HEWLETT-PACKARD JOURNAL

AUGUST 1988

HEWLETT-PACKARD
JOURNAL

Articles

Research Reports

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 makes no warranties, express or implied, as to the accuracy or reliability of such information. 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 address on the back cover that is closest to you. When submitting a change of address, please include your zip or postal code and a copy of your old label.

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 presenting 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

This issue of the HP Journal presents the design of the HP PaintJet Color Graphics Printer, HP's first color inkjet printer. Produced by a marriage of thermal inkjet printing technology and a customer-focused product design, the PaintJet printer provides excellent color and print characteristics for personal computer users at an affordable price.

The PaintJet printer is the second major product based on the thermal inkjet technology HP has developed over the last decade. The disposable print cartridges, or pens, provided by this technology make possible low-cost, reliable, high-quality, intervention-free printers. A third HP inkjet product, the DeskJet printer, will be described in the next issue of the HP Journal (October 1988). The DeskJet printer produces high-quality black print for office correspondence.

The first HP inkjet product, the ThinkJet printer, was introduced in 1984. It has 96-dot-per-inch resolution, comparable to 24-wire impact printers, but it is quiet and nearly maintenance-free. The DeskJet printer, introduced in 1988, provides 300-dot-per-inch resolution on a variety of "plain" papers and has print characteristics approaching those of more expensive laser printers. Its print quality far exceeds that of 24-wire printers for about the same price.

The development of a product like the PaintJet printer incorporates contributions from many areas. The thermal inkjet technology was pioneered by HP Laboratories and extended by several HP Divisions including those at San Diego, California, Corvallis, Oregon, Vancouver, Washington, and Boise, Idaho. Today, most of the applied technology efforts are concentrated in the Thermal Inkjet Operations in San Diego, Corvallis, and Boise, and pioneering work continues at HP Laboratories. The San Diego Technology Center designed and manufactures the PaintJet pens and ink, developed many of the manufacturing technologies, codeveloped special papers, and worked with the San Diego Division to meet customer needs in the design. The Corvallis Inkjet Components Operation developed many of the fabrication techniques and manufactures the pen orifice arrays and substrates. The San Diego Division, known for its engineering plotters, developed the product concept. This required detailed understanding of real customer needs and priorities to allow informed engineering decisions that preserved value and maintained cost objectives.

Eight articles in this issue cover the design of the HP PaintJet Color Graphics Printer. The development of the color print cartridge is discussed in the article on page 6, and the overall printer design is treated in the article on page 16. The second-generation TIJ structure is described beginning on page 28. Pen manufacturing, filling, and packaging are discussed in the articles on pages 32 and 41. Mechanical and electrical designs are the subjects of the articles on pages 21 and 51. Details of the ink, paper, and transparency film designs are presented beginning on page 45.

In the future, we can expect more products in the tradition of the PaintJet and DeskJet printers, further extending the bounds of printing capabilities, quality, throughput, cost, and user friendliness.

Walter T. Haswell, III
Operations Manager
Corvallis Inkjet Components Operation

Allen D. Johnson
Research and Development Manager
San Diego Division

Neal J. Martini
Operations Manager
San Diego Inkjet Technology Center

Also In this Issue

The first light-emitting diodes (LEDs) were made of gallium arsenide phosphide. Structural advances, combined with zinc oxide or nitrogen doping, improved the light-generating efficiency significantly. More recently, further improvements have been obtained with the aluminum gallium arsenide (AlGaAs) materials system. As the paper on page 84 explains, one type of AlGaAs diode, the double heterostructure absorbing substrate type, is more than twice as bright as previous LEDs, but its complicated structure makes it much harder to manufacture. As a result of a new HP-developed manufacturing technology, these high-intensity red LEDs can now be produced in high volume, and we may soon see LEDs taking over from light bulbs in automotive tail lights, airport markers, and traffic signals.

For many years, the Hewlett-Packard Representation Language, HP-RL, was the language for artificial intelligence research at HP Laboratories. It provided tools for the study of knowledge representation and reasoning techniques, and a variety of experimental expert systems were constructed using it. Now, there are commercially available languages that offer similar capabilities, and work on HP-RL has come to an end. In the paper on page 57, Steven Rosenberg presents a retrospective look at this research effort.

One of the expert systems originally written in HP-RL is MicroScope, an expert program analysis system, which is described by a group of HP Laboratories researchers in the paper on page 71. The current MicroScope prototype analyzes and monitors programs written in Common Lisp, a language widely used in artificial intelligence applications, but the techniques it uses are applicable to any language. Today's large, complex computer programs are so difficult to understand that maintaining them using traditional methods is inefficient, time-consuming, and error-prone. MicroScope is an attempt to give programmers a powerful new set of tools for dealing with the problems of software maintenance.

-R. P. Dolan

Cover

The inset shows print cartridges for the HP PaintJet Color Graphics Printer. The background scene shows a photograph that has been scanned electronically and reproduced by the PaintJet printer (photo by Donald J. Palmer).

What's Ahead

In the October issue, we'll continue our coverage of HP thermal inkjet printers with articles on the design of the DeskJet printer, a personal printer that produces laser-quality text on plain office papers. Another group of articles will delve into the HP-UX 6.0 operating system, which allows several discless workstations to share a central disc. We'll also have the design story of the optical shaft encoder used in the DeskJet printer, which is also available as a separate product.

Design and Development of a Color Thermal Inkjet Print Cartridge

The printhead has to tolerate bubbles, nucleation defects, and localized ink property changes. It must also have long print quality life and be manufacturable in high volume. Testing to verify reliability and manufacturability covered thousands of pens.

by Jeffrey P. Baker, David A. Johnson, Vyomesh Joshi, and Stephen J. Nigro

THE USE OF COLOR MONITORS in business and engineering applications of personal computers and engineering workstations is growing rapidly. Software application packages are available to create color text, graphics, and images on computer displays, often for the ultimate purpose of producing color hard-copy versions of the computer screens for various office and technical applications.

The selection of a technology to generate a color hard copy hinges on the user's needs. The major customer requirements are print quality, speed, available colors, low cost, reliable hardware, multiple media, and excellent software support. Thermal inkjet printing has several advantages compared to other technologies to meet these needs in a cost-effective way. This technology uses heat energy to vaporize a small amount of ink to expel a drop through an orifice. The first product implementation of this concept was the monochrome HP ThinkJet printer introduced in 1984.[1]

The HP Paintjet Color Graphics Printer (Fig. 1) uses thermal inkjet technology to produce vivid color graphics for overhead transparencies and reports. Intended for personal computer applications, it produces text and graphics with 180-dots-per-inch resolution and near-letter-quality (NLQ) text at a speed of 167 characters per second. It can produce a typical page of text in about 35 seconds, merge text and graphics (see Fig. 2), and print a full page of color graphics in about four minutes (Fig. 3).

The PaintJet printer uses two disposable print cartridges, which contain all of the elements necessary to produce a dot on the media. The black printhead has thirty nozzles and the color printhead has cyan, magenta, and yellow inks with ten nozzles per color. The printheads have built-in liquid ink supplies and drop propulsion systems. This technology works on the principle of drop on demand. Each nozzle can supply a drop of ink on demand from the printer as the printhead scans across the media. The drops are ejected by electrical heating of a thin-film resistor to vaporizes a small amount of ink.

The PaintJet print cartridge has the following specifications:

- Resolution: 180 dots per inch
- Drop volume: 100 picoliters
- Drop velocity: 12 meters per second
- Maximum drop rate: 3 kHz
- Energy per drop: 17 microjoules
- Typical drive voltage: 11.5 volts
- Typical pulse width: 5 microseconds.

Thermal Inkjet Advantages

This technology has several key advantages in meeting users' needs for a color printer.

Print Quality. Because this technology uses conventional photolithography and thin-film processing to fabricate the printhead, higher resolutions can be achieved. This con-

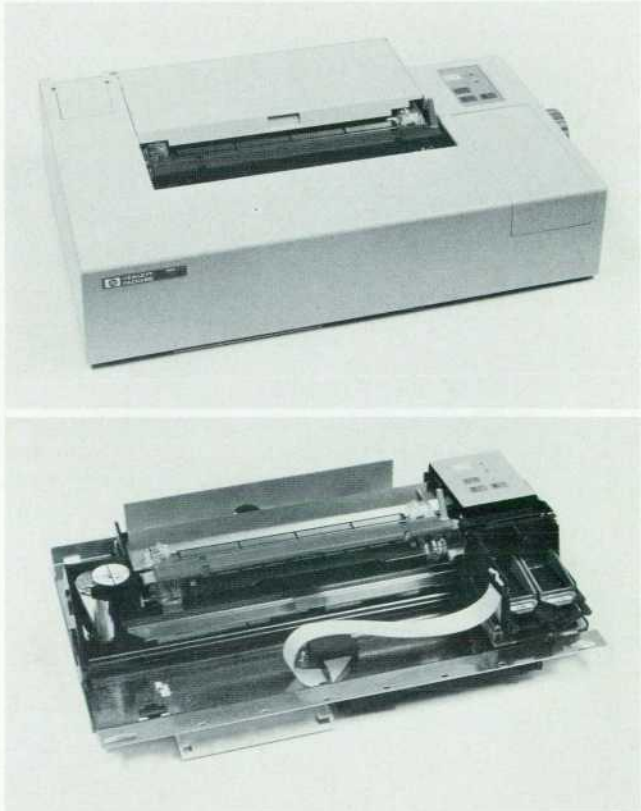


Fig. 1. *The HP PaintJet color-graphics printer combines color graphics with near-letter-quality text. It is designed to serve as a peripheral for personal computers.*

cept also helps tremendously in designing the color printhead with its multiple chambers. Remarkable color-to-color alignment is achieved, resulting in high-quality color graphics and images.

Reliability. Before disposable thermal inkjet technology, inkjet printing had always been plagued by nozzle clogs and other reliability issues. The disposable concept made it possible to develop an extremely reliable system and move away from pumps and hoses. However, to maintain an acceptable cost per color hard-copy page, the disposable printhead must be inexpensive. It must also be producible in high volume, and this requires a mature and stable manufacturing process to keep yields up and fabrication costs down. Consequently, the process engineers have had to understand fully all of the details of production and perfect the technology. The result is a cost-effective and reliable printing system solution.

Speed. The printhead architecture and fluid geometrics are designed so the printer can operate at high speeds (167 characters per second) while producing exceptional text and color graphics.

Multiple Media. Customers want to produce the hard copies on both paper and overhead transparencies. Thermal inkjet technology can print on both types of media with the same printhead.

The remainder of this article deals with three aspects of the development of the print cartridge: first, the design of the drop generator portion of the cartridge, second, the design of the ink reservoir, and third, the testing of the combined system to ensure reliable operation.

Drop Generator Design

The architecture of the nozzle/heater region is shown in Fig. 4. This design is the result of trade-offs in manufacturablity and performance issues.

A design goal for the PaintJet printer was to bridge the gap between plotters and printers. To accomplish this, the product needed to have eight primary colors with greater than a thousand dithered colors, resolution greater than 150 dots per inch, near-letter-quality text, and text speed greater than 150 characters per second. Early in the printhead development, the trade-offs among resolution, speed, reliability, and dot quality were made to meet these goals.

Drop Volume Selection

A key decision was the selection of drop volume, since this influences the media and printhead designs. Printheads with drop volumes above 140 pl did not refill quickly enough to meet the goal of 150 characters per second. A printhead with drop volume below 60 pl was deemed difficult to manufacture.

Image resolution and drop volume were traded off to satisfy media ink capacity (see article, page 45). A drop volume of 100 pl at 180 dots per inch fell within the range of acceptable ink volume per unit area for the media set, appeared to be manufacturable, and met print quality goals.

Reliable Drop Ejection

Spray, or misdirected ink, was a concern. A nominal drop velocity of 12 meters per second results in a drop with a long thin tail in flight (see Fig. 5). This tail breaks up into very small, slow-moving drops of ink which sometimes land outside of the the main spot of ink on the paper. We found that the spray did not degrade print quality, and that the high-velocity mode eliminated other problems (discussed later). Therefore, we chose the high-velocity mode.

Consistent misplacement of dots is a cause of banding. During solid or shaded area fill, this is manifested as light stripes at the spatial frequency of the paper swath advance.

Since ink drop directionality contributes to banding, it has to be controlled. There are two sources of directionality problems in an inkjet printhead. The first is manufacturing tolerances. If the orifice does not line up with the heater resistor, the drop's flight path is not normal to the orifice plate. Since the manufacturing process must have tolerances on nozzle-to-resistor alignment, it was important to minimize the sensitivity of drop directionality to alignment. The key parameter affecting this sensitivity is the ratio of orifice diameter to heater size. The smaller this ratio the less sensitive directionality is to orifice misalignment. The Paintjet printhead has a ratio of 0.67. Fig. 6 shows the resulting relationship between drop angle and misalignment.

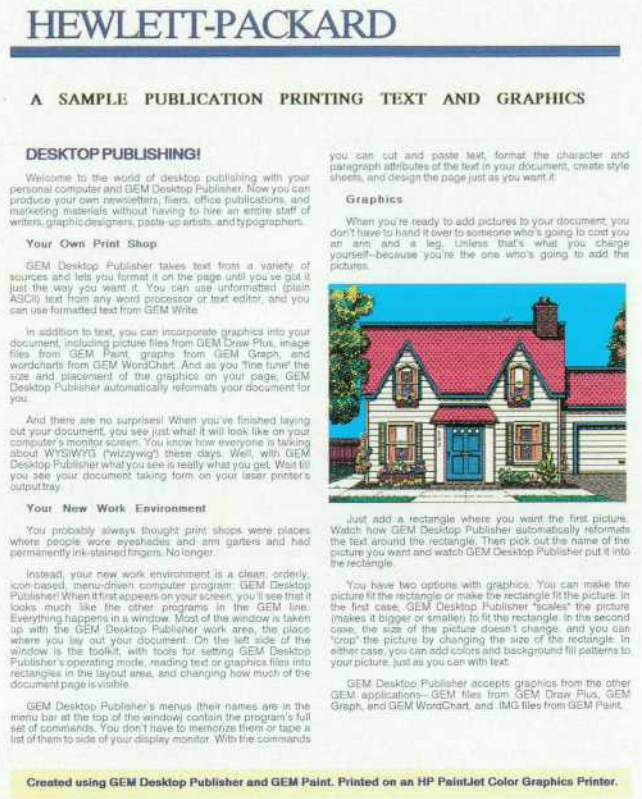
The second source of misdirection is the deflection of an ink droplet by a puddle of ink in contact with the orifice. A drop of ink can be deflected several degrees by the surface tension of the puddle acting on the drop during ejection. The presence of this puddle is determined by exterior nozzle plate wetting characteristics, drop breakoff dynamics, ink meniscus refill dynamics, and frequency of operation. During nozzle refill (after drop ejection), the meniscus of the ink in the nozzle tends to overshoot its equilibrium position, and can spill over the lip of the nozzle onto the exterior of the nozzle plate. The drop ejection mechanism can also contribute ink to the puddle. In low-frequency operation this puddle does not remain, since it contacts the nozzle and has sufficient time to be drawn back into the nozzle by surface tension before the next drop is ejected. In high-frequency operation there is not enough time between nozzle firings to allow the ink to be drawn back into the nozzle. It is important to maintain sufficient fluidic damping to keep the ink mensiscus from greatly overshooting the equilibrium position. This minimizes the puddling.

The largest challenge in designing the PaintJet printhead was making it reliable. The design must be tolerant to bubbles, nucleation defects on the resistor surface, and localized ink property changes in the orifices, and must maintain print quality over a long lifetime.

The orifice region of the PaintJet printhead consists of a three-sided fluid barrier with a convergent orifice (see Fig. 7). The three-sided barrier and the convergent orifice each have multiple functions that enhance reliability.

Bubbles

Bubbles in the ink are a continuous source of problems to the operation of the printhead. Depending on the size and location, bubbles have different effects. Large static bubbles can cause the whole printhead to stop by interrupting the ink path from the reservoir to the orifice. A static bubble next to the heater can shut down an orifice or misdirect ink drops until the bubble is purged. In this case the bubble acts as a compliant source, absorbing energy and



HEWLETT-PACKARD

A SAMPLE PUBLICATION PRINTING TEXT AND GRAPHICS

DESKTOP PUBLISHING!

Welcome to the world of desktop publishing with your personal computer and GEM Desktop Publisher. Now you can produce your own newsletters, fliers, office publications, and marketing materials without having to hire an entire staff of writers, graphic designers, paste-up artists, and typographers.

Your Own Print Shop

GEM Desktop Publisher takes text from a variety of sources and lets you format it on the page until you've got it just the way you want it. You can use unformatted (plain ASCII) text from any word processor or text editor, and you can use formatted text from GEM Write.

In addition to text, you can incorporate graphics into your document, including picture files from GEM Draw Plus, image files from GEM Paint, graphs from GEM Graph, and wordcharts from GEM WordChart. And as you "fine tune" the size and placement of the graphics on your pages, GEM Desktop Publisher automatically reformats your document for you.

And there are no surprises! When you've finished laying out your document, you see just what it will look like on your computer's monitor screen. You know how everyone is talking about WYSIWYG ("wizzywig") these days. Well, with GEM Desktop Publisher what you see is really what you get. Wait till you see your document taking form on your laser printer's output tray.

Your New Work Environment

You probably always thought print shops were places where people wore eyeshades and arm garters and had permanently ink-stained fingers. No longer.

Instead, your new work environment is a clean, orderly, icon-based, menu-driven computer program: GEM Desktop Publisher! When it first appears on your screen, you'll see that it looks much like the other programs in the GEM line. Everything happens in a window. Most of the window is taken up with the GEM Desktop Publisher work area, the place where you lay out your document. On the left side of the window is the toolkit, with tools for setting GEM Desktop Publisher's operating mode, reading text or graphics files into rectangles in the layout area, and changing how much of the document page is visible.

GEM Desktop Publisher's menus (their names are in the menu bar at the top of the window) contain the program's full set of commands. You don't have to memorize them or tape a list of them to side of your display monitor. With the commands you can cut and paste text, format the character and paragraph attributes of the text in your document, create style sheets, and design the page just as you want it.

Graphics

When you're ready to add pictures to your document, you don't have to hand it over to someone who's going to cost you an arm and a leg. Unless that's what you charge yourself—because you're the one who's going to add the pictures.

Just add a rectangle where you want the first picture. Watch how GEM Desktop Publisher automatically reformats the text around the rectangle. Then pick out the name of the picture you want and watch GEM Desktop Publisher put it into the rectangle.

You have two options with graphics: You can make the picture fit the rectangle or make the rectangle fit the picture. In the first case, GEM Desktop Publisher "scales" the picture (makes it bigger or smaller) to fit the rectangle. In the second case, the size of the picture doesn't change, and you can "crop" the picture by changing the size of the rectangle. In either case, you can add colors and background fill patterns to your picture, just as you can with text.

GEM Desktop Publisher accepts graphics from the other GEM applications—GEM files from GEM Draw Plus, GEM Graph, and GEM WordChart, and .IMG files from GEM Paint.

Created using GEM Desktop Publisher and GEM Paint. Printed on an HP PaintJet Color Graphics Printer.

Fig. 2. *An example of merged text and graphics from the PaintJet printer.*

inhibiting drop ejection. Even bubbles much smaller than the heater change ejected volume by acting as low-temperature nucleation sites if they are near the heater surface at the time of nucleation.

Three-sided barriers allow static bubbles to be purged during priming. During operation, the refilling flow continuously sweeps outgassed and small static bubbles into the firing chamber. Here they are ejected with the drop, preventing accumulation of large static bubbles.

A convergent orifice eliminates the gulping of air into the heater region. After drop ejection the meniscus is retracted deep within the orifice (see Fig. 4). With a convergent orifice the retracted meniscus is nearly spherical, which is the lowest-energy state and therefore the least likely to gulp air. The retracted meniscus of a straight-bore nozzle tends to pinch off and gulp in an air bubble, which shuts down any further drop ejection.

Cross Talk

Hydraulic cross talk between adjacent nozzles can cause banding when multiple nozzles are firing by changing the volume of ink ejected from nozzles. Localized high pressure generated during nucleation creates a pressure gradient between active and inactive orifices, which causes perturbations in the mensicus position of the inactive orifice. If this nozzle is now fired, its drop volume will be different from normal. In previous thermal inkjet devices a compliant source in the form of a slot was used to reduce cross talk. In the PaintJet printhead, hydraulic cross talk is minimized by increasing the impedance between nozzles by means of the three-sided fluid barrier, and by placing the ink feedslot close to the nozzles (see Fig. 8). The feedslot acts as a constant-pressure source or a large capacitor to absorb most of the fluid backflow from the active orifices.

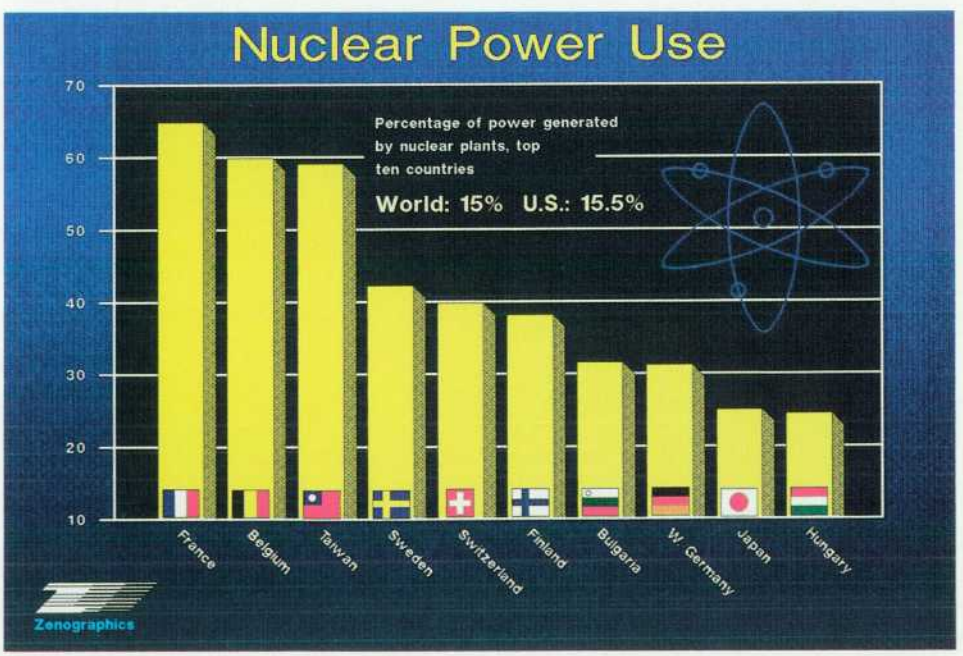


Fig. 3. *An example of color graphics printing from the PaintJet printer. A full page takes about four minutes to print.*

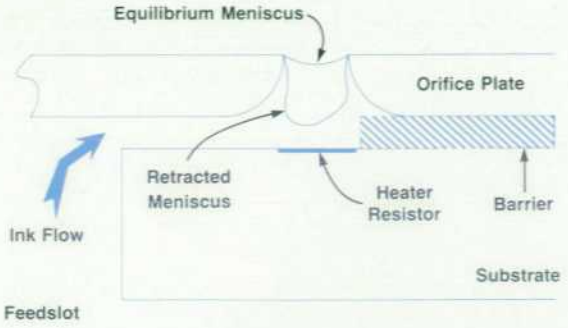


Fig. 4. *Cross-sectional view of the PaintJet printhead nozzle region showing the ink path, fluid barriers, heating resistor, and nozzle shape.*

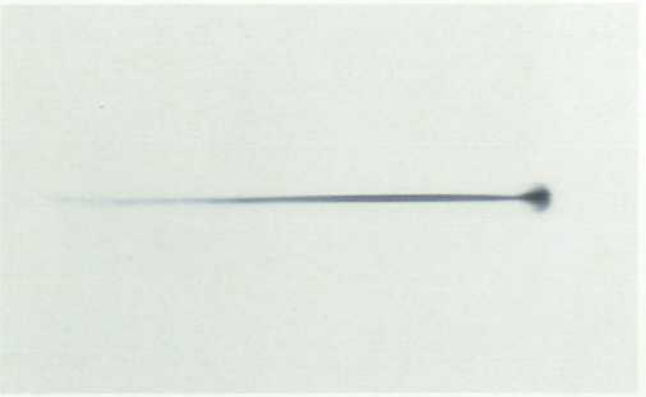


Fig. 5. *Photograph of drop ejection. The length of the tail depends on the ejection velocity.*

As mentioned earlier, a high-velocity mode of operation was chosen for the PaintJet printhead. When a nozzle is not active, water evaporates from this region. This causes a local increase in dye and diethylene glycol concentration, which results in higher viscosity. The high-velocity drop mode is able to clear these viscous plugs, eliminating the need for capping. Low-velocity drop ejection modes do not have sufficient power to overcome the effect of high-viscosity ink, and the nozzle will not eject drops after several minutes of inactivity. Fig. 9 is a plot of drop velocity versus drop number. This data is typical for a pen that hasn't operated for several minutes. The first drops out are low-velocity, but velocity quickly rises to normal as the viscous fluid is cleared. In the PaintJet printer, these slow drops are cleared from the nozzle before printing begins by ejecting the drops into an absorptive "spittoon" near the edge of the paper.

High-Frequency Operation

From the beginning of the PaintJet printer project it was obvious that the maximum carriage speed of the product would be determined by the maximum drop rate of the printhead. One of the main limitations to the speed of printing is the frequency of operation of individual nozzles. The faster individual nozzles can eject drops, the faster the printer can print.

Nozzle speed limitations can be thermal (heat buildup in the printhead) or fluidic (how fast a nozzle will refill). After initial experimentation, it appeared that the problem of nozzle refill dominated, and that the maximum frequency of operation of a PaintJet printhead would be about 3000 drops per second per nozzle.

After a drop is ejected from a nozzle, it is possible to eject another drop as soon as ink again covers the resistor. However, the second drop may be much smaller and misdirected. Since all drops should look the same, this is unacceptable. It was decided that a second drop should not be ejected until the ink meniscus is near the rest position. The refill dynamics behave like a nonlinear damped second-order system, so the meniscus position goes through a series of overshoots and undershoots (see Fig. 10). It is not practical to try to eject exactly when the meniscus crosses the rest position, so we wait until the meniscus is sufficiently damped, so meniscus position variation produces only small variations in drop volume. This is any

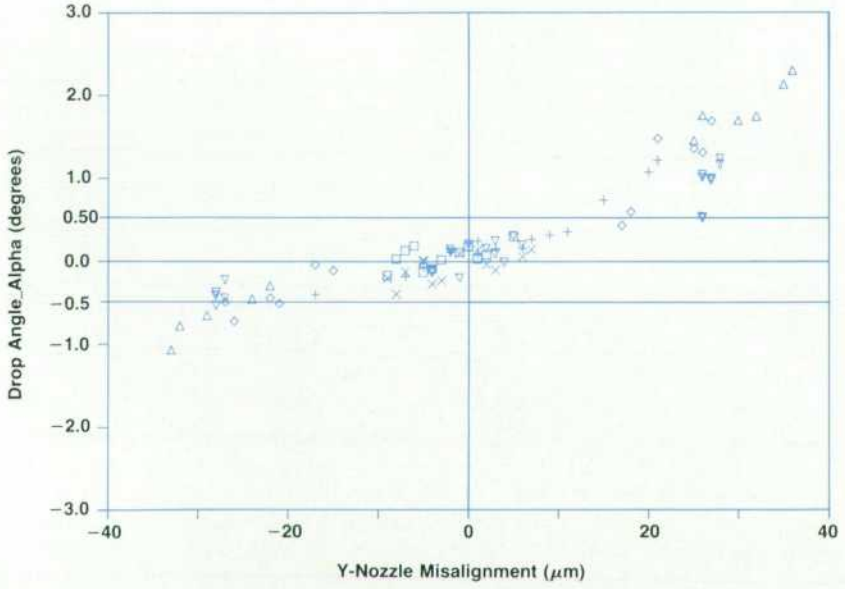


Fig. 6. *Directionality data for several printheads, showing drop ejection angle as a function of misalignment between the nozzle and the heater resistor.*

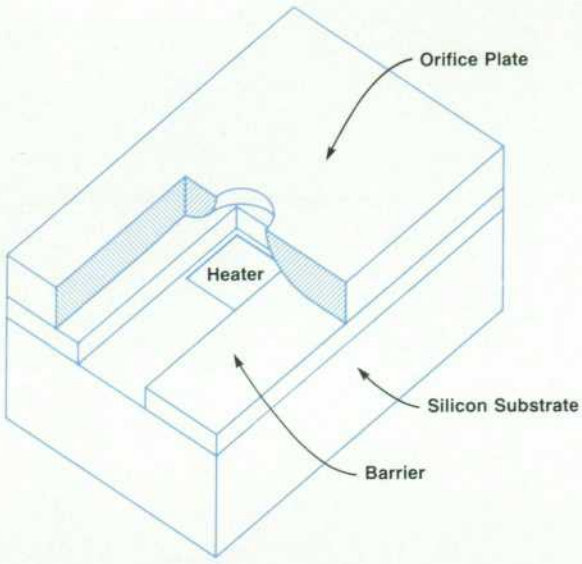


Fig. 7. *Cutaway view of the PaintJet printhead orifice region, showing the three-sided fluid barrier.*

time after the second crossing.

As ink is used up in the reservior, the negative pressure (suction) increases. This causes the nozzle to refill more slowly and moves the meniscus rest position slighly deeper into the nozzle. Fig. 11 shows the effect of negative pressure on the peak time, or time to the highest point of the first overshoot. The nominal drop volume decreases slightly as a result of the change in meniscus rest position.

Many factors affect the shape of the refill curve. The two major factors, ink properties and nozzle geometries, were also subject to severe manufacturing and performance constraints, leaving little room for refining the design to increase frequency response. Because of this, the refill characteristics were tracked during development to ensure that changes to the pen design for other reasons did not result in slow refill.

As the design progressed and manufacturing issues caused larger than expected fluid drag terms, the refill slowed enough to cause the drop ejection to be on top of the first overshoot in some situations. To compensate for this, it was decided to increase the barrier height, which decreases the fluid drag in the channel, resulting in faster refill.

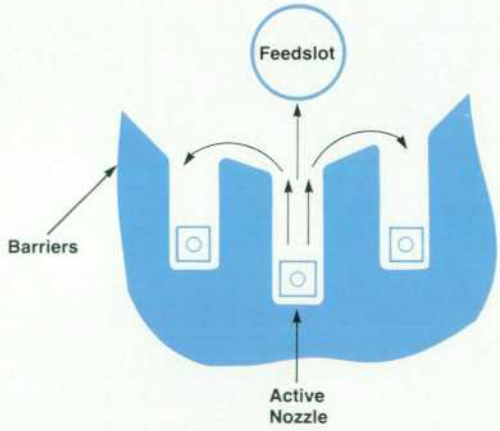


Fig. 8. *Diagram showing three nozzles, the ink feedslot, and the fluid barriers. Hydraulic cross talk consists of ink flow back down the barriers to the feedslot and to neighboring nozzles.*

Reservoir Design

The primary requirement of the reservoir system is to deliver ink to the thermal inkjet printhead at the optimal head operating pressure. This pressure is below atmospheric pressure to prevent leaking from the nozzles. The drop generator (heater, barrier, and nozzle) is designed to eject the correct volume of ink to match the dot resolution and the media (paper or transparency) in use. This local geometry and the ink surface tension determine the pulling force of the drop generator refilling process. If the reservoir resisting force is too large, the drop generator will refill too slowly and will not be ready to eject the next drop. Therefore, the design of the reservoir system dictates that the ink should be contained to prevent free flow but not slow down the drop generator refill process drastically.

In addition to the ink delivery pressure, there are numerous other design goals for the reservoir system. These include some musts—items that cannot be compromised—and wants—items that can be considered trade-offs for yield and cost considerations. Among the must items are that materials be compatible and that the reservoir be inexpensive, robust, clean, manufacturable, and able to supply bubble-free ink to the printhead under all operating conditions. Desirable but negotiable goals for the reservoir system include that it be lightweight, space efficient, and usable in a three-color pen.

Initial Pen Prototypes

Basically, three different methods were examined: a gravity system, mechanical pressure regulation, and capillary ink containment.

The gravity approach, which stores the ink below the

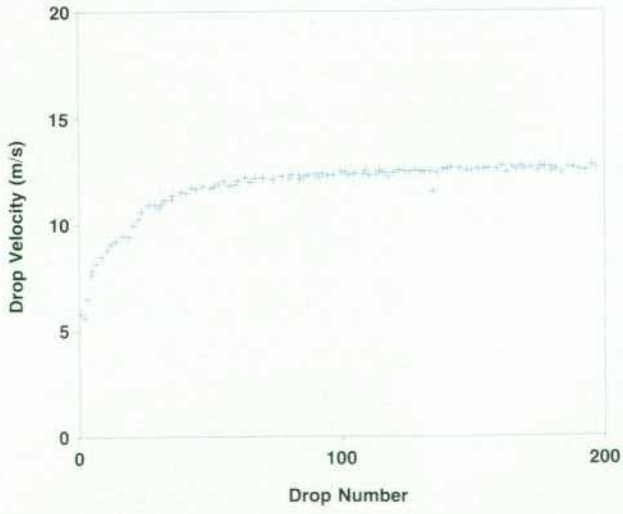


Fig. 9. *Startup plot for a pen that has been inactive for several minutes. Drop velocity increases as high-viscosity ink is purged from the nozzle. In the PaintJet printer, these slow drops are purged before printing begins.*

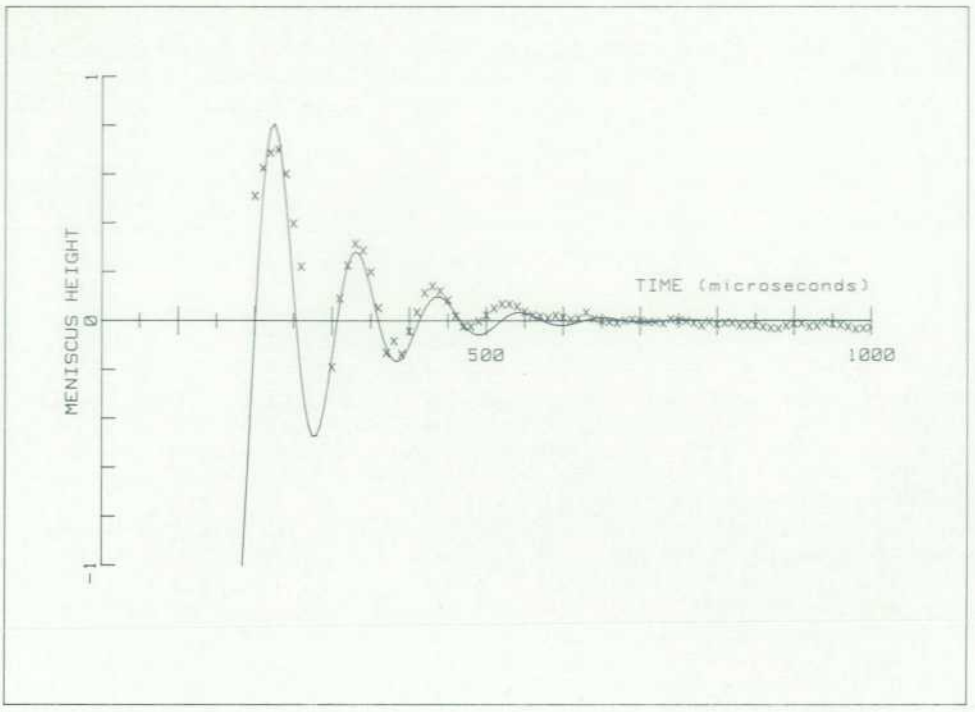


Fig. 10. *Oscillations of the meniscus position during refill. The* xs *are measured data and the solid line is a curve fitted to the data.*

nozzle position and "hangs" the ink from the orifice, is difficult to implement. The mechanical pressure regulator, a collapsible bladder like the ThinkJet printer's, contained too little ink (scaling up this design was risky). Also, if two or three of these bladders were placed inside a single small pen to make a multicolor pen, it would be difficult to avoid interaction between the bladders.

Most of the liquid-ink pens on the market today use a capillary ink containment system. Fiber-tipped and roller-ball pens hold the ink supply within a capillary matrix or tube. The reason for this is simple: fibers are cheap and capillary forces are reliable and naturally balanced. The reservoir can have any shape desired; only a continuous path is needed between the supply and the pen tip. If the body of a capillary pen breaks, the ink remains contained within the matrix and will not run out freely.

Capillary Reservoir Thermal Inkjet Pens

Our first try at using a capillary reservoir on thermal inkjet pens employed standard felt pen fibers. The pen nib has very small ink channels and therefore a strong ink pulling force. To oppose this force and to provide design margin against leakage, the reservoir of the felt-tipped pen has a capillary pressure of approximately two feet of water. That is, the bundle of fibers can pull water up two feet above its surface by capillary force. The corresponding pressure created by the meniscus at the orifice in our thermal inkjet head is approximately 9 inches of water. The felt pen type reservoir was just too powerful and emptied the drop generator region as soon as it was installed.

Glass beads can provide a capillary force because of the interstitial space between the spheres. The advantage of beads is that they come in a variety of sizes, so the force can be adjusted accurately and easily. Inkjet pens made with a glass bead reservoir worked well and showed that a capillary system could be used for thermal inkjet technology. They did not, however, meet all of the design goals. Weight, volumetric inefficiency, and material concerns convinced us that further exploration was desired.

Thicker fibers would reduce the capillary force, but they are nonstandard and therefore expensive, and their efficiency is marginal (around 40%). Other methods were explored such as rolled-up plastic film or long plastic tubing, but none met all of the requirements. The thin textile fibers would be a perfect design if a method could be found to maintain a 6-fiber-diameter spacing between the strands to give a proper capillary pressure with high volumetric efficiency. Microphotographs showed that open-celled polyurethane foam looked like the solution that we had been searching for (see Fig. 12).

Ether-Type Polyurethane Foam

Polyurethane resin is made using two different base molecules: ester and ether. The ester has the advantage of better process control during foaming, but is not very stable in a water-based-ink environment. Ether-based polyurethane is more resistant to hydrolysis but does not have the process control. The ether foam is made only one way—approximately 70 PPI (pores per inch). This pore count is too low for the thermal inkjet pen reservoir, so the foam is crushed under heat (felting) to compress the matrix. This technique can give us any capillary force required. It also allows testing of head operation at various back pressures.

When the foam is "blown," the air spaces are created by large quantities of gas generated during setting or resin cure. After the foam is cooled and finished, many of these gas bubbles still exist inside the matrix. Thin walls of polyurethane between the fibers still contain the foaming process gas. This would cause no concern except that the inkjet pen requires ink mobility, or the ability of the ink

Capillary Forces in a Foam Matrix

Capillary force is classically demonstrated by placing a thin glass tube into a pond of water and observing the water level rise inside the tube. This phenomenon occurs because the water molecules are attracted to the glass tube wall more than they are attracted to other water molecules. As the water wets the glass and spreads up the wall, it pulls the fluid up. The height reached by the column is related to the surface tension of the fluid (a measure of the water molecule's attraction to itself), the wetting angle on the tube wall (a measure of the attraction of the water to the glass), and the diameter of the tube (pulling circumference), and is inversely related to fluid weight (load). Modeling the capillary pressure is simple for a round tube but becomes difficult when using beads, fibers, or foam.

To model the more complex capillary reservoirs, a few partially scientific assumptions can be made about the energy requirements for raising the fluid. If a capillary reservoir is injected with a fluid, there is a relationship between the additional reservoir surface wetted and the volume of ink injected. In other words, the energy required to raise the fluid is balanced by the energy available from wetting the surface. For complex capillary systems with small features, that is,

$$\text{Characteristic distance} < \sqrt{\sigma/\rho g}\ ,$$

where σ is the surface tension of the fluid in air, ρ is the density of the liquid, and g is the acceleration of gravity, the effective capillary pressure can be estimated by:

$$P_{cap} = \sigma \frac{\text{Incremental wetted surface}}{\text{Fluid volume required to wet this surface}} \cos\theta$$

where θ is the equilibrium wetting angle.

To estimate the capillary pressure of the foam matrix, the density of the basic resin is determined, and by measuring the density of the foam, the void ratio is calculated. Using the approximate pore size of the foam (inverse of the pores-per-inch count), the capillary pressure is estimated by:

$$P_{cap} = \sigma \frac{\sqrt{(12\,\pi(1 - \text{Void ratio})}}{\text{Pore size} \times \text{Void ratio}}$$

This model assumes that the number of fibers wetted is large and the angle at which they meet the fluid is constant as the fluid level changes. Also, within the foam matrix, the pressure is governed locally by the smallest feature because of the equalization flow from a larger feature to a smaller feature higher in the matrix. That is, if a small empty capillary cell is above (higher than) a full larger cell, the fluid may wick into the smaller higher position.

The amount of ink delivery required governs the size of the reservoir system. For all orientations of the pen, the reservoir has to hold the ink up, not allowing leakage or vent coverage. If the operation of the head requires a very narrow range of supply pressure, the size of the reservoir must be kept small and the relative sizes of the capillary cells kept constant throughout the matrix . If not, the delivery pressure will vary considerably over the life of the pen as the fluid level changes.

When the reservoir material has been selected, the total volume of available ink can be calculated by multiplying the difference between the total volume and the capillary material volume by the efficiency of the system. The efficiency is determined by the amount of fluid left behind on the capillary surfaces. A reservoir with many microfeatures will retain a greater amount of ink than a system with smooth capillary surfaces.

to travel easily from one portion of the reservoir to another (from the middle of the reservoir to the filter area). Therefore, these walls of polyurethane resin must be removed, leaving the fibers behind. This process, called reticulation, consists of filling the foam with hydrogen and burning out the walls from one end of the loaf to the other. This works well except that a great quantity of residue consisting of wall portions and burnt resin is left behind. This requires the foam to be cleaned, a difficult process because the contamination is inside the matrix and cannot be flushed out quickly.

Preparation of the Foam Reservoir

Freon™ is used as a solvent to remove contaminants from the foam cubes. The Freon is regenerated in a distillation column that is part of the cleaning tool, thus providing a closed system with minimal solvent loss. This system provides a safe and healthful work area, minimal environmental emissions, and low solvent replacement costs.

Two tests are used to determine the cleanliness of the foam. One test involves extracting the residue remaining in the clean foam and expressing this as a weight percentage of nonvolatile residue. This verifies that the cleaning was effective and that the majority of the contaminants have been removed. It also provides a sample for chemical analysis if more information on the specific nature of the contaminants is desired. The second test involves exposing the clean foam to ink vehicle (ink without dye) and measuring the decrease in the surface tension of the ink vehicle. This test indirectly monitors the concentration of any surfactant-type contaminants that could lower the surface tension of the ink in the pen. Lowered surface tension can cause the ink to leak out of the pen or affect print quality.

Testing Program

The key to the success of the PaintJet printhead is thorough testing of the design to ensure supreme print quality and reliability. Major emphasis was placed on developing a test strategy to address these issues. The product specifications derived from user needs and a use model were the starting points for defining realistic and concrete goals for reliability and print quality. At each phase of the development, test plans were generated to characterize the technology to determine how well the goals were achieved in terms of confidence levels (50% confidence at laboratory phase and 90% confidence at introduction). This goal setting process allowed us to look at the development cycle with customers in mind.

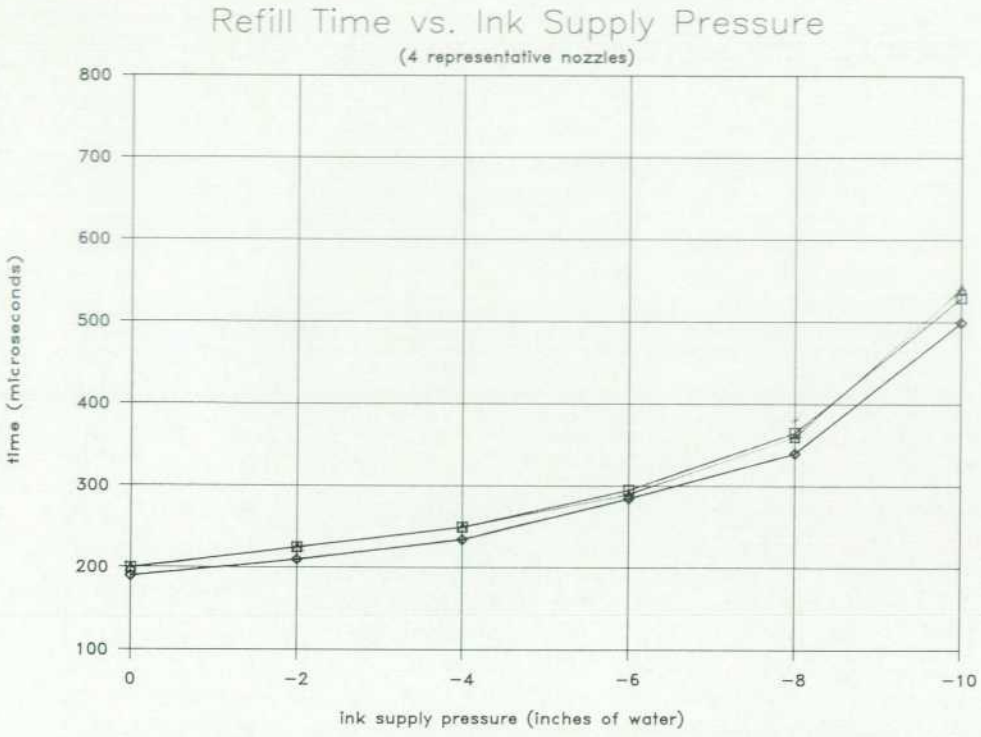


Fig. 11. *Effect of negative head on refill time.*

Print Quality Testing

Print quality is a very subjective but extremely important specification for printers. Print quality is influenced by inks, media, printing mechanism, and customers. A process was developed to set the design specifications and tolerances for dot size, dot location, color range, and other attributes that determine print quality so that inks, media, and a printing mechanism could be developed to meet user needs for their applications.

There were two major efforts for print quality evaluation during the design phase, one in engineering and one in marketing. The marketing group identified the key market segments, applications, and competition for the PaintJet printer. The engineering group developed a list of approximately thirty measurable print quality attributes and a set of diagnostic print samples to simulate errors (see box, page 14). The two groups also collected application specific print samples that represent actual customer needs. The printing technology was characterized to understand and determine the bounds for the critical attributes and a print quality survey was designed to derive design centers.

Samples were collected for three major application segments: word processing, merged text and graphics, and presentations. The word processing sample was generated in black only to evaluate text quality, while merged text and graphics and presentation samples were printed in full color. For each sample some specific print quality attribute was tested to determine the design center and associated tolerances on the basis of the acceptability of the print quality for that application. The survey also included samples generated on other similar products on the market.

Reliability Testing

The disposable printhead concept makes inkjet printing capable of high reliability. The PaintJet printer reliability testing program has been a major effort to ensure that its performance satisfies both business and engineering customers. The marketing group worked with the engineering team to set realistic and concrete reliability goals based on user needs and the use model. The printer will be used in an office environment (15 to 32.5°C and 20 to 80% relative humidity) and will have to perform with minimal customer attendance. The print cartridge can be shipped anywhere in the world without any shipping or storage restrictions

Fig. 12. *Unreticuated foam.*

Print Quality and Pen Development

Print quality characterization played a crucial role in the development of the HP PaintJet printer. This effort began with attempts to understand print quality. A vision system was assembled to characterize spot size and shape on paper and study media/ink interactions. This tool was vital to early understanding of drop characteristics and media characterization.

Later machine vision tools looked at plotter errors (paper advance errors, servo errors, carriage alignment errors) and pen errors (dot size, dot placement, and satellites), as shown in Fig. 1. Although the systems were ultimately too slow for assembly line inspection, they were used to characterize pen and plotter print quality performance.

From the data gained from these and other instuments, we created a customer survey to establish customer sensitivity to error types. Using selected pens and a special PaintJet printer with programmable dot placement, a number of error simulations were produced, showing banding, spot size variations, pen-to-pen alignment variations, satellites, and other errors. The results of the survey were vital in establishing pen and product performance specifications. The customer survey established a print quality inspection criterion for the pen production line, where performance is judged on pen functionality, consistent area fill, and stable line weight.

Third-Party Testing

Because of the risk and uncertainty inherent in the development of a new technology, a vigorous strategy of third-party testing of the pen and product design began nearly three years before product introduction, and continues today. The goal is to identify and observe operational problems and print quality changes that may occur for a typical customer under specified environmental extremes.

PaintJet printers are subjected to continuous testing in various environmental conditions. The tests identify paper dust, user interface effectiveness (priming, wiping, pen insertion success rate, deprimes during operation, start-up deprimes, interconnect reliability), print quality degradation, materials integrity, and MPBF (mean pages between failures). Besides normal print quality testing, some pens go through storage and shipping tests to make sure that the pens can survive such conditions.

Because failure mode is very often dependent on user mode, marketing feedback is used to make sure that the testing represents what a typical customer may encounter when using the printer and printhead. Each failure that occurs is analyzed to identify the cause. The results of the testing are translated into MPBF so that the relative severity of each problem can be assessed. The engineers who own each problem investigate the fundamental cause of failure and propose possible solutions. Very often, a failure is caused by interaction of the printer and printhead, requiring changes to both systems.

2500 pens have been tested. As a result, five months after product introduction, there have been no returns from customers because of pen failure.

Third-party testing is now used for qualifying any changes in processes to improve pen performance. The strategy will continue on to the next-generation inkjet printhead development.

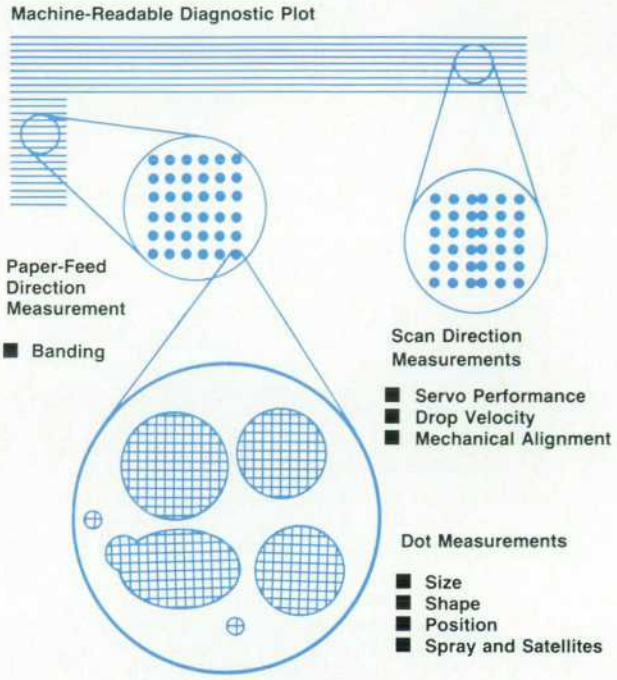


Fig. 1. *Quantitative print quality measurements used in development and production of the HP PaintJet printer.*

Audit Testing

Audit testing consists of a series of standardized tests to verify the pen performance on a daily basis. The tests are used both to guarantee the quality of pens produced and to qualify design and process changes. Most of the tests and equipment were developed in the early R&D phase to verify printhead design and performance. Later, the tests and equipment became quality assurance monitors for printhead manufacturing.

Some of these tests are: drop velocity, drop volume, operating energy, print test, life test, altitude test, and simulated storage test. Each test has a documented operating procedure and expected results. These procedures form a library, which is referenced by the daily audit test plan.

With the audit tests written in this modular form, the daily testing can be adapted to check more rigorously for suspect failure modes after various tooling changes. The test library also allows engineers working on process or design changes to use standard tests and compare their results against historical results.

Dan Beamer
Mike Borer
May Fong Ho
Don Bergstedt
Development Engineers
San Diego Division

and should have at least two years shelf life. The print cartridge should print a minimum of a million characters.

These requirements drove the reliability goals for three major testing categories: shipping and storage, shelf life, and system operational testing.

Shipping and Storage. The goal here was to be able to ship the printhead anywhere in the world without any restrictions. When customers receive these printheads, they are assured of 99% reliability with 90% confidence that the printer will operate after the activation process (priming and wiping). At each development phase the reliability was tested against the goal by simulating shipping and storage conditions for temperature (high and low), shock, vibration, and altitude.

Shelf Life. The 30-month shelf life specification is based on the use model and the deliverable ink volume for the printhead. Requirements are that the materials used for the printhead (ink, foam, orifice plate, filter, pen body, etc.) be compatible and that no major corrosion or performance problems surface during that time. Printheads are stored at 65°C for seven weeks and then tested for proper system operation. During development, other printheads were stored and periodically tested to confirm the validity of the accelerated tests.

System Operational Testing

Three major categories of system failures can be observed. A soft failure is a failure that fixes by itself during printing. A nozzle that is turned off for some reason and comes back on after some time is an example of this kind of failure. A print cartridge can print 600 pages, with a typical page having 1500 characters on it. Our goal for soft failures was for mean pages between failures (MPBF) to exceed 600 pages.

Medium failures are failures that need customer intervention by priming and wiping the printhead. Our goal for medium failures was also set at an MPBF of 600 pages. Depriming, nozzle clogs, and the like are generally medium failures.

Hard failures are failures that require printhead replacement. Resistor failure, out of ink, and the like are examples of hard failure. Two goals were set for hard failures. First, one million characters can be printed before running out of ink. Second, there should be 99% probability with 90% confidence that the printhead will run out of ink before any other hard failure will occur.

To meet the goal of 99% reliability with 90% confidence, thousands of pens had to be made, and in fact, a year before introduction, we were building thousands of pens per month for the engineers to characterize designs through the necessary testing. This multithousand-pen production line also gave the process engineers and manufacturing engineers ample opportunity to improve and iterate process designs in parallel with part design to achieve the reliability goals.

All of this led to a very stable, reliable product with a very smooth startup. In fact, at introduction, the PaintJet pen production line was in its third generation. As a result of the initial investment in manufacturability, the black and color pens are built on entirely the same tools on the same production line and all assembly consists of single-axis motions.

It is unfortunate that, since most of the processes that were developed are considered proprietary, only a cursory discussion of those processes is possible in this publication.

Acknowledgments

A program of this size and complexity had many contributors. We would like to acknowledge Dan Allen and George Lynch for directing early efforts in pen design. Jeff Groenke, Fred Beretta, Brian Canfield, and Don La made many contributions to pen design and engineering. Beth Heffernan contributed by developing the foam cleaning process. We would like to thank Chuck Hutchison and his group for supplying print cartridges, a difficult task given that manufacturing processes were also being developed. Also, we would like to thank Nancy Ritzenthaler and Bob Borden for giving marketing direction. John Page and Rick Peterson were the key contributors in the successful reliability testing. We would also like to thank Cheryl Katen's group at HP's Inkjet Components Operation and Howard Taub's group at HP Laboratories for their extensive contributions to head architecture design.

Reference

1. *Hewlett-Packard Journal*, Vol. 36, no. 5, May 1985, entire issue.

Development of a Color Graphics Printer

Full-color graphics, reliability, and software support received high priorities in the development of the HP PaintJet Color Graphics Printer.

by James C. Smith, David C. Tribolet, Hatem E. Mostafa, and Emil Maghakian

AT THE BEGINNING of the HP PaintJet printer project, an assessment was made of our technological capability, our customers' needs, and what our competitors were doing. At that time, the HP San Diego Division's product line and expertise revolved around vector pen plotters. Vector plotters, in general, produce superior-quality line graphics, so they are ideal for high-quality line charts, pie charts, bar charts, text charts, and presentation graphics. On the other hand, vector plotters are less than ideal for text and images. Plot time is roughly proportional to the number of vectors, so plots containing a large number of vectors take a long time to produce.

Raster printers are more versatile than vector machines. They are able to produce text, images, and graphics. The perceived quality of raster output is very much dependent on the resolution (Fig. 1). Output time with raster devices increases with resolution simply because more dots are placed on the writing media at higher resolutions. Also, cost generally escalates with increasing resolution. So balancing quality, throughput, and cost by selecting a resolution becomes a key engineering trade-off in the design of a raster product.

In looking at our customers' needs, it didn't take us long to realize that the main output requirement was for text. Everyone had a printer. Most people didn't want both a printer and a plotter, so they would often make do with just a printer. We saw customers creating graphics using the period of their daisy-wheel printer. It was obvious to us that we had to add printing to our graphics because our competitors would certainly add graphics to their printers.

So we contemplated the ideal output device for our customers. It should be versatile. Our customers have a broad range of output requirements such as reports, memos, graphics, images, and presentations. It should be quiet, a clear advantage in an office environment. Color is increasingly important because of color monitors, new applications software, growing use of color copiers, and color scanners. So color capability was a must. The quality should be high and the cost should not be significantly more than a monochrome printer. In addition, the throughput should be high.

Raster technology was clearly the best fit for a versatile product. But which raster technology? Within HP, significant progress had been made on the evolving thermal inkjet technology. This drop-on-demand technology showed great promise and was chosen for the color raster product that came to be known as the HP PaintJet Color Graphics Printer.

PaintJet Printer Features

The PaintJet printer offers seven colors at 180-dots-per-inch (dpi) resolution: cyan, yellow, magenta, red, green, blue, and black. At 90 pixels per inch the user can select from a palette of 330 colors (Fig. 2). Throughput on color graphics depends on the number of swaths made by the carriage. On a full page of graphics the carriage makes about 200 swaths, which takes four minutes. The PaintJet printer has a transparency mode in which the carriage double-passes the image so that each dot position has two drops of ink to improve color density. Color transparencies take eight minutes, or less for images that occupy less than the full length of the transparency. Burst speed on text is 167 characters per second at ten characters per inch and a typical page of text takes about 35 seconds.

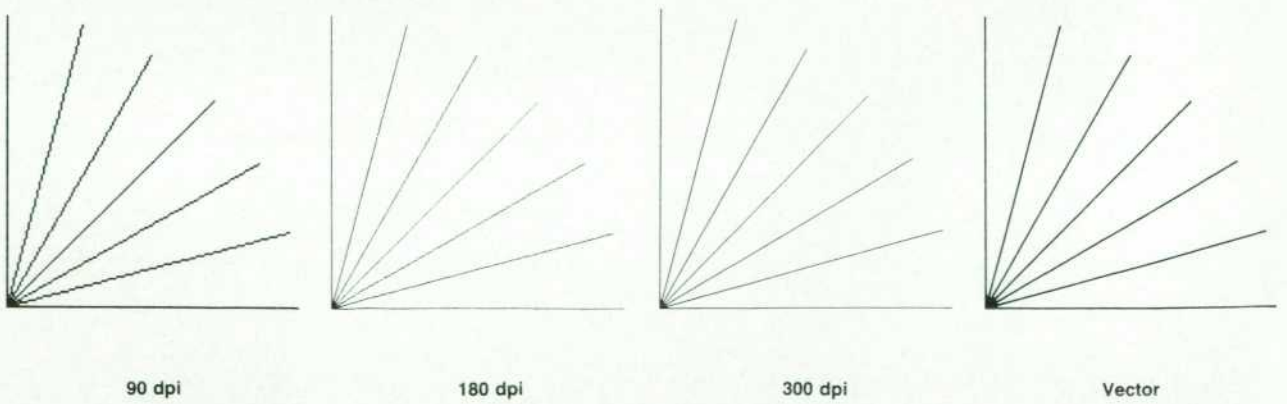


Fig. 1. *Vector plotting makes the smoothest lines. The quality of raster lines depends on the resolution.*

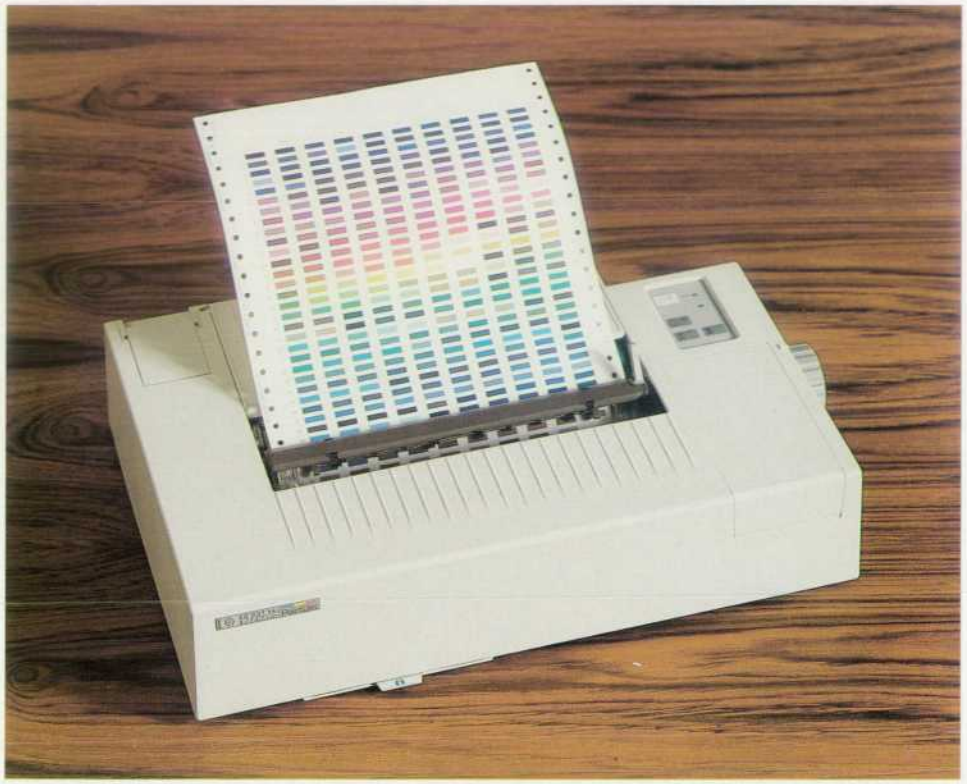


Fig. 2. *The HP PaintJet printer produces color graphics for overhead transparencies and reports and near-letter-quality high-speed text. It offers seven colors at 180 dots per inch or 330 colors at 90 pixels per inch.*

The PaintJet printer is compatible with nearly every leading personal computer. It is supported by a wide variety of software including business graphics, word processing, personal computer CAD, spreadsheet, productivity, and utility software. Available interfaces are parallel, RS-232-C, and HP-IB (IEEE 488/IEC 625).

A clear benefit of the thermal inkjet technology is its inherently quiet operation. Other technologies, such as multiwire impact printing, are very noisy. During the PaintJet printer's development we found that the writing system was so quiet that other noises in the product became noticeable. Noise produced during carriage turnaround and a snapping sound caused by the paper popping off the sprocket pins were some of the areas attacked by the design team. The ultimate result of this effort is a sound pressure level below 50 dBA.

Architecture Overview

The heart of the PaintJet printer's writing system consists of two disposable print cartridges: one black and one color. Each cartridge has a total of 30 nozzles. In the case of the black print cartridge all 30 nozzles are dedicated to black, while for the color cartridge 10 nozzles are dedicated to each of the subtractive primary colors: cyan, yellow, and magenta. The two print cartridges are held side-by-side in the carriage, which is driven by a dc motor as it scans across the paper (Fig. 3). A single-channel linear encoder is used to close the loop around the dc motor. The 90-line-per-inch linear encoder, in concert with a hardware extrapolator in a custom integrated circuit, also provides the edges used to fire the nozzles at 180 dpi.

The paper axis consists of a platen with sprockets mounted at either end. The sprockets have sprocket pins that retract to avoid interfering with the carriage. The carriage rides on a thin metal shim that presses the media against the platen. This arrangement allows the distance between the printhead and the media to be minimized without any adjustments. The paper axis handles Z-fold paper, cut-sheet paper, and transparency film.

The electronics required to control the carriage and paper axis motion and to control and pulse the print cartridges are contained in two highly integrated boards: the main board and the carriage board. The main board includes the power supplies, motor drivers, memory, I/O, microcontroller, and custom IC (see article, page 51). This board provides all motion and dot firing control. The carriage board includes the optics for the single-channel encoder plus two identical custom ICs that translate the digital dot firing control signals into the proper voltages and currents required to fire the print cartridges. These fire pulses are carried from the carriage board via a flex circuit to the beryllium copper interconnect fingers that make electrical contact to the print cartridges. The extensive level of integration and component minimization on these boards was achieved by optimizing the partitioning of the firmware and hardware functions to make full use of the capabilities of each component on the boards.

Recipe for Reliability

Our goal was to produce the most reliable product ever to come out of our Division. We took a two-pronged approach to achieve this lofty objective. First, we devoted much time up-front in designing the product for reliability. Second, we subjected the PaintJet printer to more test hours than any other product we had ever produced.

To design in reliability, we strove to minimize the com-

Color Communication Standard

How can a computer (CPU) communicate color to a peripheral device (e.g., a color printer/plotter, scanner, CRT, film recorder, etc.)? Typically, a color image is input into the CPU from a software program, scanner, data file, or other source. The CPU outputs this image to a peripheral device such as a color monitor or hard-copy printer/plotter. In most cases, the image is first displayed on the monitor and the user modifies the image until the desired appearance is achieved. The image can then be printed with a color printer such as the PaintJet printer.

To match color between devices, it is essential that the color information sent from the CPU be either precorrected to the specific device's characteristics or specified using a standard that can be interpreted and correctly rendered by the device. Presently, each color peripheral has its own way of specifying color, and standardization does not exist. For example, the color displayed on a CRT monitor is fairly arbitrary. The basic requirement is that it represent an approximation of the color that the user desires, since the human eye very easily adapts to the CRT image, compensating most color distortions. This makes it a problem to produce a color-matched hard copy, since the printer has no knowledge of how the user is interpreting the CRT color and the eye will not go through the same kind of adaptation when looking at the hard copy. Furthermore, differences in the color gamut obtainable from each device may limit the gamut of colors that can be used. Regardless of the physical limitations, however, it is accurate to say that without standardizing color communication, it is highly improbable that an accurate color match between peripherals will ever be achieved outside of the laboratory.

In an attempt to improve this limitation, the PaintJet printer has been designed to use a standardized method for color communication. While no industry standard has been fully endorsed as the preferred method for color communication, the PaintJet printer implements a method for color communication that is compatible with the well-established standards of the past and can also be easily translated into any future standard.

The PaintJet color communication standard was selected to meet the following requirements:

- The color communication system must be device independent. This infers that it should represent a color coordinate space that can be obtained from standardized color measurement methods.
- The color system should be well-characterized and its inherent limitations understood and recognized.
- The color communication system should be capable of representing the full color gamut of current and future color peripheral devices.

Since the computer industry is already familiar with various RGB (red-green-blue) color coordinate systems, it was decided that the PaintJet printer would use a coordinate space standardized to the NTSC phosphors and illuminant C as described in the following table.†

NTSC Standard Values

CIE 1931 Chromaticity Coordinates	x	y
Red	0.670	0.330
Green	0.210	0.710
Blue	0.140	0.080
Illuminant C (Alignment white)	0.310	0.316

The adoption of a standard such as this has the following advantages:

- It provides a universally understood color communication language for all peripheral devices.
- It provides a device independent color communication language for all software packages.
- It allows the specification of all visual colors, thereby allowing for future expansion to large color gamuts.
- It is upwards compatible with any future color image standard based on the principles of colorimetry.

The color communication method used by the PaintJet printer is a first attempt to direct the industry towards a universally accepted color communication standard. It is of the greatest importance to evolve this standard to one that is suitable for all peripheral devices throughout the industry.

Future color standards should supplement this current standard by providing a more robust and compact coding of color data, making it easier to compress and process a color image with little distortion. They should also provide explicit black and white compatibility and control over the basic color appearance variables so that image enhancement and user interface can be carried out in a portable and standard way. For instance, the uniform color space called L*a*b*, standardized by the CIE in 1976, is a good candidate for such a color communication standard. It offers some of the advantages just mentioned by describing color data with variables that are well correlated with the color's perceptual attributes. The adoption of such a standard for color communication will bring to the user of HP peripherals a level of functionality that has never been available.

†NTSC is the National Television Systems Committee. CIE is the Commission Internationale de l'Eclairage.

Don Palmer
Emil Maghakian
Project Managers
San Diego Division
Ricardo Motta
Member of the Technical Staff
HP Laboratories

ponent count. Therefore, we consistently opted for simple, straightforward designs, and for extensive integration of mechanical and electrical functions. Two tests that we specifically focused on to verify reliability were the environmental strife test and the BEST (board electronics strife test). Strife, which has become a standard HP test, subjects the product to seesawing temperature extremes of −20°C to 75°C at the rate of 2.5°C/minute. This cycling of temperature helps expose marginal designs and components in the product. Once exposed and understood the problem is fixed with a permanent solution.

Field data over the past few years has shown that although strife is an excellent tool for finding mechanical weaknesses in the product, it has been a less effective screen for electronics problems. Therefore, for the PaintJet printer we developed BEST, which focuses solely on testing the electrical system and its components. In this test the electronics, which have been separated from the product, are

Manufacturability of the PaintJet Printer

Five months after introduction, the PaintJet printer assembles in less than 20 minutes, has a 90% turn-on rate in the factory and has had less than ten field failures. The product successfully met its introduction finished goods inventory requirements and continues to meet its production goals. Key elements leading to this success were early manufacturing involvement, extensive use of vendor and expert inputs, and a team approach to problem solving.

Early manufacturing involvement was ensured by staffing the project with two manufacturing engineers during the investigation phase. This early presence allowed manufacturing to make numerous inputs before designs were put on paper. Receptivity to manufacturing inputs is high when the design is in a conceptual phase with little momentum. During the investigation phase, manufacturing engineers worked with the R&D team in defining the mechanical architecture of the product, reviewing conceptual designs, and investigating manufacturing process feasibility and process selection issues. Following process selection, manufacturing engineers supplied design guidelines to the R&D team who tailored designs to the limitations of the specific processes.

Expert and vendor inputs were solicited extensively during component design reviews that occurred during the product development cycle. Tooling requirements, manufacturing process requirements, tolerances, inspection strategy, material selection, and ease of assembly were all reviewed before tooling and part releases. These reviews by vendors, in-house tooling, plastics, and inspection groups exposed numerous potential problems. Design iterations solved many of these issues and others were resolved through tooling and process feasibility studies.

A team approach atmosphere was in place during the project. Manufacturing tried to avoid solely generating lists of manufacturing issues to be addressed by the R&D team. Instead, the manufacturing team identified potential manufacturing problems and then worked with R&D to find improvements or alternative solutions. This team approach helped create a more positive working relationship between manufacturing and R&D and resulted in an increased level of manufacturing influence on the product.

Acknowledgments

Numerous parties helped with the manufacturing development of the PaintJet printer. Thanks to the manufacturing engineers, the plastics engineering group, mechanical audit, the mechanical tooling department, the engineering test lab, the purchasing group, all the production workers and supervisors, and the product design team.

Eric Clarke
Manufacturing Engineering Supervisor
San Diego Division

exercised while being subjected to temperature slew rates of 10°C/minute. The circuit boards are then vibrated to a level that is four times higher than what our overall product is exposed to. Using BEST, we were able to identify and resolve six failure modes that were undetected by normal strife testing. The elimination of these six modes should markedly enhance the reliability of the overall product. Because of its extreme effectiveness, BEST has now been added to the repertoire of testing that all new products are subjected to at San Diego Division.

Design for Low Manufacturing Cost

San Diego Division's first attempt to design a very low-cost product was the HP 7470A, a two-pen X-Y plotter.[1] Although the PaintJet writing system, paper handling mechanism, and electronics are very different, many of the low-cost design concepts used in the HP 7470A are also employed in the PaintJet printer:

- Minimize part count by simple design and integration
- Eliminate adjustments
- Use injection molded parts extensively
- Avoid painting and machining operations
- Use snap fits rather than screws.

Of course, in the time since the HP 7470A, we have moved farther along the low-cost learning curve. We now believe the following items also belong on the list of best practices for low cost:

- Use standard, well-understood processes. Historically, when we have strayed from this concept we have suffered low yields, cost increases, and many engineer hours of process support and characterization.
- Minimize the number of parts that have cosmetic requirements. Cosmetic parts have lower yields and require special handling and packaging. A corollary to this is to avoid parts that have cosmetic requirements and tight tolerances. The process window that produces a part that meets both sets of requirements is usually narrow.
- Don't overintegrate. A part with too many specifications has a low probability of meeting all of its specifications. This translates into unsatisfactory yield and high cost.
- Get close enough to your vendors and their processes to understand cost sensitivities. This has several benefits. First, parts can be designed that avoid expensive or poorly controlled processes. Also, when a vendor asks for a price increase, it is easy to assess whether the increase is justified.

Software Support

For an application program to support a peripheral it is necessary for that application to have a specific driver for the peripheral. The PaintJet printer is a completely new kind of product for the San Diego Division. It is not a plotter, and it requires special software support. To address the needs of this product, existing programs were strengthened to develop relationships with new software vendors, and a product software integration group was formed in the R&D laboratory to address the technical challenges of software support. This group's objective was to develop the necessary tools and algorithms for proper support of the PaintJet printer.

It was deemed necessary that all of the key graphics packages that support HP plotters also support the PaintJet printer. Few of those packages supported raster printers at that time, and those that did support them did a very poor job. For example, some graphics packages took ten minutes to output an 8x10-inch graph on an HP LaserJet printer at 75 pixels per inch. To get a more realistic view of PaintJet software support, a driver for one of the popular business

graphics packages was commissioned. The first driver took 45 minutes to output a full-page business graph on an IBM PC class machine. This clearly was not acceptable. Our goal was to be able to output a full page of graphics to the PaintJet printer at 180 dpi in 3.5 to 4.5 minutes. This meant the software support had to improve by an order of magnitude.

Because the PaintJet printer is a raster device, its driver must first convert graphical objects using a vector-to-raster converter. It must then format the data for the PaintJet printer, attach the proper escape sequences to the data, and finally, transmit the data. An 8×10-inch graph requires 949.2K bytes of data. Our analysis of the commissioned driver identified the following bottlenecks:

- Vector-to-raster conversion. The most efficient means of raster conversion is a bit map. This is the technique used by IBM PC class machines for supporting screens. To use this method for the PaintJet printer, the driver would need 950K bytes of RAM space. This is an unreasonable amount of memory (the operating system can only address 640K or 655,360 bytes). As a result, the commissioned driver windowed the image and drew it in six pieces, each producing a 1.66-inch band of the graph.
- Data transmission. On an IBM PC class computer, data rates are very slow. At the operating system level one can expect a data rate of 250 bytes per second. At the interrupt 17 level, one can get about 1000 bytes per second. Our driver transmitted the data using the interrupt 17 calls. As a result, 950 seconds or 15.83 minutes were needed for data transmission.

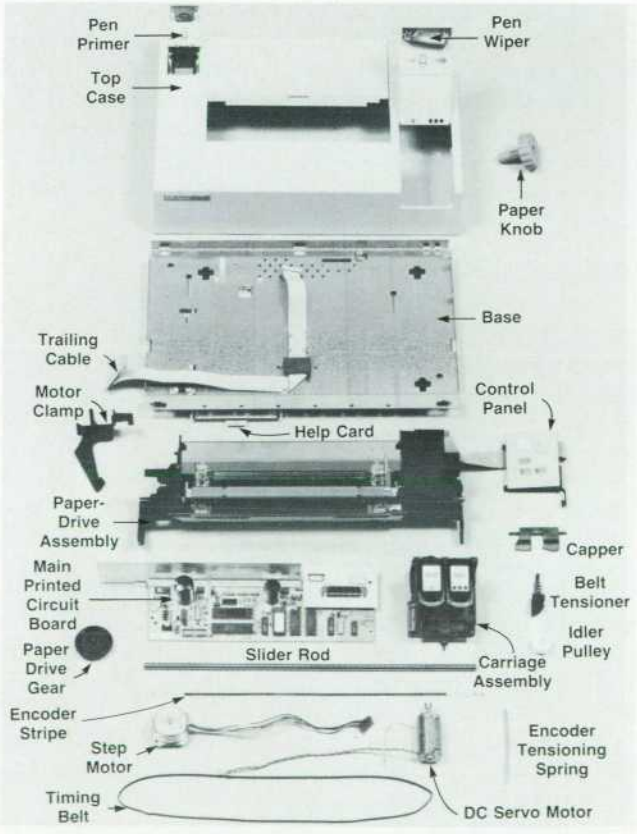


Fig. 3. *Exploded view of the PaintJet printer.*

Because software vendors did not want to spend their resources to solve these problems, a software toolkit was developed, and data compaction was implemented in the PaintJet printer. The toolkit contains code for a high-speed vector-to-raster converter, a data compactor, and fast I/O routines. The toolkit can convert an 8×10-inch image and send the data to the PaintJet printer in 45 seconds, running on a HP Vectra computer. The data compaction scheme compresses the data 4.5 times, on the average.

The Toolkit

The primary objective of the toolkit is to solve the vector-to-raster and I/O transmission bottlenecks. The secondary objective is to serve as an example that software vendors can tailor to their needs or use only parts of. It is written in Intel 8088 assembly language, the program is fully documented, and source code is included.

As mentioned above, the toolkit consists of three segments: a converter, a compactor, and I/O routines.

VPRC. The vector-and-polygon-to-raster converter first converts vectors and polygons to a compacted form that has all of the transformations applied to it, that is, all scaling, clipping, rotating, and so on. It then rasterizes the compacted objects. The compacted data form has a fixed byte format, which means it is very easy to find the next object in the list. Unlike most vector-to-raster converters, this converter directly rasterizes polygons. This not only improves the process, but also makes it possible to fill polygons with raster fill patterns (binary patterns made by turning bits on and off).

The converter uses a memory swath as wide as the printhead (10 rows). This minimizes the memory requirements of the VPRC to 12K bytes. It also allows overlapping of rasterization with printing. While the printer is printing swath i, the driver is rasterizing swath $i+1$. As a result, there is no waiting for the VPRC.

Data Compactor. To facilitate the use of the printer's compacted data transmission mode, a packing routine is provided. The packing scheme is run-length encoding in frequency and pattern byte pairs. (The pattern byte contains the raster information and the frequency byte indicates the number of times the pattern is repeated in uncompacted form.) Like any encoding process, run-length encoding can produce an encoded line longer than the original, the worst case being twice as long. To avoid this problem, an abandon clause has been added to the compactor, which transmits the unpacked data whenever the compaction process produces a longer data string.

Fast I/O. A hardware dependent I/O routine has been developed for IBM PC class machines and the Centronics interface. Data can be transmitted at a rate of 30,000 bytes per second on an IBM PC and 100,000 bytes per second on an HP Vectra computer.

The toolkit is provided to software vendors under a no-cost license. At introduction, more than 45 application programs had drivers for the PaintJet printer.

Acknowledgments

The successful development of the PaintJet product was the result of contributions from outstanding people in R&D, manufacturing, marketing, quality assurance, and accounting. We extend our thanks to each of you. In particular we'd like to recognize some of the people on the design team. Lynn Palmer, Dick Kemplin, and Paul Dubson developed the product design, enclosure, and chassis elements. Tony Ebersole contributed to the paper axis design and provided valuable ideas in many areas. Alpha Doan made numerous contributions to shock and vibration robustness and customer usability. Robert Beauchamp developed the belt drive, base, and media guide and helped on the primer. Frank Bockman developed the software toolkit. Dave Horn and Bob Dey deserve recognition for their product marketing efforts. Special thanks go to the manufacturing team led by Eric Clarke.

Reference

1. M. Azmoon, "Development of a Low-Cost, High-Quality Graphics Plotter," *Hewlett-Packard Journal*, Vol. 33, no. 12, December 1982, pp. 12-15.

Mechanical Design of a Color Graphics Printer

Among the issues were ensuring proper insertion of the print cartridge, making reliable electrical connections to it, moving the paper or film accurately, and designing a primer.

by Chuong Cam Ta, Lawrence W. Chan, P. Jeffrey Wield, and Ruben Nevarez

FROM THE MECHANICAL DESIGN VIEWPOINT, the PaintJet Color Graphics Printer offered several challenges. Among the more interesting ones were the development of the print cartridge latching mechanism, the electrical interconnect to the print cartridge, the media drive design, and the primer design.

Cartridge Latching

There were several objectives for the design of the cartridge latching mechanism. Since there are two print cartridges (also called pens), one black (Fig.1) and one three-color (Fig.2), there has to be a way to identify the correct placement of each type of cartridge in the print cartridge carrier (Fig. 3), which is called the carriage. The latching mechanism must be robust so that it won't be broken, and more important, so that it won't damage the print cartridge even if wrong placement of the cartrige occurs. The print cartridge loading and unloading procedure should be simple and obvious to the user.

Since the repeatability of print cartridge registration to the carriage directly affects the print quality, it must be controlled as tightly as possible. The 3σ repeatability goal is ± 12 micrometers. When the mechanism is latched, the print cartridges should not lose registration with respect to the carriage after a shock or other disturbance. This made it mandatory to design with stability in mind and choose registration surfaces carefully.

Finally, the latching mechanism must be manufacturable. This means a simple design, a reliable process, low part count, and low cost.

Implementation

Everyone knows one should read the instructions before attempting to operate a new product. However, many users skip this step and are frustrated if their attempt isn't a

Fig. 1. *Black print cartridge.*

success. More seriously, the user may blame the product's design if something breaks during the attempt.

To help the user find the correct place for each print cartridge, color coding is used. After priming and wiping a new pen, the user turns to the carriage. The obvious place to put the print cartridge must be one of the two openings on the carriage. The question is which one. Color matching the colored dots on the pen label with those on the carriage lid lable (Fig.4) will lead the user to the right place. Even if the cartridge is placed incorrectly, there will be no damage to either the pen or the carriage. The mechanism will refuse to latch to let the user know there is something wrong. Try again.

Loading and unloading the print cartridge is simple, as illustrated in Fig. 4. First, load the print cartridge to the correct opening on the carriage. Gravity will hold the cartridge in the right position to be clamped. Next, pull the green latch up to clamp the cartridge in place. To unload, reverse the procedure.

The keys to satisfying the repeatability and stability objectives are picking the best set of plastics for the print cartridge body, the carriage, and the latch to minimize the coefficient of friction between them, and picking the right locations for loading points and reaction points to minimize problems with frictional moments that would prevent the cartridge from moving freely to where it is supposed to be. We were faced with a narrow set of plastic options for the print cartridge body because of constraints including compatibility with the ink and dimensional stability after molding and after trigger curing (a process for quickly bonding the printhead to the pen body). Nor did we have a great variety of plastics to choose from for the carriage, because that plastic, aside from being a best match with the cartridge body plastic, should be a good bushing material, have high dimensional stability after molding, and be conductive to protect the electronics from electrostatic discharge. The easiest material to choose was the latch material, which needs to match the cartridge body material and have low wear characteristics. We picked 20% glass fiber filled Noryl (modified polyphenylene oxide) for the print cartridge body, polycarbonate with 10% glass, 10% carbon, and 15% Teflon for the carriage, and polycarbonate with 20% glass and 15% Teflon for the latch. Picking the right locations for loading and reaction points required some careful judgments and verification of the chosen set of locations by a free-body-diagram computer model of the print cartridge body.

The print cartridge body has some uncertainty in dimension. This is partly because of material shrinkage after coming out of the mold and partly because the body tends to deform after the trigger cure process, which allows the stress residue in the plastic to relax even more. To minimize this effect, the x and y registration surfaces are chosen as close to the printhead as possible. For the most positional stability, the z registration surfaces are chosen as far from the printhead as possible. Of course, these objectives have to be balanced with the physical size constraints.

Final Design

Fig. 5 shows the final latching mechanism design. The stainless steel spring A applies a load to the bottom of the latch B, which transfers the load to the two ears on the sides of the print cartridge C. These loads keep the pen cartridge against the x, y, and z registration surfaces D. The ears on the sides of the print cartridge are not identical. One ear has a beveled surface, which provides a side load to keep the print cartridge registered in the y direction. In addition to the beveled surface on the ear, a plastic spring E, which is part of the carriage, provides an additional side load to keep the print cartridge against the y registration surface in worst-case frictional conditions.

The assembly procedure for the mechanism is simple. Well-known processes are used for part making. The number of parts is minimized without risking design simplicity and design margin. Worst-case analysis of the final design was followed by exhaustive tests to guarantee high reliability.

Electrical Interconnect to the Print Cartridge

The main objectives in designing the interconnect system were high contact reliability, minimum head-to-paper distance to maintain acceptable print quality, and low cost.

The design concept is based on a direct electrical connection with the pen substrate using a set of spring-loaded metal contacts. The leadframe, which makes contact with the pen substrate, is electrically connected to the carriage printed circuit board through a flexible circuit. The leadframe has four groups of contacts, each group consisting of 16 contacts with a 1.1-mm center-to-center spacing. Fig. 6 shows a leadframe with its four groups of contacts. The leadframe is gold-plated and is laminated on both sides with Kapton, a polyimide film made by DuPont. The Kapton helps maintain the position alignment of the contacts and increases the rigidity of the system. The three holes are for alignment with the registration features in the carriage.

Fig. 7 shows a leadframe in contact with the color and black print cartridges. As the cartridge is latched into the carriage, the substrate on the cartridge pushes the leadframe contacts up to their final positions. The contact force between the leadframe and the substrate helps establish a

Fig. 2. *Three-color print cartridge.*

reliable contact. Fig. 8 is a side view of a leadframe in contact with a print cartridge, showing that one side of the leadframe is in contact with the substrate pad and the opposite side is soldered to a flexible circuit.

The relative motion between a leadframe contact and a substrate pad during latching helps dig through or remove any contamination on the pad, thereby improving contact reliability. The height of a leadframe contact tip is only 0.4 mm, which keeps the paper-to-printhead clearance to a minimum. Furthermore, because the spacing between leadframe contacts is very close, the surface area of the expensive gold pads in the substrate is reduced.

The leadframe is made of 0.008-inch beryllium copper and is laminated with Kapton on both sides. Beryllium copper is chosen because of its good formability and mechanical properties. It is plated with nickel and gold; the former serves as a diffusion barrier between beryllium copper and gold, and the latter has excellent corrosion resistance and high electrical conductivity. The flexible cable is made of 1-oz copper and is laminated with Kapton on both sides.

Fabrication Process

The flat leadframe is first chemically etched from 0.008-inch beryllium copper sheets. The Kapton base coat and cover coat are prepunched and then laminated onto the leadframe under heat and pressure. The part is then formed to its final shape. Finally, the leadframe is plated with nickel and gold.

The flexible circuit is first chemically etched and then laminated with a prepunched Kapton layer. Finally, solder is applied onto the solder joint areas by a silk-screening process.

The leadframe, the flexible circuit, and the carriage printed circuit board are soldered together using a reflow soldering process. The three parts are placed in a fixture that has alignment pins to define the parts' relative locations. The parts in the fixture are placed on a conveyor belt which passes through several reflow stations. At each station, a spring-loaded clamp is applied to ensure that the solder points are in physical contact. Then, a high-intensity infrared lamp is turned on to solder the parts together.

Problems and Solutions

The goal of the interconnect system is to have a mean time between failures (MTBF) of 20 months or more. The interconnect system contains 32 contacts between the leadframe and each of the two pen substrates. If one or more of the contacts is blocked electrically by contamination during operation, the system is considered to have failed. Because of the proximity of the substrate to the nozzle plate, wet ink can easily be spread to the substrate area. When the ink dries, the residue forms a thick nonconductive crust which could block the electrical connection between the leadframe tip and the pen substrate. Furthermore, the leadframe tips are very close to the paper. During printing, coating flakes and fibers from the paper can fall into the contact areas, and occasionally block the connection between the leadframe tips and the substrate.

To overcome these problems, each leadframe tip is shaped so that it can dig through the ink crust on the pen substrate during latching of the print cartridge into the carriage. The width of the tips was reduced to 0.25 mm from the original 0.40 mm so that it is less likely that a tip will trap any contamination underneath it. A plastic shield is placed in front of the carriage facing the paper to block any paper particles from making contact with the leadframe tips. These features of the design help the system to exceed the MTBF goal with some margin.

Another goal was to prevent corrosion of the leadframe tips for at least five years. Corrosion can be caused by poor plating or excessive wear between the contact and the pen substrate during latching. Higher contact force helps dig through any contamination layers on the substrate and thus increases the contact reliability, but it also increases wear on the surfaces. If the gold plating on the leadframe tip is worn through, causing the nickel layer to be exposed, the nickel can react with atmospheric pollutants and corrode. Also, poor plating can leave minute pores on the gold layer, which may allow atmospheric pollutants to attack the unprotected nickel layer. Therefore, the gold layer has to withstand all the latching and remain intact at the end of five years.

The solution is to select the optimal gold thickness and hardness for the leadframe based on a series of wear tests. The integrity of the gold plating is demonstrated by subjecting the leadframes to an accelerated corrosion test. By exposing the parts in an enclosed chamber to some concentrated pollutants, the reactivity of the atmospheric pollutants on the gold plating over a long period of time can be simulated. Ten days in the test environment is equivalent to five years of normal service.

Tolerances

The amount of deflection in each leadframe tip is based on three components:

- The height of the substrate relative to the leadframe z-axis datum after latching. This tolerance is relatively small and easy to control.
- The height of the leadframe tip relative to its z-axis

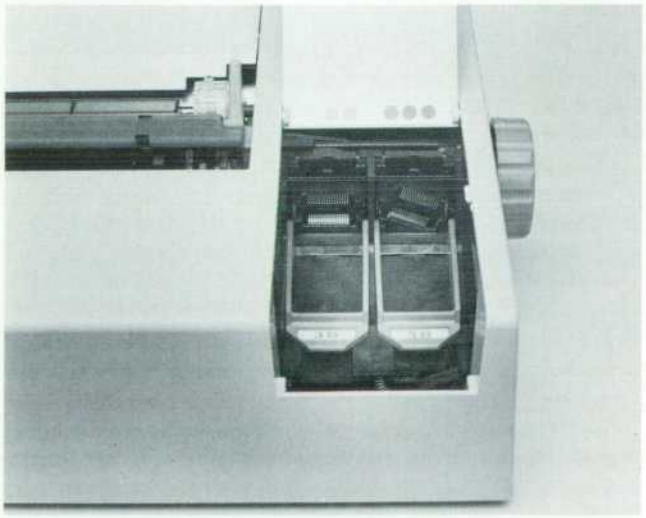


Fig. 3. *PaintJet printer pen carriage, showing the openings for the two pens.*

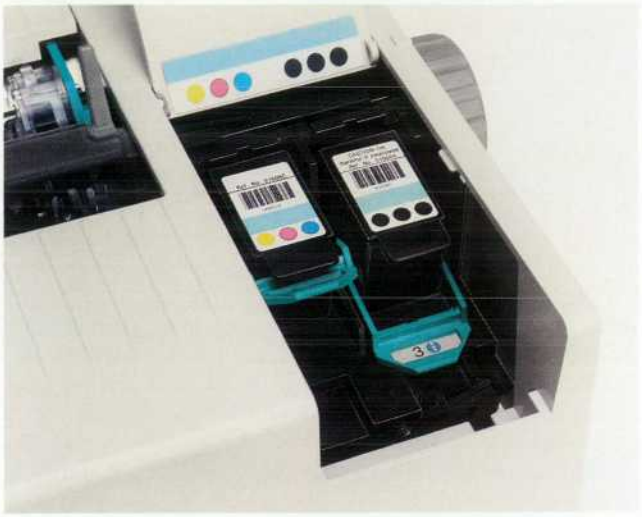


Fig. 4. *Matching the color-coded labels on the pen and the carriage opening ensures proper placement.*

datum. The higher the leadframe tip, the less it deflects.

- The height variation between adjacent leadframe tips. The variation makes it more difficult for the higher tip to make contact with the pen substrate. When the substrate touches the lower tip, the Kapton between the two tips acts like a compression spring which pushes the higher tip even higher until the deflection force of the higher tip equals the Kapton force. This mechanism is the largest tolerance in this group.

Manufacturing tolerances on all of these dimensions are chosen to guarantee that every tip makes contact with the pen substrate after latching.

Other critical tolerances on x-axis and y-axis alignment prevent situations in which certain leadframe tips make contact with the wrong substrate pads or snagging occurs between certain leadframe tips and the pen substrate or nozzle plate.

Media Drive

The PaintJet printer sprays a swath of color precisely 1/18 inch wide. The print media (paper or film) must then advance to match. An advance slightly more than the swath width will result in a gap between adjacent swaths, and falling short of the required move will overlap swaths. The challenge is to design the media drive to accommodate this very specialized movement. Modeling, measurement, and simulation were all involved in the solution. The design uses a step motor and a two-stage gear reduction (see Fig. 9).

Modeling

The difference between the two extremes of pitch radius in a gear is the total composite error (TCE), a measurable quantity. The contribution of a particular gear to the total drive inaccuracy is related to this parameter. An effective analysis of this relationship can be made by modeling the system using eccentric nonslipping discs, where the instantaneous radius of the disc is equal to the instantaneous pitch radius of the gear. A larger disc will rotate an adjacent disc more than a smaller one. The angular transmission error $\Delta\Theta$ of a gear is related to the instantaneous error in pitch radius ΔR by the following expression for small rotations Θ of the gear:

$$\Delta\Theta = \frac{\Theta\Delta R}{R + \Delta R}$$

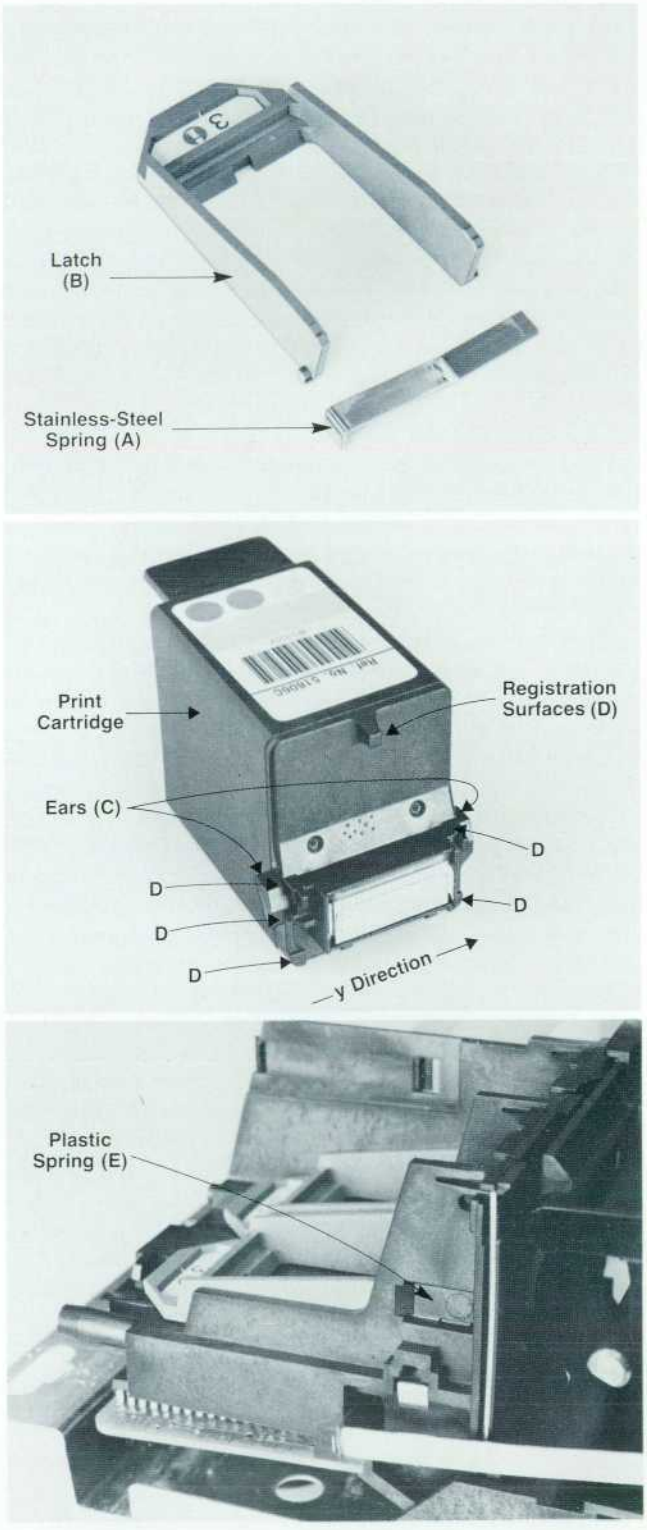


Fig. 5. *Latching mechanism design.*

When the worst-case angular transmission error is required, the largest error in pitch radius can be approximated as $\Delta R = TCE/2$. Then:

$$\Delta\Theta_{max} = \frac{\Theta(TCE)}{2R + TCE}$$

Notice that the angular transmission error is dependent on the amount of rotation (Θ) and is less for a large radius (R). Therefore, higher step accuracies can be achieved through smaller movements such as the 1/18-inch swath width, and gears contributing the most to media drive inaccuracies will be small-radius gears.

Using injection molding technology, small parts can be made more accurately than large parts. Thus the requirements lend themselves to an injection molding process provided that the concentricity of the gear teeth and the center mount is set in the mold very accurately. Through inspections and iterations, the gear molds for the PaintJet printer were adjusted to near perfection. The inherent repeatability and accuracy of the injection molding process for small parts allows these gears to fit well within their error budget.

Measurement and Simulation

Visual appearance of the graphics output shows how well the media drive advances paper, but to analyze and improve the system, a measurement technique was developed using a 50,800-count encoder coupled to the drive roller. Measurement accuracy of 1/100 degree is possible.

This measurement was used to confirm the gear model, but it also revealed another source of error in the motor. Step motors have a step accuracy specification, and the effect of this error was easily predicted as it was attenuated through the gear train. The measured error was something else. A strong every-other-swath event was observed. Its cause was found by Fourier analysis of the drive roller position output. The Fourier analysis converted the encoder signal into a frequency-versus-magnitude map of the drive roller's angular position. Each gear showed a peak at its predictable frequency. The only component matching the frequency of the every-other-swath error was the alternating phases of the step motor. Each phase exerted a different torque and the dynamic effect of this difference resulted in alternating overshoot and undershoot of the correct step size. Implementation of a motor deceleration algorithm solved the problem by minimizing the "kick" applied to the drive roller at the last step. The torque difference between phases then became insignificant.

How much step accuracy is required? Simulation of various inaccuracies involving the frequency and magnitude of each error was done by a combination of modeling and measurement. With the encoder mounted to the drive roller, the angular position of the roller could be measured very accurately. With the appropriate high-resolution input we could rotate the drive roller to any desired position, perfect or not. A step motor with a 1000-to-1 gear ratio was set in motion to drive the roller. When the right position was reached the step motor was turned off and a graphics swath was printed. Although slow, 10 to 12 hours per page, the process produced a well-defined typical graphics output that could be evaluated by a panel of judges for its appearance.

Using the model, worst-case errors caused by individual mechanism components were analyzed to assess their effect on the printed page. This ensured high-quality output in even a worst-case scenario. Exhaustive testing then proved we had a cost-effective, highly accurate media drive.

Primer Design

Whenever the PaintJet printer's output is unacceptable, the user needs a quick and easy way to correct the problem. Also, when a user opens a new cartridge, it needs to be serviced to begin printing. The primer (Fig. 10) and wiper have been designed for these situations.

Typical examples of degradation are missing drops and missing colors, which can be caused by trapped air bubbles, contaminants in and on the nozzles, and shock to the cartridge. To correct these problems, the user can take the cartridge out of the carriage, open the primer lid by rotating it 180 degrees, and insert the pen into the primer lid. The next step is to rotate the lid 90 degrees so that the pen cartridge stands up and the primer lid windows are facing upward. The primer lid is then pushed down against a spring and held down until the windows are filled with ink. This should take a few seconds. Once the windows are filled with ink, the lid can be allowed to come up and the pen can be taken out and wiped with a wiper located under the carriage lid. The problem should now be corrected and the cartridge ready for printing. However, it may occasionally be necessary to repeat the procedure to solve all of the printing problems.

When the user pushes the primer lid down, the cartridge is pushed against a rubber bladder, which causes a seal to form. As the lid is pushed farther down, the bladder is compressed, forcing air into the cartridge and pushing the

Fig. 6. *The leadframe has 64 contacts in four groups.*

Fig. 7. *A leadframe in contact with print cartridges.*

ink out towards the nozzles. Air in front of the ink in the nozzles is pushed out. Once the ink reaches the nozzles, the capillary force holds the ink in the nozzles.

In designing the primer, the objectives were to minimize the cost and to make the primer reliable and user-friendly.

To meet the objectives, a number of cost reduction approaches were taken. For example, most of the parts are made of plastic, which allows integration of many features into a single part, thereby reducing the number of parts. Plastic also makes it possible to snap fit the parts together, simplifying the assembly process. The primer lid is a good example. It holds the cartridge for priming, holds the absorber pad that captures the ink that is ejected, holds the chamois that draws ink away from the absorber pad, covers the primer unit when not in use, and snaps into the primer unit.

Having the user do the priming eliminates the need for hardware and software to run the primer. As a result, the challenge in the design was to make the primer reliable and friendly while maintaining user control of the priming.

To make the primer user-friendly, the first step was to develop a single procedure for the user to use whenever there is a printing problem and when the cartridge is first used. The procedure needed to be simple so the user would have a short learning curve and remember it easily.

User Tests

Beginning with the first prototype, and continuing with each successive iteration, a user test was conducted to determine the friendliness of the primer. The test results were analyzed and used to design the next iteration. The tests were conducted by non-R&D personnel and videotaped. In each test, nontechnical people unfamiliar with the PaintJet printer were asked to unpack a print cartridge and follow a set of directions, which included priming and wiping.

The user tests were invaluable. Our first iteration, for example, did not have windows in the lid. The user was simply instructed to push the lid down for three seconds and then allow the lid to spring up. The videotapes showed users pushing down slowly and only part way down, holding the primer lid down too long, and pushing down too many times. It became obvious that some feedback was needed to let the user know how long and how far to push down. A number of ideas were considered. The ideas ranged from controlling the effective priming time to visual, tactile, and audio feedback methods. Visual feedback—ink filling the windows in the lid—was chosen. The advantages of this feedback method are that the solution required simple tools, and most important, that it clearly shows the user what is going on during priming. This is significant since the hold-down time varies somewhat with how fast the primer lid is pushed down and how much ink is in the cartridge. Another advantage is that the amount of ink wasted is diminished, since the cartridge is pumped only until it is primed and no more.

Fig. 9. *Media drive gear train.*

Another lesson learned from the user tests was that if there is an opportunity for the ink to get on the user, it will happen. One of the problems faced while designing the primer was what to do with the ink that is ejected out of the print cartridge. The initial solution was to have a small absorber pad that would soak up the ink as it was being ejected. According to our user model at that time, the absorber pad would become saturated at about the time that one of the ink cartridges would need replacement. The idea was to send a clean absorber pad with each print cartridge so the user could replace the used one. However, the test results showed increased risk of ink transfer to the user. The design was then changed to eliminate the need for the user to replace the absorber. The solution is to have a permanent absorber pad that soaks up the ink as it is being ejected, and to have a chamois in contact with the absorber pad to pull the ink away from the pad. The chamois has a large capacity for ink, and because the user only needs to prime the cartridge approximately once per month, there is sufficient time for the ink in the chamois to evaporate between uses. The permanent absorber also eliminates the need for the user to learn and remember how and when to change the absorber.

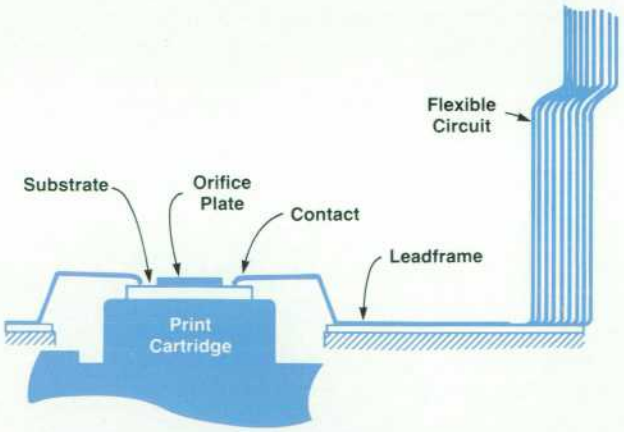


Fig. 8. *Side view of a leadframe in contact with a print cartridge.*

Single Procedure

It is not always necessary to prime and wipe the pen to solve a particular printing problem, some problems can be eliminated simply by wiping the pen. However, the tests clearly showed that the more of a thermal inkjet expert a user was asked to become, the more likely it was that something would go wrong. What we found to work best was to provide a single procedure that would be reinforced

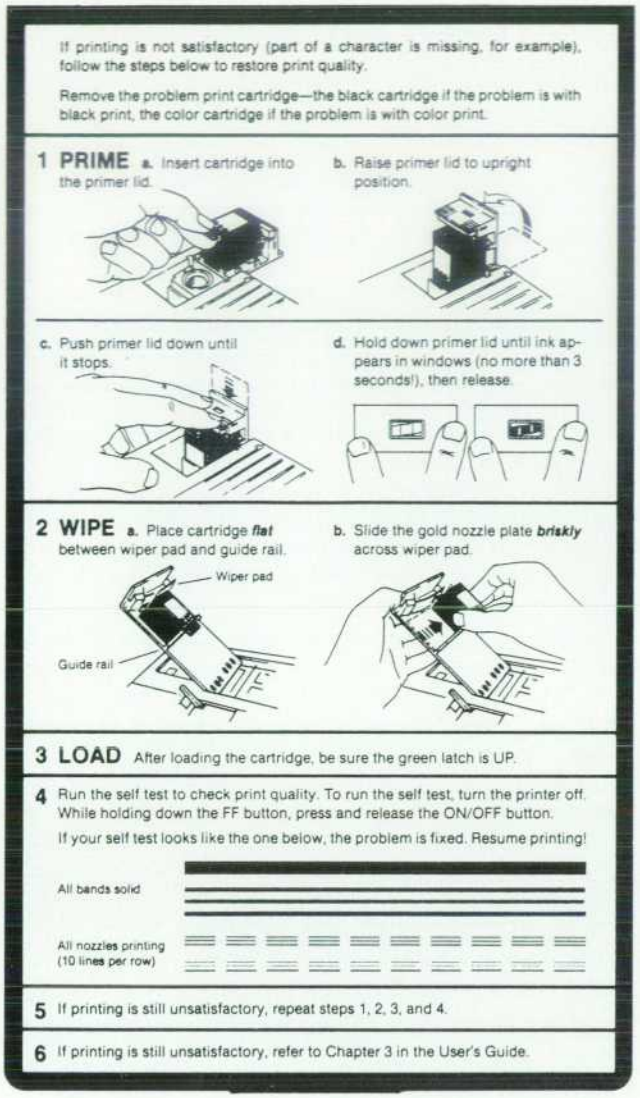


Fig. 10. *The primer allows the user to activate the pen and correct printing problems. These instructions are on a pull-out card attached to the printer.*

again and again. The procedure is included twice in the user guide, once in the beginning to guide the user when a new cartridge is used, and a second time in the troubleshooting portion of the guide. For further reinforcement and to serve as a quick reminder, a pull-out self-help card is provided, and icons are placed on both the primer lid and the front cover. The icons refer the user to the instructions and give the order of the priming and wiping operations.

Besides being user-friendly, it is imperative that the primer be effective at solving the problems that the user may encounter. Problems can be caused by paper dust getting on a nozzle or on the substrate, a small air bubble depriming a single nozzle, or large air bubbles depriming an entire color section. The whole cartridge can be deprimed because it has been dropped, for example, or some unknown contaminant has clogged the nozzles.

The different types of failures can be solved with the prime and wipe procedure because they fall into two kinds of failure modes. Contaminants on the substrate pads or on the nozzles can be removed by wiping the nozzle plate against the wiper, and the deprimes (from single nozzles to complete deprimes) and contaminants caught in the nozzle can be fixed by priming and wiping (in this case the wiping serves to clean off the ink left on the nozzle plate after priming). Combining the two procedures yields a single procedure that will solve just about any type of printing problem that is user-correctable.

An important reason for visual feedback to tell the user when priming is finished is that the print cartridge will withstand only a limited number of primes. The more the pen is primed and the higher the priming pressure, the larger is the likelihood that the cartridge will develop air paths through the foam that holds the ink. The pumped air pushes the ink in the path of least resistance. In time, a path can be created in which there is no ink, giving the air a path to the nozzle plate. However, if the cartridge is not primed for very long and the ink is allowed to reach a state of equilibrium, the chances of creating an air path are diminished. Once an air path is established, the pen cannot be primed any longer. Visual feedback helps the user see when priming is completed, reducing the tendency to prime longer than necessary.

The higher the priming pressure, the less time is required to prime the cartridge and the easier it is to dislodge air bubbles around the nozzles. The lower the pressure, the more times the cartridge can be primed without creating an air path. The pressure had to be carefully chosen to give high reliability and long life with the user in control of the priming. The cartridge design team also made improvements to the foam insertion procedure, which lessened the likelihood of air paths forming and caused a marked improvement in the robustness of the cartridge.

Another area of concern in the primer was the absorber pad which wicks the ink away from the nozzle plate. The concern was that the absorber pad material would leave some contaminants on the substrate pads or nozzles. The absorber pad needs to have a very fast absorption rate, so that when the color pen is primed, the three different inks are absorbed into the pad and do not mix with each other and contaminate the primary colors. The pad's capacity has to be sufficient to hold ink for approximately four consecutive primes. The logical choices are fibrous materials. Unfortunately, most fibrous materials tend to lose fibers, expecially as they become wet. After a large number of different materials were tried, a treated polyester was found that left an acceptably small number of fibers on the cartridge.

Acknowledgments

Robert Beauchamp was responsible for a number of design contributions in the primer, and Barry Mauerman contributed to its simplification. Henry Flournoy's use of digital signal analysis proved to be a valuable diagnostic tool for the media drive. David Ellement's analysis and implementation of the step motor's acceleration profiles resulted in a significant product improvement. Curt Torgerson and Frank Nasworthy's relentless pursuit of high-quality plastic gears has made a definite contribution to plastics technology with patents pending. Special thanks to Todd Russell, who devoted his time to developing a free-body-diagram computer model of the print cartridge body. The model contributed to the success of the cartridge latching design.

The Second-Generation Thermal InkJet Structure

Changes in materials and processes increase resolution from 96 to 180 dots per inch and extend printhead life from 2 million drops to 7 million drops.

by Ronald A. Askeland, Winthrop D. Childers, and William R. Sperry

THE PRINCIPLES OF OPERATION of the HP PaintJet print cartridge are identical to those of the HP ThinkJet print cartridge.[1] Ink is channeled to specified chambers containing a thin-film resistor on the floor and a small orifice on the ceiling. The thin-film resistors are rapidly heated to temperatures exceeding 400°C. The ink directly over an excited resistor is vaporized and a bubble is formed. As this vapor bubble grows, momentum is transferred to the ink above the bubble, which causes this ink to be propelled through the orifice onto the paper. Ink is refilled automatically to the resistor area by capillary action.

The performance specifications of the PaintJet printer required a second-generation material set rather than further tuning of what had been developed for the ThinkJet program. Print resolution is increased from 96 dots per inch to 180 dots per inch. This requirement increased the resistor count from 12 to 30 resistors per printhead. Usable ink volume is increased from 3.5 ml to 12 ml. The ink volume for each drop is reduced from 220 picoliters to 100 pl. These changes require a 3.5-fold increase in resistor life.

The most striking performance improvement offered by the PaintJet printer is its ability to generate over 330 different colors. This is achieved by combining patterns of magenta, yellow, and cyan droplets, all generated from a single cartridge. This requires significantly different ink management schemes.

Material Selection

A cross-sectional view of the PaintJet printhead is shown in Fig. 1. Silicon has replaced glass as the substrate material. Although substantially more expensive, it offers performance improvements in the areas of defect density and thermal capacity. Thin-film parameters and photolithography parameters are also more easily controlled on silicon.

A thermal capacitor is required so that the heat generated by the thin-film resistor is transferred to the ink. This heat pulse lasts less than 5 μs and creates a vapor bubble. The capacitor thickness is selected so that excess heat is removed after bubble formation. Excess heat can cause unwanted secondary nucleation and drop performance degradation. The main sources for heat removal are the silicon substrate and the ejected ink. Several thermal barrier materials were tested, including Al_2O_3, SiO_2, Si_3N_4, and SiC. The optimum barrier was found to be a thin layer of SiO_2. This material was found superior for thickness uniformity, defect density, and etch resistance to the chemicals used to define the various thin-film materials.

The resistor film is TaAl and the conductor film is Al doped with a small percentage of copper. These films have been used in several printhead applications at HP over the years. The resistor material is less than 0.1 μm thick and therefore does not substantially contribute to the thermal capacitance.

Next come Si_3N_4, SiC, and tantalum passivation layers. These layers protect the resistor and conductor materials from chemical attack by the ink. In addition, protection from severe hydraulic forces induced by the collapsing vapor bubble is provided. It is desirable to keep these layers as thin and uniform as possible because they can

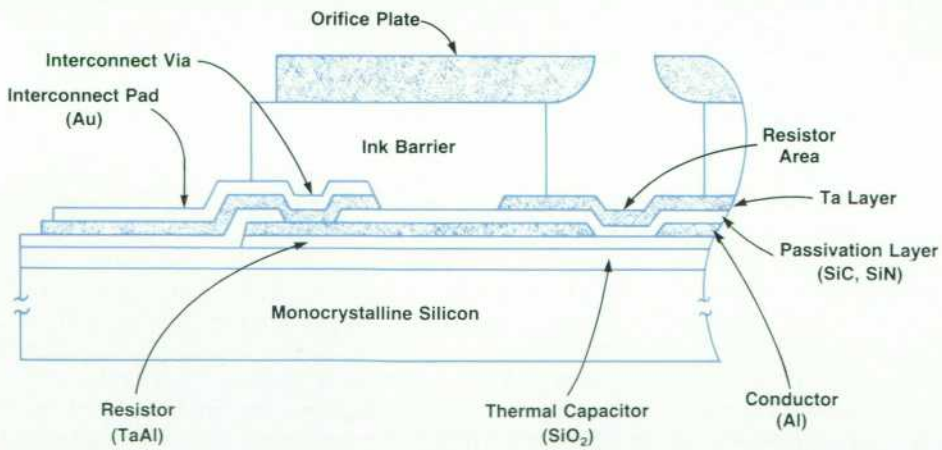


Fig. 1. *Cross-sectional view of the PaintJet printhead.*

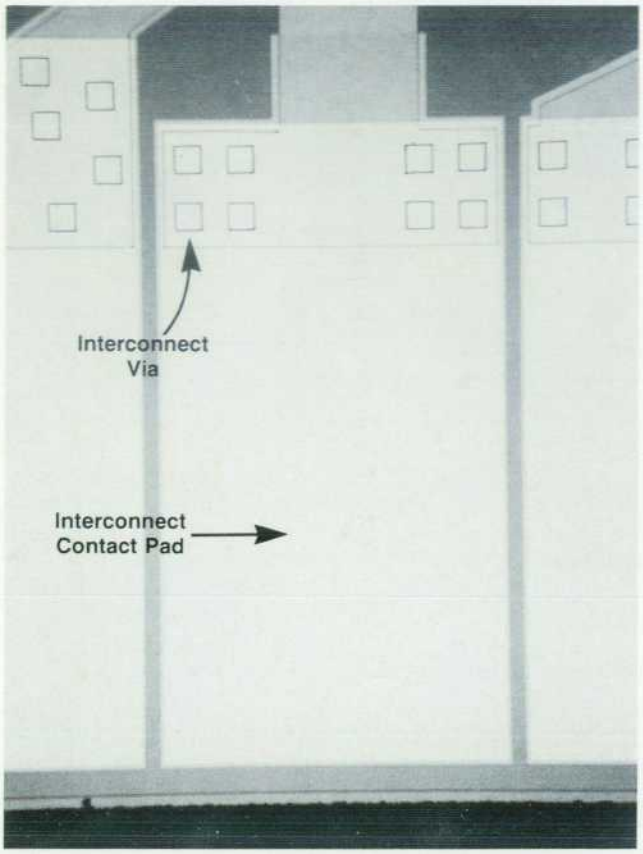


Fig. 2. *Contact pads on the printhead are connected to conducting traces by multiple interconnect vias.*

negatively affect thermal response. The design thicknesses are optimized for step coverage and pinhole density factors.

Contact between the print cartridge and the PaintJet electronics was a major challenge. The interconnect design was driven by the requirements that contact be made directly to the silicon substrate and that the cartridge be inserted easily by the customer several times throughout its life. Each time, a low-resistance contact to each of 30 resistors must be achieved. In addition, this contact scheme had to have an extremely low profile so that the minimum printhead-to-paper spacing of less than 1 mm could be maintained. Durable, low-resistance contact pads are designed along the periphery of the silicon substrate. Each pad has a contact area of 1×1.4 mm and consists of a sandwich of Ta and Au. The pad is electrically connected to a conductor trace by a series of small vias etched through the Si_3N_4 and SiC layers (see Fig. 2). The Ta film offers a very tough, scratch-resistant base, and the Au film ensures a chemically inert, low-contact-resistance surface. The multiple-via concept offers some basic design advantages. Multiple parallel, low-contact-resistance paths are formed. High reliability is achieved since a failure at any of these independent sites will not affect the overall performance of a trace. Interconnect insertion tests have shown that cartridges can survive more than 15 insertions and environments such as 65°C at 90% relative humidity without performance degradation.

Ink management was a major design consideration in the PaintJet print cartridge. In the color cartridge each group of ten resistors requires an isolated ink supply system (see Fig. 3). The material selection constraints for the printhead were driven by performance issues. To achieve 180 dpi and conserve silicon real estate, the spacing between resistors is 200 μm. The plated metal wall design used in the ThinkJet printhead cannot meet this spacing requirement. To deliver a 100-pl drop, a channel thickness of approximately 50 μm is required. Fig. 4 is a micrograph showing the resistor area and ink channel geometry. Channel features of less than 100 μm are also required to achieve both drop volume control and fluidic impedance balance. The adhesion of the channel material to the silicon substrate is very important. Finally, this material must be relatively chemically inert since it will be in contact with the various inks for periods as long as 2½ years.

A photoimageable polymer was identified and special imaging and developing techniques were engineered to meet these specifications. The 2½-year storage life was confirmed using an accelerated test procedure in which cartridges were hermetically sealed and stored at 65°C for seven weeks. The performance parameters for more than 500 cartridges were measured before and after storage and found not to deteriorate.

To supply ink from the reservoir of the cartridge to the channels, feedholes through the silicon are required. The challenges associated with feedhole drilling include the

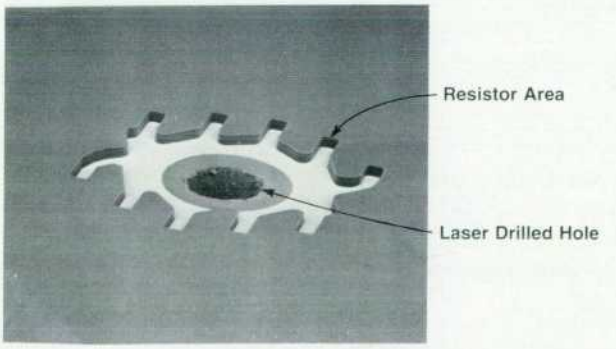


Fig. 3. *In the PaintJet color cartridge, each group of ten resistors is supplied with ink through a laser-drilled hole.*

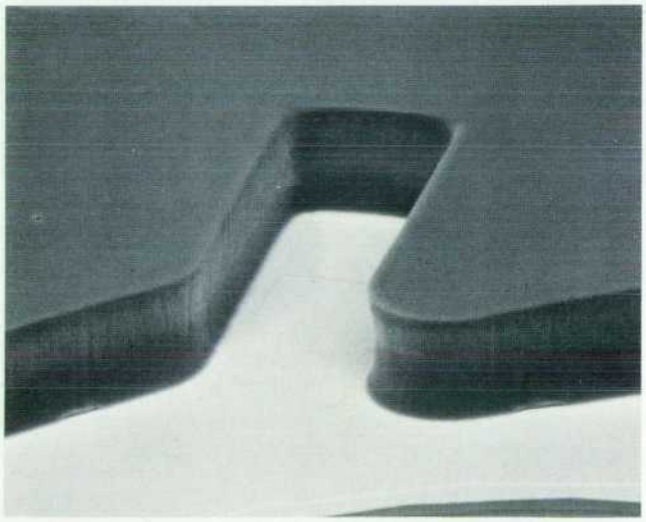


Fig. 4. *Closeup view of the resistor and ink channel area.*

diameter required (300 μm through 500-μm-thick silicon), the fragility of the thin films, and the manufacturing throughput requirements (2 seconds/hole). A number of alternative methods were investigated, including contact drilling, chemical etching, and noncontact drilling.

The first two alternatives had serious disadvantages. Contact drilling required frequent, costly tool replacement. Chemical etching tended to damage the thin films. Laser drilling was found to be the most cost-effective of the feasible alternatives. Laser drilling is achieved by focusing a high-powered Q-switched YAG laser with a beam spot size of 150 μm. The beam is trepanned around the desired circumference to create a hole. Molten silicon residue (drill slag) is minimized by strict control of all the laser cutting parameters.

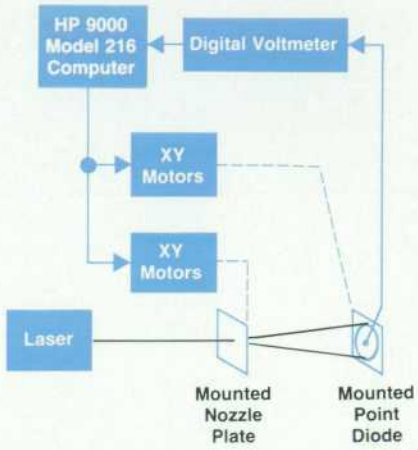


Fig. 6. *System for measuring orifice plate nozzle diameters to an accuracy of ±0.5 μm.*

Orifice Plate Properties

The orifice plate controls both drop volume and direction. This plate contains 30 precisely placed holes less than 50 μm in diameter. The dimensional tolerances required by this part necessitated advances in plating technology. In addition, new measurement techniques were developed to ensure diameter accuracy.

The orifice plate, composed of gold-plated nickel, is manufactured by electroforming. This process is illustrated in Fig. 5. A sheet of stainless steel is coated with a thin, electrically insulating layer. This layer is patterned and subsequently defines the outline of the orifice plate as well as the inkjet nozzles. The patterned sheet (mandrel) is placed in a nickel-plating tank and brought to a negative potential. Nickel electrodeposits onto the conductive surfaces. Electrodeposition cannot occur directly onto the insulation layer, so the insulating discs result in circular orifices. As electrodeposition continues, the nickel overplates the edge of the disc as shown in step 3 of Fig. 5. The nozzle diameter decreases as the nickel thickness increases. Once the electroforming is complete, the nickel sheet is separated from the mandrel, cleaned, and gold-plated.

There are a number of critical orifice plate properties. Some of the most notable are corrosion resistance, surface finish, sheet dimensions, and orifice diameter. The typical nickel-plating chemistry required some modification for this application. An organic brightener is added to improve the electroform surface finish. This helps eliminate corrosion site defects and allows the use of a thinner protective gold layer. Delamination of the electroform from the mandrel because of tensile stress is eliminated by the addition of saccharin, a commonly used stress reducer. This additive produces a sheet with a controlled level of compressive stress. The magnitude of this stress is critical to the control of the overall sheet dimensions.

The most critical property is the orifice diameter. Micrometer variations in orifice diameter cause large drop volume changes, which have a dramatic impact on print quality. The final inkjet nozzle diameter is approximately equal to the starting insulating disc diameter minus twice the plated nickel thickness. Controlling the diameter of the insulating disc is straightforward because semiconductor photoimaging techniques are used. More difficult is control of the amount of overplating. The tolerance control required is beyond normal plating industry standards. A very precise plating current and plating tank geometry are re-

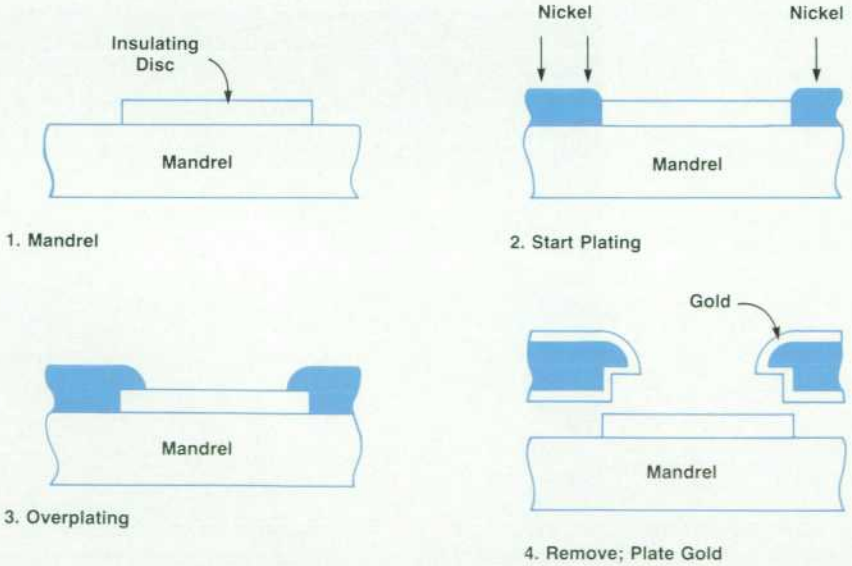


Fig. 5. *Orifice plate electroforming process.*

quired to achieve the tolerances demanded.

Measuring individual nozzle diameters to an accuracy greater than ±0.5 μm presented a challenge. This is accomplished using the Fraunhofer diffraction principle (Fig. 6). Laser light is directed through the nozzle orifice, producing a diffraction pattern which is detected by a point photodiode. This diffraction pattern is scanned to determine the diffraction ring diameter. The nozzle diameter is a monotonically decreasing function of the ring diameter.

Resistor Reliability

The PaintJet printhead reliability goal is for cartridges to run out of ink before resistor failures occur. Black cartridges can print more than 1100 pages of text using an ink volume of 12 ml supplied to 30 nozzles. Since the drop volume is 100 pl, there is enough ink for 4 million drops per nozzle if all resistors are fired equally. In normal printing, resistors near the center of the printhead are used more often than those on the edges; therefore the resistor reliability goal was set at 7 million drops. The comparable goal for the first-generation ThinkJet printhead was 2 million drops (500 pages of text).

The printhead thin-film resistor and passivation layer are subjected to chemical, mechanical, and thermal stresses during operation. When a resistor is energized, a vapor bubble is formed which causes an ink droplet to be ejected through the nozzle. As the vapor bubble collapses, the passivation layer over the resistor is subjected to extremely large fluid cavitation forces. The mechanical pounding induces cracks and craters in the passivation layer and thin-film resistor and even damages the underlying SiO_2 layer. This resistor damage leads to a constriction in the current path across across the resistor and subsequent current crowding resistor failures (see Fig. 7).

The high temperatures reached during operation result in thermal stress to the resistor and overlying thin films. While thermal stress alone does not cause resistor failures, it does make the resistor more vulnerable to cavitation damage and chemical attack.

Resistor life tests of more than 1000 cartridges were conducted to ten times normal printhead life using continuous-ink printing life testers. The life testers were built from HP 7470A Plotters modified to hold PaintJet print cartridges. The duty cycle, pulse width, and voltage supplied to the printheads were precisely controlled. All tests were run at worst-case product overpower conditions. Resistor failures were detected by monitoring resistance values during the test. Autopsies of failed resistors provided valuable feedback for printhead thin-film design.

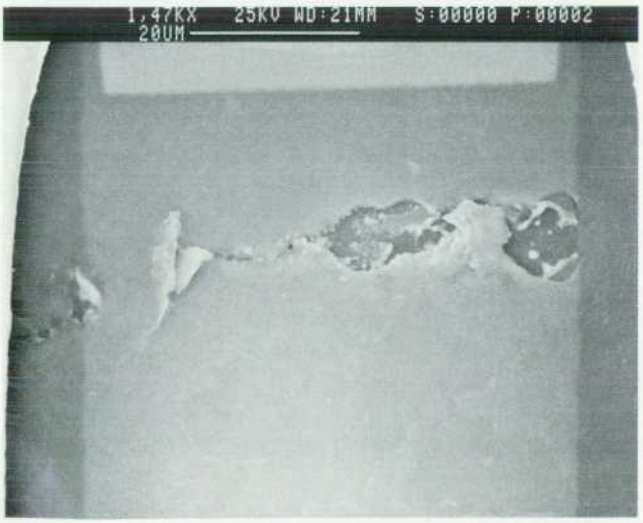


Fig. 7. *Scanning electron micrograph of a failed thin-film resistor.*

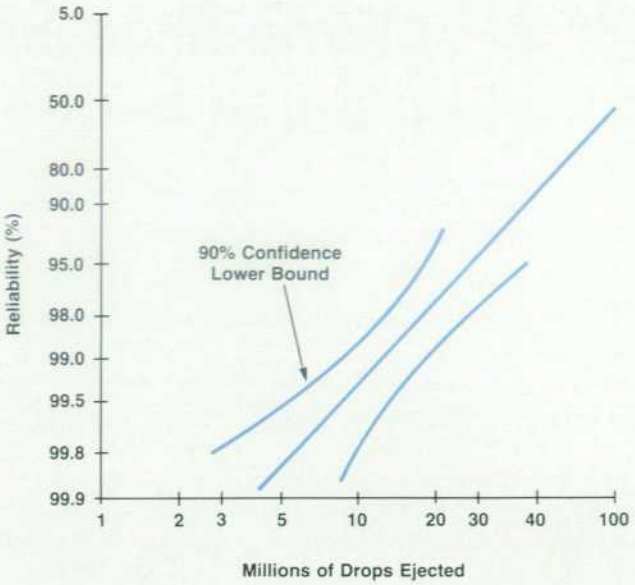


Fig. 8. *Weibull plot for printheads.*

Resistor life data was analyzed using the Weibull distribution. This method allows us to estimate resistor life at a given reliability with a specified confidence level. PaintJet printhead resistor life exceeds our goal of 7 million drops at 99% reliability and 90% confidence at worst-case overenergy conditions (Fig. 8).

Acknowledgments

Although the material was first designed, tested, and proved feasible at HP's San Diego Division, the characterization and development of the production processes were done at the Corvallis Division. Major contributions from the San Diego team were made by Ulrich Hess (SiO_2 layer), Vyomesh Joshi (resistor reliability), Wistar Rhoads and Dave Otis (ink channel), Steve Steinfield (laser drill), and John Stoffel (via design). For the Corvallis Division, major contributors were Steve Aden (via etching R&D and manufacturing characterization), Marshal Field (laser drill), Dan Kearl (passivation layers), Paul McClelland (electroforming), Rich Van de Poll (ink channel), and Shel Whittington (ink channel). With 1500 miles between sites, the management of this transfer was particularly challenging. We thank Ron Prevost, Marzio Leban, and Rob Beeson for the outstanding effort required to make this a success.

Reference

1. R.R. Allen, et al, "Thermodynamics and Hydrodynamics of Thermal Ink Jets," *Hewlett-Packard Journal*, Vol. 36, no. 5, May 1985, pp. 27-33.

High-Volume Microassembly of Color Thermal Inkjet Printheads and Cartridges

Miniature parts and micrometer mechanical tolerances make high-volume assembly challenging. Adhesive selection was the first step. Special fixtures, tools, automatic machines with vision, instrumentation, and systems had to be developed.

by Cheryl A. Boeller, Timothy J. Carlin, Peter M. Roessler, and Steven W. Steinfield

WHEN A NEW TECHNOLOGY slated for very high-volume production is being developed, manufacturing engineering concerns have to be addressed early. In the case of the HP PaintJet print cartridges, the R&D lab had its charter to invent designs and processes that would satisfy performance and life objectives. In parallel, manufacturing engineering had to ensure that the design and processes would meet the objectives of high volume, high reliability, and cost-effectiveness.

Manufacturing Constraints

The miniature sizes of the parts and the mechanical tolerances required to assemble a functional printhead proved to be one of the more challenging aspects of this development. While those working with electronic circuit manufacturing and photolithographic processes are accustomed to dealing with dimensions measured in single-digit micrometers, mechanical engineers are most comfortable thinking in terms of sheet metal, machining, or plastic molding tolerances, where on a good day one can hold tolerances of 100 micrometers.

Early tests showed that to obtain a printhead with acceptable drop directionality characteristics, the orifice plate needs to be aligned to the substrate with an error no greater than 10 μm. Furthermore, the alignment accuracy of the printhead to the plastic ink reservoir had to be less than 50 μm to maintain black-to-color print registration and electrical interconnections. These requirements seemed to border on the limits of feasibility, especially if the the assembly processes incorporate thermal cure adhesives. Fig. 1 shows a representation of the relative distances involved. Observe that the differential thermal expansions between the mating parts considerably exceed the alignment requirements for both orifice-plate-to-substrate and substrate-to-pen-body assemblies.

Assembling the Printhead

To meet the extremely tight alignment specification necessary for successful functioning of the printhead, careful conceptual analysis of the assembly algorithm was required. At the time manufacturing engineering became involved, the process that was in place consisted of placing

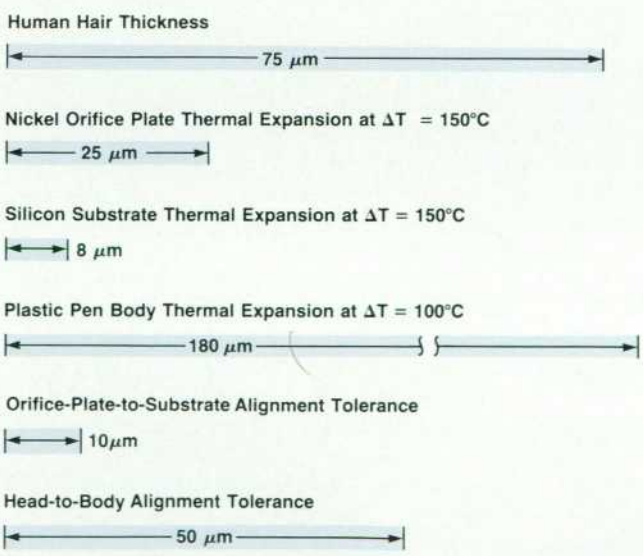


Fig. 1. *Comparison of notable thermal inkjet physical dimensions to the thickness of a human hair.*

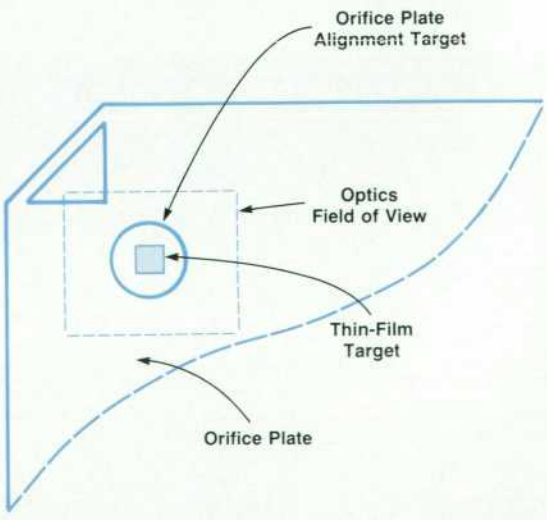
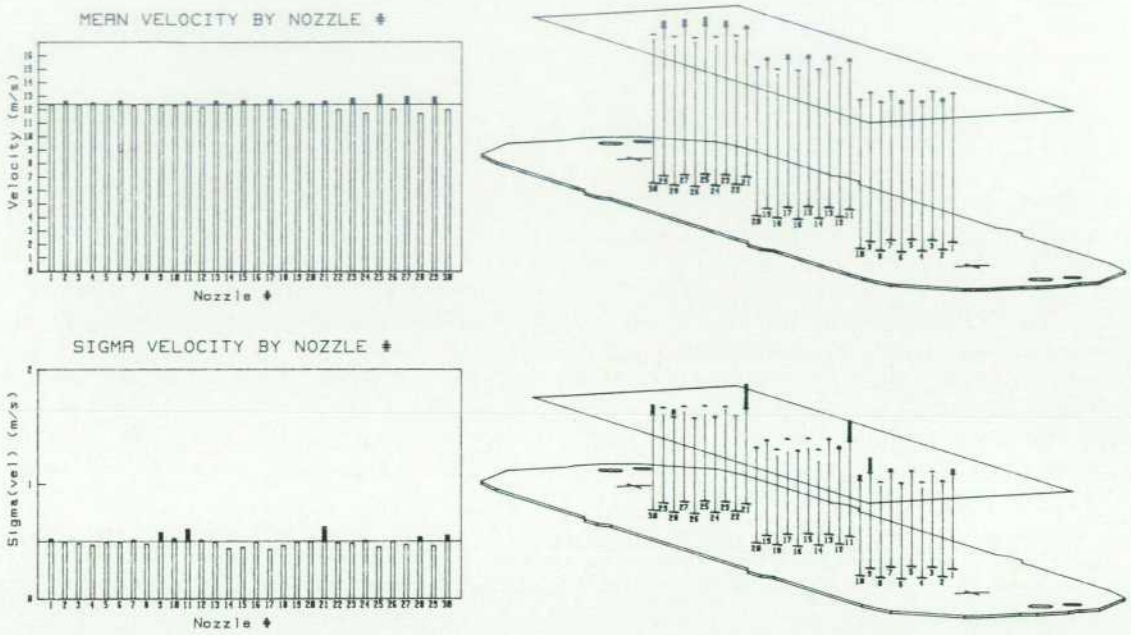


Fig. 2. *To align the substrate to the orifice plate, translation and rotation stages are moved to align thin-film targets on the substrate with target holes in the orifice plate.*

Drop Velocity Analysis Plot—Good Pen

Number of Pens in Population: 1 Total Number of Nozzle Measurements Represented: 30

Velocity Data Acceptance Window: 5 to 20 m/s



Drop Velocity Analysis Plot—Bad Pen

Number of Pens in Population: 1 Total Number of Nozzle Measurements Represented: 30

Velocity Data Acceptance Window: 5 to 20 m/s

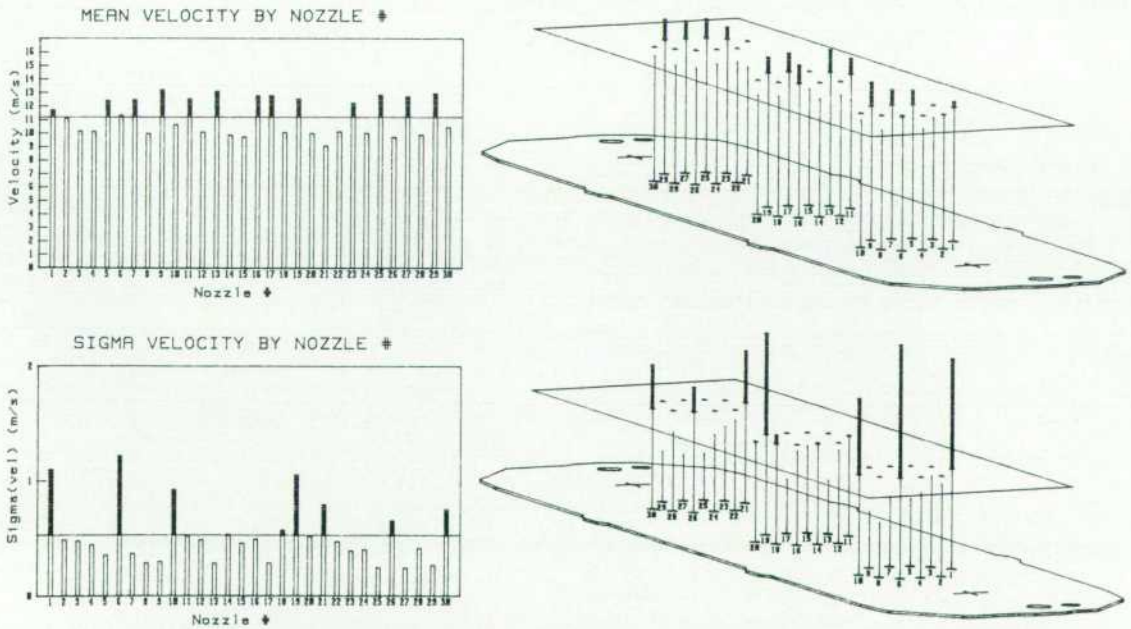


Fig. 3. *Drop velocity analysis plots for good and bad pens.*

a film of thermal cure adhesive between the substrate and the orifice plate and then aligning the assembly under a microscope in a fixture that clamped the head with over 100 pounds of force. (Today we use roughly one pound of force.) Next the entire fixture was placed in a 200°C oven for one hour and then removed and allowed to cool before removing the head from the fixture. Although the adhesive bonded the orifice plate to the substrate adequately in the short term, it was known to fail in the corrosive ink environment after a fairly mild exposure. Furthermore, the system as a whole did not meet the specified alignment tolerance. This was because of the mechanical clamping required as well as the high-temperature cure cycle (i.e., the differential thermal expansion of the material set as depicted by the chart in Fig. 1). Finally, the practical manufacturing feasibility of mechanically fixturing each printhead for over an hour was virtually zero. Clearly, this assembly process was at best a temporary one.

We immediately began to investigate alternative systems for building the printheads. For our new assembly scheme to be successful, the bond between the orifice plate and the substrate had to perform three functions: a tack function, a gasket function, and a structural function. We knew that we had to bring the parts into alignment to within ±10 μm, hold them together within the time constraints of high-volume production (tack function), and then have that bond resist continuous exposure to the highly corrosive ink environment for a projected 2.5 years without allowing any leaks between color chambers or outside of the ink delivery path (gasket function). Also, the bond would have to withstand various forces in handling and operation, as well as temperature cycling excursions that typically cause adhesive joints to fail as a result of fatigue, embrittlement, and thermal stress effects (structural function).

We brainstormed many alternatives, including pressure sensitive tapes, ultraviolet (UV) radiation activated cures, two-part epoxies, and surface activated and dual-cure-mechanism adhesives, in various combinations with mechanical, magnetic, and other processes. We did a formal decision analysis that looked at such performance factors as handling time, cure time, dispensability, process controllability, cure temperature, and number of process steps.

Twenty-eight different adhesion schemes were analyzed and ranked in order of desirability. The result was that the alternatives involving a thermal cure fell to the bottom of the stack. So we set off looking for a UV/surface activated adhesive that would give rapid holding strength with UV exposure and then continue to cure at room temperature once the surfaces had been brought into contact. Other alternatives included various pressure sensitive tapes and frozen, premixed two-part epoxies in preformed sheets that cure rapidly when brought to room temperature.

However, we soon discovered in one test or another that all of these convenient solutions didn't meet one or more

Automatic Alignment Machines

One of the first challenges confronting the PaintJet printer pen manufacturing team was performing orifice-plate-to-substrate and head-to-body alignment and attachment in high-volume production. Volume forecasts justified a fully automatic, high-speed production machine. Since HP engineering resources were not available to develop the system in-house, we contracted with two companies. One specializes in machine vision and the other in high-precision material handling.

Schedule constraints mandated that design of the pen alignment machine (PAM) begin before the assembly processes were fully developed. Because of this, a major reset occurred when the process was changed to require additional steps after orifice/substrate attachment. Because these steps didn't lend themselves to in-line production, PAM was split into two machines (Fig. 1): HAM (head alignment machine) and SAM (skeleton alignment machine). HAM performs orifice/substrate assembly and SAM performs head/body assembly, allowing the extra operations to occur off-line. The initial design of both machines was done by our contractors, but major HP efforts went into optimizing the design, improving reliability, developing the control software, and integrating the components into reliable, fully operational production machines.

Mechanics

The mechanical design of the machines was driven by many constraints such as tight alignment tolerances, high throughput, reliability, and cleanliness. The basic process steps, such as singulating orifice plates, dispensing adhesive, and aligning parts, are performed in modular stations. Substrates, orifice plates, heads, and pen bodies are fed into the machines in stackable part carriers and are conveyed from station to station using vacuum to minimize handling damage. Feedback from a variety of sensors informs the control computers of any machine malfunctions, which might include jams, misplaced parts, and mechanical failure of a machine component.

A significant mechanical challenge is to position parts repeatably using rotary arms up to 10 inches long. Motors with high-resolution encoders (4000 counts per revolution) are used in combination with hard stops to meet the positioning requirements. A carousel with eight high-precision pen body holding fixtures is used on SAM to transfer pen bodies to each workstation. Each

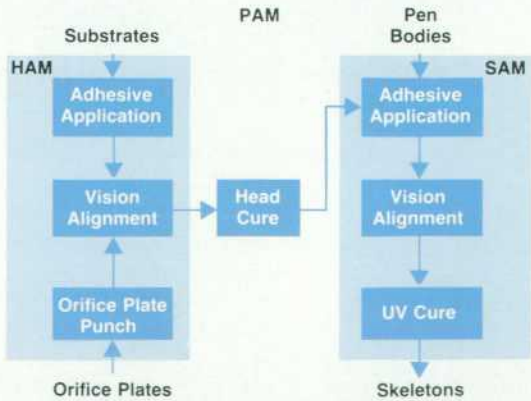


Fig. 1. *The pen alignment machine consists of the head alignment machine and the skeleton alignment machine.*

fixture accurately locates pen bodies for successive operations such as: application of three adhesives, alignment of heads to pen bodies, and adhesive curing.

To simplify the design and maintenance of HAM and SAM, a concerted effort was made to replicate components and use common parts. Many components such as dc servo motors and coordinated motion controllers were selected because they are used for other purposes at San Diego Division. This allows easy access to hardware with a history of good performance.

Machine Vision

Machine vision enables HAM and SAM to align the pen components automatically to micrometer tolerances. HAM finds an alignment hole at each end of the orifice plate and a square target at each end of the substrate. The vision system views the targets through the alignment holes and informs the controller to rotate and translate the substrate such that the targets align to within 9 μm of the centers of their respective alignment holes. After alignment, the parts are attached and become an assembled head. Similarly, SAM rotates and translates a head until the orifice plate alignment holes are within 15 μm of their ideal locations on a pen body.

While machine vision is fast and accurate, there are several elements that make human vision difficult to replace. Dirt and other surface imperfections, easily ignored by the human observer, can make alignment features impossible for the machine to find. The machine is also sensitive to changes in light levels, making periodic calibration necessary. The small field of view of the camera and the proximity of other potentially confusing features make part misplacement a problem. A major effort was required to develop complex machine vision algorithms and optics to handle these situations and to minimize scrapping of functionally acceptable parts.

Control Systems

HAM and SAM have independent electronic control systems with similar designs. The components controlled include 53 motors, 64 pneumatic valves, and 165 sensors. Each control system has three major parts:

- A master control processor, an HP Vectra computer
- A vision system, consisting of a PDP-11 computer, an image processor, and a TV camera with split-field optics
- Local control processors, which are Z80 CPU boards.

The local control processors control the use of motor, pneumatic controller, and analog-to-digital converter cards. All communication is via RS-232-C, and all major cards are off-the-shelf STD BUS components.

Software for HAM and SAM consists of extensive master controller code and Z80 assembly code. Several techniques are used to keep this large amount of software maintainable. Master controller code is split into control code and menu code. An emulator was written to enable each part to be debugged without the need for the other.

Both the master controller code and the Z80 code run under VRTX, a multitasking operating system. Reentrancy is used to reduce program size. Program changes are minimized by allowing commonly changed parameters to be downloaded to the Z80s from the Vectra computer at run time. One of the difficulties inherent in a project of this nature is that to develop the mechanics the software is required, and to develop the software the mechanics are required. The development of both mechanics and master controller software simultaneously was greatly accelerated by making the Z80 code capable of communicating with either a Vectra computer or a dumb terminal.

Results and Lessons Learned

HAM and SAM do the job of ten operators and five manual machines. Throughput and alignment quality have both improved substantially since their implementation.

As in all major development undertakings, we learned some important lessons from the PAM project. First, development time can be reduced by ensuring that the assembly processes are stable and well-characterized on manual tooling before designing automated equipment. Automated equipment should be designed in-house or contracted to experienced assembly machine builders. Specialties, such as machine vision, can then be brought in as needed. In no case should specialists be the primary system designers.

Jeff Beemer
Project Leader
Mitch Levinson
Glen Oldenburg
Mick Trego
Ed Wiesmeier
Manufacturing Engineers
San Diego Division

of the "must" requirements. Surface curing adhesives and epoxies either cured too slowly or did not survive in the ink environment, tapes did not guarantee a consistent seal and did not have long-term dimensional stability, and other schemes were either not dispensable or not available as an off-the-shelf product. The adhesive testing process is described in detail later in this article.

In the tests, the only alternatives that met the performance "musts" were thermally cured. So we needed to rethink our alternatives given the additional constraint that a thermal cycle of up to one hour would be required on the printheads. In a high-volume environment it would be impractical to hold one hour's worth of production in traveling alignment fixtures. We had to develop a tacking scheme that would allow quick removal of the assembly from the alignment station and maintain the relative position of the orifice plate to the substrate until after the thermal cycle was complete.

We chose an ultraviolet-curing tack scheme once it was shown to be possible to cure some UV adhesives in under one second if radiation of the proper wavelength and intensity is used. Originally, we used screen printing to dispense a layer of adhesive approximately 75 μm thick. The effort to make it work was abandoned following the realization that the processes could not be made reliable enough for our manufacturing environment. Later, precision dot dispensing using an air-pulsed syringe was found to be superior for our application. In the present process, one dot of a UV curable tack adhesive is placed in each of the four corners of the substrate before the alignment station. This is done with an HP-designed dispensing tool that uses a special optical sensing scheme to locate the syringe relative to the parts and thus ensure accurate dot placement. The volume of each dot dispensed in this way is typically 25 nanoliters. To put this in perspective, if you placed one dot per second into a one-gallon container, it would take over five years to fill it up. Because of this miniscule use, adhesive vendors were not interested in developing custom

formulations for our application. Therefore, we were limited to off-the-shelf products.

Once the adhesive is dispensed onto the substrate, the problem of aligning the orifice plate to the substrate remains. Very early in the development of the manufacturing plan, an automatic pen alignment machine (PAM, see box, page 34) was contracted. In the meantime, several iterations of HP-designed manual tools were designed and built. The original assembly line incorporated three of these manual tools, which facilitate production cycle times of around 15 seconds per printhead. Both the manual tools and the new automatic machine PAM are based upon closed-loop alignment of the substrate to the orifice plate, with translation and rotation stages being moved to align thin-film targets on the substrates to target holes on the orifice plates (Fig. 2). Once aligned, each tool drives an orifice plate onto a substrate under a specific load to ensure intimate contact between the parts. Once the proper load is achieved. UV light sources flood the printhead components to cure the tack adhesive.

The Gap Problem

After we had developed the process and tools to build the printheads, we began to notice variations in their performance. Fig. 3 shows a representation of the printhead where the vertical vectors correspond to parameters measured on the JULIO system (see box, page 37). To have good print quality it is critical that the drops of ink travel from the printhead to the paper at a controlled speed, and that the drop volume is held within an acceptable window. The plots in Fig. 3 show an example of JULIO data from a good pen and a bad pen. In the bad case, the nozzle-to-nozzle drop velocity variation is unacceptably high.

We discovered that this unhealthy drop velocity characteristic was associated with small gaps between the orifice plate and the substrate, as represented in Fig. 4. These gaps could be on the order of a couple of micrometers and still have a drastic effect on drop velocity. With a gap present, some of the energy from the thermal pulse of the resistor is lost in the crevices, and the resultant drop ejection rate is decreased and highly irregular. The crevices were also ideal locations for air bubbles to accumulate, further degrading drop formulation and ejection.

This problem was complicated by the interaction of the ink with the printhead materials over time. The barrier material would swell and change shape after several days

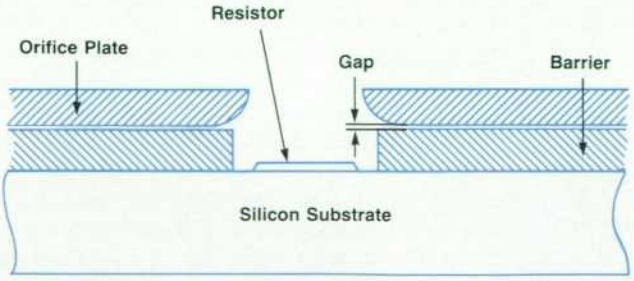


Fig. 4. *Small gaps between the orifice plate and the substrate caused reduced and irregular drop velocities. A new process eliminates them. (Not drawn to scale.)*

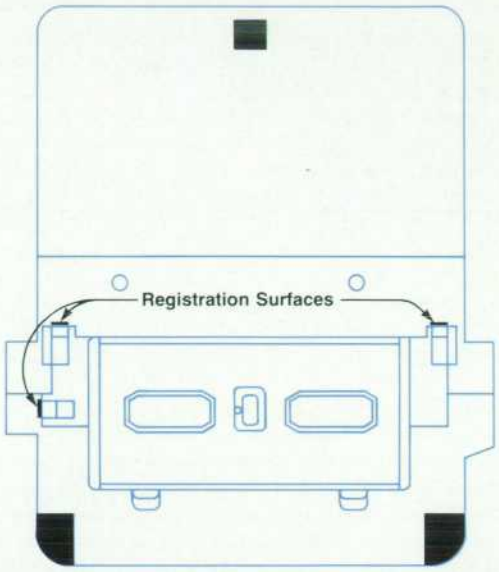


Fig. 5. *The printhead has to be aligned with molded features on the pen body, shown in black on this drawing.*

or weeks of contact with the ink, and this would affect the shape of the firing chamber and the occurrence of gaps. We needed to develop a process that would eliminate the gaps and ensure intimate contact between the orifice plate and the barrier material for the life of the pen. This is exactly what we did. Unfortunately, the process is proprietary and cannot be divulged here.

Making the Print Cartridge

In parallel with solving the problem of printhead assembly, we were developing a strategy for bonding the finished printhead to the ink reservoir. This problem had many of the same constraints that we had been facing with the printhead, namely extremely tight alignment tolerances and part handling difficulties. Also, the same cycle time, environmental, and ink exposure resistance requirements would have to be met.

There were some fundamental differences as well. Instead of adhering the orifice plate to the barrier and thin-film materials, this problem was one of adhering the silicon-backed printhead to the polyphenylene oxide pen body. The thermal expansion difference between the mating parts (i.e., silicon and polyphenylene oxide) was much greater than in the orifice/substrate case (see Fig. 1), making thermal cure adhesive systems even less attractive. Furthermore, in the head/body system, the printhead has to be aligned with respect to molded features on the pen body (see Fig. 5), rather than the much friendlier concentric targets simultaneously visible when aligning the orifice plate to the substrate.

Our initial approach to head/body alignment relied upon closed-loop optical positioning of the printhead in a fixture that held the body in a presumably known position. An operator would adjust the printhead's position with respect to a template representing the proper location. We soon encountered difficulties with this system. First, the creation of a template that satisfied our specifications was dif-

JULIO

Early in the development stages of the PaintJet print cartridge, head architects identified the need for a measurement instrument to quantify the drops being ejected by the printhead. The first such instrument is called JULIO, for "jets (of ink) under laboratory intensive observation."

JULIO can control head temperature, ink back pressure, resistor drive voltage, pulse width, and frequency. It can measure drop velocity, drop trajectory, and drop volume. Drop velocity is measured with an optical system consisting primarily of a laser beam and a pair of photodetectors (Fig. 1). When the drop flies through the beam, the signals at the photodetectors are interrupted consecutively, signaling the event. JULIO then calculates the velocity using the known distance and the measured flight time between the detectors.

Drop trajectory is measured by locating the drop in two planes approximately parallel to the nozzle plate. A strobe light illuminates the drop from two orthogonal directions parallel to the nozzle plate, and a camera and frame grabber capture the drop images. The printhead is then moved up a specified distance, the nozzle is fired again, and two new views of the drop are captured. These four images of the drop are digitized and the centroids of the drop images are calculated. Knowing where the drop is in both planes, the angle of the drop can be calculated. The drop's trajectory relative to the nozzle plate is determined by probing the surface of the nozzle plate with a sensor that determines its angle relative to JULIO's coordinate system.

Drop volume is measured by firing drops into a cup which is then weighed on a precision balance to determine the average drop weight. Volume is calculated using the known ink density. JULIO verifies that the drops are fired using the laser and detectors to ensure that the weight measured represents the correct number of drops fired.

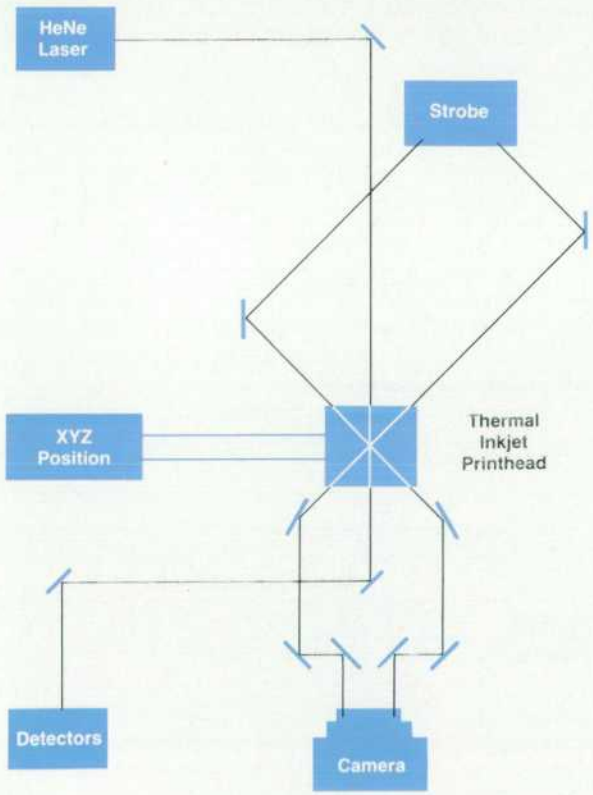


Fig. 1. *Top view of the optical path of the JULIO measurement system.*

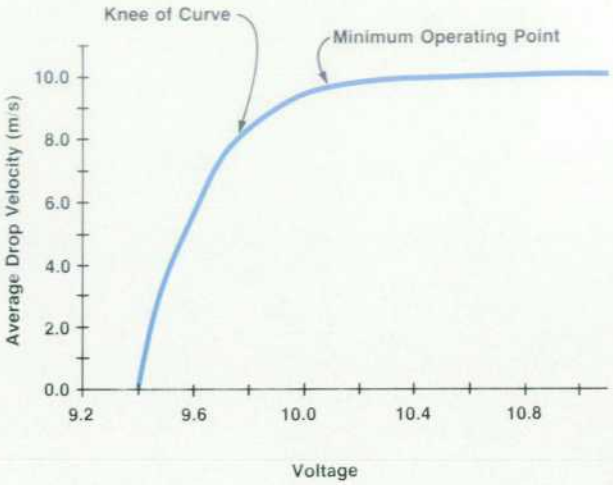


Fig. 2. *Typical result of an operating window test of a printhead.*

An application of the drop velocity measurment on JULIO is the operating window test (Fig. 2). In this test, the velocity is measured as a function of the resistor drive voltage to determine the correct drive pulse for the pen. In the start-up test, the velocities of a specified number of drops are measured and plotted against order fired to observe the variation of drop velocity over time.

The tests on JULIO were invaluable in evaluating the performance of the printhead during its development. They are still used daily to monitor the quality of printheads made on the production line.

Acknowledgments

The author would like to thank Bob Haselby, Ken Stone, and Keith Cobbs for the design and development of JULIO.

Don Bergstedt
Development Engineer
San Diego Division

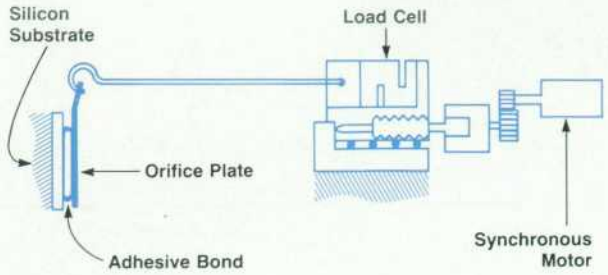


Fig. 6. *Peel tester used in the adhesive tests.*

ficult at best. Also, operators had a difficult time doing the alignment. Parallax, depth of field, and cycle time were some of the major obstacles.

A major breakthrough occured when one of the engineers suggested a simple open-loop alignment scheme. Historically, the relatively loose tolerances on the substrate edges made this approach seem impractical. To get around this, the scheme takes advantage of the near perfect photolithographic dimensions of the orifice plate. A vacuum chuck was designed which has three pins that protrude only 0.003 ± 0.0005 inch. The pins are wired to LEDs which illuminate when in contact with the electrically conductive orifice plate. When the operator sees the three LEDs light up, the printhead is fixtured in the right place. Our model shop quickly fabricated the extremely delicate vacuum chuck. Using the same pen body holding fixture and UV sensitive tack adhesive as before, along with a built-in UV light source, the new open-loop aligner proved not only feasible, but superior to the optical method in every way. We improved our initial alignment, yield, and cycle time considerably.

Adhesive Testing Process Details

Adhesives were the obvious means of fastening the PaintJet printhead together and attaching it to the pen body. However, selecting from existing products was more difficult than initially anticipated and problems with these products continue to occur occasionally in production. Unlike mechanical fasteners, adhesives are unruly and plagued with endless unknowns. They also appear to be endless in variety and potential as technology improves. Thus the adhesive selection process can also become endless.

An iterative trial-and-error mechanical testing approach was used to find a working adhesive set. A general understanding of adhesive chemistry was useful in controlling variables. Greater expertise in the field of polymer science would have streamlined the selection process, but would not have eliminated the necessity of trial-and-error selection. Most adhesive compositions are proprietary and the compatibility of various adhesives with substrates and ink was unknown. Vendors were relied upon to supply products that they felt were appropriate and these products were tested to determine their feasibility.

Reliance on chemical analysis was heavy in the later stages of adhesive screening. Those in-house with chemical expertise supplied information on optimal cure conditions, analysis of contaminants, batch-to-batch variations, and quality control methods and measurements.

Problem Definition

The selection process began with a definition of bond requirements, which were determined from:

- Physical requirements for a functional pen, such as pen geometry, pen material set, and alignment tolerances necessary for proper pen operation.
- Desired life and environment of product, including the handling loads the bond might see and the environment the bond must survive during the product's life.
- Production requirements, such as tacking time, cure time, open time, shelf life margins, and safety requirements.

Requirements were easily quantified in some cases. In other cases, we relied on engineering judgment. For example, handling loads that the bond might see during the life of the product were determined by assumptions regarding the worst-case loading that the product might be subjected to by its users. Along with the list of requirements was a long list of desires that would simplify assembly or somehow reduce cost.

Vendors were then queried as to which adhesives they would recommend to satisfy requirements. Many vendors generously supplied samples of any product they felt had the potential to perform acceptably. As with all design processes, flexibility was important. Requirements were in a constant state of change. Priorities often needed to be reordered, which meant that previous adhesive candidates had to be reevaluated in terms of the new priorities and vendors had to be recontacted with revised requirements.

Sample Preparation

For initial screening, test specimens were used instead of real parts. These had some advantages: they simplified

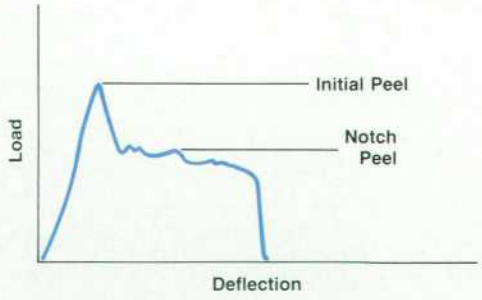


Fig. 7. *Typical data from a peel test.*

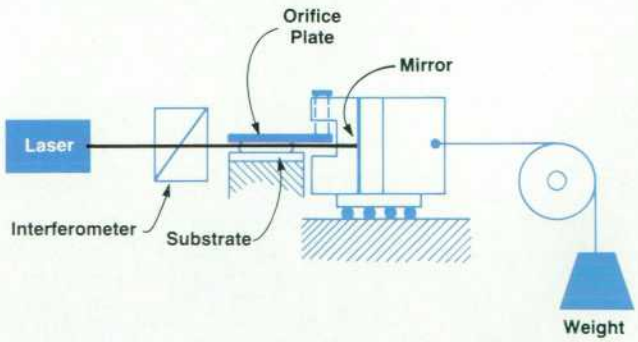


Fig. 8. *Creep shear tester used in the adhesive tests.*

Factory Systems

Information System

Foreseeing the difficult engineering challenges ahead, one of the key manufacturing objectives for the PaintJet printer pen assembly line was to have an information system with the ability to reconstruct the manufacturing environment experienced by any particular pen and to detect trends and problems in the manufacturing process. The manufacturing environment includes per-pen data such as component lot numbers, important process parameters, and quality decision data. Objectives for the information system included data accuracy greater than 97%, minimum operator interaction, easy access to the information by engineers and managers, complete independence from process or material movement control systems, ability to adapt to changing data needs, and convergence with systems used for other HP thermal inkjet lines.

An HP 9000 Model 550 Computer is used as the data base machine and information system controller. At the first step of the assembly line, each pen body is labeled with a high-density code-39 bar code to serve as the pen's identification number. At significant process steps, the bar code is scanned along with process information to record portions of that pen's manufacturing environment. Part carriers are also bar-coded to map component lot numbers to each pen identification number. Reject tracking is handled on-line with a bar-code menu system.

An easy-to-use menu system helps engineers and system users create their own reports. Through the use of Informix 4GL software, users can easily pick specific data elements from the data base and configure it according to their needs. Ad hoc analysis is performed by downloading the data to a local personal computer. The system can also report grouped data to produce standard management reports and graphs.

The information system's capabilities have proven to be extremely important in problem-solving situations as well as for engineering and management decision making. By using the system's ability to track each pen individually, potentially elusive and costly production problems can be identified and analyzed, and a course of action recommended soon after realizing that a problem exists. For example, engineers were faced with the problem of orifice plates peeling off the substrate when the tape was removed. Engineers used the information system to identify exactly when and how those pens were made, what if any unusual events were occurring during the time the pens were made, and what common elements were shared among the reject pens. They were then able to determine that the problem was caused by a malfunctioning orifice plate/substrate aligner. This was reinforced by reject Pareto charts for that time period, and the affected pens were immediately recalled and reworked. The system has proven to be an indispensable time and money saver.

Factory Design

A primary manufacturing objective for the PaintJet pen was to follow the just-in-time (JIT) or demand-pull philosophy in the factory and pen line design. Following the JIT philosophy was not an easy task during the production start-up phase, but the improvements seen in yields and uptime have proven its worth. When problems occur with particular tools, processes, or parts, the entire line is quickly forced to shut down by low work-in-process levels. This approach focuses engineering attention on problem situations, and encourages process improvements instead of temporary fixes.

Several processes in the head assembly section of the PaintJet pen line require contamination, temperature, and humidity control. Therefore, an available clean room was modified to meet these environmental requirements. With the increase in production levels, the clean room became a less-than-optimal environment, and steps are being taken to move the environment sensitive processes out of the clean room and into a normal manufacturing environment. The installation of laminar flow benches for processes that require environmental control was a major step towards this objective. This approach has major advantages such as: high flexibility, improved work environment, leverage to future processes, relatively low cost, and reusable equipment. The cleanliness of the work area and the pen performance are monitored for any changes resulting from environmental control. The plan is to provide as much operator, engineering, and manufacturing freedom as possible within the limits of the process requirements.

Stan Evans
Manufacturing Engineering Manager
Carol Beamer
Mary Ann Beyster
Diane Fisher
Manufacturing Engineers
Diane Armstrong
Systems Administrator
San Diego Division

the design of the strengh testers, they were less costly to fabricate than real samples, they were tangible subtrates of appropriate size and material that could be shown to vendors, and their uniformity could be controlled. They had the disadvantage that any deviation from the actual product implied the introduction of known and unknown variables.

Many extraneous variables sneak into tests as a result of sample fabrication. In an attempt to control them, samples needed to be carefully prepared. Substrate surface preparation for bonding needed to be consistent from sample to sample and accurately reflect the eventual real part preparation. Adhesives needed to be handled carefully. Some adhesives were sensitive to even short open times before cure. Many adhesives were refrigerated to maintain their reaction potential. Physical bond geometry also needed to be maintained from sample to sample. Bond shape, thickness, and area affected the strength test results.

Testing Apparatus

Peel Tests. Bonds can be subjected to peel stresses from user handling and product wiper operation. Since bonds are typically weak in peel, this is a vulnerable failure mode.

A peel tester was used to determine the relative peel strength of adhesive bonds. The silicon substrate, representing the circuit, was clamped to the stationary fixture of the tester. The orifice plate substrate was pulled at 90 degrees from the axis of the silicon substrate at a constant rate of 0.002 inch per second (see Fig. 6).

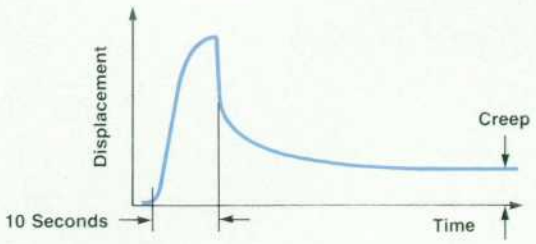


Fig. 9. *Typical creep shear test result.*

The resulting data was a load-versus-deflection curve (Fig. 7). Initial peel strength is the load required to initiate peel. Notch peel strength is the load required to sustain peel. Peel resistance is proportional to the width of the bond.

Creep Shear Tests. Although most systems have little creep in their fully cured state, creep was a consideration for partially cured tack bonds and for testing bonds made using double-sided tape.

Bonds were tested by clamping the silicon subtrate to a static fixture and clamping the orifice plate (or pen body) to a moving fixture which also held a mirror (Fig. 8). One-pound and two-pound weights were applied for ten seconds to simulate handling loads. A laser was used to measure the changing distance to the mirror. Displacement was plotted versus time (Fig. 9).

Peel and creep shear tests were run on bonds subjected to different conditions:

- Initial handling strength tests were performed shortly after cure or on partially cured tack bonds.
- Full-property tests were performed on bonds that had acquired full strength.
- Post-environmental tests were performed after bonds were subjected to fifty thermal cycles between −40°C and 75°C.
- Post-reflux tests were performed on bonds after immersion in ink for 10 or 30 days at 65°C.
- Post-bag tests were performed on bonds that had been placed in an air-tight chamber with ink for 10 or 30 days at 65°C.

Ink resistance was the most difficult criterion to meet. Most bonds subjected to ink degraded significantly. Failure was typically in adhesion to one of the substrates when bond surface sites were displaced by moisture.

Adhesives that best met the requirements included ultraviolet cure tack adhesives and low thermal cure one-part epoxies. The ultraviolet cure mechanism allows infinite open time before cure and tack strength in one second following exposure to UV light. The disadvantage of UV curing adhesives is that they are typically acrylic-based systems with limited long-term ink resistance. They are not employed for structural bonding but for fixturing substrates following alignment. They are required only to survive long-term exposure to ink without coming off and potentially disturbing pen function. Some of the low thermal cure one-part epoxies proved to have high bond strength and good long-term resistance to degradation in ink.

Acknowledgments

The authors would like to thank Winthrop Childers, Beth Heffernan, Steve Nigro, LuAnne Rolley, Chris Snyder, and Bill West for their contributions to the assembly process.

Ink Retention in a Color Thermal Inkjet Pen

Keeping the ink in the pen and off the user is a nontrivial engineering problem.

by Erol Erturk, Brian D. Gragg, Mary E. Haviland, W. Wistar Rhoads, Jim L. Ruder, and Joseph E. Scheffelin

THE DEVELOPMENT OF THE PAINTJET PEN and its assembly processes was recognized as being one of the toughest projects undertaken by HP's San Diego Division. Our objective was to supply our customers with a high-quality pen that could be mass produced. The primary job of the final assembly processes is to get ink into the pen and ensure that it stays there until the pen is used by the customer.

Fig. 1 shows special clear-bodied empty and full PaintJet pens. Inserting the foam (ink reservoir), filling it with ink, and sealing the pen are very complex tasks. Aside from the inherent problems encountered when using a low-tech material in a high-tech application, the foam is required to fit precisely into the pen cavity. Once saturated with ink it must retain the ink over a wide range of environmental conditions throughout the pen's life. The plug must hermetically seal to the pen and allow it to vent to the outside atmosphere. The packaging must provide a physical barrier to ink leakage at the nozzles, and must limit the pressure drop and water loss that the pen may experience during shipping and storage.

Foam Insertion

The process used to insert the foam reservoir into the pen cavity has a major impact on the pen's ability to retain ink. The foam must be compressed on the filter, it must remain in contact with the internal walls and corners of the pen, and it must be inserted to a controlled depth. If any of these three conditions is not met, the pen may leak.

If the foam does not contact the filter, the negative pressure in the pen is lost. Foam compression on the filter prevents the ink leakage that would otherwise result if a column of unsupported ink contacted the filter. This column of ink can include both the ink between the filter and the nozzles and the ink in the void space near the filter. The ink in the space around the filter is eventualy absorbed into the foam as it becomes less saturated during printing. Without the negative pressure provided by the foam, ink leakage will occur.

A second and less obvious benefit of compressing the foam against the filter is improved ink extraction efficiency. A region of locally higher-density foam is created when the foam is compressed on the filter. The resultant increase in density (pores per inch) tends to provide a slightly higher capillary force in that region of foam. This attracts ink from other lower-foam-density regions, which results in less ink left in the foam toward the end of the pen's life. Thus the user gets more printed pages per pen.

Several steps are used to insert foam into the pen cavity. A vibratory bowl feeds the foam to a mechanical compressor station. The compressor station uses a compress-release-compress sequence to compress the foam, which is initially larger than the pen cavity. The compressed foam is then pushed out of the low-friction compressor blocks into a thin-walled square tube. The tube is inserted into the pen cavity, and the foam is extruded into the cavity by pushing from behind as the tube is withdrawn.

Ink Fill

Tightly coupled to foam insertion is the ink fill process. A very dense ink fill is required for two reasons: it allows more ink to be put in the pen, and it minimizes trapped air. Any air trapped in the pen will expand at high temperatures or altitudes, forcing ink out of the foam. Ink not held by the foam can lead to leakage.

Achieving a dense ink fill is particularly challenging. Initially, the foam is hydrophobic, so ink must be forced into it. The fill is accomplished by placing the pen in a vacuum environment and injecting ink using hypodermic needles. Evacuating the pen cavity in the presence of a gas that is more soluble than ambient air in ink tends to minimize the volume of gas left in the foam. Furthermore, degassing the ink before injecting it into the foam decreases the volume of gas initially injected into the foam. The proprietary process used to fill the PaintJet print cartridge with ink requires tight control of evacuation rate, vacuum levels, needle location, ink injection rate, vacuum release rate, and a host of other variables. This process, complemented by a precise and repeatable foam insertion before ink fill, produces the ink delivery system necessary for a drip-free

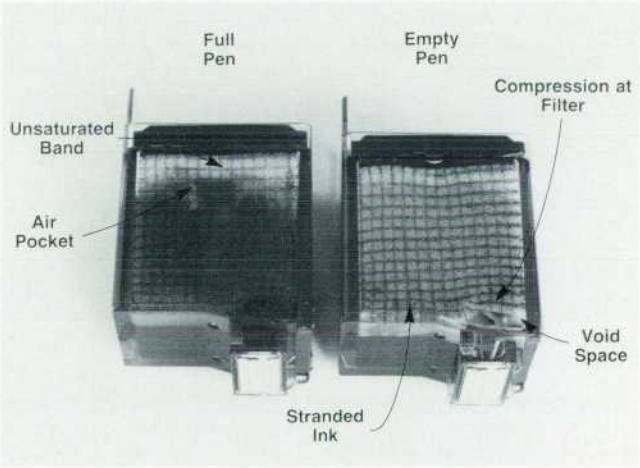


Fig. 1. *Clear-bodied full and empty PaintJet printer pens (print cartridges), showing foam ink reservoir.*

pen.

Foam/Ink Interactions

Fig. 2 shows critical foam insertion and ink fill features and their failure modes.

Foam/ink process interactions were prevalent throughout the development of the ink delivery system. Only after uniform compression of the foam on the filter was achieved could a repeatable ink fill process be developed. As the ink fill process evolved and higher quantities of pens were manufactured, the more subtle problems became evident. Among these were foam tilting in the cavity and foam gaps in the corners, both of which eventually could lead to ink leakage.

Foam tilting occurred during the foam insertion process. The foam was slightly tilted as it was compressed and transferred into the square tube. The foam remained slightly tilted after it was transferred from the tube to the pen cavity. If the foam tilted toward the filter, it resulted in a local high-capillarity region, which improved ink extraction efficiency. If the foam tilted away from the filter, it resulted in the opposite phenomenon. The extreme cases of tilting resulted in gaps along the foam-to-pen-wall interface. The gaps complicated ink fill by allowing ink to bypass the foam and flow randomly on top of the foam and into the tooling as it was injected. Tilting was virtually eliminated after extensive redesign of the foam compression mechanism.

The last major hurdle encountered after foam tilting was eliminated was ink "spouting." Spouting was caused by insufficient contact between the corners of the foam and the internal corners of the pen cavity. Inadequate contact let ink spout along the corner gap during the fill operation and end up on either the ink fill tooling or on the top of the foam (see Fig. 2). Because the ink is rapidly forced into the hydrophobic foam, any areas where the foam does not make intimate contact with the pen walls form low-resistance paths for the ink to travel along. Ink ends up where it wants to, instead of where it should be. The most difficult area in which to achieve good contact during foam insertion is the corners of the cavity.

Spouting can contribute to ink leakage through several different mechanisms. First, an unsupported column of ink remains in the gap in which the spouting initiated. A net positive head can occur if the foam is saturated (as it is early in the pen life) and ink leakage out of either the vent or the nozzles can occur. Second, the spouting ink is deposited on the intentionally dry band of foam located on the plug end of the pen. The vent is not designed to handle large volumes of ink coming into contact with the plug. Therefore, any ink that is not absorbed into the foam may leak through the vent holes. Finally, the low-resistance ink paths along the corners alter the fluid front formed as the pen is filled. This results in dry pockets in the foam. The air in these pockets expands at high temperatures or high altitudes, forcing ink out of the pen.

A foam design change and several insertion process modifications eventually eliminated the spouting issue. The physical dimensions of the foam were changed to increase the forces it exerts on the cavity walls. Although this makes insertion more difficult, it results in more repeatable and robust processes, both at foam insertion and at ink fill. The thin-walled square tube used to insert the foam was refined by modifying corner radii, improving its position relative to the pen cavity, and reducing friction between the tube and the foam. The improved design greatly enhances our

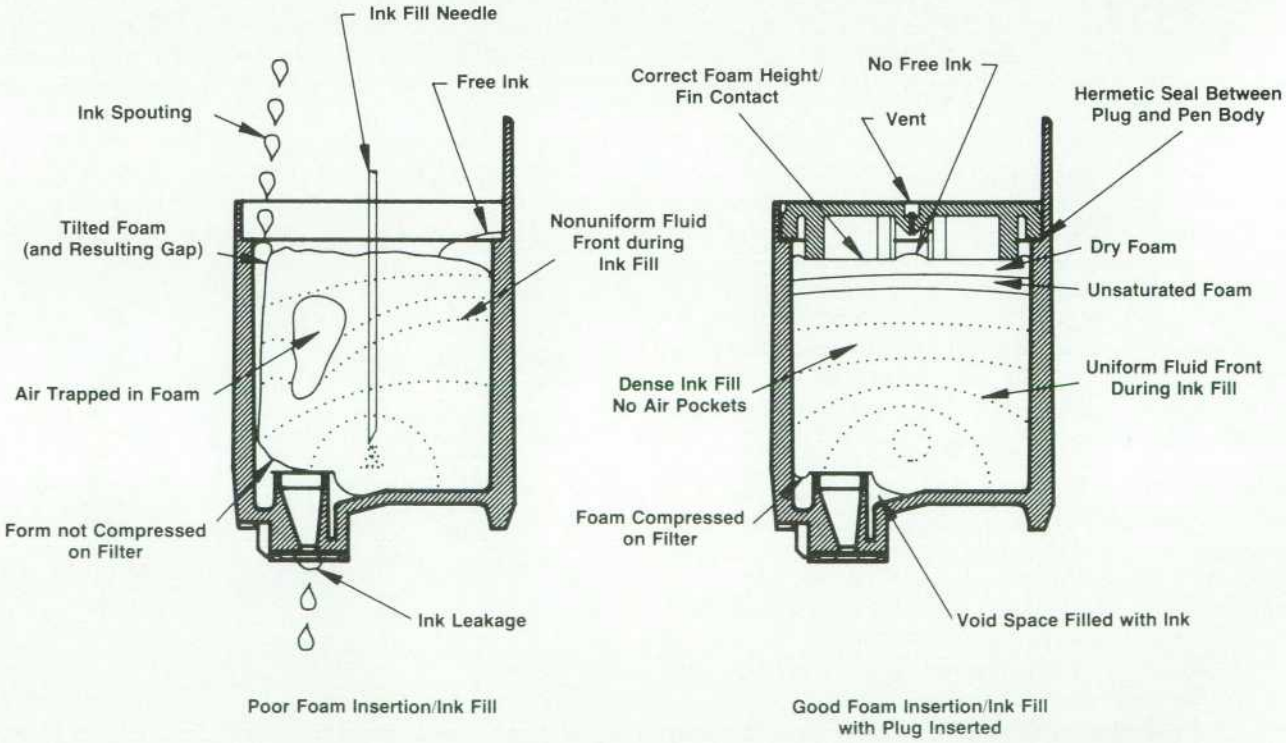


Fig. 2. *Critical foam insertion and ink fill features and their failure modes.*

Activating the Pen

Whether a pen is in a package or being built on the production line, it requires an activation step before it can be used. The activation process must provide a continuous, air-free ink path between the filter and the meniscus in each nozzle. To get ink flowing from the filter to the nozzles, a differential pressure is applied to the pen. This process is called priming.

The priming process consists of injecting air through the vent holes on the back of the pen, thereby moving the ink front through the pen. The purged ink is drawn off from the nozzles. The pressure profile applied to the pen is defined by a pressure-drop/flow-resistance model of the ink path from the filter through the nozzles. The model assumes that pressure drops are inversely proportional to the cross-sectional area of the air/ink interface and are heavily dependent on geometrical constrictions.

When pressurized ink comes out of the foam, it flows through the filter. Until the air/ink interface reaches the nozzles, the cross-sectional area for fluid flow remains relatively large. The pressure required to get ink up to the nozzles is typically around 0.75 psi. The air/ink interface requires much more pressure in the area of the orifice plate and substrate, where channels are small and constrictions are abundant. Since it takes more work to move the air/ink interface through the smaller channels, the pressure required is increased to around 6 psi. Under ideal conditions, first applying 0.75 psi and then applying 6 psi would result in an air-free pen every time.

Actual conditions are less than ideal because of imperfections of the parts and capillary forces that cause air bubbles to be trapped in the ink path. If the bubbles are large they will most likely be around the filter area, and if they are small they are most likely to be under the orifice plate. Large bubbles are dislodged from the walls by mechanical shocking and are flushed to the substrate area. The bubbles then get forced through smaller channels where their leading edges take on the curvature defined by that geometry. It takes work to deform a bubble and force it through the constriction against capillary resistance. Going through such a constriction, a bubble's length-to-diameter ratio becomes unstable and the bubble breaks into many smaller bubbles. To purge out the smaller bubbles, a high pressure differential is applied to the pen to overcome capillary resistances and geometrical constrictions. The pressure is limited by how quickly the foam can replenish the purged ink and the requirement that no air go through the filter.

For the reasons mentioned above, the priming process that gives the best results is one where low pressure is applied first, followed by higher pressure, with mechanical shocks applied throughout the process. The optimal low and high pressure levels and durations were defined empirically.

The PaintJet printer prime station used by the customer has only a single pressure source. It is designed to solve the most common failure mode, small bubbles under the orifice plate. Since the prime station cannot clear all bubble failures, it is necessary to ensure that all pens are virtually free of bubbles before they are shipped. This is done by dedicated production machinery capable of delivering both low and high pressures and mechanical shocks, and by carefully handling the pen until it is taped. The tape keeps the pen primed by maintaining an airtight seal until the pen is ready for use.

Erol Erturk
Manufacturing Engineer
San Diego Division

ability to inject ink to a known location and keep it there.

The final requirements of the foam insertion and ink fill processes for a drip-free pen are to control the foam insertion depth and to avoid forcing ink out of the foam as the plug is attached. During the ultrasonic welding of the plug to the pen cavity, the plug is positioned to a depth at which it contacts the foam. Contact is necessary to prevent the foam from moving away from the filter if the pen is mechanically shocked. Foam inserted too deeply into the cavity eliminates the foam-to-plug contact, resulting in possible shock failures or leaky pens.

If the foam is above the desired level in the cavity, the plug overcompresses the foam as it is welded. This can force ink out of the pen or cause localized ink stranding. The free ink may move to locations within the cavity that can cause leakage. Although the foam depth is tightly controlled at the insertion process, additional margin is obtained by providing a band of unsaturated foam on the plug end of the pen. Besides eliminating the possibility of free ink after the plug is attached, this band of foam provides an overflow safety zone for any ink that is displaced at extreme environmental conditions by small amounts of gas that remain trapped in the ink reservoir.

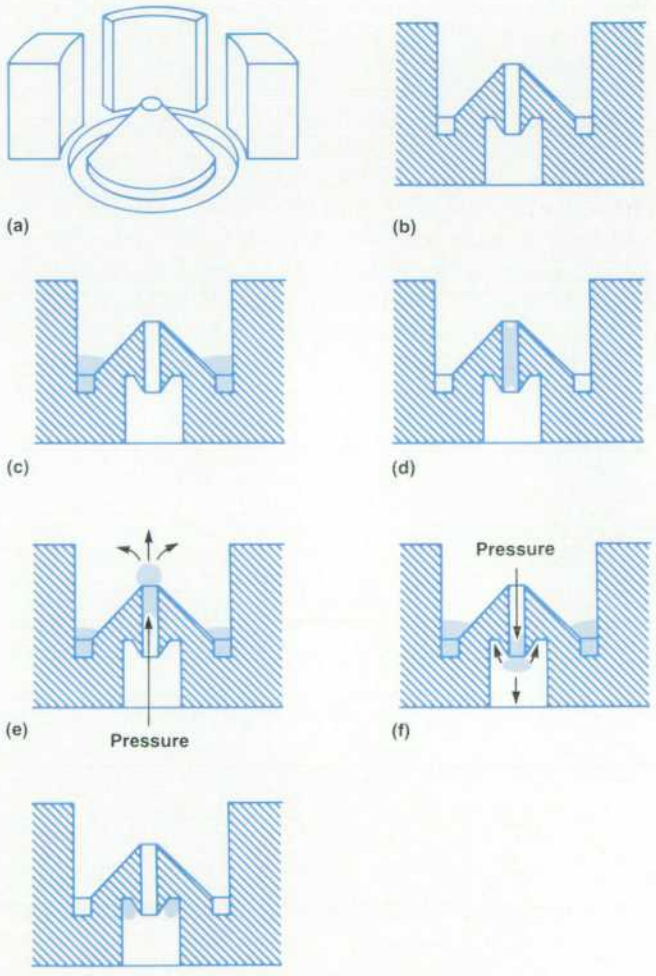


Fig. 3. *The molded plastic plug is designed to seal the pen and allow it to vent without losing ink.*

Plug and Vent

The molded plastic plug is also critical to keeping ink in the pen. It must hold the foam in place against the filters, hermetically seal the pen, seal between chambers on a color cartridge, and allow the pen to vent without allowing ink to escape.

Fins are designed into the plug to press against the foam. This keeps the foam away from the vent, which is critical to vent performance and keeps the foam in place even during mechanical shocks.

A hermetic seal between the plug and the pen body is essential to both retaining ink and priming the pen. Ultrasonic welding was chosen from several bonding processes for this application. The welding process proved to be dependent on many things, including the mass and geometry of the plug and the foam height. Although this interdependence of parts made process development a real challenge, welding the plug has proven itself extremely reliable.

The most challenging design aspect of the plug was the vent. Bidirectional air flow is required for pressure equalization as ink is removed through printing or if environmental conditions change, but the ink must stay inside the pen. The design met with serious stumbling blocks until breakthroughs in foam insertion and ink filling guaranteed only small amounts of free ink, if any, around the plug under worst-case conditions. This allowed a vent design that can be molded as part of the plug. The vent begins with a volcano-shaped projection into the pen (Figs. 3a and 3b), which allows the pen to breathe as long as the ink level does not cover the peak in any pen orientation. If ink gets between the volcano and the walls, the geometry draws the ink toward the base of the volcano regardless of the orientation of the plug (Fig. 3c).

At the base of the volcano and around the inside of the plug are small channels, which draw in the ink and spread it throughout the channel system. The capillary force is strong enough to hold this ink independent of pen orientation. In this manner the capillary system can keep ink away from the vent regardless of pen orientation.

If ink manages to get into the vent, a second capillary system takes over. The tube's small size makes it a capillary trap. Once ink gets into the tube (Fig. 3d) no more ink can enter. The capillary force is large enough to keep more ink from entering the vent, since the new ink would have to eject the ink currently in the tube. However, the force is small enough that during normal operation, as ink is depleted during printing, the vent will clear itself (see Fig. 3e) without the pen's exhibiting any print quality problems. On the other hand, if a positive pressure out of the pen is experienced because of an increase in temperature or altitude, the small amount of ink in the tube clears as shown in Fig. 3f. The small outer volcano shape acts like the large volcano shape by drawing the ink to its base, where the ink will dry (Fig. 3g).

This system, all part of the one-piece injection molded plastic plug, keeps ink in the pen and allows air to flow freely through the vent.

Pen Packaging

The pen package provides two additional barriers to leakage. Tape provides a physical barrier to nozzle ink leakage. A sealed canister provides a physical barrier to ink evaporative loss and isolates the pen from pressure changes at high altitude. The requirements are no visible free ink and <1% evaporative ink loss over 1.5 years.

Pressure and temperature extremes can cause ink leakage by disturbing the force between the foam and the nozzle capillary pressures. At high temperature, ink viscosity decreases and surface wettability increases, allowing ink to flow more easily. Low pressure (high altitude) and high temperature cause air bubbles to expand. These two factors tip the balance in favor of ink leakage, not retention.

The tape consists of adhesive on a polyester carrier. The nozzle plate is wiped with deionized water to remove ink residue. Then the tape is pressed onto the nozzle plate. The process is done quickly to prevent ink from being wicked through the nozzles before the adhesive produces a seal. After customer removal no adhesive residue remains.

Fig. 4 shows an exploded view of the pen package. The canister consists of a PVC-coated aluminum deep-drawn flanged cup and a peelable PVC-coated aluminum lid, heat-sealed together. The seal must be hermetic and survive altitude-induced stresses, yet be easily peeled apart by the customer.

Aluminum is nearly impermeable to water vapor, satisfying the water loss requirement. Unfortunately, only very thin gauges can be deep-drawn or peeled, so the resulting canister is not puncture-resistant. Drop-in plastic liners add the necessary puncture resistance.

The canister works as a pressure vessel at high altitudes by restricting the expansion of the trapped air surrounding the pen. The canister limits this air volume expansion (and thus the pressure drop) to about 5%.

The single most important lesson learned from the thermal inkjet pen development project is that many of the processes involved in this technology are interrelated. The successful development of a drip-free pen could only be achieved by developing all of the assembly processes simultaneously, from foam insertion to packaging.

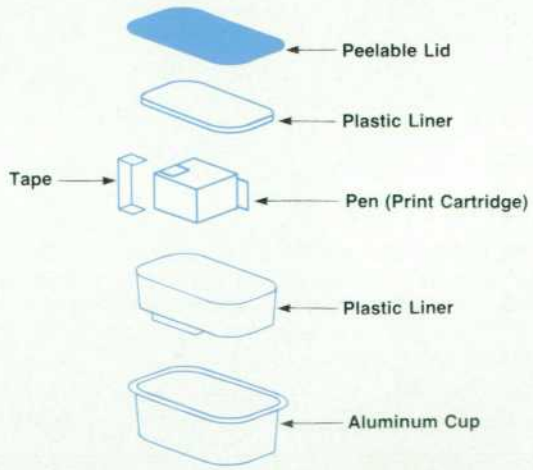


Fig. 4. *Exploded view of the pen package.*

Acknowledgments

Excellent teamwork helped make us successful in designing parts and assembly processes for a mass produced drip-free PaintJet pen. There are many people without whose help the ink would be all over the place. Thanks to Wally Bartz, Carol Beamer, Dan Beamer, Jeff Beemer, Fred Berretta, Dave Burney, Brian Canfield, Steve Card, Bill Colburn, Ken Courian, John Dewez, Jon Fong, Tom Frankie, May Fong Ho, Dave Johnson, Brian Kent, Don La, John Lane, Nick Nicoloff, Richard Peterson, Bruce Reid, Joe Schreiber, Jim Sykora, Curt Torgerson, Doug Watson, Art Wilson, Craig Wright, the production crew, the tool room and model shop crews, all the folks at the Inkjet Components Operation, and everyone else who helped.

Ink and Media Development for the HP PaintJet Printer

The ink, paper, overhead transparency film, and printhead for the HP PaintJet Color Graphics Printer had to be designed as a system because of the complex interactions between these elements.

by Donald J. Palmer, John Stoffel, Ronald J. Selensky, Peter C. Morris, M. Beth Heffernan, and Mark S. Hickman

THE DESIGN OF INK, paper and overhead transparency film for the PaintJet Color Graphics Printer required substantial interaction between the ink, media, print cartridge, and product teams. Throughout the program, design issues were approached from a system perspective so that the best overall performance and reliability could be achieved.

Ink Design

The PaintJet printer combines black, magenta, yellow, and cyan ink drops in a 2×2-pixel cell to generate a palette of 330 different colors. The inks are composed of a solvent carrier (vehicle) and a colorant (dye). The vehicle functions to provide the essential thermodynamic, kinetic, and fluid properties required to generate a superheated vapor bubble and eject a drop. Additionally, the vehicle acts as a carrier for the dye, bonding it onto the surface media (paper or film) with the necessary spot diameter and permanence to meet user needs for print quality.

The vehicle is a combination of water and hydroxylated alkyl ethers and the dyes are organic compounds that have been solubilized in the vehicle using sulfonates and monovalent species such as Li^+ and Na^+, or cationic organic amines.

The inks have many chemical and physical requirements. They must withstand changes in pH, suppress bacterial and fungal growth, resist decomposition, and have chemical compatibility with the material set used in the print cartridge. Additionally, the ink must exhibit little change in its physical properties as a result of evaporation of the vehicle solvent, which can affect viscosity, wettability, and surface tension. This is important to maintain drops that are ejected with controllable volumes, velocities, and shapes.

During the firing of a drop, the layer of ink covering the surface of the heating element can reach a temperature of about 340°C. At this temperature, the decomposition of ink can deposit residue on the surface of the heating element, a process known as kogation. Kogation affects the volume, shape, and velocity of the ejected drop, causing the quality of the printed output to vary. Consequently, it is essential to design an ink that resists such decomposition over the useful life of the print cartridge.

The residue from kogation was found to be largely carbonaceous with varying amounts of dye and inorganic salts included in the carbon matrix. In the PaintJet printer, kogation was eliminated by careful selection of vehicle solvents, dyes, and electrical drive characteristics.

Impurities introduced into the ink from the vehicle components, dyes, or other additives can also cause kogation. Therefore, all of the constituents used in the PaintJet ink are purified.

Prevention of Clogging

Traditionally, clogging of the pen has been the biggest complaint of the the inkjet printer user. With closer inspection, the types of clogging can be divided into two categories. First, a soft plug can occur, causing the initial droplets to be missing. Second, a hard plug can form, requiring the customer to clear an obstructed nozzle.

The root of the clogging problem lies in the exposure of the ink to air. In a roller-ball or felt-tip pen, the ink is protected against exposure to air by a tight cap. A tight cap could not be used in the PaintJet printer because of the position of the interconnect and the potential problem of forcing air into the nozzle. Therefore, the ink is exposed to air and the water component of the vehicle does evaporate. Consequently, the concentration of dye and cosolvent (nonvolatile component of the vehicle) can increase dramatically at the ink/air interface.

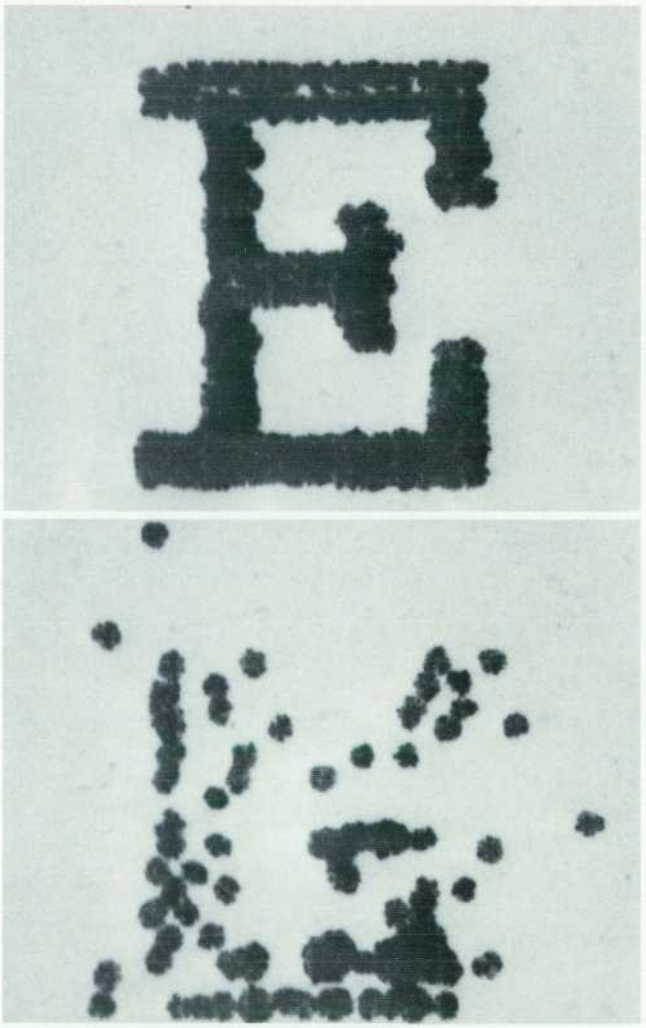


Fig. 1. *Misinformed character (bottom) resulting from soft plug formation. The top photo shows the character as it should look.*

This rise in dye and cosolvent concentration causes the customer to experience clogging by two different mechanisms. First, the viscosity increase in the orifice, caused by increased dye and cosolvent concentration, can prevent the ink droplet from being ejected. This soft plug can usually be cleared by repeated firing of the pen, but the initial characters printed would be misformed or missing (Fig 1). This condition is especially prevalent under cold and low-humidity conditions. Evaporation of water from the orifice can also cause a hard plug to develop at the ink/air interface and prevent the ink droplet from being ejected. This hard plug (crusting) forms as the dye crystallizes out of solution because of the increased dye concentration and the change in the vehicle composition (less water and more cosolvent) at the ink/air interface (see Figs. 2 and 3).

In the development of the PaintJet inks, a hard plugging problem was experienced with the black ink. Lowering the dye concentration would solve this problem, but the printed image would suffer a loss in optical density. Increasing the initial cosolvent concentration would reduce the increase in dye concentration near the orifice, but would increase the viscosity of the ink, which could actually increase the amount of viscous plugging, force a design change in the pen, or create problems in media development. Still another solution to the crusting problem would be to increase the solubility of the black dye. Increased solubility would keep the dye in solution as its concentration increased as a result of water evaporation.

Alternate cations were used in the hope of increasing the solubility of the dye. Increasing solubility (or the solubility product constant K_{sp}) by exchanging cations looked promising because the solubility product for the black dye is proportional to the fourth power of the cation concentration:

$$K_{sp} \propto [\text{dye}][\text{cation}]^4$$

Thus, the K_{sp} (and therefore the solubility) can be increased greatly by replacing one cation (sodium) with another (lithium, potassium, TMA, etc.).

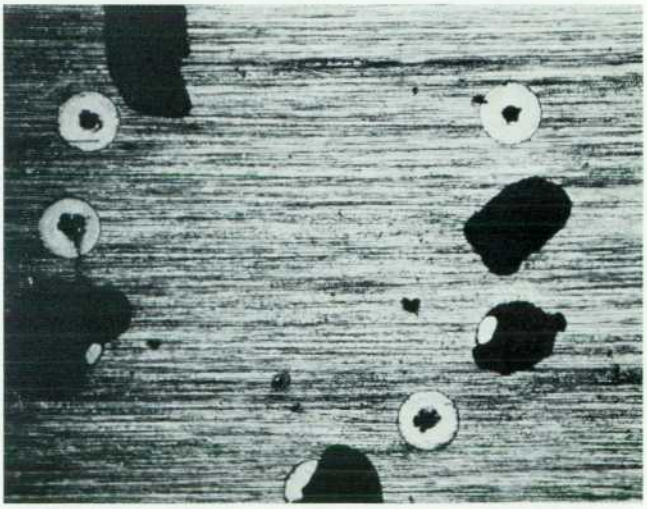


Fig. 2. *Hard plug formation (crusting) on an orifice plate.*

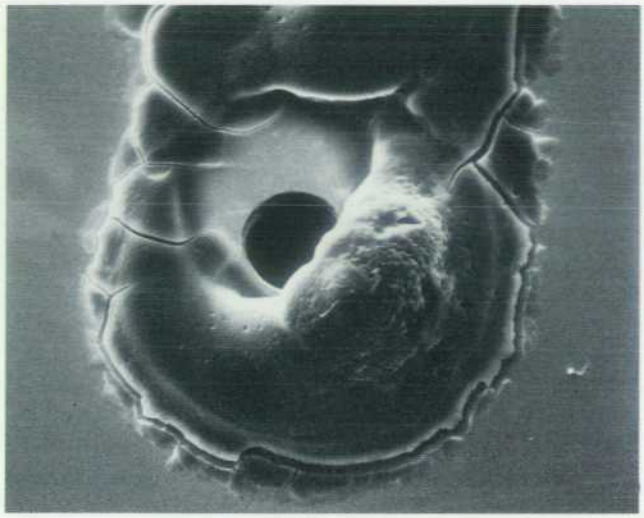


Fig. 3. *Scanning electron micrograph of crusting on the orifice plate.*

Selecting a cation that would increase solubility of the dye in the ink vehicle followed two different theories. First, it was thought that increasing the solvation of the cation would increase the stability of the dye in solution (make it more favorable energetically). The second pathway to increased solubility involved increasing the size of the cation. It was thought that increasing the the size of the cation would destabilize the crystal structure of the dye. Thus, increasing the solubility could be obtained by two methods: increasing the stability of the solvated state or decreasing the stability of the crystal structure.

Lithium, potassium, TMA (tetramethylammonium), and other organic cations were then substituted for the sodium cation in the black ink. Results showed that the solubility of the dye increased dramatically when using Li or TMA as the cation for the black dye. Crusting was also eliminated by using these cations. Thus, both methods of increasing solubility seem to work. Increasing the solvation of the cation by using lithium (Table I) and decreasing the crystal stability (TMA) both eliminate crusting in the pen.

Table I

Enthalpy and Free Energy of Hydration

Ion	Enthalpy of Hydration dH (kJ/mole)	Free Energy of Hydration dG (kJ/mole)
Li^+	−544	−506
Na^+	−435	−406
K^+	−352	−330
Rb^+	−326	−310
Cs^+	−293	−276

With the substitution of cations, the rate of viscous plugging at room and low temperature is affected dramatically. The lithium ion decreases the amount of time before the viscous plug forms while the TMA ion increases the amount of time before the viscous plug forms. This increase in the amount of time before the viscous plug forms, combined with the elimination of crusting, makes TMA the most desirable cation for use.

Strong correlation was found between the time it takes for viscous plugs to form and the increase in effective ionic radius due to hydration. The effective radius actually decreases going down the column in group I of the periodic table (see Table II). This decrease in effective radius of the larger ions results from a lesser amount of hydration because of a more dispersed charge.

Table II

Approximate Effective Ionic Radii in Aqueous Solutions at 25°C

Ion	a (Å)
Li^+	6
Na^+	4
K^+	3

In conclusion, the TMA ion reduces both the crusting (hard plugging) and the viscous plugging (soft plugging) of the black ink to such a large extent that the amount of dye in the ink can be increased. Thus, both the start-up performance and the optical density of the black ink are improved.

Paper

The paper designed for the PaintJet printer has a coated surface so that the optimum print quality can be achieved. The coating serves two basic functions. First, it interacts with the ink and localizes the dye to the coating surface. This surface localization provides maximum optical absorption, since the dye is not hidden in the fibrous paper structure. Second, the coating interacts with the vehicle to generate the required spot diameter and shape. The optimum spot diameter is a function of dot resolution and differs between graphics and text printing. Optimizing spot diameter for text requires a knowledge of the desired font designs. Spots that are too small will not sufficiently overlap adjoining spots, resulting in scalloped edges and poor

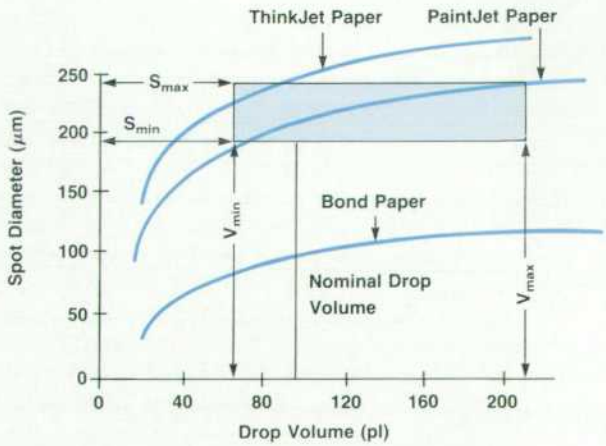


Fig. 4. *Spot diameter as a function of drop volume for the PaintJet printer ink/paper system, with other papers shown for comparison.*

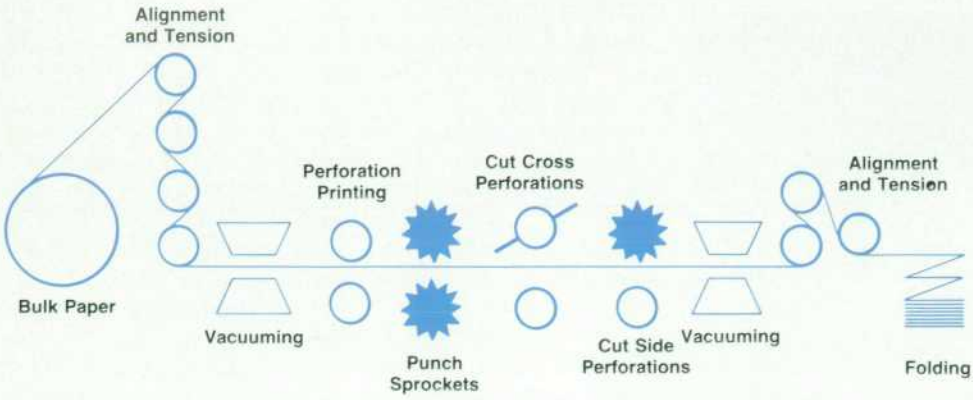


Fig. 5. *PaintJet printer paper conversion press schematic diagram.*

optical density. Spots that are too large will suffer a loss of character resolution and edge acuity. In graphics applications, spots that are too small result in the loss of optical density (because of the admixing of the white paper between spots) and dot placement patterns (banding). Spots that are too large degenerate resolution and increase the minimum line width. A plot of spot diameter versus drop volume serves as a useful tool for developing and characterizing paper (Fig. 4). It is necessary to establish the allowable spot diameter range for both text and graphics applications. The drop volume range is also plotted in Fig. 4; the minimum volume represents the black, magenta, yellow, or cyan drops and the maximum drop volume reflects the double dotting of the subtractive primaries to generate red, blue and green. The minimum and maximum spot diameters and the minimum and maximum drop volumes define the box in Fig. 4. Papers suitable for use with the Paintjet printer must not exceed the spot diameter range over the given drop volume. Hewlett Packard's ThinkJet paper and an office bond paper are shown for comparative purposes.

Converting PaintJet Paper

Fanfolded paper is presently used in a variety of HP inkjet printers. While the printed image is of key importance in the paper design, the ability to produce a final form of the highest quality is also vital.

To produce the fanfolded PaintJet product, bulk roll paper is converted. Paper in very large rolls is slit into smaller rolls that can be used on the converting press. On the press, the smaller rolls are fed and tensioned through a series of stations that vacuum the paper, print key information along the sprocket strips, punch sprocket holes, cut cross perforations, cut side perforations, and perform a final vacuuming before the paper is folded and separated into discrete stacks (see Fig. 5). Proper control of the perforation strength must be maintained to ensure product reliability.

Many factors affect the perforation strength, including coating abrasiveness, web tension on the press, sharpness and alignment of the blades, depth of cut, precise matching of web speed to perforation drum rotation, thickness variation of the paper, nature of the fibers in the base paper, tooth geometry, number of teeth per inch, and the direction the sheet is folded at the end of the press.

PaintJet paper is converted using blades having tooth densities great enough to produce microperforations. Microperforations produce a nearly clean edge when separation occurs; they result from using blades having 40 or more teeth per inch. As the number of teeth increases, the distance between teeth remains constant from one blade to the next but the width of the tooth decreases. Therefore, the cumulative cutting length varies inversely with the number of teeth per inch. The result is an increase in the strength of the perforations with increasing number of teeth (see Table III and Fig. 6).

Table III
Cumulative Cutting Length as a Function of Teeth per Inch

Teeth/ Inch	Distance between Teeth	Tooth Width	Cumulative Cutting Length	Perforation Strength
40	0.006 in	0.019 in	0.760	4.0 lb/in
50	0.006	0.014	0.700	5.4
60	0.006	0.010	0.600	7.0
72	0.006	0.008	0.576	9.5

Many of the factors affecting perforation strength are strictly functions of the press used in the conversion. Other

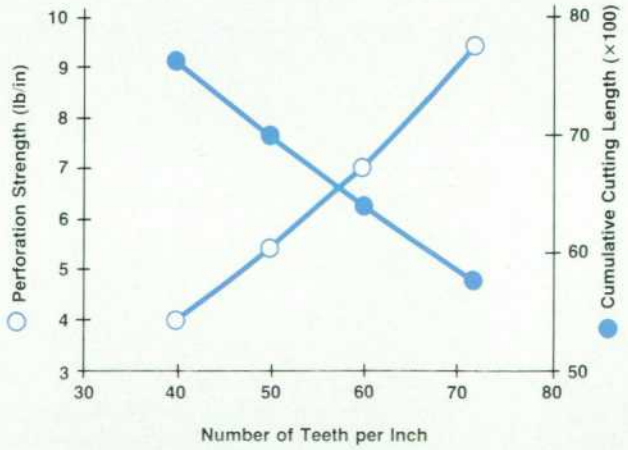


Fig. 6. *Cumulative cutting length decreases and perforation strength increases as the number of teeth per inch increases.*

factors are more easily adjusted and controlled. Blade alignments and tooth geometries are optimized for the thickness and abrasiveness of the coating while penetration of the blade is controlled by correct blade alignment. Figs. 7a and 7c show the tooth geometry of two widely available blade types. The resultant blade impressions in the paper are shown in Figs. 7b and 7d.

The underlying source of perforation strength is the base paper on which the coating is applied. Depending on the depth of cut and the direction of folding, fibers in the base sheet can be completely separated and broken (see Fig. 8). The result is a reduction in perforation strength. Papers with greater fiber resiliency show smaller variation in perforation strength.

Overhead Transparency Film

Overhead transparency film differs from paper. It is constructed from an optically clear polyester substrate, which is nonabsorbent. Therefore it is essential to apply an ink-receptive coating to the film. The same rules regarding print quality and spot diameter that apply to paper also apply to overhead transparency film.

Post-printing "development" of film images is part of the PaintJet solution to providing high-quality inkjet transparencies. Image protection and development control are provided by specially designed protective sleeves supplied with each sheet of film.

We needed to establish a coating or treatment for clear polyester film that would absorb ink quickly and spread the drops by a factor of 3½ until optimal overlap of adjacent dots is achieved. This is required to obtain high image density and bright colors. Handling flexibility was considered crucial. This includes the ability to operate at elevated humidity and to store the film in the protective plastic sleeves typically used with three-ring binders.

Characteristics of paper and film differ greatly. Porosity, and therefore ink absorptivity, can be much greater in paper. Film porosity is restricted because of transparency requirements. The portion of film contributing to ink absorptivity is limited to a smooth surface coating a few micrometers thick. The result of lower porosity is that ink penetration at the film surface is slower than for paper, allowing ink drops to coalesce if touching while on the surface. This coalescence causes images to appear blotchy, or "puddled." To avoid this, initial dot diameters must be less than the dot spacing. The dot size must then grow

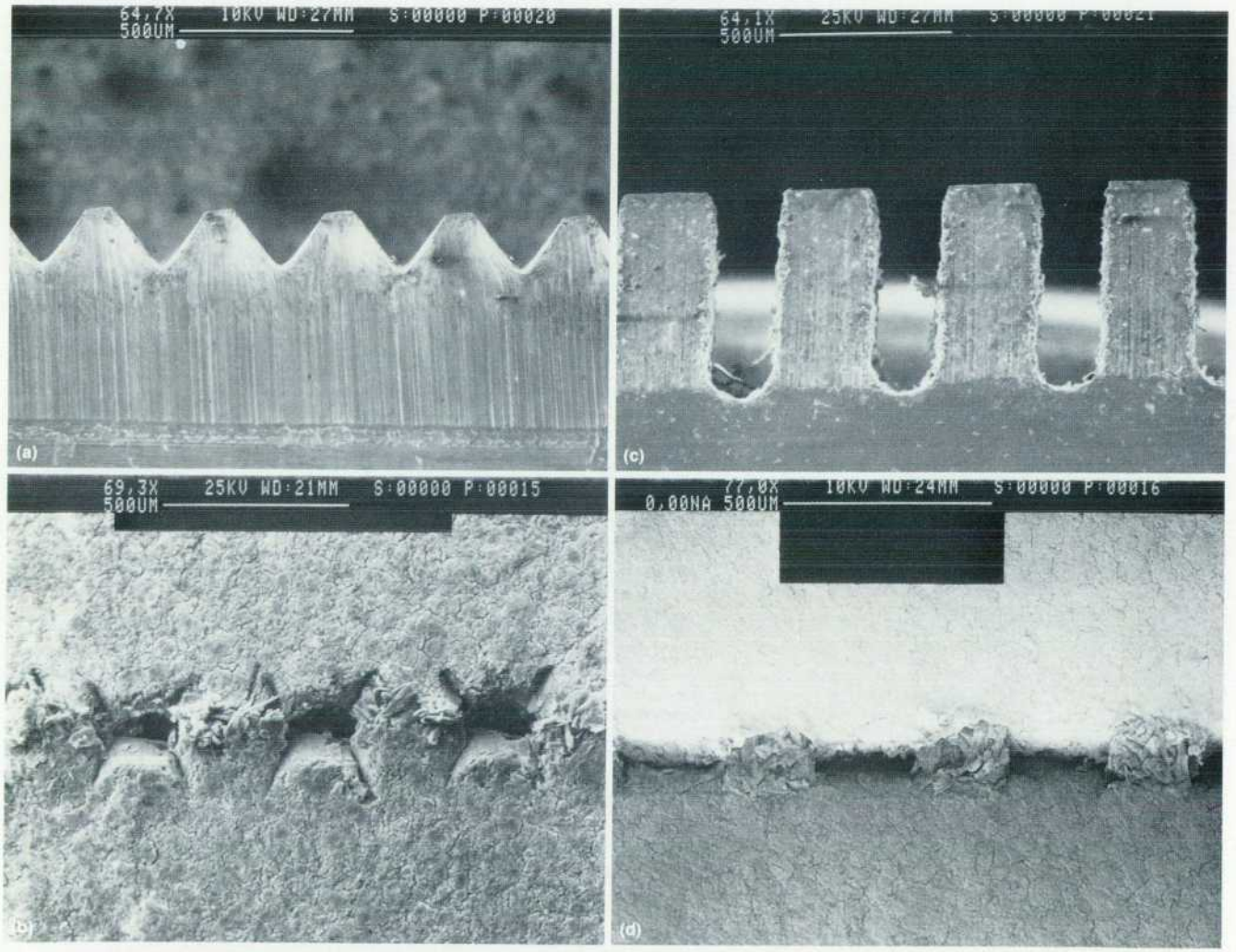


Fig. 7. *Two blade types and the impressions they make in the paper.*

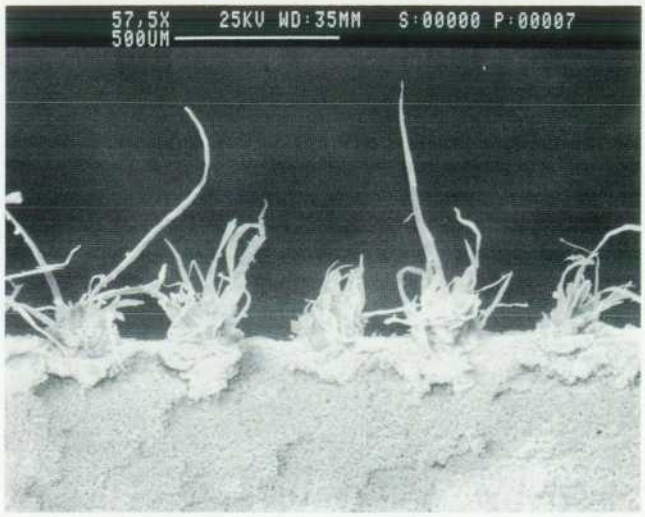


Fig. 8. *Depending on the paper and other factors, fibers in the paper can be separated in the perforation process.*

considerably, after the drops have penetrated the surface of the film, to reach the optimal dot size.

Coating design is further complicated by the fact that hydrophilic coatings, required to accept our water-based ink, have a tendency to absorb environmental moisture and become tacky at high humidity. Fingerprinting, printer feed, and drying speed are all significantly affected by this tendency.

The handling robustness requirement turned out to be our most challenging design constraint. In particular, being able to store freshly imaged film in protective sleeves without damaging the image was a difficult goal to achieve. Evaporation of the ink vehicle from the film is critical to the stabilization of the images created using our post-printing dot-spread mechanism, yet many commercially available sleeves have virtually no water vapor transmission.

To implement the principle of post-printing development, the film coating is matched to the inks to achieve a dye transport mechanism similar to thin-layer chromatography. The dot growth is controlled by evaporation and absorption of the ink vehicle after imaging. Once the ink vehicle concentration in the coating falls below a critical level, the transport of the dyes through the coating is halted.

To use a protective sleeve with the film, yet allow it to dry as if it were in air, a sleeve was designed from a breathable, clear plastic. This plastic absorbs and transmits moisture, and acts as a barrier to environmental changes, leading to more uniform image development across a wider range of conditions than we would get from film dried in air alone.

Because a special sleeve is required for this film, we decided to provide a sleeve with each sheet of film, as part of our commitment to providing the customer with a complete solution. This does not meet our initial aim of being able to store the film in any commercially available protective sleeve, but it has allowed us to provide film that has both high image quality and good handling robustness, which was our primary goal.

Acknowledgments

We gratefully acknowledge the members of the ink, media, pen and product design teams for their many contributions to the development of the PaintJet printer. We would like to acknowledge the work of Ron Askeland and Bill Kappele who were key members of the team working on the crusting problem, Dave Johnson, Karen Brinkman, and Brian Keefe for their work on measuring the start-up properties of the different cations and developing a start-up routine that would enable the ink to work at low temperature, and Adrienne Meuter for helping us to examine the potential problems and test the viability of our changes. Andrea Ongstadt and Irene Kling deserve thanks for the large amount of lab work that was needed in this project. We would like to thank Norm Pawlowski for his cation suggestions, and especially Young Lauw for her work in methods development. Her response to our needs made it possible for us to make the cation changes without delaying the project. Special thanks to Pete Tauriello for his contributions to this article and for the extensive work in making PaintJet paper a product.

Color Thermal Inkjet Printer Electronics

The design objectives were to minimize part count while maximizing cost/performance.

by Jennie L. Hollis, Philip C. Schultz, and William J. Walsh

THE PAINTJET PRINTER PROJECT had to develop a low-cost, high-volume product on an aggressive schedule. It was important that the product have a low part count and be easy to manufacture. For the electronics, we needed a high level of integration, and wanted to select the cheapest parts that could do the job. Flexibility was required, however, to fix the inevitable development and production problems. We also needed to respond to changes in the product as more was learned about customer needs. Such changes could occur at any point before product introduction.

When weighing design trade-offs, these were our main constraints. We would like to show how our constraints affected one particular area of the PaintJet electronics and firmware design, namely, some circuitry that was added to the PaintJet printer's custom chip to alleviate weaknesses in the microprocessor we chose.

Performance Requirements

The PaintJet printer was designed for a very high target production volume and there was a maximum cost target for the electronics. These two constraints suggested a very simple design, a minimal part count, and automated assembly. Since low part count would not only aid design and assembly but also enhance reliability, eliminating parts became a primary focus.

On the performance side, the printhead had to fire at 3 kHz. To achieve 180 dots per inch (dpi) across the printer's eight-inch width, firing forty nozzles (ten nozzles each in cyan, magenta, yellow, and black), 57,600 fire pulses must be sent to the pens each half-second. The commands being sent to the printer need to be converted to fire pulses with little or no delay in carriage motion.

Design Alternatives

Selection of a microprocessor became another focus of the design team. Using a general-purpose microprocessor can vastly reduce the quantity of discrete electronics required. Because of the cost constraint, however, higher-performance processors (such as the Motorola 68000) were eliminated early on.

We knew that the carriage servo would require some custom electronics for modulating the motor-drive signal (see "Low-Cost Servo Design," page 54). We also knew that a low-cost processor couldn't send each fire pulse to the head drivers at exactly the right microsecond. So adding a custom-designed VLSI chip to work with the microprocessor seemed to be the best way to meet our objectives.

The main areas of this custom chip were defined to be: servo motor support, dot-firing control, I/O support, and firmware support. The custom chip was named the Diaper chip.

Firmware Support

One firmware support feature of the Diaper is a three-byte barrel shifter.

The first step in printing is to construct a RAM image of the swath of dots to be fired on the next pass of the carriage. At a resolution of 180 dpi and a width of 8 inches, this image is 1440 dots wide. Text (or black-and-white graphics) uses only the black printhead, which is thirty nozzles high. For color printing, forty nozzles are active, as explained earlier. So the swath buffer is thirty nozzles by 1440 dots per nozzle for black and white, or forty nozzles by 1440 dots per nozzle for color.

To print text at 12 characters per inch (cpi) requires a character bit map 15 dots wide and 30 dots tall. To build

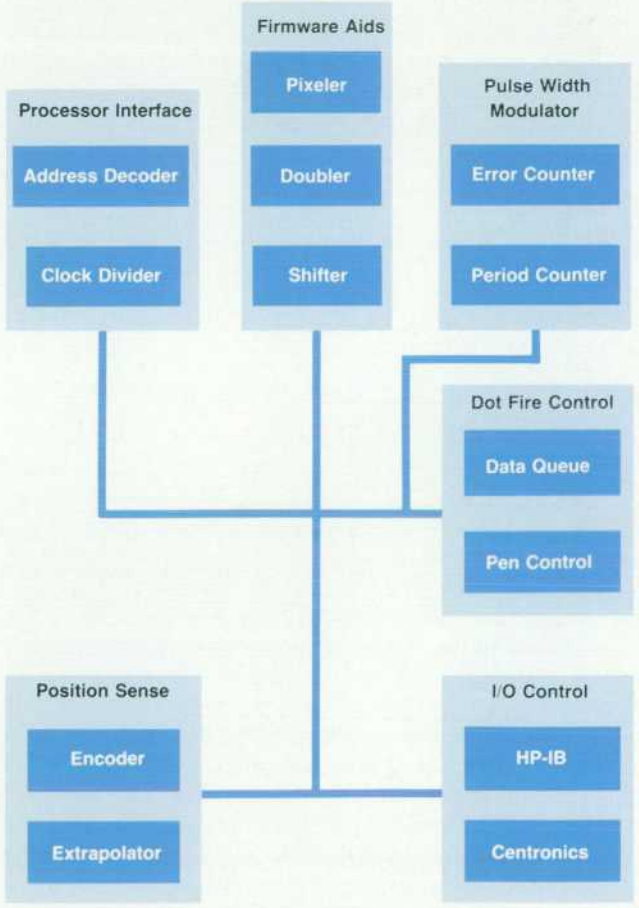


Fig. 1. *Block diagram of the custom CMOS IC in the HP PaintJet Color Graphics Printer.*

up the swath buffer, each of the 30 bit rows of a character needs to be "pasted" to the previous characters in the buffer. Since our RAM is organized in bytes, a bit-map row may have to be shifted before pasting, to maintain byte alignment. The worst case is a seven-dot shift, requiring a three-byte (fifteen dots plus seven dots) shifter.

Only one of the low-cost processors being evaluated had accumulators wider than eight bits. For the others, shifting would have to be done in three parts, with bits passing from one byte to the next through the carry bit. And these bytes would have to be continually swapped in and out of the accumulator. This scheme resulted in unacceptable performance, so a shifter was added to the Diaper. This shifter is also used in preparation for dot firing.

There was a finite cost per gate in the design of the Diaper, and the number of input and output pads was limited, so there had to be some value judgments concerning cost, number of gates, and performance. Based on a generic eight-bit processor, a preliminary interface between the Diaper and the processor was roughed out.

Two firmware tasks were chosen as being probable performance bottlenecks: sending fire data to the thermal inkjet head drivers, and copying a character bit map from ROM to the RAM swath buffer. Assembly code for these tasks was written for each processor under consideration. Standard benchmarks were of little use since the tasks were so specialized, and because the Diaper chip could be counted on to fix specific problems. This code was hand-assembled, and the speed and size of the code were entered into the processor decision matrix along with the increased cost of the Diaper (because of firmware support), interfacing components required, availability of development tools, and vendor reliability.

The Choices

The combination chosen was an Intel 8032 microprocessor, an 8K-byte ROM, 8K bytes of RAM for the swath buffer, and the following circuits in the Diaper:

- A three-byte shifter
- A byte doubler
- A pixeller for 2 × 2 superpixels and RGB-to-black mapping
- Dot-firing support
- Servo motor support.

The servo support circuit is described in the box on page 54. This article will only touch on the dot-firing circuitry and the shifter.

There were two main contenders for the custom chip technology: standard cell and full custom. Full custom designs are more flexible and have a lower per-piece price. But standard cell designs have a shorter lead time. So we did it both ways.

The Diaper chip is a 5000-gate standard cell device. It was used in our prototype development since we couldn't wait for a full custom design. Once the Diaper was debugged, a full custom chip design (called the Spider) was begun. This is our production chip. Both chips are made by HP's Integrated Circuits Division using HP's CMOS-H process technology. Fig. 1 shows a block diagram of the Spider chip.

The Intel 8032 was chosen for several reasons. It was being used in the HP ColorPro plotter (HP 7440A), under development at the same time. Some coding tools existed for the HP 64000 Logic Development System. It was an easy chip to interface to the main board, requiring only +5V and an oscillator/resonator. Our 8032 bottleneck benchmarks were the second fastest in performance. Finally, the chip was the least costly by far, especially when purchased in volume with the ColorPro plotter.

For contrast, another chip in the matrix was the NEC 7809. The 7809 was unsupported on standard code development systems, and it cost twice what the 8032 did. But it won the performance tests hands down, especially before adding the shifter. (It has two accumulators, one of 16 bits and one of 8 bits. Thus, three bytes can be shifted around in two parts with no accumulator-to-RAM swapping.) Adding a shifter to the Diaper diminished the 7809's performance advantage. The aggressive schedule (under two years) precluded developing our own code development tools, and cost was the final straw.

The 8032 is the ROM-less version of the 8052. While our projected code size was under the 8052's 8K-byte limit, we chose to use the 8032, despite adding a ROM to the part count. Why? There was no EPROM version of the 8052, so prototype development would be difficult. We wanted to avoid retooling the processor just to change the code. (Retooling a processor is more expensive and takes longer than changing a ROM or EPROM.) Also, we wanted to use the same part as the ColorPro plotter.

We decided to add just enough functionality to the Diaper to allow the 8032 to run at around 80% of its bandwidth while firing dots at the paper. This allowed 20% for servo and I/O interrupts and future code fixes. While 20% margin may seem large, the servo interrupt was projected to take 5%. To put this in perspective, no San Diego Division plotter servo had ever run at less than 25% bandwidth per axis. And this design was agreed to before a breadboard even existed.

Changes

Over the course of development, the flexibility of the design was tested many times. Because of the lead time and tooling cost of changes to the Spider, its design was frozen about a year and a half into the project. After that, our flexibility was limited to code and main-board changes.

It should also be noted that all code was written in assembly language. No high-level languages were supported at the time we started, and the tight performance margins made size and speed scarce commodities from the beginning.

New Character Fonts

About a year before introduction, some of the character sets were changed from 15-dot-wide elite fonts to 18-dot-wide pica fonts (see Fig. 2). It was mentioned earlier that the Spider's shifter is only three bytes wide. Shifting an 18-dot character seven bits would favor a 25-bit shifter. But the Spider design was frozen.

In addition, the ROM storage required for pica character bit maps is 50% higher than for elite fonts, unless the characters are packed. Of course, unpacking slows the performance while printing.

The compromise adopted was to pack only the bit rows that exceed 15 dots, to shift piecewise only if the character

bit row is 18 dots wide, and to limit downloaded characters to the original 15-dot width. (Increasing the width of downloadable characters would have required more RAM, which cost too much.) The design was flexible enough to allow this change with only small cost and performance changes. The development effort was nontrivial, but firmware was far from the critical path.

ROM and RAM Expansion

The original design assumed code (ROM) requirements of 8K bytes, and 8K of RAM beyond what was internal to the 8032. It was decided that performance could be increased by having an 8K I/O buffer in addition to the 8K swath buffer. Particularly for full-page graphics, adding this buffer could dramatically increase throughput. This RAM is also used for implementing downloadable characters, which was not on the original feature list.

As development problems arose, we always attempted to fix them with firmware. Code added little or nothing to the final production cost, and didn't require any tooling changes. And as more was learned about customer needs, product features were added. So the final product has a 64K-byte EPROM, with 34K bytes of character sets alone.

Increasing code size beyond 64K bytes would have required a paging scheme because of the 8032's addressing limits. Since the Spider was frozen, this would have added new parts to the main board. So the PaintJet printer's code limit was 64K bytes.

EPROMs come in fixed sizes, and the sizes increase by powers of two. So if you have 9K of code, you jump from an 8K EPROM to a 16K chip. And then 32K, 64K, 128K, and so on.

In general, the code was squeezed to stay within the current EPROM size. Then some "must have" feature would come along and justify the cost of doubling the ROM. The extra room would quickly fill up, since features and fixes could be added at no cost. Then it was time to squeeze again.

Since there was so little performance margin, the slightest addition could visibly affect throughput. Our general philosophy was to make the new feature pay the performance penalty. For example, if downloadable characters are used, the I/O buffer shrinks and the throughput goes down slightly. But if you don't activate downloadable characters, the full 8K buffer and the top speed are available.

Having a flexible design made the expansion of RAM and ROM relatively straightforward. Changes to the Spider were not necessary. Implementing the new features increased code development time, but the time was available, and substantial functionality was added.

Additional I/O Options

In the original design, the only I/O interface to be offered was Centronics. Just before freezing the Spider design, it was decided to add an HP-IB option. This had been anticipated, so only minor changes were required.

The 8032 has a built-in UART (universal asynchronous receiver/transmitter), which is used for RS-232-D input/output. But the UART output pins can be used for other purposes, and we were short of input/output pins. It was felt that the UART might be useful in setting up an internal monitor for debugging in the environmental chambers, where we couldn't put all of our development tools. So we

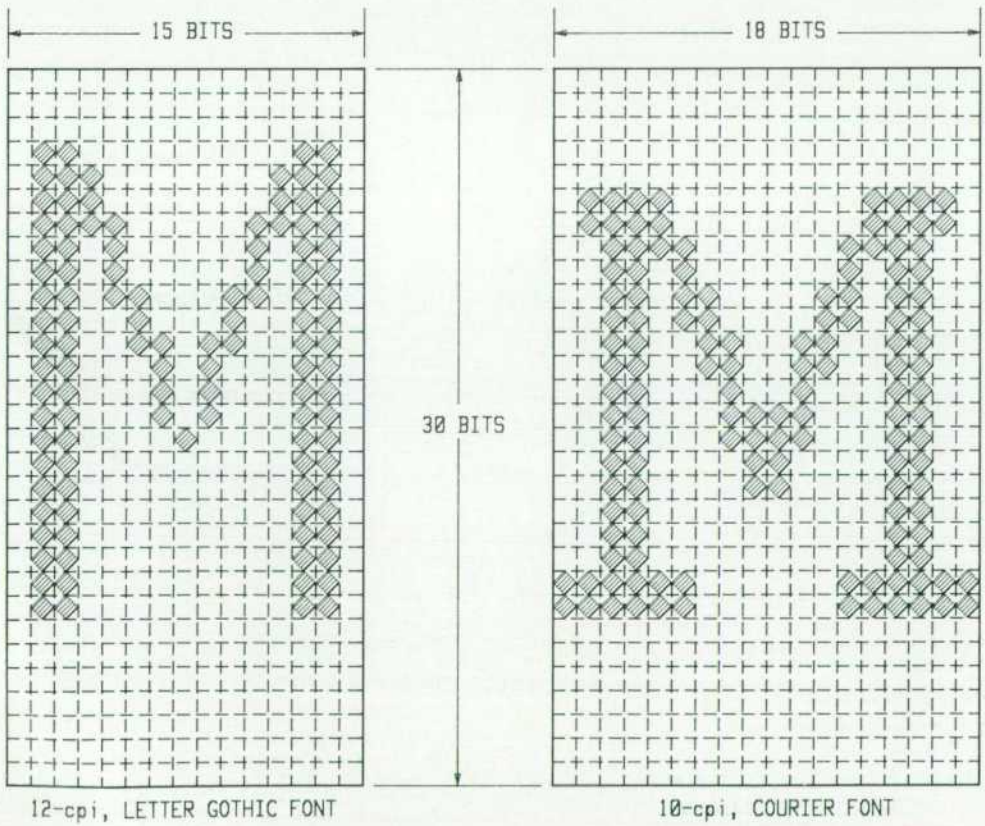


Fig. 2. *During development, some of the fonts changed from 15-dot-wide characters to 18-dot-wide characters.*

Low-Cost Servo Design

As a raster output device, the PaintJet printer differs from its plotter predecessors in a number of ways. One unusual feature is the optical encoder on the carriage axis. This linear, single-channel encoder provides carriage position data that is used both to control the firing of drops from the inkjet print cartridges and as feedback for the servo controlling the carriage motion. Its design meets objectives of high accuracy and low cost, but introduced problems for servo design and development. The encoder and its impact on the PaintJet servo are described here.

The PaintJet encoder is designed to deliver high measurement accuracy and low hardware cost. It consists of an encoder unit, a linear scale, and a special circuit called the extrapolator. The encoder unit is a pair of injection-molded plastic parts which hold an infrared detector, an aperture plate, and an emitter. The scale, shown in Fig. 1, is made of clear polyester film with a photographically produced pattern of opaque bands. The scale mounts to the PaintJet chassis parallel to the carriage axis and passes between the emitter and detector in the encoder unit, which rides on the carriage. Carriage motion is encoded by the detector as a logic signal representing the presence or absence of an opaque band between it and the emitter. Only the falling edges, or light-to-dark transitions, are decoded. The extrapolator is a circuit that operates on the detector output in a way that multiplies the effective scale resolution. It keeps track of time between the last two encoder transitions and uses this value to insert up to three additional transitions following the most recent one. If the carriage speed is held constant the extrapolated data is very close to that of a scale with four times the resolution. This allows the use of a low-resolution scale, which in turn allows the use of lower-performance, less expensive optoelectronic parts. Accurate photographic production of the scale and direct measurement of the carriage position also contribute to low cost and high accuracy.

In servo architecture, the PaintJet printer shows some similarity with its plotter cousins. The carriage is driven by a dc motor via a pulley and a timing belt. The PaintJet printer's custom IC (see accompanying article) extrapolates the encoder data as described above and decodes it into a position word. This is read by the printer's processor, in which the loop is closed and a control word is generated. The processor writes this back to the custom IC where a pulse width modulator converts it to a signal controlling a motor driver IC, which forces the dc motor with a voltage. The main difference from plotters is the PaintJet encoder. Its single-channel output limits decoding to simply counting transitions, with no measurement of the direction of motion as in a quadrature encoder. As a result, the position of a velocity sign change is uncertain and a position measurement error results. Potentially, this creates a problem each time the carriage reverses its direction at the end of a sweep. To overcome this, opaque bands wider than the normal scale pattern are added to the encoder scale (see Fig. 1) to serve as absolute position references. These reference bands mark the limits of the printing area and the carriage sweep area, and can be detected in the encoder signal with firmware. The print limit band is also detected by a circuit in the custom IC, which in turn signals the drop firing hardware to begin printing on the next valid encoder transition.

These encoder limitations complicate the servo that controls carriage motion. The primary control objectives are regulation of carriage speed while printing and adequate transient response while reversing after a sweep. While printing, speed is regulated by a position controller with velocity feedback. The control law can be expressed in the form:

$$U_n = K_p(R_n - X_n) - K_v(X_n - X_{n-1})/T,$$

where
U_n = motor voltage at time t_n
R_n = reference position
X_n = measured position
$(X_n - X_{n-1})/T$ = estimated velocity
K_p = position gain
K_v = velocity gain.

A reference speed is set by ramping the reference position at a constant rate. The position loop guarantees a steady-state speed error of zero. This control law is also used in part to reverse carriage velocity by profiling the position reference. The print and sweep limit bands complicate this by introducing errors into the feedback signal. To compensate for this, algorithms in the servo firmware detect the limit bands and open the loop. Additional algorithms adjust control parameters depending on which gap is involved and whether it is the beginning or end of a sweep. The loop is reclosed upon exiting a limit band.

Design of the carriage servo containing these algorithms required explicit solution of the velocity response. Root locus design was used to determine gain values in the above control law that would provide desired response characteristics away from the effects of the limit bands. In and immediately following the limit bands the response is best characterized through explicit solution because of interaction of the open-loop compensations with the control law. A simulator was written to compute the servo's response numerically under these conditions to aid in algorithm design. Implemented in BASIC on an HP 9000 Model 216 Computer, the simulator is built around a dynamic model of the carriage axis. This model is a matrix difference equation forced with voltage and Coulomb friction and having motor current and carriage velocity and position as states. Models of the PaintJet encoder, extrapolator, and pulse width modulator are

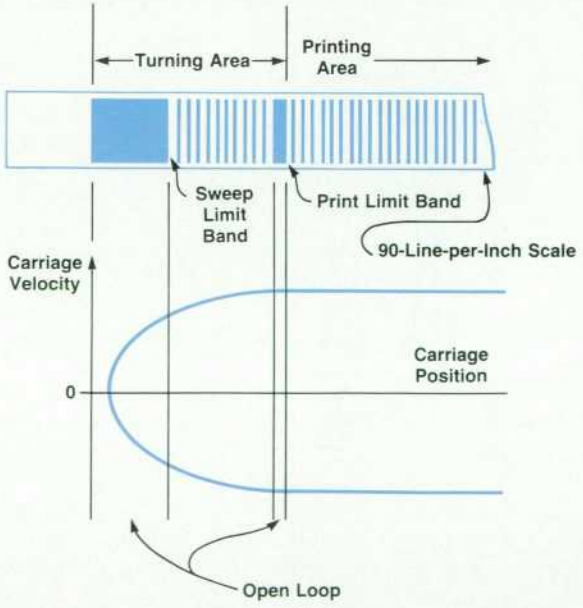


Fig. 1. *PaintJet printer low-cost encoder scale.*

added to this. The result is a block that accepts a digital control word and returns a position word as in the hardware architecture described above. A second block contains a software state machine in which the control law, reference generation, and limit band compensations are implemented. This block reads the position word from the first block and writes a control word back, which is also analogous to the hardware architecture. Confidence in simulated responses is gained by verification of the carriage axis model against measured responses in prototype hardware and by numerical equivalence between servo software in the simulator and firmware in the product. As a design tool the simulator allows tracking of a large number of variables over a wide range of model parameters. This was very helpful in the development and worst-case verification of the carriage servo, in particular the limit band algorithms.

Testing of the carriage servo was also affected by the limitations of the PaintJet encoder. Compensation for the limit bands increased firmware complexity both in the number of algorithms and in the potential for interaction. Some servo failures occurred only after effects had rippled through several algorithms. In these cases, tools providing traceability back to the initial cause were needed. This was obtained with two tools. One is called the servo snapshot. At each servo interrupt it writes values of the servo firmware state pointer, the position error, and the last position change into a circular buffer in the printer's RAM. If a failure occurs, the servo interrupts stop and the data is frozen. It can be read out through the I/O or printed out by the PaintJet printer if the failure was soft, and provides a picture of the servo's behavior leading up to the failure. The second tool is the encoder monitor, which is an external board that times the period of successive falling edges in the encoder signal. The storage of these values into a circular buffer is driven by encoder transitions which stop in a failure. In this case the data can be read and reduced to a plot of carriage speed versus position. Because these two tools sample at different points in the hardware and one is internal while the other is external, they complemented one another very well. They were very useful in determining if a failure was hardware or firmware driven and where the problem started.

In summary, servo hardware cost in the PaintJet printer was lowered at the expense of additional complexity in the servo firmware and the custom IC. The simulator and debug tools were essential ingredients in developing a servo around the PaintJet encoder. The servo has very been reliable in production.

Acknowledgments

We would like to acknowledge the contributions of Bill Walsh, who wrote the servo firmware and servo snapshot, and of Phil Schultz, who built the encoder monitor.

Mark Majette
David Ellement
Development Engineers
San Diego Division

decided to reserve those pins as long as we could.

Eighteen months before introduction, an RS-232-D option was requested. The Spider's design had been frozen long before. Fortunately, we still had the UART pins available, so no change to the Spider was necessary. The design of the RS-232-D main board was straightforward. We were able to keep the three main boards the same, except for one corner dedicated to the I/O connector and support chips. This also allows the use of a single board tester.

The main problem was performance. Reception of data at 19.2 kilobaud while firing the heads would result in the loss of either bytes or dots. Since the processor just didn't have any spare cycles, 19.2 kilobaud was impossible.

Even at 9600 baud, interrupts couldn't be locked out for more than a millisecond or bytes could be lost. This had been one of our main speed boosters. Therefore, a substantial amount of code had to be reworked, in addition to adding the RS-232-D code.

There is only one version of firmware, even though there are multiple main boards. The code checks to see which board it has been plugged into, and acts appropriately. This aids in assembly, purchasing, and version control. Of course, it made pin allocation during the design phase a little more difficult.

Nozzle Heating and Spitting

As mentioned in other articles, nozzle clogs can develop at low temperatures or when a nozzle has been idle for some time. Viscous plugs can form because of evaporation or low-temperature thickening of the dye-carrying agent. Some means of automatically detecting and removing these plugs was desired.

The dot-firing mechanism of the Spider is designed to be flexible. We had no idea what the final parameters of the firing pulse would be. So most of these parameters (pulse on time and off time, for example) are set by the microprocessor. There is a temperature-sensing device on the carriage so we can determine whether the temperature is above or below some threshold value. Adding this capability required only a minor change to the main board.

Much work was done by the pen group to profile nozzle performance at various temperatures, idle times, and ratios of dye to carrier. We wished to incorporate as much of this knowledge into the code as we could. While it would have been nice to log the use of each nozzle, we had neither the

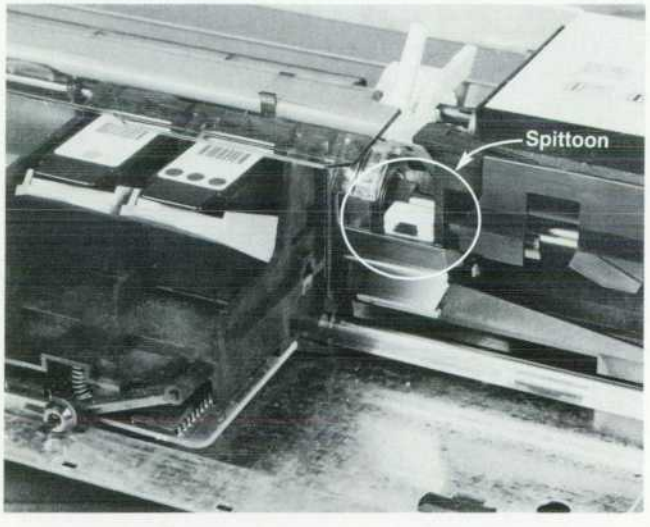


Fig. 3. *The firmware tracks idle time and fires each nozzle into a cube of absorbent pulp called a spittoon to clear viscous plugs before printing begins.*

processor power nor the RAM for keeping track of everything. We had no way of detecting plugs, either. What emerged was a scheme that tracks carriage idle time and temperature, and uses a spittoon.

A cube of absorbent pulp is glued to the chassis near the right sprocket assembly. This cube functions as a spittoon (see Fig. 3). Printing is interrupted every five minutes by firing each nozzle into the spittoon. This prevents viscous plugs from forming. And if the machine is ever idle for over half an hour, the idle time and current temperature are used to determine whether the nozzles need to be heated and how many drops will be required to clear plugged nozzles. (The PaintJet printer has a "soft" power button. As long as the machine is plugged in, the processor is running. Thus, idle time can be tracked, even when the machine is off.)

Pens are heated by defining a firing pulse width so small that it doesn't create enough force to eject a drop from the nozzle. Several thousand of these pulses can be fired in a second, heating the ink in the nozzle considerably. This loosens viscous plugs. Heating can be done without moving the carriage, since the Spider provides a manual fire mode. (Spitting uses the encoder-based fire mode and normal pulse widths.)

This is perhaps the best example of how the design was altered to fix an unanticipated problem. The electronics didn't change, the spittoon itself was cheap, and the ROM required was small. The development and implementation of the antiplug model took a fair amount of development effort, but the schedule was not affected.

In Hindsight ...

What would we do differently now? When designing the Diaper (standard cell), we had no idea how many gates would fit into the Spider (full custom). So anything that could save a gate or two looked good. One gate-saving shortcut was to mix synchronous and asynchronous circuits in the chip. This was a debugging nightmare. Any time a problem showed up, the timing of the circuit in question had to be checked laboriously. Even if it checked out, we were never quite sure we'd thought of all the possible races. Adding logic to the chip to aid design verification would have helped, too.

The main headache, though, was the acknowledge timing for the dot-firing portion of the Spider. As stated earlier, the Spider takes care of firing forty nozzles at a time. So the 8032 prepares a forty-nozzle batch and hands it off to the Spider for firing.

There are three timing constraints in communicating with the Spider's firing circuits. Every nibble of dot data (four dots) must be prepared within 2400 μs for all forty nozzles. (The pixeller works with nibbles, and is used in the preparation of dot data.) Dot interrupts from the Spider must be acknowledged by the 8032 within 300 μs. These interrupts occur every 600 μs, worst case. And dot data must be loaded into the Spider every 600 μs. If any of these windows is missed, some dots won't show up on the paper.

While the 300-μs and 2400-μs windows are tight, we never really bumped into those limits. The dot loading was another matter. Although the servo interrupt only takes 5% of the processor's time, it is a 220-μs interrupt every 5 ms. The I/O interrupt can interfere, since its priority is equal to the dot interrupt's. The worst-case I/O interrupt is the RS-232-D routine, at around 90 μs, and the code to prepare and load the dots needs around 300 μs. These routines were constantly being modified for more speed, resulting in some loss of code elegance. Adding gates to buffer dot pairs (or quads) would have been helpful, since we could have loaded less frequently, but this is a significant number of gates.

The Bottom Line

While parts of the design are not elegant, it meets the objectives we set for it. The final electronics cost is within acceptable limits, the main board is assembled on an automated line, and the performance meets our design goals.

Flexibility/cost trade-offs present some truly challenging choices to an engineer. We hope that we have illustrated one way of tackling such a design.

Acknowledgments

We would like to acknowledge the engineers who either left the project early or came late: David Ellement, Henry Flournoy, Hatem Mostafa, Ed Snow, Jeff Sunamoto, Mick Trego, John Wickeraad, and Art Yasui. And, of course, those who (frequently) set PaintJet's direction: Neal Martini, Jim Smith, and Bob Dey.

HP-RL: An Expert Systems Language

HP-RL is an integrated set of artificial intelligence programming tools that has been used at HP for many types of expert systems experiments.

by Steven T. Rosenberg

FOR SEVERAL YEARS, the expert systems department of HP Laboratories investigated knowledge representation and reasoning techniques in artificial intelligence (AI). One of the ways we approached this was through the construction of an expert systems language called HP-RL (Hewlett-Packard Representation Language). Our goal has been to create a powerful and integrated collection of tools that is usable over a wide range of application domains.

HP-RL is now a mature experiment, and work on it has come to an end. Much of the research involving applications has already been published. This paper complements previous articles describing applications in whose development HP-RL played a part and presents a retrospective look at the HP-RL research effort.

HP-RL has been supported in the past for experimental use by interested projects within HP, and has been distributed to selected participants in HP's U.S.$50,000,000 University Strategic Grants program, where it has been used in teaching graduate students and as an aid in research. Over the years, groups at HP Laboratories and HP divisions have used HP-RL to construct a variety of applications including smart software tools,[1] intelligent instrumentation,[2] and natural language processing.[3] Fig. 1 shows a partial list of the types of HP-RL-based expert systems experiments conducted at HP. We have used the feedback from these experiments to evolve the language to its final state. In addition, through the University Strategic Grants program, we received feedback from external users over a different range of uses. The result of this engineering loop has been the evolution of a robust and powerful expert systems technology that has been applied to a wide range of problem areas of interest to HP. Although HP-RL is a completed experiment, the increased availability of third-party tools and HP Prolog on the HP 9000 Series 300 Computers provides a variety of commercially available tools for those who wish to experiment with expert systems technology.

Expert systems languages such as HP-RL are part of an evolution in computer science to increasingly higher-level programming languages. Expert systems programming is characterized by the attempt to take maximum advantage of the knowledge and problem-solving skills of human experts. Consequently, expert systems programming languages are designed to simplify the task of translating an expert's knowledge into computer-usable form. The goal is to raise the level of description possible in languages such as C or Pascal through the use of various constructs for knowledge representation and reasoning.

HP-RL is an example of a modern high-end, frame-based integrated expert systems tool. Commercial products with similar functionality include ART, KEE, KnowledgeCraft, and GoldWorks. Today, integrated knowledge representation and reasoning tools are the predominant paradigm for expert systems applications. Frame-based solutions to providing this integration have been the most favored and widely used approach. HP-RL is one of many expert systems languages that have emerged in recent years as useful tools for supporting the development of expert systems applications.

HP-RL contains a frame-based component to support knowledge representation, a rule-based component to support reasoning, and a powerful query language. It supports a range of functionality and a variety of programming styles. For example, frames (the basic unit used in knowledge representation) support message passing that is consistent with object-oriented programming in Common Lisp, and the rule-based inference engine supports both backward chaining and forward chaining reasoning paradigms. In addition, a great deal of customization and modularization is possible. The frame-based knowledge representation component can be loaded and used without the rule-based inference engine. The user has the ability to declare away functionality to customize a simpler, more convenient system. On the other hand, the user can also customize a more sophisticated solution. For instance, the user can define an individual algorithm for inheritance to control the use of abstraction hierarchies in the knowledge representation component, and can also define an individual strategy for controlling reasoning.

A part of our philosophy has been that an experienced programmer needs a variety of programming tools, some of which are expert systems tools, while others are more conventional programming tools. Like a craftsman, the programmer needs the flexiblity of choosing which tools to use for a particular task. While HP-RL can function as an expert systems programming shell, we prefer the toolbox metaphor, in which various tools from the expert systems drawer can be readily intermixed with Common Lisp, object-oriented programming in Common Lisp, or conven-

HP-RL Applications

- Smart Instruments
- Fault Diagnosis
- Natural Language Processing
- Software Analysis
- Integrated Circuit Design and Manufacturing

Fig. 1. *Partial list of the types of expert systems experiments based on HP-RL that have been conducted at HP.*

tional programming languages. Historically, most of the applications built using HP-RL have been heterogeneous in nature, so our toolbox approach has proven to be fruitful.

History

Frame representation languages were first developed in the mid-1970s in response to attempts to apply expert systems techniques to areas where knowledge is richly structured. Previous research had often focused on tasks that emphasized reasoning strategies, which in turn used simpler types of knowledge bases. HP-RL is a third-generation expert systems language. It is based on the notion of frames as the fundamental data structure for knowledge representation. Frames were first described in a paper by Marvin Minsky in 1975.[4] HP-RL and other frame-based languages are the result of over a decade of development.

Minsky's theory of frames was first embodied in languages like FRL,[5] a precursor frame representation language developed at MIT in 1977. Two major developments that distinguish FRL from HP-RL are that FRL did not support reasoning, and that FRL was built around a straightforward and simple implementation model. A simple implementation allowed users to experiment readily with the language, modifying features to suit their needs with a minimum of effort. The cost, however, was a relatively inefficient language. In the early 1980s, while working at the Lawrence Berkeley Laboratory, Douglas Lanam and I extended FRL to include simple tools to support reasoning.[6] The earliest version of HP-RL[7] was created at HP Laboratories in 1982-83 from the preceding FRL version. In this implementation, attention was paid to improving efficiency and increasing the power of the reasoning component so that the knowledge representation and reasoning components were of equal levels of utility.

This version of HP-RL was successfully used within HP Laboratories and HP divisions for a variety of projects. Among other examples, it served as the vehicle for prototyping systems known internally as PLATO and AIDA. PLATO[2] is an expert system that interprets low-resolution mass spectra, infrared spectra, and other user-supplied information, and produces a list of the functional groups present in an unknown organic compound. AIDA[8] is a prototype of an intelligent HP 3000 core dump analyzer. This early version of HP-RL was also used to support several of the demonstration applications that were shown at the ninth International Joint Conference on Artificial Intelligence in 1985 to launch HP's AI Workstation product.[9] In addition, it was delivered to several universities. At the University of California at Berkeley, it was used in two master's theses on CAD.[10,11]

In 1984-86 we created the next generation of HP-RL. The language was reimplemented to allow for greater space and run-time efficiency. The syntax was redefined for greater clarity and elegance of expression, and the knowledge representation component was refined and extended, based on feedback from previous applications. The power of the reasoning component was also refined and extended further. Support for the merger of frames and objects was added by providing message passing for frames. The user was given more control over the trade-offs between efficiency and functionality, allowing a range of choices from a rich but slow prototyping environment to a faster, leaner run-time environment. This version has supported the development of the Photolithography Advisor and MicroScope, among other applications. The Photolithography Advisor[12] is an expert system that diagnoses process errors causing defects in the manufacturing of integrated circuits. MicroScope[1] is a knowledge-based tool designed to assist programmers in developing an understanding of the design and structure of large software programs.

An Example

To give the flavor of HP-RL, it is worthwhile to look at an extended example. HP-RL consists of three main components. These are a frame system for representing knowledge, a query language for asking questions, and an inference engine for reasoning over the knowledge base. The following example briefly illustrates some of the features of each of these components to provide a feel for their usefulness. The example shows how software can be described with a frame system, how this description can then be used to generate answers to questions, how questions can be embodied in rules so that they can be reused, and how the use of rules allows us to deduce the answers to more complex questions.

The example is patterned after our intelligent software analysis tool, MicroScope,[1] but is deliberately simplified. While it is illustrative of how MicroScope works, it is not drawn from actual code. A knowledge base of a program's structure and components is created using frames. Fig. 2 shows how information about a typical function is represented as a frame. The frame is a cluster of information about a single software component, in this case, Procedure-1. Various features of the function, such as which variables it sets, which functions call it, which module it belongs to, or whether it is good-quality code, are all combined in the frame. Thus frames provide the basic mechanism for representing and describing code.

Besides describing individual entities, the task of knowledge representation often involves representing relations among entities. This may involve representing organizational or structural relationships. For instance, MicroScope creates descriptions of programs. These descriptions are

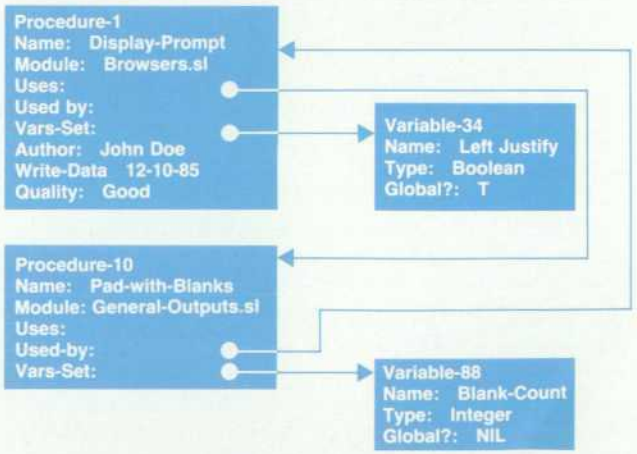


Fig. 2. *A single function, Procedure-1, represented as a frame, with its relationships to other frames.*

About HP-RL

HP-RL is an expert systems construction tool, intended for use on a personal workstation, such as the HP 9000 Series 300 workstations. Expert systems are gaining increasing popularity as a method of automating technical and white-collar skills. They are a means of extending a scarce resource, namely human skill and expertise, by "cloning" it in a computer program—an expert system. Among the major areas for which commercial expert systems applications are being developed are: fault diagnosis, intelligent signal interpretation, computer-aided work, computer-aided design, and financial applications.

Expert systems languages are tools designed to make the construction of expert systems applications easier, through the use of constructs for organizing knowledge and reasoning about it. Typical high-end tools, such as HP-RL, consist of three components (Fig. 1):

- A knowledge representation component for organizing a knowledge base of information
- A reasoning component, which solves problems using the information in the knowledge base
- A query language for accessing information in that knowledge base.

In HP-RL the knowledge representation component is frame-based. A frame is similar to a record in Pascal. It allows the clustering of information common to a concept in a single data structure through the use of a structured description. Frames go beyond simple records by providing advanced features such as abstraction hierarchies and inheritance. In addition, frames can have active procedural components for computing values and achieving side effects. The collection of frames used to represent the knowledge concerning the task we are applying the expert system to is referred to as the knowledge base.

The reasoning component in HP-RL consists of a data base of rules (called the rule base), together with a rule interpreter, which applies these rules to the knowledge base to deduce new facts. Rules in HP-RL are modular if-then statements, which express logical implications (e.g., "If Socrates is a man, then Socrates is mortal."). The rule interpreter uses the rules in conjunction with known facts from the knowledge base, and deduces new facts. The query language is used to extract information from the knowledge base. These queries can also be used to invoke reasoning, causing the deduction of new facts needed to answer questions. A rich query language allows us to ask interesting questions.

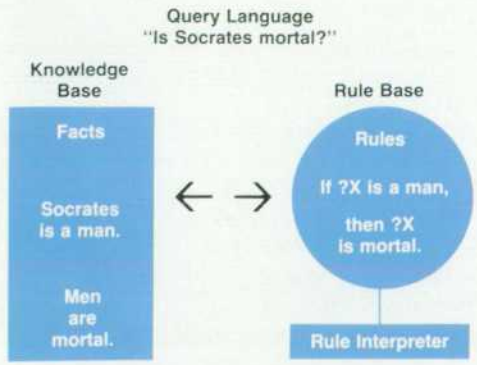


Fig. 1. *HP-RL consists of a knowledge representation component or knowledge base, a reasoning component or rule base, and a query language.*

represented as a set of relations among the frames describing the individual procedures and other components of the program. For example, in Fig. 2, we can also see how Procedure-1 is linked to other functions and to the program variables. Procedure-1 uses Procedure-10. We also record that Procedure-10 is used by Procedure-1. Links also exist to the variables that are set. We exploit the frame descriptions we have created of the individual program components to create a description of the structure of the program.

Frames are organized in an abstraction hierarchy. Abstraction hierarchies are one of the principal means used to achieve economy of description and conceptual clarity in a frame system for knowledge representation. An abstraction hierarchy allows us to represent information common to many entities in a single location. This means that the values associated with frame descriptions are stored at the highest level of generality. These values can then be inherited. For example, Fig. 3 shows a hypothetical abstraction hierarchy for MicroScope concepts. We can see that routines are broken down into types, methods, procedures, and so on. Information common to routines in general is stored on the routine frame. Thus information common to a class or conceptual category is stored just once, on the frame representing that class. Information specific to the various subframes, such as the procedure frame, is stored there. The frame describing Procedure-1 can now be seen as a specific instance of the procedure class, which in turn is a subclass of the routine class.

Through the mechanism of inheritance, values and procedures common to a class are made available to appropriate subentities. Abstraction hierarchies provide an elegant way to share common information. For many applications they provide a very natural way to group knowledge and organize concepts. HP-RL supports powerful extensions to simple inheritance of single values. For instance, HP-RL supports the inheritance of multiple values associated with a feature of a class, as well as multiple inheritance (i.e., an abstraction hierarchy in which a class can have multiple parents). HP-RL also supports higher-order classes, which allow more complex descriptions to be constructed. Higher-order classes allow the user to describe the abstraction hierarchy itself through higher-order classes

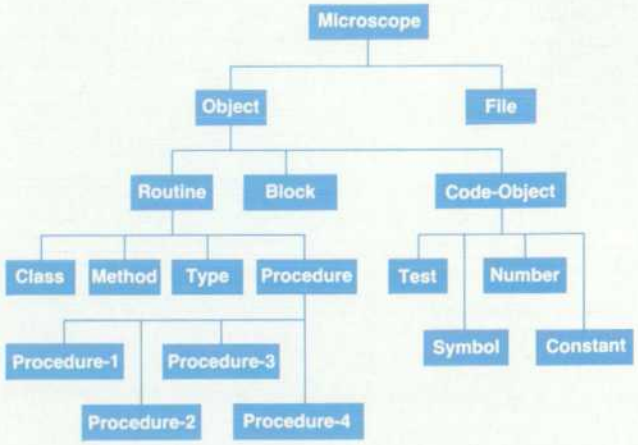


Fig. 3. *A hypothetical abstraction hierarchy for an expert system called MicroScope.*

whose members are the classes of the abstraction hierarchy.

Frame languages contain mechanisms that allow frames to be more than records. For instance, they can contain active values that compute a current value with each request. In particular, HP-RL provides several kinds of procedural attachments for frames, such as methods and different types of daemons. These can be used to do side-effect programming, maintain consistency in the knowledge base, and determine values dynamically. Procedural attachments can also be inherited.

Fig. 4 shows how one kind of attachment, a daemon, can be used to maintain consistency in a MicroScope knowledge base by automatically calculating and recording inverse links. Here we see two procedures, Procedure-12 and Procedure-54. A daemon watches the Uses slot of these procedures. If a value is added to the Uses slot of Procedure-12, namely that Procedure-12 uses Procedure-54, the daemon will recognize this fact. The daemon then finishes the job by calculating that the correct value for Procedure-54's Used-by slot is Procedure-12. This value is automatically added by the daemon. Daemons can be inherited. The daemon illustrated in this example would be stored on the frame defining a routine. It will then be available to all types of routines we have defined, such as methods and procedures, through inheritance. The actual daemon exists in only a single copy, but is made available to all appropriate entities and applied when relevant.

Knowledge representation can be difficult. Depending on the task, it is not always easy to decide on the right organization and description of knowledge. We can think of this task as analogous to designing a data base. Constructing the knowledge base is then similar to depositing data in a data base. However, once this is done, how do we make effective use of our knowledge base? For instance, once we have created descriptions of code in MicroScope, how do we use them? We can, of course, programmatically access the information stored on particular frames. Within an expert systems language such as HP-RL, another way we can make use of our knowledge base is by using a query language. If we think of a knowledge base as analogous to a data base, then a query in HP-RL can be thought of as similar to a data base query.

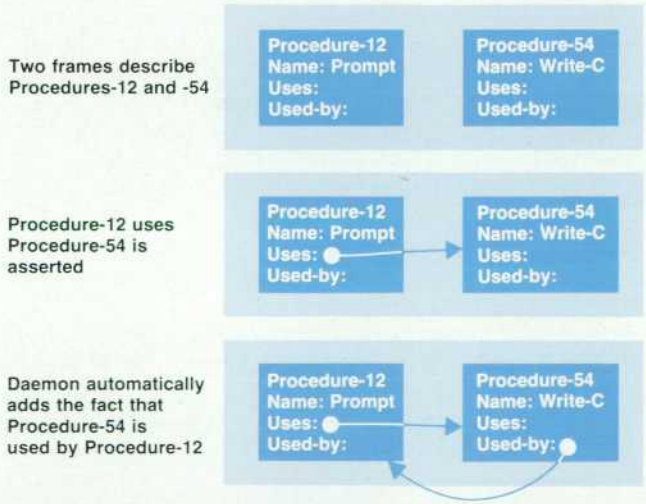


Fig. 4. *A daemon is one kind of attachment for frames. This example shows how a daemon can be used to maintain consistency in a knowledge base.*

```
Knowledge Base Queries

Simple Queries

Find all procedures used by Procedure-1.
  (Solve-All ((Procedure-1) Uses ?X))

Find all Variables whose data type is Vector.
  (Solve-All (?Var Type Vector))

Find all relationships between any procedure and the
variable Var-45.
  (Solve-All (?Procedure ?Relation (Var-45)))

Complex Query

Example Query to Find Dead Code:

Find all internal procedures that are not called by any
other procedures.
  (Solve-All
    (AND
      (NOT (Some ?Caller)
        (?f Called-by ?Caller))
      (?f Exported NIL)))
```

Fig. 5. *Examples of questions that can be asked using the HP-RL query language.*

One of the tasks we will want to accomplish in analyzing the program structure is to ask questions about the program description we have created. HP-RL provides a query language that makes this easy to do. For instance, in Fig. 5 we see examples of some of the simple questions we can ask using the query language. These queries can be embedded in code analysis programs that need this information.

These queries are fairly straightforward. HP-RL also provides a rich set of constructs that allow a user to compose more complex queries. For example, the query in Fig. 5 to find dead code makes use of the logical operators AND and NOT. A query similar to this has been used by MicroScope to identify dead code in an implementation of Portable Standard Lisp.[13] HP-RL also provides procedural escapes so that programmatic computations can be embedded within a query.

A problem with queries, however, is that they must be regenerated each time we need the information. The questions are not reusable. Of course, if we have embedded a query in some code it can be reused, but then its use may be restricted to just the routine that executes that query. We can make queries more useful by turning them into rules. As rules they can be used in answering questions posed by users, by programs, or by other rules. For example, Fig. 6 shows how our earlier query would look as a rule for finding dead code.

A rule is represented as a piece of knowledge in the knowledge base. HP-RL contains a rule interpreter, which understands how to use rules to answer questions. Because rules are stored in the knowledge base, they are not ephemeral, and can be used whenever necessary. This rule is an example of a backward chaining rule. If the premise of the rule is true, namely that "An internal procedure is not called by any other procedure," then we are able to conclude that the "then" portion of the rule, "It is a piece of dead code," is also true.

Individual rules work to determine answers to questions that are not directly answerable by the lookup of values.

Thus our rule answered the question, "Is there dead code?," for which no answer was directly recorded. It did this by asking a question for which the answer could be found, namely, "Are there internal procedures that are not called by any other procedure?" Notice that our rules use variables. This allows us to answer a wide range of questions using the same rule. For instance: "Is procedure X dead?," or "Is there a procedure that is dead?," or "What are all the procedures that are dead?" are all questions that this rule might be invoked to help answer.

Suppose there were many different kinds of bad code. We could have several different rules that each knew one answer to the question. By giving a single query, we could get all the answers, without having to ask all the specific questions embodied in the rules. Fig. 7 illustrates this. Queries are able to invoke rules. Rules are able to work together, in a process called backward chaining, so that one rule asks questions that another rule is able to help answer.

A query can invoke all rules that match its pattern. This is how a single query can cause several rules to provide answers. For example, in Fig. 7, the single query "Is there bad code?" matches against the conclusions of Rules 1, 2, and 4. Individual rules can invoke other rules through the process of backward chaining. This essentially means that if a rule's premise is unknown, then that premise is turned into a query in the hope that other rules are available that can provide the answer. For instance, Rule 2 needs to know whether a bug is minor or not. Rule 3 provides the answer to this question.

HP-RL also provides a host of other features for reasoning. For instance, forward chaining rules add new facts whenever their premises are true. These rules are useful in data-driven programming. For example, rule 5 in Fig. 7 is a forward chaining rule. It will succeed whenever data is added to the knowledge base concerning a code module whose quality is bad. When this case happens, rule 5 prints out the name of the module containing the bad code. This means that whenever rules 1, 2, or 4 succeed and add new information about bad code modules to the knowledge base, rule 5 will print out the name of the module.

The Language

Our goal in developing the final generation of HP-RL was to create an industrial-strength tool for constructing expert systems. The term "industrial-strength" means that we wish to maximize the range of suitable applications while minimizing the cost of developing and running these applications. These goals can be achieved through satisfying a corresponding set of design constraints. We will maximize the range of suitable applications by maximizing the expressive power of the language. We will minimize development costs by maximizing the usability of HP-RL. Finally, we will minimize run-time costs by maximizing the efficiency of the language.

The expressibility of a language determines what it is possible to say in that language, and as a result, what problems can be solved. For example, without the expressive power of arithmetic, problems involving numeric calculations could not be attempted. For an expert systems language, questions of expressive power involve the scope of the knowledge representation and reasoning tools. For example, first-order logic provides considerable expressive power for symbolic reasoning.

Usability refers to how easy the language is to use. Are there good tools for debugging, or browsing knowledge? Is the syntax easy to understand? Can design decisions be easily undone?

Efficiency refers to how convenient it is to build large applications. Do they run fast enough? Can they be delivered in a small, cost-effective machine? We would like an application to be able to run efficiently enough for its intended use.

These goals are to some degree mutually incompatible. That is, maximizing any one goal can result in minimizing the others. For instance, first-order logic is a highly expressive language. It is, however, not highly usable, unless you are an experienced logic programmer. We also know that it is possible in first-order logic to have theorems that take extremely long times to prove. On the other hand, there are very simple syntaxes that are easy to learn and easy to use. Because they are simple, they can be made to run quickly and efficiently. However, they are not very useful because the limited expressive power reduces the range of applications they are relevant to. A good expert systems language strikes a balance between these goals. HP-RL tries to achieve a balance of these goals that provides the best all-around utility.

HP-RL is an integrated set of artificial intelligence pro-

Knowledge Base Inferencing

Example Rule to Infer Dead Code:

```
If an internal procedure is not called by any other
procedure, then conclude that it is a piece of dead code.
  (Define-Backward-Rule Dead-Code
    (Premise
      (AND
        (NOT (Some ?Caller)
          (?f Called-by- ?Caller))
        (?f Exported NIL)))
    (Conclusion
      (?f Attribute Dead-Code)))
```

Fig. 6. *Queries can be turned into rules. Here is how the complex query of Fig. 5 looks as a rule for finding dead code.*

Knowledge Inference Using Rules

"Find all bad quality code."

Rule 1.
If a module's author is John Doe,
then the code quality is bad.

Rule 2.
If a module's number of minor bugs is greater than 5,
then the code quality is bad.

Rule 3.
If the severity level of a bug is less than 3,
then it is a minor bug.

Rule 4.
If a module's number of serious bugs is greater than 2,
then the code quality is bad.

Rule 5.
When the code quality is bad, then print the module name.

Fig. 7. *Rules can work together, resulting in reasoning.*

gramming tools that form an extension of the HP Common Lisp Development Environment,[9] as Fig. 8 shows. Thus the first feature offered by HP-RL is Common Lisp itself. HP-RL allows the user to escape into Common Lisp from a rule or frame, and to call HP-RL from Common Lisp. An application developer can program the procedural components of an application in Common Lisp without being forced to emulate procedural programming constructs within an expert systems shell. HP-RL is similarly integrated with object-oriented programming, allowing frames to receive messages in the same syntax as is used for CommonObjects.[14] This allows the programmer to intermix frame and CommonObject code as needed.

An objective in designing HP-RL has been to use a declarative syntax throughout. Declarative representations are easier to formulate and understand than procedural representations. By using a uniform syntax for both reasoning knowledge and domain knowledge, the expression and understanding of code are simplified. Knowledge becomes easier to examine and understand in this form. It also becomes easier to change. For example, in editing a frame, we merely replace one piece of data with another. In editing a procedure, we must be aware of how arguments are passed and of the flow of control.

Using a declarative representation, uniformity of description is achieved by describing all kinds of information as frames. Economy of description is achieved through inheritance, which allows attributes and processes common to a class of entities to be represented just once, and shared by all relevant instances. Finally, power of description is achieved through attached procedures, constraints, default values, and methods. Many other features exist, of course, and can be found in the manual.[15]

Within HP-RL, frames are used as the single vehicle for representing rules as well as domain knowledge. As a result, rules, which express reasoning knowledge, share in the usefulness of a declarative representation. Rules can be annotated with belief metrics, print strings, or whatever the programmer finds useful. Rules can be related to one another or to other frames. Segmenting rules into different classes can be used to focus search. The space of rules can be accessed via the query language, resulting in a powerful method of extracting information concerning the rules themselves. In short, all the advantages of frames for knowledge representation are available for rules. The syntactic burden is simplified, and the accessibility of rules is enhanced in the development environment.

Fig. 8. *HP-RL is an integrated set of artificial intelligenge programming tools forming an extension of the HP Common Lisp Development Environment.*

The reasoning component provides facilities for both backward chaining (goal-directed reasoning) and forward chaining (opportunistic reasoning). The defined semantics allow interaction between the two modes in which forward chaining can interrupt backward chaining, and vice versa. This is very useful in constructing expert systems that interact with the user, as many diagnostic applications do. Various mechanisms for the control of search are provided. These include hooks for agenda control and mechanisms for doing a type of reasoning called metareasoning. Finally, multiple rule domains allow the partitioning of problems into more compact subsets. To complete the picture, the rule interpreter itself can be represented as a frame, allowing users to examine and change the features of the rule interpretation process. This is useful in metareasoning, and in the control of search during reasoning through agenda control. (Metareasoning allows a program to reason about its own decision processes. Agenda control allows the programmer to control search, in cases where enough is known to improve on the default strategy.) These and other tools for tackling more complex reasoning tasks are described in more detail in the manual.[15]

The Development Environment

HP-RL's development environment is an extension of the Hewlett-Packard Common Lisp Development Environment. In particular, we use the notion of browsing as a way to access information. The environment for using frames provides various mechanisms for browsing frame data structures. Since frames are the single representation device for both knowledge and rules, these knowledge browser capabilities are also available for examining reasoning knowledge. This provides a uniform metaphor for accessing knowledge, and reduces the cognitive burden on the programmer. A uniform set of tools for creating, modifying, and examining frames provides the user with a rich environment of debugging and development tools. These tools allow the user to create, examine, and modify knowledge at an appropriate level of abstraction.

We require of our expert systems tool that it be efficient. Efficient in this case means that it minimizes the effort required to prototype an application, locate defects in a running program, modify the program to correct the defect, and deliver as small and fast a final program as possible. Thus efficiency here means development-time efficiency as well as run-time efficiency. Achieving these goals requires an expert systems development environment that facilitates fast prototyping as well as the development of a fast delivery vehicle. The HP-RL development environment facilitates fast prototyping. Methods are also available in HP-RL that help turn the initial prototype into a smaller, faster run-time program. However, achieving an optimal run-time version of HP-RL is still an open issue.

A specialized environment is necessary to exploit fully the power of HP-RL. HP-RL contains complex tools to represent knowledge and control reasoning. Effective use of these tools can become difficult, for a variety of reasons. As the size of a knowledge or rule base grows, it can become hard to keep track of the interrelations and constraints. Syntactic details can cause problems. Debugging becomes correspondingly more complex as the chains of reasoning

grow. HP-RL's development environment is intended to aid in the rapid prototyping of expert systems software by providing three features.

First, it allows the user to create and modify code at the descriptive level embodied in HP-RL's constructs. Thus a programmer need not be overly concerned with syntactic issues as opposed to semantic ones. Through the use of various templates and menus, a programmer can focus on the task of writing rules and representing knowledge, largely insulated from details of syntax. The use of various browsers allows code to be examined at this level of abstraction, and modified as needed.

The HP-RL user environment also allows the user to follow the execution of a program at this level of abstraction. A variety of tracing and stepping tools can be used to understand program execution without needing to drop down to the level of Common Lisp. To gain further insight into the behavior of a program, the user can examine decision trees, ask for justifications of inferences, or examine the consequences of alternative choices on the program's behavior.

Finally, the environment allows the user to control the execution of a program dynamically, again at this higher level of abstraction. The user can dynamically take control of various rule and goal agendas, and by so doing select execution choices different from those built into the application. This allows the user to experiment with alternatives without having to modify code.

Since all knowledge, including reasoning knowledge, is represented declaratively as frames, at any moment in time an HP-RL program can be considered a structured data base, which can be browsed and queried. As a structured data base, an HP-RL program can be accessed by various kinds of browsers. Browsers provide the unifying metaphor for accessing information in the HP-RL user environment. For example, a frame browser allows the user to browse, filter, and organize collections of frames. A stepper browser allows the user to follow the execution of rules and to change the sequence dynamically. Reasoning is typically annotated with traces and dependencies that the user can browse to understand why reasoning followed the path that it did.

Program developers typically want an environment that supports fast prototyping. Yet, when delivering a program, considerations of speed and size become important. HP-RL tries to give the application developer some control over the trade-off between functionality and efficiency through the use of declarations. These declarations allow a developer to declare away functionality on either a global or local basis. When this is done, speed is improved and space is saved. An application developer will typically start with a fully functional version of HP-RL to ensure maximum flexibility. As an application takes shape and the developer becomes aware of which language features are used in the application, the other features can be successively dropped from the language, creating a smaller, more efficient version for use with the run-time program.

Conclusions

HP-RL is now a mature experiment. It has been a successful vehicle for supporting the development of expert system applications within HP. Along the way, we have been able to extend our understanding of the needs of application builders and gain valuable experience in constructing this class of expert systems tool. There are currently no plans to turn HP-RL into a supported product. HP-RL's role has been as an experimental tool for fast prototyping of expert systems applications.

Once an application has been prototyped, the developer has several choices. The application can be transformed into one built in a more conventional programming language. For instance, both our natural language parsing project and our MicroScope project were able to produce run-time versions where frames are replaced by CommonObjects, resulting in a program that is built entirely in the Common Lisp product supported code. Of course, this transformation may require some care. Conversion to Prolog, C, or Pascal may be possible as well. For example, the Photolithography Advisor[12] has recently been converted to run in HP Prolog.

Often when starting an application, a developer does not know what functionality will be required. This makes the choice of a commercially available expert systems tool difficult. If an inexpensive tool is purchased, it may lack essential features. On the other hand, an expensive tool requires a large initial investment. By prototyping in HP-RL, an application developer can determine what the real needs are, and even whether the project is feasible, before investing large sums of money in software tools. By understanding what functionality the application requires, the developer may be in a better position to determine which, if any, low-end tools can satisfy the needs of the application.

The use of an internal tool such as HP-RL also gives the developer access to the source code. This can be very useful. It is often the case that an application developer wishes to use only a part of HP-RL in an application. For instance, only the knowledge representation component may be required. In some cases, it is necessary to augment or modify the features of HP-RL for an application. For example, PLATO[2] required a special solution for the propagation and combination of negative and positive evidence associated with various possible partial solutions. In other cases, such as the Photolithography Advisor,[12] a handcrafted interface is necessary for success with end users. This requires the ability to access and modify the source code as needed.

For these and other reasons, HP-RL has proved useful as an experimental prototyping tool for the construction of expert system applications within HP. HP-RL is a prototype, and not a product. Thus it lacks some important aspects of commercially available products. HP-RL is an average-size expert systems development tool compared to the commercially available high-end shells. However, HP-RL lacks a sophisticated graphics-based development interface of the type usually available on commercial products. This has made HP-RL harder to use, especially for less experienced programmers, than the comparable commercial products. As a delivery vehicle, HP-RL may be slower than some tools. Little time has been spent optimizing the implementation. In addition, a performance cost was paid for the flexible implementation we chose to use. Many commercial tools now provide C-based (rather than Lisp-based) delivery environments, which may offer

further benefits in terms of size, speed, portability, and maintainability.

Commercial Systems Similar to HP-RL

HP-RL, as a completed lab prototype, is no longer available for general use within HP. However, there are a variety of third-party expert systems tools available on both the HP 9000 Series 300 Computers and the HP Vectra personal computer. The commercial expert systems offerings typically contain reasoning and/or knowledge representation tools embedded within a programming environment that provides a uniform shell for application development. The products can be divided into the high-end shells, which run on the HP 9000 Series 300 machines, and the low-end shells, which run on the Vectra. Some of these products offer functionality similar to that of HP-RL.

Evaluating third-party products is a complicated task. An excellent recent review article[16] provides a framework for assessing commercial expert system tools. The article also contains valuable references to the literature on evaluating expert systems tools.

The high-end shells are suitable for developing large or complex expert system applications. Both Intellicorp's KEE and Inference's ART have features similar to HP-RL's, and will be available on the HP 9000 Series 300 workstations. The PC-based shells offer less power, but are relatively cheap and easy to learn. These shells can be used to construct smaller applications. Since these tools typically have a much smaller and more restricted set of features than the large tools, it is more important that there is a good match between the features of the particular tool and the requirements of the application. However, since these tools are not expensive and can be learned quickly, it is easier to experiment with them. There are numerous vendors of PC-based shells. GURU and Nexpert are two Vectra-based tools that have been used within HP. While no PC-based tool has the full functionality of the HP 9000-based shells, some do provide similar (although simpler) functionality to HP-RL. One that does is Texas Instruments' Personal Consultant Plus.

Another alternative is a product that falls between the large tools and the small tools. This is Gold Hill's Goldworks. Goldworks provides frames for knowledge representation and an inference engine for backward and forward chaining. Goldworks requires an add-on board for the Vectra, but provides significantly more power than most other PC-based tools. Finally, for some users, HP Prolog may provide a a useful alternative. For instance, the Photolithography Advisor has recently been converted from HP-RL to HP Prolog.

Expert systems shells in their present form, while useful, still lack certain features that can restrict their use in some cases. Currently, there may not be an acceptable way to deliver applications to end users. These shells are just beginning to tackle the problem of delivering applications in conventional programming environments. The high-end shells can be large and sometimes slow. Depending on the application, this may present problems. Finally, these shells do not provide access to conventional data in an easy fashion, nor do they provide convenient ways to create large data bases of knowledge, which require the support that a data base management system provides.

Current technology trends are addressing all of these issues. The expert systems shell vendors are developing ways to deliver applications in standard computing environments, and in conventional languages. They are making their shells smaller and more efficient, porting them to C in many cases. Both the shell developers and the data base community are working on solutions to the problem of large, heterogeneous knowledge bases.

The low-end shells may lack functionality that is important in a particular application. Restrictions on the number of rules and the amount of data can similarly restrict their range of applicability, and require care on the user's part in matching the tool to the task. However, the marketplace is changing at a rapid rate. Expert systems technology is becoming more generic, while PC-class machines grow in power. As a result, low-end shells are adopting more of the features and capabilities of high-end shells. This should result in cheaper and more powerful tools in the near future.

Acknowledgments

HP-RL is the product of many peoples' efforts over a number of years. Major contributors were Terry Cline, Jill Dailey, Dell Fields, Wendy Fong, Pierre Huyn, Douglas Lanam, Joachim Laubsch, Reed Letsinger, Mike Lemon, Alan Shepherd, Randy Splitter, Bill Stanton, Ivan Tou, Tom Vrhel, and Steve Weyer.

References

1. J. Ambras, et al, "MicroScope: An Integrated Program Analysis Toolset," *Hewlett-Packard Journal*, this issue, pp. 71-82.
2. B. Curry, "An Expert System for Organic Structure Determination," *Proceedings of the ACS Symposium Series 306*, 1986.
3. D. Flickinger, C. Pollard, and T. Wasow, "Structure-Sharing in Lexical Representation," *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, Chicago, 1985.
4. M. Minsky, "A Framework For Representing Knowledge," in P.H. Winston, Ed., *The Psychology of Computer Vision*, McGraw-Hill, 1975.
5. I.P. Goldstein and R.B. Roberts, "NUDGE: A Knowledge-Based Scheduling Program," *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 1977, pp. 257-263.
6. S. Rosenberg, "An Intelligent Support System for Energy Resources in the United States," *Proceedings of the American Society For Information Science 43rd Annual Meeting*, Anaheim, California, October 1980.
7. S. Rosenberg, "HP-RL: A Language for Building Expert Systems," *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, 1983.
8. L. Slater, K. A. Harrison, and C. M. Myles, "AIDA: An Expert Assistant for Dump Readers," *Hewlett-Packard Journal*, Vol. 37, no. 11, November 1986.
9. M.R. Cagan, "An Introduction to Hewlett-Packard's AI Workstation Technology," *Hewlett-Packard Journal*, Vol. 37, no. 3, March 1986.
10. C. Lob, *RUBICC: A Rule-Based Expert System for VLSI Integrated Circuit Critique*, Master's Thesis, Electronics Research Laboratory, College of Engineering, University of California at Berkeley, Memorandum No. UCB/ERL M84/80, September 1984.
11. M.F. Klein, *Specifying Integrated Circuit Photolithography Processes Using Heuristic and Algorithmic Techniques*, Master's Thesis, Electronics Research Laboratory, College of Engineering, University of California at Berkeley, Memorandum No. UCB/ERL M85/73, September 1985.

12. T. Cline, W. Fong, and S. Rosenberg, "An Expert Advisor for Photolithography," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, 1985.
13. M.L. Griss, E. Benson, and G.Q. Maguire, Jr., "PSL: A Portable Lisp System," *Proceedings of the ACM Symposium on Lisp and Functional Programming*, Pittsburgh, 1982, pp. 88-97.
14. A. Snyder, "CommonObjects: An Overview," *Sigplan Notices*, Vol. 21, no. 10, October 1986, pp. 7-18.
15. R. Splitter, *HP-RL Reference Manual*, Software Technology Laboratory, Hewlett-Packard Laboratories, November 1986.
16. W. B. Gevarter, "The Nature and Evaluation of Commercial Expert System Building Tools," *IEEE Computer*, May 1987, pp. 24-42.

Authors

August 1988

6 Color Print Cartridge

Vyomesh Joshi

As the R&D project manager for thermal inkjet technology developments, Vyomesh Joshi played a key role in the design of the PaintJet print cartridge. His previous HP development projects include a variety of input and hard-copy devices. He is currently R&D section manager responsible for development of the next-generation inkjet technology. He received his MS degree in electrical engineering from Ohio State University in 1980, the same year he joined HP's San Diego Division. Vyomesh's BS degree in instrumentation and control engineering (1975) was earned in his native India. He spent several years at a research institute in Ahmedabad, India, developing process control technology for the textile industry. For three semesters, Vyomesh taught computer organization and microprocessor technology at the California State University at San Diego. He is married and lives in San Diego. In his leisure time, Vyomesh enjoys listening to classical music, traveling, cooking, playing bridge, and playing with his children.

Stephen J. Nigro

A third-generation native of San Diego, California, Steve Nigro joined HP in 1982. He earned his BSME degree from the University of California at Santa Barbara in 1982, and an MSEE degree from Stanford University in 1986. As a development engineer for the PaintJet printer, Steve's interest focused on design of the drop generator used in the print cartridge. Now a project manager, he heads a design team working on the next-generation print cartridge. A patent application is attributable to his efforts. Steve is married and has a son and a daughter, with whom he enjoys spending much of his free time. Other pastimes include lunch-time basketball and golf ("when I get a chance").

Jeffrey P. Baker

At HP's San Diego Division since 1979, Jeffrey Baker has had project responsibilities that included inkjet fluid components, materials, and adhesives. On the PaintJet cartridge, his design efforts concentrated on fluid channels and the ink reservoir, and a patent is pending on one of his ideas. Jeff was involved in the tool design and factory processes for the initial production line. His BS degree in mechanical engineering is from Colorado State University. He was born in Glendale, Missouri. Before joining HP, he was self-employed as a carpenter/contractor. Jeff is married and has two children. He coaches little-league baseball and girls' softball in San Diego, where he lives. He enjoys camping, hiking, and music, and plays tennis and softball.

David A. Johnson

A development engineer at HP's San Diego Division, David Johnson was assigned to the PaintJet printhead and ink reservoir design. He has since begun work on future thermal inkjet printheads. His BS degree in mechanical engineering is from the University of Utah (1982), and his master's degree from Brigham Young University (1984). Dave came to HP in 1984. He is a member of the American Society of Mechanical Engineers and the National Computer Graphics Association. Dave was born in San Jose, California, and now resides with his wife in Escondido, California. Travel ranks first among his avocations, but he also likes backpacking, running, volleyball, playing acoustic guitar, and endurance clog dancing.

16 Color Graphics Printer

James C. Smith

Before becoming project manager on the PaintJet project, Jim Smith held a similar position in the development of HP 7585A and HP 7580/85B Drafting Plotters. A patent is pending for an inkjet temperature control he designed. Jim earned his BSEE degree at the University of Wisconsin in 1978, and his MSEE degree at Stanford University in 1981. He was born in Monroe, Wisconsin, and now resides with his wife and two children in San Marcos, California. His recreational activities include soccer, running, bicycling, and science fiction.

Emil Maghakian

As a project manager for PaintJet software support toolkits, Emil Maghakian drew from his experience with the HP 7310A Graphics Printer and associated software developments. Emil joined HP in 1979, soon after he received his MS degree in computer science from the Virginia Polytechnic Institute. His BS degree (1976) is from the same institution. Before joining HP, Emil was an instructor in computer science at Hollins College in Virginia. His professional interests focus on peripheral languages, software productivity tools, and human interfaces. His present assignment at HP's San Diego Division includes languages, drivers, and toolkits for future products. Born in Tehran, Iran, Emil is active in various Armenian-American organizations and the Boy Scouts of America. He is married, has two children, and resides in Escondido, California. Among his varied avocations are soccer, camping, history, music, and public speaking.

21 Mechanical Design

P. Jeffrey Wield

A design engineer at HP's San Diego Division, Jeff Wield was responsible for mechanical development of the paper axis for the PaintJet printer. Before this assignment, he developed microelectronics processes. His educational background is in both chemistry and mechanical engineering, with a BA degree from the University of California at San Diego (1977) and a master's degree from the San Diego State University (1980). This diverse background is reflected in Jeff's earlier activities as a stable-isotope chemist for the U.S. Geological Survey. A patent application is pending for a method he developed for molding plastic gears. He is a registered professional engineer and resides with his wife in San Diego, California.

Chuong Cam Ta

A project leader at the San Diego Division, Chuong Cam Ta has been involved with the design of printers and plotters since he joined HP in 1979. Among his projects as a design engineer were the HP 7470A, HP 7550A, and HP 3630A Plotters. Presently, Chuong heads the design of a new color graphics printer. He was born in Hanoi, Vietnam, and completed his education in Saigon, where he received his BSME degree from the National Technical Center in 1974. He also holds a BSME degree from the University of Minnesota (1979) and an MSME degree from California State University at San Diego (1984). Before coming to HP, Chuong was a planning engineer at The Nestle Company in Vietnam. He is married and lives with his wife and two children in San Diego, California. His recreational activities include camping, waterskiing, and downhill skiing.

David C. Tribolet

As a project manager at HP's San Diego Division, Dave Tribolet was responsible for the design of the PaintJet mechanism. In the past, he has been involved in the development of other HP plotters as both engineer and project leader. Two patents resulted from his work, one for a switchless pen sensor, the other for a bidirectional pen changer. Dave's BSME degree is from the University of Arizona (1978), an MSME degree (1979) and an MSEE degree (1982) are both from Stanford University. He was born in Tucson, Arizona. Now a resident of San Diego, California, Dave likes to spend his leisure time bicycling.

Lawrence W. Chan

The focus of Lawrence Chan's professional interests is mechanical design. His BS degree (1979) and his master's degree (1980) are in mechanical engineering and are from the Massachusetts Institute of Technology. When he came to HP's San Diego Division in 1985, Larry was assigned to the PaintJet development team, where he worked on the design of the electrical interconnect system. With the project completed, he has now redirected his efforts to the design of a new plotter. Before coming to HP, Larry worked in the paper industry, where he designed a patented test apparatus. He was born in Hong Kong and now resides with his wife in San Diego, California. Larry enjoys traveling and photography.

28 Thermal Inkjet Structure

Winthrop D. Childers

With a bachelor's degree in physics from the Colorado School of Mines (1982) and an MS in physics from the University of California at San Diego (1983), Win Childers joined the San Diego Division of HP in 1983. As a development engineer on the PaintJet project, his responsibilities included the orifice plate, barriers, printhead assembly, and cartridge assembly. In a previous position, Win has worked on materials science research for a large chemical concern. He was born in Princeton, New Jersey, is married, and makes his home in San Diego, California.

Hatem E. Mostafa

An R&D project manager at the San Diego Division of HP, Hatem Mostafa was involved in the development of firmware and electronics used in the PaintJet printer. Since he came to HP in 1979, Hatem has worked on other printer/plotter designs. In particular, he contributed to the electronics for the HP 7580A Drafting Plotter and to the electronics and servo for the HP 7550A Plotter. Hatem earned his BSEE degree at the University of Minnesota in 1979 and holds an MSEE degree (1982) from Stanford University. He has coauthored articles for the HP Journal on some of his previous projects. Hatem was born in Cairo, Egypt, is married, and lives in San Diego, California. Running, scuba diving, and bodysurfing are his favorite pastimes.

Ruben Nevarez

After receiving his BSME degree from the University of California at Santa Barbara, Ruben Nevarez joined the San Diego Division of HP in 1982. As a design engineer on the PaintJet Color Graphics Printer project, Ruben worked on the interconnect system and the primer. His ideas contributed to two PaintJet patent applications. Ruben was born in Durango, Mexico, and is active in the Society of Hispanic Professional Engineers. He is married and lives in San Diego, California. In his off-hours, Ruben enjoys running, soccer, and participating in bible discussion groups.

Ronald A. Askeland

As an R&D project leader, Ron Askeland's specific area of interest in the PaintJet printer was the reliability and failure analysis of the thin-film resistors in the printhead. Ron came to HP in 1984 with a BS degree in biology from Mankato State University (1976), and an MS degree in microbiology (1980) and a PhD degree in chemistry (1983) from Colorado State University. His previous professional activities include work with environmental chemistry at the U.S. Naval Ocean Systems Center. He has published three articles on subjects closest to his interests. Ron is a member of the American Chemical Society. Born in Madelia, Minnesota, he is married and has a daughter. In his off-hours, he likes racing model sailboats, tending his bonsai trees, and playing tennis and golf.

William R. Sperry

Joining HP in 1973 as a production engineer, Bill Sperry has worked on a variety of assignments, but the many performance and manufacturing aspects of printhead design have been central to his career. He has been a production engineer, an R&D engineer, and a project leader. A project manager for the PaintJet printer during the past several years, his responsibilities included the development of thin-film components, the barrier, and the nozzle plate, and the prototype assembly of the printhead. Bill's 1968 BS degree in physics is from California State University at San Diego. His several publications are on the subject of microwave delay line design and the fabrication of TaAl resistors. He is married and lives with his wife and three children in Poway, California. Bill is a member of the National Ski Patrol and coaches youth soccer activities. He enjoys skiing, soccer, and golf.

32 High-Volume Microassembly

Timothy J. Carlin

Developing processes and designs that satisfy both performance and high-volume manufacturing requirements were Tim Carlin's responsibility on the PaintJet project. He is a manufacturing engineering manager at the San Diego Division, which he joined in 1984. Before coming to HP, Tim's professional activities included R&D in energy, fluid modeling, automation, robotics, and machine design. He earned his BS degree in engineering science at the University of California at San Diego (1972). Master's and engineer's degrees in mechanical engineering (1974 and 1975) are from the Massachusetts Institute of Technology. Tim is a member of the American Society of Mechanical Engineers and the Society of Manufacturing Engineers. He's married, has four children, and lives in San Diego, California. He enjoys playing soccer and coaches a youth soccer team.

Steven W. Steinfield

Before joining the PaintJet development team and becoming an R&D project leader, Steve Steinfield was a manufacturing engineer working on the media-moving and autocalibration systems of the HP 7580A and HP 7585A Drafting Plotters. He came to HP in 1981. As part of an engineering rotation program, he is presently designing an automated print quality evaluation system at HP's Inkjet Components Operation in Corvallis, Oregon. Steve's BS degree in applied science is from the University of California at San Diego. Although born in Chelsea, Massachusetts, he considers himself a native of San Diego. Steve is a musician and enjoys playing keyboards, guitars, and drums. In his home recording studio, he is currently producing his second collection of original jazz and rock songs. He also likes bicycling, volleyball, and photography, is a connoisseur of beers from around the world, and enjoys cooking Mexican food.

Cheryl A. Boeller

In the development of the PaintJet printer, Cheryl Boeller's responsibilities included selection of the adhesives and establishing the ultraviolet curing processes used. At present, her professional interests focus on the print quality of various thermal inkjet products. A development engineer at the San Diego Division, Cheryl came to HP in 1980. Her BS degree in mechanical engineering is from California State University at Long Beach.

Peter M. Roessler

In 1984, after receiving his BS degree in mechanical engineering from the Universtiy of California at Santa Barbara, Pete Roessler joined the San Diego Division of HP. As a manufacturing engineer on the PaintJet project, his assignments focused on establishing processes that would ensure both performance requirements and manufacturability of the print cartridge. He has worked as a project leader and was recently promoted to project manager for printhead assembly. Pete was born in Torrance, California, and now lives in Del Dios. He spends his leisure time playing basketball and golf, skiing, and collecting compact discs.

41 Ink Retention

Brian D. Gragg

As a student, Brian Gragg worked at HP during summer recesses, but joined the San Diego Division full-time in 1985, after receiving his master's degree in mechanical engineering from Harvey Mudd College. His BS degree (1984) is from the same institution. A manufacturing development engineer on the PaintJet project, his responsibilities included design of the plug, hermetic seal, and vent, and the ink fill process, foam insertion, and tape process development. His work on the vent resulted in a patent application. Presently, Brian's efforts focus on development of an automated assembly line for the PaintJet pen. He was born in Groton, Connecticut, is married, and lives in San Diego, California. His varied avocations include photography, computers, gardening, bicycling, and flying radio-controlled gliders.

Jim L. Ruder

Before coming to HP on a permanent basis, Jim Ruder had worked two summers at the San Diego Division, while attending Kansas State University. He received his BSME degree in 1983. As a manufacturing engineer, Jim belonged to a team working on design of the PaintJet pen. His responsibilities included the method of foam insertion, and inkfill and priming techniques. Now a manufacturing engineering supervisor, Jim continues to be involved with plotter pen assembly. He is a member of the American Society of Mechanical Engineers. Born in Salina, Kansas, he now resides with his wife Lisa in Escondido, California. Jim's spare time is shared by church activities, landscaping and other home improvement projects, and "enjoying Southern California."

Erol Erturk

The areas of manufacturing process development, manufacturing and design integration, and automation are Erol Erturk's professional specialties. In the development of the PaintJet printhead, he was responsible for the plug, vent, primer, wiper, and tape processes and their manufacturing implementation. His BS degree in mechanical engineering is from the University of Wisconsin (1985), and he is planning to begin a master's degree program in fall. Erol was born in Ankara, Turkey, and joined the San Diego Division of HP in 1985. He is a member of the American Society of Mechanical Engineers and the Society of Manufacturing Engineers. His favorite sport is soccer.

Mary E. Haviland

As a manufacturing engineering manager, Mary Haviland headed a team working on the PaintJet pen assembly. She has since directed her efforts toward processes for materials and media for the printer. Before coming to the San Diego Division of HP in 1979, Mary's work included developing measurement techniques for materials used in Titan II and Titan III rockets. Her bachelor's degree in chemistry is from California State University at Sacramento (1977). She is a member of the American Chemical Society, the Society of Women Engineers, and the

American Association of Women in Science. Mary was born in Sacramento, California. In her time off, she is restoring a 1965 Mustang convertible, but also likes water skiing, bicyling, and playing volleyball.

Joseph E. Scheffelin

The packaging, wiper, and tape cap for the PaintJet print cartridge were Joe Scheffelin's focal projects. As a manufacturing engineer, he was involved in developing ways to make a high-quality pen mass-manufacturable. Joe's current assignment, the development of automated production lines, builds on his past experience. Joe came to HP in 1981, joining the Inkjet Components Operation in Corvallis, Oregon. His BS degree in mechanical engineering is from the University of California at Davis. He was born in Rantoul, Illinois. Joe is married and lives in La Jolla, California. He plays the piano and likes running for recreation.

W. Wistar Rhoads

Shortly after receiving his BSME degree from Lehigh University in 1980, Wistar Rhoads joined the San Diego Division of HP. As a manufacturing development engineer, he was involved in the design of the pen assembly for the PaintJet printer. Presently, he is responsible for adhesive dispensing and ultraviolet curing on an automated printhead alignment machine. Much of Wistar's past projects were associated with thin-film engineering. He was born in Rochester, Minnesota, but raised in Philadelphia, Pennsylvania. In his spare time, he enjoys bicycling, horseback riding, and working on his house in Escondido, California.

45 Ink and Media

Ronald J. Selensky

As a San Diego Division development engineer, Ron Selensky was involved in developing PaintJet paper. No stranger to inkjet, he previously worked in a similar capacity on many phases of inkjet technology, among them materials development and testing, and the development of dye, ink, and print media. Ron now works as a manufacturing development engineer. In a previous position, Ron was a quality control chemist for a large food manufacturer. His primary professional interest lies in ultrafast laser techniques. Ron was raised in Great Falls, Montana, where he also earned his bachelor's degree in 1977. His MS degree in physical chemistry (1981) is from Washington State University. Ron currently lives with his wife Margo and their three children in Poway, California. He uses his woodworking skills in home remodeling and likes golf and fishing.

Peter C. Morris

Peter Morris developed the overhead transparency film for the PaintJet printer. As a development engineer in HP's San Diego Division, ink and media design for thermal inkjet printers has been Peter's primary responsibility. Presently, he is working on a polyester film medium for a new printer. His chemistry degree is from the University of California at San Diego (1979). Previously, Peter worked as a research chemist for a cosmetics manufacturer, where his developments resulted in patents for polymerization and a soft contact lens. Peter is a member of the American Chemical Society. He and his wife Cheryl live in El Cajon, California. Peter has an identical twin brother.

Donald J. Palmer

Don J. Palmer received his BS degree in chemistry from California State University at San Diego in 1977, later attended graduate school at Oregon State University (1978, chemistry) and Stanford University (1979, electrical engineering). He joined HP's San Diego Division in 1979 and consecutively served as an R&D engineer, project leader, and project manager of inkjet products. Don was responsible for several developments associated with the PaintJet printer: pen cartridge, ink, paper, overhead transparency medium, and color communications standards. His work on ink design and manufacturing processes resulted in a patent. Don is a member of the American Chemical Society and the Society for Imaging Science and Technology. He was born in San Luis Obispo, California, is married, and lives in San Diego, California. His recreational interests include running, public speaking, landscaping, and racquetball.

John Stoffel

With BS degrees in both chemistry and biology from the University of California at Irvine (1984), John Stoffel joined the San Diego Division of HP in 1984. He was responsible for formulating the ink chemistry used in the PaintJet printer. This technology continues to be the main focus of his development work. In past projects, John has worked on thin-film structure, passivation, and conductor/resistor film. A patent related to thin-film structure and two others on ink design are attributable to his work. He remains mainly interested in physical chemistry and is a member of the American Chemical Society. Born in Fullerton, California, he likes scuba diving, tennis, and playing chess.

M. Beth Heffernan

Just after joining HP in 1981, Beth Heffernan belonged to a team concerned with plant safety and industrial hygiene at the San Diego Division. Later, as a manufacturing engineer, she was assigned to the group responsible for development of the PaintJet pen. Her work resulted in a patent for a foam cleaning process. Beth was born in Peoria, Illinois, and now lives at Encinitas, California. In her off-hours, she likes buying and trading cars, snow skiing, traveling, and wine-tasting.

Mark S. Hickman

As a member of the PaintJet printer development team, Mark Hickman was responsible for design of the transparency film. He is now a development engineer working on special media for future HP printers. Mark joined the HP San Diego Division in the fall of 1986, shortly after he earned his BS degree in mechanical engineering from Washington State University. He also holds an earlier bachelor's degree (1979) in vocal music education from the University of Wisconsin, a training background he used in his previous vocation as a singer and voice teacher. Currently, Mark sings for recreation in the Escondido Oratorio and the Palomar Chorale. He also likes rock climbing and racquetball. He was born in Milwaukee, Wisconsin, is married and has a one-year-old son.

51 Printer Electronics

Jennie L. Hollis

Jennie Hollis joined HP's San Diego Division in 1984, just after she received her MS degree in computer science from the University of California at San Diego. Her BS in psychology (1976) is from the same institution. She became a member of the team developing the PaintJet Color Graphics Printer, working on the firmware. Before coming to HP, Jennie worked on the development of an automated tracer gas

monitor and a PCB analyzer. Born in Santa Monica, California, she now resides with her husband and two children in Olivenhain, California. In her off-hours, Jennie enjoys horseback riding, backpacking, and bicycle touring.

Philip C. Schultz

When he came to HP's San Diego Division in 1984, Phil Schultz was assigned to the PaintJet project, and for the ensuing years was responsible for design, testing, and support of the printer's input/output and main electronics. He has since moved to a new project involving a semicustom integrated circuit. Phil received his BSEE degree from the University of Illinois in 1984. He was born in Mount Prospect, Illinois, and now lives with his wife in Escondido, California. He is interested in Chinese cooking and likes to play tennis.

William J. Walsh

Before Bill Walsh was assigned to the PaintJet project, he had worked on the HP 7580/85 family of Drafting Plotters. His responsibilities as a PaintJet design engineer included the servo firmware, front panel code, and the prototype PCL code and stepper. His work contributed to two patent applications. Bill came to HP in 1981 with BS and MS degrees in computer science from Michigan State University (1978, 1979). His previous professional experience includes a post on the teaching staff of the University of California at Berkeley. After hours, Bill works as a volunteer aide in his daughter's first-grade math/computer lab. He is also a competitive swimmer.

57 HP-RL

Steven T. Rosenberg

Currently a member of the technical planning staff, Steven Rosenberg has been working at HP Laboratories since he joined HP in 1981. He has served as head of the expert systems department and as manager of the HP-RL and photolithography advisor projects. Most recently, he has been involved in setting up research collaborations with universities for HP Laboratories. Steven earned his bachelor's degree at McGill University in 1969 and his master's and PhD degrees in psychology at Carnegie Mellon University. Before coming to HP, he was a research scientist involved in artificial-intelligence development at the University of California Lawrence Berkeley Laboratory, and a member of the research staff at the artificial intelligence laboratory of the Massachusetts Institute of Technology. At MIT, Steven also held a position as a special lecturer in the Division for Study and Research in Education. He has authored or coauthored over a dozen papers on the subjects of artificial intelligence and human information processing. He belongs to the IEEE, the American Association for Artificial Intelligence, and the Canadian Information Processing Society.

71 MicroScope

Alan L. Foster

As a member of the technical staff for the MicroScope project, Alan Foster developed the static analysis component, a template-based cross-reference tool, and an execution history browser. Before coming to HP in 1984, he was a software engineer at the Lisp Company, where he contributed to an implementation of LOGO for personal computers. Alan received his BA degree in mathematics from the California State University at San Jose in 1982. His favorite pastimes include skiing and softball.

James P. Ambras

Since joining HP in 1978, Jim Ambras' assignments have focused on office automation application software and Lisp technology. As project manager for the MicroScope project, Jim had overall responsibility for the development of this software. He coauthored a paper on MicroScope, delivered at the International Conference on System Sciences, and a journal article on the same subject. Jim's BS degree in computer science is from Syracuse University (1977). He is a member of the IEEE and the American Association for Artificial Intelligence. He was born in New York City and lives in Cupertino, California. Jim's hobbies include mountain bicycling, windsurfing, and skiing.

Randolph N. Splitter

As a member of the team developing MicroScope, Randy Splitter's responsibilities included the execution monitoring implementation and user interface of the toolset. He is now a part of a group working on an object-based programming environment. Randy holds a bachelor's degree in English from Hamilton College, N.Y. (1968), and a PhD degree in English from the University of California at Berkeley (1974). His BA degree in computer and information sciences is from the University of California at Santa Cruz (1983). Before joining HP's Information Networks Division, he held a post as assistant professor of English at the California Institute of Technology. In a previous HP project, Randy's assignments included writing a reference manual for HP-RL, a knowledge representation and reasoning language. He was born in Kew Gardens, N.Y. Randy is married, has two daughters, and lives in Aptos, California.

Mark L. Chiarelli

The static analysis component was Mark Chiarelli's assignment in the development of the MicroScope software. Since he joined HP Laboratories in 1985, he has spent most of his efforts on Common Lisp development and intelligent programming environments. Mark has since focused his interests on the marketing aspects of engineering, with particular emphasis on academic institutions. His BS degree in engineering and masters' degree in community planning are from the University of Cincinnati.

Lucy M. Berlin

An R&D engineer at the HP Laboratories since 1984, Lucy Berlin was responsible for the design and implementation of the MicroScope user interface. She has been instrumental in developing many programming aids, including the HP Common Lisp Development Environment. Her professional interests include human factors in user interfaces, programming environments, and help systems. She is a member of the IEEE and a special-interest group of ACM on computers and human interaction. Lucy's BA degree in physics and computer science is from Queens College, New York (1981), and her MA degree in computer science from Stanford University (1983). She was born in Prague, Czechoslovakia. For recreation, Lucy likes hiking, racquetball, and table tennis. She also enjoys keeping up to date in physics, biology, and medicine.

Vicki O'Day

Since joining HP Laboratories in 1984, Vicki O'Day has been a member of a team working on a programming environment project. In the development of the MicroScope program, she participated in designing the execution monitoring component.

MicroScope also has been the subject of two publications Vicki has coauthored, a paper given at the International Conference on System Sciences in Hawaii and a journal article. She is a member of the IEEE, the Association for Computing Machinery, and the American Association for Artificial Intelligence. Vicki's BA degree in mathematics is from Mills College (1979) and her master's degree in computer science is from the University of California at Berkeley (1984). She was born in Albuquerque, New Mexico, and lives in Menlo Park, California.

Serge L. Rudaz

Serge Rudaz was born in Vevey, Switzerland, and studied physics at the Swiss Institute of Technology at Lausanne, where he received a Diplome D'Ingenieur-Physicien (MS physics) in 1975. He served in the Swiss army, serving as a radio operator in the mountain artillery troop. Serge continued his studies in physics at the University of Illinois at Urbana-Champaign (MS Physics 1976, PhD 1983). He joined HP's Optoelectronics Division in 1985 as a development engineer. He worked on the AlGaAs LED project in the areas of metallization and metal-semiconductor interfaces, wafer fabrication development, and characterization. Serge is the author or coauthor of 21 papers on nuclear magnetic resonance and Mu spin resonance, and the coauthor of two papers on AlGaAs LEDs. His professional interests include nuclear magnetic resonance, chaos and fractals, and metal semiconductor interfaces. He is a member of the American Physical Society and the Swiss Society of Architects and Engineers. Serge is married and lives in Sunnyvale, California.

Chin-Wang Tu

Ching-Wang Tu is currently an engineering section manager at HP's Optoelectronics Division. He joined HP in 1984, after receiving a PhD degree from Colorado State University. Ching-Wang was responsible for the development and the transfer of a production-scale wafer fabrication and die fabrication process for the red AlGaAs LEDs to manufacturing. He is the coauthor of seven published papers on epitaxial growth and material properties of compound semiconductor materials. Ching-Wang's professional interests include epitaxial growth of III-V materials using chemical vapor deposition, liquid-phase epitaxy, and molecular-beam epitaxy. He is also interested in device processing of LEDs and laser diodes. His outside interests include the game Go, hiking, tennis, volleyball, and watching football games. Ching-Wang is a native of Taiwan. He currently lives in Cupertino with his wife and two children.

Michael D. Camras

Mike Camras studied electrical engineering at the University of Illinois at Urbana-Champaign (BSEE 1980, MSEE 1981, PhD 1984). He joined HP in 1984. As a development engineer on the red AlGaAs LED project, his activities encompassed both absorbing-substrate and transparent-substrate red AlGaAs LEDs. Mike is the author or coauthor of over 25 papers on III-V semiconductor lasers and LEDs, and is a member of the American Physical Society.

Dennis C. DeFevere

Born in Phoenix, Arizona, Dennis DeFevere worked as an R&D engineer at Fairchild Semiconductor for seven years before joining HP in 1973. One of his earlier assignments at HP was the development of an epitaxy process for GaAs field effect transistors. On the red AlGaAs project, Dennis was responsible for the design and construction of a liquid-phase epitaxy reactor and the development of an epitaxy process for production of the red AlGaAs LEDs. After serving four years in the U.S. Air Force, Dennis attended the University of Arizona from 1957 to 1961 to study engineering science. Dennis has three patents, two on an apparatus for liquid-phase epitaxy and one on gas-discharge displays. He is also the coauthor of two papers on high-efficiency red LEDs. His professional interests include crystal growth and epitaxy, and he is a member of the Northern California Crystal Growers. His hobbies and outside interests include golfing, skiing, opera, and radio-controlled model airplanes, and he is a member of the San Francisco Opera Guild. Dennis is married, has two children, and lives in Palo Alto, California.

Wayne L. Snyder

With HP since 1970, Wayne Snyder is currently an R&D project manager at the Optoelectronics Division. He was project leader for the red AlGaAs LED project. Wayne studied electrical engineering at the University of Pennsylvania (BSEE 1965) and at Stanford University (MSEE 1966, PhD 1970). He has worked on the development of VPE, LPE, and MOCVD epitaxial processes, and on various visible and infrared LEDs. He is the author of 10 technical articles and conference proceedings on LEDs and LED materials. Wayne was born in Easton, Pennsylvania, and currently lives in Palo Alto, California. He is married and has two children. In his spare time, he enjoys basketball, skiing, and volleyball.

Louis W. Cook

A native of Henry, Illinois, Louis Cook attended the University of Illinois, earning a BSEE degree in 1978, an MSEE degree in 1981, and a PhD degree in 1982. He joined HP in 1982 and worked as a development engineer on the large-chamber GaAsP(N) vapor-phase epitaxy project. He is currently working on transparent-substrate red AlGaAs LEDs. Lou has authored over 30 papers on compound semiconductors. He has submitted two patent disclosures for transparent-substrate red AlGaAs LEDs. Lou is single and lives in Santa Clara, California. His hobbies and interests include hiking, camping, photography, and geology.

David K. McElfresh

David McElfresh attended the University of California at Davis where he earned a BS degree in electrical engineering and material science in 1980, and MS and PhD degrees in material science in 1982 and 1984, respectively. He joined HP in 1984. David worked on developing the AlGaAs processing technology and transferring the processes into production for the red AlGaAs LED project. Currently he is doing the same for transparent-substrate AlGaAs processing. David has published almost 20 papers covering such subjects as nuclear waste, diffusion in glasses, the structure of glasses, the thermodynamics of glasses, and AlGaAs devices. He is a member of the Northern California Crystal Growers. David is a native of Savannah, Georgia, and currently resides with his wife in Union City, California.

Frank M. Steranka

A native of Pittsburgh, Pennsylvania, Frank Steranka studied electrical engineering and physics at the University of Pittsburgh, earning both BSEE and BS degrees in 1978. He continued his study of physics at the University of Illinois, earning an MS degree and a PhD in 1980 and 1984 respectively. He joined HP's Optoelectronics Division in 1984. He worked on characterization and reliability studies of the red AlGaAs LEDs and on the development of an apparatus for semiconductor characterization. Frank is the author or coauthor of six papers on excitons and electron-hole liquid in silicon and germanium, and the coauthor of two papers on red AlGaAs LEDs. He is a member of the American Physical Society, and his professional interests include solid-state physics, semiconductors, and optoelectronics. Frank is married, has one child, and lives in San Jose, California. His leisure interests include basketball, camping, and photography.

MicroScope: An Integrated Program Analysis Toolset

MicroScope supports evolutionary software development by helping programmers understand complex programs written in Common Lisp.

by James P. Ambras, Lucy M. Berlin, Mark L. Chiarelli, Alan L. Foster, Vicki O'Day, and Randolph N. Splitter

PROGRAMMERS OFTEN NEED TO MODIFY existing programs for the purpose of fixing bugs, adding features, or integrating these programs into new hardware or software environments. Since this task is time-consuming and error-prone, the maintenance (evolution) of existing software is one of the most inefficient activities in the software life cycle.

The main reason for this inefficiency is the difficulty programmers have in understanding the increasingly large and complex programs made possible by advances in computer hardware. Such programs may evolve over a long period of time, during which programmers may forget the details of code they wrote earlier. They may also forget the higher-level design decisions behind the code. The difficulty is compounded when the person who is trying to understand the program didn't make the original design decisions or didn't write the original code.

Most programmers today approach the task of program understanding as they have for the last twenty years—by analyzing source code listings, reading documentation, and asking questions of local experts. These techniques are insufficient for large and complex programs. Programming environments that support evolutionary software development must include tools that help programmers understand complex programs.

MicroScope, an experimental program analysis system developed at HP Laboratories, tries to provide such tools. But MicroScope is not a loosely coupled collection of unrelated utilities. It is designed to be an integrated set of tools that share common data and a uniform user interface. The fact that information is shared means that the user does not have to transfer data manually from one tool to another. The uniform interface makes it easier for the user to switch from task to task. Moreover, this interface is graphical and interactive. The visually oriented, two-dimensional way in which information is presented makes such information easier to retrieve and easier to assimilate. The mouse-based style in which users point at the items they want to select simplifies the task of entering commands and queries.

The information MicroScope uses is stored in a memory-resident data base composed of record-like complex data structures with named components (attributes). Such complex data structures are usually called objects.* Objects with the same attributes are created as instances (members) of the same type or class. This layer of abstraction makes it easier to deal with complex information.

Since MicroScope is a research prototype, it does not try to provide a complete set of program analysis tools. The current system includes a static component that analyzes the cross-reference structure of a program and a dynamic component that lets users monitor the run-time behavior of the program. Together, these components can help a programmer understand the relationships between different parts of a program, see how those parts interact during the running of the program, and locate the sources of bugs in a program that doesn't work as expected. Both components maintain a rich representation of data that allows them to respond to a wide range of complex user queries and requests. The dynamic component also employs rule-based reasoning so that it can more easily monitor complex execution events while enabling the user to specify these events in a simple, high-level way.

The current MicroScope prototype analyzes and monitors programs written in Common Lisp. It also provides some support for CommonObjects, an object-oriented extension to Common Lisp.[2] The code examples presented in this article are given in Lisp, but most of the features of MicroScope are applicable to conventional languages.

The next section of this paper discusses some related work. The section after that describes the graphical interface, at the same time giving an overview of the capabilities MicroScope provides. The subsequent section discusses MicroScope's cross-reference analysis component in greater detail, including representation and implementation issues. The final section does the same for MicroScope's execution monitoring component.

Related Work

Graphical (mouse-based and window-based) interfaces are now becoming common. These interfaces originated in programming environments such as those created for Smalltalk[3] and Interlisp-D.[4,5] The Smalltalk environment introduced the concept of browsers, two-dimensional presentations of a collection of items that let one zoom in on the contents of individual items. The browser construction toolkit used by MicroScope provides the ability to display

*In the current MicroScope prototype, static information and dynamic information are stored in two different kinds of object-like data structures, as explained later in this paper. For more detailed explanations of some of the data representation and reasoning terms mentioned in this article—terms such as class, frame, slot, and rule—see reference 1.

various kinds of collections in a variety of ways.

MicroScope's table-driven approach to static analysis, similar to the approach taken by Masterscope[5,6] for the Xerox Interlisp environment, separates semantic information from the control structure of the code analyzer. This separation makes it easier to adapt the analyzer for use on other dialects or languages.

Existing debugging tools generally allow users to trace function calls, monitor variables, and set breakpoints. Some tie debugging actions to user-specified events or conditions.[7-11] MicroScope differs from these in allowing the user to specify event combinations and conditions in a declarative (higher-level) way. It also lets the user make use of cross-reference information derived by the static analysis component.

MicroScope stores program analysis data in object-like data structures that are accessible by different components of the system. The idea of storing program information in a shared (and persistent) data base of complex objects, rather than in text files, underlies the design of many current experimental programming environments, including the Common Lisp Framework system (CLF),[12-14] Garden,[15-17] and others.[18]

MicroScope itself is based on a prototype developed by Jed Krohnfeldt at the University of Utah.[19]

Graphical User Interface

MicroScope's graphical user interface is built upon the X Window system, the XRLIB user interface toolkit library,[20] and an experimental software development environment called Ivo.

In this environment, multiple windows, some overlapping others, can be visible on the screen simultaneously, like documents on a desk. Fig. 1 shows a sample screen. The narrow rectangle at the top of most of the windows is a *titlebar*, which identifies the window's contents. Many windows also have narrow rectangles containing *scrollbars* along their right and/or bottom edges. Scrollbars allow a user to see parts of a buffer not currently visible in the window.

A user of this graphical interface can type characters into editing buffers and can issue some keyboard-based commands. But most of the work of switching between windows and issuing commands in general is done by means of a device called a *mouse*. Moving the mouse moves a special mouse cursor on the screen; clicking a mouse button applies some operation to whatever item the mouse (i.e.,

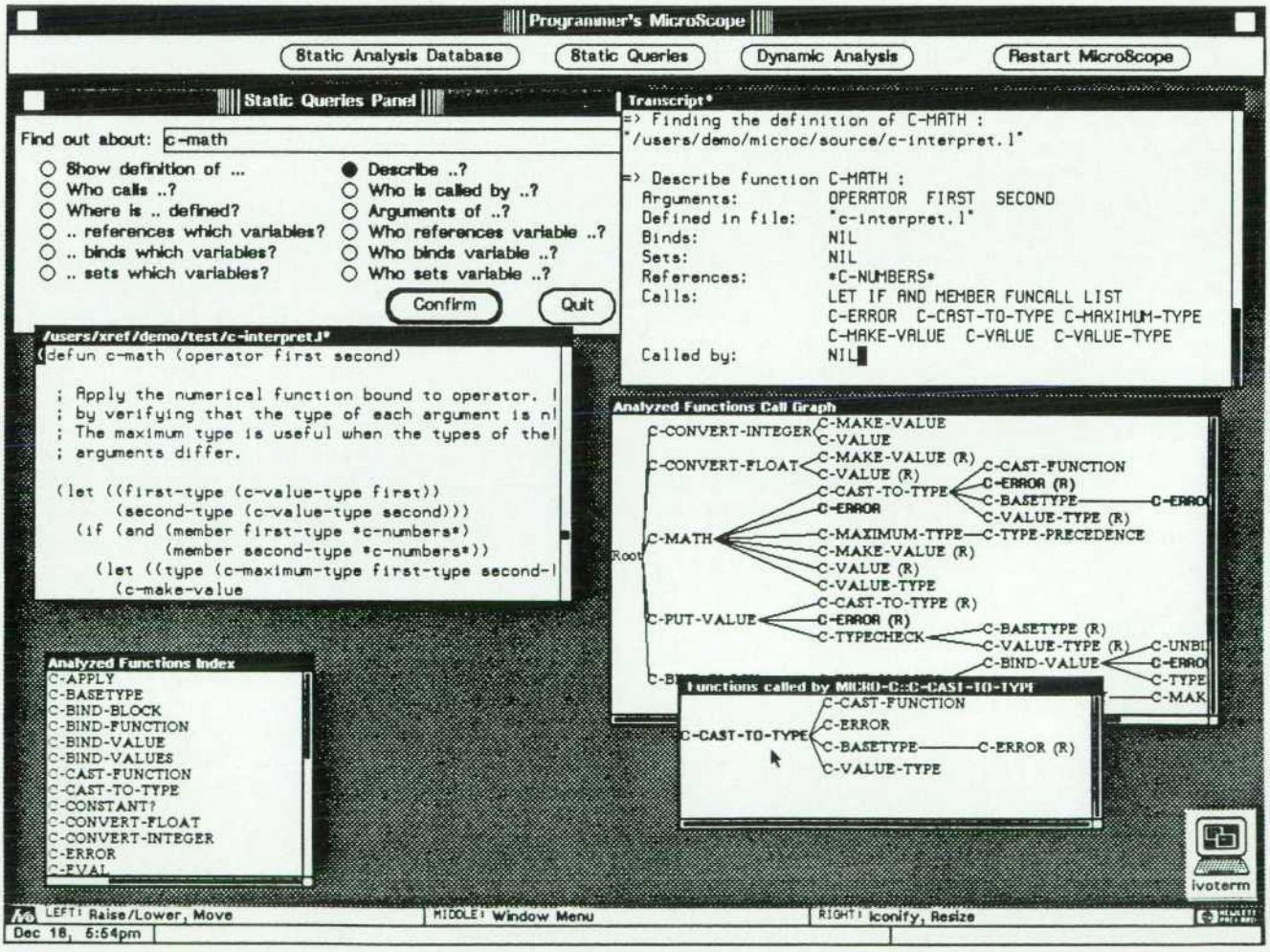


Fig. 1. *A sample MicroScope screen.*

the cursor) is pointing to. Actually, mice may have more than one button; in the current MicroScope system, they have three. You can *click* one of the mouse buttons (pressing and releasing the button while the mouse and the mouse cursor are stationary) or *drag* the mouse (holding a button down while moving the mouse). Which command is invoked depends on the item the mouse is pointing to, which mouse button is pressed, and whether the mouse is clicked or dragged. For example, clicking the left button when the cursor is on a window's titlebar brings that window to the "top" of the display "above" other windows (or pushes it to the bottom if it's already fully exposed). Clicking the right button in a scrollbar region scrolls the window forward, making the next "page" visible.

The sample screen in Fig. 1 shows four types of windows: buffer windows, icons, panels (dialogue boxes), and browsers. The window labeled with the long file name and the one labeled Transcript are buffer windows. They contain text that can be edited. The tiny window in the lower-right corner (Ivoterm) is an icon, which is the shrunken counterpart of some other, full-sized window, usually one that is hidden from view. Clicking with the mouse over an icon restores the full-sized window to the screen, making it appear as if the icon were expanding to its normal size.

The windows labeled Programmer's MicroScope and Static Queries Panel are XRLIB *panels*. Like menus, panels enable a user to make a selection from a list of items or commands. But panels allow more complex interaction. They let the user make selections from multiple sets of items, not just one. They let the user select more than one item from a given set of choices. They let the user enter and edit text for the cases in which a complete set of possible choices is not known. Unlike pop-up or pull-down menus, panels remain visible on the screen, letting the user make multiple choices, edit text, and make changes in the selections that have already been made. Like menus, panels can be "cascaded": selecting certain commands can bring up additional panels.

The terms *menu* and *panel* (i.e., control panel) imply somewhat different kinds of interaction models. In the menu model, the user selects one item from a vertical list of text items on the screen. In the control panel model, the user *pushes buttons* to make choices and give commands. For example, the rectangular/elliptical "buttons" with rounded corners are called *pushbuttons*. The labels inside these buttons represent commands. Clicking the physical left button of the mouse when the mouse cursor is over a graphical pushbutton causes the specified command to be executed immediately. The small, circular buttons with adjacent labels are called *radiobuttons*. Clicking the mouse when the cursor is on a radiobutton (in the circle) causes the associated item to be selected (the circle is "filled in"). The item may represent a command, but the command is not immediately executed. Selecting one radiobutton item, like pushing a button on a car radio, causes the last selection to be deactivated; only one item in a set of radiobuttons can be active at any one time. A third kind of graphical button, consisting of a small square and an adjacent label of text, is known as a *checkbox*. Clicking on the square selects the associated item but doesn't deactivate the previous selections; all checkboxes selected remain filled in.

The long, horizontal rectangle in the Static Queries panel is a single-line text-editing buffer, in which the user can enter or edit a line of text. Multiple-line buffers are also available.

The remaining windows contain *browsers*, which display collections of graphical or textual items in different ways. Browsers permit various operations to be performed upon the individual items or upon the collection as a whole. One such operation is the action of *browsing* into a given browser item, which shows the contents of that item or displays more information about it, possibly in another browser. For more information about browsers, see the box on page 76.

Exploring Static Structure

MicroScope provides both static and dynamic information about a program. The static information, which is independent of the program's execution, consists of cross-reference relations among the program's functions and variables.

To initiate static analysis of a set of files and retrieve the cross-reference information that has been acquired, a MicroScope user may either call special Lisp functions or use the interactive, graphical interface described here. The programmatic interface provides more options, but the graphical interface is generally more convenient to use and displays information in ways that make the information easier to assimilate.

MicroScope's top-level control panel contains several pushbuttons (see Fig. 1). Clicking on the pushbutton labeled Static Analysis Database brings up another panel. Selecting the appropriate radiobutton and entering the file name(s) in the editing buffer causes MicroScope to analyze the source code contained in those files. After the files are analyzed, an Analyzed Functions Call Graph browser, showing the static call relationships among the functions, appears automatically (Figs. 1 and 2). Lines going from left to right connect each function with the functions it calls. Each function appears wherever it is called, so that a given function may appear several times. A screen-dump program can be used to generate a hard copy of the graph. (The screen images used in this article were generated in this way.) The graph can be very large, but one can scoll the graph by clicking on the horizontal or vertical scrollbar to show information not currently visible.

Pop-up menu commands associated with the Call Graph browser allow a user to get more information about the functions listed in the graph. One of these commands, Browse Call Tree, creates a new browser that displays (on a larger scale) a call graph starting at the currently selected function. Similarly, Browse Definition pops up a new window showing the appropriate source file and positions the cursor at the beginning of the function definition. The command Highlight Self shows all the places in the graph where the function appears. Other commands result in the function node being "decorated" by its formal parameter list or the name of the file in which it is defined.

Additional radiobutton selections on the Static Analysis Database panel make available textual browsers that contain indexes of the analyzed files, functions, and variables in the data base. These browsers also have pop-up menus

associated with them.

If one wants to ask specific questions about the program's cross-reference relationships, one can return to the top-level panel and click on the pushbutton labeled Static Queries. In the Static Queries panel (Fig. 1), one can name a function or variable and select a particular command or query. The available commands and queries include Show definition, Who calls, Who binds, and so on. One can specify an item by typing the name into the editor or by copying a name that has been selected in an existing buffer or browser (e.g., a source code buffer or call graph browser).* Copying the item saves typing, avoids typing mistakes, and makes it unnecessary to switch back and forth between the keyboard and the mouse.

Since the item specified in the editor remains selected until it is replaced, the user can ask a follow-up question about the current item simply by clicking on another radiobutton and then pushing the Confirm button.

The scrollable Transcript window (Fig. 1) records each query and its answer.

*Dragging the mouse cursor over a region of text while the left button is down selects that piece of text. Clicking the right button when the cursor is over a panel editor copies the selected text to the editor. The whole process is like physical cutting and pasting, but not as messy.

Monitoring Execution Events

Besides providing static cross-reference information about a program, MicroScope allows one to monitor the program's execution. One can specify source-level events one wishes to monitor and actions one would like performed when these events occur.

Pushing the Dynamic Analysis button on MicroScope's top-level panel causes the Execution Monitor Request panel (Fig. 3) to appear. By clicking on the appropriate radiobutton in this panel, the user can select whether a single event or a combination of events should be monitored. Pressing the Event Menu pushbutton causes a new, Execution Monitor Event panel to appear on the screen. A set of radiobuttons on this panel enables users to specify the kind of event they want to monitor (function invocation, variable reference, etc.). They can list the name(s) of the functions or variables explicitly by typing them into the text editor or by copying them, via the mouse, from existing buffers or browsers. They can also specify a static cross-reference relation which MicroScope can use to determine the functions or variables it needs to monitor.

The way to specify a static relation is to click on the pushbutton marked Static Condition. Another panel will pop up, offset from the first two panels. This panel allows the

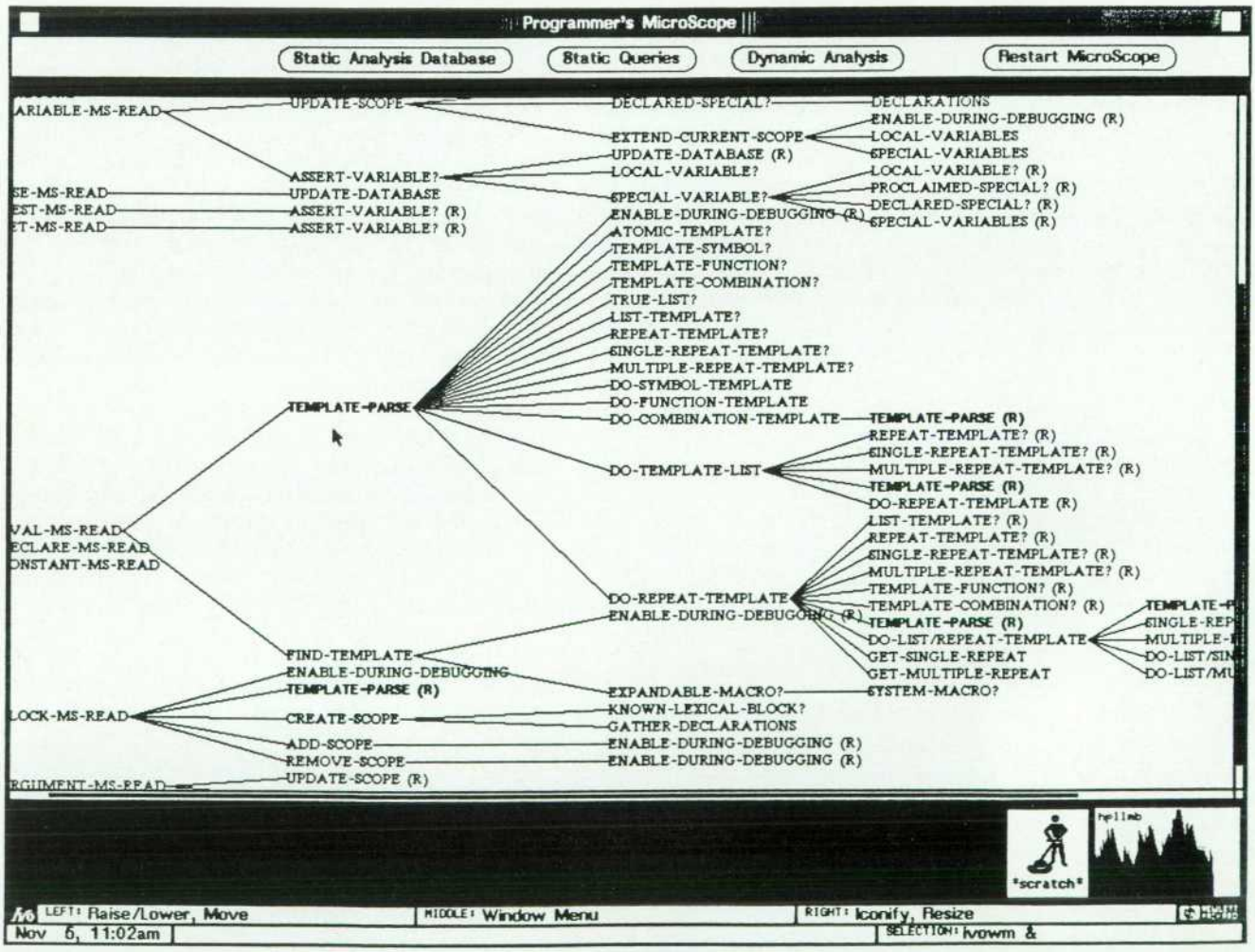
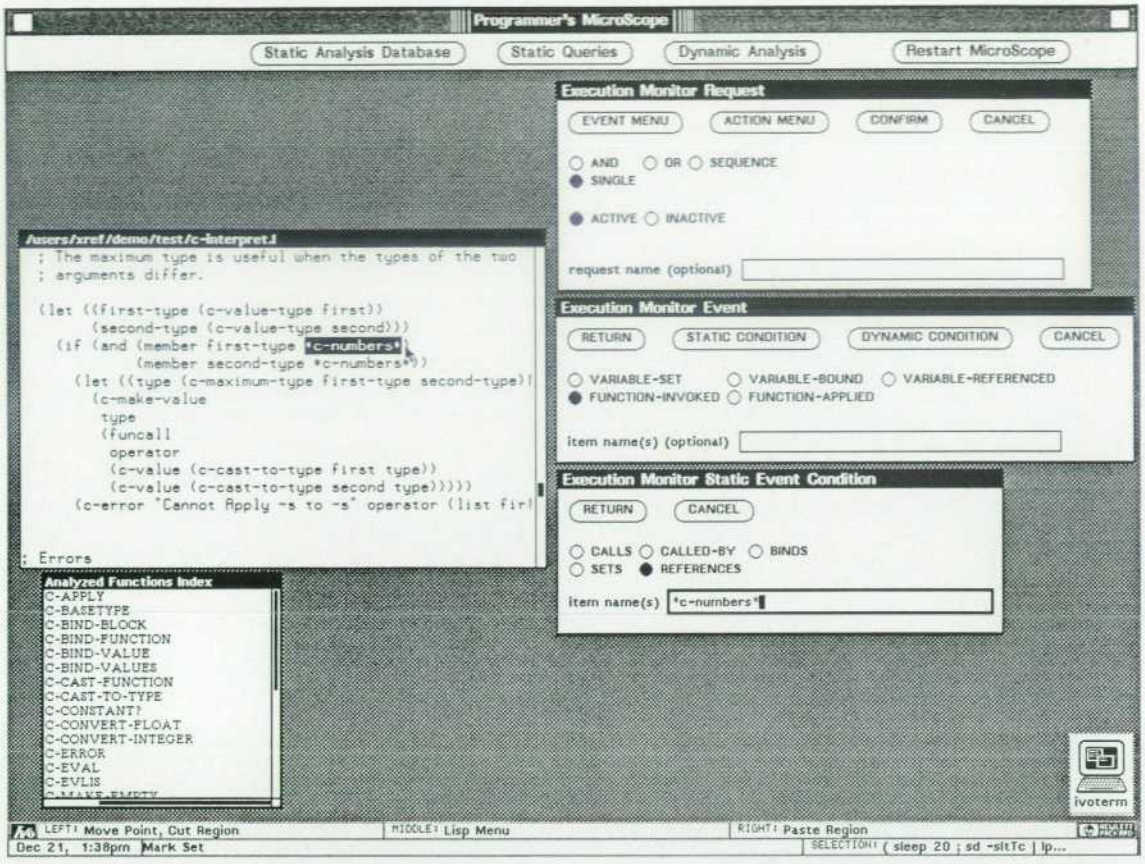


Fig. 2. *A Function Call Graph browser.*



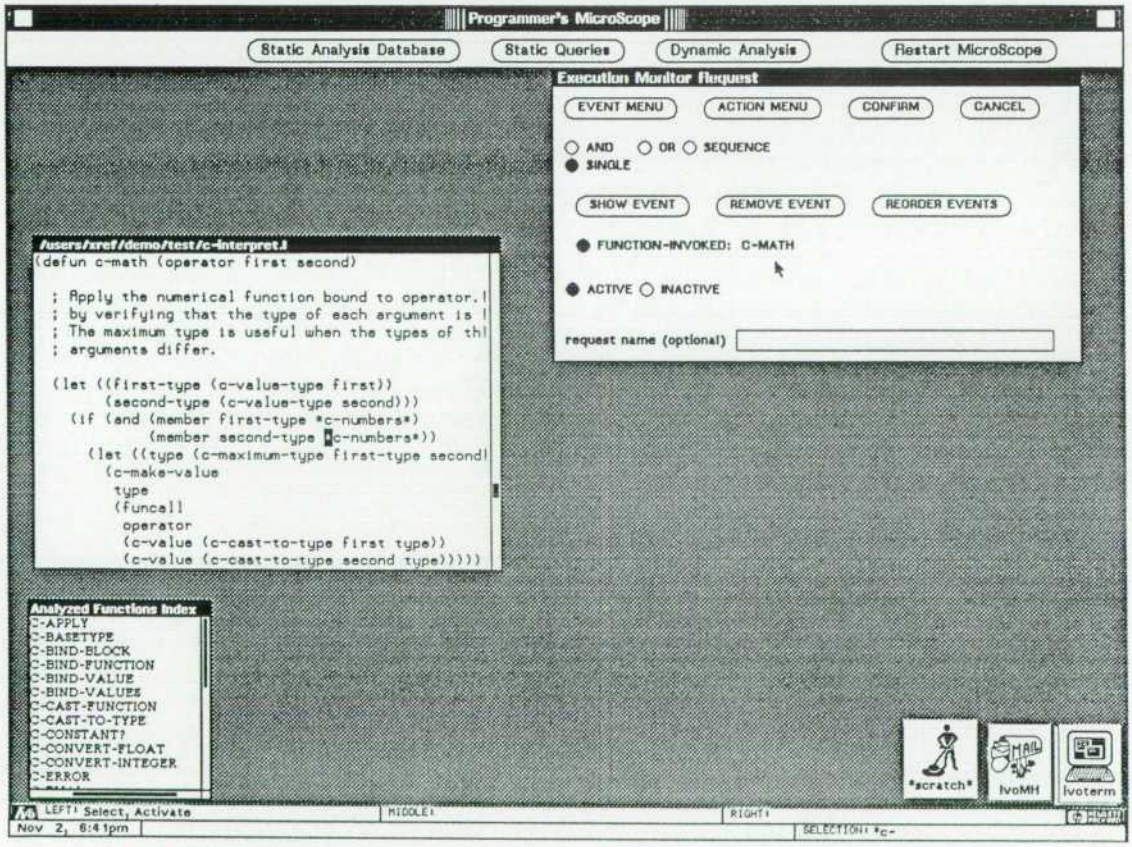


Fig. 3. *(Top) An execution monitoring request with the monitored item specified by a static condition. (Bottom) The result of the static condition.*

The Browser Construction Toolkit

The browser toolkit (part of the Ivo software development environment) provides a framework for constructing browser-style interfaces. This toolkit assumes a three-layered model of a browser and its contents. The bottom (*semantic*) layer consists of whatever data the browser is "viewing." The semantic layers of MicroScope's browsers correspond to objects containing static or dynamic information about a program.

The middle (*manipulation*) layer consists of objects that mediate between the semantic data and the view of the data that is presented in the browser. The objects in the manipulation layer do not know how to draw themselves on the screen; they simply contain information that has been extracted from the semantic layer. Manipulation objects are created during the construction of the browser.

The highest of the three layers is the *presentation* layer, which supplies various building blocks for displaying information, including text, tree diagrams, collections arranged in various ways, and buttons of various kinds. These building blocks (views) are combined according to layout rules that prescribe how particular kinds of objects should be displayed. One set of rules may state, for example, that a "function" manipulation object, when displayed as a node in a tree, should be represented by its name (a text string).

Thus the browser toolkit separates presentation from semantics, making it possible to display different information via the same basic views and to display the same information in a variety of different ways. By providing standard views, a way of linking them to the objects that contain the needed information, and a way of combining them, the toolkit makes it easier to present information in a meaningful and consistent way.

user to select the appropriate static attribute and to indicate a related function or variable. In the example illustrated in Fig. 3, selecting the event type function-invoked in the Event panel, selecting the cross-reference relation references, and specifying the variable *c-numbers* asks MicroScope to monitor invocations of all functions that reference the variable *c-numbers*. The data base of static information is accessible during the formulation of execution monitoring requests; MicroScope's static and dynamic components can share information without special intervention by the user.

Similarly, a MicroScope user can call up a Dynamic Event Condition panel to specify a condition that will be checked at run time (such as the value of the variable being referenced). Dynamic conditions do not determine which functions or variables should be monitored. Rather, they determine whether the event that occurred should trigger the corresponding action.

Another pushbutton on the top Execution Monitor Request panel, marked Action Menu, triggers the appearance of a panel that allows one to specify an action to be performed when the indicated event occurs. One can specify multiple events or actions by selecting the Event Menu and Action Menu pushbuttons as many times as one wants.

The panels needed to specify an execution monitoring request are cascaded: pushing a button on one panel causes another one to pop up. Pushing a Return button on the child panel hides that panel and moves the mouse cursor back to the parent panel. Because selections are distributed over several panels and panels are displayed and hidden as needed, a user can make many choices without becoming overwhelmed by the number of possible combinations. Indeed, the graphical interface enables one to specify complex combinations of events and conditions without having to use a complicated command language.

Each parent panel shows information from its child panels in summary form. In particular, if you ask MicroScope to use a static condition to determine which functions or variables should be monitored, the *results* of querying the cross-reference data base are displayed in the Execution Monitor Request panel. In Fig. 3, the one function that references the variable *c-numbers* is listed: c-math.

Besides providing the ability to spawn new child panels, various pushbuttons on a parent panel enable a user to remove an existing child panel, thereby removing an event, condition, or action from the monitoring request, or to redisplay the panel to see the information in fuller detail or modify it. Other pushbuttons on the Execution Monitor Request panel enable the user to change the order of multiple events or actions.

The monitoring request isn't translated into its internal, rule-based representation until the user pushes the Confirm pushbutton on the Request panel. Delayed confirmation permits users to compose complex requests as they go along, possibly changing their minds along the way, rather than forcing them to specify the information correctly all at once. Waiting for confirmation also avoids needless computation.

Confirming a request hides the Request panel and brings one back to MicroScope's top-level control panel. A pushbutton on this panel lets one initiate the execution and monitoring of a program. Clicking on this button brings up another panel containing a multiline editing buffer. As usual, one can type in the Lisp expression to be executed or copy it from another buffer by cutting and pasting.

During execution, a graphical Execution History browser (Fig. 4) appears automatically and is dynamically updated as the program runs. This browser displays a tree whose nodes are the monitored events that actually occurred. (The parent of a variable event node always represents the function call that set, bound, or referenced the variable, whether or not there was an explicit request to monitor that function call.) As in the case of the static Call Graph browser, the tree of events can be very large, but one can use the horizontal or vertical scrollbars to move around the tree as needed. A pop-up menu associated with the browser provides other options, letting one see the information associated with a given event (e.g., variable value, function arguments, function return value).

The actions specified in a monitoring request may cause additional information to appear on the screen. For example, the contents of variables may be displayed (dynamically) in separate windows. The user can examine this information or explore the Execution History browser when the program has finished executing. But one can also suspend the program during its execution by letting the *suspend* action be triggered by an execution event or, interactively, by clicking on a pushbutton attached to the Execution History browser. After examining the information available when the program is suspended, the user can click the abort or continue pushbutton.

Using Templates in Cross-Reference Analysis

The following piece of code uses the Common Lisp special form case:

```
(case sequence-type
    (list (create-list) (notify-user))
    (array (create-array)))
```

The case form evaluates its first argument (sequence-type) and matches the result against a series of constant "keys" (the first elements of the following sublists—in this case, list and array). The keys are not evaluated. If any key matches the original test result, the remaining expressions in the sublist are executed.

When trying to find a template that matches this expression, the pattern matcher notices that the expression is an unquoted list. The matcher checks the first element of the list to see whether that element is the name of a special form or macro that has a corresponding template. It is, so the matcher applies the following template to the expression:

```
(use eval (repeat (constant (repeat eval))))
```

The template symbol use corresponds to the invocation of a function, macro, or special form. The special symbol eval corresponds to a recursive call on the pattern matcher. The matcher, finding that sequence-type is an unquoted symbol, will match it with the template symbol test, meaning a variable reference.

The special operator repeat repeatedly matches its arguments against the remaining subexpressions in the expression being analyzed. In this case, the first repeat has one argument, the template list beginning with the symbol constant. This template list is matched against each sublist of the case form in turn. The symbol constant corresponds to the unevaluated key. The next repeat template corresponds to the rest of each sublist.

The effect of this second repeat operator is to match its one argument, the symbol eval, against each subexpression in the list—in other words, to search for a template matching each subexpression. In this example each subexpression is a function call form (not a macro or special form with its own special template), so that the default function call template is applied to each such form.

Static Program Analysis

MicroScope's static analysis component collects cross-reference information about functions and variables. The graphical interface greatly simplifies the interaction between MicroScope and the user, but the programmatic interface—a set of Lisp functions with various keyword options—offers a somewhat wider range of capabilities, including the possibility of combining predefined functions into a more specialized, user-defined query function. The programmatic interface also allows the user to work in a basic Lisp environment, without X-Windows-based graphics.

Queries

Programmatic and panel-based queries let the MicroScope user retrieve cross-reference information from an existing data base. The Static Queries panel in Fig. 1 shows an example of a panel-based query.

The names of program components, including the names of source files, often provide clues to their purpose. For this reason MicroScope includes a query that returns a list of the names of all components of a particular type (files, functions, or variables). The query Where defined returns the full pathname of the file in which a given function is defined.

Like most cross-reference systems, MicroScope can provide a list of the functions called by a given function (Who is called by). In addition, the query Who calls lets one obtain the names of all the functions that call a particular function. One can also see the whole tree of functions that are called, directly or indirectly, by a root function.

In Common Lisp, local variables have lexical scope: they can be referenced only within the lexical (textual) construct in which they were introduced. Because it is usually easy to track such references down, MicroScope's default behavior is not to collect cross-reference information for local variables. (That option is available, however.) Global (*special*) variables can appear anywhere in the program. In the case of these variables, the cross-reference information that MicroScope collects helps one to locate the places, often scattered in the program's text, where their values are accessed or modified.

MicroScope keeps track of three cross-reference relations involving variables: which functions *set* them (assign a value to them via a Lisp assignment form), which functions *bind* them (give them a value in some other way), and which functions *reference* them (access their current value). Besides asking which functions refer to a given variable, the user can find out which variables are set, bound, or referenced by a given function.

The query Describe returns all the cross-reference information that MicroScope has about a particular program item. For example, the Transcript window in Fig. 1 shows the result of invoking the Describe query on the function c-math.

Data Representation

In the current MicroScope prototype, cross-reference information is stored in a memory-resident data base of structured objects.* Each kind of program item (including functions, variables, and files) is represented by a particular object type. The slots (components) of these objects, representing attributes of the program item, contain cross-reference information. For example, a function object contains such slots as file, arguments, binds, references, sets, uses (i.e., calls), and used-by (called by). A variable object contains such slots as bound-by, referenced-by, and set-by.

Notice that a function object has slots containing the names of both the functions it calls and the functions it is called by. Having both of these slots is redundant since the relations are inverses of each other, but this redundancy makes access to the information quicker. Instead of figuring out all the functions that call function foo by checking the uses slot of all analyzed functions in the data base, MicroScope can simply check the used-by slot of function foo.

The values of slots that represent inverse relations must remain consistent. MicroScope uses daemons—pieces of

* These objects are Common Lisp structures that have been augmented in certain ways.

code attached to a slot and executed when the slot is accessed or modified—to maintain this consistency. If the reference to function b on the uses slot of function a is deleted, a daemon attached to this slot will remove function a from the used-by slot of function b.

Implementation of the Cross-Reference Analyzer

MicroScope determines relationships among program items by using a table-driven *codewalker* that matches templates against the source code. Since all language-dependent semantic information is isolated in the table, the templates could be replaced or modified without changing the overall architecture of the analyzer. Using modified templates, one could analyze a different dialect of Lisp or even a language other than Lisp. Since the cross-reference analyzer is not tied to a particular compiler or preprocessor, it is not implementation dependent.

In general, the templates MicroScope uses are lists of symbols. Each symbol in this list matches an element in the Lisp expression (list) being analyzed. The correspondence between the template symbol and the program item establishes what kind of item it is or what kind of semantic action is being applied to it. For example, the symbol variable means that the corresponding variable is a new lexical (local) variable. The symbol set means that the corresponding variable is being assigned a new value. The symbol test means that the corresponding variable is being referenced. The symbol use means that the corresponding function (or macro or special form) is being called.

In addition to these symbols, which match program items directly, the first symbol in a template (or subtemplate) list may be a special pattern-matching operator that applies the rest of the list (its *arguments*) to the corresponding source code expression in some way. For example, repeat repeatedly matches its arguments against the elements of a Lisp list. Other operators apply a particular function to the Lisp expression, either to extract some information from the expression or to transform it so that another template can be matched against it. Another special symbol, eval, searches for the right template to match against the Lisp expression.

How is the right template found? If the Lisp expression being analyzed has the form of a function application (i.e., an unquoted list), MicroScope checks whether the function is actually a macro or special form; if so, it applies the appropriate template to the expression. There are individual templates for some Common Lisp macros and all special forms. If MicroScope doesn't recognize the function

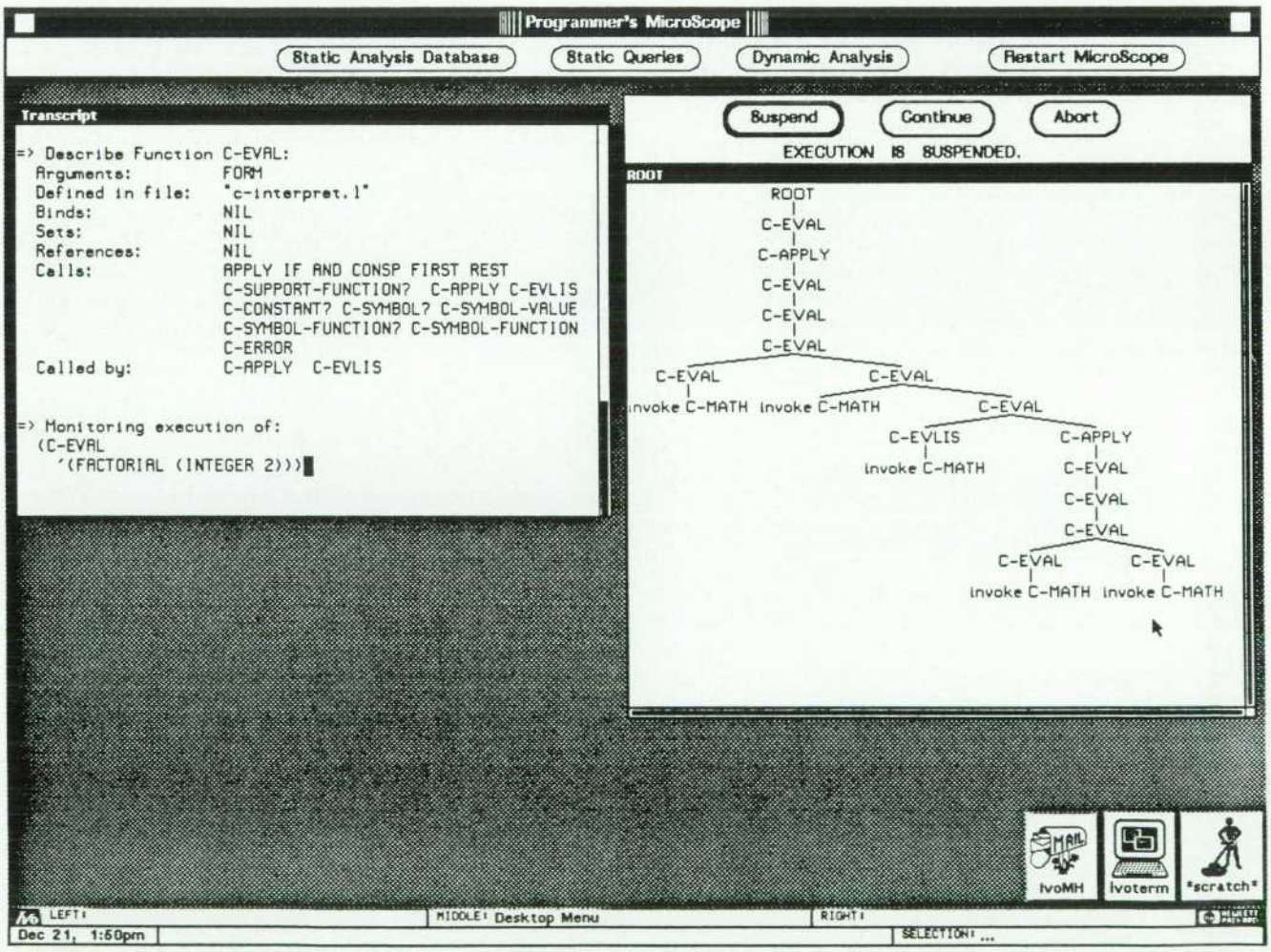


Fig. 4. *An* Execution History *browser.*

Rule-Based Execution Monitoring

In this example all the arguments of function foo are intended to be numbers. To check whether this function is ever called with a non-numerical argument, the user tells MicroScope to monitor all applications of the function and suspend the program if any argument is not a number. In Fig. 1, the user has checked the box labeled function-applied and has specified the name of the function, foo.

The user has also specified a dynamic condition, to be tested at run time, which checks whether the list of evaluated arguments (arg-value-list) satisfies the predicate non-numeric-arg-p, which the user has defined. This predicate checks each argument and returns true if any argument is not a number.

The action specified is to suspend the program. The Execution History browser will appear automatically, displaying nodes cor-

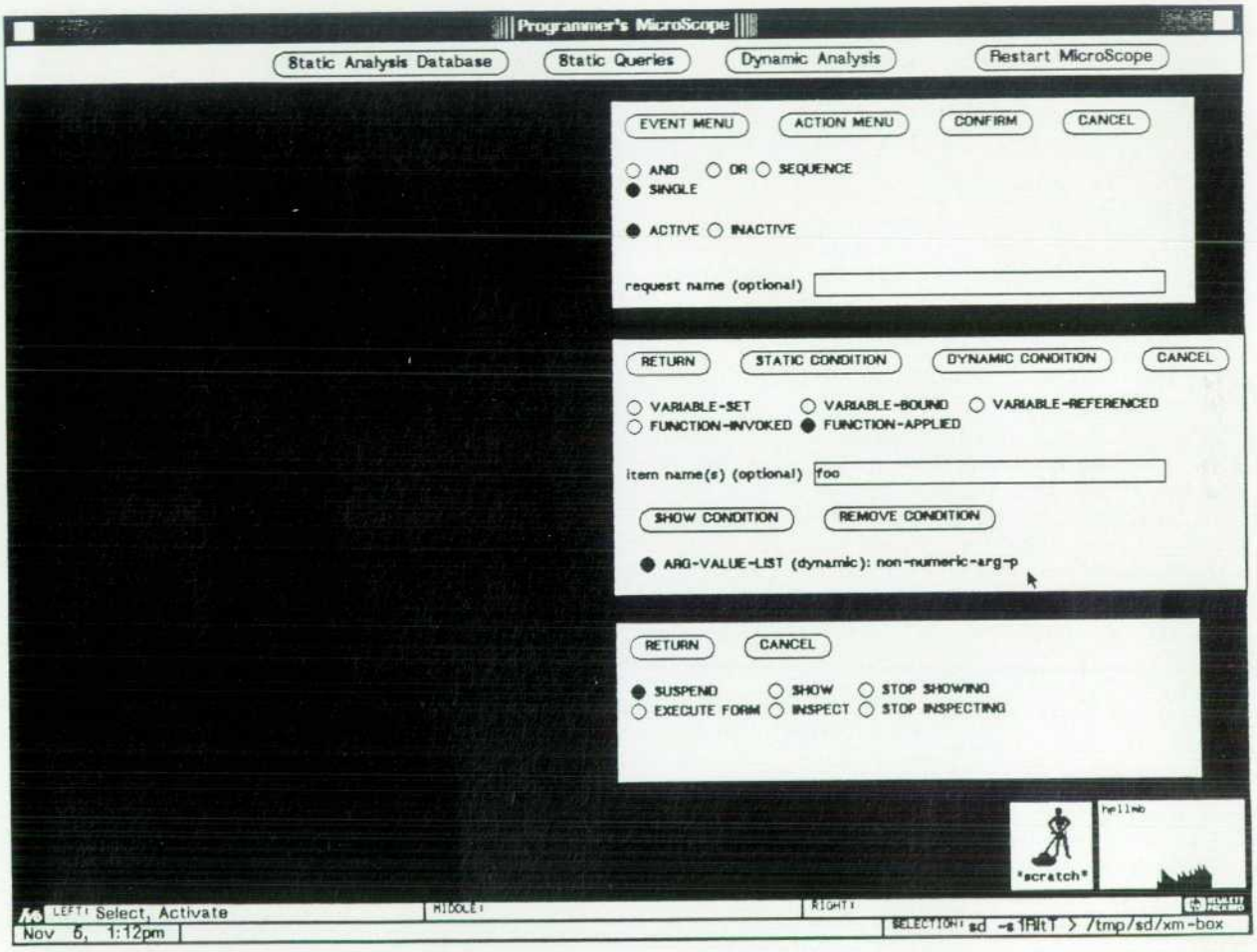


Fig. 1. *An execution monitoring request.*

as a macro or special form, it applies the generic function/macro application template to the expression. The form of this template is:

(use (repeat eval))

The symbol use, meaning function call, is matched against the function name, and the pattern-matcher is applied recursively to the remaining elements of the list (the function's arguments).

If the expression being analyzed is a quoted expression, a keyword, or some other constant, the expression is matched against the template symbol constant. Otherwise the expression is treated as a symbol to be evaluated, that is, a variable reference (matched against the symbol test). Templates for such special forms as setq and let determine when a variable is being set or bound.

More information about templates can be found in the box on page 77.

Execution Monitoring

Events and Actions

Besides providing static cross-reference information about a program, MicroScope allows one to monitor events that occur while a program is running. Monitoring these

responding to each application of the monitored function, but the program will not be suspended unless the condition associated with the monitored event is also met.

When the user confirms this request, it is automatically translated into an HP-RL forward-chaining rule of this kind:

```
IF:     (AND  (?event1 APPLIED ?seq-number2)
              (?event1 NAME FOO)
              (?event1 ARG-VALUE-LIST ?value3))
THEN:   (MONITOR-SUSPEND)
```

Although the request specified a single event, the premise of the rule is a conjunction (logical AND) of clauses. The three terms in each clause (constant or variable) correspond to an event frame, the name of a slot on the frame, and the value of that slot; terms beginning with a question mark are variables. The first clause ensures that the event is a function application (the actual slot value, a sequence number that is unique for each event, doesn't matter). The second clause ensures that the name of the function is correct. The last clause corresponds to the dynamic condition on the function's arguments.

A type restriction on the variable ?event1 ensures that the rule is invoked for the correct kind of event frame. A type restriction on the variable ?value3 causes the predicate that checks whether the function's arguments are numeric to be applied.

If, during the execution of the program, function foo is called, a MicroScope routine attached to the Lisp interpreter creates an HP-RL frame corresponding to this event and stores information about the event on the frame's slots:

```
Frame: {function-call 1} Class: { FUNCTION-CALL}

        NAME:           FOO
        APPLIED:        1
        ARG-VALUE-LIST: (1 2 3)
```

Forward chaining rules are triggered by changes in the frame data base. The assertion (storing) of a slot value on the event frame, in this case when the frame is created, triggers any rule with a premise clause matching that piece of information. To avoid triggering the same rule multiple times, only one such assertion is allowed to trigger a rule (namely, the assertion that records the sequence number).

Since this assertion does match a clause in the rule given above, the rule is triggered with the frame variable bound to the new event frame. The HP-RL rule system then matches the other clauses against the information on the event frame. The system checks whether the value of the name slot matches the one in the rule; it does. The system looks up the value of the arg-value-list slot and applies the specified predicate to it. If the predicate is satisfied, the entire premise succeeds, and the action specified in the rule's conclusion is performed: the program is suspended.

events can help one trace the dynamic behavior of the program and locate the sources of bugs.

The kinds of events MicroScope is currently able to monitor are function calls (either before or after the function's arguments are evaluated) and variable references, bindings, and assignments. As indicated earlier, one can list the specific functions or variables to be monitored by naming them explicitly, but one can also make use of existing cross-reference information to indicate a set of functions or variables that meet a certain statically determined condition. Given that condition, MicroScope determines (at request time) which functions or variables it needs to monitor.

The user can also set dynamic conditions on events—for example, that the function being monitored must have certain arguments or that the variable being monitored must have a certain value. In this case another level of indirection is possible. Instead of specifying arguments or values that the actual run-time values must match, the user can supply a Lisp predicate that the actual values must satisfy. Any value satisfying that predicate satisfies the dynamic condition. The actions associated with an event will not be performed unless the dynamic conditions on the event are met at run time.

Besides specifying individual events, one can also ask MicroScope to watch for a logical or temporal combination of events. In these cases the specified actions will be taken when and if all the events occur (AND), if any of the events occurs (OR), or if all of the events occur in the order specified (SEQUENCE). The AND and OR combinations are not concerned with the order in which events occur, while SEQUENCEs are. The ability to specify the order of events—for example, that a given function sets a variable *after* it is called by another function—allows the user to zero in on critical moments in the program's execution.

Source-level debugging allows programmers to think about bugs at the same conceptual level as they think about the code they write. But MicroScope does more than this; it gives the programmer the ability to specify events declaratively, in terms of high-level combinations and conditions. Given these declarative specifications, MicroScope determines what functions and variables to monitor and, during the program's execution, whether a given combination of events and conditions has in fact occurred.

A graph of the program's execution history, showing the monitored events, is displayed automatically as the program is running. In addition, users can specify various other actions that should be taken when the specified event occurs. They can monitor the value of a variable (or the contents of a complex data structure) as the value changes. They can have the program suspended at an event-determined breakpoint. They can provide an arbitrary Lisp expression to be executed when an event occurs. And, as with events, they can specify a group of actions that should be performed together.

Internal Representation and Implementation

When a monitoring request is made, MicroScope translates the request into one or more forward chaining (i.e., data-driven, condition-action) *rules*. A rule is a conditional

Fig. 5. *Microscope's platform.*

relation between a premise (the condition) and a conclusion. In a forward chaining rule, the conclusion contains an action that is performed when the condition represented in the premise becomes true. MicroScope's execution monitoring rules are currently written in HP-RL, a high-level, Lisp-based representation and inferencing language developed at HP Laboratories.[1]

In general, an execution monitoring rule's premise corresponds to the event(s) specified in a request, and its conclusion corresponds to the specified action(s). Logical combinations of events and (dynamic) conditions are represented by compound premises. Static conditions, which rely on cross-reference information known at request time, are evaluated before the rule is formulated, and the results are incorporated into the rule explicitly.

Temporal sequences of events are translated into multiple rules, one for each event in the sequence. Initially, only the rule corresponding to the first event in the sequence is active. (Only active rules can be triggered.) The conclusion of the rule corresponding to the last event in the sequence contains the action(s) to be taken when and if the whole sequence occurs. The conclusions of the other rules contain an action that will activate the rule corresponding to the next event in the sequence. This representation ensures that the actions associated with the sequence will be taken only if the events occur in the order specified.

The Lisp interpreter has built-in *hooks* that allow one to alter the interpreter's behavior when it evaluates a form or applies a function. Taking advantage of this capability, MicroScope ensures that when a monitored event such as a function call or variable reference occurs, an HP-RL *frame* corresponding to the event is created. (A frame is an object-like complex data structure.) Information relevant to the event, such as a function's arguments or a variable's value, is stored on the frame at this time, and the frame is updated with additional information, such as a function's return value, when it becomes available.

The collection of event frames serves as a memory-resident data base of dynamic information about the program. The Execution History browser allows the user to peruse this information. In addition, the act of recording event information on a frame triggers any rule whose premise contains a clause that corresponds to that event. If the combination of events and conditions in the rule's premise is satisfied according to information in the data base, the rule succeeds and the actions specified in the conclusion are performed. If the conditions are not met, the rule fails and the actions are not performed. But the rule may be triggered again by another event. In the case of a sequence, of course, the success of one rule may activate another, which may itself be triggered at a later time. In the current implementation, rules do not trigger other rules directly.

Why does MicroScope use forward chaining rules to represent execution monitoring requests? First of all, the ability to specify execution events declaratively frees the user from having to worry about how information is retrieved and combined, and the internal rule representation is close to the user's declarative specification.

The argument could be made that the user's high-level specification should be translated into low-level procedural code. One strategy would be to insert a series of conditional tests in the Lisp code attached to the interpreter hooks. But the modularity of independent rules makes them easier to assemble and modify than a monolithic piece of code. Besides, the matching of events and rules limits the number of tests that need to be performed. An alternative strategy would be to attach daemons to the event frames. But rules make it easier to combine information about different events. Multiple daemons associated with multiple events would be harder to control. In general, procedural code, whether distributed in daemons or centralized in a routine attached to the interpreter, needs to use specialized procedures to find and combine the necessary information. A rule-based system, by contrast, can employ pattern matching and backtracking to determine whether complex combinations (specified declaratively) have been satisfied. It is easier to translate a high-level specification into rules and let the system do the work of finding and combining information automatically.*

Conclusion

To maintain a program over a time—to make it workable under new conditions and in new environments—a programmer must have some understanding of the way the program works. MicroScope provides tools that help a programmer understand the structure and behavior of large, complex programs. MicroScope is not a single tool; it is an integrated set of tools that share common data and a uniform user interface. The combination of a graphical, interactive interface, object-based data representation, and rule-based reasoning make MicroScope easy to use but powerful in its capabilities. Fig. 5 shows a schematic model of the platform on which MicroScope rests.

The current MicroScope prototype has been demonstrated within Hewlett-Packard and at artificial intelligence conferences in the U.S.A. and Europe. These demonstrations have generated interest among software developers both inside and outside of Hewlett-Packard, in companies that are customers of HP and in universities. The static component of this prototype has been distributed to selected test sites.

A discussion of MicroScope's long-term goals appears in reference 21.

Acknowledgments

MicroScope is a project of the Software Technology Laboratory of HP Laboratories. We greatly appreciate the work of Jed Krohnfeldt at the University of Utah, who designed and implemented the original MicroScope prototype. MicroScope's graphical user interface was made possible by the Ivo project at HP Labs, which supplied a modern, X-Windows-based environment for us to build upon. Particular mention should go to Brian Beach, who developed the browser construction toolkit which we used to construct our browsers. Martin Griss and Terry Cline have given us many valuable suggestions for improving MicroScope. Alan Snyder and Steven Rosenberg have given us valuable suggestions for improving this paper.

*The rules themselves could be compiled into procedural code, but this feature is not currently available in the rule system used by MicroScope.

References

1. S. Rosenberg, "HP-RL: An Expert Systems Language," *Hewlett-Packard Journal*, this issue, pp. 57-65.
2. A. Snyder, *CommonObjects: an overview*, Technical Report STL-86-13, Software Technology Laboratory, Hewlett-Packard Laboratories, 1986.
3. A. Goldberg, *Smalltalk-80: The Interactive Programming Environment*, Addison-Wesley, 1984.
4. W. Teitelman, *INTERLISP Reference Manual*, Xerox Palo Alto Research Center, 1978.
5. W. Teitelman and L. Masinter, "The interlisp programming environment," *IEEE Computer*, Vol. 14, no. 4, April 1981, pp. 25-33.
6. L. Masinter, *Global program analysis in an interactive environment*, Technical Report SSL-80-1, Xerox Palo Alto Research Center, 1980.
7. P.C. Bates and J.C. Wileden, "EDL: a basis for distributed system debugging tools," *Proceedings of the 15th Hawaii International Conference on System Sciences*, 1982, pp. 86-93.
8. B. Bruegge and P. Hibbard, "Generalized path expressions—a high level debugging mechanism," *Journal of Systems and Software*, Vol. 3, 1983, pp. 265-276.
9. N.M. DeLisle, D.E. Menicosy, and M.D. Schwartz, "Viewing a programming environment as a single tool," *Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments, SIGPLAN Notices*, Vol. 19, no. 5, May 1984, pp. 49-56.
10. H. Carr and R.R. Kessler, *Event-driven program probes (with state)*, Technical Report, University of Utah, 1986.
11. G. Ross, "Integral C; a practical environment for C programming," *Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments, SIGPLAN Notices*, Vol. 22, no. 1, January 1987, pp. 42-48.
12. D.S. Wile and D.G. Allard, "Worlds: an organizing structure for object-bases," *Proceedings of the 2nd SIGSOFT/SIGPLAN Symposium on Practical Development Environments, SIGPLAN Notices*, Vol. 22, no. 1, January 1987.
13. R.M. Balzer, "Living in the next generation operating system," *Information Processing 86: Proceedings of the IFIP 10th World Computer Congress*, Dublin, September 1986, Amsterdam: North-Holland, 1986. Reprinted in *IEEE Software*, Vol. 4, no. 6, November 1987, pp. 77-85.
14. *CLF Manual*, University of Southern California Information Sciences Institute, 1987.
15. S.P. Reiss, "An object-oriented framework for graphical programming," *SIGPLAN Notices*, Vol. 21, no. 10, October 1986, pp. 49-57.
16. S.P. Reiss, "A conceptual programming environment," *Proceedings of the Ninth International Conference on Software Engineering*, January 1987, pp. 225-231.
17. S.P. Reiss, "Working in the Garden environment for conceptual programming," *IEEE Software*, Vol. 4, no. 6, November 1987, pp. 16-27.
18. D. Garlan, *Views for Tools in Integrated Environments*, Carnegie-Mellon University Technical Report CMU-CS-87-147 (CMU PhD Thesis), May 1987.
19. J. Krohnfeldt and R. Kessler, "MicroScope—rule-based analysis of programming environments," *Second IEEE Conference on Artificial Intelligence Applications*, Miami, 1985.
20. F. Hall, "XRLIB: An Overview of the X Window System," *Proceedings of the Fifth Annual Pacific Northwest Software Quality Conference*, October 1987, pp. 254-263.
21. J. Ambras and V. O'Day, "MicroScope: a program analysis system," *IEEE Software*, Vol. 5, no. 5, May 1988.

Reader Forum

The HP Journal encourages technical discussion of the topics presented in recent articles and will publish letters expected to be of interest to our readers.
Letters must be brief and are subject to editing.
Letters should be addressed to:

Editor, Hewlett-Packard Journal, 3200 Hillview Avenue, Palo Alto, CA 94304, U.S.A.

Editor:

This is in response to Mr. David Martin's article on "Software Quality Assurance on the HP Printed Circuit Design System Project" in the February 1988 issue.

We feel that Mr. Martin's article and ones like it continue to make the job of being a software quality assurance engineer a difficult one. It is tough enough dealing with software engineers and managers who express the ideas contained in this article, but when it is printed under the banner of "software quality assurance" by someone who should know better, it just adds fuel to the fire.

Here are some specific rebuttals to points made in Mr. Martin's article. The article constantly referred to the "QA phase." In none of the software development life cycle models that we are aware of is there a "QA phase." QA is not a phase but is an activity directly overlaying the entire development and maintenance cycles. After software is developed it cannot enter into a "QA phase," be magically processed and come out as "quality" software. No product (hardware or software) can have quality inspected or tested into it.

Mr. Martin's statement that "Focusing on quality alone has

the inherent danger of never releasing the product because it is not perfect" is misleading. It might be true if an organization had unlimited resources, but in the real world this never happens. Quality usually takes a back seat to cost and schedule, both of which are adversely affected by poor quality. The industry as a whole, and management in particular, needs to understand this relationship if they ever hope to develop cost effective, quality software.

The QA plan Mr. Martin described seems to focus solely on the software testing activities. A QA plan should define the software quality activities that are performed throughout the software development process. Most QA plans detail the activities performed by a separate QA organization. A software quality organization should provide an independent technical evaluation of all the software products, along with monitoring software engineering's performance of their QA tasks.

The statement "The confidence in the product's quality is in direct proportion to the amount of testing that has been performed on the product" is incorrect. Testing a software product provides confidence in the functions and execution of the software, not in its quality. Software quality consists of several factors, some of which are maintainability, reusability, and flexibility. These listed factors cannot as yet be determined by testing methods, but require human evaluation of the software and the documentation.

We do like the software testing activities described in the article. Independent testing teams are necessary in order to ensure that there is no bias in the software testing activities. We would like to have seen a bit more detail in the description of the actual test activities performed by the independent testing team. Unfortunately, this was not provided in the article.

In summary, then, it is our collective impression that the title of Mr. Martin's article, and his uses of the term "quality assurance" are misleading. A more appropriate title for his article should have been "Software Testing and Error Tracking on the HP Printed Circuit Design System Project." The number of articles published each year that specifically address or relate to the field of software quality assurance is relatively small. Fewer still provide more than a cursory exposition of the subject, with little or no discussion of new techniques, methodology, or tools. For this reason then, the information published should make the best possible use of the limited opportunities for exposure. A point may be made of the limited space available for such articles, and thus the necessary brevity of the discussion. However, this in no way sustains the misapplication of terms or techniques. The understanding on the part of program managers as to the role of the software quality engineer can only be further confounded by such misleading information as was presented in your February issue.

The undersigned represent a group of software quality engineers working in Huntsville, Alabama.

Kevin Preston
Steve Robinson
Tony Peters
Kelly Ford

In response to the concerns about my article, I would like to point out (as was mentioned in Elaine Regelson's introductory article "Developing a Printed Circuit Board Design System") that we were in the situation of attempting to meet the challenge of quickly turning code purchased from an outside source into an HP product. The purchased code was already running, and for us to achieve the benefits of reuse of the code it did not make sense to return to the early stages of the traditional software life cycle model. My article addressed the lessons learned and the techniques tried when we applied the "textbook" ideal situation models to this real-world problem. I do not disagree with many of the comments as they apply to traditional models of software development. Having said that, I would like to respond to some specific points.

I heartily agree that QA is a set of activities spanning the entire software life cycle. The phrase "QA phase" used in the article does not imply that all QA activities are confined to one phase, and I did mention such activities in other phases in the article. In our environment, "QA phase" refers to the stage in our product development cycle where the product about to be released is audited and tested for quality. In some organizations within and outside of HP this phase is called the "system test phase" or the "functional test phase."

My statement that "Focusing on quality alone has the inherent danger of never releasing the product because it is not perfect" was meant to point out the differences between the ideal and real worlds. In the ideal world everyone would produce perfect software. Given that this does not happen, the challenge is to produce quality software on a timely basis. Poor quality certainly has adverse effects on cost and schedule. I do not believe that my statement is a contradiction of this concept. I agree that an independent technical evaluation provided by a separate QA organization is the preferred mode of operation. Unfortunately, such a luxury is often unavailable. In such cases, the software development engineer assumes even more responsibility. My article dealt with a QA plan that specified activities throughout the life cycle performed by the development engineer, the software QA engineer, and others.

It was asserted that the statement "The confidence in the product's quality is in direct proportion to the amount of testing that has been performed on the product" is incorrect. I agree that a product's quality cannot be determined strictly by test execution. In HP we evaluate a product's quality based upon factors including functionality, usability, reliability, performance, and supportability. To judge product quality based on these factors requires both extensive test execution and human evaluation. For instance, we use our human factors group to evaluate the usability of the product, and regression test suites to evaluate reliability and performance. Determination of the reliability and supportability aspects of a product is based upon the evaluation of the defect data collected during testing.

Regardless of which terms are employed to describe activities relating to QA, what is paramount is the customer. We strive at HP to perform activities during all phases of the life cycle that will ensure that the customer receives a quality product.

David Martin

Red AlGaAs Light-Emitting Diodes

HP has recently released indicator and display products containing a new type of red light-emitting diode (LED) based on the aluminum gallium arsenide (AlGaAs) materials system. These LEDs offer a significant improvement in efficiency over the red LEDs that have previously been available but cost only slightly more.

by Frank M. Steranka, Dennis C. DeFevere, Michael D. Camras, Chin-Wang Tu, David K. McElfresh, Serge L. Rudaz, Louis W. Cook, and Wayne L. Snyder

UNTIL RECENTLY, all commercially available visible LEDs were homostructures. That is, they consisted of pn junctions formed in one type of material (see Fig. 1). Over the past few years, new types of LEDs made of several layers of materials having different bandgaps (heterostructures) have appeared on the market. Heterostructure LEDs have several advantages over the standard homostructures, and LEDs made with them offer significant improvements in light output efficiency. The increase in efficiency is the result of the single-sided injection and reduced internal absorption that heterostructures can provide.

In homostructures under forward bias, electrons are injected into the p-type material and holes are injected into the n-type material. Some fraction of these minority carriers then recombines with the majority carriers on the p and n sides of the junction and emits the near-bandgap light characteristic of the LED. The radiative efficiency on the p and n sides is usually quite different and more light could be generated if minority-carrier injection into the less-radiatively-efficient material could be eliminated.

One way of achieving this is to have the pn junction occur at the interface between two materials of different bandgap. One can change the bandgap by changing the alloy composition in the AlGaAs system. This is done by changing the ratio of aluminum to gallium in the compound. The energy diagram for such a situation is shown in Fig. 2b with the homojunction case in Fig. 2a for comparison. The discontinuity in the valance band adds much more to the hole potential barrier than the conduction band discontinuity adds to the electron potential barrier. The band configuration depicted in Fig. 2b is that of a single heterostructure (SH) device, and it effectively eliminates hole injection into the wide-gap n-type material under forward bias and thus provides single-sided injection. The second advantage of this structure is that the wide-gap material is transparent to the light generated on the narrow-gap side of the junction. Hence, there is much less reabsorption inside the material than in the homostructure case.

The reabsorption can be further reduced by putting a second wide-gap layer on the other side of the narrow-gap layer as shown in Fig. 2c. This is called a double heterostructure (DH) device. The second heterointerface prevents the injected electrons from diffusing out of the narrow-gap active layer and gives the light that is generated a better chance to escape. A problem with making such devices, however, is that the wide-gap and narrow-gap materials must have nearly the same lattice constant to avoid dislocations at the material interfaces, which severely reduce the light-generation efficiency. In the AlGaAs system, the lattice constant changes very little with alloy composition, and this makes it a near-ideal system for making such devices. It has been used extensively for the last 15 years or so to fabricate high-efficiency infared LEDs and low-threshold lasers.

The AlGaAs system can also be used to fabricate efficient diodes that emit in the red portion of the spectrum. However, it is quite difficult to grow the high-aluminum-content layers that are required. Liquid phase epitaxy (LPE) is the only technology that can provide material quality comparable to that of the infared devices. It has taken many years of development to create LPE reactors capable of growing large volumes of high-quality multilayer devices. Many of these problems have now been solved and devices of this type are now available. This paper provides an overview of the different types of AlGaAs devices that are available and compares their performance to that of the other red

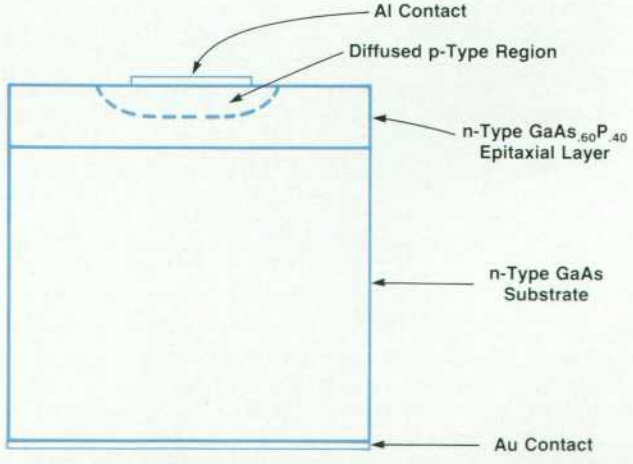


Fig. 1. *Structure of a typical homojunction GaAsP LED. The pn junction is formed by a Zn diffusion into an n-type GaAsP epitaxial layer. Light is generated on both sides of the pn junction and escapes primarily from the top surface.*

LED technologies.

Types of Red AlGaAs LEDs

There are three versions of commercially available red AlGaAs LEDs and schematics of all three are shown in Fig. 3. The device shown in Fig. 3a is an SH LED and consists of at least two layers of AlGaAs on a GaAs substrate. The first layer is the active layer made of $Al_{.35}Ga_{.65}As$. The top layer is a "window" layer made of AlGaAs with an Al mole fraction greater than ≈0.6. The light generated in the active layer can escape out the top and sides of the epitaxial material, but virtually all of the light that hits the substrate is absorbed there.

The SH device is the easiest to grow because it has the fewest layers and good thickness control is not essential. It is also the dimmest type of red AlGaAs chip that is available and is only marginally brighter than the best nitrogen-doped GaAsP (GaAsP:N) devices that will be described in the next section. Since it is the simplest to fabricate, it has been commercially available the longest.

A DH chip is shown in Fig. 3b. It consists of at least three AlGaAs layers on a GaAs substrate. The first layer is a wide-gap injecting layer and the second is a ≈2-μm-thick active layer. Above the active layer is a thick wide-gap confining layer. As is the case for the SH device, most of the internally generated light in the DH chip depicted in Fig. 3b is absorbed by the substrate, so it has been labeled DH-AS for double heterostructure absorbing substrate. However, it is still twice as bright as the SH version because of reduced absorption in the epitaxial material.

To manufacture this structure requires much greater thickness control than is necessary for the SH device, in addition to at least one extra layer of epitaxial material. At HP, we have developed a manufacturing technology that allows us to reproduce this complicated structure in very high volumes. Our chip design incorporates two features to improve manufacturability. One is a very simple post-epitaxial-growth wafer process that requires no photolithography. The other feature is an aluminum top contact for good bondability. The HP technology is capable of producing DH-AS LEDs for only a small increase in cost over the previous-generation SH devices.

The relative efficiency curves as a function of forward current for the SH and DH-AS structures are shown in Fig. 4 along with the curve for GaAsP:N which currently dominates the high-performance, high-volume market. Only the DH-AS AlGaAs chips are significantly more efficient than GaAsP:N. The increase in performance is particularly dramatic at forward currents less than 2 mA where the DH-AS AlGaAs chips are more than five times as bright. Because of this, they are expected to open up many new low-current applications for red LEDs.

The last structure shown in Fig. 3 is that of a DH transparent substrate (DH-TS) device. It is similar to the DH-AS chip in that it consists of at least two wide-bandgap layers on either side of a thin active layer. However, one or both of the wide-gap layers will be grown several thousandths of an inch thick and the entire GaAs substrate etched away. This eliminates the absorption in the substrate and produces the highest-efficiency AlGaAs LEDs—more than a factor of two brighter than the DH-AS LEDs. Unfortunately, growth of the thick AlGaAs layer is technically very difficult and quite costly. Thus, the DH-TS version is presently only used in special low-volume applications.

Comparison to Other Red LEDs

To see the advantages that red AlGaAs LEDs provide, we will summarize here some of the performance characteristics of the other types of red LEDs that are available. Most of the LEDs now on the market are based on the GaAsP alloy system. GaAsP LEDs come in two versions: non-nitrogen doped and nitrogen doped. The structure of the non-nitrogen doped version (which we label here as just GaAsP) is shown in Fig. 1. The first commercially

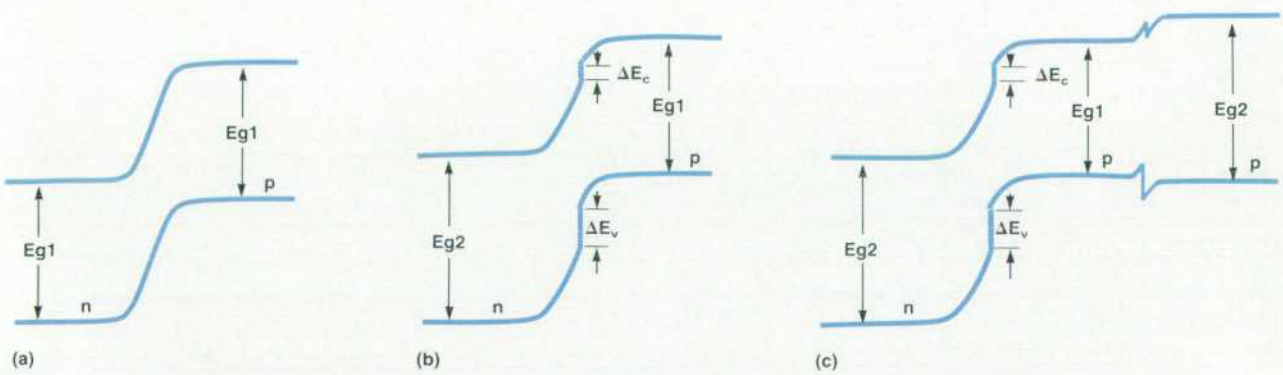


Fig. 2. *Typical energy diagrams for (a) a pn homojunction where the bandgap E_{g1} is the same on both sides of the junction, (b) a single heterojunction (SH) where the bandgap on the n-type side (E_{g2}) is greater than that on the p-type side (E_{g1}), and (c) a double heterojunction (DH) where a second wide-gap layer has been added to the SH in (b). For AlGaAs material in the composition range necessary to obtain visible red emission, the discontinuity in the valance band (ΔE_v) far exceeds that in the conduction band (ΔE_c) which gives rise to electron-only injection in the SH and DH structures under forward bias. The second wide-gap layer in the DH configuration produces a barrier that confines the injected electrons in a thin active layer. This reduces much of the internal absorption and is the reason that a DH LED is the most efficient.*

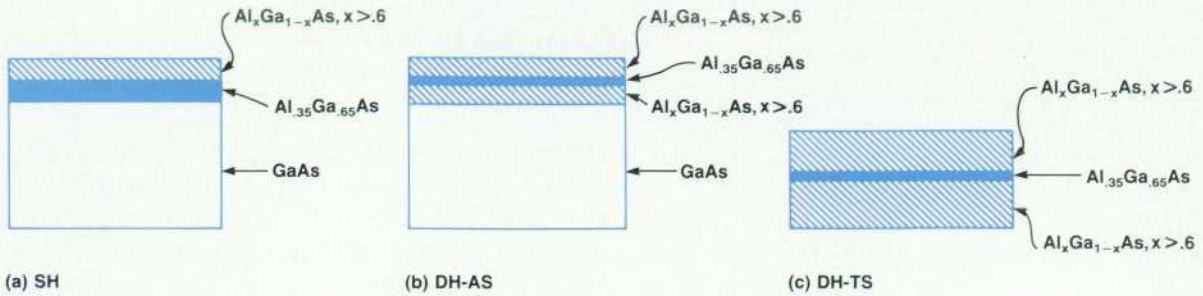


Fig. 3. *The three types of red AlGaAs LEDs that are commercially available.*

available red LEDs were made with this material, and since it is inexpensive to manufacture, it still holds a fair share of the world market. GaAsP is also the dimmest of the red LEDs available.

An order of magnitude increase in photometric efficiency can be obtained by growing GaAsP epitaxial material with a slightly larger bandgap on a GaP substrate (which is transparent to the light generated) and by doping both sides of the pn junction with nitrogen. The nitrogen dramatically increases the radiative efficiency of this material. LEDs made this way are labeled GaAsP:N and now dominate the high-volume, high-performance market. Both versions of GaAsP LEDs are grown by vapor phase epitaxy (VPE) which is amenable to large-scale production.

Between GaAsP and GaAsP:N in photometric efficiency is ZnO-doped GaP (GaP:ZnO) which is grown by a two-step LPE process. First, an n-type layer 10 to 20 μm thick is grown on a GaP substrate. Then a p-type ZnO-doped layer is grown on top. All of the red light emission from this material comes from recombination at the ZnO centers on the p side of the junction. Although this material can be quite radiometrically efficient (>10% external quantum efficiency for an encapsulated chip), the peak emission wavelength is at ≈700 nm where the human eye response is fairly weak. A second drawback to this material is that the ZnO centers are quickly saturated as the forward current through the junction is increased. This causes the overall efficiency to fall rapidly at currents greater than a few milliamperes.

In Fig. 5 we have plotted for comparison the photometric efficiency ranges at a forward current of 20 mA for currently available lamps made using all of the different red LED technologies. They cover almost two orders of magnitude in efficiency with GaAsP at the low end and DH-TS AlGaAs at the high end.

A comparison of growth method, speed, peak emission wavelength, typical forward voltage at 20 mA, and estimated chip production is given in Table I. AlGaAs LEDs have a speed advantage over GaP:ZnO and GaAsP:N LEDs and can emit near 655 nm. Because of their efficiency and these two characteristics, they are excellent candidates for emitters in plastic fiber systems. There is a minimum in fiber attenuation at 655 nm, and the shorter fall time increases the maximum transmission rate. In terms of forward voltage, SH and DH AlGaAs lamps operate at significantly lower voltages than GaAsP:N and GaP:ZnO lamps. This is because of lower substrate and contact resistances, as well as a slightly smaller bandgap, and it gives these lamps an edge in multilamp applications where power consumption is an important consideration.

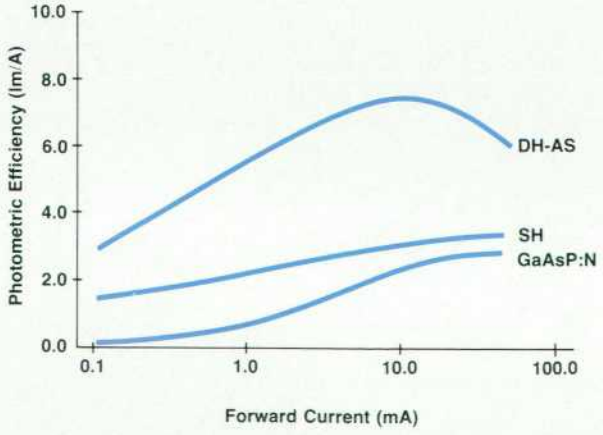


Fig. 4. *Photometric efficiency in lumens/ampere versus forward current for typical SH and DH-AS AlGaAs LED lamps. Also shown for comparison is the curve for a typical GaAsP:N lamp. GaAsP:N now dominates the high-performance, high-volume market.*

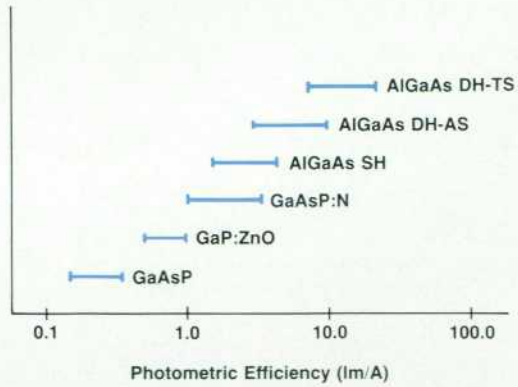


Fig. 5. *Photometric efficiency ranges for plastic lamps made with the different red LED chips at a forward current of 20 mA.*

Table I
Comparison of Red LED Technologies

Material	Growth Method	Speed: 90% to 100% Fall Time (ns)	Peak Emission Wavelength (nm)	Typical Forward Voltage @20 mA	% of Red LED Chip Production
GaAsP	VPE	30	655	1.6V	15-20
GaP:ZnO	LPE	600	700	2.2V	35-40
GaAsP:N	VPE	350	645	2.2V	40-45
SH AlGaAs	LPE	80	660	1.8V	4-5
DH-AS AlGaAs	LPE	80	645-660	1.8V	New
DH-TS AlGaAs	LPE	80	645-660	2.0V	0.2-0.3

The last column in Table I shows an estimate of the breakdown of total red LED chip production by technology. At the present time, AlGaAs LEDs hold a very small part of the overall market even though they have been available for several years. This is primarily because of the expense of the DH-TS version and the fact that the SH devices are only marginally brighter than GaAsP:N LEDs.

An interesting difference between AlGaAs and the other red LED materials is in its degradation performance. The degradation characteristics of GaAsP, GaAsP:N, and GaP:ZnO diodes are generally quite good. The photometric efficiency of these diodes decreases slowly as they are operated, falling typically by ≈15% after 1000 hours at the maximum drive current. Well-made AlGaAs LEDs, on the other hand, actually increase in photometric efficiency as they are used. Data from accelerated aging studies of HP's DH-AS AlGaAs LEDs is presented in Fig. 6, showing that after stress, the average LED increases in efficiency by almost 10%. Such performance, however, requires excellent understanding and control of the fabrication process. Some AlGaAs products on the market have exhibited severe degradation. The increase in efficiency appears to be related to the movement of zinc in the device during operation, since LEDs grown with magnesium instead of zinc as a p-type dopant do not exhibit this behavior.

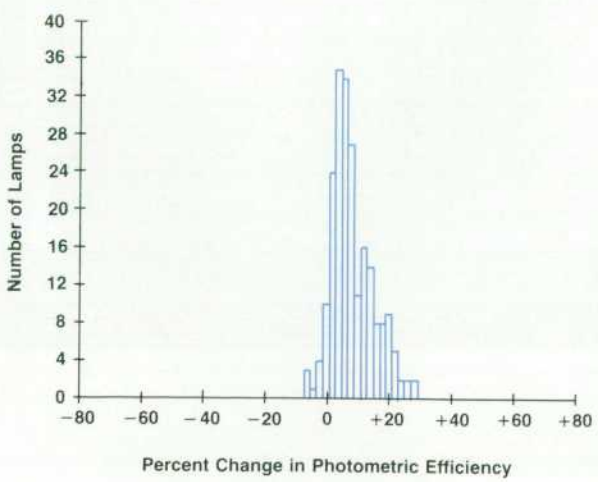


Fig. 6. *Histogram of the percent change in photometric efficiency for 215 HP DH-AS AlGaAs lamps subjected to 1000 hours of 30-mA dc, 55°C stress.*

LED Ratings

Most LEDs are rated in terms of their on-axis intensity at a given drive current. The unit of measure for this parameter is the millicandella (mcd). This number, however, can be misleading when comparing different lamps because it is package dependent. There is a trade-off between on-axis intensity and radiation pattern; lamps made using chips that have the same efficiency can have dramatically different axial intensity (mcd) ratings depending upon the viewing angle. For example, a plastic lamp made using a typical HP DH-AS chip can have any of the mcd/viewing angle combinations listed in the table. For a given application, care must be taken to choose a lamp that has both the on-axis intensity and the viewing angle required.

On-Axis Intensity vs. Viewing Angle
HP DH-AS LED

On-Axis Intensity (mcd)	Viewing Angle (degrees)
1000	8
500	17
250	24
200	30
100	45
60	60

HP DH-AS AlGaAs Products

HP has introduced lamps, seven segment displays, and light bars containing DH-AS AlGaAs chips. The part numbers and product descriptions are:

- High-Brightness Lamps

HLMP-K105	T-1, Undiffused, Untinted
HLMP-K101	T-1, Diffused, Tinted
HLMP-D105	T-1¾, Undiffused, Untinted
HLMP-D101	T-1¾, Diffused, Tinted
HLMP-Q101	Subminiature, Diffused, Tinted

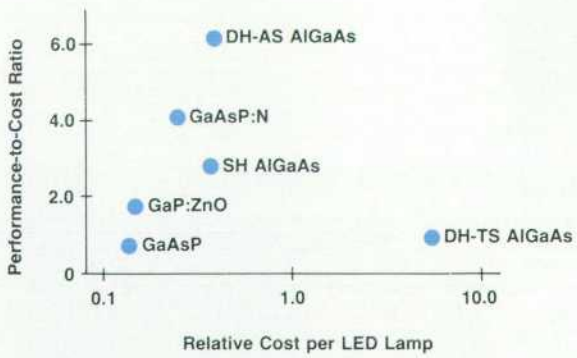


Fig. 7. *Typical performance-to-cost ratio in mcd per unit cost versus relative lamp cost for all of the red LED technologies. Lamps with a viewing angle of 24 degrees measured at 20 mA were used to generate the mcd values.*

- Low-Current (1 mA) Lamps

HLMP-K155	T-1, Undiffused, Untinted
HLMP-K150	T-1, Diffused, Tinted
HLMP-D155	T-1¾, Undiffused, Untinted
HLMP-D150	T-1¾, Diffused, Tinted
HLMP-Q150	Subminiature, Diffused, Tinted

- Very High-Intensity Lamps (750, 1000 mcd)

HLMP-4100	T-1¾, Undiffused, Untinted, 750 mcd
HLMP-4101	T-1¾, Undiffused, Untinted, 1000 mcd

- High-Brightness Seven-Segment Displays (Sunlight Viewable)

HDSP-A15X	0.3 in. Mini
HDSP-E15X	0.43 in.
HDSP-H15X	0.56 in.
HDSP-N15X	0.8 in.

- Low-Current (1 mA/segment) Seven-Segment Displays

HDSP-A10X	0.3 in. Mini
HDSP-E10X	0.43 in.
HDSP-H10X	0.56 in.
HDSP-N10X	0.8 in.

- Low-Current Light Bars (3 mA) and Bar Graph Arrays (1 mA)

HLCP-A100 to H100	Eight Light Bars of Different Sizes
HLCP-J100	Ten-Element Bar Graph Array

Conclusions

In this paper, we have described the three types of red AlGaAs LEDs based on the AlGaAs material system that are now commercially available. We compared their performance relative to each other and to that of the competing red LED material systems. The DH-AS and DH-TS versions of AlGaAs LEDs are significantly brighter than the GaAsP:N LEDs that now dominate the high-volume, high-performance market.

The cost-performance characteristics of the different red LED technologies are summarized in Fig. 7. The DH-AS version of AlGaAs LEDs is comparable in cost with many GaAsP:N devices and has the highest performance-to-cost ratio of any red LED. Because of this, it is expected to become the dominant technology for many high-performance applications. SH AlGaAs and GaAsP:N will share the midrange, with GaP:ZnO and GaAsP sharing the low end. DH-TS AlGaAs is the highest-performance material, but until production technologies can be improved, it will be restricted to specialty applications by cost considerations.

In the near future, we can expect to see new types of "power-signaling" products that will use the efficient DH-AS AlGaAs chips for applications that were previously unthinkable for LEDs. These include automotive tail lights, airport marker beacons, stop lights, and other applications that require an intense source of red light.

Acknowledgments

The authors wish to thank Dr. M. G. Craford, R&D manager, for his support and technical assistance during the course of this project. We would also like to thank Charlotte Garvey, Ann Davis, MaryBeth Lang, Dimokratia Patterakis, Chris Lardizabal, Paul Chow, Su Chew, Rick Pettit, and Rosy Impellitteri for their indispensable help in the growth, processing, and testing of the AlGaAs LEDs.

Hewlett-Packard Company, 3200 Hillview Avenue, Palo Alto, California 94304

HEWLETT-PACKARD JOURNAL

August 1988 Volume 39 • Number 4

Technical Information from the Laboratories of Hewlett-Packard Company

Hewlett-Packard Company, 3200 Hillview Avenue
Palo Alto, California 94304 U.S.A.
Hewlett-Packard Central Mailing Department
P.O. Box 529, Startbaan 16
1180 AM Amstelveen, The Netherlands
Yokogawa-Hewlett-Packard Ltd., Suginami-Ku Tokyo 168 Japan
Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive, Mississauga, Ontario L4V 1M8 Canada

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

5953-8570