YIG resonator consists of a 600-μm-diameter 550-gauss YIG sphere in a half loop of 950-μm-diameter wire.

The 2-to-11-GHz low-band oscillator is built on the same carrier as the high-band circuit. It uses two 0.010-inch sapphire microstrip circuits. The first circuit contains the bipolar transistor, the YIG coupling loop, and the collector matching circuit. Both the transistor and the loop are epoxy-attached to the circuit, which is also epoxy-attached to the carrier. The traveling-wave GaAs IC buffer amplifier is soldered to a heat spreader and epoxy-attached to the carrier. An output circuit provides low-pass filtering and transition to a right-angle SMA connector.

The YIG resonators require a dc magnetic field to tune the frequency. This magnetic field is applied perpendicularly to the applied RF field. The resonant frequency is related to the dc magnetic field by the equation:

$$f_o = \gamma(H_o + H_a)$$

where H_o is the applied dc magnetic field, H_a is the internal magnetic field and γ is the charge-to-mass ratio of an electron. The magnetic field is created by winding 1640 turns around a 6-mm-diameter pole tip as shown in Fig. 5. The 1.7-mm air gap under the pole tip is optimized to maximize tuning sensitivity and field uniformity, which affects the rejection of spurious resonances. The magnetic material used is a 50-50 nickel iron alloy that results in an effective magnetic saturation of over 30 GHz. The windings are wound in such a way as to optimize the internal forces when self-heating occurs. These forces can change the pole gap and therefore affect the tuning sensitivity. FM is accomplished by using a small 17-turn coil mounted on the pole tip, which adds to or subtracts from the main field.

Switched Amplifier Filter Detector

An integrated output microcircuit developed for the HP 83750 Series synthesizers provides a leveled output from 10 MHz to 20 GHz with exceptionally low harmonics and broadband noise. The goal was to create a low-cost circuit containing the required filtering, amplification, switching, and leveling in one package. Through integration, savings are realized in packaging, printed circuit boards, assembly, and testing.

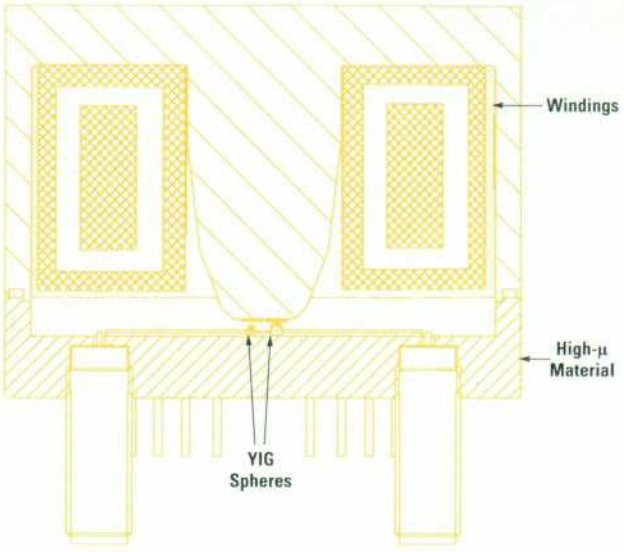


Fig. 5. Cutaway drawing of the DYO and SAFD magnet.

This microcircuit, called the switched amplifier filter detector, or SAFD, has two input paths as shown in the block diagram, Fig. 6. The low-frequency path passes 10-MHz-to-2.0-GHz signals through a thin-film low-pass filter. This filter is primarily designed to reject local oscillator feedthrough signals in the 5.4-to-7.4-GHz range. It also helps reduce harmonic signals above 3 GHz.

The high-frequency path covers 2.0 to 20 GHz. This path is designed to produce better than –45-dBc harmonics at an output power of +10 dBm. The filtering is provided by a pair of magnetically tuned YIG resonators. Since YIG filters are both lossy and exhibit power limiting characteristics, each one is carefully tuned to provide sufficient power and bandwidth. Of particular concern in this path is shielding among the various components. To attain low harmonics, it is essential to isolate each section, and in particular to isolate the YIG filters from each other. With this design, the performance is limited by the harmonics generated in the amplifier between the two resonators (without this buffer amplifier, the filters cannot be tuned independently). The amplifier is a broadband GaAs MMIC traveling-wave amplifier covering the 2-to-20-GHz range.

The magnetic field for the YIG filters is provided by two coils. A 1640-turn coil wound on a very high-permeability core provides the main 7000-gauss field within a 1.7-mm air gap to tune the YIG filters up to 20 GHz. The coil is specially designed to minimize any change in the gap size resulting from internal or external temperature variations, which would change the field intensity and thus the center frequency of the filter. It is critical that both filters are tuned to the same resonant frequency to minimize filter loss. Therefore, a small offset coil is placed close to the input filter to correct for slight differences in field strength which increase with frequency. A linear current-versus-frequency ramp is sent through this coil to compensate for the difference. Because the YIG filters must tune with the 2-to-20-GHz YIG oscillator, the packaging and magnetic design are similar to the oscillator's to reduce tracking errors.

A pair of p-i-n diodes has a dual function. Switching between the low-frequency and high-frequency paths is just a matter of turning both diodes on or off. The second function involves the bridge detector on the output. The bridge is a GaAs integrated circuit with thin-film resistors. Because of their small geometry, these resistors must be protected from dissipating excessive power. If an excessive level is detected by the bridge, clamping circuitry on the bias board shuts off the p-i-n diode bias. Under normal operation, this clamp is only a safety feature because the ALC loop in the instrument also limits the input power to the SAFD. Following the p-i-n switch, a thin-film 20-GHz low-pass filter reduces out-of-band harmonics.

The last circuit in the SAFD contains the leveling bridge. Because of proper ratioing of resistors, the voltage across the bridge diode is proportional only to the voltage incident on the load and is independent of the signal reflected from the external load. Therefore, the bridge exhibits directivity. Because the diode is operating at low power levels, its dc output voltage is proportional to the square of the voltage

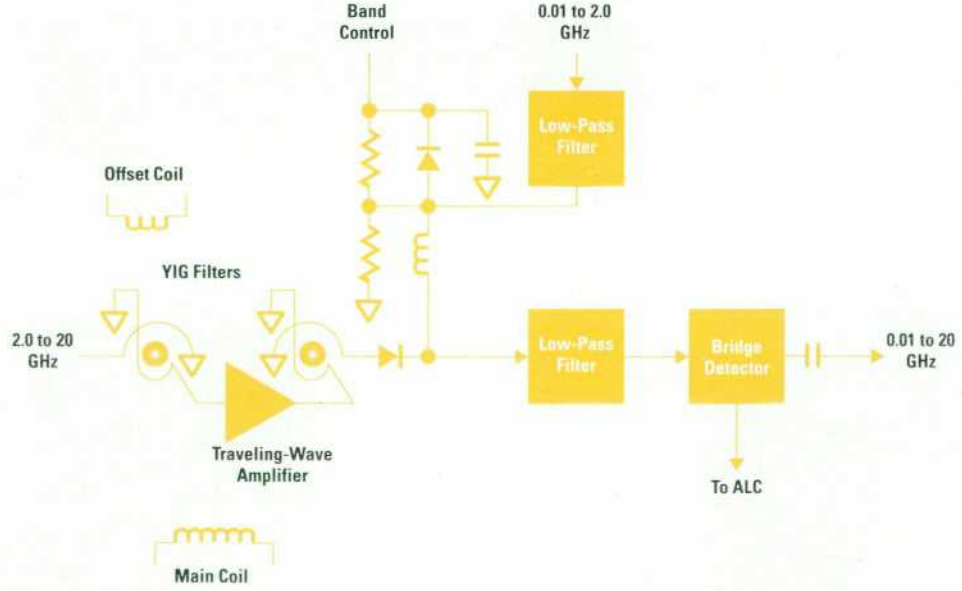


Fig. 6. SAFD microcircuit block diagram.

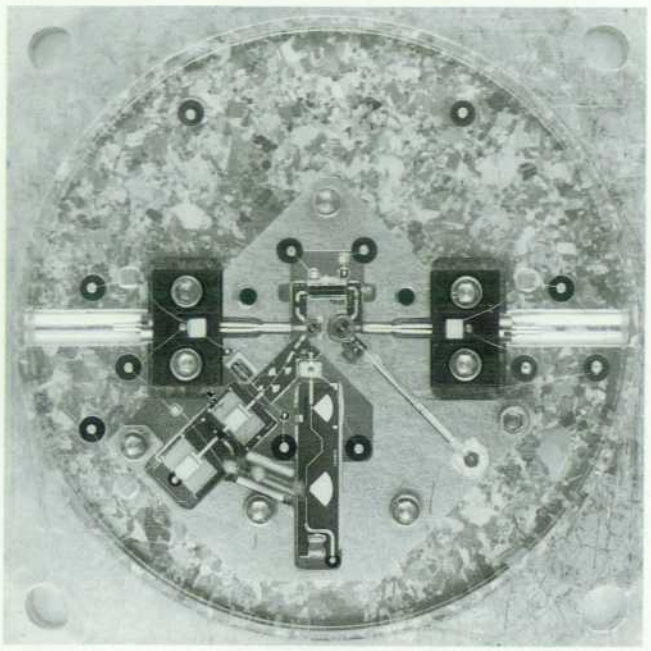


Fig. 7. SAFD microcircuit.

across it, which is proportional to the incident output power. The dc voltage from the diode is used by the ALC loop to provide improved output power flatness as a function of frequency. Because of the bridge's directivity, the leveling circuit ignores reflections from the load and thus provides a good source match.

Fig. 7 is a photograph of the SAFD microcircuit.

0.01-to-2-GHz Heterodyne Band

The 0.01-to-2-GHz band in the HP 83752A/B sweepers is generated by mixing 5.410 to 7.4 GHz (f_{LO}) from the DYO with a phase-locked 5.4-GHz (f_{RF}) oscillator in the heterodyne band microcircuit, or HetBand for short. The heart of this microcircuit is a GaAs RFIC mixer. Fig. 8 is the HetBand block diagram.

The GaAs RFIC mixer uses a doubly balanced FET bridge mixer with on-chip RF and LO phase splitting amplifiers and an on-chip differential-to-single-ended IF amplifier. Mixer design considerations include LO and RF drive levels, signal purity, and port match. Port match can be a problem at the sum and difference frequencies and at the harmonics of f_{LO} and f_{RF}. Poor port match can cause reflections back into the mixer and degrade the level of spurious signals. With a discrete mixer, performance can often be dependent on assembly techniques. The GaAs RFIC mixer has buffered RF, LO, and IF ports, making it insensitive to these assembly variations. The bias on the FET bridge mixer is the only critical adjustment. It was determined that for best performance over temperature, the mixer bias must be held to within ±0.01V of its room-temperature setting.

Measurements on the GaAs RFIC mixer indicated that there was some dependence of the 2-1 spurious product (second harmonic of f_{RF} mixing with f_{LO}) on the RF port source match at the frequency of the second harmonic, 10.8 GHz. Because it was desired to reduce the mixer's spurious products by controlling the harmonics at the RF input, there is a 6.6-GHz low-pass filter at the RF port for this purpose. As is characteristic of simple low-pass filters, the match at 10.8 GHz is very bad, since it is in the stop band. The second harmonic generated by the mixer comes out the RF port and is reflected by the poor match of the filter. The second harmonic then reenters the mixer where it is amplified and adds, in or out of phase, causing a variation in the 2-1 spurious performance. An attenuator was added to the RF input of the mixer to improve the source match and reduce the reflection back into the mixer. This reduces the 2-1 spurious product to a low enough level that a filter at the IF output can reduce the level below the spurious specification.

Amplitude modulation is done in the RF path. A broadband stagger-spaced p-i-n diode modulator is used because a quarter-wave-spaced modulator has less attenuation at the second and fourth harmonic frequencies. This ensures that there will be no increase in the harmonic levels through the modulation range, which would degrade the spurious performance.

The output power of the GaAs RFIC mixer is +5 dBm. The relatively high output power from the mixer requires less gain from the output amplifier. The lower overall gain requirement results in a low level of broadband noise. The additional power and gain are provided by a new GaAs RFIC power amplifier designed for this application. This IC is a dc coupled feedback amplifier designed specifically for high

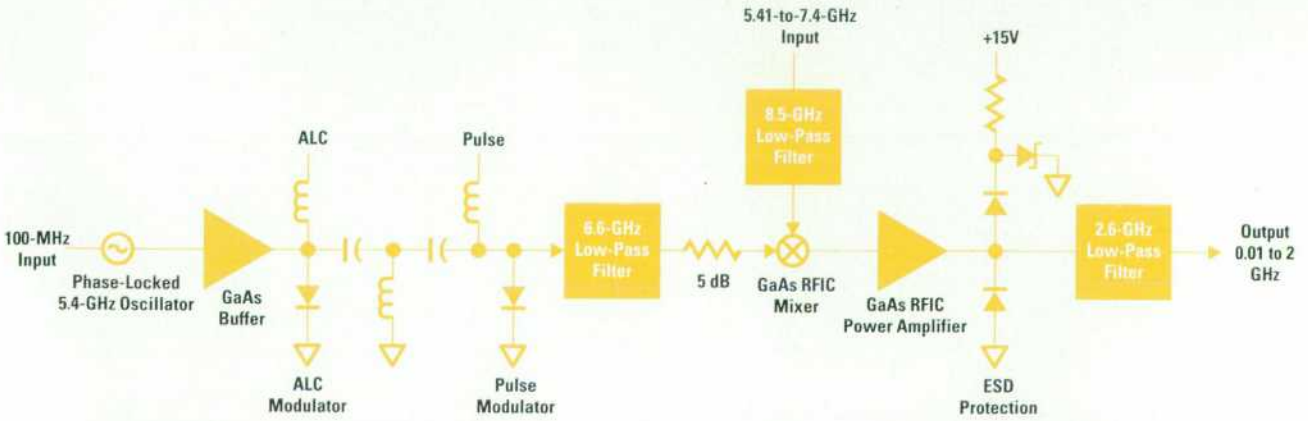


Fig. 8. HetBand microcircuit block diagram.

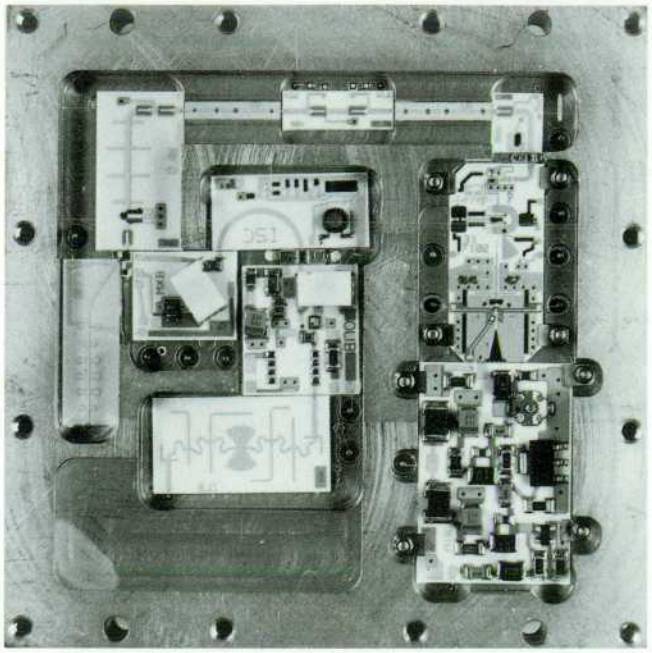


Fig. 9. HetBand microcircuit.

power, good harmonics, and wide bandwidth. ESD protection and a combination harmonic and spurious low-pass filter are added to the output.

The microcircuit is constructed in a deep-well stainless-steel package using large thick-film alumina circuits epoxy-attached directly to the package floor (Fig. 9). This eliminates the need for circuit clamps. A tellurium copper heat sink is mounted into the package floor with the RFIC power amplifier mounted directly to this heat sink. The heat sink extends out the back of the package and has fins for cooling machined into it. This combination provides the low thermal expansion of stainless steel and the high thermal conductivity of copper.

Combiner Modulator Amplifier

The ModAmp, short for combiner modulator amplifier, combines the low-band and high-band signals from the dual YIG oscillator (DYO) and redirects them to the HetBand or SAFD microcircuits. A signal is also sent to the sampler for phase locking. The ModAmp provides a pulse modulator and amplitude modulation for the ALC circuit.

The DYO has two outputs: a low band, 2 to 11 GHz, and a high band, 11 to 20 GHz. Each DYO output must be switched to the buffer amplifier and must be available at the sampler output port. In addition, the DYO low-band output must be switched to the 0.01-to-2-GHz HetBand microcircuit. The switch and coupler configuration are shown in Fig. 10, the ModAmp block diagram. The 11-to-20-GHz coupler is the load for the 2-to-11-GHz coupler. The advantage is that instead of having to design a 2-to-20-GHz coupler, we need only two simpler narrowband couplers. If a single broadband coupler were used, then there would have to be a switch at the input to combine the DYO outputs and a switch after the coupler to switch the output to the HetBand. By using two couplers, one switch is eliminated from both the heterodyne path and the 11-to-20-GHz path, thereby reducing the path loss in these two bands.

A GaAs IC buffer amplifier is placed between the reflective ALC modulator and the DYO. This buffer prevents amplitude modulation resulting from the reflective modulators from entering the sampler port and causing AM-to-PM conversion in the sampler. The pulse and AM modulators use the p-i-n diode modulators developed previously.[3] A high-pass filter between the two modulators reduces the cross talk between the modulators. Following the pulse modulator is a gain and a power stage to drive the SAFD.

The ModAmp is built in a deep-well stainless-steel package with 0.010-inch sapphire and alumina thin-film circuits epoxy-attached directly to the package floor (Fig. 11). The modulators are placed in a 3-mm channel machined into the package floor. The channels form a waveguide operating below cutoff to provide the isolation that deep modulation requires.

Acknowledgments

For the DYO, the authors would like to thank Andy Smith for his contribution to the magnet design, Kristi Rasmussen and Gary Gilbreth for process development, Rick James and Jim Grishaw for the low-band oscillator development, Michio Furukawa, Dale Albin, and Andy Howard for the high-band

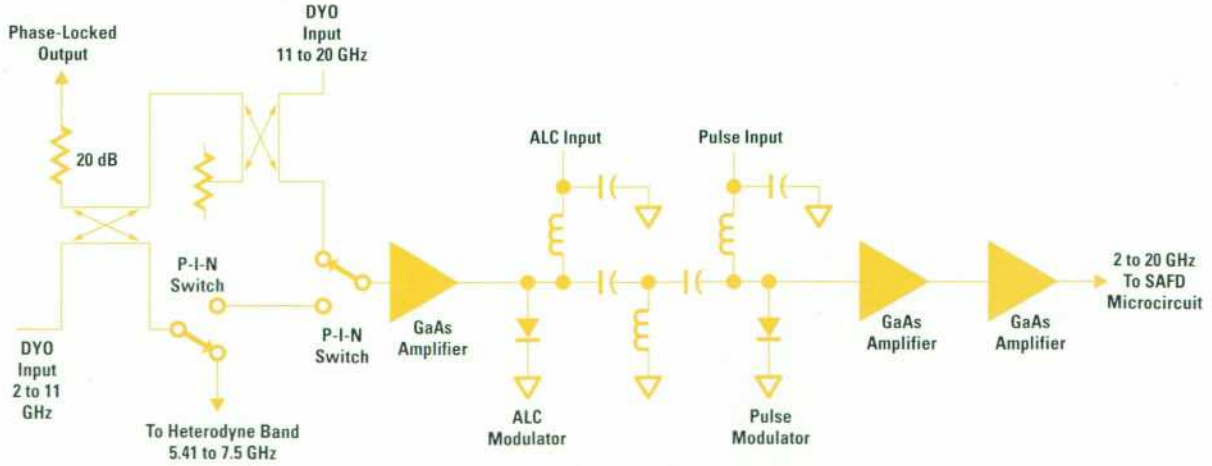


Fig. 10. ModAmp microcircuit block diagram.

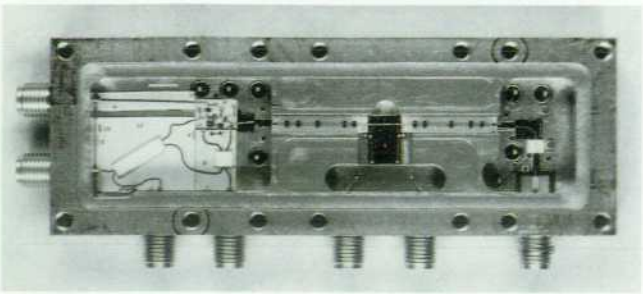


Fig. 11. ModAmp microcircuit.

oscillator development, Doug Fullmer and Julio Perdomo for their work on the high-band GaAs IC, and Mike Sohigian and Arlen Dethlefsen for their guidance.

The SAFD microcircuit was made possible by the efforts of Jim Grishaw, microwave design engineer, Andy Smith, mechanical design engineer, Eric Ehlers, ALC bridge designer, and Lance Haag, systems engineer.

For the HetBand and ModAmp microcircuits, the mechanical design was done by Andy Smith and the new process development was done by Mike Shook. For the HetBand microcircuit, Tim Shirley designed the GaAs RFIC mixer and power amplifiers and Pat Harper offered invaluable advice and developed the production test process.

References

1. G. Basawapatna and R. Stancliff, "A Unified Approach to the Design of Wide-Band Microwave Solid-State Oscillators," *IEEE Transactions on Microwave Theory and Techniques*, Vol. MTT-27, May 1979, pp. 379-385.
2. J. Helszajn, *YIG Resonators and Filters*, John Wiley and Sons, 1985.
3. M.K. Koenig, "A High-Speed Microwave Pulse Modulator," *Hewlett-Packard Journal*, Vol. 42, no. 2, April 1991, pp. 34-36.

A Programmable 3-GHz Pulse Generator

This new one-or-two-channel pulse generator provides precise edge placement, extensive functionality, and an interactive user interface. It is designed to help characterize and debug CMOS, ECL, and GaAs devices and signal integrity problems.

by Hans-Jürgen Wagner

A major trend for the computer and communications industries is the need to process more and more data in less and less time. This impels designers of digital devices for these industries to develop faster and more complex devices. For example, the clock speed of CMOS microprocessors has increased by a factor of two every three to four years. Designers are expected to achieve these higher speeds while maintaining system reliability and reducing the price. This often means that for a given technology, processes may be driven to their limits, and the problems faced by designers of digital devices may be analog problems. Typical of such problems are crosstalk, bandwidth limitations, ground bounce, jitter, reflections caused by mismatches, pattern dependencies, and so on.

The HP 8133A pulse generator, Fig. 1, is designed for customers who have to characterize and debug these signal integrity problems on the bench or in a test system with a sampling oscilloscope as a response unit. Fast CMOS, ECL, and GaAs devices can be stimulated by the HP 8133A, which runs at clock rates up to 3 GHz in single-channel or two-channel configurations (up to six channels with an accessory kit). Margin testing, worst-case testing, device characterization and debugging, and analysis of signal integrity problems are typical applications. Fig. 2 shows typical waveforms that can be generated by the HP 8133A.

The key contributions of the HP 8133A are its precise edge placement, its extensive functionality, and its interactive human interface. Precise edge placement is achieved by fast, fixed transition times (< 100 ps, 10% to 90%), very low jitter (< 5 ps rms), fine timing resolution (1 ps), and stable edges regardless of parameter settings (delay accuracy < 150 ps, width accuracy < 100 ps). The small jitter and the fine resolution allow, for example, characterization of the metastability of a flip-flop. The typical jitter of less than 2 ps results in a peak-to-peak jitter of about 12 ps. Thus, measurement resolution of 10 ps is possible without averaging.

The functionality of the HP 8133A pulse generator includes programmable pulse amplitude, offset, delay, width, and repetition rate. Table I shows the parameter ranges. Additionally, adjustable phase and skew allow the instrument to address multiphase clock applications without an external controller. The square mode offers a 50% duty cycle with variable frequency. The hardware architecture allows adjustment of the delay parameter over its entire range at any frequency; for example, at 1 GHz, ±5 periods of delay are available. Also, the width parameter does not suffer from any recovery time limitations as it does in traditional pulse generator architectures. The trailing edge can be moved anywhere within the period, limited only by the instrument bandwidth.

Second Channel Options

For the optional second channel the customer can choose either a second pulse channel or a pulse/data channel. The second pulse channel offers the same parameters as the first channel. In addition, the frequency of this channel can be divided by 1, 2, 4, 8, 16, 32, or 64. Thus, the two channels can run at different frequencies, which is important for testing the setup and hold times of flip-flops, for example.

Fig. 1. The HP 8133A pulse generator provides pulses with repetition rates up to 3 GHz. All pulse parameters are accurately programmable. An optional second channel can be either a pulse channel or a pulse/data channel capable of generating a 32-bit data word or a pseudorandom binary sequence.

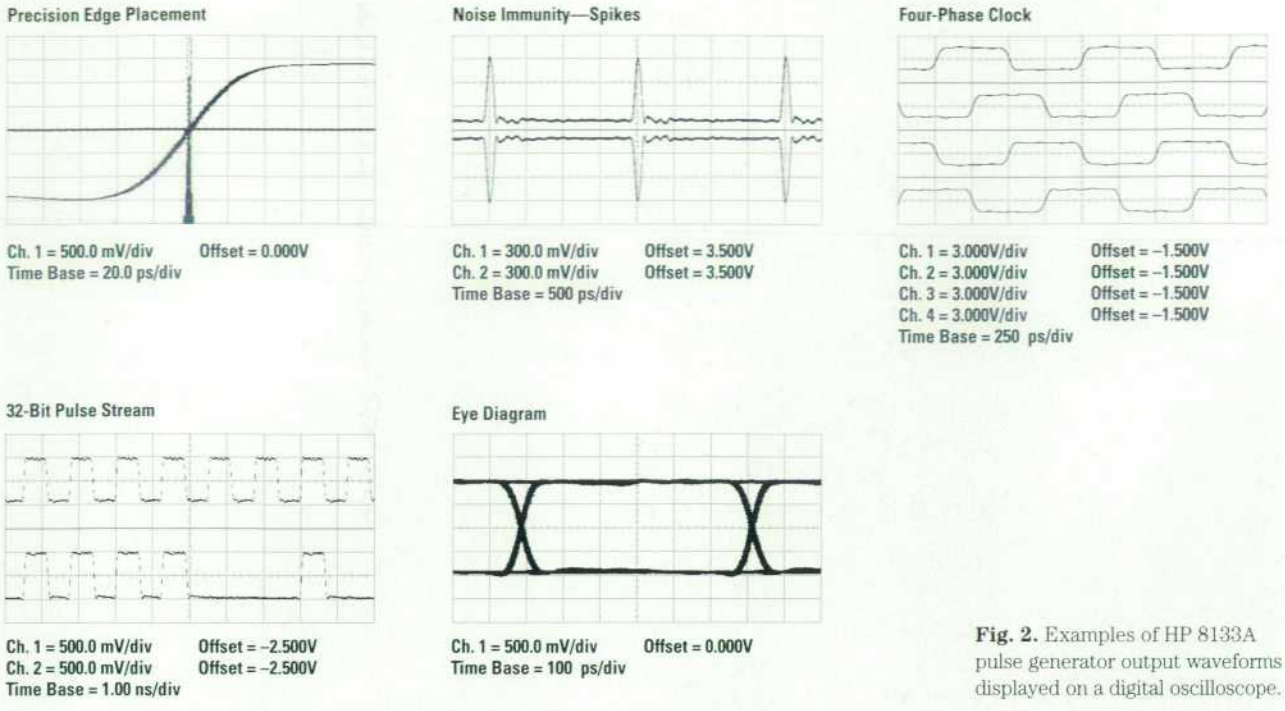


Fig. 2. Examples of HP 8133A pulse generator output waveforms displayed on a digital oscilloscope.

Table I
HP 8133A Pulse Generator Parameter Ranges

Repetition Rate: 32 MHz to 3 GHz

Delay: –5 ns to +5 ns (–5 ns to +15 ns, square mode only)

Width: 150 ps to 10 ns

Amplitude: 300 mV to 3V pp into 50 ohms, 600 mV to 6V pp into an open circuit

Output Voltage Window: –2V to +4V into 50 ohms, –4V to +8V into an open circuit

Trigger Out Amplitude: 0.5V pp to 1.8V pp into 50 ohms

Trigger Out Voltage Window: –4V to +4V

The pulse/data channel offers a square wave mode, a programmable 32-bit pattern, and a pseudorandom binary sequence (PRBS) of length $2^{23} - 1$ bits. The square wave signal can be divided by 1, 2, 4, 8, 16, or 32. The 32-bit pattern is useful for testing the pattern dependent settling of a GaAs line driver or a chain of ECL gates with different rise and fall times, for example. The PRBS makes it possible to generate an eye diagram that shows the noise, timing and bandwidth problems, and margins of the device under test in a single picture. The design of the pulse/data channel is discussed in the article on page 56.

The trigger output is a programmable square wave signal. The trigger frequency can be divided by 1, 2, 4, 8, 16, 32, or 64. This makes it possible to trigger an oscilloscope even if the oscilloscope has a limited trigger bandwidth. It can also improve the jitter performance of the measurement. The programmable output amplitude of up to 1.8V pp allows the use of a power splitter to get two ECL signals, one to trigger the oscilloscope and the second for the device under test as a third pulse channel.

In the external clock mode the input frequency can be divided by 1, 2, 4, 8, 16, 32, or 64. In a master/slave four-channel configuration the slave can run at a divided frequency. The input frequency is measured and displayed, and the phase and duty cycle parameters are programmable with the same accuracy as in internal mode; no external controller is needed.

Architecture

Fig. 3 shows the block diagram of the instrument for the different two-channel configurations. The interconnection between the timing board and the two output channels is at the rear panel so that additional delay lines for each channel can be added. This is useful in a multichannel master/slave configuration (also shown in Fig. 3) to deskew the output channels of the master compared to the output channels of the slave.

For each functional block we tried to find the lowest-cost technology that fit our manufacturing process and met the performance goals with sufficient margin. The technologies include:

- IC Processes
  - ECL programmable counter ASIC
  - ECL multiplexer ASIC
  - ECL sequencer/PRBS gate array
  - Si pulse formatter
  - GaAs output amplifier
- Hybrid Processes
  - Analog delay thick-film hybrid
  - Switched delay thin-film hybrid
  - Output amplifier thick-film hybrid
  - Pulse formatter thick-film hybrid.

Fig. 3. Two two-channel HP 8133A pulse generators in a four-channel master-slave configuration. In the lower instrument channel 2 is a pulse channel like channel 1, while in the upper instrument channel 2 is a pulse/data channel. External delay lines compensate for timing skew between the output channels of the two instruments.

- Printed Circuit Boards
  - Standard FR-4 HP boards, surface mount components
  - Discrete shaper amplifiers (GaAs FETs)
  - Customized semirigid interconnections

Design details of the various functional blocks of the instrument can be found in the article on page 60.

User Interface

The interactive human interface allows direct access to each pulse parameter with the press of a key or two. The front panel and user interface were developed with the following major goals in mind:

- Make the HP 8133A "look and feel" like an HP 8130A or 8131A pulse generator[1] to show that they belong to the same family of pulse generators and to make it easy for a user who is familiar with the HP 8130A/31A to work with the HP 8133A.
- Enhance the parameter display.
- Make it quick and easy to change parameters.
- Allow the user to focus on the device under test while varying parameters.

Parameter Display. After experimenting with different display modules, we decided to use four 8-character, 5-by-7 dot matrix LED display modules, which offer several advantages. This arrangement can display two parameters at a time with their names and units. It can display a graphic representation of the 32-bit data word of the data channel using the user-definable character feature of these modules. It can easily highlight the current parameter using the software-adjustable brightness feature of the display modules. It also has the desired look, close to that of the HP 8130A/31A.

Keyboard and LEDs. For the front-panel keys and LEDs, we decided to stay as close as possible to the HP 8130A/31A. We use the same keys as in the HP 8130A/31A because they are known to be reliable (although relatively expensive). The LEDs have similar meanings to those in the HP 8130A/31A. LEDs in the keys show the currently selected parameters. LEDs above the keys show which of the two parameters or modes that this key represents is selected. All parameters are accessible through one or at most two key presses. The operating modes of the instrument can be seen at a glance without switching display pages.

One conceptual change was made. The vernier keys behave differently than in the HP 8130A/31A. There are no range keys, and there are five pairs of up-down keys (instead of three). Each vernier key has a fixed delta assigned to it—depending on the parameter, of course. The advantage is that each press of a vernier key causes a constant increment or decrement, unlike the HP 8130A/31A, where the increment or decrement changes with the range of the parameter.

Error Behavior. All parameters can be modified within their hard limits, whether or not they generate a conflict with other parameters, such as width greater than period or high level less than low level. In case of such conflicts, the user is directed out of the error state with a full-text error message and a flashing arrow in the display that indicates in which direction the parameter must be modified to resolve the conflict.

Reference

1. W. Berkel, et al, "500-MHz and 300-MHz Programmable Pulse Generators," *Hewlett-Packard Journal*, Vol. 41, no. 4, August 1990, pp. 64-78.

Pulse/Data Channel Extends Programmable Pulse Generator Applications

This optional second channel for the HP 8133A pulse generator has a dividable square wave mode, a 32-bit data burst mode, and a pseudo-random binary sequence (PRBS) mode. Its major components are a data gate array, a multiplexer, a phase-locked loop, and an output section. Most circuits are ECL.

by Christoph Kalkuhl

The HP 8133A 3-GHz pulse generator is available in single-channel and two-channel configurations. Two options are available for channel 2: a second pulse channel like channel 1 or a pulse/data channel. The pulse/data channel is designed to offer some additional features:

- A square wave that can be divided by 1, 2, 4, 8, 16, or 32
- A 32-bit data pattern that can be edited
- A pseudorandom binary sequence (PRBS) with a length of $2^{23} - 1$ bits, according to CCITT recommendation O.151.

Fig. 1 shows waveforms from a pulse channel and a pulse/data channel illustrating the divided square wave capability. The pulse/data channel (channel 2) is generating a square wave at half the frequency of the pulse train generated by channel 1. Figs. 2a and 2b show pulse/data channel data patterns in RZ (return to zero) and NRZ (nonreturn to zero) data formats. Fig. 3 shows a PRBS eye diagram with sampling clock. The data format is NRZ as in most applications. The PRBS is also available in RZ format.

Applications

The square wave mode addresses applications in which a clock signal is needed. The divided square wave is useful for testing flip-flops or for applications that require multi-frequency clock signals.

The 32-bit data mode provides repetitive bursts for use with an oscilloscope. The settling behavior of amplifiers and line

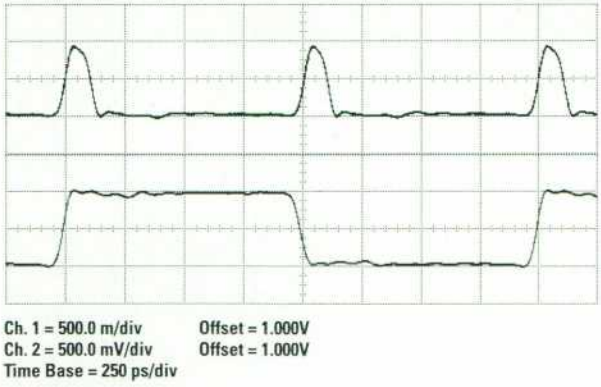
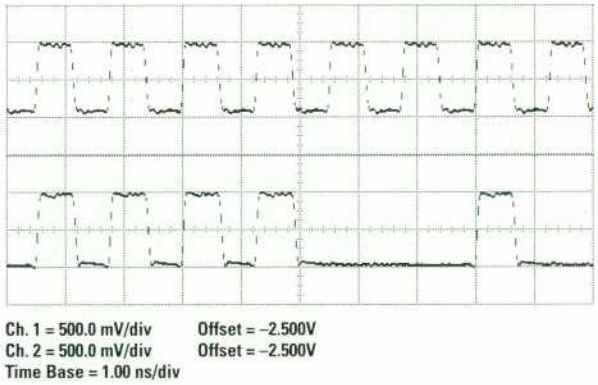


Fig. 1. Channel 1 (top): Normal pulse output. Channel 2 (bottom): divided square wave at 50% duty cycle and half the frequency.



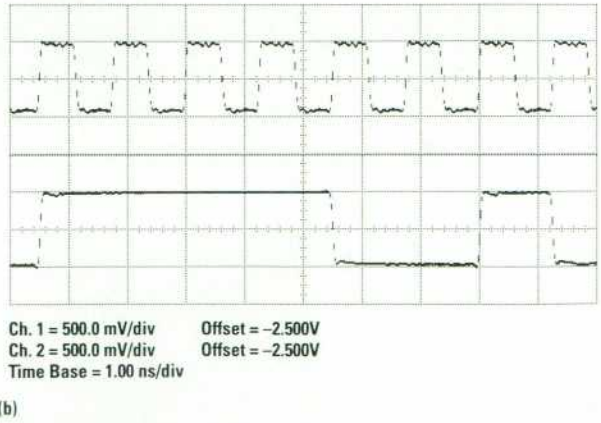
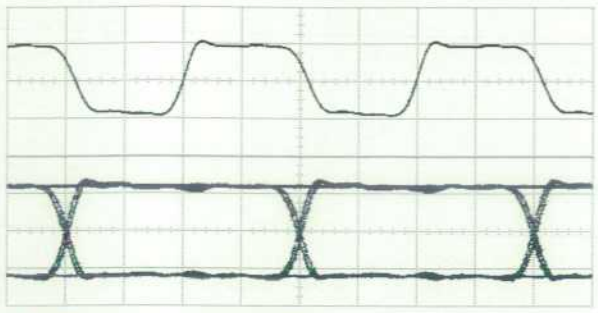


Fig. 2. Channel 1: Normal pulse output. Channel 2: (a) RZ (return to zero) data pattern. (b) NRZ (nonreturn to zero) data pattern.



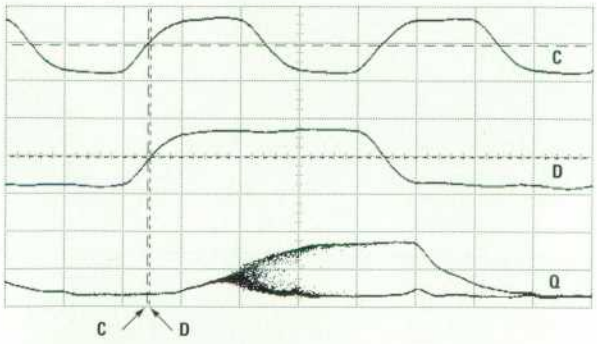
Ch. 1 = 500.0 m/div Offset = 1.000V
Ch. 2 = 500.0 mV/div Offset = 1.000V
Time Base = 250 ps/div

Fig. 3. Channel 1: Sampling clock. Channel 2: PRBS eye diagram.

receivers can be seen with some bits set to 1 and the rest set to 0. Editing of the pattern is quick and easy. A rotate feature scrolls the pattern through the oscilloscope screen.

Fig. 4 illustrates a measurement of the metastability of a flip-flop. The picture shows the clock signal C, the data input D, and the output Q of the flip-flop. C and D are monitored at the input of the DUT (device under test). The transitions of the pulse generator have been slowed down by transition-time converters (HP 15435A, 150-ps transitions) to avoid pulse distortion by the capacitive load of the flip-flop inputs. Metastability is caused by violating the setup and hold times of the flip-flop. Clocking the flip-flop exactly on the edge of D sets the flip-flop to a metastable condition. Thus, the metastability can be used to find the sampling point of the flip-flop and the corresponding D-Q timing relationships. The time window where the metastability occurs is very narrow—about 2 ps. If the clock C moves only 1 or 2 ps left or right with respect to D, the metastability will disappear. For this reason, this is a very difficult measurement to make, and the fast slopes, accurate timing, and low jitter of the HP 8133A are essential.

The purpose of the PRBS feature is not primarily for use with a bit error rate tester (BERT), although it will work



Ch. 1 = 500.0 mV/div Offset = –840.0 mV
Ch. 2 = 500.0 mV/div Offset = 750.0 mV
Ch. 3 = 500.0 mV/div Offset = 2.290V
Time Base = 500 ps/div Delay = 21.2900 ns
Delta t = 20.8 ps
Start = 22.5026 ns Stop = 22.5234 ns

Fig. 4. Metastability of a flip-flop. Channel 1 (top): Flip-flop clock input. Channel 2 (middle): Flip-flop D input. (bottom) Flip-flop Q output.

well and provide excellent pulse performance. A very important application is the characterization of high-speed devices. To a user with a little experience, the PRBS can show all of the problems of a circuit in one picture, since it contains multiple frequencies and duty cycles. Bandwidth problems within a circuit that are hidden by faster circuits downstream in the signal path cause increased jitter, for example.

Block Diagram

Fig. 5 shows the block diagram of the pulse/data board. The square wave clock signal from the timing board is refreshed by a shaper amplifier. A 4:1 multiplexer (MUX) adds the data information and determines the data format (RZ = return to zero, NRZ = nonreturn to zero). The signal then goes to an output section identical to that of the standard pulse channel, consisting of a GaAs amplifier with support circuits and disable relays.

The multiplexer receives the data bits from the data gate array, which contains the bitstream sequencer and a hardwired PRBS generator. This gate array, which is leveraged from the HP 80000A data generator, has a maximum operating frequency of 1 GHz. The data memory consists of simple (low-power) static latches, which contain the programmable 32-bit word. The PRBS generator outputs four $(2^{23} - 1)$-bit PRBS bitstreams at one-fourth the clock rate which are multiplexed together to form the final $(2^{23} - 1)$-bit PRBS bitstream. The data gate array also generates a frame signal to mark the start of the 32-bit word.

The bitstreams between the data gate array and the multiplexer must be synchronized. Since the data gate array has a propagation delay of several nanoseconds, the clock for this device has to be early by this amount with respect to the CLK/4 reference output of the multiplexer. This negative delay is realized by a phase-locked loop (the negative delay is only possible in repetitive applications).

The phase-locked loop consists of the common functional blocks: phase detector, loop filter (or regulator), VCO (voltage-controlled oscillator), and frequency divider. The VCO has a one-octave range (500 MHz to 1 GHz), and lower frequencies are produced by division. The feedback and reference paths are divided by 16 because of the frequency limit of the phase detector. A limit-and-detect circuit ensures proper control voltage conditions for the VCO. This circuit detects the end of a frequency range and generates an interrupt. Details of the phase-locked loop are discussed below and are shown in Fig. 6.

A temperature sensor measures the chip temperature of the data gate array. This is necessary for delay drift compensation in the gate array. The temperature drift of the gate array is compensated with internal adjustable delay lines.

The frame marker signal goes through a retiming circuit to avoid internal synchronization jitter on the trigger signal (which can occur in the phase-locked loop). The output signal of this retiming circuit is internally called STROBE but is actually the trigger signal in BIT0 trigger mode. Jitter on this trigger would cause a jittery display on the oscilloscope even when the output signals are clean. The delay circuit ahead of the retiming circuit is used for delay drift compensation of

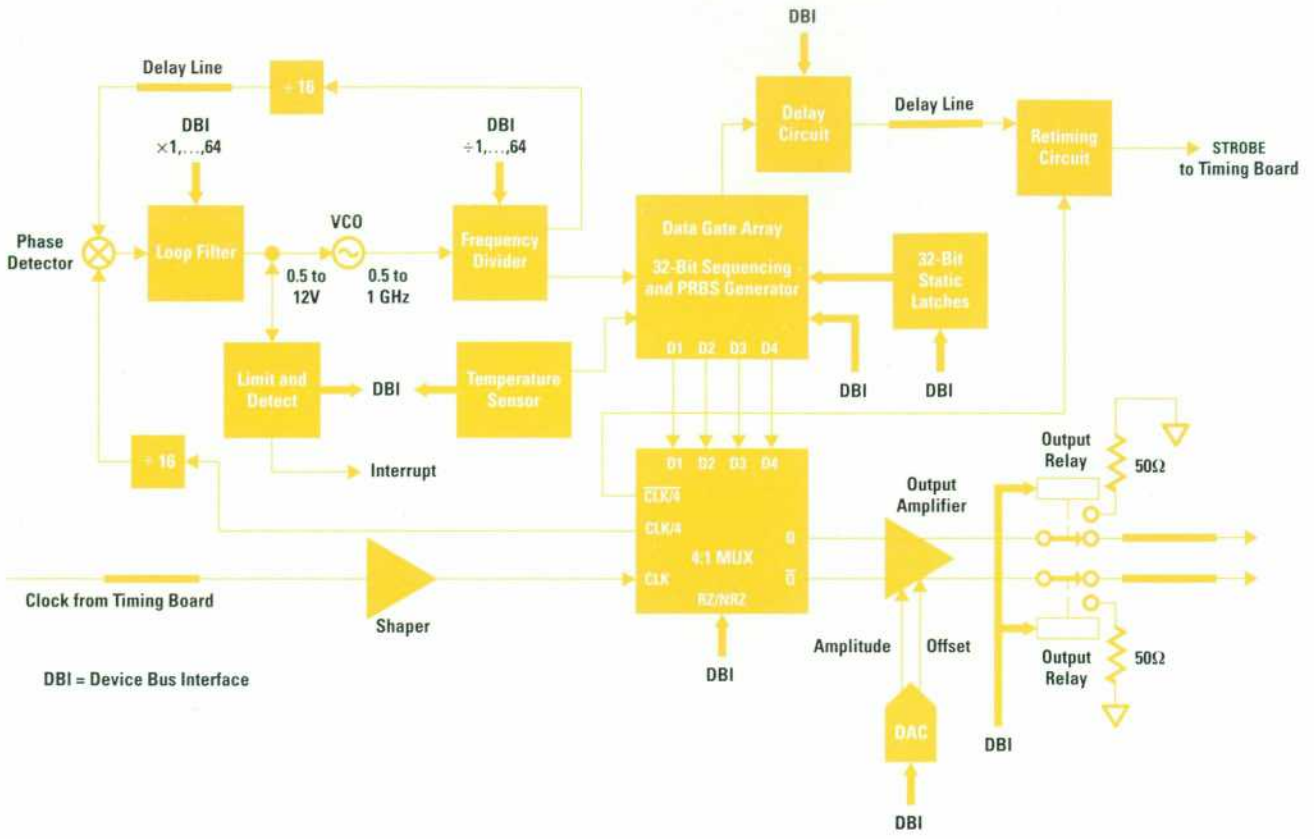


Fig. 5. Block diagram of the optional pulse/data board for the HP 8133A pulse generator.

the frame marker signal over temperature while compensation of the D1, D2, D3, and D4 gate array output data channels is handled by internal adjustable delay lines in the data gate array.

Phase-Locked Loop

The phase-locked loop on the pulse/data board generates a negative delay of 12 ns. This negative delay is required because of the internal delay of the data gate array as mentioned above. As shown in Fig. 5, the loop reference is the clock/4 (CLK/4) output of the 4:1 multiplexer (MUX). The loop output to the data gate array is one of the two outputs of the programmable frequency divider.

The demands on the performance of the phase-locked loop are severe. It must work over a 7-octave range with high phase accuracy and low synchronization jitter. At high frequencies, jitter and timing accuracy are the main concerns, while phase accuracy becomes important at low frequencies. The timing window at a clock frequency of 3.2 GHz is 1.25 ns (clock divided by 4). Setup and hold times, jitter, and delay deviations (only 200-ps steps are possible in the data gate array) must fit within this window.

By factory calibration the data acquisition point (the time when the signals on the data lines are considered valid and can be sampled) is set to 650 ps (half the period at the highest frequency) after an internal data transition. Thus the time from the data transition to the acquisition point remains nearly constant while the time from the acquisition point to the next data transition is longer at lower frequencies. At low frequencies phase accuracy becomes important. 650 ps is only about two degrees of phase error at the lowest frequency (CLK/4 = 8.25 MHz). There is a high risk that errors in timing could occur if the phase were to lag only a few degrees. Therefore, a prephase shift is built-in, which shifts the data transitions ahead at all frequencies. A few degrees are enough, but the direction is very important. The absolute amount of the prephase shift depends on the individual hardware and is measured during internal delay calibration.

Dividers. The loop phase detector has a maximum operating frequency of 80 MHz, while the CLK/4 signal is between 8 MHz and 800 MHz (at an instrument frequency of 32 MHz to 3.2 GHz). Therefore, the reference and the VCO output after the programmable frequency divider have to be divided by 16. These dividers consist of ECL D-type flip-flops. Two ICs are used, each containing four flip-flops ($16 = 2^4$). However, instead of the VCO signal path going through one IC and the reference path going through the other, both paths go through both ICs and are divided by four in each IC. The advantage of this arrangement is better matching of the temperature dependent delays in the two paths, at the expense of a slight increase in cross talk.

Negative Delay. The effect of the loop and phase detector is to cause the divided VCO output to track the reference input with zero phase difference. The required negative delay between the reference and the divided VCO output is produced by a delay line in the loop. The delay line is placed after the dividers to reduce the influence of cable loss, which increases at higher frequencies.

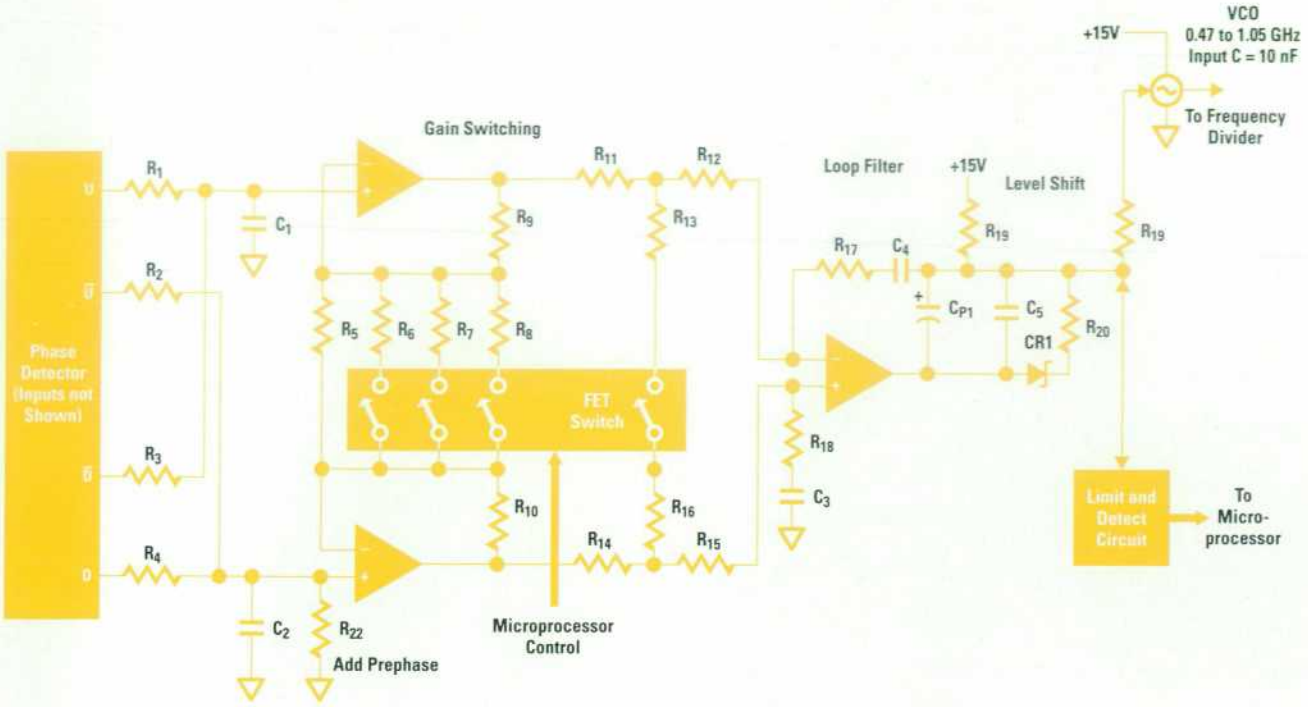


Fig. 6. Phase-locked loop details.

Phase Detector. The MC12040 phase detector is a digital phase-and-frequency-sensitive ECL type with a maximum operating frequency of 80 MHz. It must cover a 7-octave range and not lock on harmonics, which would cause improper range detection. Proper termination of both input lines is important because of the fast slopes. The phase detector produces four outputs (U, $\overline{U}$, D, $\overline{D}$) which are cross-wired (U to $\overline{D}$ and D to $\overline{U}$) for temperature compensation of the ECL output levels, which have a strong influence on the phase error. The resulting two outputs go to the loop filter (see Fig. 6).

Loop Filter. The first stage of the loop filter is a symmetrical low-pass filter consisting of R_1 to R_4, C_1, and C_2. The following stage is a combination of an instrumentation amplifier (consisting of three operational amplifiers) and a PI (proportional integral) regulator. Range switching of the programmable frequency divider requires related gain switching of the filter, which is realized by varying the gain-controlling resistors and a switchable double-T attenuator. This design was chosen for its low dc offsets (which cause phase error). An earlier, nonsymmetrical design, using a DAC for attenuation, had unacceptable dc offsets.

In the first stage of the loop filter, a slight offset is added by resistor R22 to achieve the prephase shift discussed above. The phase detector has ECL-level outputs (the 50-ohm terminators to –2V are not shown in Fig. 6), so the average dc level at the + inputs of the symmetrical operational amplifiers is about –1.3V. Thus a current is injected from ground through R22.

The symmetrical PI regulator has a level shifter at its output so that the operational amplifier has a higher margin to the positive limit of its output. The limit and detect circuit limits the VCO control voltage window and detects when the limits are reached. A serial resistor between the loop filter and the VCO forms a low-pass filter with the 10-nF VCO input capacitance.

Oscillator and Frequency Divider. The oscillator is a varactor LC type. A resistive attenuator and a blocking capacitor adapt the output to ECL levels for the programmable frequency divider. The programmable frequency divider is a high-speed ECL ASIC that permits division by 1, 2, 4, 8, 16, 32, or 64. One of the outputs is used to close the loop while the other feeds the data gate array with the negatively delayed signal.

Acknowledgment

I would like to thank Thomas Thoet, who designed the data gate array.

Design of a 3-GHz Pulse Generator

Period, delay, and width generation for the HP 8133A pulse generator depend on several thick-film and thin-film hybrid circuits and custom GaAs and bipolar ICs. The high frequencies and fast transitions made radiated interference suppression challenging.

by Peter Schinzel, Andreas Pfaff, Thomas Dippon, Thomas Fischer, and Allan R. Armstrong

This article discusses the internal design of the HP 8133A 3-GHz pulse generator (see article, page 52), including the timing board, the width board and output amplifier, and the EMC design.

Timing Board

The timing board generates the basic pulse train of the instrument with programmable pulse repetition period (or equivalently, frequency). Its inputs are the external clock signal, if any, and the STROBE signal from the pulse/data board, if installed. Its outputs are a delayed or undelayed pulse output to the width board (depending on the instrument option), a dividable pulse output to the pulse/data board, a trigger output to the front panel, and a divided clock signal to the processor board. The main goals were timing resolution of 1 ps for period and delay, jitter less than 5 ps rms, and frequency accuracy better than 0.5%. Fig. 1 is a block diagram of the timing board.

Period Generation

Because very low jitter and high stability were major goals, a YIG oscillator is used as the main timing source of the instrument. The YIG oscillator frequency is programmed between 2 GHz and 4 GHz by a 12-bit DAC to achieve a period resolution of 1 ps. To achieve low jitter a special reference ground plane for the circuit that drives the YIG is integrated on the printed circuit board to avoid noise and cross talk between the YIG driver and other components. To reduce the noise of the YIG driver further, the bandwidth of the driver components is reduced almost to the fastest programming speed.

When the instrument runs below 2 GHz the YIG oscillator signal is divided by two by a static GaAs divider. For output frequencies between 2 GHz and 3 GHz the divider is bypassed by two GaAs switches (switch 1 and switch 2 in Fig. 1). This additional division by two is necessary because the programmable frequency divider used to generate frequencies down to 33 MHz operates at input frequencies only up to 3 GHz while the YIG oscillator frequency can be as high as 4 GHz.

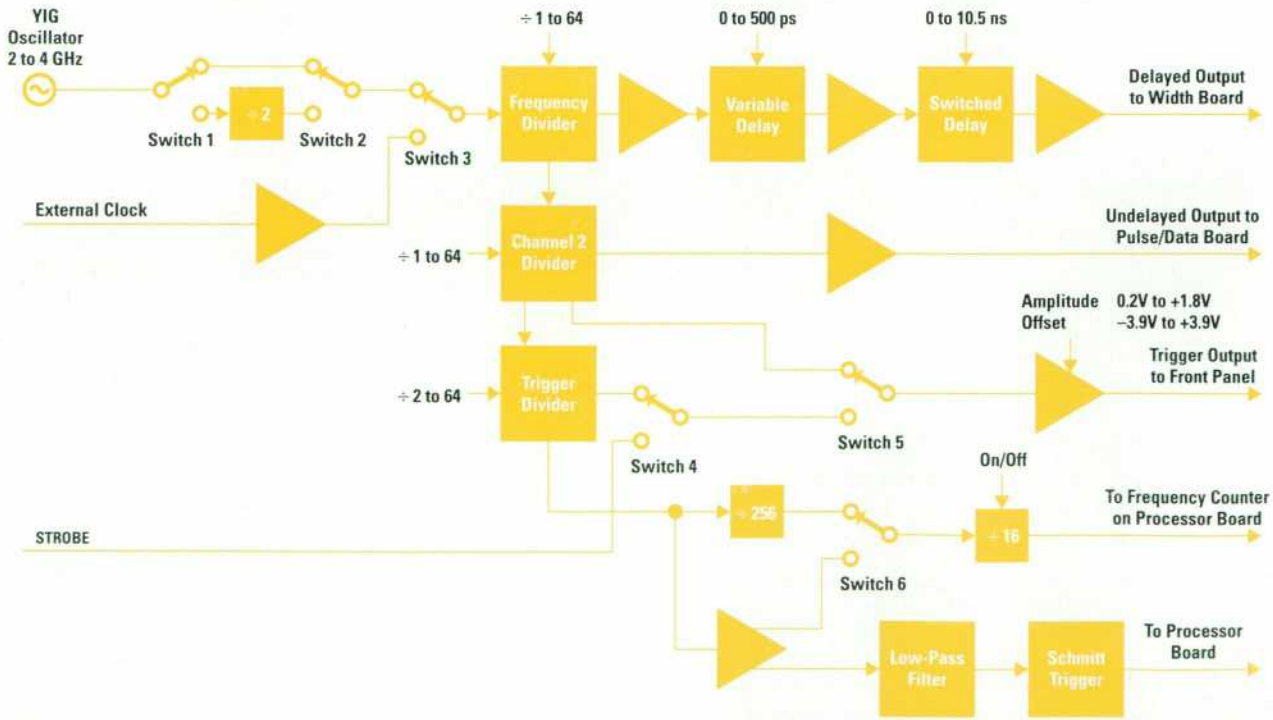


Fig. 1. Block diagram of the timing board of the HP 8133A pulse generator.

Cooling of the Frequency Divider IC

For the timing board of the HP 8133A pulse generator, surface mount technology is used to attach the components to the board. The frequency divider IC has a special package. (The multiplexer on the pulse/data board has a similar package.) It is a cylindrical ceramic housing with 24 radial leads. The leads are located around the cylinder halfway between the top and the bottom (see Fig. 1). Therefore, a hole is required in the printed circuit board for the bottom half.

The power consumption of the IC is 2.5W. Because of the small size of the package (6.7 mm diameter), efficient cooling with a heat sink is necessary. However, it is not possible to glue a heat sink on the bottom half of the package where the chip is mounted because the leads could be damaged by shock or vibration, and the IC would not be exchangeable because the leads would be above the board and the heat sink below the board.

Another problem is the printed circuit board. The thickness of the board is 1.6 mm and the tolerance is ±10%. The solder process also has a tolerance in the thickness of the solder area. These tolerances have to be accommodated by the heat sink design and the mounting of the components.

The solution we used in the HP 8133A pulse generator is shown in Fig. 1. The prescaler IC is soldered to the surface of the printed circuit board. Above the IC package a spring is mounted. The spring is a sheet-metal part made of spring-tempered bronze (Cu-Sn alloy) and has a thickness of 0.6 mm. A spacer is pressed into the spring. Together, this spacer and the package above the printed circuit board are higher than the two spacers beside the IC, so the spring is always slightly preloaded. Between the spacer on the spring and the top side of the package is a thermally conductive foil that isolates the package from the metal parts to prevent electrical coupling.

Under the printed circuit board is a milled heat sink with two threaded studs pressed into it. The spring and the spacers are mounted on these studs. The next step in mounting the assembly is to screw the M6 screw into the heat sink. The screw is brass for good thermal conductivity and has a flat point. The mounting has to be done with the minimum possible torque (optimal would be zero torque) to avoid breaking the ceramic package. Between the flat end of the screw and the bottom side of the package is a thermally conductive foil like the one on the top.

The last step in mounting the assembly is to mount the printed circuit board on the board shield. The heat sink is screwed to the board shield with M3 screws and M3 plastic shoulder washers for electrical isolation. The insulator sheet between the board shield and the heat sink is a thermally conductive foil that provides both thermal coupling and electrical isolation. All of the thermally conductive foils are self-adhesive for simple mounting.

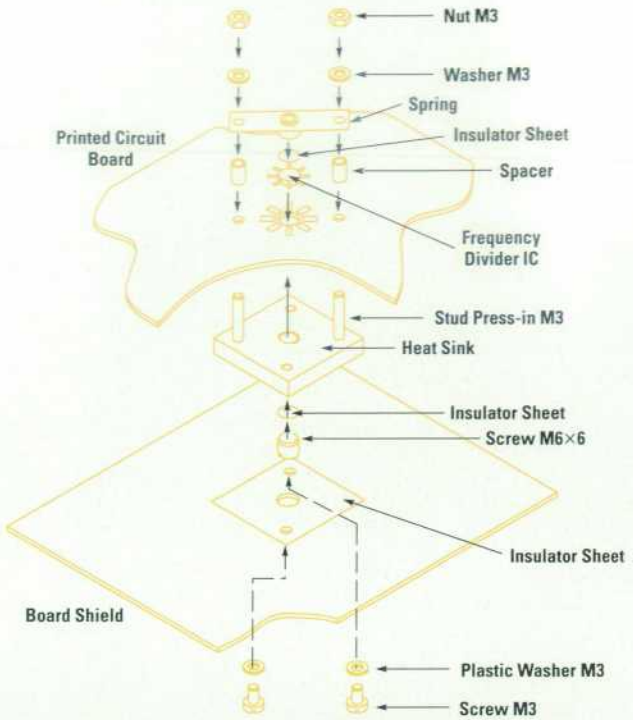


Fig. 1. Components for mounting and cooling the frequency divider IC on the timing board of the HP 8133A pulse generator.

The board shield and the printed circuit board are also joined with additional spacers and other heat sinks. The board shield has different functions: it is used as a heat sink, as protection against board damage, as an electrical shield, and as a guide for the board in the cardcage.

Thomas Fischer
Development Engineer
Böblingen Instrument Division

The programmable frequency divider is a bipolar ASIC. Its key specifications are:

- Input frequency range: dc to 3 GHz
- Division factor: 1, 2, 4, 8, 16, 32, or 64
- Output levels: ECL, differential
- Input sensitivity: 200 mV pp single-ended, 100 mV pp differential.

After the frequency divider, the signal is split into a delayed path and an undelayed path. The delayed signal goes through a delay block, while the undelayed signal goes through a second programmable divider that divides the pulse/data output by 1, 2, 4, 8, 16, 32, or 64. Another programmable divider is integrated in the trigger path so the trigger can also be divided by 1, 2, 4, 8, 16, 32, or 64. This feature makes it possible to trigger an oscilloscope when the instrument is running at 3 GHz, even if the trigger circuit of the oscilloscope does not work up to 3 GHz. Dividing the trigger output when the instrument is running at high frequencies generally improves the trigger performance of the oscilloscope. Jitter is reduced and edge placement measurements can be made more accurately.

If the instrument is running in the data mode the trigger is synchronized with the data on the pulse/data board (see article, page 56). In this mode the STROBE signal from the data board is switched into the trigger path by switches 4 and 5.

The trigger output amplifier is a shaping amplifier, which is described in detail later. The trigger output is programmable in amplitude (200 mV to 1.8V) in a –4V-to-+4V window. The programming resolution is 10 mV.

The signals from the timing board to the output boards are fed to the rear panel. There the signals are conducted out of the instrument through exchangeable semirigid cable bows. This makes it possible to change the electrical length and therefore the delay between the output channels and the trigger output. The advantage of deskewing the channels at this point in the signal path is that no additional cable that might distort the output pulse is necessary at the outputs of the instrument.

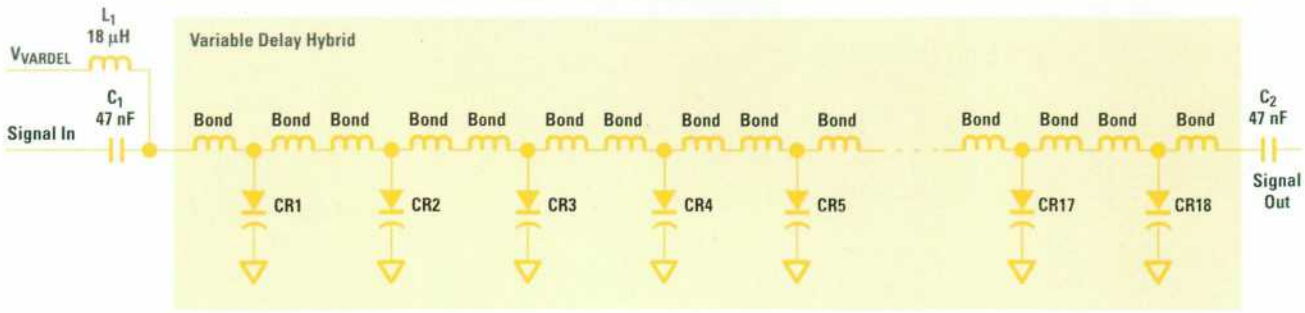


Fig. 2. Schematic diagram of the variable delay hybrid.

Delay

The delay block consists of an input shaper amplifier, a variable delay followed by a second shaper amplifier, a switched delay, and an output shaper amplifier.

Variable Delay. The variable delay circuit is a thick-film hybrid. Fig. 2 shows its schematic diagram and Fig. 3 is a photograph of the hybrid. Diodes CR1 to CR18 are varactor diodes. By changing their reverse voltage V_{VARDEL}, the capacitance of each diode can be tuned between 0.6 pF and 2.1 pF. The voltage V_{VARDEL} is programmable between –1.5V and –36V. Fig. 4 shows the delay shift versus the bias voltage at 50 MHz. To get high resolution over the entire range a 1/x DAC is used. The bias voltage V_{VARDEL} is equal to:

$$V_{VARDEL} = V_1 + V_2/(\text{DAC Value}),$$

where V_1 and V_2 are constants.

To achieve a variable delay without distorting the signal on the hybrid the capacitance of the diodes and the inductance of the bond wires must both be varied. Because the bond-wire inductance cannot be varied the signal is distorted and the delay is a function of frequency. Therefore, a calibration of the delay as a function of the DAC value and frequency is necessary. Fig. 5 shows the delay of the hybrid as a function of frequency before calibration and Fig. 6 shows the delay after calibration. With the calibration, which runs 100% automatically, a delay accuracy of ±15 ps is achieved over the full delay and frequency ranges. In the shaping amplifier that follows the variable delay hybrid the signal is refreshed (transition times better that 100 ps) and amplified to 2V pp.

Switched Delay. The schematic diagram of the switched delay circuit is shown in Fig. 7. Fig. 8 shows the layout of the switched delay hybrid. To achieve low-loss microstrip lines, the hybrid is produced in thin-film technology. The different delays are achieved by switching between microstrip lines of different lengths. Resistors R_1 to R_{12} compensate for the on-resistance of the GaAs switches. Without these resistors the signal would be distorted by the reflections caused by the on-resistance. The advantage of using GaAs switches instead of diodes is that no biasing is necessary in the high-frequency path. The disadvantage is that these switches have a higher insertion loss, but this is not critical because the refresh amplifier following the switched delay hybrid has a higher gain (better than 20 dB) over the full frequency range of the instrument than the loss of the switched delay hybrid at maximum delay (typically 16 dB). The nominal electrical lengths of the delay lines are: $\tau_1 = 222$ ps, $\tau_2 = 401$ ps, $\tau_3 = 748$ ps, $\tau_4 = 1421$ ps, $\tau_5 = 2727$ ps, and $\tau_6 = 5261$ ps. In general:

$$\tau_{n+1} = 1.94\tau_n - 30 \text{ ps}.$$

Thus, by adding a variable delay between 0 and 222 ps to the switched delay, any delay between 0 and 11 ns can be achieved with a resolution determined only by the variable delay, even if there is a mismatch of ±3% ±15 ps between the different switched delay lines. Of the total 11-ns delay capability of the instrument, 10 ns is used to delay one channel of the instrument by ±5 ns with respect to the other channel, and 1 ns is used to deskew the two channels of the instrument, that is, to compensate for the mismatch in the transmission times of the output boards.

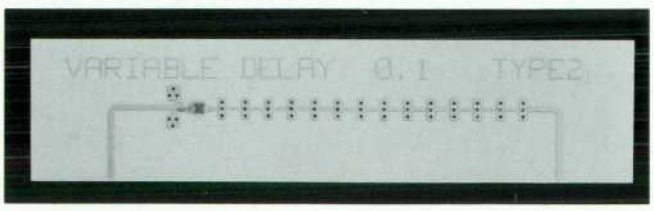


Fig. 3. Variable delay hybrid.

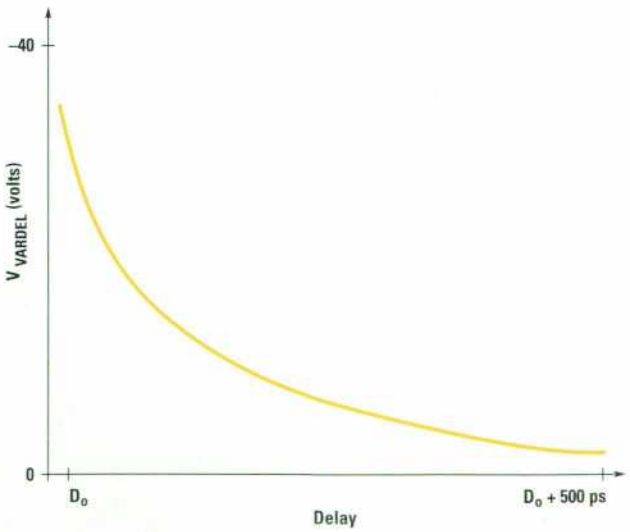


Fig. 4. Delay of the variable delay hybrid as a function of bias voltage.

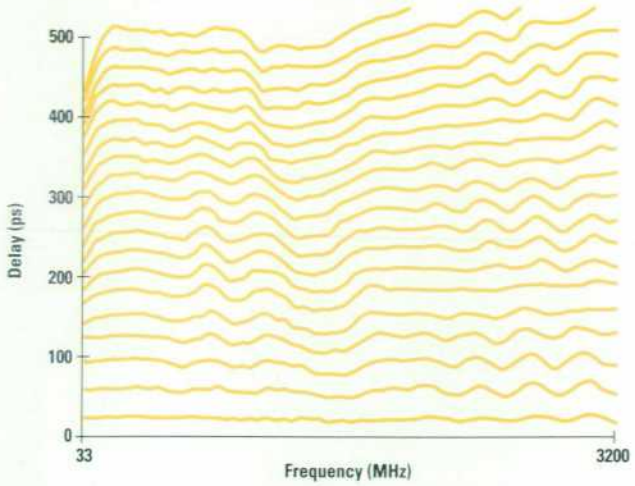


Fig. 5. Delay of the variable delay hybrid as a function of DAC value and frequency before calibration.

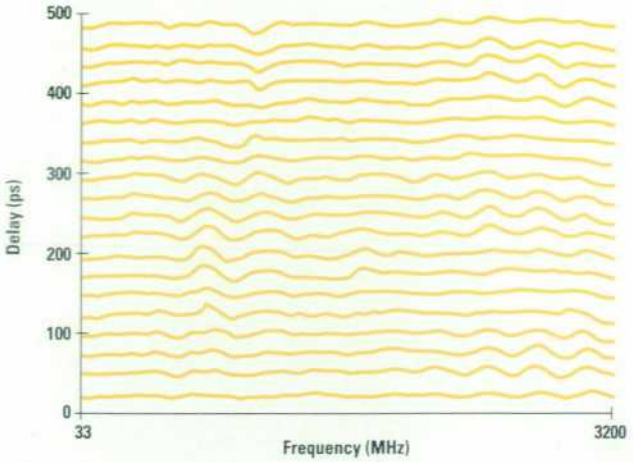


Fig. 6. Delay of the variable delay hybrid as a function of DAC value and frequency after calibration.

Shaping Amplifier

The shaping amplifier is one of the key building blocks of the instrument. The challenge of the design was to develop a discrete amplifier with a frequency range of 10 MHz to 3.5 GHz, an output amplitude of 0.2V to 2V pp (adjustable), and transition times better than 100 ps (60-ps typical 20%-to-80% transition times are achieved). The amplifier was to be built only with autoload surface mount components on a standard FR-4 HP printed circuit board.

The amplifier is a four-stage, ac coupled GaAs FET amplifier. The GaAs FETs are mainly used in TV satellite receivers and therefore they are low-cost components available on reels for autoloading. Fig. 9 shows the schematic of one stage of the shaping amplifier. U1 determines the dc current I_{FET} through $R_5 \| R_6$, that is, through L_1, L_{13}, and the FET Q1, by controlling the gate source voltage of Q1. R_{V1} is used to adjust the current I_{FET}. At high frequencies $C_5 \| C_6$ is a short circuit to ground and the inductors L_{13} and L_1 have very high impedance. This means that the input of each stage is terminated in 50 ohms within the frequency range of the amplifier. The resistor at the gate of the FET prevents the amplifier from ringing or oscillating. Diodes CR1 to CR3 protect the GaAs FET from damage when the power is switched on or off. They also protect the FET if there is a malfunction in the biasing circuit. This is very critical, because exceeding the maximum ratings of the FET can cause a malfunction in the instrument weeks later, even after the bias circuit is repaired and the FET is biased properly.

Frequency Measurement

To make it possible to offer duty cycle, phase, and the same precision of delay and width in both the internal and external modes, a frequency measurement capability is built into the instrument. The key specifications of the frequency measurement capability are:

- Accuracy: better than 0.1%
- Measurement time: 300 ms
- Resolution: 100 kHz
- Frequency range: 2 MHz to 3.3 GHz.

Fig. 1 shows the frequency measurement circuits on the timing board that prepare the signal for the frequency counter, which is on the microprocessor board.

The main challenge for the frequency measurement was to get a low-cost divider that works from 10 MHz up to 3 GHz. Low-cost dividers for telecom applications operate only from 10 MHz to 2.7 GHz. To solve this problem the trigger divider

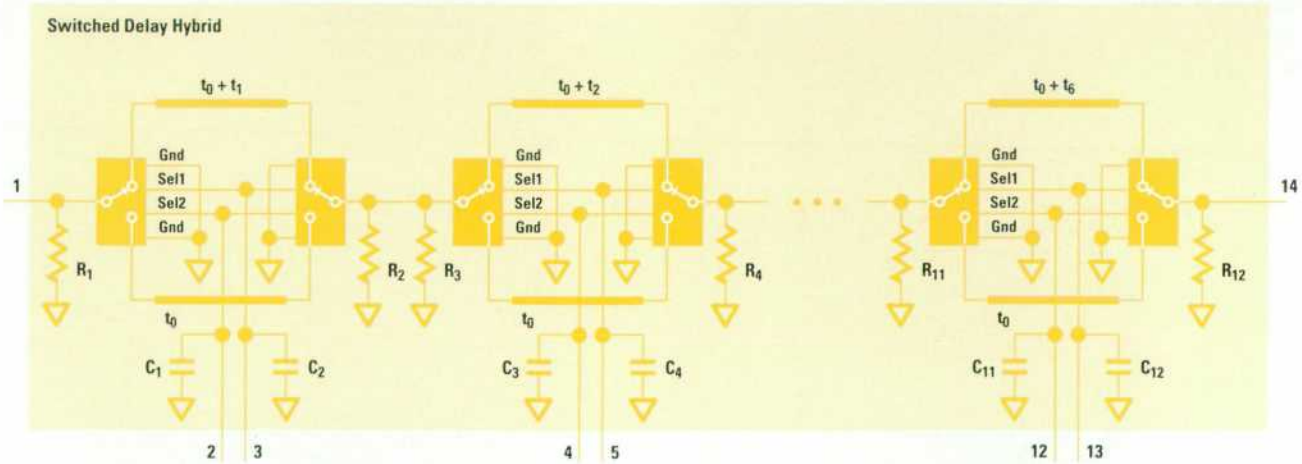


Fig. 7. Schematic diagram of the switched delay hybrid.

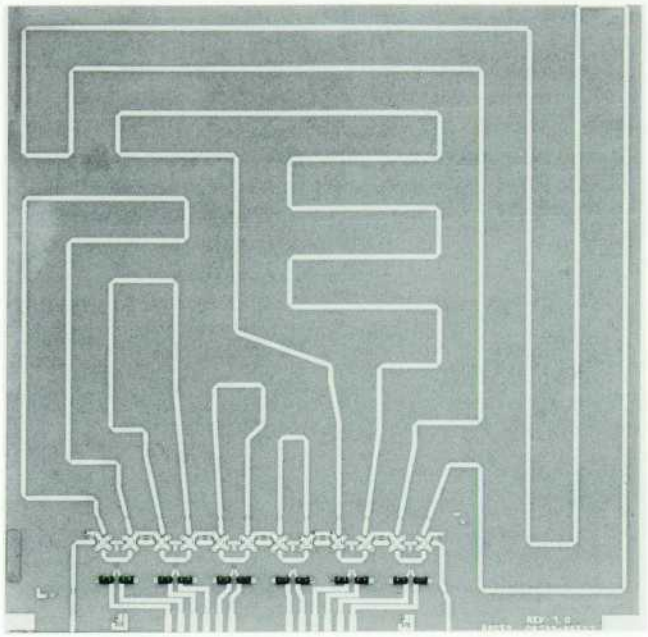


Fig. 8. Photograph of the switched delay hybrid.

is used. When the instrument is running without dividing the trigger, the trigger divider is bypassed and programmed to divide by 64. If the trigger is divided by 2, 4, 8, 16, 32, or 64 the trigger signal goes through the trigger divider. Therefore, at the output of the trigger divider the maximum frequency is 1.5 GHz (instrument running at 3 GHz, trigger divided by 2). This means that a low-cost, low-power frequency divider can be used after the trigger divider.

The frequency measurement must work at frequencies down to 10 MHz at the external input, so the frequency counter must work in a frequency range from about 150 kHz (10 MHz divided by 64) to 1.5 GHz (3 GHz divided by 2). Therefore, the signal after the trigger divider goes through a static divider and is divided by 256. In a parallel path, a differential amplifier converts the signal to TTL levels. One output of the amplifier feeds a low-pass filter with a cutoff frequency of 15 MHz. A Schmitt trigger circuit detects the amplitude of the filter output and tells the microprocessor whether the frequency is below or above 15 MHz. The microprocessor uses the Schmitt trigger output to control switch 6. For frequencies below 15 MHz the signal goes through the amplifier, bypassing the divide-by-256 divider, while for frequencies above 15 MHz the signal is first divided by 256 and then converted to TTL levels.* The resulting signal is divided by 16 in a standard TTL divider and sent to the frequency counter on the microprocessor board, which is an ASIC leveraged from the HP 8153A lightwave multimeter.

* The bypass is necessary because the divide-by-256 divider does not operate below 10 MHz.

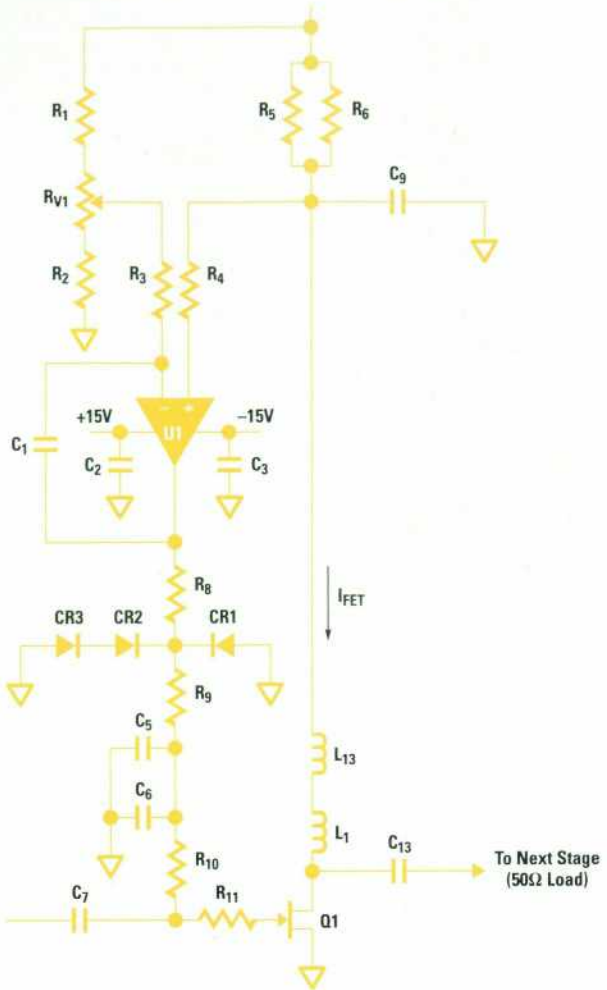


Fig. 9. Schematic diagram of one stage of the shaping amplifier.

Width Board

The delayed output signal of the timing board, a single-ended square wave, is the input signal of the width board. The width board generates the variable pulse width and contains the output amplifier.

Variable Pulse Width Generation

As shown in the block diagram of the width board, Fig. 10, the input signal is split into two signals. One of these signals

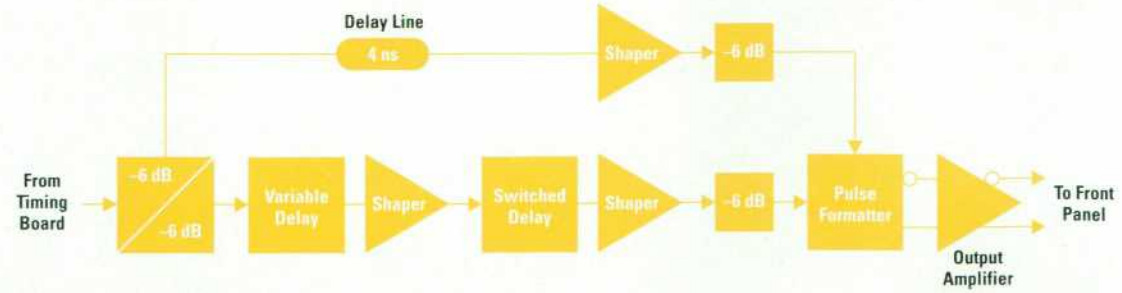


Fig. 10. Block diagram of the width board.

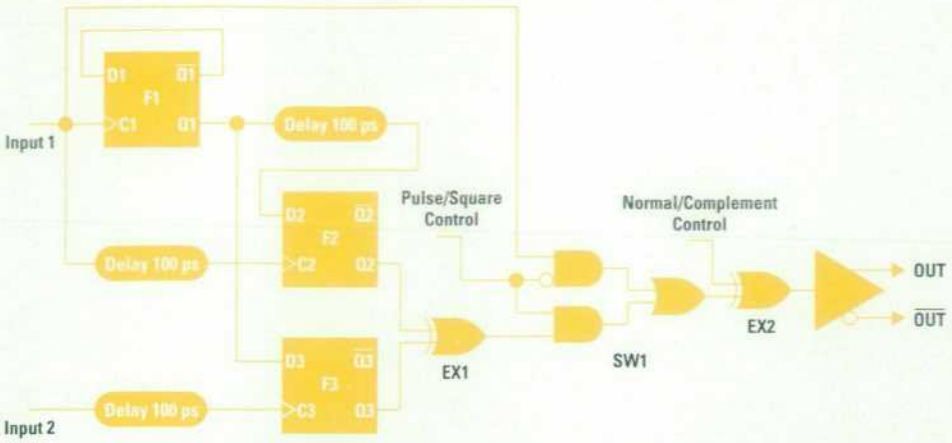


Fig. 11. Block diagram of the pulse formatter on the width board.

is delayed by about 4 ns with a fixed delay line (semirigid cable). The other signal goes through a delay block that consists of two different hybrids—a variable delay hybrid and a switched delay hybrid—and two shaper amplifiers. The two hybrids and the shaping amplifier are described earlier in this article.

Thus, two signals with the same frequency are generated: one with a fixed delay of 4 ns and the other with a programmable delay of 4 ns to 14 ns (4 ns is the minimum delay of the delay block). In the pulse mode, the delay of the second signal is limited to a maximum of 4 ns plus the actual period value by the instrument software. According to the instrument specification, this delay can only be programmed from 4.15 ns to 3.85 ns plus the actual period value because the output pulse width is equal to this delay value minus the delay of the first signal (which is 4 ns, as mentioned above) and the specified minimum pulse width is 150 ps.

Pulse Formatter

The two square wave, single-ended signals are the input signals of the pulse formatter. The pulse formatter is a custom bipolar IC designed in HP's HP1X process. It is packaged on a thick-film hybrid.

The pulse formatter (Fig. 11) is basically a high-speed EXOR (exclusive-OR) gate (EX1 in Fig. 11). If an EXOR gate gets two square wave signals of the same frequency that are delayed with respect to each other, it generates an output signal with a pulse width equal to the delay between the two input signals but with twice the frequency. Therefore, the frequency of the input signals is divided by two by the on-chip flip-flop F1.

An additional EXOR gate (EX2) enables the user to choose between normal mode and complement mode. The user can also switch the pulse formatter between the pulse mode and the square wave mode. In the square wave mode, the input signal with the programmable delay is connected to the output of the pulse formatter. Thus, the user can switch off the width capability of the instrument and have an additional 0-to-10-ns delay capability instead. This is especially valuable in a standard instrument, which does *not* have the delay block on the timing board.

If the frequencies of both input signals were divided separately, using two flip-flops, the pulse formatter could be either in the normal mode or in the complement mode after powering up the instrument or changing the frequency, delay, or width range. This would depend on the initial state of the two flip-flops and could not be controlled by the microprocessor. This is why only one divide-by-two flip-flop (F1) is used here. The output of F1 is connected to the inputs of flip-flops F2 and F3, which are clocked by the input signals. As a result, the pulse formatter always starts running in the normal mode (self-initialization).

The pulse formatter works from dc to 3.5 GHz and can produce pulse widths as narrow as 120 ps or less. It has a differential output with a swing of 500 mV and transition times of about 80 ps (typical). The high level of the output signal is 0V. The input signals are ac coupled and must have a swing of 800 mV. The pulse formatter requires –5.1V and +1.0V supply voltages and dissipates 2.5W worst-case. Therefore, the backside of the thick-film hybrid is mounted on a heat sink made of solid aluminum.

Output Amplifier

The output amplifier is driven by the differential outputs of the pulse formatter IC. The amplifier consists of two identical GaAs ICs designed in HP's MMIC-B process and packaged in a thick-film hybrid.

The MMIC-B process is a depletion-mode MESFET process with a double-recessed[1] 0.35-μm direct-write E-beam gate.[2] It achieves an f_T of 23 GHz with a pinchoff of –1.3V. The active layer is grown with molecular beam epitaxy and features a low-temperature buffer[3] to eliminate backgating.[4,5]

Each IC consists of two differential amplifier stages and one 4.5V level shifter with a source follower between them, as shown in Fig. 12. The layout of the hybrid is shown in Fig. 13, and Fig. 14 is a photograph of the hybrid. The first IC drives the second IC through a level shifter. The result is a four-stage dc-coupled differential amplifier in which stages one and two are provided by the first IC and stages three and four are provided by the second IC.

The GaAs ICs require five voltage and four current sources each. These sources are located on the printed circuit board. Elastomeric contacts are used to connect the power supplies from the printed circuit board to the thick-film hybrid.

Low-impedance bypassing of the voltage supplies is essential to minimize overshoot and prevent oscillation in the source followers of the amplifier ICs. Bypass is provided at low frequencies by surface mount capacitors on the printed

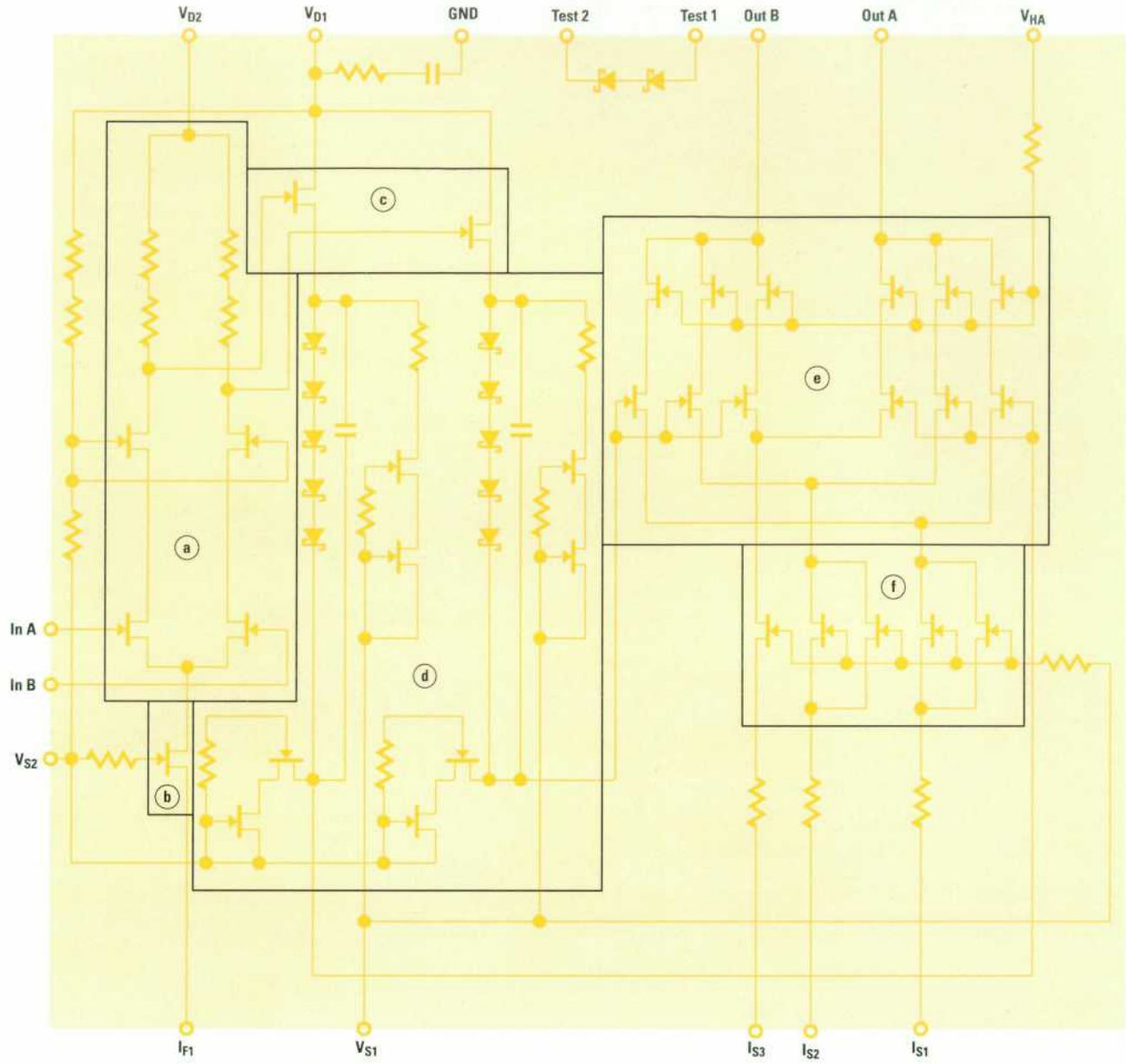


Fig. 12. Schematic diagram of one output amplifier IC. (a) First differential amplifier with cascode. (b) Current source buffer. (c) Source followers. (d) Level shifter. (e) Second differential amplifier with cascode. (f) Current source buffer.

circuit board. High-frequency bypass is provided by 500-pF capacitors printed on the hybrid using a paste that has a relative dielectric constant $\epsilon_R \cong 1000$. If surface mount capacitors had been used on the hybrid instead of the dielectric paste, the capacitors would have been so large that they could not have been located close to the ICs. This would have increased bond-wire inductances and caused problems with ringing and oscillation.

Elastomeric contacts are also used to connect the input RF signal from the pulse formatter, but are too reflective at high frequencies to be used for the output signal. Instead, SMA connectors are used. They are connected to the hybrid with low-inductance ribbon bonds.

Both GaAs ICs require a high level of about –11V at the RF inputs. The output signals of the pulse formatter and the first amplifier IC have a high level of 0V. Therefore, two level shifts of 11V are required. Dc level shift is accomplished by an operational amplifier circuit on the printed circuit board. The levels are sensed with resistors located directly on the RF striplines so as not to add reflective stubs. The high-frequency signal is passed by 10-nF coupling capacitors in parallel with the level shifting circuit.

The 50-ohm termination resistors at the outputs of both ICs are located extremely close to the ICs to avoid reflections from stubs. No reverse termination is provided at the input of the second IC because 11V across the resistors would cause a very large power dissipation. Because the reverse termination of the output of the first amplifier is so good, the effect of this single reflection is minor.

A good RF termination is essential at the input of the first IC. Because the bias at the input of the amplifier IC is –11V, simple termination results in a very large power dissipation.

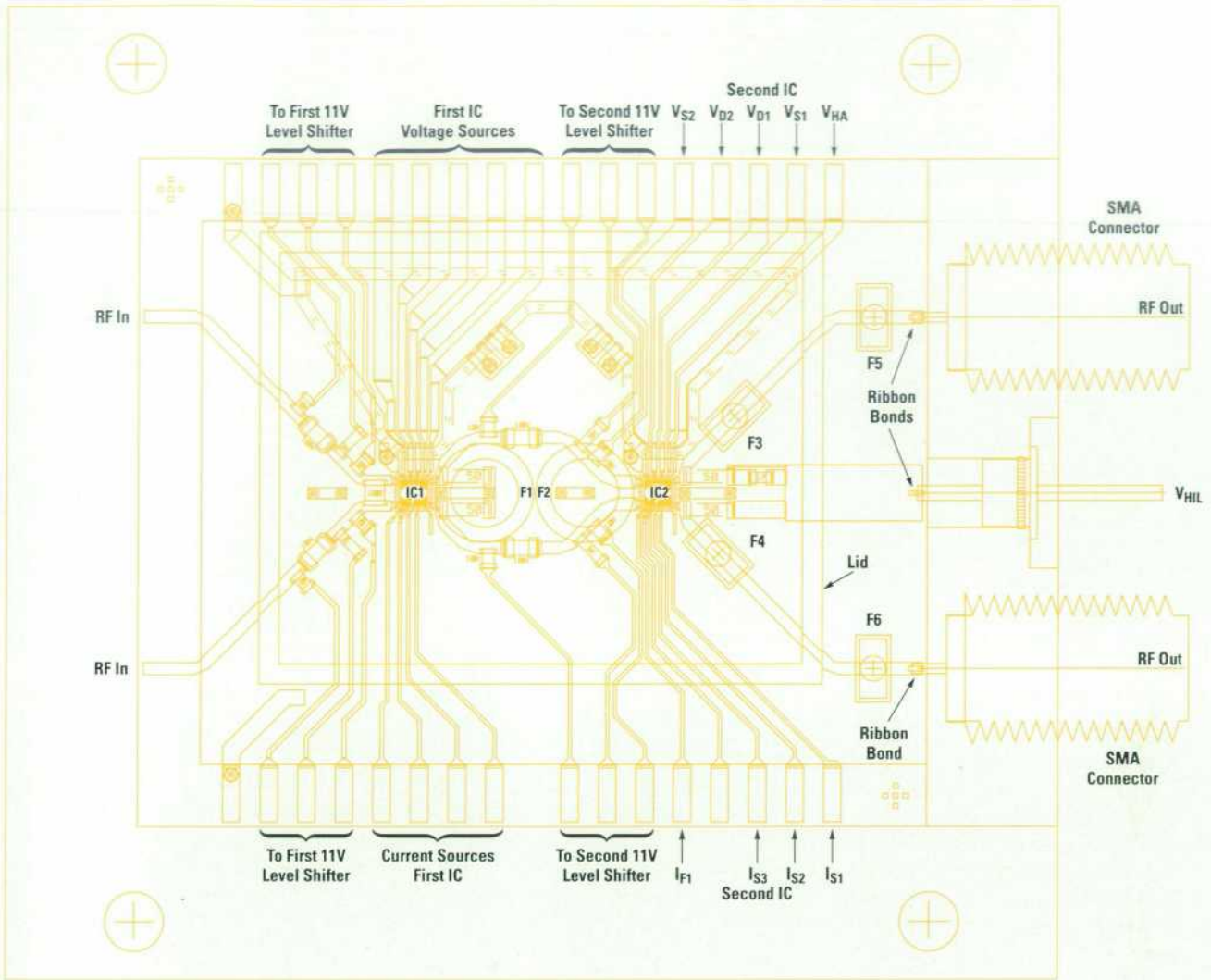


Fig. 13. Layout of the output amplifier hybrid. IC1 and IC2 are identical output amplifier ICs.

By providing a dc termination on the printed circuit board (at a lower dc bias) and an ac termination on the hybrid (a 100-ohm resistor between the true and complement inputs), a good RF termination is provided without excessive power dissipation.

Ferrites F3 and F4 (under the lid) and optional ferrites F5 and F6 (outside the lid) reduce high-frequency ringing (at about 10 GHz) on the RF outputs. F1 and F2 reduce reflections between the two ICs caused by the impedance mismatch of the output termination of the first amplifier IC at about 10 GHz.

The total power dissipation of both ICs is 4.5W worst-case. To maintain low die temperatures, the backside of the thick-film hybrid is mounted on a heat sink made of solid aluminum. The heat sink is also used as a fixture for the SMA connectors.

The high level of the output signal can be adjusted with V_{HIL} (see Fig. 13) from –2V to +4V. For high-level values greater than +1V, all voltage sources of the second IC are increased linearly with V_{HIL}.

The output amplitude is adjusted from 0.1V to 3.0V peak-to-peak with the currents I_{S1}, I_{S2}, and I_{S3} of the second amplifier IC (see Fig. 12). For amplitudes below 1.0V, I_{S1} is switched off, and for amplitudes below 0.33V, I_{S1} and I_{S2} are switched off. The output devices are scaled 6:2:1 in width so that FET current densities are similar in each range.

The output amplitude range is segmented to reduce overshoot. An extensive calibration optimizes transition time and ringing over the specified amplitude, temperature, and frequency ranges. The parameters adjusted in this optimization are the operating current in the third stage of the four-stage differential amplifier (I_{F1}), the high level of the third stage (V_{D2}), and the cascode bias voltage of the fourth stage (V_{HA}).

The output amplifier generates pulses with typical transition times of 50 to 60 ps (10% to 90%). The minimum pulse width specification is 150 ps, but pulses with a width of less than 120 ps are possible. Overshoot and ringing are specified to be less than 15% + 20 mV; typical values are 5% to 10%.

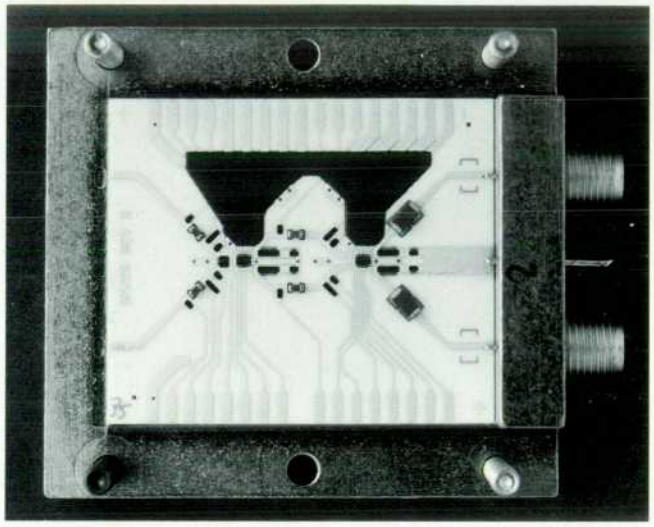


Fig. 14. Photograph of the output amplifier hybrid.

Wafer Test of the Output Amplifier

The amplifier ICs are tested on-wafer using an HP 8131A pulse generator and an HP 54121T digitizing oscilloscope. The ICs are probed with a thin-film probe card. Dc functional measurements and 20%-to-80% output rise time measurements are made at a number of output amplitude settings.

High-speed on-wafer testing is generally limited to devices having a limited number of RF signals and dc biases. Probe card technologies typically offer either excellent RF performance at low complexity or high complexity and mediocre RF performance. Because of the fast transition times that must be verified in wafer test, the large number of dc connections, and the requirement that power supplies be well-terminated to control overshoot and ringing in the output signal, no commercially available probe technology was deemed capable of meeting our needs.

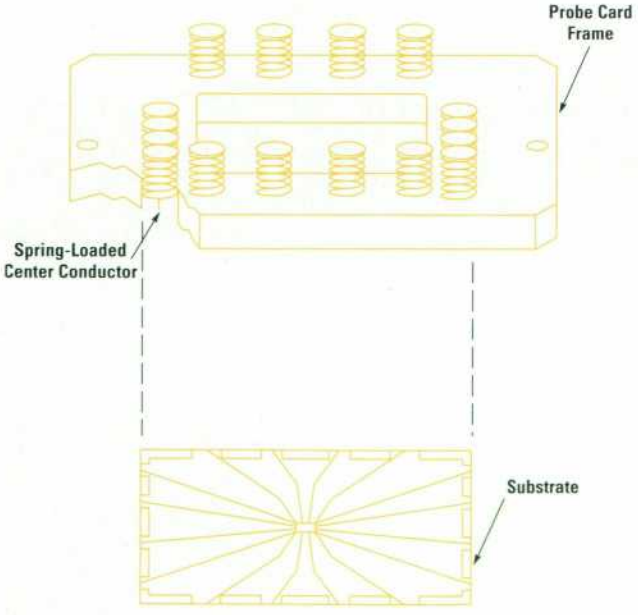


Fig. 15. Exploded view of the probe card for wafer-testing the output amplifier ICs.

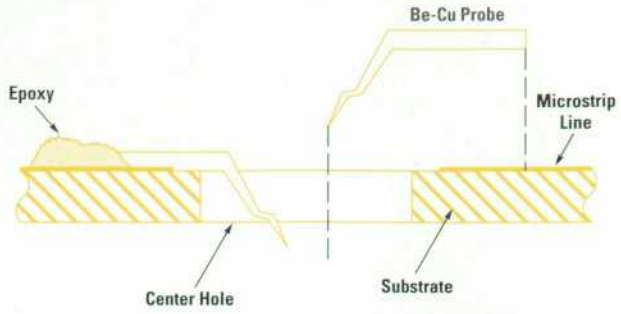


Fig. 16. Cross section of the probe card showing the probe mounting.

The probe card technology chosen is a custom thin-film probe card[8] in which short (1-mm) Be-Cu probes are epoxy attached to 50-ohm microstrip lines in a hole in the center of a sapphire substrate. The substrate is mounted in a machined metal frame into which are mounted female SMA connectors. The SMA connectors are spring-connected to coax-to-microstrip launches on the substrate, allowing RF and dc connection to each of the probes. An exploded view of the probe card assembly is shown in Fig. 15, and detail of the probe attachment is shown in Fig. 16.

Each dc source is connected to the probe card via a bias tee as shown in Fig. 17. Ac termination is supplied by connecting a 50-ohm load to the RF port of the bias tee. The probe inductance, modeled as a single lumped element, is approximately 0.7 nH. Thus, the RF termination seen by the dc pad of the chip is approximately 50 ohms in series with 0.7 nH.

An alternate solution to provide bypassing would be to use surface mount capacitors on the substrate to bypass the power supply lines to a ground provided by a plated via hole. This would provide near zero impedance at low frequencies. Unfortunately, the physical size of surface mount capacitors would necessitate placing them some distance away from the probes and the short circuit would be rotated around the Smith chart and would actually appear open at about 3 GHz. Thus, the amplifier would tend to ring or even oscillate.

Because the sapphire substrate cannot be attached to a heat sink without mechanical interference with the wafer probing operation, power dissipation of resistors on the substrate is limited. Because shorted ICs or an erroneous test setup may force arbitrary abusive currents through the output load resistors, and a damaged thin-film resistor would necessitate an expensive replacement of the substrate and probe assembly, we chose to provide the reverse termination resistor for both outputs off the probe card.

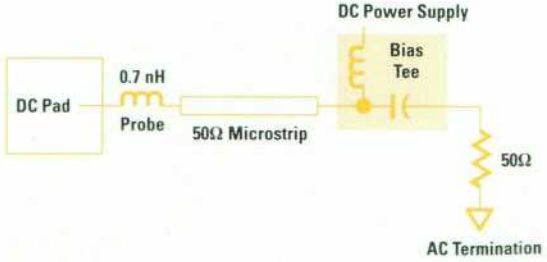


Fig. 17. Probe card dc bias technique.

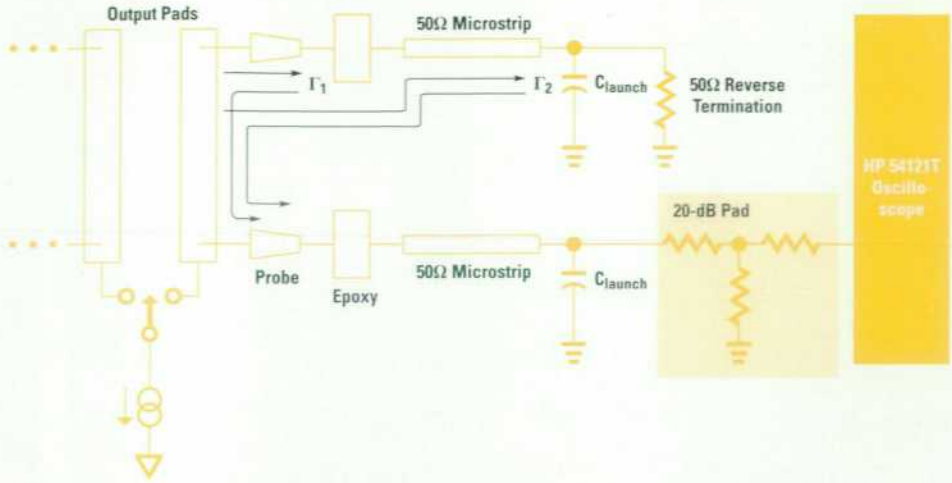


Fig. 18. Model of the probe card output transmission lines showing reflections from discontinuities.

A model of the output probe circuit is shown in Fig. 18. The same circuit is used for both true and complement outputs of the IC. The IC does not contain its own reverse termination resistor; its output is a switched current source and appears to be a high impedance. The output requires a 50-ohm reverse termination as well as a 50-ohm customer load. These loads are provided at the ends of two transmission lines. The reverse termination is a replaceable 50-ohm SMA load (HP 0960-0053); the customer load is an HP 54121T oscilloscope with a 20-dB pad (HP 33340C) for input protection. Each output pad has two probes; one for each transmission line.

The probe card has two known discontinuities; a simple capacitive discontinuity at the coax-to-microstrip transition and a more complex discontinuity in the area of the probes and probe-to-microstrip transition consisting of tapered high-impedance transmission lines for the probes and a low-impedance line in the area of the epoxy. Reflections from these discontinuities in the reverse termination arm cause a two-step undershoot in the step response as shown in Fig. 19. The longer undershoot is caused by the coax launch; the shorter undershoot is caused by the epoxy and probes. The total amplitude of these two undershoots is approximately 8% to 10% of the input voltage step.

Because a 10%-to-90% transition time measurement requires a repeatable crossing of the 90% point, small variations in ringing from die to die make a 10%-to-90% transition time measurement undependable. Reflections are far better

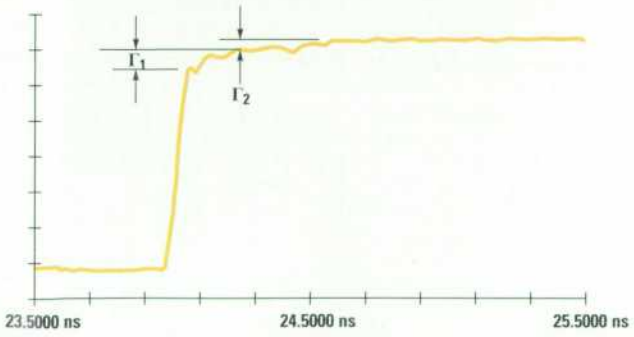


Fig. 19. Probe card output waveform showing undershoots caused by reflections.

controlled in the instrument application, but this probe technology allows us to make a 20%-to-80% transition time measurement that can be correlated with the 10%-to-90% transition time in the instrument.

High-speed pulse measurements on-wafer allow the test to reject dice with slow transition times or incomplete switching problems before they are assembled into hybrids. This reduces rework and lowers manufacturing cost.

Calibration

All adjustable signal parameters of the instrument (frequency, width, delay, amplitude, etc.) are controlled by analog voltages generated by digital-to-analog converters (DACs). The same is true for the voltages and currents that are responsible for signal performance in the amplifier stages.

The relationships between these control voltages and the corresponding signal parameters are usually nonlinear and temperature dependent. They may also depend on other signal parameters (typically frequency) and vary from board to board. The advantage of using DACs for generating these control voltages instead of variable resistors is that the microprocessor can compensate for these influences.

The ideal solution would be to measure and store all parameter values at all possible DAC settings and temperatures. In practice this is impossible because of the huge amount of data and the time it would take to measure it.

In the HP 8133A, all DAC settings are calibrated as functions of the corresponding signal parameters, and most of them are also calibrated as functions of frequency:

$$DAC = f(parameter, frequency).$$

To keep the amount of data manageable and the calibration time and parameter setting time reasonable, a two-dimensional linear interpolation algorithm is used for all calibrations.

A linear interpolation algorithm approximates the relation between the independent value (programmed signal parameter) and the dependent value (e.g., the DAC value) with pieces of straight lines. In the two-dimensional case there are a number of points of the first independent parameter

(e.g. delay) and another number of points of the second independent parameter (e.g., frequency) which form an array of parameter pairs. For each parameter pair a dependent value is measured and stored. When the instrument is programmed to a certain parameter setting, the dependent value is interpolated between the four adjacent parameter pairs.

Each relation between a signal parameter and the corresponding DAC value is represented by a calibration structure within the instrument. This allows calibration of each parameter independently of the others. An example of a command to load data into one of these structures is:

```
:DIAGnose:CALibration:YIG 2.0E9,888,3.0E9,1969,4.0E9,3049
```

This calibration command for the YIG oscillator defines three interpolation points at 2, 3, and 4 GHz with DAC values of 888, 1969, and 3049, respectively.

Although the RAM on the microprocessor board is battery buffered (and could therefore be used to store the calibration data) we decided to store the data in an EEPROM on the corresponding hardware board and load it into the microprocessor RAM after power-on. We use 8K-byte EEPROMs. The advantage of using EEPROMs is that the calibration data is secure even when the battery is disconnected. It also makes the instrument more easily serviceable, because replacement boards can be calibrated in the factory. The only thing that has to be recalibrated after board exchange is the delay skew between the two output boards.

Temperature Calibration

As mentioned above, most parameters have a temperature dependent variation. For most parameters the temperature dependency can be described with adequate precision as a linear function of the temperature difference between the current temperature and the temperature at calibration time (ΔT). For other parameters the temperature dependency is a linear function of ΔT times the current interpolation value.

To compensate for these two kinds of temperature dependencies, we store an absolute and a relative temperature coefficient together with the ambient temperature at calibration time with each calibration structure. The ambient temperature in the instrument is measured with a temperature sensor on the timing board.

When the instrument is programmed to a certain parameter setting, the result of the interpolation is modified using the following formula:

$$\text{Value}_{new} = \text{Value}_{old} + \text{Value}_{old}\text{TC}_{rel}\,\Delta T + \text{TC}_{abs}\,\Delta T,$$

where Value_{old} is the parameter value as calculated with the interpolation, TC_{rel} and TC_{abs} are the relative and absolute temperature coefficients stored with the calibration structure, and ΔT is the difference between the current temperature and the temperature at calibration time.

Using this method reduces the delay drift, for example, from 150 ps to typically 20 ps over the operating temperature range.

EMC and Mechanical Design

The EMC (electromagnetic compatibility) design of the HP 8133A pulse generator is a combination of mechanical design and electrical design. Since 1992, the RFI standard for HP products has included the new European standard EN55011 (CISPR11) class B limit (similar to the U.S. FCC class B limit). The product had to meet this new standard.

Background

The HP 8133A Pulse Generator is built in an HP System II+ cabinet with a height of 123.6 mm and a depth of 497.8 mm. In this mainframe a main chassis is mounted to carry the power supply and above it the microprocessor board (see Fig. 20). The cardcage is built into the chassis, and the timing board and one or two output boards are installed in the cardcage.

At the beginning of the project we planned to leverage the mechanical design from the mainframe of the HP 8130A and 8131A pulse generators. However, during the first implementation of the HP 8133A hardware in the HP 8130A/31A mainframe, we found that the radiation was over the limits. What were the reasons for the greater emissions? The HP 8130A/31A mainframe was designed for a 500-MHz maximum frequency and 200-ps transition times. The HP 8133A has a 3-GHz maximum frequency and < 100-ps transition times. This means that the HP 8133A generates more power at high frequencies than the HP 8130A and 8131A, especially above 500 MHz.

Measurements showed that the main sources of radiation are the timing board, the output boards, and the cables to the rear panel and the front panel. For sufficient shielding we had to improve the design of the mainframe.

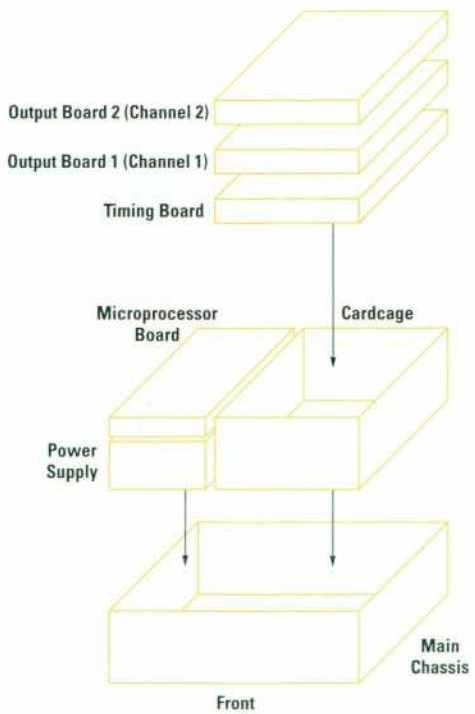


Fig. 20. Main assemblies of the HP 8133A pulse generator.

Improvements

The cardcage with the timing board and the one or two output boards inside is a sheet-metal assembly. We made the following improvements to increase the shielding (see Fig. 21):

- Reduce the diameter of the perforation holes. We found that a hole diameter of 2.4 mm is acceptable. We also found that an expensive honeycomb filter panel for the fan area is not necessary when using a 2.4-mm-diameter perforation pattern.
- Reduce the holes in the corners of the cardcage to the smallest possible size. These holes are necessary for production of the sheet-metal parts whenever two or three bends form a corner of a part.
- Reduce the distance between screws or rivets. A distance of about 25 mm is acceptable for good shielding in the HP 8133A frequency range. If the distance between the screws or rivets is greater than 25 mm, use contact springs to close the seams.
- Avoid seams where ever possible. If there are any seams, they must be closed with contact springs.
- Close all remaining holes that are necessary for manufacturing the parts of the cardcage assembly with contact springs or additional parts.
- Use semirigid cables for the high-frequency interconnections to the front and rear panels. For this purpose the cardcage is extended to the front panel where the SMA connectors are located. The cardcage is also extended to the rear panel in the area of the SMA connectors. Here we use a milled part called the rear support to mount the SMA connectors and semirigid cables.

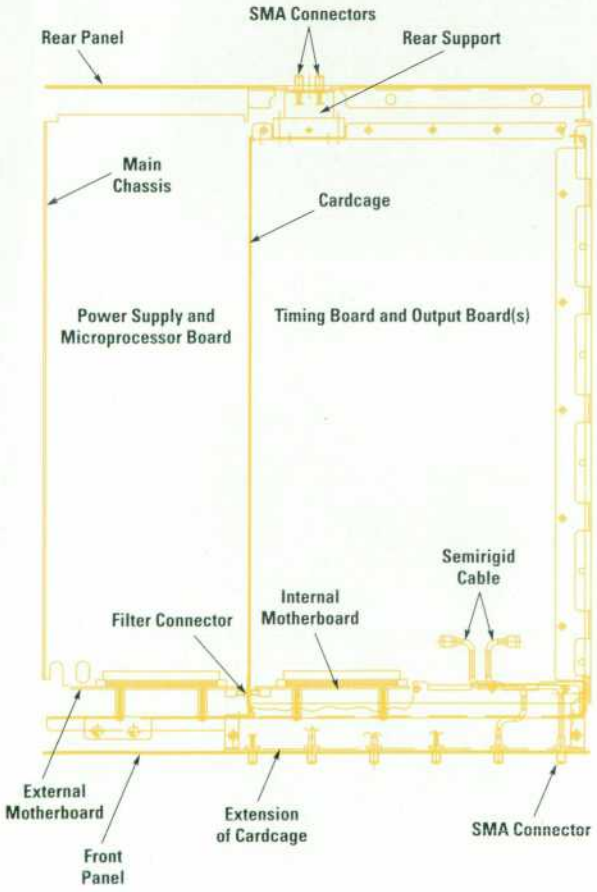


Fig. 21. Top view of the HP 8133A mainframe.

New Ideas

The improvements listed above were not sufficient to reach the RFI limits. Therefore, additional ideas were needed.

We divided the motherboard into two parts. One part is mounted in the cardcage assembly and makes the CPU and power interconnections between the timing board and the output boards. The second part of the motherboard is located on the outside of the cardcage assembly and is connected to the CPU board and the power supply.

Between the two parts of the motherboard is a filter connector with 34 pins, mounted with its housing inside of the cardcage assembly. The feedthrough pins of the filter connector are plugged on one side into the connector of the internal motherboard and on the other side into the external motherboard. The filters are pi filters consisting of two capacitors and an inductor. The result is a high insertion loss for high-frequency energy.

The filter connector must have a low impedance at high frequencies to the ground of the cardcage. The filters are soldered into a filter plate and the filter plate is screwed to the side of the cardcage to provide the required low impedance.

We also looked at filtered D-subminiature connectors instead of the filter connectors, but these D-types are too big for our application.

Because of the capacitive load of the filter connectors, ICs of the ACT family are needed as device bus drivers on all the printed circuit boards in the cardcage and on the microprocessor board.

No additional RFI strips or RFI gaskets are necessary in the cabinet. No additional measures were necessary to avoid RF emissions in the design of the keyboard, microprocessor board, or power supply.

EMC Summary

After many measurements and modifications we were able to meet the RFI standards. The HP 8133A pulse generator has passed the CISPR11 Class B regulation. The measurements required by the EN55011 regulation will be done using production instruments. This regulation allows a transition period until mid-1993 to do the measurements. The EMC design also helped meet the EN50082-1 standard, which prescribes immunity to electrostatic discharge (ESD), radiation, and fast transients.

Acknowledgments

The authors would like to thank Dale Pittock, Mark Mathews, Jim Taylor, and John Tulloch, who helped design and develop tests for the variable delay hybrid, the pulse formatter hybrid, and the output amplifier hybrid. They also thank Bill Murphy, who developed the pulse formatter IC, John Domokos, who supported the design of the variable delay hybrid, and Bob Fisher, who worked on the wafer test for the output amplifier IC. Finally, the authors wish to thank Jochen Weimer, Ralf Meltsch, and Bernd Kegreiss for their work on the

printed circuit board layouts, Heiko Steingraeber, Ernst Bauer, and Martin Schweizer, who made it possible to mount the surface mount parts of the shaper amplifier so close together, Werner Kornmayer, who provided a lot of information about the printed circuit board process, and the people of the metal shop in Böblingen.

References

1. E.H. Wong, B. Keppeler, B. Yeats, N. Fernandez, and D. D'Avanzo, "A Self-Aligned Double Recessed, Sub-Half Micron Gate Process for MMICs Using DUV or E-Beam Lithography," *presented at the Thirteenth State-of-the-Art Program on Compound Semiconductors (SOTAPOCS XIII), Electrochemical Society Meeting*, October 1990.

2. L.G. Studebaker, "Electron-beam written multilevel resist process applied to GaAs FET gate fabrication," *SPIE Electron-Beam, X-Ray, and Ion-Beam Technology: Submicrometer Lithographies VII*, Vol. 1089, 1989, pp. 112-123.

3. N.G. Fernandez, M.J. Lightner, D. D'Avanzo, G. Patterson, and J.E. Turner, "Growth, Processing and Reliability of Low Temperature Buffers for High Performance MMICs," *Proceedings of the Twelfth State-of-the-Art Program on Compound Semiconductors (SOTAPOCS XII), Electrochemical Society Meeting*, May 1990, pp. 121-131.

4. C. Kocot and C.A. Stolte, "Backgating in GaAs MESFETs," *IEEE Transactions on Electron Devices*, Vol. ED-29, July 1982, pp. 1059-1064.

5. C.P. Lee, S.J. Lee, and B.M. Welch, "Carrier Injection and Backgating Effect in GaAs MESFETs", *IEEE Electron Device Letters*, Vol. EDL-3, no. 4, April 1982, pp. 97-98.

6. S.G. Klein and H.J. Wagner, "A 500-MHz Pulse Generator Output Section," *Hewlett-Packard Journal*, Vol. 41, no. 4, August 1990, pp. 79-84.

7. A. Armstrong and H.J. Wagner, "A 5-V P-P 100-ps GaAs Pulse Amplifier IC with Improved Pulse Fidelity," *IEEE Journal of Solid-State Circuits*, Vol. SC-27, no. 10, October 1992.

8. D.P. Hornbuckle, R.L. Van Tuyl, V. Peterson, and R.A. Fisher, "A Microwave Probe System," *Hewlett-Packard RF and Microwave Measurement Symposium*, 1980.

A Multirate Bank of Digital Bandpass Filters for Acoustic Applications

Real-time frequency analyzers have been used for over twenty years for acoustic noise measurements. Recent advances in digital signal processing technology have improved the performance and usefulness of these analyzers. The HP 3569A portable real-time frequency analyzer, for example, makes complex acoustical measurements easier and more affordable than ever before.

by James W. Waite

An acoustics test laboratory will typically have a variety of test equipment for the characterization of product noise. Fundamental to the measurement of noise emissions is a 1/3-octave-band frequency analyzer, but sound level meters and FFT (fast Fourier transform) analyzers may also be present and useful. Digital signal processing (DSP) has for some time allowed custom digital hardware to be used in the design of both octave-based recursive-filter analyzers and instruments relying on the FFT. It has only been recently, however, that DSP chip technology has advanced to the point where it is possible to implement a variety of acoustic measurement algorithms on a single hardware platform and run at the required real-time rates. This paper discusses the development of these DSP algorithms and their implementation in a small (3.2-kg) battery-operated instrument, the HP 3569A portable real-time frequency analyzer (Fig. 1).

Octave-Band versus Narrowband Analysis

When measurements of acoustic sound pressure level are presented as a function of frequency, a logarithmic format is normally chosen. Acousticians prefer to see the data distributed in constant percentage bandwidths, usually octave or 1/3-octave bands, since the auditory perception of sound is logarithmically related to frequency and several regulations require such presentation.

Many modern spectrum analyzers operating in the standard 20-Hz-to-20-kHz frequency range use the fast Fourier transform for quickly translating time-domain signals to the frequency domain spectrum. This approach is widely accepted and offers some unique capabilities for evaluating cross-channel system behavior in the frequency-domain data by means of coherence and frequency response functions. However, the FFT has a severe limitation for use in acoustics. The output of the FFT has a constant bandwidth distribution rather than a constant percentage bandwidth distribution, that is, the frequency scale is linear rather than logarithmic. When 1/3-octave frequency bands are synthesized by performing a weighted sum of linear-resolution FFT bins, the frequency of the lowest band is limited by a lack of resolution at the low-frequency end of the spectrum.

A typical FFT spectrum analyzer has 800 lines of resolution. For a measurement span of 20 kHz, this allows a spectral bin spacing of 25 Hz. Since acoustic measurement standards require at least 5 to 10 FFT bins to synthesize a single 1/3-octave band correctly, the lowest 1/3-octave band that can be accurately synthesized using this data is 500 Hz, five octaves higher than desired. Workarounds for this problem have their own limitations. One possibility is to use a much longer FFT, say 16,384 bins, to provide much better low-frequency resolution, and another is to do multiple-span FFTs, effectively generating a log-resolution output.

These and other approaches that start with the FFT, however accurate in output, must still address the issue of real-time performance. Acoustic signals are by their very nature nonstationary in time, so an FFT analyzer that claims to match the performance of a traditional bank of analog

Fig. 1. The HP 3569A real-time frequency analyzer and the HP 35230A sound intensity probe.

1/3-octave bandpass filters must overlap data acquisition and processing, and cannot disregard samples at its analog-to-digital converter at any time. Even in this world of ever faster processing horsepower, this is a difficult task given the considerations listed above.

Filter analyzers are not without their own shortcomings. Constant percentage bandwidths allow only coarse frequency resolution toward the higher end of the acoustic spectrum, making the measurement of discrete tones difficult. Many computer and disk-drive manufacturers must locate irritating high-frequency noises emanating from their products. The bandwidth of the 16-kHz 1/3-octave band is nearly 4 kHz, making a precise spectral estimate of the tone impossible. Also, at low frequencies, the very narrowband 1/3-octave filters have long impulse responses, resulting in lengthy filter settling times. The potential for zooming in on a narrower frequency range using a local oscillator is not realistic for filter analyzers, although this feature is widely available in FFT analyzers.

Wavelet Analysis

One promising technology that bridges the gap between constant bandwidth FFT analysis and constant percentage bandwidth real-time analysis is called the theory of wavelets. Wavelets have properties that make them very attractive for the measurement of sound. Wavelet transforms allow a logarithmic frequency axis (i.e., good frequency resolution at low frequencies), and have arbitrarily good time resolution at higher frequencies, unlike the block-oriented FFT.[1] They have much more flexible bandwidths than filter analyzers. Instrumentation implementing discrete wavelet transforms has yet to appear, and faces significant obstacles in supplanting 1/3-octave analyzers because of the regulatory nature of acoustic noise measurements. Practical application of wavelets is beginning to appear in the acoustic trade journals,[2] but for the immediate future, the world of sound is seen through the poles and zeros of bandpass filters, analog or digital.

DSP in Acoustic Measurements

Digital signal processing has not bypassed the acoustics community. Digital filters have for some time replaced the old analog 1/3-octave filter banks, and are available as dedicated real-time frequency analyzers or filter analyzers, as distinguished from FFT spectrum analyzers, dynamic signal analyzers, and others that derive their results from the FFT. The digital filter analyzers are typically composed of a subset of single-purpose filter gate arrays, bit-slice microprocessors, and medium-scale integration DSP multiplier-accumulators. Such dedicated hardware has made it difficult for the digital filter analyzers to perform other signal processing tasks such as, for instance, the FFT. Acousticians do use the FFT for certain measurements, especially cross-channel analyses and tone determination, so any analyzer that could integrate digital filter analysis in 1/3-octave bands with the FFT would have a distinct advantage in the marketplace.

In 1991 Hewlett-Packard introduced an FFT analyzer, the HP 35665A, that also performs true digital 1/3-octave analysis using the combination of a 20-MHz Motorola 56001 DSP chip and a digital filter gate array. The gate array performs band-limiting low-pass filtering in an all-pass, multiple-sample-rate mode. The same gate array serves as a bandwidth limiter in the FFT measurement mode, so the DSP measurement hardware is shared between the measurement tasks of the instrument. This configuration, though integrating two traditionally separate instruments into a single box, still relies on a very costly development, the filter gate array, because there is not enough power in the 56001 to perform all of the tasks required for real-time digital filter analysis to 20 kHz.

Just one year later, the clock speed of the Motorola 56000 series processor doubled to 40 MHz, removing any remaining barrier to the software implementation of a combined FFT and real-time 1/3-octave analyzer on a chip. This goal was realized with the introduction of the HP 3569A, which incorporates the combined functionality of a dual-channel FFT analyzer and a real-time 1/3-octave sound intensity filter analyzer, in addition to an integrating sound level meter and a reverberation time processor. The wholly digital processing allows very high precision, unknown to date in a handheld package.

In the following sections, we will explore how the standard 1/3-octave filter shapes are implemented as digital filters, starting first with a numerical design methodology, and continuing to the actual software architecture. Then we will examine some of the design elements, unique to the digital domain, that make possible very accurate real-time cross-channel calculations of acoustic intensity and full-bandwidth correction of microphone phase mismatch in sound intensity probes.

Signal Processing Block Diagram

A description of a 1/3-octave filter analyzer using an off-the shelf DSP chip is primarily concerned with a software algorithm, but we will first present a general picture of the HP 3569A measurement hardware. Fig. 2 is a block diagram. All measurements take place in the Motorola 56002 DSP chip, while the Intel 80186 processor is responsible for display processing, management of the nonvolatile RAM disk, and the user interface. For any stand-alone instrument, the user interface firmware is a significant aspect of the product development effort. In this paper, however, we are concerned with a description of the HP 3569A measurement algorithms and signal processing. We shall show that the entire measurement of acoustic pressure or two-channel acoustic intensity, including triggering, averaging, scaling, and coordinate transformation to dBSPL (sound pressure level) is performed by the 56002 DSP chip. Data output is over the 56002 host interface, which in our case had a DMA channel dedicated to it for fast result storage. Because of the modularity of the system with respect to the measurement algorithms, a DSP architecture similar to Fig. 2 can be used as the measurement engine for any instrument or computer-based user interface.

We designed the system to include two independent channels, since cross-channel analysis (e.g., frequency response, correlation) is one of the benefits of FFT analysis, and since it makes possible the measurement of real-time sound intensity in 1/3-octave bands.

Sound Intensity

Sound intensity is defined as the rate of sound energy transmission per unit area, and represents a vector quantity that has many uses to noise control engineers. For example, intensity can be used to estimate the sound power emitted

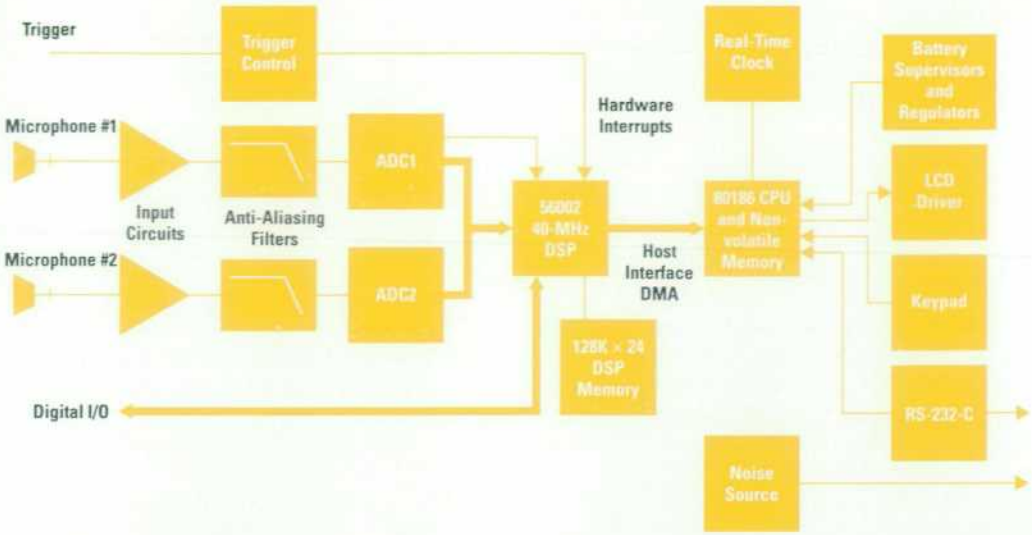


Fig. 2. Hardware block diagram of the HP 3569A real-time frequency analyzer.

from a device or the sound absorbed by a panel, or to map the sound field in a noisy aircraft cabin.

The primary attractiveness of offering intensity measurements in a portable analyzer is that it allows accurate in-situ measurements. Single-channel pressure measurements require assumptions about the environment—for example, that the sound field is either anechoic or reverberative. Since this is likely not the case for most industrial environments, a noisy device under test must frequently be moved into a special chamber for acoustic evaluation. Using intensity measurements, a two-microphone probe (Fig. 3) can discriminate between background noise and a directional sound source, facilitating accuracy without severe environmental constraints.

Mathematically, intensity can be expressed as the time average of pressure times velocity. Euler's equation relates velocity to the time rate of change of pressure at a microphone.[3] The resulting expression for the mean sound intensity takes the form:

$$I = \overline{-\frac{p_1 + p_2}{2\rho\Delta r}\int (p_2 - p_1)dt},$$

where p_1 is the sound pressure at microphone 1, p_2 is the sound pressure at microphone 2, Δr is the distance between the microphones, and ρ is the air density.

Fig. 3 depicts how signals from two pressure microphones are combined to compute intensity in 1/3-octave bands. For this time-domain calculation of intensity to have sufficient accuracy, the microphones as well as the measurement channels must be very closely matched in phase (0.02 degrees below 250 Hz according to IEC 1043, a standard for intensity measuring instruments). There are also dynamic range requirements placed on the processing system, in that the pressure difference between the two channels can be very small while the pressure magnitude can be large, resulting in precision errors for fixed-point microprocessors. We chose to implement the digital integrator using the trapezoid rule, which has a desired 90-degree phase shift at all frequencies less than the Nyquist rate. The output of the sound intensity calculation is time averaged, converted to dB, and sent to the host processor. The frequency range of interest for sound intensity is 20 Hz to 10 kHz, which results in 28 bands of intensity information calculated in real-time.

Sound Pressure

In the more common single-channel or dual-channel pressure measurement where intensity is of no concern, the output of each 1/3-octave filter is magnitude squared and time averaged over a user-entered integration time. Since the 1/3-octave digital filters run in real time and provide an output for every sample input, integration times can be very short, limited only by the speed of the DSP chip. Realistic

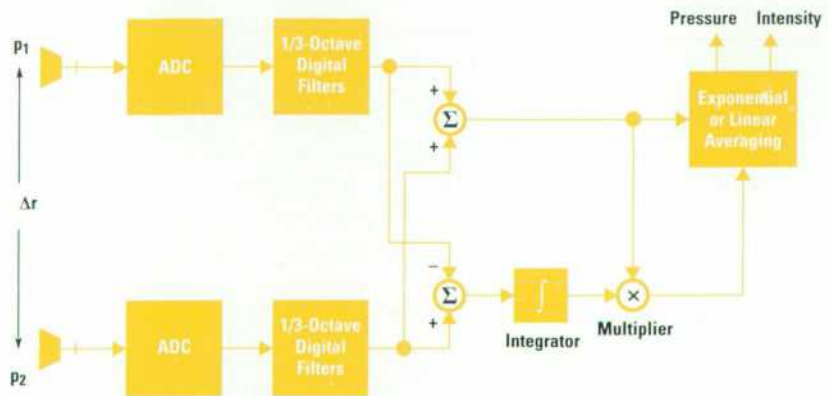


Fig. 3. Processing method to compute sound intensity.

minimum integration times are on the order of a few milliseconds, independent of frequency. Only the impulse response of the bandpass filter limits the practicality of using very short average times to monitor transient events. We again contrast this to the FFT, which requires a time record length proportional to the number of lines of resolution. Normally this is on the order of 20 to 30 milliseconds at the 20-kHz acoustic frequency span. Even with overlapping of the input time records, the FFT tends to smear out transient events both in time and frequency.

Not shown in Fig. 3 are optional processing steps subsequent to the calculation of pressure and intensity. For example, a maximum or minimum hold feature can hold the highest or lowest output of the averager within each 1/3-octave band over some specified time. Minimum hold is useful for measuring background noise levels measured in a room that may be impacted by temporary noise events, like a ringing telephone. For each 1/3-octave filter, an amplitude histogram is maintained having resolution in dBSPL. Thus statistics are available as a postprocessing operation on the histograms. One common statistic is called the exceedance level, defined as the percentage of time that a particular 1/3-octave band sound level exceeded a proscribed dBSPL value.

Sample Rate Decimation

One way to design a bank of digital bandpass filters that simulates the traditional analog 1/3-octave filter sets is to derive a unique set of filter coefficients for each desired center frequency. These filters would all run at the sample rate of the analog-to-digital converter (ADC), resulting in huge demands in processing, particularly when the number of desired frequency bands is large. Instead, the HP 3569A uses a multirate system, as shown in Fig. 4. By progressively lowering the sample rate of the system after each set of three bandpass filters, we reduce the amount of work needed to process the entire bank of filters. According to the Nyquist sampling theorem, we require a sample rate for each octave only a factor of about 3.3 above the center frequency of the highest 1/3-octave filter within each set of three filters. For every ADC sample input to the highest stage, the average work turns out to be equivalent to six bandpass filters plus two decimating low-pass filters (or twice the work required to process the highest octave's filters), as depicted by the discrete processing blocks in Fig. 4.

Another important advantage of the decimation approach to the digital filter bank is that each set of three bandpass filters, centered at 1/3-octaves within each octave of processing, is independent of sample rate, such that the pole-zero positions of the filters in the z plane are identical for all octaves. This is a result of the relationship between the sample rates and the center frequencies of the filters. As an illustration, the 1/3-octave center frequency of 20 kHz divided by the sample rate of the highest octave, 65,536 Hz, is precisely the same as the center frequency of the 10 kHz band divided by its sample rate of 32,768 Hz. Thus these filters have identical formulations in the z-domain, and only three distinct bandpass filters need be designed for the entire 1/3-octave filter bank. Similarly, the decimating low-pass filters all have the same pole-zero topology.

This is a nontrivial advantage over any analog filter bank, in which each bandpass filter must be individually designed for flatness and shape. However, a small problem arises because traditionally accepted center frequencies have followed the base-10 system, $f = 10^{3n/30}$, where $0 < n < 45$. The digital method follows a base-2 formula, $f = 1000(2^{(n-30)/3})$. For n = 43 (20,000 Hz), this results in a center-band frequency error of about 1 percent. Although the difference is small, both of the major 1/3-octave filter standards in existence (ANSI S1.11-1986 and IEC 255-199X-Draft)[4,5] allow either method

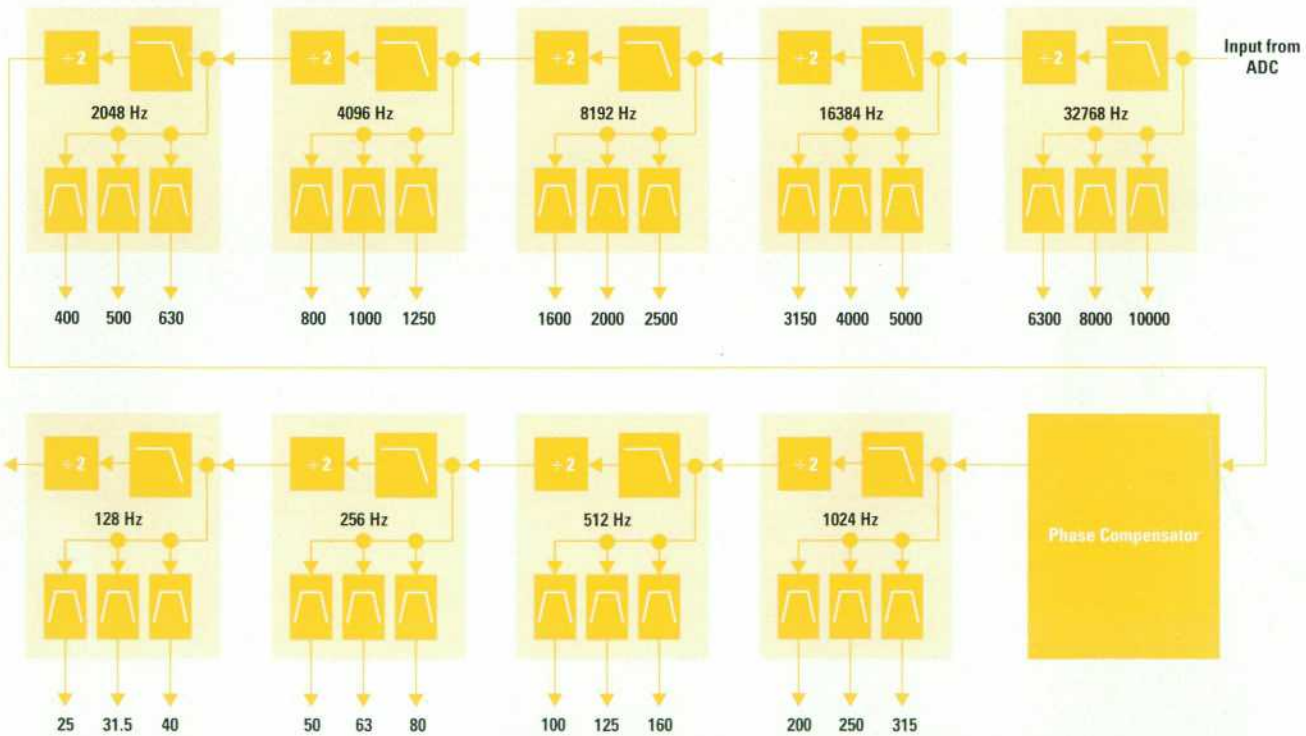


Fig. 4. The digital 1/3-octave filter bank for one channel. The numbers under the filters are the center frequencies of the 1/3-octave filters.

and require that a filter set be marked as either base-10 or base-2.

Phase Accuracy

Another advantage of the digital method relates to cross-channel phase accuracy. Referring to Fig. 3, we see that the calculation of intensity occurs after the bandpass filters, so that the intensity output represents a value band-limited within a particular 1/3 octave. Since digital filters have stable and deterministic transfer functions, we can be assured that no relative phase difference between channels is introduced in the filtering process. For the digital 1/3-octave analyzer, only the analog anti-aliasing filters and the ac-coupling capacitors, both of which precede analog-to-digital conversion, can significantly affect cross-channel phase accuracy. In contrast, it may not be possible to design an accurate analog 1/3-octave intensity analyzer because of the required phase match between the bandpass filters. Keeping such systems calibrated and matched would be an arduous task.

Finally, the all-digital processing allows a phase compensation technique for which there is no equal in the analog processor. Fig. 4 shows a phase compensator placed within the processing chain at the 1024-Hz sample rate, such that the phase of all downstream calculations is adjusted to the second channel as a function of frequency. Note that Fig. 4 shows only one channel of a two-channel sound intensity processor. The compensator is a unity-gain digital filter with adaptive coefficients to correct for phase mismatch between the individual microphones of a sound intensity probe, in real-time.[6]

Depending on the accuracy required, additional phase compensators can be placed at any sample rate in the signal flow graph. A multiorder digital compensation network can tailor the phase-versus-frequency characteristic by cascading the effects of several low-order compensators. The multiple-sample-rate topology used is especially well-suited to this, since a phase anomaly that may only affect very low frequencies need only be corrected at those frequencies.

Filter Design

Five types of filtering are performed in the HP 3569A digital 1/3-octave analyzer, all operating in real-time simultaneously:

- Low-pass filtering and decimation between sample rates for each octave
- Either sixth-order 1/3-octave or fourteenth-order octave digital Butterworth bandpass filters
- Discrete integration using a simple IIR (infinite impulse response) filter (the trapezoid rule)
- A single-pole low-pass smoothing filter to allow exponential smoothing of the detected filter output
- Variable-coefficient IIR phase compensation filters.

Digital low-pass filter design is well-known and we called upon a PC design tool called Dispro® (manufactured by Signix, Inc.) to determine the coefficients of an eighth-order (8 poles, 8 zeros) elliptic low-pass filter. This digital anti-aliasing filter has 90-dB rejection at a frequency of $0.3f_s$ and less than 0.003-dB passband ripple. The passband edge begins at $0.2f_s$, and at the decimated Nyquist frequency of $0.25f_s$, the attenuation is 30 dB. After discarding every other sample, the new Nyquist frequency is decreased by a factor of two. Aliasing is not a concern since the bandpass filter attenuation is significant in the frequency range $0.25f_s$ to $0.30f_s$. The combined effect of low-pass and bandpass filter stopband rejection at the new Nyquist frequency is over 80 dB, as shown by Fig. 5.

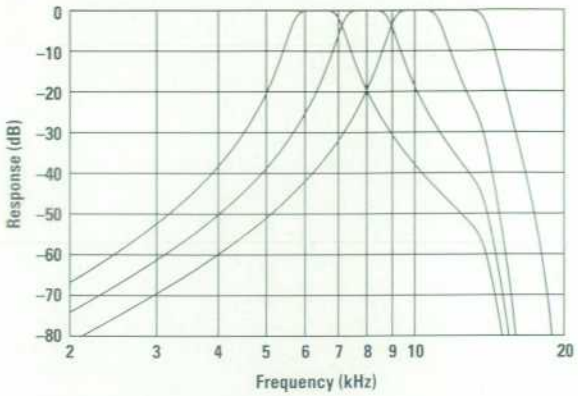


Fig. 5. 1/3-octave filter shapes used in the HP 3569A real-time frequency analyzer. The plot shows the effect of the digital decimating low-pass filter. The low-pass filter has an input sample rate of 65,536 Hz. After low-pass filtering, every other sample is discarded and the new sample rate is 32,768 Hz. This is used as the bandpass filter sample rate. Thus the Nyquist frequency is 16,384 Hz.

Polytope Optimization

For the bandpass filter design, a different methodology was used. We used a brute-force numerical method called polytope optimization to find the best position for the poles of the filters. Standard filter design programs (like Dispro) do not allow transition bands to be specified on a constant percentage bandwidth basis. In the polytope method, each of the six pole positions is compared with the nominal ANSI third-order Butterworth filter shape, resulting in a minimization problem in six variables (the pole positions). The polytope method, described by Wright,[7] assumes nothing about the continuity of the function. It compares the error (in the least-squares sense) of the transfer function resulting from the current placement of the six poles to the target filter shape. Each pole is positioned at the center of a small triangle inside the unit circle on the z plane. Two vertices of the triangle are held constant on each iteration. A new pole position is evaluated by flipping the triangle on one of its sides, so that the new pole position is at the center of the flipped triangle. Vertices are stretched according to the magnitude of the error. Eventually the pole positions converge to the best location for the bandpass filter. Severe penalties are levied by the design program if poles are placed outside of the unit circle, for instance. Excessive passband ripple is handled in the same way. All the zeros of the system lie on the real axis at either dc or the Nyquist frequency, and thus are independent of the filter's center frequency. The positions of the zeros are predefined and are incorporated into the target equation. Fig. 5 shows the magnitude response of all three 1/3-octave bandpass filters over the 80-dB operating range of the system. The topology of the filter is shown in Fig. 6.2

The polytope error function failed to provide the necessary flatness in the passband for the fourteenth-order octave filter (one filter per octave). Since only one filter had to be designed and replicated across all octaves, a standard analog Butterworth filter was designed and transformed to the z

domain using the bilinear transformation. Excellent passband ripple performance was obtained by this approach.

Flow Graph Optimization

The positions of the 1/3-octave filter zeros follow from the analogue to Butterworth bandpass filters. We want to limit the power transmitted through the filter close to the Nyquist frequency, and hence place a zero there ($z = -1$). The filter also must attenuate low frequencies, and therefore we place another zero at dc ($z = +1$). Through evaluation of the final error function as output by the polytope design method, we find that increasing the number of zeros at dc reduces the ripple in the passband of the filter. In fact, we place three zeros at dc, resulting in a very convenient direct-form FIR (finite impulse response) filter for the zeros. By cascading zeros, we find in the z domain:

$$(1-z)(1-z)(1-z)(1+z) = (1-z)(1-z)(1-z^2),$$

which is implemented as a cascade of three simple FIR filters having unity coefficients.

Passband ripple is one ingredient in the accuracy grade of an ANSI 1/3-octave filter. The filters shown in Fig. 5 have less than ±0.008 dB ripple, and when this is combined with the decimating low-pass filter ripple of ±0.003 dB, the worst-case passband ripple is 0.022 peak-to-valley. This is equivalent to a Type 1 ANSI S1.11 filter or Class 0 IEC 225 filter.

A small complication arises in that the three bandpass filters have different gain factors. Synthesis of the filters shows gains on the order of 100 to 200, which must be accounted for in the processing, particularly for fixed-point DSP chips, where bit growth and overflows are a concern.

Discrete Integration

The integration filter used in the sound intensity measurement is implemented as a simple IIR digital filter. In the calculation of intensity, it is very important that the integrator impart a precise 90-degree phase shift to the sample data stream. This is accomplished by use of any symmetric integral function, for example the trapezoid rule:

$$y(n) = y(n-1) + (1/2)[u(n) + u(n-1)].$$

Simpson's rule also has good phase performance and better amplitude accuracy over all frequencies.[8] However, in the presence of high-frequency noise, the trapezoid rule excels since it attenuates signals close to the Nyquist frequency instead of accentuating them as does Simpson's rule. As is evident from the impulse response y(n), the coefficient magnitudes are either 1.0 or 0.5, so the filter can be implemented by adds and shifts. Although the Motorola 56002 can multiply as fast as it can shift by one, not having to maintain pointers to multiple coefficient tables is a significant advantage. Fig. 7 shows the magnitude response of the digital integration filter. Although not shown, the phase characteristic is precisely –90° from dc to the Nyquist frequency. The magnitude of the filter response is a function of frequency. To compensate for the difference between the trapezoid-rule magnitude and the ideal, a gain correction is applied to the center-band frequencies of the 1/3-octave intensity spectrum.

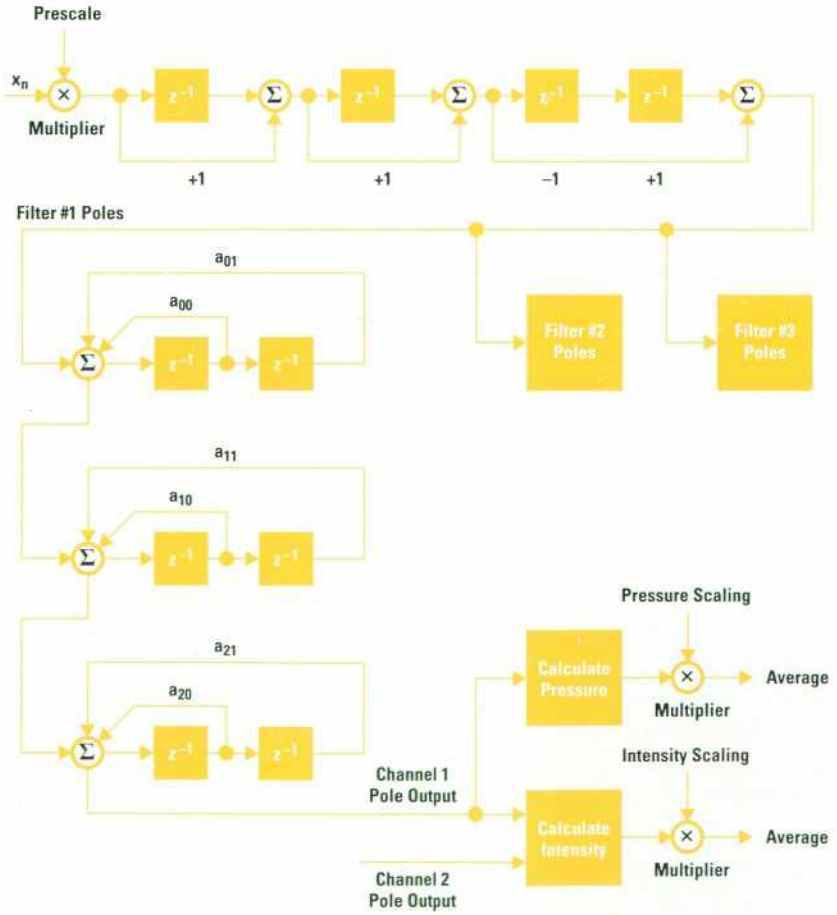


Fig. 6. Flow graph of the bandpass filter zeros and poles, single-channel.

Fig. 7. Magnitude characteristic of the trapezoid integration filter, showing conformance to the –6-dB-per-octave roll-off of a theoretical integration function. The phase characteristic is precisely –90° to the Nyquist frequency of 16,384 Hz.

DSP Software

The most central aspect of the processing represented by Fig. 3 is that some logic must control when the filters of each selected octave are run. This is handled by a "pass" look-up table. At every new sample from the ADC, an element is read from the table, which has a modulo length of 2^{N-1}, where N is the number of octaves. The value returned from the look-up table is the number of octaves to process before returning to read another ADC sample. The first few elements of the table are:

1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,...

In the limit, the average number of passes performed on each ADC sample is two. One iteration of a "while" loop executes all the processing encompassed by the discrete blocks shown in Fig. 4 for both input channels, as well as detection and averaging of the output. Fig. 8 shows C-language pseudocode for the high-level control flow. For performance reasons, the actual implementation is in 56002 assembly language.

Fixed-Point Scaling

Scaling of the fixed-point 1/3-octave analyzer calculations is a significant implementation issue. With the multiple filters, sample rates, and gains embodied in the processing system, one might be tempted to suffer a higher part cost and embrace floating-point DSP chips for this application. Actually the choice is not so obvious. The floating-point chips remove the possibility of overflow and underflow results from the signal processing, a major advantage. However, the clock rates and multiply-accumulate times are not significantly different. Both the Motorola 56002 and the Motorola 96000 run at 40 MHz. The power consumption of the fixed-point chip is much lower and the price differential between the chips is a factor of four.

```
while ( ADC FIFO not empty )
{
  Read ADC sample from FIFO for Ch1&2
  Detect overload or trigger condition
  Look up current pass_count from table

  while ( pass_count- -)
  {
    for ( each of three 1/3-octave filters)
    {
        Ch1 bandpass filter
        Ch2 bandpass filter
        Do intensity calculation
        Accumulate results
    }
    Perform low-pass decimation filter
  }
  if ( ADC sample modulo 64)
  {
    Convert accumulators to block float
    Perform exponential and linear averaging
  }
}
```

Fig. 8. Pseudo-code for the octave processing loop.

We chose to take on the increased software burden and solve the multitude of scaling issues that arise in the fixed-point case, because of the attendant benefits. The scaling considerations are many. Since we have a multiple-sample-rate system, the accumulation of power in the detection and averaging section varies greatly across octaves. The bandpass filters have very large gains. Programmable integration times can range from milliseconds to hours. One way to limit the percentage of processing time used in detecting and managing arithmetic overflows is to design the software so that scaling is done at deterministic intervals, rather than testing for overflows on every multiply-accumulate operation.

This deterministic scaling has been built into the pass look-up table. One of the higher-order bits of the pass count is encoded with a flag to force a normalization of the output data every 64 ADC samples (approximately 1 ms at a sample rate of 65,536 Hz). This amortizes the overhead of scaling over a much longer time than if the built-in 56002 extension bits were put to use. Also, since the rate of bit growth per octave is reduced by one-half for each lower octave, the same pass count table can be used to determine the number of octaves to normalize on each 64-point boundary. This is accomplished by a second pointer into the table that increments only once per 64 points. Thus, after the first millisecond, only the highest three 1/3-octave bands (i.e., the highest-sample-rate processing block in Fig. 3) are normalized, since they are the only ones with the full 64 accumulated filter outputs. This approach minimizes the amount of work done to maintain full 24-bit precision fixed-point calculations. We estimate that less than two percent of the available CPU cycles are used in the deterministic scaling algorithms.

Normalization as implemented in the 56002 1/3-octave analyzer forms a 48-bit fractional mantissa and a 24-bit power-of-two exponent for each 1/3-octave band output. Until the 64-point boundary is reached, only the accumulator outputs (without exponent) of the averaging process are saved between ADC samples. Each successive 64-point normalization interval combines the latest accumulator output with the current floating-point sum (including exponent).

Scaling of data before output includes adjustments for linear averaging time, aggregate gain (through all filters and integrators), and unequal accumulation between octaves. These operations are performed as pseudo-floating-point operations in that they may operate on the exponent, the mantissa, or both. Maintaining an exponent with the data is especially convenient because the end result of all the scaling is a conversion to base-10 logarithms. This amounts to a base-2 log table look-up (using the mantissa as the index), which when added to the existing exponent gives the full base-2 logarithm. A simple multiply then converts to base 10.

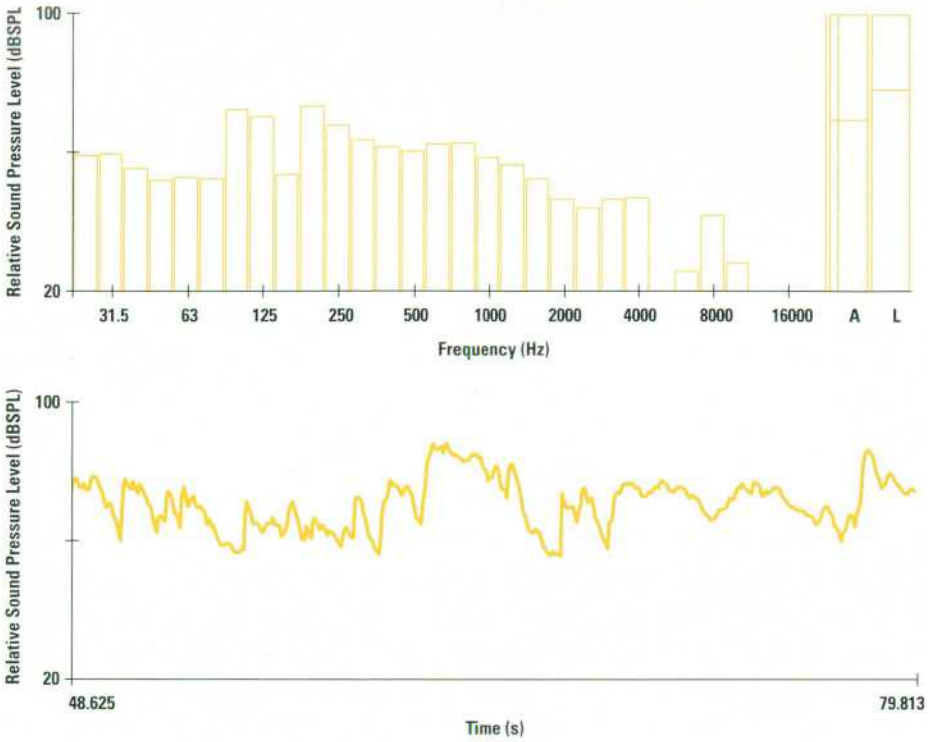


Fig. 9. (top) HP 3569A 1/3-octave display with A-weighted and linear levels. (bottom) Slice display from the HP 3569A showing the time history of the A-weighted sound pressure level.

In software, a circular FIFO is maintained by an ADC interrupt routine so that the main filtering loop is allowed to fall temporarily behind the data converter. This allows for longer processing delays resulting from data normalization, conversion of accumulated pressure and intensity to dB, and data output. Although it occurs infrequently, it can take up to 1/2 millisecond to take the log of both the pressure and intensity spectra (all 1/3-octave bands). For this purpose, a FIFO length of at least 32 samples per channel is required. In reality, a large 512-sample FIFO is used to allow asynchronous command processing from the 80186 controller. Some of these commands (e.g., clearing the current average accumulators and amplitude histograms) can take several milliseconds.

Overall Sound Level Bands

Traditional A-weighted sound measurements have been acquired using simple sound level meters. Most octave-based analyzers have eased comparison of the octave data to dBA by providing an overall level at the right-hand side of the octave data (Fig. 9). The HP 3569A has two overall bands that can simultaneously show both the A-weighted and linear overall levels. When a marker is positioned on the A-band, the readout is precisely what would be measured using a sound level meter. The calculation is performed in the DSP as a simple weighted block multiply of the linear-weighted 1/3-octave bands, unless the analog A-weight filter is already switched into the signal path.

Optionally, the linear band can be replaced by an overall impulse measurement, another function found in sound level meters for measurement of impulsive noise. This is a special 35-millisecond exponential time weighting applied to the broadband time data after frequency weighting by the analog A-weight filter. The measurement is performed at the maximum sample rate of the analyzer (65,536 Hz in one-channel mode) in parallel with 1/3-octave analysis. Fig. 10 is a graphical overview of the sound level meter capability of the HP 3569A.

Triggering

Triggering and overload detection are handled by the 56002 interrupt routine. When these events occur, a status word associated with the current ADC sample is tagged with the appropriate event and placed into the input FIFO. Later, as the filtering loop reads the ADC sample out of the FIFO, actions are taken on the events. Triggering in a real-time filter analyzer does not initiate data collection, since the 1/3-octave and low-pass decimation filters all must be running and settled when the trigger occurs. Instead, the status of the latched trigger line is polled on each new sample. This allows special trigger modes not normally associated with FFT spectrum analyzers. For example, gated averaging specifies that the output of the bandpass filters be linearly averaged while a TTL external trigger is asserted. The time resolution for this gate can be as little as 1 ms, making possible triggered transient event analysis in 1/3-octave bands.

Averaging

Filtered pressure and intensity outputs can be averaged by either linear or exponential methods. When exponential averaging is selected, the filter outputs are smoothed by another simple digital filter:

$$y(n) = (1-A)y(n-1) + Au(n),$$

where u(n) is the current filter output and y(n) is the current average output. The constant A is related to a time constant of the exponential rate of change of the analyzer's output in response to an instantaneous change at the input. Standard time constants used by acousticians are 1/8 second (fast) and 1 second (slow). The factor A can be expressed as $A = t/\tau$, where t is the decimated sampling interval and τ is the desired time constant. In normal cases A is quite small for the

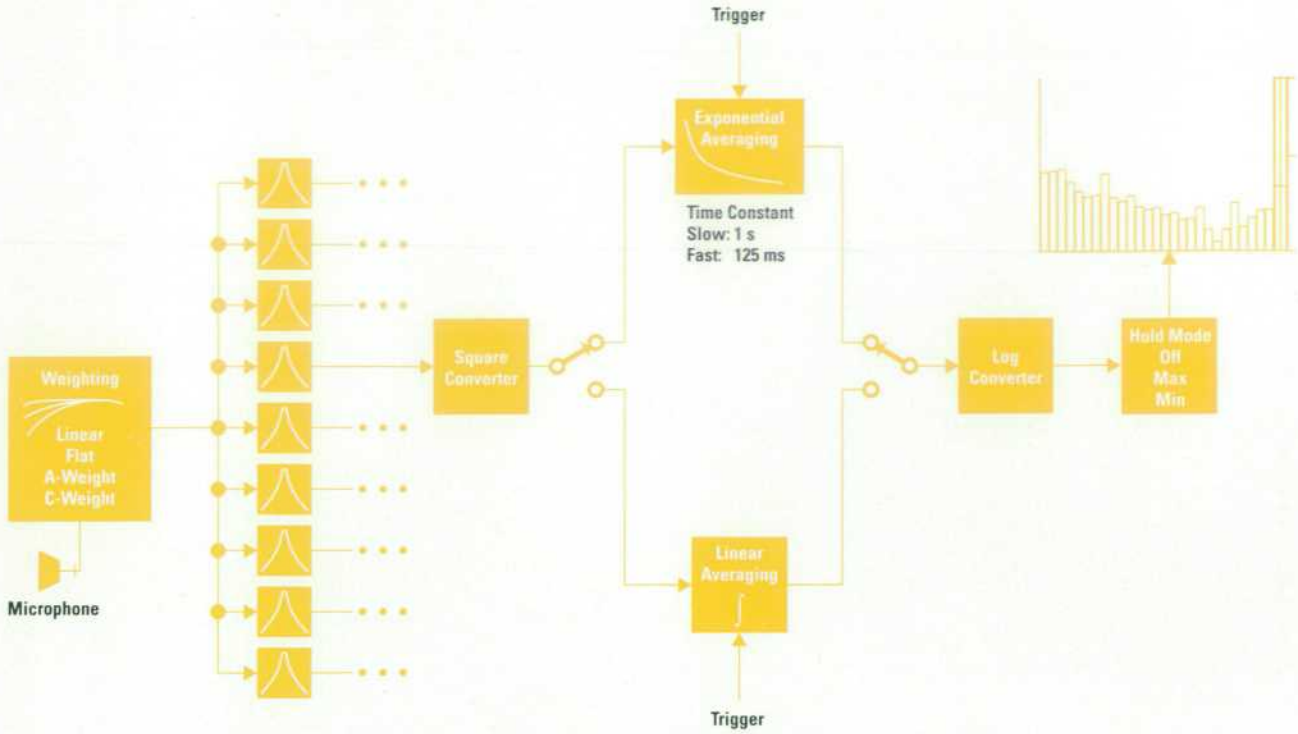


Fig. 10. Linear and exponential averaging are done on the squared output of each 1/3-octave frequency band. The output of the averager is converted to dBSPL (sound pressure level).

higher-frequency 1/3-octave bands and increases for the lower bands. A special case of exponential averaging is called equal confidence averaging, which varies the time constant for each 1/3-octave band so as to distribute the variance equally across the spectrum. In this case, A is constant for all bands.

Linear averaging is simply a uniformly weighted time average of the accumulated power passed by a particular 1/3-octave filter over a predefined interval. If the measurement is set to repeat, then the accumulator outputs are set to zero and the intensity integration filter is reset without missing any new data samples. Fig. 10 depicts how linear and exponential averaging are implemented in the context of a digital real-time frequency analyzer.

One of the primary advantages of real-time frequency analyzers is the ability to output repetitive spectra at very short intervals. The HP 3569A collects each averaged spectrum (at intervals as short as 4 milliseconds) into a multispectrum buffer. A view of a time slice through this buffer is presented on the lower display of Fig. 9. A slice of spectral values is available for each 1/3-octave frequency band, enabling precise examination of the components of any transient event.

Conclusions

The design philosophy of a digital 1/3-octave filter analyzer, as implemented on a single fixed-point DSP chip, has been presented. The main goal of this development is to allow general-purpose signal analysis using both FFT and digital filter methods without hardware dedicated to either purpose. This flexible approach has many advantages, especially as it pertains to real-time phase correction of sound intensity probes.

Acknowledgments

Many thanks to Chris Sutton, who was the hardware designer for the HP 3569A. Early product brainstorms with Chris showed that the hardware used for FFT analysis was nearly identical to what was needed for acoustics applications and 1/3-octave analysis. Without the skill of Phil Hollenhorst, HP 3569A project manager, to generate enthusiasm for entering a new market area, the described developments would have never taken place. Randy Sartin, Neel Malik, Mike Hall, Eric Tilman, Rick Van Ness, and Dave Dintenfass all made key contributions during the course of the project.

References

1. O. Rioul and M. Vetterli, "Wavelets and Signal Processing," *IEEE Signal Processing Magazine*, October 1991, pp. 14-38.
2. P.V. Bruel, "Practical Use of Acoustic Technology for Measurements of Energy Flow, Holography, and Wavelets," *Proceedings of NoiseCon*, July 1991, pp. 11-28.
3. G. Rasmussen, "Intensity—Its Measurement and Uses," *Sound and Vibration*, October 1989, pp. 12-21.
4. *American National Standard: Specification for Octave-Band and Fractional-Octave-Band Analog and Digital Filters, ANSI S1.11-1986*, Standards Secretariat, Acoustical Society of America, New York.
5. *International Electrotechnical Commission Draft Standard, IEC 1043-199X, Instruments for the Measurement of Sound Intensity*, 1991 Draft.
6. J.W. Waite, "Time-Domain Compensation for Microphone Phase Mismatch," *Proceedings of Internoise-92*, Toronto, pp. 1187-1190.
7. P.E. Gill, W. Murray, and M.H. Wright, Practical Optimization, *Academic Press*, 1981, pp. 94-96.
8. R.W. Hamming, *Digital Filters*, Second Edition, Prentice Hall, 1977, pp. 59-63.

Continuous Monitoring of Remote Networks: The RMON MIB

An introduction to the capabilities of the Remote Monitoring Management Information Base of the Simple Network Management Protocol and its implementation in the HP LanProbe II network monitor.

by Matthew J. Burdick

Over the past several years, standards and open systems have become important in the computer industry. Interchangeable elements in customers' computing systems offer a number of benefits: no dependence on a single vendor, the ability to compare similar products from multiple vendors, and the safety of an assured future upgrade path. In network management, no standard has more successfully ridden the crest of the standards wave than the Simple Network Management Protocol (SNMP).* Nowhere have the limits of the SNMP protocol been explored more fully than in the Remote Monitoring Management Information Base (RMON MIB).[1]

As networks have grown from isolated islands supporting small groups to pervasive and critical components in the operation of large organizations, the need to monitor these networks at all times has become increasingly important. When networks were small, portable protocol analyzers were sufficient to ensure continuous operation. When problems developed, an analyzer could be carried to the network in question and troubleshooting could be performed. As networks grew, however, it became more difficult to transport analyzers to trouble spots. Certainly it was impractical to tie up the analyzers to perform routine network monitoring in an effort to provide proactive network troubleshooting. Distributed network monitors or probes fill this role, providing historical trend data, lists of network users, and continuous monitoring of network activity.

These probes are physically similar to network routers—they contain a network interface or interfaces and provide no user interface. The RMON probes continuously collect network data, which can be gathered from the probes by a separate management station via the SNMP API (application program interface). Because all the probes can be managed from a single management station, they are low-cost management tools.

The capabilities of an RMON probe will be discussed after the following background information.

RMON History

In 1988, the Internet Activities Board (IAB) released RFC 1052, *IAB Recommendations for the Development of Internet Network Management Standards*.[2] In it, the older Simple Gateway Management Protocol (SGMP) was generalized and renamed SNMP. The name change reflected the evolutionary changes applied to the gateway management protocol to transform it into a general network management protocol.

* Unfortunately, the use of acronyms is ubiquitous throughout the Internet standards process. Please refer to the Glossary at right to decode unfamiliar terms.

Glossary

Quoted items are from document RFC 1310, *The Internet Standards Process*, of the Internet Activities Board.

Agent: A probe, device, or process that collects data and makes it available via SNMP in the form of an MIB.

API: Application program interface, a well-defined functional interface between a software application and a service.

CMIP: The Common Management Interface Protocol, a complex OSI network management protocol. Pronounced "see-mip."

Gateway: A node that joins two networks.

IAB: "The Internet Activities Board (IAB) is the primary coordinating committee for Internet design, engineering, and management."

IETF: "The IAB has delegated to its Internet Engineering Task Force (IETF) the primary responsibility for the development and review of potential Internet Standards from all sources. The IETF forms Working Groups to pursue specific technical issues, frequently resulting in the development of one or more specifications that are proposed for adoption as Internet Standards."

JetDirect MIB: An MIB defining the HP LaserJet printer's interface to the network.

LanProbe: HP's remote monitoring probe. SNMP and proprietary versions are available.

MIB: Management Information Base, a tree-shaped hierarchical database containing objects. Pronounced "mib."

OpenView: HP's umbrella network management system.

RMON: Remote Monitoring. Monitoring a network using a management station physically separated from a series of distributed probes. Pronounced "ARE-mon."

RFC: A request for comments issued by the IAB. Each RFC describes a protocol used on the Internet.

Router: A gateway that performs network-layer routing.

SGMP: The Simple Gateway Management Protocol, the precursor of SNMP.

SNMP: The Simple Network Management Protocol, an evolutionary development of SGMP.

SMI: Structure of Management Information, the definitions of object types in an MIB. Described in RFC 1155.

SunNet Manager: Sun Microsystems' umbrella network management system.

Naturally, in view of the gateway roots of SNMP, the first implementations of SNMP occurred in gateways such as hubs and routers. These devices offered a set of network statistics known as a Management Information Base (MIB). Later, other devices offered expanded sets of statistics, each referred to as an MIB. The original MIB, MIB-I, was obsoleted and replaced by MIB-II. In addition, each vendor offered a different set of data known as a private, or enterprise-specific, MIB. Each of these MIBs is registered, or released to the public, at well-publicized archive sites on the Internet. At the time of this writing, enterprise-specific MIBs from 12 different vendors have been registered.*

One such MIB was registered for the Novell LANtern network monitor. It provided some of the advanced network management features included in the HP LanProbe,[3] but access to the probe was available via SNMP. This valuable feature allowed network management applications such as HP OpenView and SunNet Manager access to the probe's data.

The RMON MIB builds on features in the HP LanProbe, Novell LANtern, and other remote network monitors. It was developed jointly by a group of people from a variety of vendors, including HP. Its goal is to standardize the data presentation and control of these dissimilar network monitors. Today, there are more than 10 announced implementations of the RMON MIB, and support continues to grow. This article will familiarize the reader with the capabilities of the RMON MIB and its implementation in the HP LanProbe II.

SNMP Overview

Although a thorough overview of SNMP would be quite lengthy, some understanding of the basic ideas is necessary to understand the RMON MIB. Before describing the RMON MIB, the reader must grasp the organization of data in the SNMP tree, the types of objects contained in the tree, and operations available to manipulate those objects.

The foundation of SNMP is a tree-structured data base composed of subtrees called MIBs (Management Information Bases). Each MIB is intended to contain information specific to various network devices such as routers, bridges, remote monitors, and perhaps even printers and other devices not devoted solely to network activity. For instance, MIB-I and MIB-II allow monitoring and control of general network devices, while the enterprise-specific MIB for the JetDirect card (an network expansion card used in some HP printers and plotters) provides information relevant to the status of the printer.

Fig. 1 shows a subset of the SNMP tree. The full tree is not maintained at a single site. It consists of a number of MIBs, each of which may change independently of the others. This is particularly true of MIBs located under the Enterprises branch, since each MIB here may be maintained by a separate company.

Although the tree in Fig. 1 shows only one end or leaf node (gdStatusPaperJam), each such node represents an SNMP object, and is named by listing the path from the root of the tree to the object in question. Objects that are not columns in a table have an additional suffix of 0 appended. For example, the gdStatusPaperJam variable in the JetDirect MIB indicates the current status of the printer—a value of 1 indicates a paper jam while a 0 indicates no jam. Its object ID is 1.3.6.1.4.1.11.2.3.9.1.1.2.9.0. The trailing zero is there because gdStatusPaperJam is not located in a table. A management station such as HP OpenView must query the printer's agent for the value of this object ID to determine the status of the printer.

* For access to the registered enterprise-specific MIBs, ftp anonymously to host venera.isi.edu and retrieve files from the mib directory.

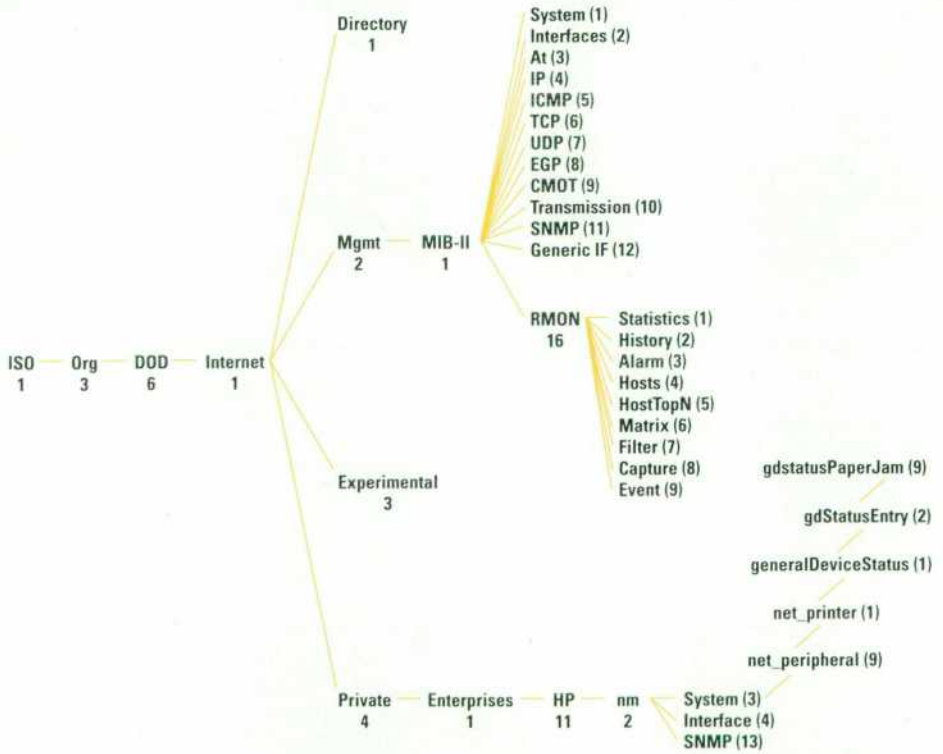


Fig. 1. A portion of the SNMP tree.

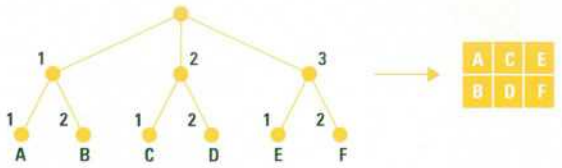


Fig. 2. Tree-to-table mapping.

More complicated structures such as tables are also supported in SNMP using the tree representation. A two-dimensional table with rows and columns can be represented by a subtree as shown in Fig. 2. Objects in the table are labeled with letters A through F, and the subtree's mapping to table form is also provided.

The ability to view a branch of the tree as a table is a useful abstraction that makes it easy to organize the information into a readable format. Columns can represent categories such as the number of bytes ("octets" in SNMP parlance) or Ethernet broadcasts detected on the LAN, while rows can represent different sample times to provide a history of the network activity. In fact, this is exactly what the RMON MIB history group (described later) provides.

Similarly, the tree-to-table mapping can be extended to represent three-dimensional tables as shown in Fig. 3.

A three-dimensional table can be viewed as a collection of two-dimensional tables. As before, each two-dimensional table can represent a historical sampling of network activity over time, but each table can be sampled at different rates. Such a collection of tables can, for example, hold data sampled at 2-second, 30-second, and 1-hour intervals. The RMON MIB contains several instances of three-dimensional tables.

A limited number of data types are available for end nodes in the tree. They are defined in RFC 1155, *Structure and Identification of Management Information for TCP/IP-Based Internets*.[4] Some of the available data types are:

- Integer
- Octet string
- Object identifier
- IP address
- Network address
- Counter (an integer from 0 to $2^{32} - 1$)
- Gauge
- TimeTick.

Operations on these objects are limited. Only four operations are necessary to deal with the data in the SNMP tree: SET, GET, GET-NEXT, and TRAP. The SET and GET commands write a value into the tree (possibly creating a new leaf node) or retrieve the value of a leaf node, respectively. The GET-NEXT command, applied repeatedly, traverses the tree from left to right. The objects labeled with letters in Figs. 2 and 3 are named such that GET-NEXT traversals of the subtrees would retrieve the objects in alphabetic order. This command allows a management station to determine the scope of the data available in the agent. Finally, the TRAP mechanism allows unsolicited notification of unusual events. In other words, an agent in charge of the data in an MIB can choose to alert a management station via a TRAP if conditions warrant.

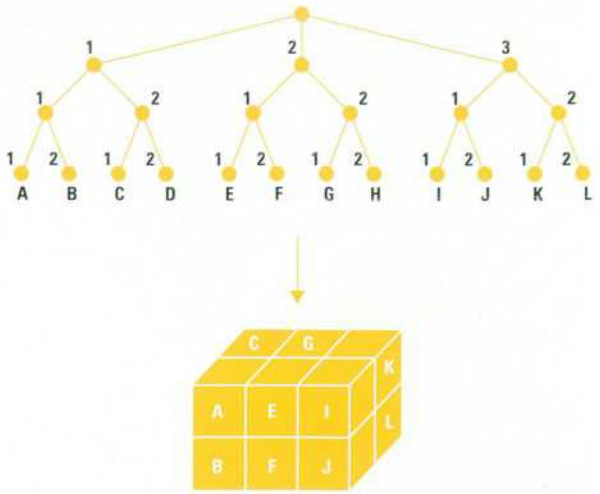


Fig. 3. Tree-to-3D-table mapping. Objects D and H are obscured in this picture.

RMON in the HP LanProbe II

The original HP LanProbe contained two NEC V50 CPUs which shared the processing load and communicated using shared memory. In addition to the two main CPUs, an Intel 82586 Ethernet chip also had access to the shared memory. For the new LanProbe II, it was decided to switch to a single, faster CPU to reduce the complexity of the design. The new Intel 82596 Ethernet chip shares the bus, but transfers data into memory using DMA. The design uses a PC-AT chipset from Texas Instruments, a 20-MHz Intel 80386 CPU, and the Ethernet chip.

The LanProbe II is also similar to PCs in that the DRAM is expandable using off-the-shelf memory SIMMs. The SNMP version of the LanProbe II allows 1M-byte, 4M-byte, and 8M-byte configurations. No additional configuration is needed when new memory is added to the probe—all table sizes expand dynamically as new memory is added.

In addition to the standard PC architecture, 128K bytes of battery-backed RAM holds configuration values and RMON configuration tables through a power cycle. This means that if power to the probe is interrupted while monitoring the network, the LanProbe resumes monitoring with the same configuration (filters, events, alarms, and other control structures) when power is restored.

RMON Overview

Although SNMP was first proposed as a simple alternative to the more complicated OSI management protocol CMIP, the RMON MIB proves that SNMP agents aren't limited to primitive systems.

The RMON MIB consists of nine distinct groups: statistics, history, alarms, host, hostTopN, matrix, filters, packet capture, and events. To be RMON-compliant, only one or more of these groups is required in an RMON agent. In fact, many RMON implementations do not contain all nine groups. The HP LanProbe II and a small number of other implementations, however, do support all nine. Fig. 4 illustrates the functions of each of the RMON groups.

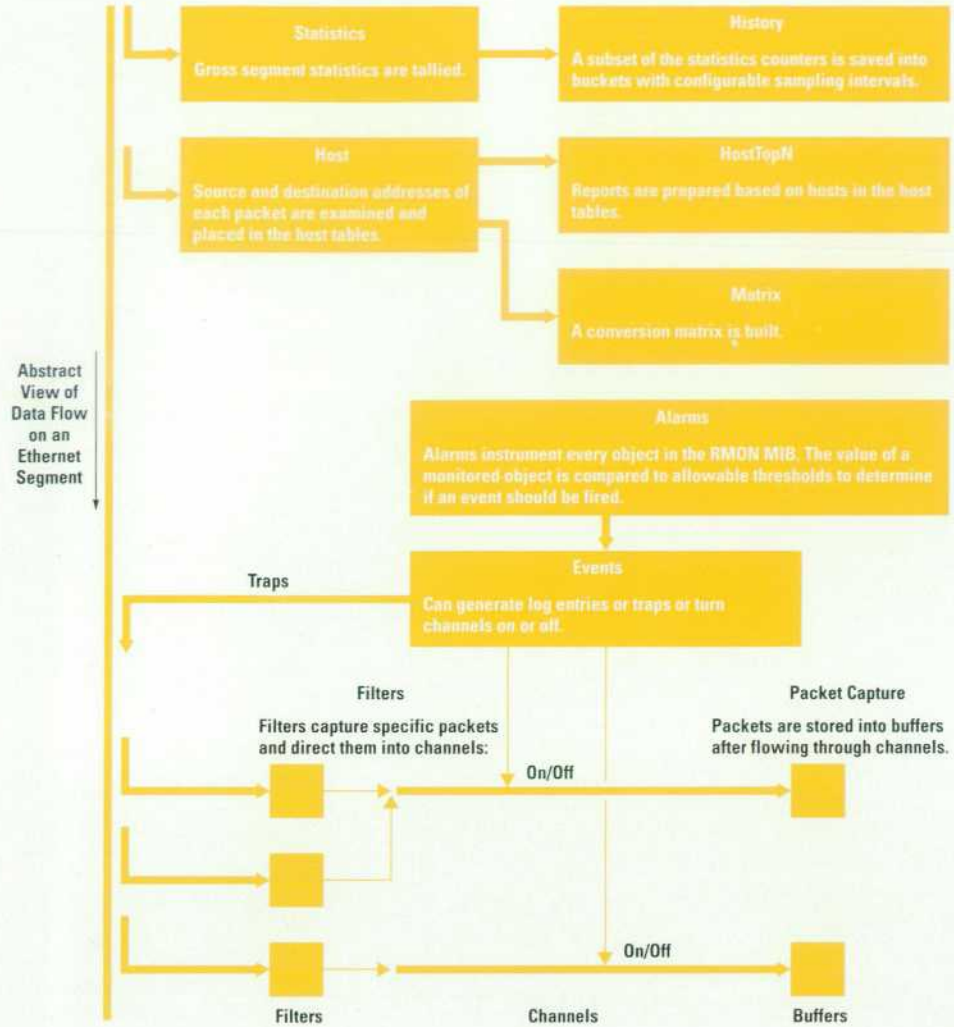


Fig. 4. Overview of an RMON agent.

An RMON probe analyzes every packet transmitted on the network. A great deal of work may be performed on each of these packets. It may be sorted by category and tallied in the statistics and history groups. The Ethernet source and destination addresses may be noted and appropriate entries created or updated in the host table and conversation matrix. The packet may match installed filters, triggering events in the probe and perhaps being captured in one of the probe's capture buffers. The 4M-byte HP SNMP LanProbe (Release A.00.00) keeps track of a maximum of:

- 8 network activity history reports
- 7500 hosts
- 10 separate reports showing top transmitters, receivers, etc.
- 30,000 separate conversations between hosts
- 50 objects with settable alarm thresholds
- 50 packet filters funneling into 30 channels, each of which can store packets in one of 12 independent capture buffers
- 1500 log entries of network events.

The structure of the MIB produced by the Internet Engineering Task Force (IETF) contains a number of similarities to the functionality provided by the original LanProbe. The statistics, history, host, hostTopN, filter, and packet capture groups all have direct analogs in the original LanProbe, although each of these groups expands on the capabilities in that probe.

The nine groups in the RMON MIB are described in more detail in the following paragraphs.

Statistics. This group provides an overall view of current network activity. Sampled values include octets, packets, error counters of various types, and packet size distribution. Although this group consists of a table with one row per network interface (Fig. 5), the table is limited to one row in the LanProbe implementation since only one interface is

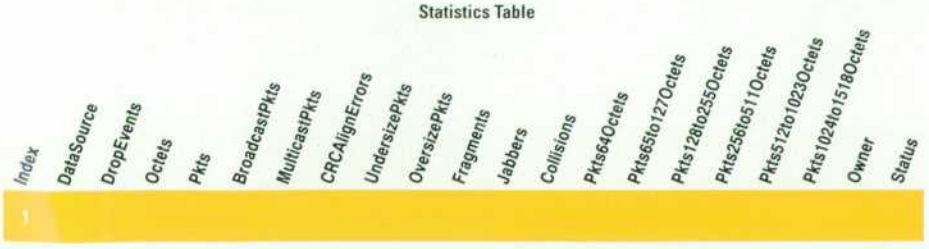


Fig. 5. RMON MIB statistics group.

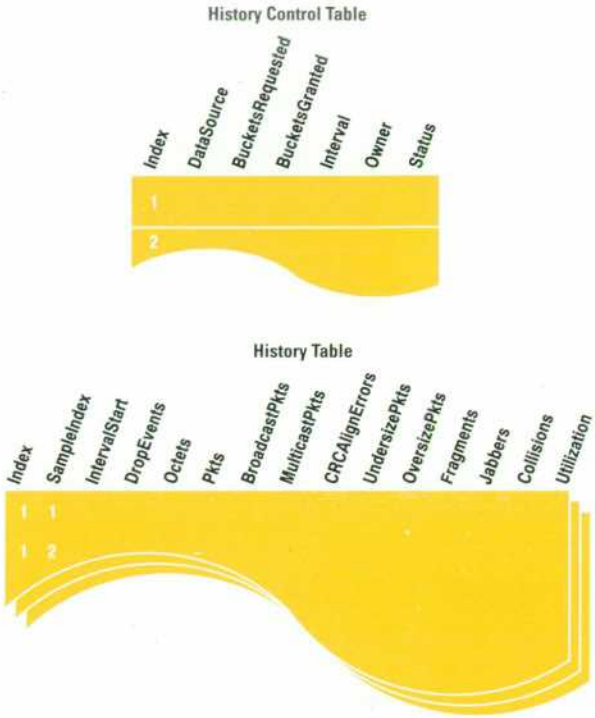


Fig. 6. RMON MIB history group. The history control table contains one row per history study. The history table is three-dimensional; it contains one two-dimensional table for each row in the history control table.

available on the probe. In effect, the statistics group provides a snapshot of current LAN activity.

History. As its name implies, the history group provides a mechanism for abstracting and preserving the statistical data collected by the statistics group in the probe (see Fig. 6). The user can specify a number of separate history studies to be collected simultaneously from one or more network interfaces, each with a different interval. For instance, two studies with intervals of 30 seconds and 1 hour can be created to collect data from the same network interface. The study containing 30-second interval measurements can be displayed in near real time on a graph showing network activity while the 1-hour measurements can be uploaded and stored for future use to study daily or monthly activity for network planning purposes. The number of intervals saved in each study can be set by the user and is referred to as the number of buckets in that study. The 4M-byte HP RMON LanProbe can keep track of 8 studies simultaneously, with a total of 550 buckets.

Alarm. The alarm group (Fig. 7) acts as the RMON probe's watchdog. Each row in the alarm table instruments an object in the probe's MIB. When the value of that object exceeds or drops below a specified threshold, an event is activated. For this reason, the event group must be implemented in any probe that includes the alarm group. Each threshold can be specified to be either an absolute value or a delta value, delta meaning the difference between the previous measurement of the object and the current measurement. This distinction is necessary since many measurements in the RMON MIB (and all those in the statistics group) are absolute counters zeroed at boot time and incremented thereafter until the counters roll over. The alarm feature allows

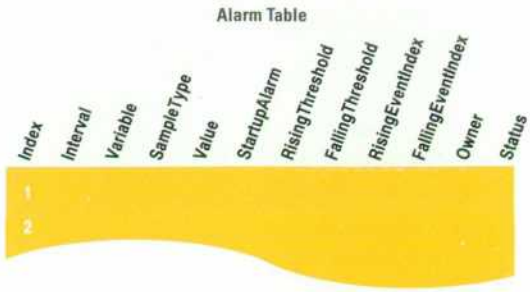


Fig. 7. RMON MIB alarm group. The table contains one row for each object being monitored.

a network manager to set thresholds on objects such as etherStatsBroadcastPkts (the number of broadcast packets detected to date) and to be notified via the trap mechanism when the object exceeds given limits. Alarms can be rising, meaning that an event is generated when a certain (rising) threshold is exceeded, falling, meaning that an event is generated when the object's value drops below a certain (falling) threshold, or both. Rising and falling thresholds are also used to rearm an alarm to prevent the same alarm from firing repeatedly when network activity causes a threshold to be crossed and recrossed rapidly.

Host. Network managers commonly need statistics detailing each node in the network. If errors per second suddenly rise sharply, it is useful to know whether any one particular host is responsible for the increase. Therefore, the host group provides a table of hosts for each network interface available on the RMON probe. Each table contains statistics on a per-node basis, including transmitted and received packets and octets, and transmitted errors, broadcast packets, and multicast packets. In a fashion similar to more common databases, each host table has two separate views, in which the same data is displayed in different ways. The first view is indexed by each host's network address, providing fast access to a particular host's statistics if the network address is known. This table is sparse in the sense that there can be, for example, up to 96 empty rows in the table between rows containing host 080009-002003 and host 080009-002100. The second view is indexed by discovery time order. That is, hosts are entered in the table in the order in which they are discovered by the probe. This table is therefore densely packed, and is useful for management applications wishing to retrieve a list of all hosts without regard to host address. Fig. 8 shows the host tables.

HostTopN. The hostTopN group provides just that—a number of lists of the top N hosts in a given host table, with ordering criteria configured by the management application (see Fig. 9). A control table contains settings for each top-N list (or report), with one row per report. The control table holds information on what value the report should be ordered by. Transmitted or received packets and octets, transmitted errors, broadcasts, and multicast packets are all valid measures by which to sort. Each row in the control table also specifies the interval over which data for the report is collected. When the collection period is finished, a new hostTopN table is created containing a sorted list of values and host addresses. This information is used to discover hosts transmitting unusually large amounts of traffic or errors,

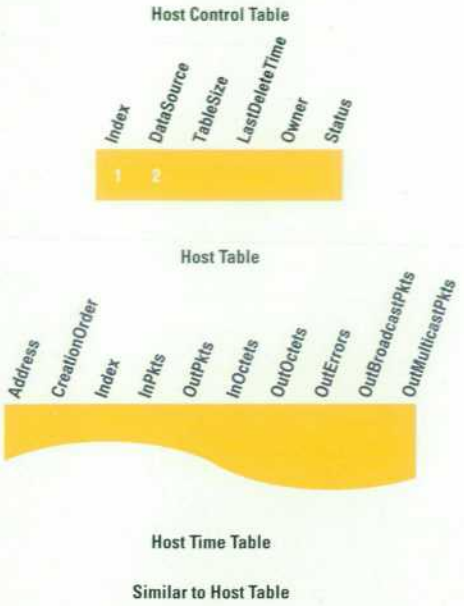


Fig. 8. RMON MIB host group. The host table is a sparse table. It is indexed by host address, so not every row will exist. The host time table is similar to the host table but is indexed by CreationOrder instead of Address; hence it is densely packed.

providing the network manager with an overview of network anomalies at a glance.

Matrix. The matrix group characterizes conversations between hosts on the network. It consists of three tables: a control table containing information on a per-interface basis, a table presenting the data from a transmitter-oriented view (the source-destination table), and a table presenting the same data from a receiver-oriented view (the destination-source table). Like the host tables, these are based on the network address of the hosts. Fig. 10 shows the matrix control table.

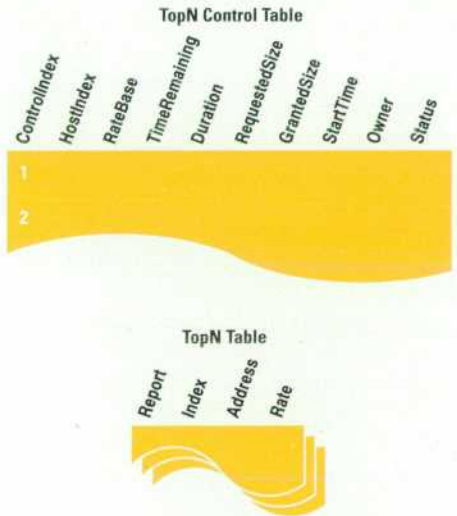


Fig. 9. RMON MIB hostTopN group. The TopN table is three-dimensional, containing one two-dimensional table (report) for each row in the TopN control table. The rate in each report may refer to the number of packets sent or received or the number of octets, errors, broadcasts, or multicasts.

Fig. 10. RMON MIB matrix group. The two data tables accompanying the control table contain identical data, but one is indexed by source address and the other by destination address.

Filter. Packet filters and channels are included in this group. Filters identify which packets are of special interest to the network manager. Packets that pass the filter test are funneled into channels, allowing a variety of actions for each such packet. Acting like a pipe in the network plumbing, channels can be turned on or off, allowing or restricting packet flow. The valve providing this control is the channelDataControl variable. When it is set to ON, packets can flow through the channel into packet capture buffers or can trigger events, which can in turn create log entries or turn on or off other channels. Whether a channel is ON or OFF, another variable (channelMatches) counts the packets that have been directed to it. A number of filters can be associated with each channel. Any packet that is accepted by at least one of these filters is directed to the channel. Channels can choose instead to accept only packets that fail all of the attached filters. Using combinations of these conditions allows a great deal of flexibility. Fig. 11 shows the filter group.

Packet Capture. This group (see Fig. 12) consists of buckets in which the packets flowing through channels can be stored. A number of capture buffers can be actively storing packets at the same time. Each of these capture buffers can be set to lock or wrap when full, that is, the buffer can stop collecting packets when full or it can return to the beginning of the buffer and overwrite the oldest packets captured. Since a wrap-when-full buffer always contains the most recent packets captured, this feature can be used as a network "black box" similar in function to the endless loop recorders found in airplanes. When the network crashes or misbehaves in some way, a detailed history of LAN activity is available for downloading and decoding for the network

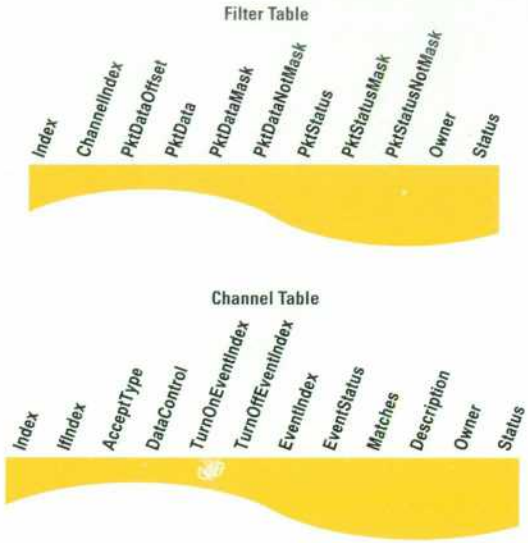


Fig. 11. RMON MIB filter group.

manager. To allow a large number of packets to be captured, buffers can be configured to save only a "slice" of each packet. This can be very useful since most of the interesting information in each packet is located in a relatively small header at the beginning of the packet.

Event. The event group (Fig. 13) provides a means to associate actions with network conditions. Events can be triggered either by packets flowing through channels or by alarms. In turn, each event can fire some action within the probe such as creating a log entry or, if the event is important, a trap. Events can also turn channels on or off, acting as packet capture triggers. Since events can generate log entries, a log table is also contained in this group. In it, a

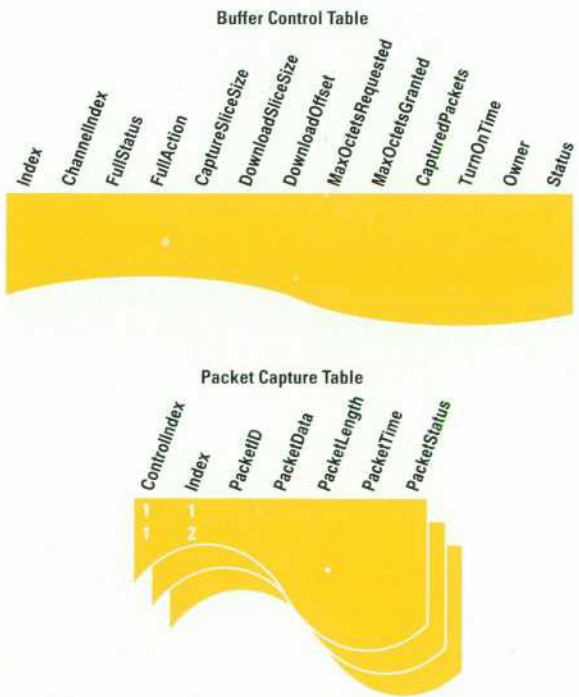


Fig. 12. RMON MIB packet capture group. Each row of the buffer control table specifies a different capture buffer. The packet capture table is three-dimensional, containing several two-dimensional tables, each of which is "owned" by a row in the buffer control table.

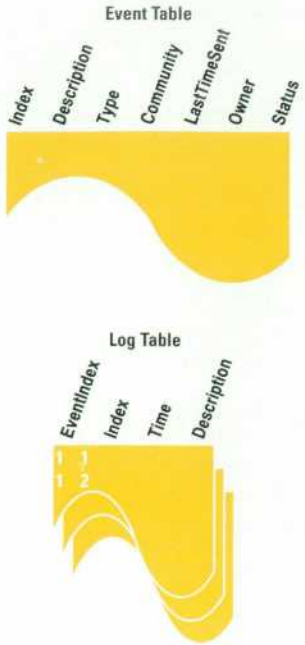


Fig. 13. RMON MIB event group. The log table is three-dimensional. Each subtable is owned by a separate event in the event table.

number of log entries can be associated with a single event. This group provides a great deal of flexibility for a management application to customize the behavior of an RMON probe.

Programming an RMON Agent

In some sense, a network manager must program an RMON agent before the agent can begin to provide useful data. As a simple example, history studies must be configured in the history control table before the agent will begin to build reports.

A more interesting example involves setting up the probe to monitor the network for broadcast storms, capture them as they occur, and store the data for later perusal. To perform this duty, the probe must constantly collect packets from the network, discarding the oldest and replacing them with new packets. Filters must be installed that capture every packet and direct it down a channel. The channel, in turn, must be directed into a packet capture buffer created for this purpose. The capture buffer is set to WRAPWHENFULL (creating essentially a circular buffer of the latest packets) and is set to some limited size to reduce the total number of packets held at any one time, say 100K bytes of data. The channel's DataControl setting must be ON to allow packets to flow. Since the beginning of a broadcast storm is most interesting, we need some method of shutting the channel off after the start of the storm to preserve packets in the buffer (presumably those that caused the storm). To do this, we'll set an alarm to trigger on the etherStatsBroadcastPkts counter in the statistics group. When the rate at which that counter rises exceeds 20 broadcast packets per second, the alarm will trigger an event we've previously created. That event logs the activity (etherStatsBroadcastPkts has exceeded 20/second) and sends a trap to a management station to alert it to the problem. Meanwhile, triggering the event has turned the channel off and packets stop flowing into the capture buffer. Later, the network manager can examine network activity trends over

the course of the storm by looking at data in the history group, then upload and decode the captured packets, including whatever packet or packets initiated the storm.

In a different scenario, packet filters capturing packets of a given protocol can be directed into a channel monitored by an alarm. When that protocol's activity rises or falls outside given thresholds, logs or traps can be generated and other channels turned on or off. Since alarms can be created to monitor any object in the RMON MIB, the number of possible useful agent configurations is limited only by the imagination of the network manager.

Future Directions

Although the RMON MIB was originally created for Ethernet interfaces, the rise in popularity of the token-ring protocol will spur the need for similar monitors for these networks. There is work currently in progress that will produce an RMON MIB specification for token-ring networks. In addition, SNMP version 2 has been proposed and is currently being evaluated by the IETF. It contains several features missing in SNMP and useful for an RMON agent, including security, encryption, and bulk data retrieval.

Conclusion

Despite the "simplicity" of SNMP, the RMON MIB demonstrates that SNMP agents can be complex and offer a rich feature set. This complexity is necessary to allow the agents to monitor the network in a useful manner. Because of the programmable and inexpensive nature of RMON agents, a single management station can observe and manage a large network with many segments.

A full description of the RMON MIB is contained in RFC 1271 and can be obtained via ftp from venera.isi.edu. Interested readers can be added to the RMON MIB mailing list by mailing to rmonmib-request@jarthur.claremont.edu.

Acknowledgments

I would like to thank Gigi Chu and Gary Ellis for their guidance and leadership throughout the project. Reva Bailey and Judy Tsai both contributed a great deal to the success of the LanProbe II, and Fill Denton deserves much credit for the hardware design. Project managers Don Manoogian and Sud Verma initiated and concluded the project, respectively. Thanks also to Steve Hand, Steve Witten, and Lyle Weiman, who worked on the Probe Manager user interface and had the pleasure of discovering bugs in our code. Finally, thanks to the entire IETF Remote Network Monitoring Working Group for their efforts in seeing the RMON MIB through the standardization process.

References

1. S. Waldbusser, *Remote Network Monitoring Management Information Base*, RFC 1271, Carnegie Mellon University, November 1991.
2. V. Cerf, *IAB Recommendations for the Development of Internet Network Management Standards*, RFC 1052, NRI, April 1988.
3. W.W. Crandall, "Design Challenges for Distributed LAN Analysis," *Hewlett-Packard Journal*, Vol. 43, no. 1, February 1992, pp. 66-76.
4. M. Rose and K. McCloghrie, *Structure and Identification of Management Information for TCP/IP-based Internets*, RFC 1155, Performance Systems International, Hughes LAN Systems, May, 1990.

The HP 64700 Embedded Debug Environment: A New Paradigm for Embedded System Integration and Debugging

The HP 64700 embedded debug environment gives embedded system developers complete access to state-of-the-art real-time measurements and controls in addition to C and C++ static debugging capabilities on HP and Sun workstations.

by Robert D. Gronlund, Richard A. Nygaard Jr., and John T. Rasper

Hardware and software development for microprocessor systems embedded in other systems and products requires sophisticated emulation, code development, and analysis tools. The HP 64700 embedded debug environment (see Fig. 1) is an emulation interface system designed to create a new debugging tool paradigm for embedded systems developers and integrators who use emulators for today's powerful 16-bit and 32-bit CISC and RISC microprocessors. The major contribution of this system is its ability to provide not only the expected static, C and C++ language debugging capabilities for large embedded systems, but also easy access to an extensive set of modular real-time control and analysis capabilities, all with a common, easy-to-use X11 OSF/Motif user interface. This real-time capability together with the powerful HP 64700 emulation hardware system sets this embedded debug environment apart from other emerging graphical interfaces and integrated software tools focused on embedded software development and debugging.

Embedded System History and Trends

There has been a surge in the application of microprocessors to enhance the operation of products that most people do not think of as computers—products ranging from remote controls to the space shuttle. These products contain microprocessor systems embedded within them for various reasons including cost, usability, and flexibility. Hence an embedded system is any microprocessor-based system that is essential to the operation of a product not thought of primarily as a computer.

Fig. 1. Full HP 64700 debug environment session.

The Value of Usability

Emulators are complex instruments that combine sophisticated real-time analysis with microprocessor run control, memory management, and symbolic software debugging. In addition to understanding the target hardware and application software, emulator users must also master the nuances of emulator configuration, symbol referencing, trace sequencing, and other emulation control functions. As a result, most emulators are difficult to use.

The original HP 64000 design team was concerned about this complexity, and made ease of use a key element in the first HP 64100 system. The result was a user interface based on softkeys. It was menu-driven and offered a hierarchical, directed-syntax command structure. The ease with which users could sit down in front of an HP 64100 and become productive helped launch this successful product. Since then, developments in workstations and personal computers have established new expectations for human interfaces. At the same time, application program size has grown exponentially, redefining the kind of information users need to access to debug and integrate embedded systems.

Beginning in 1989, several research projects were initiated to better understand the changing requirements of embedded designers. A telephone survey of 72 emulator users was conducted by the Gediman Research Group in 1989. One of their key findings was that "development systems are criticized for lack of speed ... and difficult/poor user interfaces." Their findings were troubling; many owners had come to view emulators as a tool of last resort because of their complexity. A customer visit program was conducted by R&D and marketing experts in 1990. They interviewed 20 users of the most sophisticated HP emulator and found that 19 did not use the full capabilities of the instrument, largely because of interface and configuration complexity. It became clear that the effort to create and market a capability is wasted if users cannot or will not access it. As a result, a product priced on the basis of potential performance will fall short of expectations when the marketplace assigns a value to it consistent with realized performance. This finding led one designer to remark, "If a feature isn't understood, it doesn't exist."

Two important studies helped measure the importance of ease of use. The Technology Research Group conducted an embedded CASE study in 1989. When asked what role 14 key attributes played in tool selection, 545 embedded system professionals ranked ease of use most important. The second study was conducted by Oasis Computer Solutions in 1990. This study measured the importance of ease of use against 17 other top-level requirements. Ease of use was ranked second, just after real-time execution. Notably, source-level debug capability ranked third.

As embedded software grew in complexity, the tools used for code development and debug grew more sophisticated. Compilers replaced assemblers, and C debuggers, as a companion to emulator interfaces, were rapidly growing in popularity. The Oasis study and the actions of our competitors confirmed the importance of C debuggers to embedded designers. Our C debugger had a different look and feel than our emulator interface, and worse, was not easy to use together with our real-time measurement capabilities.

We were convinced that user interface improvements were key to success, but still needed to choose a course of action from several alternatives being considered at the time. In late 1989 we commissioned a SUMM study. The SUMM (Single Unit Market Model) study is a proprietary choice modeling technique of Eric Marder Associates, Inc. It offers a mathematical description of how customers choose products based on their desires (user needs) and beliefs (vendor perception). When applied to customer survey data collected in the current marketplace, it predicts what product customers will choose in response to changes in product and/or market strategies. Based on this research, we aggregated user needs against key competitors, and created a program in January of 1991 to improve the usability, consistency, and integration of the emulator and source debugger interfaces for the HP 64700 Series emulators. The result of that effort is the debug environment described in the accompanying article.

John D'Alessandro
Market Research Analyst
Colorado Springs Division

Throughout the past decade, embedded systems have grown exponentially in size and complexity, driven by the same silicon technology that has created low-cost, high-performance personal computers. Just as this complexity has challenged developers of computer systems and software, it has also challenged embedded developers. These developers are often required to guarantee an exacting standard of real-time responsiveness to meet operational requirements. For many embedded developers, real-time really matters.

Microprocessor emulation systems first appeared in the late 1970s and early 1980s to address the system integration needs of embedded developers.[1] At that time the only microprocessors commercially available were relatively simple 8-bit implementations. The embedded software usually consisted of a few hundred to a few thousand lines of assembler code that controlled the hardware directly and was often an afterthought in the system implementation. This was also a natural consequence of the limited address space and primitive language tools that were available with these processors. The challenge that has emerged in today's large, high-performance embedded systems is the synthesis of complex, structured software systems with powerful hardware to meet real-world demands and stimuli in real time.

In parallel with this evolution of real-time emulation capability, another class of debugging tool evolved that has also had a revolutionary impact—the high-level language debugger (referred to hereafter as the C debugger because of the prevalence of the C language in embedded development). The need to create and implement complex software systems has made abstraction essential, to keep developers from being overwhelmed. Abstraction has mainly come in the form of high-level languages. In embedded development the language that has become far and away the most popular is C, with C++ emerging as its logical successor.

While emulation interfaces were initially focused on dynamic understanding of machine states and assembler language (and remain very capable at this task), C debuggers were initially implemented on minicomputers to allow developers to understand the operation of their software at the same level of abstraction at which it was written. The capabilities of most modern debuggers include stack backtrace display, high-level data typing, complex data structure displays, high-level expression evaluation, and source code referencing.

Another emergent trend in larger embedded systems has been the real-time operating system (RTOS).[2] This has served as an important point of leverage for embedded developers by creating an additional level of abstraction, much like high-level languages. While the RTOS vendors have endeavored to provide specialized debuggers capable of understanding the state of stopped (static) systems in terms that developers can understand, they have been notably lacking in the provision of debuggers that can provide nonintrusive,

abstracted views of real-time operation—a seemingly logical requirement for a real-time system.

Another trend is the demand for usability. Products that are laden with all the "right" features but are effectively inaccessible will likely fail in the marketplace (see "The Value of Usability" on page 91). This has been supported by the emergence of graphical interface systems aimed at making powerful applications faster to learn and easier to use.

Finally, there has been an emerging trend in the area of software tool frameworks and integrated development environments. Two notable leaders in this area are Borland's integrated development environment on the PC and the HP SoftBench CASE framework for UNIX*-system-based workstations. The focus of these systems has been the automation of common tasks for software development teams and increasing the usability of common tools by reimplementing them using graphical interface technology.[3]

HP 64700 Embedded Debug Environment

Drawing on its roots in the original highly integrated HP 64100 microprocessor development system, the HP 64700 embedded debug environment synthesizes elements from all of these sources to form a new paradigm that realizes the most advanced implementation to date of the "electronic workbench" vision for embedded systems developers as described by Chuck House in 1979.[4] Key capabilities of the system include:

- A full complement of emulation-based debugging and analysis tools including a C/C++ debugger, a real-time emulation control and state analysis interface, a true real-time software performance analyzer interface, and a timing waveform analysis interface.
- The ability to shift debugging and analysis views and tools rapidly to zoom in and out on the really hard debugging problems—notably problems in the real-time domain.
- Advanced, X11 OSF/Motif user interfaces that conform to the IBM Common User Access standard and provide fast, consistent operation, powerful support for symbolics, intuitive system configuration, and powerful support for user customization.
- An adjunct set of tools including a debugger simulator, a branch test tool, and a dynamic real-time operating system analysis tool (see "A Real-Time Operating System Measurement Tool" on page 97).

The paradigm shift embodied by the debug environment is the leveraging of tool integration to allow quick perspective shifts to the view most appropriate to testing the current debug hypothesis—from static C and C++ debugging to true real-time measurements including state, timing, and software performance analysis. Because of the modular implementation of the tools, developers new to the system can start by simply using the window they feel most comfortable "driving" the emulator with. Hardware designers might choose the emulator/analyzer, software designers might prefer the C debugger, and hardware-software integrators working on complex problems might start with multiple windows simultaneously (see Fig. 1). The system provides headroom for future growth in a modular fashion: add a new analysis board to the HP 64700 cardcage and add a window to tap its capabilities, or just add a window for tools not requiring hardware instrumentation.

Emulation Interface Evolution

From their roots in the early emulation systems and hosted debugging tools, emulation interfaces have evolved into two different primary types: the emulation/analysis interface and the C/C++ debugger. The focus of each of the interfaces was different, as shown in Fig. 2. With the debug environment these interfaces are now treated as complementary rather than competing tools and can be run synchronously and simultaneously with the same emulator.

Along the way, HP developed other new analysis technologies, one of which was the software performance analyzer.[5] The software performance analyzer system was a totally new concept in the mid-1980s. It provided the hardware and the user interface for sampled performance measurements. The the newest generation of this technology, the HP B1487 software performance analyzer (see article, page 107), adds new, true real-time, nonsampled duration measurements.

The contribution of the new HP 64700 debug environment is the realization that all of these capabilities are important to the integrators of embedded systems and must be made understandable and quickly accessible, especially for demanding projects under intense time-to-market pressure. All of the elements of the system automatically work together and have a common look and operation model, so software developers can more easily explore the use of emulation hardware breakpoints and real-time software performance analysis to examine even algorithmic problems, instead of spending time adding elaborate debugging code (such as printfs) to their systems. Hardware developers can become more productive by writing diagnostic and test code in C.

The rising value of making the embedded software developer more productive has lead us to define the debug environment in a somewhat broader scope than just that of traditional emulation. The environment is structured to allow contributions to extend in scope beyond those that use only the emulation cardcage.

Emulator/Analyzer	C/C++ Debugger
Debugs assembly code (can display C/C++ source lines)	Debugs C programs
Deals with real-time problems	Can only look at static (stopped) systems
Understands bits and bytes	Understands and displays C data types and expressions
Excellent measurement capabilities including state and timing traces	Only static stack backtrace
Measurements can be taken and displayed without disturbing the real-time operation of the target system; target runs at full speed by default	Must stop target execution to measure
Excellent emulator configuration control	Emulator configuration "a necessary evil"

Fig. 2. C/C++ debugger versus emulator/analyzer interface comparison.

The Debug Environment Connection to HP SoftBench

HP SoftBench provides a framework in which a variety of software development tools can be placed.[1] Once properly connected to the SoftBench environment, the tools communicate with each other by means of messages. Communication via messages allows automation of more tasks of the development process, and thereby speeds up the development process. The HP 64700 debug environment described in the accompanying article can be connected to the SoftBench development environment to provide a SoftBench embedded development environment.

The primary tools of the SoftBench embedded development environment consist of the SoftBench editor, the SoftBench static analyzer, the SoftBench build tool, and the HP 64700 embedded debug environment (including the HP Branch Validator). A user can also add such tools as a configuration management system, SoftBench mail, or a complexity analysis system. Almost any combination of SoftBench tools can be added to the SoftBench embedded development environment.

Connection of the debug environment to SoftBench is made by using the SoftBench Encapsulator tool, which provides a SoftBench message interface for the debug environment.[2] The SoftBench message interface allows the debug environment to receive debug messages such as step, run, set breakpoint, load executable, and so on, and to send edit, build, static analysis, and other messages to the other SoftBench tools.

The SoftBench message interface for the embedded debug environment (**Debug64700**) acts as a translator between the SoftBench and debug environments (see Fig. 1). SoftBench messages sent to **Debug64700** are translated to debug environment commands and sent to the command parser of either the debugger or the emulator or both, depending on which interface is running. Once parsed, the commands are executed as if the user had entered them directly. The debug environment, conversely, can forward commands to **Debug64700**, which translates the commands into appropriate SoftBench messages. Simple syntax allows a user to forward almost any command or message to the SoftBench environment.

In addition to message translation, the **Debug64700** encapsulation provides a startup interface to the debug environment. This startup interface simplifies the process of choosing a debug environment system and deciding which interfaces to start. The user simply selects a system to start and then the interfaces to start (debugger, emulator, or software performance analyzer). Pressing the start button will start all of the requested debug environment interfaces and complete the connection between the SoftBench and debug environments.

The **Debug64700** message translator and startup interface is a simple program. The Encapsulator tool made it possible to code and test this connection to SoftBench in about two months. We were able to produce a prototype within one week. Completing the final form of the interface was simply a process of modifying the prototype. Very little of the program was wasted or thrown away. In fact, our initial intent was to use the Encapsulator only to create a prototype startup interface, but as has happened in the past,[3] the prototype was so usable that we elected to extend the prototype rather than create a new interface from scratch. This is the second time that we used the Encapsulator to create a prototype and discovered that it would take less effort to extend the prototype than to create a complete implementation of an OSF/Motif-based interface. Our hats are off to the SoftBench Encapsulator, a tool that allows a user to design an OSF/Motif interface in a matter of minutes when hours of effort would be required without it.

References

1. M.R. Cagan, "The HP SoftBench Environment: An Architecture for a New Generation of Software Tools," *Hewlett-Packard Journal*, Vol. 41, no. 3, June 1990, pp. 36-47.
2. B.D. Fromme, "HP Encapsulator: Bridging the Generation Gap," *ibid*, pp. 59-68.
3. D.L. Neuder, "A Test Verification Tool for C and C++ Programs," *Hewlett-Packard Journal*, Vol. 42, no. 2, April 1991, pp. 83-92.

David L. Neuder
R&D Design Engineer
Colorado Springs Division

Fig. 1. **Debug64700** SoftBench connection window for the HP 64700 embedded debug environment.

The ability of the debug environment tools to connect with the HP SoftBench CASE framework is one of the keys to this implementation (see "The Debug Environment Connection to HP SoftBench" above). Fig. 3 highlights how the debug environment tools fit into a simplified embedded development lifecycle and work with the SoftBench environment if available. However, since we realized that not all customers would be users of HP SoftBench, the debug environment toolset is designed in such a way that the tools automatically form a powerful framework and coordinate among themselves using a technology specifically designed for this purpose. This highly responsive messaging capability has been a major system design challenge and subsequent sections detail this technology.

Some further aspects of this broader environment definition are of interest:

- The debug environment has built-in capabilities to form a "mini-SoftBench" in the areas of code editing, rebuilding, and reloading for users who have not adopted SoftBench.
- An optional C/C++ debugger simulator allows simulation of target system code operation to begin even before hardware availability.

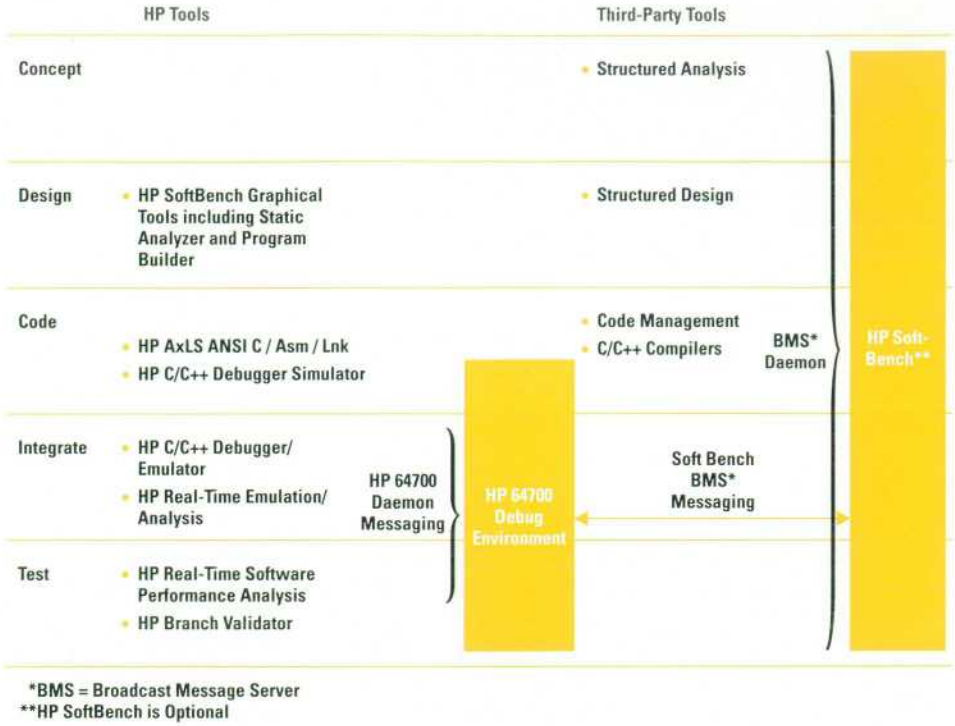


Fig. 3. The embedded development lifecycle and HP 64700 products.

- A graphical branch test tool called the HP Branch Validator gives accurate feedback on branch coverage in code executed by the emulator or simulator.[6]
- The debug environment allows large teams to share resources including emulators via the LAN. Ethernet-connected HP 64700 emulation cardcages can be controlled from across a building or even across the world via the global internet network.
- The debug environment runs on both HP 9000 and Sun SPARC workstations.

System Architecture

The debug environment uses a multiprocess architecture with a background daemon to coordinate the operation of one or more user interface processes. A debug environment session consists of one or more user interfaces all attached to the same emulator and coordinated by the emul700dmn daemon. The result is a highly interactive and flexible family of user interface products that can be joined together into a single session with the following capabilities:

- Separate user interface products attached to the same emulator operate concurrently and cooperatively.
- The customer can buy additional products later and add them to the system.
- HP can develop additional user interfaces and add them to a growing product family.
- Defects, maintenance, and enhancements are contained within the firewalls provided by the process boundaries.
- Overall performance is improved by using UNIX*-system concurrency to timeshare operations among the user interface windows.

The debug environment system architecture has its roots in the development of the HP-UX*-workstation-based user interface for the HP 64700 series of microprocessor emulators.[7] In turn, this was based on the interface pioneered by the HP 64100 microprocessor development station in 1979 and later ported to the HP-UX operating system in 1986.[8] This user interface operated in a character screen and used syntax-directed softkeys to prompt the user in the command line.

Two goals were established for the application of the original HP 64100 user interface to HP 64700 Series emulators: consistency in look and feel and support for modularity in adding new types of user interfaces.

At the time, additional user interfaces were envisioned to support hardware timing analysis, software performance analysis, and of course, C language debugging capability. All of these were to operate concurrently on the workstation host while connected to the same HP 64700 emulator cardcage. X Window support was limited to the use of terminal windows for concurrent operation of the character-based user interfaces. This product was released early in 1989 with the emulator/analyzer user interface only. The timing analyzer window soon followed. The debugger remained a separate product until the debug environment was released.

The debug environment architecture described here extends the original one to support the additional information sharing expected in an OSF/Motif environment while maintaining backward compatibility with the original products (see Fig. 4). Consistency and backward compatibility remained essential objectives. The efforts applied in the OSF/Motif-based user interface to maintain consistency with the earlier softkey-driven user interface are described elsewhere in this article. The underlying architecture that allows both OSF/Motif and non-OSF/Motif user interfaces to operate concurrently in a single session is described next.

Multiprocess Architecture

A multiprocess architecture presents several fundamental challenges when applied to a real-time instrument product such as an emulator. The user expects the "live" look and feel of a benchtop instrument front panel. This means that all parts of the user interface must update concurrently whenever new information becomes available and command

Fig. 4. The debug environment supports both the earlier softkey interface and the new OSF/Motif interface, represented here by a graphical emulator/analyzer.

entry must always be enabled. Supplying this operation requires solutions to several problems:

- Arbitrating access to the shared emulator and analysis hardware
- Distributing notification to all windows of changes to the state of the hardware and of the session
- Managing session information, including the current working directory, the name of the program loaded into the emulator, and so on
- Performing polling operations required for the session but not the exclusive interest of any one user interface. This includes polling for emulator and analyzer status and simulated I/O transfers
- Performing lengthy operations in background, such as uploading trace data and notifying the user interfaces upon completion.

The debug environment uses an event-driven messaging system coordinated by a daemon to conduct a session. This is shown in Fig. 5. It is similar to the broadcast message server used by the HP Visual User Environment[9] and the SoftBench system.[3] However, unlike the broadcast message server, the daemon in the debug environment also operates as a peer of its clients in its connection to the emulator hardware and in its data processing activities related to emulation hardware polling. Each process, including the daemon, has a direct connection to the communications channel attached to the emulator as well as a connection to the message channel anchored by the daemon. Let's discuss the communications channel first.

Emulator Communications

The physical channel to the emulator can be either a dedicated serial line or a network connection via sockets over the LAN. Communications are not funneled through the daemon to avoid the overhead of multiple process context switches and multiple buffer copies for each transaction with the hardware. Instead, a kernel semaphore is used to arbitrate ownership of the channel. Once the channel is obtained, ownership is retained until the complete transaction (or atomic set of transactions) with the emulator is finished. At this level the daemon is a peer of the other user interface processes with no special privileges or responsibilities.

Arbitration

Semaphore-based arbitration offers different advantages for serial and socket connections to the emulator. In the case of a serial channel, the semaphore keeps multiple processes from intermingling their traffic on the single physical channel. In the case of a LAN-based socket, this cannot happen. There is a different socket for each process connected to the emulator. However, one and only one socket can be active at any given time since the emulator serves each socket in a round-robin manner. This is because the emulator is inherently a shared resource that cannot be reentrantly accessed. By continuing to use a semaphore to control access to the emulator, each process blocks at the level of the semaphore operation until the emulator is available. This makes it much easier to define appropriate timeouts for each stage of a transaction. Blocking on the semaphore is allowed to continue indefinitely whereas blocking on the reply to a command times out to detect faults such as communications failures. In either case, the user can abort a transaction by pressing **Control-C** in the user interface.

In addition to the semaphore associated with the communications channel to the emulator, another semaphore is used as a count of how many processes are currently running in a session. This includes the daemon and all its clients. Each process increments the value of the semaphore when it starts. The operating system automatically decrements it when a process exits. This is used for status checking and as part of the locking system to prevent other users from accidentally disturbing a session in progress.

System Start

Most daemon-based systems require the daemon to be explicitly started by the user or the system administrator and to remain running until explicitly terminated. The start and shutdown are typically added to boot scripts that run when the workstation is rebooted. This was deemed to be unacceptable not only because of the additional complexity

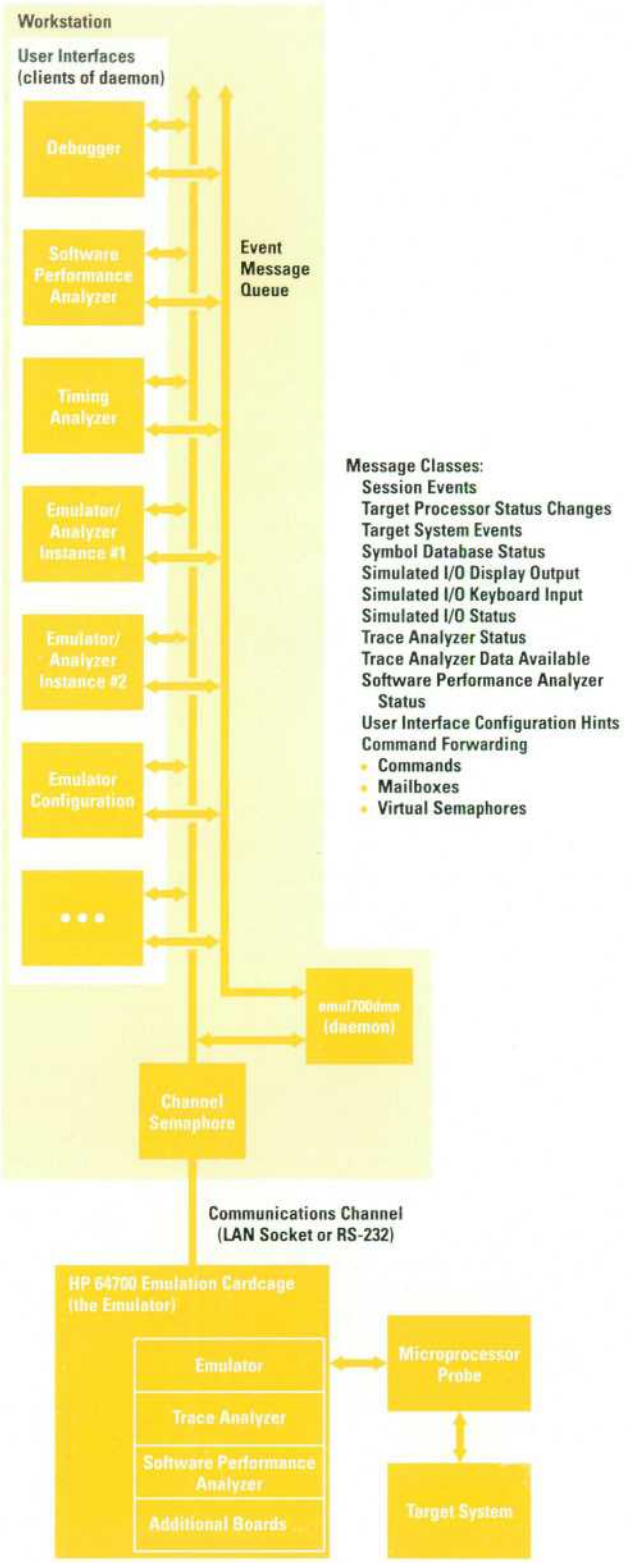


Fig. 5. Debug environment multiprocess architecture.

exposed to the end user, but also because of the system resources that would be tied up permanently by each daemon. Instead, the daemon for an emulation session is started automatically by the first client to run and exits automatically when the last client in the session ends. A third semaphore, directly associated with the daemon, indicates that the daemon is running. It has three states: none, starting, and running. This is used to detect and resolve race conditions that can occur if two or more user interfaces are started simultaneously and each tries to start the daemon. Once the daemon is running this semaphore is used to avoid the overhead of attempting to start another one.

Message Distribution Channel

The other channel in Fig. 5 is the message distribution channel. This channel is different from the communications channel in that the daemon has unique responsibilities for message distribution. All event messages are first routed to the daemon for processing. After processing the daemon forwards a copy of each (possibly modified) event message to each interested client. The client that generated the message does not act upon it until it comes back from the daemon. This is another aspect of pacing the session to ensure that all clients act upon all events in the same order. It also simplifies client design in that no distinction is made between events generated within a client and those generated elsewhere. In either case a client receives an event message and acts upon it.

Message Subscription

Which events is a client interested in? No client supports all of the features of the debug environment. Routing all events to all clients would result in much more message traffic than is necessary. Most messages would simply be ignored. Events in the debug environment are grouped into classes (see the list in Fig. 5). When a client starts, the only class it receives by default is the Session Events class. Events in this class include notification of session state and changes to the current working directory.

Clients subscribe to additional classes of events by sending a channel protocol message to the daemon listing the classes desired. Classes may also be deleted, but this feature is not used by any current debug environment clients. Two things happen when a client subscribes to a new class of messages. First, that client is added to the distribution list for that class. Second, a copy of the most recent instance of each event in that class is sent to the client. Some events are purely transitory (e.g., step taken) whereas others represent current values of retained state information (e.g., target processor status). It is the latter that are echoed to each newly subscribed client.

The client makes no special queries for this to happen. Instead, the client is automatically brought up to date with all other clients already in the session. From the perspective of the client there is no difference between a regenerated message and one being freshly distributed. In a query-based approach the newly started client would specifically ask the daemon for each piece of current state information it needed. This was rejected on the grounds that this would require a special body of software in each client and special support in the daemon for a one-time need. Ensuring that the client software was complete was seen as an ongoing source of maintenance and defects.

(continued on page 98)

A Real-Time Operating System Measurement Tool

The real-time operating system (RTOS) measurement tool of the HP 64700 embedded debug environment uses the emulation bus analyzer and the software performance analyzer to capture operating system software activity in real time. It works with the HP 64700 Series emulation trace analyzer and includes a specially developed inverse assembler that uses the trace display to show program flow information. The trace display is easily readable and includes a fully interpreted display of all parameters passed into and returned from the RTOS service calls along with any other pertinent data (see Fig. 1). The captured and displayed data is a series of memory writes to a data table. These writes can contain information about an operating system service call that was just executed or a task switch that just occurred.

The base measurement that is provided by the RTOS tool is the ability to trace task switching. Task switches are always a concern of an RTOS developer. When did they occur? In what order? Where within the application? The basic power of an emulation analyzer is appropriate for capturing this data.

To capture task switches, a feature common to RTOS kernels is used. This feature is the ability of the user to define a routine that will be called every time a task switch occurs (or when a task starts). These routines are called *callout routines*. A callout routine is defined in the kernel's configuration table by placing the starting address of a function in a specific place in the table. Whenever a task switch occurs, just before the new task is started, a subroutine call is issued that jumps to the defined address. The callout routines are passed two input parameters consisting of task control block pointers of the tasks that are being exited and entered. Within the task control blocks, a unique ID can be found for each task.

To provide measurement data, unique memory locations signifying a task's entry and exit are defined and linked into an application. A callout routine is written to take the pointer to the task control block being switched out of, find its task ID, and then write that ID to the exit location. A similar scenario applies for the task being entered. If the analyzer is set to capture only writes to the entry and exit locations, the result is a display showing the time any task was exited and the time any task was entered. If the analyzer is set to capture other states, they are shown relative to the entries and exits of the tasks and the flow of task execution can be followed. This provides a limited display of the RTOS interaction.

Many important events also occur at the C interface library. This is a standard piece of assembly code for an RTOS application, often provided by the RTOS vendor, that allows an application to make C function calls to the operating system. A function is provided for each available RTOS call to take the parameters off the stack and place them into registers. One of the registers, normally D0 for 68XXX family processors, is loaded with a function code and a trap instruction causes a jump to the RTOS kernel. The RTOS kernel looks at D0 to determine what call is being made and processes the data in the registers accordingly. Before a return to the function occurs—and it may not happen until other tasks have been run—registers are loaded with return values and the return to the function occurs right after the trap instruction.

Just before the trap to the RTOS occurs, when all of the registers are loaded with the input data, a simple move multiple instruction (MOVEM in 68000 assembly language) causes all of the data passing between the application and the RTOS to be written to a data table. The same thing is done on the return. When these writes occur, all of the data can be captured by the analyzer. Thus, all of the transactions between a task and the RTOS are available along with the times at which they occurred. To provide a memory destination for these writes, a data table was created. Thus, a fairly full set of data can be captured by the analyzer with a single specification of the data table range.

The captured data is displayed through an inverse assembler. Although called an inverse assembler, there is no reason why it needs to output assembly code, and in this case, it is more of an inverse interpreter. The interpreter reads through the set of states captured by the analyzer and is able to display the data found in any order and with any chosen ASCII text. With this technology, a trace can be set up to capture all of the writes to the data table, interpret them, and display them in a very readable form. On the display there can be single lines for task entries or exits, lines for every function call and return showing all of the input and output parameters, highlighted error return lines, parameter values decoded into English equivalents, and even stack information. Enough information can be displayed to give a user a complete knowledge of the interaction between an application and the RTOS kernel.

Display of the RTOS trace is instantly available and a user is able to switch between normal trace display and RTOS display. A user displaying a captured RTOS trace (which consists of writes to the data table) using the normal inverse assembler will only see meaningless data and address values. Using the RTOS inverse interpreter, function calls with parameters, task switches, and stack information are displayed.

The data table created for the RTOS measurement tool has memory locations for each defined function call to the RTOS. There are entries for calls to the function and returns from the function. Each entry is allocated as much memory as is needed to hold the full set of parameter data. Specific addresses for each function entry and exit are defined so that when the interpreter is interpreting the captured state addresses, it knows exactly which function's data it is going to display. Beyond the entries for the functions, there are extra entries for task entry and exit, operating system overhead, tool intrusion, clock ticks, stack information, and user-defined entries.

The stack information is an important addition to the capability of the RTOS tool. Stacks are always a concern, and when implementing RTOS applications, a user has to declare the size of each stack for each task. Stack problems are always difficult to debug, but when multiple stacks are used within a single application, the problems multiply. To instrument the supplied code for stack tracking, memory space is reserved for each task. Each task's space is referred to as a bucket. Within each task's bucket, there is space to store stack information. A task start callout routine stores the stack size and stack base in the bucket. When a task switch occurs, the

Trace List Label: Base:	Offset = 0 Real-Time Operating System with Symbols	More Data Offscreen Time Count Relative
after	NON-RTOS: addr=2F58C data=00000000	
+001	-> sm_ident(name='sem1',node=00000000)	16.4 µs
+005	<- sm_ident(SMid=000C0000)	124.0 µs
+009	-> sm_p(SMid=000C0000,NOWAIT)	19.3 µs
+015	<- sm_p()	111.0 µs
** Return code=66: SEMAPHORE NOT AVAILABLE		
+017	-> ev_receive(evs=B:0001, WAIT/AND,FOREVER)	1.01 ms
+023	STACK BYTES LEFT ON EXIT: Supr 00000100 User 000001A8	120.0 µs
+031	Exiting Task : 'paal'	8.76 µs
+033	ENTERING TASK: 'sllk'	11.1 µs
+035	STACK BYTES LEFT ON ENTRY: Supr 00000100 User 000001CC	2.8 µs
+043	<- ev_send()	95.7 µs
+045	-> q_receive(qid=00040000:'cssl',WAIT,FOREVER)	39.1 µs
+053	STACK BYTES LEFT ON EXIT: Supr 00000100 User 000001B4	151.0 µs
+061	Exiting Task : 'sllk'	8.76 µs
+063	ENTERING TASK: 'cosp'	11.1 µs

Fig. 1. Real-time operating system measurement tool trace display.

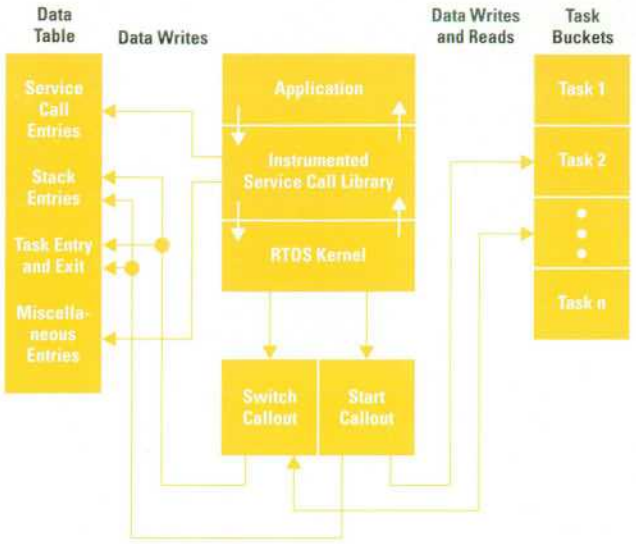


Fig. 2. Real-time operating system measurement tool data flow.

switch callout routine gets the stack information from the bucket and the stack information from the current task control block. With both sets of information, the routine can write to the data table the current stack status. Depending on the RTOS kernel being used (which affects the data available at start times and switch times), the number of bytes used on the stack, the number of bytes left on the stack, or the current stack pointer value can be written to the special entries in the data table. This data can then be displayed by the interpreter every time a task switch occurs, which is very useful for following stacks and determining where a problem might be occurring. By editing the supplied command files, a user can track a specific task from the point where its remaining stack becomes too small and then follow the actions that cause a possible stack overflow.

Fig. 2 shows the data flow of the RTOS measurement tool.

Software Performance Analyzer Support

The HP B1487 software performance analyzer (see article, page 107), a plug-in card for the HP 64700 emulation system, can provide valuable operating system-level profiling measurements. Beyond providing dynamic histograms of task execution, the number of times each task is run can be displayed, providing valuable information on system thrashing. Also, the number of times each separate operating system service call is invoked from an application can be tracked, helping to isolate bottlenecks from overused system features.

For software performance analysis support for the RTOS tool, a little extra instrumentation needs to be done. The software performance analyzer uses separate memory locations for the start and end of each interval it is measuring. Since each task must be defined as a unique interval, each task must have its own unique start and end memory entries in its defined bucket. (This data area is application dependent and must be modified with the application's task IDs.) Writes to these entries signify a task being set to running or blocked respectively. The callout routine is able to write to these unique locations depending on which tasks are switching. By using the task IDs as an index to the task data buckets, the switch callout routine can write to the entry and exit positions of a task's bucket, effectively starting and stopping the software performance analyzer duration timer for each task that starts running or becomes blocked, respectively.

With this instrumentation, the software performance analyzer can display dynamic histograms of the running tasks. The user can watch as a task starts to use more time and its histogram begins to grow relative to the other tasks. This can give the user insight into the detailed working of an application.

Command Files and Action Keys

RTOS measurements are made easy to access by use of the action keys provided by the debug environment interface. To run a measurement, the user simply points and clicks on the appropriate action key (which runs a command file), and the setup is done automatically. If parameters are required, the user is prompted for them. In the graphical interface, these prompts appear as dialog boxes in which the user can either type or cut and paste the required parameters. The user can modify the command files provided and set up action keys for user-defined RTOS measurements. The command files provided with the RTOS tool represent a set of measurements that a typical designer of an RTOS-based application would be likely to need.

Summary

The RTOS measurement tool uses the complete capabilities of the emulator to provide measurements for RTOS-based applications. Some instrumentation must be done, which does produce some intrusion, but the results are well worth the minimal extra time. A complete understanding of the interaction between an application and a RTOS kernel is possible. Users will learn things about the way their applications run that they could not learn by any other means.

Acknowledgments

Special thanks to Eric Kuzara who provided the spark, many ideas, and much encouragement for the RTOS project.

Mike Dotseth
Software Development Engineer
Colorado Springs Division

Message Queue Implementation

The message channel is implemented as a single UNIX-system message queue. A message queue offers low overhead and simple operation as long as message distribution is limited to a single host. Sockets, and domain† sockets in particular, were not mature when this choice was made (1987). Today the choice is still appropriate since there is no current need to distribute a single debug environment session across multiple platforms. The X Window System and various network services provide support for remote network access to a session.

The message type attribute provided on each message is defined as the combination of a client address and a packet type. The address is assigned when a client attaches to the daemon. The packet type distinguishes between connection, event, acknowledgment, and channel protocol packets. Unlike a socket-based channel, the presence of the message type allows message acknowledgments to be read from the head of the queue even if other messages destined for the same client were placed in the queue earlier.

Most routine transmissions from clients to the daemon on the message queue are actually synchronous, with the sender waiting for an acknowledgement that the message was received and processed by the daemon. This was found to be necessary for pacing the system properly. Clients cannot overrun the daemon since each transmission is acknowledged and, just as important, a sequence of events generated in one client stays in proper sequence with events generated in other clients.

† Domain sockets are sockets optimized for throughput when both ends of the connection reside on the same node in the network.

Avoiding Deadlock

This picture of synchronous transmissions from each client to the daemon and asynchronous transmissions from the daemon to its clients is a little too simple, however. The most important exception occurs when multiple transactions with the emulator occur atomically within a single ownership of the channel semaphore. The daemon may very well be blocked on the channel semaphore since it performs polling activities as well as message distribution. If a client attempts to forward an event message to the daemon synchronously from within the channel lock, the acknowledgment will never arrive since the daemon is blocked on the channel—a classic deadlock.

Messages are only sent within a channel lock when multiple commands must be treated atomically, such as during emulation configuration. Therefore, acknowledgment is disabled when messages are sent from within a channel lock. However, synchronism is not lost. The client that sent the unacknowledged messages is required to send one more synchronously to ensure that it pauses until all messages are processed by the daemon.

In addition, if another interface issues a command that does not require accessing the hardware, it will not hang even if this requires sending the daemon a message. This is because the daemon is designed to suspend its channel wait temporarily, process software-only messages (including any acknowledgments), and then resume the channel wait.

Messaging Citizenship and Command Forwarding

The message queue provides the transport layer for distributing events within the session. Subscription and distribution have already been discussed. But which events should be distributed? Which are of session-wide importance? Which are not? The definition of session-wide interest is termed *citizenship*. The designer of each user interface is responsible for going through the list of events supported by the daemon and determining when, if ever, the user interface might generate the event. An example is the user command to step to the next source line in the target program. Each program view, whether it is at the source level, the assembly level, or the machine level, subscribes to the step taken event message. Therefore, each place in the user interface of each window that can generate a program step must also generate the event message that announces that the step was taken. Any currently active program view will then be notified of the step and take whatever actions are required to update its window contents.

A second example of a session-wide concept is the current working directory. It was too confusing (and only marginally useful) to allow each window to have its own value for the current directory. Therefore, each user interface is required to notify the daemon, via the appropriate event message, whenever the user changes directories. Each user interface responds to this event by making the system call to update its own current working directory. The OSF/Motif user interfaces also respond by updating any file selection dialogs that are currently raised as well.

A design rule related to this citizenship requirement is to defer the secondary effects of a user command until the event message arrives back from the daemon. In the case of a program step, the step is performed immediately and, if successful, an event message is sent. The secondary effects of display updates are deferred until the event message is received back from the daemon. In this way, the client responds to user-created events in the same way whether they were initiated locally or in a different user interface in the same session. This approach also creates very modular code—the step code needs to know nothing about memory or source display code. The only software connection between the modules is through messages.

Another aspect of citizenship is related to the operation of command processing in the HP 64700 Series emulators. Each transaction with the emulator consists of a command, its reply, and any asynchronous events that may have occurred in the emulator since the previous transaction. An asynchronous event message is generated when the target program hits a breakpoint, accesses guarded memory, and so on. As mentioned above, to improve overall efficiency, each user interface process has direct access to the emulator. However, asynchronous messages must be distributed throughout the session to update displays and perform other actions. Furthermore, the contents of the asynchronous message may be of no local interest whatsoever. For example, the software performance analyzer window has no local response to a software breakpoint message. The citizenship rule is that any such messages are stripped from the command reply and forwarded to the daemon for distribution. No local action is taken until the message is distributed back to the client by the daemon. Fortunately, asynchronous messages are rare relative to the number of transactions with the emulator. Thus, a breakpoint event returned to the software performance analyzer window is dealt with in exactly the same way as if it were returned to the debugger. It is not acted upon immediately, but is sent to the daemon and routed to interested clients, one of which is certainly the debugger.

An important consequence of this design is that the system implements a bidirectional facility called command forwarding in each interface. Each interface is capable of sending any legal command to any other interface running in the session, and any interface is always able (except for normal blockages) to receive and execute commands sent from other interfaces. This exposes almost unlimited possibilities to advanced system users, including the creation of complex command scripts capable of automated testing using the measurement strengths of several windows at once.

Managing Message Queue Overflows

During a session, any of the user interface clients may be suspended at any time to perform shell escapes or other commands of unknown duration. Arrangements must be made to deal with event messages bound for a suspended process. If the message queue fills up with messages for a suspended (or otherwise busy) user interface, the entire session would freeze until those messages were read. The daemon would block waiting for space to become available, and the session would come to a halt—another deadlock.

The approach taken to deal with this situation is as follows. When the message queue is full, the daemon pulls the oldest message from the queue and determines which client it is for. All messages for that client are then pulled from the queue and placed into a temporary file, as are all future

messages until further notice. A token is then placed into the queue to be read by the client when it resumes operation. This token indicates that messages have been redirected and provides the name of the temporary file in which the messages are being put. When the client eventually returns from the suspension, it reads this token instead of an actual event message. The client acts on this by informing the daemon (in an acknowledged, synchronous transaction) that it is ready to resume and then reads the backlog of messages from the temporary file. The file is removed when all messages in it are processed. The daemon then resumes the use of the message queue for this client. This approach imposes overhead only when a backlog actually develops. An alternative would be to inform the daemon of every user interface operation that might take too long, discontinuing the use of the message queue until the operation is complete. Not only does this add overhead to every operation, but it relies on the developer to identify all such situations.

One aspect of the need to swap out messages from the queue is unique to the use of a single message queue as the transport mechanism. The queue may become full when any non-suspended client attempts to transmit a message. However, only the daemon can actually swap out messages. Since the queue is full it cannot be used to notify the daemon of the need to swap out messages. Instead, a UNIX-system signal, SIGUSR2, is sent from the client that detected the full queue to the daemon. When the daemon receives this signal it aborts its current task (if any) and immediately swaps out the oldest messages from the queue.

The daemon is also capable of swapping out all messages directed to itself if no other messages are found in the queue. This can happen in the rare situation where there is high messaging activity and the daemon gets temporarily behind in processing its own messages.

Other Daemon Functions

Other tasks are also performed by the daemon. Some events returned by the emulator as asynchronous messages require further processing before transmission to the user interface clients. An example of this is the receipt of an unknown software breakpoint message. This means that a breakpoint that was not set by the emulator was executed by the target program. The run-time library provided with the HP Advanced Cross Language System (AxLS) compiler uses this mechanism to announce the completion of a target program (return from main() or call to exit()) and for faults such as divide by zero or any other kind of message that the monitor or user code wants to send to the host. The conversion from the raw message to its final form is performed in the daemon before the message is copied for distribution. Each class of messages is provided with a filter function, which is invoked to perform this operation. The filter can either delete the message, modify it by obtaining additional information from the emulation system, or change it to a different message altogether (see Fig. 6).

Another task performed by the daemon is polling for status changes related to certain classes of event messages. This is closely related to the filter functions since the filter functions accumulate the retained state information needed to bring newly attached clients up to date. Just as there is a filter function for each class of messages, there is also a

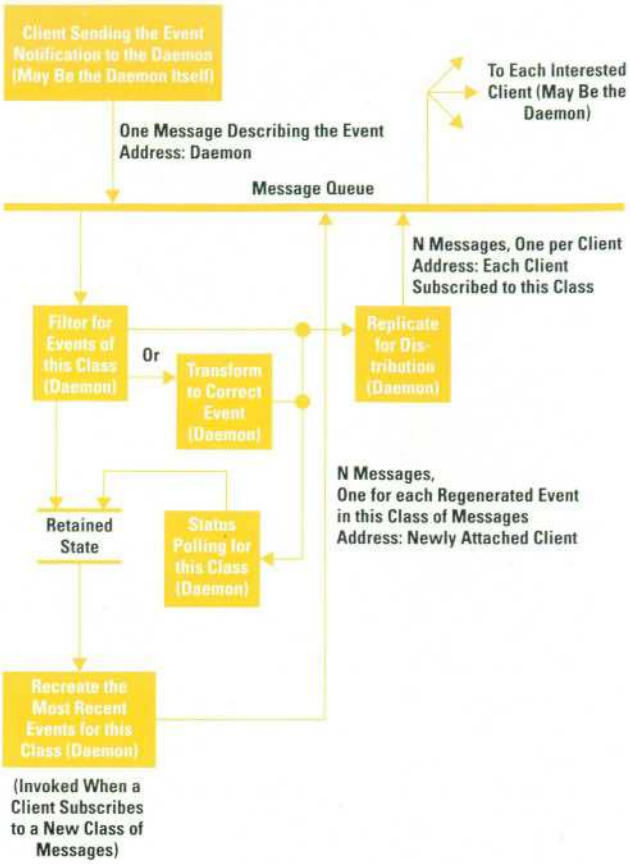


Fig. 6. Debug environment message routing and processing.

subscription processing function for each class. This function is invoked each time a client subscribes (or desubscribes) to a specific message class as described above. In the case of a new subscription the appropriate event messages are recreated and sent specifically to the new client window. In addition to this, some classes of messages require status polling to determine state changes. Emulator and analyzer status are two examples of this. Polling is only enabled when one or more clients are currently subscribed to the associated message class. Thus, the first client to subscribe to a given class causes polling for status changes related to that class to start and the last client to remove itself from the subscription list causes status polling to stop. A new event message is transmitted each time a change in status is detected.

Usability Issues

Usability is a major concern and expectation with any product. This is especially true of products with computer-based graphical user interfaces. The intense competition among computer vendors to improve the toolkits and style guides available to developers of applications running on their platforms is testimony to this fact. The decision was made early in the development of the debug environment to use the OSF/Motif toolkit and to conform to the IBM Common User Access standard, which applies to personal computers as well as to workstations. This decision was necessary but by no means sufficient to complete the design of the graphical user interface for the debug environment. Additional factors considered were:

- Concurrent operation of a debugger and an emulator style interface
- Application of direct manipulation concepts
- Operation of scrollable windows with unbounded contents
- Provision for user-supplied macros on pushbuttons or action keys
- Retention of all user-typed responses for later use
- Provision of an immediate interrupt or abort on lengthy operations
- Compatibility with the previous command-line-based user interfaces.

Synchronous Window Operation

There are two major issues associated with the concurrent operation of a debugger and an emulator style user interface in the same session. The first is that a debugger pauses whenever the target program is running, whereas the emulator style is to stay alive while the target program runs. This allows the user to peek and poke while the target system is running (using the HP dual-port emulation memory architecture), albeit at a low level. The debugger pauses since most of its displays are based upon the current state of the program stack, which is essentially undefined while the target program is running. Therefore, allowing the emulator/analyzer window to continue while the debugger/emulator window is paused is a useful feature. The problem is that the debugger must now be much more robust and able to handle the situation when the target program is started (or stopped) from the emulator/analyzer window. Fortunately, the event system described previously in this article provides a natural way to deal with this. The debugger responds to the "Target program now running" event by going to its paused state. When the "Target program now paused" event arrives the debugger queries the target for its stack and register contents and displays the current state.

This ability of the "live-keyboard" interfaces (such as the emulator/analyzer and the software performance analyzer) to continue operation while the debugger is paused because of a running user program is crucial to the power of the debug environment. If the user's code has gone astray the debugger may be useless in helping to understand the real problem since it requires interaction with the emulation monitor code to do anything. It might be more appropriate to take some state traces to look at the errant operating situation. The emulator/analyzer allows the user to take the traces and display them all without disturbing the operation of the user code (which may be disturbed enough already). The individual interface windows complement each other naturally and powerfully in this manner.

The other issue concerning the concurrent operation of the debugger and the emulator windows is the difference in viewpoints. The debugger is typically used with a source-level view of the target program, whereas the emulator window is typically used for a machine or mixed C and assembly source view. Operations such as single-stepping the target program and setting breakpoints are defined for both views. However, not all operations at the low machine level correspond to those at the source level. Again, the event system is used to communicate these events and care was taken to ensure proper operation.

Pop-up Menus

Direct manipulation is fundamental to most successful graphical user interface components such as sliders, scroll bars, toggle buttons, and resize handles. It allows the user to point at the item to be changed and to change it with immediate feedback. Indirect methods, such as pull-down menu systems and pushbuttons, are useful when a direct method would be confusing or obscure. The debug environment uses a combination of these methods. Furthermore, an additional direct manipulation method is used to supplement some functions available in the pull-down menu system: the pop-up menu.[10] Pop-up menus can be obscure since not all portions of the user interface support them. The user may become lost trying to remember when a pop-up is available and when it is not. In the debug environment the availability of a pop-up menu is announced by a change of the mouse sprite. Normally, the position of the mouse is indicated by an arrow-shaped sprite. Whenever the mouse is over a display area that supports a pop-up menu the sprite is changed to a small hand shape, as in the left half of Fig. 7. Holding the right mouse button down for more than one second will raise the pop-up menu. The choices on the menu apply to the highlighted line in the display (see the right half of Fig. 7). These are chosen to make the most frequently used operations quickly available. In the source window, for example, these include setting and clearing breakpoints, raising the text editor to examine or modify the source file

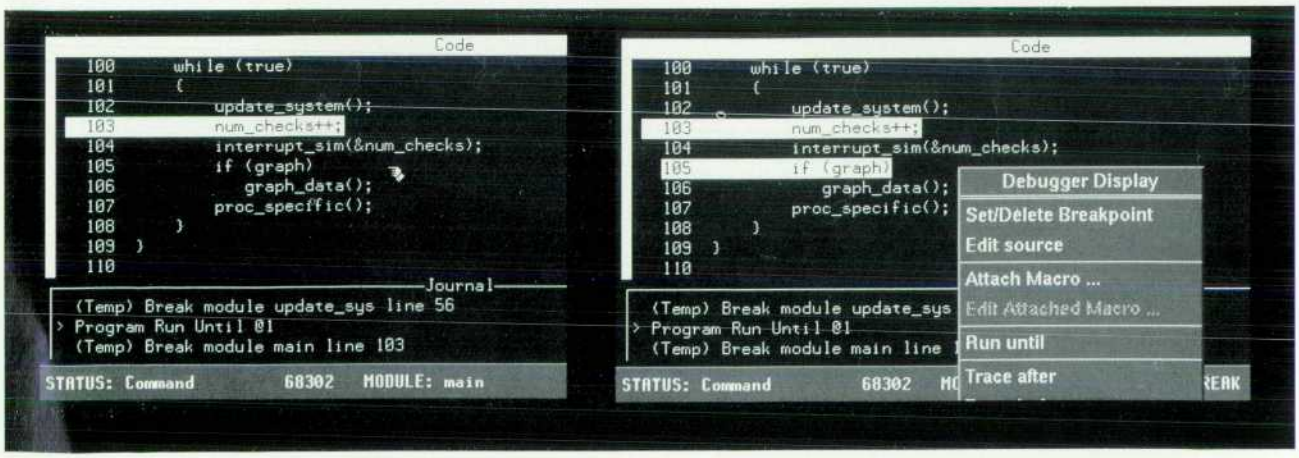


Fig. 7. Hand cursor indicates pop-up menu availability.

A New Perspective on Emulation Hardware Modularity

The HP 64700 microprocessor development system is a modular instrument frame that allows users to do in-circuit hardware and software debugging of an embedded target system. The cardcage (Fig. 1) accommodates a variety of optional cards to support all of the customer's in-circuit debugging needs. These include microprocessor bus analyzers, software performance analyzers, emulation control cards, and communication cards. The graphical user interface running on the user's host computer communicates with the emulation frame over a LAN or serial connection. The frame then connects to an emulation probe board over a custom high-speed cable. Finally, this probe is plugged into the user's target system in place of the target microprocessor. This architecture allows the user to measure and control the target system from the familiar and easy-to-use interface on the host computer.

Several advances in integration technology have allowed us to rearchitect the HP 64700 emulation system and achieve a substantial reduction in system cost. Traditionally, the run control and memory subsystems of an in-circuit emulator required several through-hole boards in the system cardcage. Now all of this is integrated onto a double-sided surface mount probe card about one-fourth the size. The result is a product with essentially the same feature set for about half the cost.

As processor clock speeds increased in the 1980s, we recognized that some microprocessor signals had to be handled as close to the target system as possible. However, we were still very sensitive to the mechanical intrusion of the emulation probe. We simply could not put a substantial amount of circuitry on the probe because this would prevent us from physically plugging into target systems. Thus the first active probe emulators were architected to have as little hardware on the probe board as possible. This required us to use very expensive cabling to route high-speed processor signals up to the emulation frame and back. It also put a strict limit on the length of the cable because of concerns over signal propagation delay.

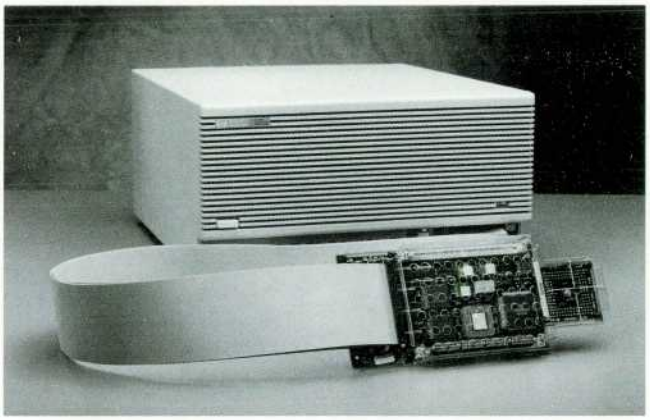


Fig. 1. HP 64700 microprocessor development system emulator cardcage and probe.

Recent advances in integration, including fine-pitch surface mount technology and highly integrated, high-speed SIMMs, have caused us to rethink our philosophy about active probes for emulation. Today's emulation control probe handles all of the processor-specific signals right at the target system for maximum electrical transparency. We are also able to put up to 8 megabytes of emulation memory on the probe. This typically allows us to operate out of emulation memory with zero wait states (i.e., as fast as the processor can access the target memory system).

We are certainly still sensitive to the mechanical intrusion that even this relatively small probe may introduce. Therefore, we have developed a flexible adapter that can be connected from the emulation control probe to the target system. This combination allows us to plug into virtually any target system with minimal electrical and mechanical intrusion.

Handling high-speed, processor-specific signals on the emulation control probe has allowed us to rearchitect the rest of the emulation system to be substantially more generic and less expensive. First of all, we are able to use much cheaper cables because instead of sending high-speed signals from the target system to the cardcage and back, we have simply extended the backplane of the cardcage down to the probe.

Secondly, the emulation hardware in the cardcage is no longer specific to a target processor. A generic analysis bus generation card in the frame can be customized via firmware in flashable ROMs in conjunction with programmable hardware to work with any of the emulation control probes.

Aside from reducing the system cost, isolating the processor-specific aspects of the emulation system to a single probe board substantially lowers the cost of upgrading an emulator to work with a new processor. For example, to convert from a 68020 emulator to an 80960SA emulator, the customer only needs to buy a new emulation control probe along with new software for a fraction of the cost of buying an entire new system.

Another advantage of the new architecture is time to market. The development effort to design and implement a new emulation control probe is far less than that required to design a traditional emulator. The result is that we are able to support more of the popular microprocessors for embedded systems at a much lower cost.

Thomas C. Ferguson
Software Development Engineer
Colorado Springs Division

containing the line in the display, attaching a macro, running until the highlighted line, and starting a trace of program activity from the highlighted line in the program. In any event, if the user never discovers the existence of pop-up menus in the product at all, all operations available on the pop-up menus are also available from the pull-down menu system. Thus, pop-up menus are an auxiliary, expert mode of operation.

The behavior of the contents of the pop-up menu associated with each type of display follows the same rules as the contents of pull-down menus. The entries in the menu are the same no matter what line in the display is highlighted. If an individual item is not appropriate for the highlighted line it is halftoned to show that it does not apply. Related choices are grouped together. An additional rule is that the first item is always the one chosen if the user does a "quick click" of the right mouse button. For example, the pop-up menu in Fig. 7 shows Set/Delete Breakpoint as the first item. This is the default action taken if the user holds the right mouse button for less than a second (a quick click). These rules, together with the behavior of the mouse sprite, lead to a short learning time and accurate operation of the pop-up menus.

Sticky Scroll Bars

Many of the displays in the debug environment provide scrolling capability to allow the user to view the data of interest. However, the direct application of OSF/Motif scroll bars is very difficult in some cases. One example is the display of memory contents in the target system. The display covers the entire address space of the target processor—up to four gigabytes. Another example is the display of trace data (the results of a trace of program execution). Here the length is bounded by the size of the trace buffer in the analyzer, but the number of display lines is a nonlinear function

of the display mode and may be very expensive in time to compute. For example, in one mode the display is restricted to only the source lines that correspond to program execution in the trace buffer. Each source line reference may represent one to fifty lines of text. Each entry in the trace buffer may correspond to zero, one, or even fifty source lines including leading comments.

Thus, in the first case the number of lines is known but enormous, and in the second case the number of lines is unknown but too large to compute quickly. The solution is a "sticky slider" together with arrow buttons to perform the home-up and home-down functions. The sticky slider acts like a spring-loaded return-to-center linear control. Dragging the slider upwards scrolls the list a few lines, just like an ordinary scroll bar slider. Dragging it back to the center restores the original position of the list. Releasing the slider allows it to return to the center of the slider trough, ready for the next operation. An additional feature is that dragging and holding the slider against either limit causes repetitive scrolling for browsing. The basis for this design is that most list scrolling operations are of limited range and it is very important to be able to undo a previous scroll in a predictable way. The absolute position feedback formerly provided by the position of the slider is conveyed in the list itself: by the memory address column in the memory data display and by the trace list number offset in the trace data display.

User Customization

Customization of the user interface to fit the specific needs of each user was a high priority. X Window System resources provide one level of customization. The debug environment supplies over 50 application resources to allow user adjustment of display size, timeouts (e.g., the pop-up menu hold time), the initial state of the command line, the initial contents of selection lists, preferred text editor, and more. Perhaps the most significant of these is the action key panel—a set of buttons for which the labels and the associated actions are both specified by the end user. The action is in the form of a command to execute when the button is pressed. One or more rows of action keys can be specified by the user. A default set is shipped for each window in the debug environment. An action key can be defined to remake and reload the target program, to reset and run the target system, to specify and start often-used traces, to display data, or to perform any other frequently required task. The command for an action key can use the current contents of the cut buffer as part of the command. Syntactically, this is indicated by the use of an empty pair of parentheses, (), in the command text. Thus, the user can cut the name of a variable in the display and then use an action key as an operator on it, perhaps to initiate a trace of all accesses to that variable.

Acceleration for Symbolic Operations

Typing entries in an otherwise fully graphical user interface can be an annoyance. Typing the same entry many times during a session can be truly aggravating. The debug environment solves this problem by providing a recall dialog for nearly every text entry field in the user interface. This includes the command line, file selection dialogs, the cut buffer, and others. Each time a text entry is made and applied it is saved in the associated recall dialog. Furthermore, application X Window System resources permit the initialization of each of these recall dialogs with a user-specified list. For example, the Previous Files section of the file selection dialog can be initialized with the names of commonly loaded absolute files, no matter what directory they are in. The recall buffer for the cut buffer can be initialized with commonly used symbols, such as main, loadHardware, and so on.

To ease the pain of long, difficult debugging sessions, the current contents of each recall buffer are saved whenever the session is ended for continuation. Because the windows are designed to be viewed simultaneously, each window uses the system messaging facility to echo cuts to its local cut buffer to all other session windows. This allows symbols and expressions to be shared easily with other windows where they may be useful in following a complex debug hypothesis.

Signal Support for X Windows

In most powerful applications, the user must be able to cancel lengthy operations. The X Window System provides several mechanisms to allow this. Unfortunately, none is based on the UNIX-system signal mechanism. The internals of the debug environment were reused directly from the older character-based user interface, which uses the signal mechanism to allow the user to abort lengthy operations. The user is told to "Press Control-C to abort." The terminal driver intercepts the Control-C character and issues an interrupt signal to the process. The X Window System makes no such provision for special handling of certain characters in this way. Furthermore, the remote client/server relationship permitted by the X Window System rules out any straightforward implementation.

The solution taken by the debug environment is shown in Fig. 8. A special-purpose daemon, XSigServe, is run in the background on the same host that is running the debug environment. This is not necessarily the same host that is displaying the session (the latter is known as the X server host). The X Window System allows many different clients to receive keyboard events from the same window. XSigServe

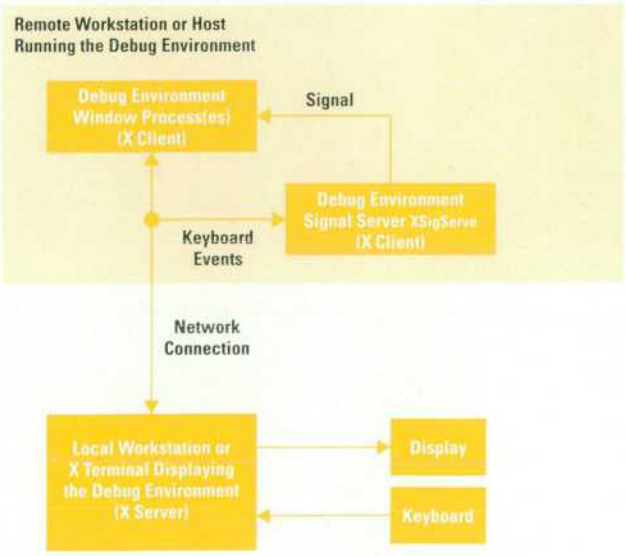


Fig. 8. Debug environment XSigServe architecture.

takes advantage of this to monitor the characters bound for each debug environment window. When a Control-C arrives, XSigServe sends a SIGINT signal to the process associated with the X window in which the Control-C was pressed. The process that owns the window may be blocked or busy in a low-level operation when this occurs. Ordinarily it would not see the Control-C until the operation is completed. However, the SIGINT is delivered from XSigServe immediately and the operation can be aborted. The window and process IDs as well as the characters and signals to be monitored by XSigServe are passed from each debug environment to XSigServe when the debug environment is started by using an application-specific property on the root window to find first XSigServe and then client messages to pass the information.

Backward Compatibility

Compatibility with previous user interfaces for the HP 64700 was very important. There is a large installed base of now expert users of the older interface. An important objective was to allow these expert users to transition to the new, graphical, user interface at a pace of their own choosing. The most salient feature of the previous interface in this regard was the softkey-driven command line area. Eight softkeys prompted the user with choices that were legal at each point in the command entry process, including modifications in the middle of the command. This command line area was retained in the debug environment as an optional region at the bottom of each window. Initially off, the command line area is added automatically when the user types in the display area of the user interface. It can also be added by using an entry in the pull-down menu system. From the keyboard, the command line operates exactly the same way as in the older character-based product. Habits and tricks learned by users still apply. In addition, enhancements were made to take advantage of the mouse and the cut buffer. The mouse can be used to position the text cursor and to perform cut and paste operations. A graphical recall buffer is provided to hold previously issued commands.

In addition to these directly visible usability issues, the implementation dealt with two hidden problems. First, the debug environment reuses the substantial body of software that implemented the previous screen-oriented, but character-based, user interface. Second, the work required to deliver over a dozen different products containing four different types of user interface windows (the debugger/emulator, emulator/analyzer, software performance analyzer, and emulation configuration windows) within a year had to be minimized.

The original ASCII-oriented product core used a package based on curses to write character strings to specified locations on the terminal screen. The product core was isolated from curses by a display utility library that managed virtual screens and provided printf functionality to write into those screens. Since curses is based on support for terminals, the notion of dynamic screen resizing was not built into the display library nor into the product core. The strategy adopted for the debug environment was to replace curses and the other drivers under the display utility library with look-alikes that actually write to X windows. Neither the display package nor the product core was modified to accomplish this. Unfortunately, the prohibition against dynamic screen resizing had to stay in effect. Thus, the main display area in the debug environment cannot be interactively resized by the user. Instead, the size of the window is fixed by application resources (lines and columns) that the user can place in an Xdefaults file to customize. Windows can be ended and restarted easily for resizing without disturbing the state of the debugging session.

XcActionKeys
Manage one or more rows of user-defined action keys.

XcBrowse
Browse a text file or an array of strings. Provide the means to save the contents to another file.

XcChoose
Present the user with a choice of one of N alternatives.

XcCutbuf
Display the most recently cut text and provide a recall buffer to select previously cut text strings.

XcFilSel
Allow the user to select a file from the current directory or from the list of previously selected files. Change directory whenever the current working directory event message is received.

XcHelp
A simple help system with a list of topics and a file for each. An XcBrowser is used to view the help text.

XcPopup
Manage a pop-up menu, including timeouts and callbacks.

XcRecall
Maintain a list of previously selected items in a scrollable recall dialog.

XcSelect
Similar to XcRecall, except the list is fixed at startup.

XcTWind
A character-based, random-access text display, including highlighting, text cutting, and pop-up menu support.

Fig. 9. Debug environment user interface component hierarchy.

User Interface Components

The strategy taken to minimize the amount of work required to produce the plethora of products required was to create user interface components. Each component is a software object built from OSF/Motif widgets and perhaps other components (see Fig. 9). Some are very simple, such as the recall dialog and the action keys. Others, such as the text window that supports the display package, are quite complex. The public interface for each requires no detailed knowledge of the X Window System or OSF/Motif operations internal to the component, except that certain operations, such as raise

and lower, are specific to graphical components in general. The use of these components even for very simple operations, produces consistency "by construction" in all of the products based on them.

Debugger Macro System

A very powerful debugging feature provided by the debugger/emulator is the use of predefined and user-defined macros. Unfortunately, the use of this feature in previous debugger versions was hobbled by user unfamiliarity with the built-in macros and a poor interface for the entry of user macros. The addition of the graphical features to the debugger gave us a chance to clean up this interface, and provides a good example of how the debug environment makes the debugging task more friendly.

Debugger macros look like C functions: they have input parameters, return a value, have local variables, and contain C expressions and statements. In addition, debugger commands can be embedded within the macro. Within a macro, target program variables, debugger variables, and processor registers can be referenced, and variables local to the macro can be defined. Macros can be called from an expression, including expressions within a macro, although they cannot be called recursively. Since they can call other macros, and one of the built-in macros allows forwarding commands to any of the other concurrent interfaces, debugger macros can even cause a command to be executed in a different interface.

Macros have the following uses within the debugger: they can be called directly to execute a complex series of debugger commands, they can be called after a breakpoint or step command to implement complex or data driven breakpoints, and they can be used to patch target code without recompiling. To implement complex breakpoints, a macro can be attached to a breakpoint to be called when the breakpoint is hit. The macro return value will then be checked, and if nonzero, the program will resume execution at the current instruction pointer. Program execution will stop (and a message will be generated) if the return value is zero. This feature is used to patch target code by attaching the patch macro at the start of the improper code, executing the new code within the macro, setting the instruction pointer to point to the end of the improper code, and returning a nonzero value to continue execution at that point. Macros can also be attached to every program step using the Step with Macro menu selection to form debugger assertions, as in most host-based debuggers.

Before the debug environment, using the macros took a level of skill few users mastered. For example, in the code fragment shown in Fig. 10, to cause program execution to cease at line 104 when the variable num_checks is equal to 10, the user needed to remember that a macro could be attached, remember the macro name when, and remember the type of parameter it takes, resulting in the command:

```
Breakpoint Instruction #104;when(num_checks==10).
```

Even more difficult was creating a user-defined macro to accomplish the same task. This required the user to execute the command Debugger Macro Add int stop() and then enter the body of the macro in the journal window:

```
{
  if (num_checks == 10)
    return(0);
  else
    return(1);
}
```

If an error was encountered following entry of the macro body (say the user misspelled num_checks as num_cks), then the entire macro had to be reentered. This was not particularly friendly.

The graphical version of the new debugger supports these uses by providing a selection dialog of built-in and user-defined macros (Fig. 10). This dialog shows the defined macros and their parameters, and whether they are built-in or user macros. The selected macros can be edited (if a user macro), called, or attached to a breakpoint line, with the parameters specified in the dialog. Editing of a user macro takes place in a terminal window using the editor that the user has specified in the resource file. If a syntax error is detected in the macro after editing, the user is given a chance to edit again, with the cursor positioned on the line containing the error. The graphical debugger provides three sample user macros, which can be used as templates by the user to create macros.

Macro editing and selection are also supported by several pull-down menus. User-defined macros can be stored into a

Fig. 10. Macro dialog system.

file by using the File→Store→User_Defined Macros file selection dialog, and can be restored using the File→Load→User_Defined Macros file selection dialog. These dialogs allow the user to save macros before exiting a debug session and restore them upon starting a new session. This feature was very difficult to use with the nongraphical interface and resulted in a great deal of frustration with the interface when a great amount of time was spent entering macros only to lose them upon ending the debug session.

Summary

The HP 64700 embedded debug environment provides a very powerful and automatically integrated suite of debugging tools for embedded system development teams using emulators. The ability of this system to move back and forth rapidly between static and real-time debugging and the power of its emulation and measurement hardware make it suitable for even the most difficult embedded system debugging problems. Its ability to integrate with the HP SoftBench CASE framework allows natural extensions to handle advanced embedded software development using larger teams. It is designed to fit the needs of advanced embedded developers and excels at meeting the challenge of delivering quality real-time systems under severe time-to-market pressures.

Acknowledgments

We would like to acknowledge the vision, determination and understanding of Dave Richey and Tom Kraemer. Without their support, the creation of this system would have been impossible. Specific and well-deserved thanks go to Manfred Arndt and Mike Upham for key contributions when the going was tough. Special mentions should go to Sherry Adair, Cheryl Brown, Denny DeBoer, Joe Hawk, Tim Chambers, Tom Batcha, and Bruce Kosbab. Eric Kuzara and his group also provided key help and counsel. And as always, the marketing, quality, and manufacturing organizations made things work when all else failed, especially Tom Rogers, Maureen Strong, Chuck Rice, and Blaise Copeland.

References

1. T. Saponas and B. Kerr, "Logic Development System Accelerates Microcomputer System Design," *Hewlett-Packard Journal*, Vol. 31, no. 10, October 1980, pp. 3-13.
2. T. Sperry and G. Bay, "Real-Time Operating Systems," *Embedded Systems Programming*, Vol. 5, no. 1, January 1992, pp. 67-71.
3. M.R. Cagan, "The HP SoftBench Environment: An Architecture for a New Generation of Software Tools," *Hewlett-Packard Journal*, Vol. 41, no. 3, June 1990, pp. 36-47.
4. C. House, "Viewpoints—Chuck House on the Electronic Bench," *Hewlett-Packard Journal*, Vol. 31, no. 10, October 1980, pp. 30-32.
5. G. Hamilton, et al, "An Electronic Tool for Analyzing Software Performance," *Hewlett-Packard Journal*, Vol. 33, no. 6, June 1984, pp. 26-32.
6. D.L. Neuder, "A Test Verification Tool for C and C++ Programs," *Hewlett-Packard Journal*, Vol. 42, no. 2, April 1991, pp. 83-92.
7. W.A. Fischer, Jr., "Host Independent Emulator Software Architecture," *Hewlett-Packard Journal*, Vol. 39, no. 6, December 1988, pp. 52-56.
8. J.B. Donnelly, et al, "Emulators for Microprocessor System Development," *Hewlett-Packard Journal*, Vol. 31, no. 10, October 1980, pp. 13-20.
9. M.A. Champine, "A Visual User Interface for the HP-UX and Domain Operating Systems," *Hewlett-Packard Journal*, Vol. 42, no. 1, February 1991, pp. 88-99.
10. J. Seymour, "New Interface Dilemmas," *PC Magazine*, July 1992, pp. 99-100.

Software Performance Analysis of Real-Time Embedded Systems

The HP B1487 software performance analyzer is a plug-in card for the HP 64700 emulator system. It makes activity and interval measurements on instrumented code for embedded microprocessor systems. The design is able to deal with difficult analysis situations involving caches and prefetches.

by Andrew J. Blasciak, David L. Neuder, and Arnold S. Berger

How can code written by 15 or more individual programmers, with the number of lines of source code extending into the hundreds of thousands, be analyzed and optimized? Problems that arise through the subtle interactions of many different modules of code are in the realm of the HP B1487 software performance analyzer. The HP B1487 consists of an option card for the HP 64700A emulation cardcage and analysis software for HP 9000 Series 300 and 700 workstations and the Sun Microsystems SPARCstation.

Fig. 1 shows the HP B1487 user interface. This interface operates under the HP 64700 debug environment, which is described in the article on page 90.

The first software performance analyzer for embedded systems design was the HP 64310A, an option card for the HP 64000 family of microprocessor development products. Using the microprocessor emulator as the input device, the HP 64310A monitored the memory and I/O addresses that were output by the microprocessor under emulation. A local processor on the software performance analyzer would cycle the analyzer through as many as 16 preselected address ranges, stopping at each range for a user-specified period of time. Any time the value of the address on the emulation processor fell within the range currently being monitored, a counter was incremented. After all ranges were monitored, the entry point into this list of ranges would be randomly selected and the process would start all over again.

The HP 64310A could monitor program activity, memory activity, function duration, and intermodule linkage. Although it collected data in real time (the emulation processor was never stopped to collect data), the data that it gathered was essentially statistical in nature. Memory activity could be analyzed only if it occurred when the analyzer was looking at the particular location where the activity occurred. If a program ran long enough, on the average enough data could be gathered to give an accurate picture of the efficiency of execution of the code under analysis. All of the

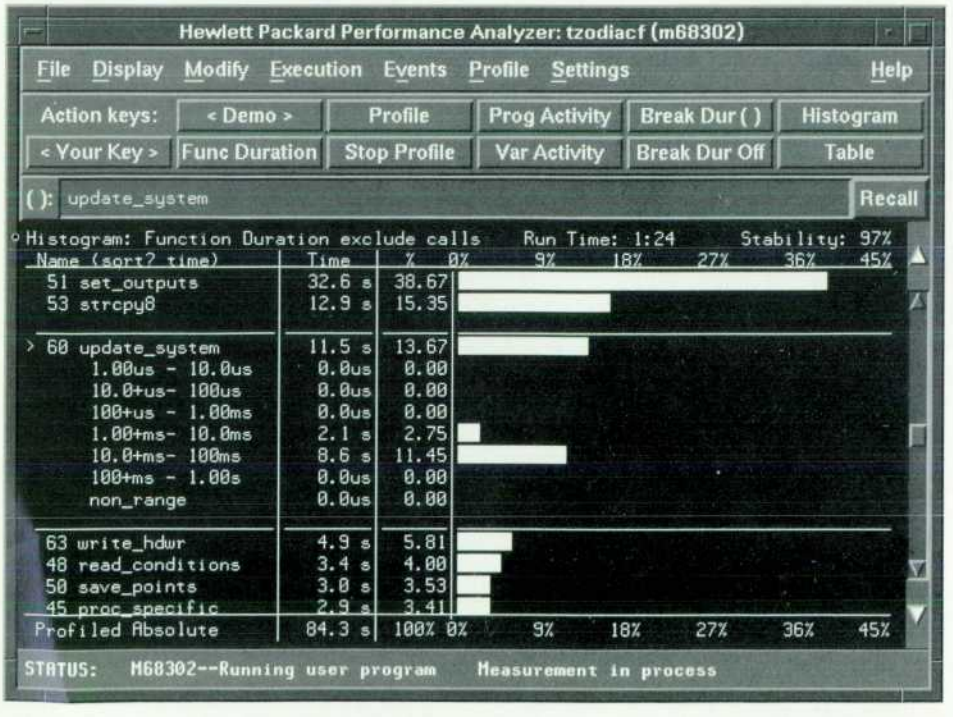


Fig. 1. HP B1487 software performance analyzer user interface.

measurements made by the HP 64310A were statistically sampled. This analyzer had some severe limitations that are addressed by the HP B1487 analyzer.

The HP B1487 analyzer can perform real-time, nonsampled measurements. This gives the analyzer the capability of recording accesses to every address that the emulation microprocessor requests. Thus, extremely low-duty-cycle events can be readily tracked and analyzed. These types of events, such as interrupts from the outside world, were very difficult to catch with statistical sampling methods.

Modern microprocessors are much more complex in their operation than the processors that the HP 64310A had to deal with. Modern processors implement techniques such as prefetching, pipelines, and caches to speed up their operations. These techniques are used because accesses to the outside world, such as RAM and ROM memories, are inherently slower than internal accesses. Therefore, the processor attempts to minimize the need to fetch instructions from external memory. As a result, accesses to external memory are decoupled from the sequence of execution of the fetched instructions.

Caches present a severe challenge for a software performance analyzer. A cache is a data and/or instruction memory within the microprocessor from which the processor attempts to execute successive instructions. Operations from within the cache are inherently much faster than operations that require accesses to external memory. The microprocessor's cache control unit attempts to keep the cache filled with the most current set of instructions and data. Only when the processor needs new instructions or data in its internal cache does any observable bus activity occur. If the cache is large enough, and "large enough" is loosely defined as when both possible destinations of a program branch are contained in the cache, then long periods of time can go by without any observable bus activity. In such a situation, a software performance analyzer is virtually useless.

A designer might solve this dilemma by instructing the microprocessor to turn off the internal caches, thereby causing all accesses to go to external memory. This would solve the problem, but the system would obviously not be running at full speed. An alternative solution is to insert certain instructions called markers into the code. The markers instruct the microprocessor to perform write operations to external memory or I/O space at critical locations in the program, such as function entry or exit points, or when the embedded operating system initiates a task switch. The software performance analyzer can then be configured to record and make measurements based upon these marker-based events.

Measurements

The software performance analyzer is capable of making a variety of software performance measurements. These measurements answer the questions:

- Why does it take so long to execute my program?
- Which modules are slowing down overall program execution?
- How intensive is my program's use of data or I/O ports?
- How much time is spent getting between various points in my program?
- Which function is called most often?
- Which functions are taking the most processor time?
- What is the time distribution of calls to my functions?

Activity Measurements

Activity measurements are bus cycle counting measurements designed to present the user with an overview of how a program is using the microprocessor resources. This type of measurement provides answers to the first three of the above questions. These measurements can reveal which library a program is executing in most of the time, or how often a region of program space is being accessed.

Memory Activity Measurements. The software performance analyzer makes two types of activity measurements. The first is a raw count of cycles. This measurement will show how intensive is the use of data or I/O ports. The software performance analyzer counts cycles and records the durations of cycles by sequentially scanning each of the memory regions being monitored. If a cycle of a particular type occurs in the memory region being monitored, it is counted. To make this measurement, each of the specified memory regions of a user's program are inspected one at a time for a period of 2.5 ms. All cycles of the specified type in this memory region are counted. In addition, a timer records and sums the lengths of these counted cycles. At the end of the 2.5-ms period the results are recorded and the next memory region is inspected.

After scanning each of the user's memory regions once, the analyzer has, in essence, completely dissected a 2.5-ms trace of all cycles for all memory regions. It used approximately 2.5 ms times the number of memory regions to do it, and it looked at different samples for each memory region. Assuming that the user's program is repetitive, the statistics of this sampling will approximate the true execution of the program. A maximum of 254 user-defined memory regions can be inspected once each to capture 2.5 ms of trace in approximately 640 ms.

Fig. 2 shows the result of a typical memory activity measurement. This measurement shows that the user program is spending 17% of the execution time of the processor accessing a memory region labeled data. The majority of the processor's time, 66%, is being spent accessing memory regions that are not defined. The amount of time that has been completely processed is 5.3 seconds. This implies that if you could make a trace of all cycles for 5.3 seconds and sort all of the cycles into the appropriate memory ranges, a display like that of Fig. 2 would appear.

A limitation of this type of cycle counting measurement occurs in trying to measure the performance of a program. Raw cycle counts do not give an adequate measurement of a function's performance because they exclude memory cycles associated with program execution. Consider the example of trying to measure the performance of the two functions, a_func and b_func. One measurement of the performance of a_func and b_func using a raw cycle count of memory activity might yield a histogram like Fig. 3a. The undefined activity is the activity associated with reads and writes to the stack and variables. Variables in embedded programs are typically located outside the address range of the instructions of the

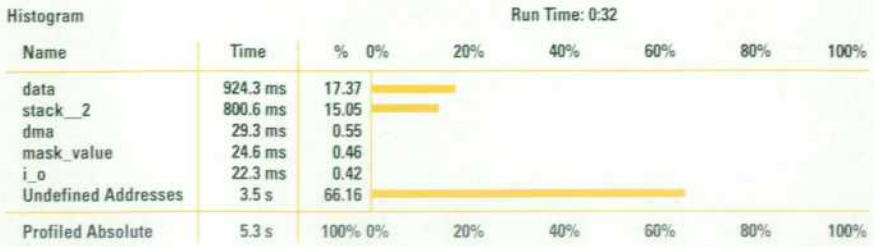


Fig. 2. Software performance analyzer memory activity measurement.

function. Thus, the cycle counting (memory activity) measurement only records opcode fetches or instruction execution in the range of the function and misses the associated read and write operations that occur outside the range of the function.

Program Activity Measurements. A different activity measurement overcomes this limitation by using the opcode cycle as the key cycle to determine if cycle recording should be turned on or off. If the opcode cycle is in the active memory region it will be recorded, but in addition all other cycles occurring after it will also be recorded until the next opcode. This allows the software performance analyzer to record the opcode and all cycles associated with the opcode. In this program activity measurement, all of the time will be properly divided among all of the active functions, as shown in Fig. 3b. The undefined cycles are allocated to the functions that generated them, thereby giving a more correct understanding of program operation.

Fig. 4 shows the results of making a program activity measurement. The histogram in Fig. 4 shows that the apply_productions module is using most of the processor resources (51%). The table in the lower half of Fig. 4 is just an expansion of the information shown in the upper histogram. A variety of additional information is provided for each of the captured memory regions. In particular, the number of cycles and the total time that the cycles have been active are recorded. The time per cycle is a measure of the average execution time of a cycle. The mean is an estimate of the amount of time that each memory region is active in a one-second period. The standard deviation or variance indicates the variation about the mean.

The advantage of program activity measurements is that they allow the user to define events that cover large segments of memory. The user can define an event for each of the libraries of a program, and the software performance analyzer can quickly determine which library is using the most processor resources. With this information, the user

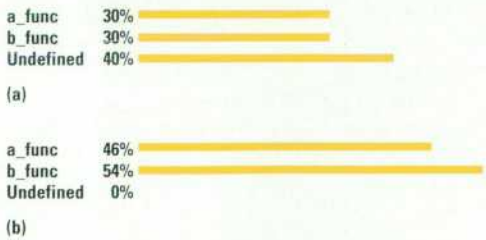


Fig. 3. Software performance analyzer activity measurements. (a) Memory activity measurement. (b) Program activity measurement.

can define events to represent each of the functions in the slow library and make a new profile measurement to find out which function is taking too much time.

After isolating in a general manner the libraries and modules that are taking too long to execute, the user can focus more directly on exactly which functions are perfoming poorly. In addition, it can be determined if a function (such as malloc) is being called too often or if a function is on occasion behaving in an unexpected manner, leading to a single long execution time for the function. Finally, the user can examine the time between specific points of the program and determine if time-critical intervals are within specifications.

Duration Measurements

A software performance analyzer duration measurement answers the last four questions listed above. Duration measurements are real-time, nonsampled measurements. The measurements continuously capture information about all of the functions or intervals selected. No sampling occurs and each start and exit of each of as many as 84 functions or intervals are continuously recorded. The only limitation is that function or interval start and exit address events must not continuously occur at an average rate higher than one every 100 μs. Bursts of less than 500 events can be handled even if they are only nanoseconds apart, but continuous sub-100-μs intervals will lead to situations where the software performance analyzer cannot keep up. This will be explained in more detail later and in most typical situations is not a constraint that a user has to deal with. The software performance analyzer can make interval and three different function duration measurements.

Interval Duration Measurements. Interval duration measurements are address point-to-point measurements. These measurements reveal how long it takes to get from one point to another in a program. To make this measurement, the software performance analyzer isolates the unique address points, qualifies the address points with the proper status, and then stores the address points along with a time tag. A dedicated software performance analyzer microprocessor then evaluates and matches the start and end points of each interval and generates an event occurrence. The time associated with the event occurrence is calculated and tabulated.

The dedicated software performance analyzer microprocessor can process a start and exit pair and do all of the associated math tabulation in less than 85 microseconds. The results of an interval duration measurement are then transmitted to the host interface, which translates the saved values into floating-point values and presents the results to the user.

Histogram

Run Time: 0:31

Name	Time	%	0%	20%	40%	60%	80%	100%
apply_productions	2.6 s	51.58						
stack_library	802.1 ms	15.71						
scan_string	354.8 ms	6.95						
math_library	241.5 ms	4.73						
move_byte	218.4 ms	4.28						
Undefined Addresses	964.2 ms	18.89						
Profiled Absolute	5.2 s	100%	0%	20%	40%	60%	80%	100%

Table

Run Time: 0:31

Name	Cycles	Time	Time %	Mean(1s)	StDv(1s)	Time/cyc
apply_productions	4.81E06	2.6 s	51.58	515.8 ms	443.4 ms	546.9 ns
stack_library	1.49E06	802.1 ms	15.71	157.1 ms	254.4 ms	536.8 ns
scan_string	669250	354.8 ms	6.95	69.5 ms	207.4 ms	530.1 ns
math_library	443201	241.5 ms	4.73	47.3 ms	119.9 ms	544.8 ns
move_byte	406905	218.4 ms	4.28	42.8 ms	132.3 ms	536.8 ns
Undefined Addresses	1.79E06	964.2 ms	18.89			
Profiled Absolute	9.62E06	5.2 s	100%			

Fig. 4. Software performance analyzer program activity measurement histogram and table.

A typical interval duration measurement is shown in Fig. 5. Different intervals can be measured, such as the time from the exit of a_func to the entry of a_func or the time between successive calls to b_func or the time between the start of c_func and the start of e_func. The interval duration table provides additional information about the interval duration measurement.

Function Duration Measurements. Function duration measurements are measurements of the length of time it takes a function to execute. The software performance analyzer can make three types of function duration measurements. These are function duration including all calls, function duration excluding all calls, and function duration excluding profiled calls. The first (function duration including all calls) will measure the time from the start of the function to the end of the function and include all time spent executing code associated with functions that are called from a function. This including calls measurement will also include time servicing an interrupt if an interrupt occurs in the middle of a function being measured.

The second duration measurement (function duration excluding all calls) will only record the time spent executing the selected function. All calls and all time spent servicing interrupts will be excluded from the measurement.

The third duration measurement (function duration excluding profiled calls) will include into the measurement of function duration all time associated with the function and all calls except those functions that are also being profiled. An interrupt function will be included or excluded from the time associated with the measurement of a function depending upon whether the interrupt is being profiled.

To make function duration measurements the software performance analyzer isolates the start and end address points, qualifies the address points with the proper status, and then stores them along with a time tag. The dedicated software performance analyzer microprocessor then reads the stored events, matches the start and end points of each function, and generates an event occurrence. The time associated with the event occurrence is then calculated and tabulated.

All this is similar to how interval durations are calculated except that two additional hardware and dedicated microprocessor operations are required. The software performance analyzer hardware, in addition to isolating the start and end points of a function, will also recognize the address range of a function. This is used with the excluding all calls function duration measurement to stop an internal timer when the function being measured is not in range.

A second hardware consideration is the prefetch correction circuitry. Prefetch correction circuitry removes 90% of all of the prefetches associated with the start and end address points of a function. A program loop near the exit of a program may prefetch the exit and possibly the entrance to a second function. An algorithm using the in-range feature of a function removes most of this prefetching. Addresses that

Histogram

Name	Time	%	0%	20%	40%	60%	80%	100%
a_func_end__a_func	12.5ms	0.12						
b_func__b_func	10.1 s	99.87						
c_func__e_func	3.9 s	38.25						
Undefined Addresses	?	?						
Profiled Absolute	10.1 s	100%	0%	20%	40%	60%	80%	100%

Table

Name	Calls	Time	Time %	Max	Min	Mean	Std Dev
a_func_end__a_func	1193	12.5 ms	0.12	10.5 µs	10.5 µs	10.5 µs	0.0 µs
b_func__b_func	1191	10.1 s	99.87	8.5 ms	8.5 ms	8.5 ms	0.0 µs
c_func__e_func	1192	3.9 s	38.25	3.2 ms	3.2 ms	3.2 ms	0.0 µs
Undefined Addresses	?	?	?				
Profiled Absolute	3576	10.1 s	100%				

Fig. 5. Software performance analyzer interval duration measurement.

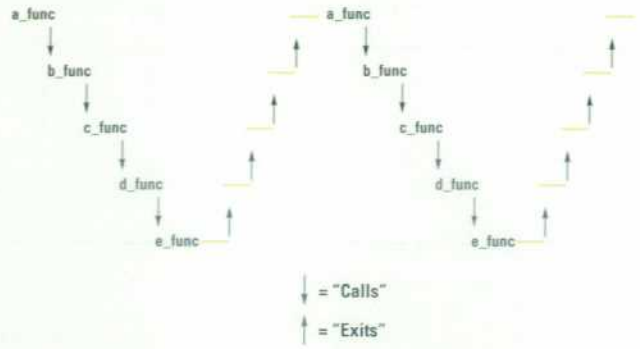


Fig. 6. Example program. Function a_func is called repetitively. Function a_func calls b_func, and so on.

are prefetched and get through the hardware correction algorithm are then typically removed by the dedicated software performance analyzer microprocessor. The software performance analyzer microprocessor contains an internal stack that attempts to mimic the call stack of the program. This stack contains each of the function start calls. When a function exit occurs the stack is searched to find the appropriate function entry. Algorithms used in association with the call stack attempt to minimize the occurrence of unused prefetch.

Consider the case of the five functions (a_func, b_func, c_func, d_func, e_func) all of which are identical in execution time. Assume that a_func calls b_func, which in turn calls c_func, which then calls d_func, and finally, d_func calls e_func. Also assume that a_func is repetitively called so that as soon as a_func exits, it is called again.

A picture of the execution of these functions might look something like Fig. 6. Assume that the user only measures functions a_func, b_func, and c_func. Software performance analyzer measurements of these functions might show results like those shown in Fig. 7. Observe the differences in the three types of duration measurements. Each measurement provides information about the duration of a function in a manner that a user may want to observe it.

Call Stack. The dedicated software performance analyzer microprocessor is responsible for combining start and exit function events to create a function duration time. A simple approach is just to subtract the time associated with the entry event from the exit event time to create a time duration. This will work in an including calls measurement but not in an excluding calls measurement. The excluding calls measurement must subtract out the time of called functions from the duration of the function currently being measured.

To overcome this limitation and others involving prefetch, the dedicated software performance analyzer microprocessor uses its duplicate call stack. Proper calculation of function durations is dependent upon the validity of this internal stack.

Statistics. The dedicated software performance analyzer microprocessor is responsible for determining function or interval duration and summing the duration results for statistical analysis as quickly as possible ($< 100\ \mu s$). The approach taken is to use integer math and avoid floating-point conversions and floating-point math. Thus, every event duration that is recorded is summed into a total duration for the particular event. The event duration is then compared to the current maximum and minimum values to determine these statistics. Finally, the event duration is squared and summed for a standard deviation calculation to be performed later.

Each event is summed by the dedicated software performance analyzer microprocessor into an event history, which resides with the dedicated software performance analyzer microprocessor. The integer values are moved to the host, converted to double-precision floating-point values, and processed by the host processor to generate the maximum, minimum, mean, and standard deviation values presented to the user.

Address Alignment. A user interface feature is the ability of the interface to look up function and static variable symbols automatically and define events for them. It does this by examining the symbol data base and locating function and static variable symbols. These symbols are converted to address ranges that are aligned to the microprocessor fetching address. For example, the microprocessor address range of a function may be 1002h through 1247h. The fetched addresses of this microprocessor may be on 4-byte boundaries and may appear as 1000h and 1244h. The software performance analyzer corrects for this and defines the start of this function on the occurrence of address 1000h and the exit of this function on the occurrence of address 1244h.

A second problem now appears if a function precedes the function discussed and ends on address 1001h. If this preceding function is also aligned it will appear to exit on address 1000h while the second function starts on address 1000h. This then becomes a problem for the software performance analyzer that is impossible to solve. If address 1000h is observed, is it a function exit or a function entry? To correct this problem the software performance analyzer allows the start and exit addresses to be adjusted. In this particular situation the user may wish to push the start addresses forward by two bytes so that the new start address of the original function is now on the fetched address 1004h instead of 1000h. Adjusting the addresses to align them on 32-bit (4-byte) boundaries will sometimes introduce a small error, which is insignificant in most cases.

Markers. If a user is using markers, the software performance analyzer will automatically define both the address range of the function and the start and end addresses of the markers. The function duration and interval duration measurements will then use the marker addresses.

Markers are write statements placed at the start and end of each function. The software performance analyzer uses these statements to determine function duration and interval duration. One advantage of markers is that the user can turn on the microprocessor instruction cache and data cache (provided that write statements can write through the cache) and the software performance analyzer can still make valid performance measurements.

To use the marker feature of the software performance analyzer, certain conditions need to be satisfied. The software performance analyzer must have unique start and end markers for each function that is to be measured. The best way to provide these is to add two static variables (unsigned short

Function Duration Include Calls:

Name	Time	%	0%	20%	40%	60%	80%	100%
1 a_func	10.2 s	99.75						
2 b_func	8.2 s	79.85						
3 c_func	6.1 s	59.87						
Undefined Addresses	19.2 ms	0.19						
Profiled Absolute	10.2 s		0%	20%	40%	60%	80%	100%

Name	Calls	Time	Time %	Max	Min	Mean	Std Dev
1 a_func	100	10.2 s	99.75	10.2 ms	10.2 ms	10.2 ms	0.5 µs
2 b_func	100	8.2 s	79.85	8.2 ms	8.2 ms	8.2 ms	0.5 µs
3 c_func	100	6.1 s	59.87	6.1 ms	6.1 ms	6.1 ms	0.5 µs
Undefined Addresses	?	19.2 ms	0.19				
Profiled Absolute	300	10.2 s					

Function Duration Exclude Calls:

Name	Time	%	0%	20%	40%	60%	80%	100%
1 a_func	2.0 s	19.94						
2 b_func	2.0 s	19.96						
3 c_func	2.0 s	20.01						
Undefined Addresses	4.1 s	40.08						
Profiled Absolute	10.2 s	100%	0%	20%	40%	60%	80%	100%

Name	Calls	Time	Time %	Max	Min	Mean	Std Dev
1 a_func	100	2.0 s	19.94	2.0 ms	2.0 ms	2.0 ms	0.5 µs
2 b_func	100	2.0 s	19.96	2.0 ms	2.0 ms	2.0 ms	0.5 µs
3 c_func	100	2.0 s	20.01	2.0 ms	2.0 ms	2.0 ms	0.5 µs
Undefined Addresses	?	4.1 s	40.08				
Profiled Absolute	300	10.2 s	100%				

Function Duration Exclude Profiled:

Name	Time	%	0%	20%	40%	60%	80%	100%
1 a_func	2.0 s	19.94						
2 b_func	2.0 s	19.96						
3 c_func	6.1 s	60.01						
Undefined Addresses	9.2 ms	0.09						
Profiled Absolute	10.2 s	100%	0%	20%	40%	60%	80%	100%

Name	Calls	Time	Time %	Max	Min	Mean	Std Dev
1 a_func	100	2.0 s	19.94	2.0 ms	2.0 ms	2.0 ms	0.5 µs
2 b_func	100	2.0 s	19.96	2.0 ms	2.0 ms	2.0 ms	0.5 µs
3 c_func	100	6.1 s	60.01	6.1 ms	6.1 ms	6.1 ms	0.5 µs
Undefined Addresses	?	9.2 ms	0.09				
Profiled Absolute	300	10.2 s	100%				

Fig. 7. Three types of software performance analyzer function duration measurements for the example program of Fig. 5.

to save memory space) to each function that the user wants the software performance analyzer to measure.

The names of the start variable and the end variable should be the same as the name of the function with the addition of a prefix. For example, the function main could have a start variable labeled s_main and an end variable labeled e_main. At the start of the function a value must be assigned to s_main. This can be zero or any other value, since the software performance analyzer does not read data values. Similarly, at the end of the function a value must be assigned to the variable e_main. The user must write values to the markers because the software performance analyzer is looking for writes to these specified variables. If the user chooses to use the prefix s_ for the start marker and e_ for the end marker of a function, these same prefixes must be used for all functions that the user wants the software performance analyzer to recognize.

One method of adding markers to a program is to add #ifdef statements at the appropriate places, as shown in Fig. 8. The use of the #ifdef statements will allow the user to conditionally compile the markers in and out of the code.

To configure the software performance analyzer to use markers, the user sets the shell variable HPSPAMARKERS="yes s_ e_" before starting the emulation session. All functions then defined will pick up the full address ranges of the function and marker addresses. The marker addresses will be automatically used to make function and interval duration measurements.

Software Performance Analyzer Architecture

The block diagram of the HP B1487 software performance analyzer is shown schematically in Fig. 9. The HP B1487 hardware consists of three main sections: the microcontroller core, the data acquisition section, and the host interface.

```
#ifdef MARKERS
    static short int s_testvalue;
    static short int e_testvalue;
#endif

boolean
testvalue(myvalue)
int myvalue;
{
#ifdef MARKERS
        s_testvalue = 0;
#endif
        if (myvalue > 100)
        {
#ifdef MARKERS
        e_testvalue = 0;
#endif
            return(TRUE);
        }
#ifdef MARKERS
        e_testvalue = 0;
#endif
        return(FALSE);
}
```

Fig. 8. Using ifdefs to insert markers in a function.

The microcontroller core consists of a processor, 256K bytes of RAM, 128K bytes of electrically erasable ROM, interrupt control, and a delta-time interrupt generator. The microcontroller is responsible for coordinating and sequencing all aspects of the measurement system. This includes programming of the hardware for the specified measurement and processing the results into a format that is suitable for transmitting to the host. Because of the large amount of high-speed data processing required, a 68EC030 was chosen as the controlling processor.

The ROM holds all of the operating and performance verification code (firmware). Since ROM is slower than RAM, the contents of the ROM are copied into RAM after reset. Delta-time interrupts are generated every 2.5 ms by a down-counter derived from the processor clock. These interrupts are used to implement a real-time clock and help to control activity measurements that are running in a statistically sampled mode.

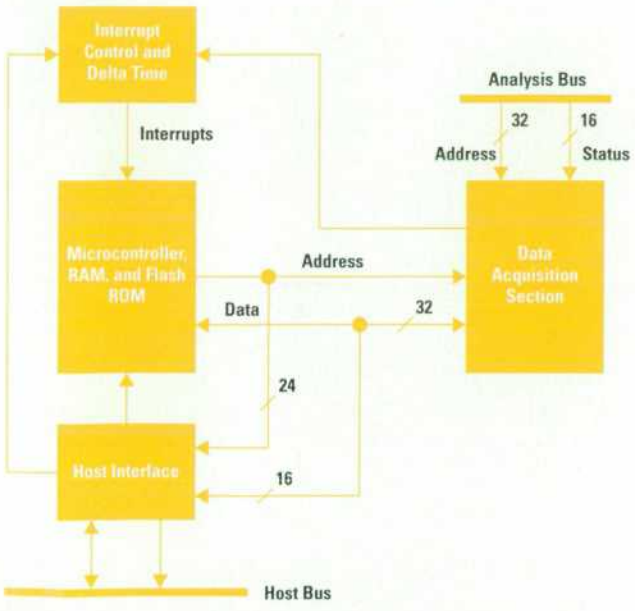


Fig. 9. Hardware block diagram of the HP B187 software performance analyzer.

Data is transferred between the HP B1487 and the HP 64700A emulation cardcage via the host interface. This interface consists of a dual-port RAM (8K by 16 bits) and a control register. When the cage is first powered up, the microcontroller is held in a reset state. The software performance analyzer is released from reset when firmware changes the state of the reset bit in the control register.

Data (and commands) are exchanged between the host and the microcontroller via dual-port RAMs. When the host writes to a command location within this memory a microcontroller interrupt is generated. The microcontroller reads the command location, clearing the interrupt. It then processes any data that the host has placed in the RAM.

When the microcontroller wants to signal the host it writes to another command location within the RAM. This action sets a bit in a status register that is visible to the host processor in the cardcage. When the host sees this bit set, it reads the command location and clears the bit.

The data acquisition section performs the actual measurements. Fig. 10 is a simplified schematic diagram of this hardware block. Its main components are ranging hardware, counter/timers, state machines, and high-speed first-in, first-out (FIFO) buffers.

The ranging hardware is responsible for identifying the incoming events. An event can be a 32-bit address or address range. The range comparators are constructed using 64K-by-4-bit high-speed static RAMs. Two stages of RAMs are used with a pipeline register between them. Without pipelining, the goal of 25-MHz operation could not have been met within cost goals. The range information is programmed into the RAMs by the onboard microprocessor from data provided by the host workstation. These high-speed RAMs are called *compare RAMs*.

Measurement control is handled by two state machines. One is used for the activity measurements and the other for the time duration measurements. The state machines are implemented in programmable array logic (PALs), one PAL for each state machine. PALs gave us the flexibility to change complex measurement algorithms during development. They also allow for field upgrading and enhanced measurements in the future.

Activity Measurement Hardware

All measurements involve the counting or timing of events. This requires the design of two sets of counters, one for counting events (state counting) and one for counting elapsed time. Storing or sampling of time counts requires careful design. The counter must be able to run at high speed to give the time resolution needed. It must also have sufficient width to record long elapsed times. This often leads to the counter's still changing its output state long after the input clock tick has arrived.

To solve this problem a Gray-code count sequence is used. Gray code permits only one bit to change per clock pulse. If the store strobe (the signal that reads the clock state) occurs during a count transition the maximum error is ±1 count. Since the clock source is 50 MHz, this translates to a ±20-ns error per count store.

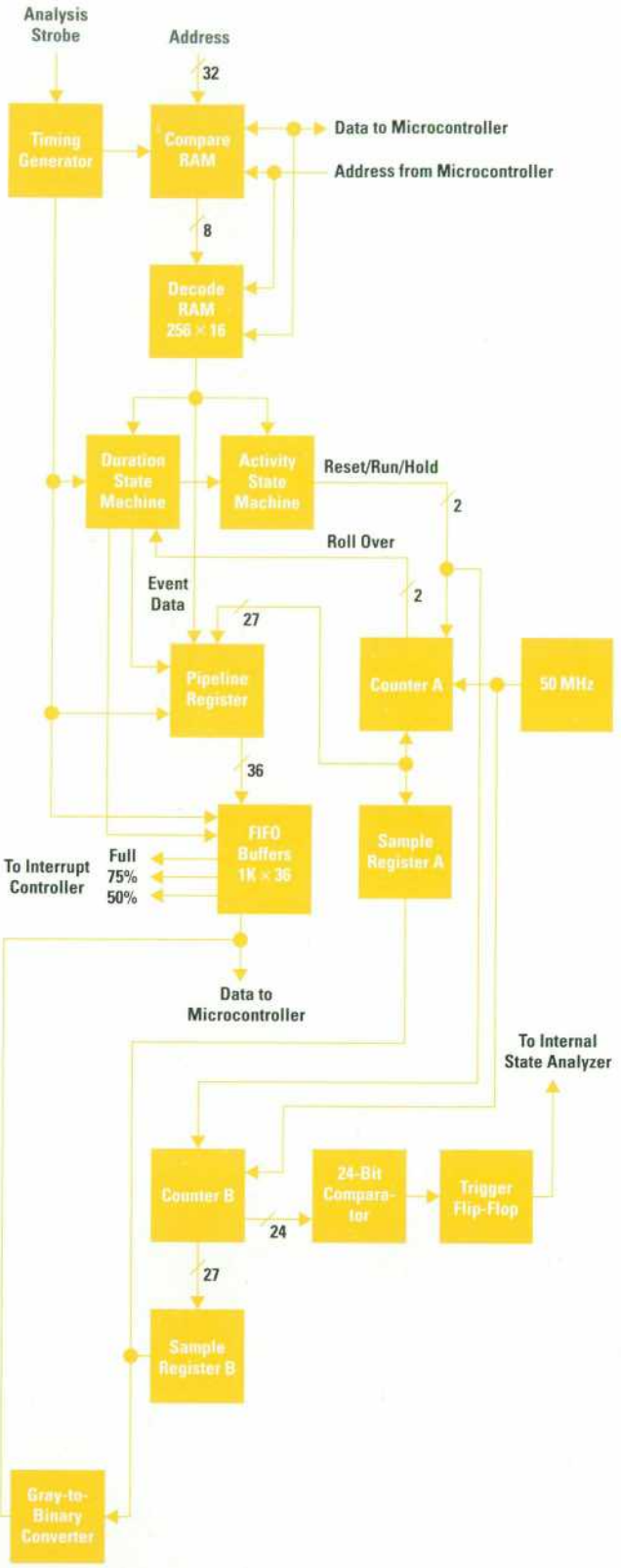


Fig. 10. Simplified block diagram of the data acquisition section of the software performance analyzer.

One drawback of Gray code counting is that the counts must be converted to binary at some point. This could be accomplished in firmware, but this would seriously reduce the effective data processing speed. For this reason the Gray code is converted to binary by a hardware decoding circuit, implemented using PALs. Conversion time is approximately 230 ns for a 27-bit Gray-code-to-binary conversion.

Activity measurement ranges are defined by programming up to 254 ranges into the compare RAMs. A second bank of RAM, called the *decode RAM*, is programmed to provide a true output when it identifies a current range of interest coming from the compare RAM. The activity state machine observes the state of this signal and the type of cycle in which it occurred (i.e., opcode fetch, memory read/write, etc.). The counters are then turned on or off depending on the type of activity measurement being performed. Each range of interest is scanned for 2.5 ms. At the end of this time interval the onboard microcontroller strobes the contents of all counters into the sample registers simultaneously. The decode RAM is then reprogrammed to observe another range of interest. By programming only the compare RAMs with all of the ranges of interest and only reprogramming the decode RAM between ranges, dead time between range samples is held to an absolute minimum. The measurement is then allowed to continue for another fixed length of time. The microcontroller is responsible for accumulating these counts and transmitting them to the host.

Duration Measurement Hardware

The basic function of duration measurements is to record the time between the occurrence of two events in the user's program. These can be function entry or exit points, access to memory, I/O, or the processor's writing out markers because of instrumented code. It is very important that the duration measurements properly record the time interval for every execution under observation. Therefore, a large fraction of the design effort was spent to ensure that data would be acquired by nonsampling methods. This ensures that all interval execution times, no matter how small, are represented in the measurement results. Also, maximum time resolution was required (40 ns) for very long intervals (hours). Furthermore, it was important to minimize or remove the effect of processor prefetching from the measurement results.

The design implemented is able to measure up to 84 function intervals directly (nonsampled) with a time resolution of ±40 ns for very long periods (months). Up to 95% of prefetch conditions can be corrected for by a combination of hardware and firmware processing. Measurement results can include or exclude time spent in functions called by the function under observation.

A duration measurement starts when a qualified event is recognized by the address comparators. The duration state machine then decides (depending upon the measurement type) if the event number and time should be stored into the pipeline register. The pipeline register can be flushed if the state machine determines a prefetch has occurred. Any previous valid entries that may be contained in the pipeline register are written into a first in, first out buffer (FIFO). The state machine considers a prefetch to have occurred if the current event under observation is exited via the function return point and then is reentered without going through the function entry point (Fig. 11). Also, if a function is entered more than once without exiting through the return point the state machine will flush all but the last entry point from the pipeline register. The user can defeat this

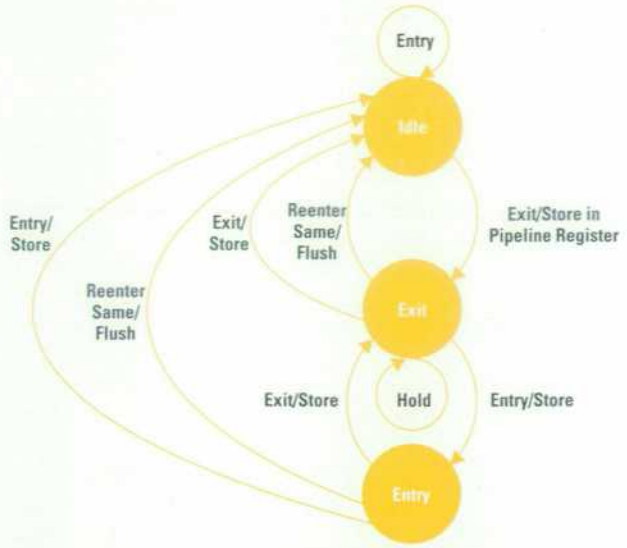


Fig. 11. Simplified prefetch correction state diagram.

action by marking functions as recursive. The elimination of prefetches prevents the FIFO from overflowing when code loops exist at the end of a function. If this situation were not properly handled in hardware, prefetches could cause the FIFOs to fill faster than the microcontroller could process them and errors would occur.

The FIFO provides an interface to the microcontroller. As long as events enter the FIFO at an average rate less than one every 100 μs the controller can continue to process data in a nonsampled mode. If the FIFO overflows an interrupt is sent to the microcontroller and the user is notified. Interrupts are also provided for half and 75% full. The firmware makes use of these to delay noncritical tasks until the controller has caught up with the data stream. This allows the HP B1487 to maintain a very high rate of data throughput.

Duration events may have very large time intervals between them. To measure large intervals that exceed the maximum limit of the time counter (2.6 seconds) the counter provides two signals to the state machine: half count and full count. These signals generate a FIFO store if none is pending as a result of an entry or exit. By using these intermediate time counts the firmware can compute large time intervals between events.

An additional feature of duration measurements is the ability to send a trigger signal to the internal state analyzer of the HP 64700 cardcage whenever an interval exceeds a user-specified limit. This is useful for correlating a time limit with the state trace of the target microprocessor. This measurement is limited to one event and a maximum of 2.5 seconds. This is accomplished by first having the decode RAM tag the interval of interest by having the time event signal go true when the interval of interest is recognized. The state machine then activates a timer when the entry point of the interval is recognized. A comparator monitors the state of the counter and triggers a flip-flop if the interval exceeds a user-specified time limit. The counter is reset to zero by the state machine when the end of the interval is recognized.

Conclusion

The HP B1487 represents a significant advance in the field of software performance analysis. As an integrated part of the HP 64700 product family and the HP 64700 debug environment it helps provide the embedded systems designer with the powerful tools needed to design complex and demanding embedded applications.

Acknowledgments

The authors would like to acknowledge the efforts of Grant Grovenburg, the HP B1487 firmware designer, and Mike Upham, who designed the host workstation communication protocols. In addition, we wish to thank Grant for his excellent paper on the operation of his algorithm for programming the compare RAMs. We regret that space limitations prevented our including it in the final revision of this article.

Authors

April 1993

6 Microwave Signal Generators

William W. Heinz

R&D section manager Bill Heinz has contributed to the development of various microwave components and products at the Stanford Park Division since joining HP in 1966. He initially designed microwave oscillators and other microwave components and later was a project manager for the HP 8673A/B/C/D synthesized signal generators. He has also been section manager for power and noise products and microwave sources. Bill was born in Havana, Cuba and attended the Polytechnic Institute of Brooklyn, from which he received a BEE degree in 1960 and an MSEE degree in 1962. He has coauthored several articles on ferrite devices and is a senior member of the IEEE, a member of the Communications Society, and a member of the Group on Microwave Theory and Techniques. In the past, Bill was actively involved in the Boy Scouts and is now a mentor for a college preparatory program. He has two sons and enjoys photography, hiking, listening to music, and traveling abroad.

Ronald E. Pratt

Born in Buffalo, New York, Ron Pratt attended the New Jersey Institute of Technology, from which he received a BSEE degree in 1967. He joined HP's Microwave Division the same year and developed power sensors, noise source components, mixers, and other microwave hardware used in numerous HP products. For the last ten years, he has managed projects in the areas of power measurement, noise figure measurement, and signal generators at the Stanford Park Division, including the HP 70340A/41A signal generator projects. He contributed to the 7th edition of Reference Data for Radio, Electronics, Computers, and Communications Engineers. He's also a member of the IEEE and of the Groups on Microwave Theory and Techniques and Instrumentation and Measurement. For nearly 20 years, he taught a microwave measurement course at a local college. Ron and his wife have two daughters in college. His leisure interests include cabinet making and collecting tinplate toy trains.

Peter H. Fisher

A software engineer at HP's Stanford Park Division, Peter Fisher was born in Wiesbaden, Germany, and attended the Engineering School in Dieburg, from which he received his Diplom Ingenieur in electrical engineering in 1982. He joined HP's Stanford Park Division in 1984, and has developed software for a number of microwave products, including the HP 8370 and HP 70340 Series signal generators. Earlier, he contributed to the development of the HP 8980A vector analyzer, the HP 8981A vector modulation analyzer, the HP 8782B vector signal generator, and the HP 11757B multipath fading simulator. He is working on a project to develop a common software platform for HP test and measurement products. Peter served in the German Navy for fifteen months. He is married and lists boardsailing, hang gliding, and flying airplanes as favorite leisure activities.

12 Fundamental Frequency Synthesis

Brian R. Short

Brian Short has been with HP's Stanford Park Division since 1984. He was one of the design engineers for the RF phase-locked loops for the HP 8370 Series and HP 70340 Series signal generators. Earlier, he was a production engineer and he's now designing video test equipment. Brian was born in Redwood City, California and attended Washington State University, from which he received a BSEE degree in 1984. He was awarded an MSEE degree by Stanford University in 1987. His outside interests include basketball, softball, and weight lifting.

Thomas L. Grisell

Design engineer Tom Grisell came to HP in 1965, the same year he graduated with a BSEE degree from San Jose State University. While at HP he continued his studies at Stanford University and received his MSEE degree in 1968. He has served as a design engineer for many HP products, primarily in the area of frequency synthesis. He was project manager for the HP 8444A tracking generator and was responsible for the design of the LO/Ref synthesizer assembly and other parts of the HP 8370 Series and HP 70340 Series signal generators. He's currently working on video products at the Stanford Park Division. Tom was coauthor of several previous HP Journal articles and is coinventor of a patent related to frequency generation for a split-comb frequency synthesizer. He's a member of the IEEE. A California native, Tom was born in San Francisco. He has two sons, is an amateur radio operator, and enjoys spending time working on his personal computer.

Edward G. "Bud" Cristal

A native of St. Louis, Missouri, Bud Cristal attended Washington University, from which he received an AB degree in mathematics and a BSEE degree in 1957 and an MSEE degree in 1958. He continued his studies at the University of Wisconsin, where he completed work for a PhD degree in 1961. He was a senior research engineer at Stanford Research Institute and an associate professor at McMaster University before joining HP Laboratories in 1973. After transferring to the Stanford Park Division in 1975, he worked on the development of several

signal generators before being named a project manager. He has been a project manager for several power sensor and power meter products and comanaged development of the frequency synthesis system for the HP 8370 and HP 70340 Series signal generators. Bud is the author or coauthor of more than 50 technical articles and has contributed to two books. His work on hairpin filter design and meanderline transformers has resulted in two patents. He was named an IEEE Fellow in 1980 and received the IEEE Microwave Application Award in 1973. He's also a member of the Electromagnetics Academy and the Society of Motion Picture and Television Engineers. He is married and has two children.

17 Microwave Chain

William D. Baumgartner

A native of Denver, Colorado, Bill Baumgartner studied electrical engineering at the University of Colorado, receiving his BSEE degree in 1979. Joining HP's Stanford Park Division the same year, he worked on fiber-optic sources and power measurement, the HP 8673C/D synthesized signal generators, the HP 8684R/Q waveguide power sensors, and noise figure meters. He was R&D project manager for the microwave chain of the HP 70340 and HP 8370 signal generator families, and now works on video products. Bill is a member of the IEEE and is active in his church. He and his wife and two sons enjoy exploring the outdoors while hiking, camping, and scuba diving. His other hobbies include woodworking and toy making.

John S. Brenneman

Development engineer John Brenneman joined HP in 1983, the same year he completed work for his BSEE degree from Pennsylvania State University. He continued his studies while at HP as an HP Resident Fellow and received an MSEE degree from the University of Illinois at Urbana-Champaign in 1988. He designed the broadband filters, couplers, and switches for the HP 8370 signal generator family, and in the past designed the microwave portion of the HP 8971B noise figure test set. Currently, he's responsible for manufacturing engineering for the HP 8780/82 vector signal generators. He coauthored two papers on microstrip antennas and is a member of the IEEE. His professional speciality is microwave circuit design and his interests include antennas and computer-aided design. John was born in Columbia, Pennsylvania, is married and has one daughter. He is active in his church. A practitioner of judo, he has achieved a first-degree black belt ranking. He likes to travel and lived in Vienna, Austria for seven years.

John L. Imperato

With HP's Stanford Park Division since 1983, R&D engineer John Imperato designed the ALC board and the GaAs divider circuitry in addition to doing system-level microwave power analysis for the HP 8370 and HP 70340 signal generator families. He also developed the algorithm for calibration and power flatness correction. Earlier, he was a microcircuit production engineer, pioneering production shipments of HP's GaAs MMICs. Currently, he's working on video products. Born in Rockaway Beach, New York, John received a BSE degree in electrical engineering from Duke University in 1983 and an MSEE degree from Stanford University in 1987. Before joining HP he conducted research related to the treatment of cancerous tumors with microwave heating. John has published other technical papers and is named as a coinventor in a patent on feedback control systems. He's also a member of the IEEE. An avid surfer and mountain bike rider, he is an active member of Save our Shores and the Surfrider Foundation. He's interested in solar energy and has installed solar panels at his home.

Douglas A. Larson

With HP since 1984, hardware engineer Douglas Larson has contributed to the development of several microwave products, including the HP 8370 and HP 70340 signal generator families, the HP 8770A/S arbitrary waveform synthesizers, and the HP 8791 frequency agile signal simulator. He is currently working on the HP 11759 Series of RF channel simulators. A graduate of Iowa State University, Douglas completed work for his BSEE degree in 1984. He is a member of the IEEE and his professional speciality is mixed analog and digital circuit design.

Ricardo de Mello Peregrino

A California native, Ric Peregrino was born in Palo Alto and attended the University of California at Davis, from which he received a BSEE degree in 1983. He joined HP's Stanford Park Division the same year and continued his studies at Stanford University, completing work for an MSEE degree in 1991. He designed microwave amplifiers, mixers, and a UHF attenuator for a frequency agile upconverter and contributed to the development of the modulation module for the HP 8370 and HP 70340 families of signal generators. He is currently working in the area of video electronics. His other professional experience includes work at the National Aeronautics and Space Administration and at a radio station. Ric's hobbies include music, sports, mathematics and physics.

Gregory A. Taylor

Development engineer Greg Taylor joined HP's Automatic Measurements Division in 1974 and later moved to the Stanford Park Division. He has been involved in the design of various signal generators and noise figure products, and designed the 10-MHz-to-1-GHz signal band for the HP 83731A/32A signal generators and the HP 70341A frequency extension module. He is currently working on video broadcast products. He is a graduate of the University of California at Santa Barbara (BSEE 1974) and of Stanford University, from which he received an MSEE degree in communications in 1978. Greg is married and enjoys scuba diving, skiing, and sampling wine from his collection.

30 Concurrent Engineering and Manufacturing

Christopher J. Bostak

A manufacturing development engineer at the Stanford Park Division, Chris Bostak came to HP in 1989. His first project was related to the development of the HP 8370 and HP 70340 signal generator families, including the design of the final tests in manufacturing and design of the computer network used on the production line. He is now developing manufacturing processes and tests for a new video product. A native of Fairfax, Virginia, Chris received a BSEE degree from the University of Virginia in 1989. His MSEE degree from Stanford University is expected in 1993. His professional interests include digital communications and signal processing. Hobbies include cooking, baking bread, hiking, gardening, and reading.

Camala S. Kolseth

Industrial engineer Cam Kolseth came to HP's Stanford Park Division in 1989 and has been a production engineer for several manufacturing projects. She has worked on printed circuit board manufacturing and microcircuit subassembly, led the implementation of Kanban and JIT systems in printed circuit, microcircuit, and sensor assembly, and implemented a PC-based database for semiautomated printed circuit board equipment. For the HP 8370 and HP 70340 signal generator families, she implemented online video image procedures, line layout, the material flow strategy, and the process simulation model. She's currently involved in planning and executing a self-directed work group environment for the division's manufacturing organization. Before joining HP she worked at Electronic Data Systems and at McDonnell-Douglas Helicopter Company as a member of the manufacturing research and development group. She is the author or coauthor of two papers related to computer integrated manufacturing and is a member of the Institute of Industrial Engineers. Cam was born

in Madison, Wisconsin and attended the University of Wisconsin (BS Industrial Engineering 1986) and Arizona State University (MS Industrial Engineering 1989). She currently resides in San Francisco. Her outside interests include jogging, mountain bicycling, and reading.

Kevin G. Smith

Born in Alexandria, Louisiana and raised in Houston, Texas, Kevin Smith is a graduate of the University of Houston (BSEE 1981) and Stanford University (MSEE 1986). Before joining HP in 1990, he worked on radio receiver design at ESL/TRW and on arctic geophysical and oceanographic instrumentation for Exxon Production Research. For the HP 8370 and HP 70340 signal generator families, his responsibilities included the hardware, software, and image integration for the pretest process. He was also responsible for instrument power budgets for the HP 8370 Series and for qualification, modification, and test of switching power supplies. He is currently working on wideband communications and video projects. Kevin is a registered professional engineer in Texas, a licensed amateur radio operator, and active in the IEEE. His professional interests include analog and RF electronics, digital signal processing, electromagnetic compatibility, and design-for-manufacturability/design-for-testability methods. He's married and enjoys reading, jogging, backpacking, and tinkering with old radios. He comments that he is also trying to learn to play Scottish bagpipes as well as his father does.

38 Microwave Sweepers

Alan R. Bloom

A design engineer at HP's Microwave Instruments Division, Alan Bloom was born in Gettysburg, Pennsylvania and attended Wesleyan University (BA physics, 1972) and Rensselaer Polytechnic Institute (MS engineering, 1976). He designed radio equipment at an Ohio firm before joining HP in 1979. He has been a components engineer and designed most of the printed circuit boards for the HP 83550A RF plug-in. For the HP 83750 family of sweepers, he designed the CPU, timer, sweep generator, and rear-panel boards and developed firmware for the DSP. He has written articles for amateur radio magazines and sections of several amateur radio books. An amateur radio operator (N1AL), Alan also enjoys classical music and remodeling his home.

Jason A. Chodora

A California native, R&D engineer Jason Chodora was born in San Jose and attended the University of California at Berkeley. He received a BS degree in electrical engineering and computer science in 1983 and came to HP the same year. He was a production engineer at the Network Measurements Division for the HP 83590 Series RF plug-ins and designed analog circuits for the HP 83597A RF plug-in. He designed the synthesis and automatic level control circuitry for the HP 83750 Series sweepers. Jason is married and has a young son. An avid golfer, he jokes that he works at HP in his spare time. He also enjoys racquetball and camping with his family.

James R. Zellers

Jim Zellers is R&D project manager for HP 83750 Series sweepers at HP's Microwave Instruments Division. With HP since 1968, he has also served as project manager for the HP 83597A 0.01-to-40-GHz sweeper plug-in, the HP 853 spectrum analyzer display, and RF network analyzers. Born in Los Angeles, he received his BSEE degree from the University of California at Berkeley in 1966 and his MSEE degree from the University of Michigan in 1968. He is married, has two teenage children (one in college), and enjoys swimming, hi-fi music, and computers.

46 Sweeper Microcircuits

Eric V. V. Heyman

Development engineer Eric Heyman came to HP in 1980. He has worked on several 40- and 50-GHz frequency doublers and has been a production engineer for various amplifiers and oscillators. Now at the Microwave Instruments Division, he contributed to the development of the dual YIG oscillator for the HP 83750 Series sweepers. Eric was born in Wilmington, Delaware and graduated from Lehigh University with a BSEE degree in 1980. His MSEE degree was awarded by National Technological University in 1992. He is married and says that working on a degree in history has kept him off the golf course.

Rick R. James

A California native, Rick James was born in San Francisco and attended California State Polytechnic College at San Luis Obispo, from which he received a BSEE degree in 1980. After coming to HP the same year, he developed test systems for various microcircuits and MMS modules for network and signal analyzer products. His contributions to the development of the dual YIG oscillator for the HP 83750 Series sweepers include the design, test, and production engineering of the 2-to-11-GHz band and the switched amplifier filter detector microcircuit. Earlier, he was employed at TRW, where he worked on pulse code modulation for telephone systems. His professional interests include computer-aided design, microwave design, and test system development. Rick is married and has a one-year-old son. He enjoys playing with his son and working on his home in addition to golf and racquetball.

Roger R. Graeber

With HP since 1972, R&D engineer Roger Graeber designed the modulator amplifier and the 0.01-to-2-GHz heterodyne band microcircuits for the HP 83750 Series sweepers. Past HP projects include the HP 436A power meter, the HP 8662A synthesized signal generator, and the HP 83550A RF plug-in. Born in Oakland, California, he completed work in 1965 for an AA degree in electronics from the College of Marin and received a BSEE degree from the University of California at Davis in 1982. Before coming to HP he worked on analytical instruments, magnetometers, and frequency standards at Varian Associates. Roger is married and has a son and a daughter in college. A train and car collector, his trains range in size from HO gauge to three-foot narrow gauge. He owns a 1932 Nash and 11 1956 DeSotos, six of which run. He's also the 1956 Desoto Technical Advisor for the National Desoto Club.

52 3-GHz Pulse Generator

Hans-Jürgen Wagner

R&D section manager Hans-Jürgen Wagner was born in Ludwigsburg, Germany and completed work for his Diplom Ingenieur in electronic science at the University of Stuttgart in 1984. After joining HP's Böblingen Instruments Division in 1985, he contributed to the development of the HP 8131A pulse generator. He was a project manager for the HP 8133A pulse generator, and now has R&D responsibility for pulse, data, and function generators. He has published in the area of GaAs IC design and is interested in microelectronics. Hans-Jürgen is married and has three children. When time allows, he enjoys bicycling, skiing, soccer, and working on his house. Reading about physics and philosophy is another interest.

56 Pulse/Data Channel

Christoph Kalkuhl

Project engineer Christoph Kalkuhl joined HP's Böblingen Instruments Division in 1990 and contributed to the design of the pulse/data board for the HP 8133A pulse generator. He is now investigating digital design characterization hardware. Born in Sigmaringen, Baden-Württemberg, Germany, he studied electrical engineering at the University of Stuttgart, from which he received a Diplom Ingenieur in 1990. Christoph's favorite pastime is music. He sings in a choir, plays several instruments, and enjoys dancing.

60 3-GHz Pulse Generator Design

Peter Schinzel

Peter Schinzel was born in Stuttgart, Germany and received a Diplom Ingenieur in communications and theoretical electronics from the University of Stuttgart in 1987. After joining the Böblingen Instruments Division that same year, his first project was working on the hybrid technology and slope generation for the HP 8130A pulse generator. He contributed to the development of the timing board for the HP 8133A pulse generator and is now developing test fixtures. Peter's hobbies include bicycling and badminton.

Andreas Pfaff

A graduate of the University of Aachen, Andreas Pfaff received a Diplom Ingenieur in electrical engineering in 1989. He joined the Böblingen Instruments Division the same year and designed the output amplifier and pulse formatter for the HP 8133A pulse generator. He is now investigating a new stimulus-response system for digital device characterization. He's an expert in the design of thick-film hybrids as packages for high-frequency, high-bandwidth ICs. Andreas was born in Hanau, Germany and is interested in politics.

Thomas Dippon

With HP since 1991, Thomas Dippon is a project engineer at the Böblingen Instruments Division. He was born in Stuttgart, Germany and completed work for his Diplom Informatik from the University of Stuttgart in 1990. He contributed to firmware design for the HP 8133A pulse generator, and is currently working on software for digital device characterization. Before joining HP, Thomas was a software developer in the area of computer communications.

Thomas Fischer

A mechanical designer at the Böblingen Instruments Division, Thomas Fischer has been with HP since 1984. He was born in Stuttgart, Baden-Württemberg, Germany and attended the Engineering School in Esslingen, from which he received a Diplom Ingenieur in mechanical engineering in 1984. He has contributed to the mechanical design of several pulse and data generators, including the HP 8175A digital signal generator and the HP 80000 data generator system. He worked on EMC, mechanical design, and cooling the frequency divider IC for the HP 8133A pulse generator. Thomas lists CAD software as a professional interest, and model railways, woodworking, and reading as leisure activities. He is married and has two sons.

Allan R. Armstrong

With HP since 1986, Allan Armstrong is a product development engineer at HP's Microwave Technology Division. He has worked on pulse amplifiers for the HP 8130A/31A pulse generators and for the HP 71604B pattern generator, on frequency dividers, and on other GaAs ICs. He contributed to the product engineering, design and design support, and development of test methods for the GaAs output amplifier for the HP 8133A pulse generator. He is currently involved in circuit research on heterojunction bipolar transistor and MODFET processes. Allan was born in Whitby, England, and is a graduate of the Massachusetts Institute of Technology (BSEE 1986). He is the author or coauthor of four technical papers and is a member of the IEEE. His leisure activities include bicycle racing and touring, photography, hiking, gourmet cooking, and wine tasting.

73 Real-Time Frequency Analyzer

James W. Waite

Jim Waite has been with HP's Lake Stevens Instrument Division for eight years as a measurement software engineer in R&D. He received his BS degree in engineering physics in 1979 from the University of Colorado and his MSE degree in 1984 from the University of Washington, focusing on the areas of digital signal processing and underwater acoustics. A member of the IEEE Acoustics, Speech, and Signal Processing Society and the Acoustical Society of America, Jim takes pleasure in finding new applications for digital signal processing in the fields of acoustics and vibration. His contributions have included implementing digital swept sine software for the HP 3565S multichannel dynamic signal analyzer (DSA) and order tracking and real-time octave measurements for the HP 35665A two-channel DSA. Recently, he has been representing Hewlett-Packard on several ANSI standards committees dealing with noise issues. Outside of work, Jim sails a wooden boat and enjoys outdoor and travel photography. He and his family live in a 70-year-old home in a historic district near the HP plant.

82 RMON LanProbe II

Matthew J. Burdick

A native of Brooklyn, New York, Matt Burdick graduated in 1987 from Indiana University at Bloomington with BS degrees in both computer science and astrophysics. He joined HP's Information Networks Division in 1988 and worked on the SNA LU6.2 application programming interface for the HP-UX operating system. Now at the Network Test Division, he has worked on earlier versions of HP LanProbe firmware and on the HP ProbeView user interface. More recently, he was a software developer for the RMON Management Information Base for the HP LanProbe II, responsible for the filter, packet capture, and event groups. He's currently developing new features for a management station using the RMON data. Matt's professional interests include network management and SNMP protocol. He's married and likes reading and scuba diving.

90 Embedded Debug Environment

Robert D. Gronlund

Born in Fargo, North Dakota, Bob Gronlund attended North Dakota State University, completing work for a BS degree in architecture in 1980 and an MS degree in computer science in 1983. Before joining HP's Logic Systems Division in 1983, he designed building HVAC systems. At the Colorado Springs Division, he has been a member of the technical staff and an R&D project manager for cross-language tools and was project manager for the HP 64700 embedded debug environment. He's a member of the ACM and the IEEE and specializes in graphical interfaces, language systems, and building automation systems. He enjoys tutoring gifted students in local public schools. Bob is married and likes skiing, racquetball, personal computing, and reading.

Richard A. Nygaard, Jr.

Rick Nygaard has been with HP since 1977 and is an engineer/scientist at the Colorado Springs Division. He has designed hardware for several HP logic analyzers, including the HP 1611A, the HP 1610B, and the HP 64620. He also was project manager for the emulation portion of the UNIX user interface for the HP 64000-UX microprocessor development system. He was responsible for the emulator user interface for the HP 64700 debug environment. His work on time-tag counters and RAM-based multiple range comparison has resulted in three patents. Rick and his wife are the parents of a son and new twins, a boy and a girl.

John T. Rasper

A design engineer at HP's Colorado Springs Division, John Rasper joined the company in 1979. He has contributed to product development in the areas of debugger emulator connection and vector graphics displays and was responsible for high-level debugger system integration for the HP 64700 debug environment. In addition to his HP work, he has been a software consultant on control systems and has developed printed circuit board prototyping

systems and PBX systems. He is named as an inventor in a patent on a computer-controlled system for producing printed circuits. His professional interests include biological information processing systems and graphics. Born in Tokyo, John attended Northwestern University and received a BSEE degree in 1976. He's married and enjoys cycling, scuba diving, and skiing.

107 Software Performance Analyzer

Andrew J. Blasciak

An engineer at the Colorado Springs Division, Andy Blasciak has been with HP since 1979. He has worked on the HP 64310A software performance analyzer, developed emulator control cards and software-based dequeuers for Intel386 marker technology, and worked on embedded system software performance analysis. He was responsible for the overall definition and the hardware design of the HP B1487 software performance analyzer and is named as an inventor in a patent on performance analysis. Andy was born at Travis Air Force Base, California and received a BSEE degree from Clemson University in 1979. His outside interests include skiing, backpacking, alpine climbing, and bicycling.

David L. Neuder

With HP since 1979, R&D design engineer Dave Neuder was one of the software developers for the HP B1487 software performance analyzer. Earlier, he was the principal software designer for the HP Branch Validator and contributed to the development of HP Teamwork and the HP 64610 timing analyzer. Dave was born in Wyandotte, Michigan and is a graduate of Michigan State University (BSEE 1977 and MSEE 1979). He is named as an inventor in a patent on marking technology in timing analyzers, and is a member of the IEEE. Dave is married and has a baby daughter. He enjoys singing in choirs, skiing, backpacking, bicycling, and mountain climbing.

Arnold S. Berger

With HP's Colorado Springs and Logic Systems Divisions since 1979, R&D project manager Arnie Berger was a cathode ray tube engineer for the HP 1727A storage oscilloscope and was team leader for the HP 54300A probe multiplexer. He also managed the development of the HP 64700 Series emulators and was project manager for the HP B1487 software performance analyzer. Born in Brooklyn, New York, Arnie attended Cornell University, receiving his BS degree in materials science in 1966 and his PhD degree in the same field in 1971. His other professional experience includes work on metal fatigue at Ford Motor Company and study of crystalline defects in metals at Argonne National Laboratory. He is the author or coauthor of 25 papers in the fields of material science, solid state physics, and technologies related to HP products. His work on microprocessor development systems has resulted in a patent, and he's a member of the American Physical Society. An HP liaison to schools in his area, Arnie also built 30 HP 64700A emulators from excessed parts for donation to local universities. He is married, has a daughter, and likes bicycling and running. He also enjoys writing humorous captions for the "Immortal Works" section of Electronic Engineering Times, and has had 122 captions published.

HEWLETT-PACKARD
JOURNAL

April 1993 Volume 44 • Number 2

Technical Information from the Laboratories of Hewlett-Packard Company

Hewlett-Packard Company, P.O. Box 51827
Palo Alto, California, 94303-0724 U.S.A.

Yokogawa-Hewlett-Packard Ltd., Suginami-Ku Tokyo 168 Japan

HEWLETT PACKARD

5091-6920E