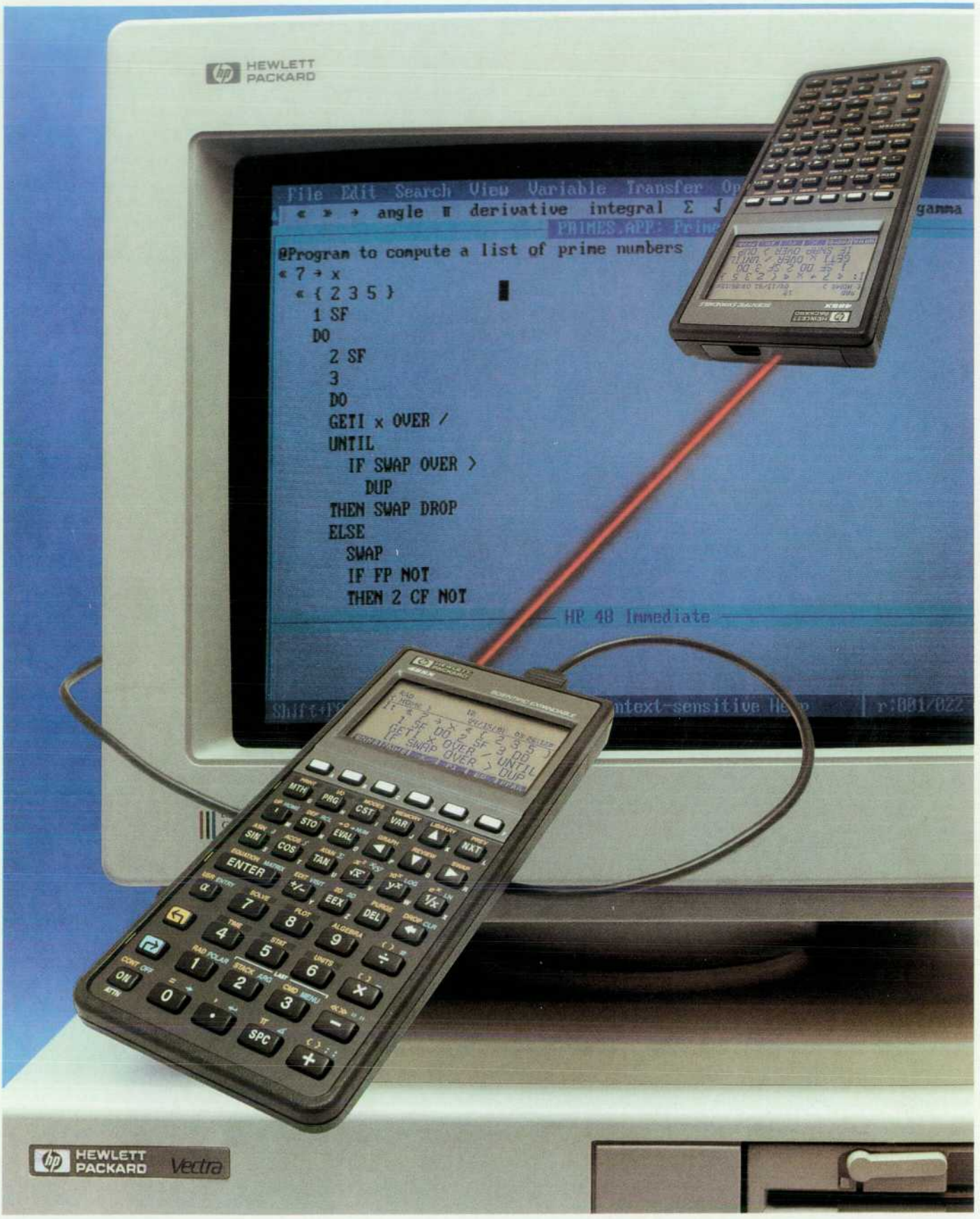
HEWLETT-PACKARD
JOURNAL

JUNE 1991



HEWLETT-PACKARD JOURNAL

June 1991 Volume 42 • Number 3

Articles

Departments

The **Hewlett-Packard Journal** is published bimonthly by the Hewlett-Packard Company to recognize technical contributions made by Hewlett-Packard (HP) personnel. While the information found in this publication is believed to be accurate, the Hewlett-Packard Company 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 presented in recent articles and may publish letters expected to be of interest to readers. Letters should be brief, and are subject to editing by HP.

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

In this Issue

Just glance at an HP 48SX scientific expandable calculator running the EquationWriter application and you realize immediately that there's something different about this calculator. The equation appears in the display almost as it would if you wrote it down on paper. The EquationWriter application is only one of this advanced handheld calculator's many new features, which also include enhanced graphics and plotting, automatic units conversion, two-way communication with PCs and other HP 48SXs, and two expansion slots for additional memory or application software cards. You'll find a more complete list of features in the article on page 6. Also in that article, you can read about the internal software structures and mechanisms that support the feature set and the expansion cards. In the article on page 13, there's a discussion of how the HP 48SX manages multiple applications, followed by a description of some high-level built-in applications: MatrixWriter, EquationWriter, and the graphics and plotting system. The remarkable HP Solve Equation Library application card (page 22) gives you a library of 315 equations, the periodic table of the elements, a constants library, a multiple equation solver, a finance application, and engineering utilities—all on one card! The hardware design of the HP 48SX is described on page 25, and the input/output system is described on page 35. Like some of its predecessors, this calculator has an infrared link to the HP 82240A/B printer, but unlike earlier designs, the HP 48SX's infrared link is two-way, so that a pair of HP 48SXs can use it to exchange data and programs. There's also an optional RS-232 serial cable connection to personal computers. The HP 48SX is produced on the same production line and at the same time as earlier, simpler designs. The article on page 40 tells how.

The traditional analog swept spectrum analyzer converts the input frequency range to an intermediate frequency by mixing the input signal with a swept-frequency local oscillator signal and filtering the result. The effect is the same as sweeping the filter over the input frequency range. By measuring and displaying the filter output, the analyzer shows how the input signal energy is distributed over the input frequency range. The bandwidth of the filter is called the resolution bandwidth of the analyzer and is usually selectable, but there's a trade-off. A narrow resolution bandwidth requires a slower sweep speed. All-digital analyzers, using digital filtering and the fast Fourier transform, offer both higher speed and extremely fine resolution, but only at relatively low input frequencies. The HP 3588A spectrum analyzer combines the traditional swept local oscillator (plus two fixed local oscillators) with digital resolution bandwidth filters and fast Fourier transform techniques. This allows it to make swept measurements four times faster and extremely fine-resolution narrowband zoom measurements hundreds of times faster than traditional swept analyzers. Its upper frequency limit of 150 megahertz is much higher than all-digital analyzers with comparable speed and resolution in narrowband zoom measurements. In the article on page 44, the HP 3588A's designers discuss its principles of operation, its design, and some of its features, such as self-calibration, hypertext help, and adaptive data acquisition.

Computer system diagnostic performance tools are used to diagnose abnormal situations such as poor terminal response time. They can reveal such things as a process that is using an inordinate percentage of CPU time or a disk drive that is too heavily used. Typically, they've been applied by technical experts and not much thought was given to making them easy to use. HP GlancePlus (see page 65) is a family of diagnostic performance tools designed to be easier for novice users to apply. There are three versions, one for each of the MPE V, MPE XL, and HP-UX operating systems. As much as possible, they all have the same look and feel, which is helpful in shops that have more than one kind of system, and they all use exception-based reporting, which keeps uninteresting data off the display so it can't confuse the user.

Process improvement is a hot topic these days. Managers have learned that opportunities for quality and efficiency improvements are more easily identified by looking at a complete process rather than individual steps. At HP, product development is being analyzed as a single process that is jointly managed by marketing, manufacturing, and R&D. As explained in the article on page 71, improvement efforts rely on the concepts of break-even time, postintroduction product reviews, in-process project retrospective reviews, and quality function deployment.

Three papers in this issue are from the 1990 HP Software Engineering Productivity Conference. All deal with aspects of software development. Two describe configuration management systems and the third tells how an HP division built a tightly integrated workstation network for the software development lab. The Domain Software Engineering Environment or DSEE (see page 77) is a configuration management system that's useful for managing software development projects from small, simple ones to large, complex ones. It runs on the Apollo Domain operating system and is in use at over 6000 sites worldwide, including the HP Apollo Systems Division. HP's Imaging Systems Division took a custom approach to configuration management for the HP-UX operating system, basing their system on the HP-UX revision control utility RCS (see page 84). In planning their integrated project support network, HP's Roseville Networks Division had to consider finances, cooperative computing, network architectures, and system administration, including security in the traditionally open R&D environment.

R.P. Dolan
Editor

Cover

Two HP 48SX scientific expandable calculators can use their infrared input/output link to exchange data and programs. There's also a serial RS-232 cable link to a personal computer. The cover photograph illustrates this dual capability (the "infrared beam" is a fake, of course).

What's Ahead

THERE WILL BE NO AUGUST 1991 ISSUE!

The next issue will be October 1991, Volume 42, number 4. It will be produced on our new workstation-based publishing system and will have a new cover design and a different internal appearance.

The October issue will present articles describing the design and manufacture of the HP Component Monitoring System, an advanced, highly modular patient monitoring system for the intensive care unit and the operating room. Also featured will be the design of the memory controller, EISA connector, and BIOS of the HP Vectra 486 personal computer. Other papers will discuss software code inspections and estimating the cost of software defects.

The HP 48SX Scientific Expandable Calculator: Innovation and Evolution

Many of the features of this advanced handheld calculator have evolved from its predecessors, the HP 41C and HP 28S. Others, such as its unit management system, are new.

by William C. Wickes and Charles M. Patton

SINCE THE INTRODUCTION OF THE HP 65 in 1974, Hewlett-Packard has developed a succession of customizable scientific calculators of ever expanding capability. The HP 48SX scientific expandable calculator (Fig. 1) maintains this trend with an unprecedented combination of features and flexibility. Its major features include:

- An RPN-style calculator interface with a dynamic stack of arbitrary depth for operations on eighteen types of mathematical or logical objects (plus twelve additional object types used by system programs).
- Numerous arithmetic, transcendental, and statistical functions, applied uniformly wherever meaningful to complex as well as real numbers.
- Vector and matrix operations on real and complex arrays of arbitrary size. A spreadsheet-like screen editor is provided for simplified entry and editing of arrays.
- Symbolic mathematics including evaluation, expansion, simplification, summation, differentiation, and integration. The EquationWriter application provides graphical, "textbook" entry of expressions and equations.
- String manipulations.
- Binary integer operations, with arithmetic, bit, and byte manipulations, and a variable word size.
- Numerical and symbolic equation solving.
- Eight types of automatic mathematical and statistical plotting, including interactive root finding, calculus, labeling, and digitizing. There are also interactive and programmatic line and arc drawing and creation of custom text and graphics displays.
- An integrated unit management system. Quantities that include physical units can be used in computations, solving, and plotting, while the calculator automatically performs unit conversions and dimension checking.
- Time management, including a clock/date display and appointment and program-execution alarms.
- Two-way communications via a wired serial port for connection to personal computers, printers, and other serial devices or via infrared light for printing to the HP 82240A/B printer or for wireless transfers between two HP 48SXs.
- Customization with plug-in 32K-byte or 128K-byte RAM or ROM memory cards, which may include command libraries for extending the built-in feature set. Libraries can also be imported into RAM using either I/O mechanism.
- A user-definable keyboard and custom menus.
- Programming in the RPL language, which provides program-flow structures, recursion, global and local variables, passing procedures as arguments, input prompting and output labeling, user and system flags, logical tests and operations, and user-defined functions.

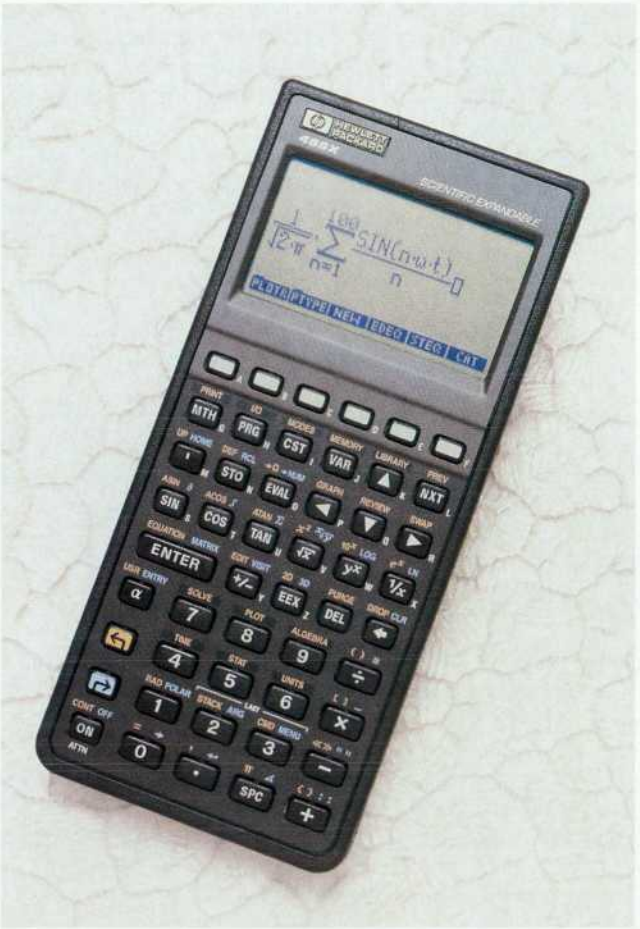


Fig. 1. *HP 48SX scientific expandable calculator, showing the EquationWriter application.*

These features are supported by a hardware set that includes a vertical-format package with 49 keys, a 131-by-64-pixel LCD display with support for fast scrolling of virtual displays that are larger than the physical screen, two plug-in slots for memory cards, a four-wire serial communications port, and an infrared transmitter and receiver (see articles, page 25 and 35).

Design Objectives

The fundamental design objective for the HP 48SX was to create a product that combines the software technology of the HP 28S[1] with the hardware flexibility and customizability of the HP 41C.[2] Although in many respects the HP 28S was itself a descendant of the HP 41C, its advanced capabilities and limited hardware have made its application and range of customers somewhat different from those of the HP 41C. For example, in the academic field, the HP 41C was very popular in college engineering departments, but it had little appeal to mathematics instructors. By contrast, the HP 28S has had a significant effect on mathematics instruction, with many colleges adopting it as a standard teaching tool. Engineering departments have been much slower to adopt the HP 28S, since it does not have the software exchange capabilities they are accustomed to with the HP 41C. Similarly, the HP 41C was very popular with surveyors, but the HP 28S is of limited use in this field because of its lack of I/O capability.

The HP 48SX project started, therefore, with a review of the strengths of its two predecessors. The HP 41C's include plug-in memory ports, HP-IL I/O capability, a redefinable keyboard, and a vertical format convenient for handheld operation. The HP 28S's include extensive real and symbolic mathematical capabilities, RPL operating system and user language, a graphics display, and a menu key system.

At the same time, we focused on common enhancement requests from HP 28S owners. These include a bigger display, more graphics and plotting features, I/O capability, especially for importing or saving software, symbolic integration, and more help from the calculator in using some of its more complicated features.

All of these strengths and enhancements are incorporated in the HP 48SX. In some cases, the implementation of one of these items evolved into a major feature that wasn't necessarily anticipated from the HP 28S/HP 41C combination or a customer request. For example, the HP 28S's powerful numerical integrator was obscured by an arcane syntax for entering the integration arguments. Consideration of this problem in the HP 48SX investigation led to a review of the general problem of entering and recognizing mathematical expressions, which ultimately led to the development of the EquationWriter application (see article, page 13). This solves the integration problem—one enters an integral by "drawing" a textbook-like expression on the screen, including the integral sign, upper and lower limits, and integrand, all appropriately positioned. However, the scope and utility of the EquationWriter far exceed what is needed for this particular use.

The HP 48SX also contains important features that derive more from "next bench" research than from HP 41C or the HP 28S strengths or from customer input. The prime example of these is the HP 48SX's unit management. Simple one-to-one physical unit conversions have been available on calculators for years. Several HP 41C plug-in modules improved on this by providing a general-purpose conversion mechanism which could calculate any conversion factor from input and output units specified as text strings. The *HP 41C Petroleum Fluids Pac* incorporates this mechanism into its calculations so that the user can include units for the values entered for the programs, and ask for answers in particular units. The HP 48SX takes advantage of its multiple-object-type operating system and symbolic manipulations to provide a new level of unit management, in which numerical quantities can have physical units attached to them and carried throughout arbitrary calculations. The collection and cancellation of units and conversions between dimensionally consistent different units are handled automatically by the calculator. For example, a problem such as, "How fast is an object traveling after accelerating at 1 m/s^2 for half a minute, if its initial speed was 20 mph?" reduces to

```
(1_m/s^2)*.5_min+20_mph EVAL
```

on the HP 48SX, which returns 871.1_mph. In HP 48SX notation, the underscore _ acts as an object type identifier linking a floating-point number with a unit expression which can contain arbitrary products, powers, and quotients of physical units. The HP 48SX has 121 units built into ROM, from which the user can construct arbitrary compound units. Unit objects are supported in numerical and symbolic calculations, plotting, equation solving, and integration. This HP 48SX capability removes a great deal of the drudgery from calculations involving physical units.

In one aspect of the HP 48SX design it was not possible to satisfy both HP 41C and HP 28S owners: programming language. To support its other design objectives, the HP 48SX needed to use an RPL operating system and language similar to that used in the HP 28S. Unfortunately, this meant that the considerable body of programs written for the HP 41C would not be executable directly on the HP 48SX. To solve this problem, the plug-in HP 82210A HP 41C emulator card provides a keyboard emulation of the HP 41C and the ability to execute HP 41C programs. The infrared port and the HP 82242A infrared printer module for the HP 41C can be used to transmit programs from the HP 41C to the HP 48SX for execution with the emulator card.

HP 28S users have a smaller problem in program conversion. The great majority of HP 28S commands can be executed without modification on the HP 48SX. Only a few commands are different, primarily those associated with display operations (and the integral command, as mentioned previously), and the various system flags have changed. With optional software, the HP 28S can also use its infrared printer output to "print" its programs to the HP 48SX, where they can be executed after minor or no modification.

Internal Mechanisms

The remainder of this article will discuss some of the mechanisms the HP 48SX uses to support its feature set. The memory maps shown in Figs. 2 through 7 illustrate the concepts discussed in this article. The implementations of many of the higher-level applications are discussed in the article on page 13.

The fundamental basis of the HP 48SX system is the RPL operating system, which occupies about 18K bytes of the system ROM. This system first appeared in the HP 18C Business Consultant calculator in 1986.[3] In brief, the system combines elements of Forth and Lisp, providing a multi-

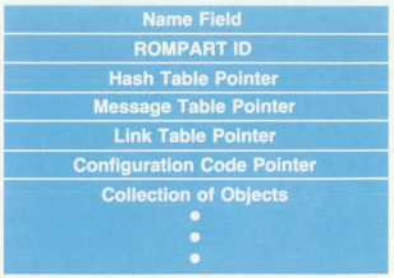


Fig. 2. *Structure of a ROMPART.*

object RPN stack and direct and indirect threaded execution, with both atomic and composite objects, temporary (lambda) variables, and the ability to pass unevaluated procedures as arguments. The objects are similar to Forth words, containing the address of the executable code that defines the object and the data that makes up the body of the object. The object types provided in the initial versions of RPL were:

- Identifier class: identifiers (global names), temporary identifiers (local names), and ROM pointers (XLib names). These objects are used for storing and retrieving other objects.
- Procedure class: secondary (program) and code objects. These objects are executable.
- Data class: floating-point real and complex numbers, character and string objects, hexadecimal strings (binary integers), real and complex arrays, linked arrays, extended precision real and complex numbers, lists, symbolic (algebraic), unsigned short integers, library, RAM/ROM pair (directory). Under normal execution, these objects merely return themselves, as passive data. However, symbolic objects and lists are composite objects, and can be evaluated like procedure class objects.

The body of a composite object is a sequence of other objects terminated by an end marker that serves as a program return if the body is executed as a procedure.

To support HP 48SX operations, several additional data-class objects were added to the above list:

- Graphics object. These are LCD bit maps, used for storing and manipulating graphical images.
- Tagged object. A tagged object contains a text string plus another object. The text is used to label the object. Operations applied to the tagged object ignore the tag and apply themselves directly to the "inner" object. Thus, a program might return the tagged object Speed:10_m/s, where Speed is the tag. Executing 10 • (times) then returns 100_m/s.
- Unit object. This consists of a floating-point real number combined with an algebraic expression representing physical units.
- Backup object. This object is designed for the archival storage of a single object in an independently configured RAM port. The backup object contains a second object plus a name, a length field, and a checksum. The HP 48SX contains commands for storing and retrieving objects from within backup objects when the latter are installed in RAM ports.
- Library data object. This object provides a memory buffer for use by plug-in applications that need to preserve data between executions.

In addition to the new object types, three object types that were present in the HP 28S are given more visibility in the HP 48SX:

- In the HP 28S, a user can create a directory object stored in a variable, but has no access to the directory as an object. In the HP 48SX, a directory has the same status as other objects—it can be recalled to the stack, edited, copied, stored, and so on.
- Built-in commands in the HP 28S and HP 48SX are organized in libraries, which are similar to compiled directories in which the linked list of named objects is compiled to a table-driven organization. Name resolution of the objects within libraries is necessary during parsing, where text names are replaced by ROM pointers. The latter contain indexes into library object tables, which in turn provide for fast location of an object's name and executable code. In the HP 48SX, libraries are available as ordinary objects, so that a user can move libraries in and out of the calculator via one of the I/O ports or on plug-in memory cards. When a library is installed in HP 48SX memory, it extends the HP 48SX's language by adding its own internal commands to the built-in set.
- ROM pointers are visible to the HP 48SX user as XLib name objects, the library analog of the global names that provide access to objects stored in global variables in RAM. Executing an XLib name executes the object within a library that is associated with the name. XLib names

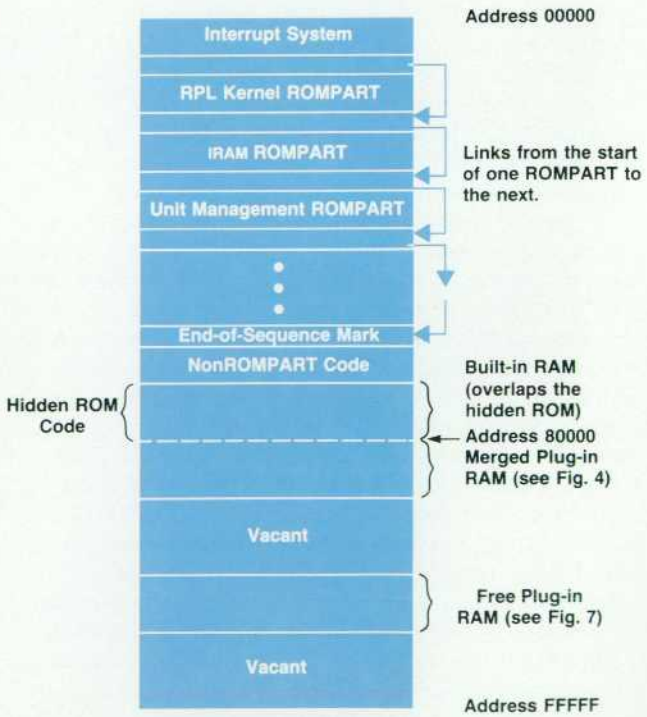


Fig. 3. *Overview of the address space layout of the HP 48SX calculator with one merged and one free (unmerged) RAM card.*

can be compiled as stand-alone RPN objects or included within the definitions of algebraic objects. As long as the relevant library is present, an XLib name decompiles to the text name stored in the library. If the library is removed, the XLib name is decompiled to show the library number and command number within the library.

As part of the strategy to maximize the use of ROM over RAM, the RPL system has included, from its inception, ROM-like structures that are analogs of the user's program and variable space (directories). These are called ROMPARTs. By attaching ROMPARTs to subdirectories, the user can create a context-sensitive customization so that typing the same thing can have very different results depending on the context directory. ROMPARTs were designed to provide context-sensitive customization, localizations of keywords and messages, system extensions, and run-time linking.

In the process of developing a set of programs, the user first creates programs and utilities in a customizing directory. When the programs are debugged and ready to keep, they can be transferred to a ROMPART and loaded into ROM. Attaching the ROMPART to the same directory provides the same functionality with less RAM use.

RPL Plug-in Management

While the overall scope and function of plug-in management did not change from the original RPL definition to its first released implementation in the HP 48SX, a number of design details did change in response to outside reviews of the system. Implementing these changes posed a number of design challenges.

ROMPART Structure. A large portion of the RPL plug-in management design is based on the concept of a ROMPART, which is not a standard RPL object in the same sense as complex numbers, directories, programs, and so on, but is instead a set of data-structure conventions.

A ROMPART is a collection of RPL objects together with a field containing the name of the ROMPART, a ROMPART ID number which uniquely identifies the ROMPART, a pointer to a hash table for use in identifying objects by name, a pointer to a link table for use in identifying objects by their unique object numbers, a pointer to a message table containing messages specific to this ROMPART, and a pointer to configuration code which is executed whenever the ROMPART needs to be configured (see Fig. 2).

The name field can contain any characters and is used only as information for the user. The ROMPART ID number uniquely identifies the ROMPART to the system and is in the range 000-7FE (hexadecimal). ID numbers 000-0FF are reserved for the RPL kernel and other built-in ROMPARTs. ID numbers 700-7FE are reserved for use by ROMPARTS providing application language extensions.

The hash table provides a two-way correspondence between names and objects in the ROMPART. It is used during the process of interpreting the characters typed in by the user to determine whether the characters name any object in the ROMPART, and then again to determine if an object should be displayed as a name rather than according to its internal structure.

The link table provides a list (in object-number order) of all accessible objects within the ROMPART.

Configuration. A ROMPART simply residing somewhere within the system does not provide for any of the services described above. The ROMPART needs to be registered with the system during the configuration process, which occurs in several stages.

Whenever the machine is first turned on, a routine check is made to see if any cards have been plugged in or removed from the system and adjustments are made to compensate for the changes. Then a number of known areas are scanned for the presence of ROMPARTs and a list of currently extant ROMPARTs and their locations is made and compared with the previous list. If no change is detected, the system continues the normal process of turning on the display and resuming the state it had when it was turned off.

On the other hand, if a change is detected, the system performs its warm-start code. Among other things, the warm-start code resets any pointers that could be referencing ROMPARTs that are no longer present. This includes the data and return stacks, updateable system pointers, and the ROMPART pointers connected to the home directory. The system then rebuilds the table of ROMPARTs and their

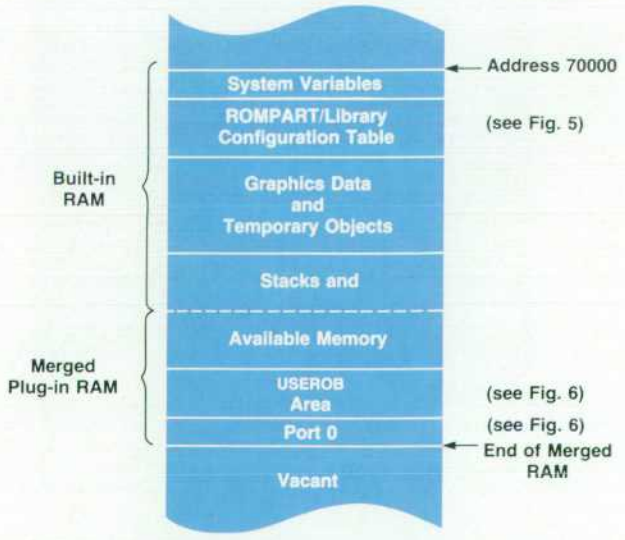


Fig. 4. *Overview of the layout of HP 48SX main RAM with one merged RAM card.*

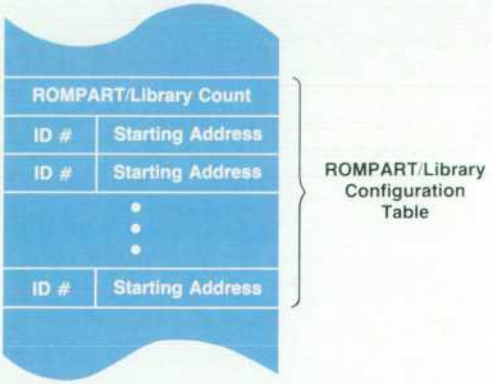


Fig. 5. *The ROMPART/library configuration table. Whenever the HP 48SX turns on, all ROMPARTs and libraries in ROM, port 0, and other unmerged plug-ins are registered here.*

locations and restarts RPL execution.

ROM Poll. One of the first things done when RPL execution is restarted is to proceed through the current list of ROMPARTs, executing each ROMPART's configuration code in turn. At this point, the RPL system is in a valid, stable state and the full resources of the system are available for use by the configuration code.

Although a ROMPART can take over the system at this point (as is done, for example, by the HP 48SX demo ROM), typical tasks are much less ambitious. More typical examples of tasks done at the ROM poll include:

- Attaching a ROMPART to the home directory so that the names of objects within the ROMPART are universally recognized.
- Replacing the hash and/or message tables of other ROMPARTs with versions localized for a particular language.
- Creating a custom subdirectory structure for use with this ROMPART.

ROM Pointers. The RPL system's ROM pointer (ROMPTR) objects provide a name and location independent method of specifying an object within a ROMPART. The data in a ROMPTR gives both the ROM ID number unique to a ROMPART and the particular object's object number.

As long as a ROMPART has been registered as being present in the system, and independent of whether it is attached to any directory, ROMPTRs referring to a ROMPART can be converted to the object or objects they specify. In this process, the current address of the ROMPART whose ROM ID is specified in the ROMPTR is found in the ROMPART table, and the ROMPART's link table is found. The object number specified by the ROMPTR is used as an index into this table to find the actual current address of the specified object.

ROMPTRs occupy a middle ground in terms of execution speed and flexibility between address pointers, which require no resolution but must be updated whenever memory moves, and ordinary identifiers, whose value can change in the course of execution but must be resolved by searching through the current context. Every programmable function and operation in the HP 48SX has an associated ROMPTR that specifies it. However, these are not normally used in programs since the address pointers will suffice. There is one case in which these ROMPTRs must be used, however. If the user stores a programmable function in a variable, what is stored is actually the corresponding ROMPTR, since storing an address pointer is contrary to the RPL conventions, and storing a copy of the object is clearly not what is desired.

ROMPTRs are normally created in the process of converting typed-in text to RPL objects (parsing). If the currently considered piece of text is not an object delimiter, number, or other fixed-syntax item, it is considered to be a name whose meaning is determined by the current context.

Names, ROMPTRs, and Localization. To determine the current interpretation of a name, the system first searches through all the variables in the current directory. If the name matches any one of these, the name is determined to be a variable name (ordinary identifier). If not, the system searches through the hash tables of the ROMPARTs attached to the current directory, if there are any. If a match is made, the name is determined to be a ROM word name, and it is converted to the corresponding ROMPTR. If no match is made, the search is continued with the parent directory, and so on. If the home directory is searched without a match, the name is determined to be a formal variable (also an ordinary identifier).

Every directory except the home directory can have at most one ROMPART attached to it. The hash table used in searching such a ROMPART is the one supplied with the ROMPART. The home directory, on the other hand, can have multiple ROMPARTs attached to it. Recorded with any ROMPART attached to the home directory is a pointer to its hash table. This allows the hash table provided by a ROMPART to be superseded by another hash table either in RAM or in another ROM (localization). Only ROMPARTs attached to the home directory can be so localized.

Structure of Plug-in Modules. The original RPL design provided for two kinds of plug-in modules: one associated with read-only devices (ROM) and one associated with read/write devices (RAM). Whenever a ROM device was detected, it was assumed that its data consisted of a linked list of ROMPARTs. The devices would be configured at some convenient but otherwise arbitrary address and the individual ROMPARTs would be registered as described previously. Whenever a new RAM device was detected, it was assumed that the device contained no viable data and the device would be configured to be a contiguous segment of the system's overall RAM, with current RAM contents

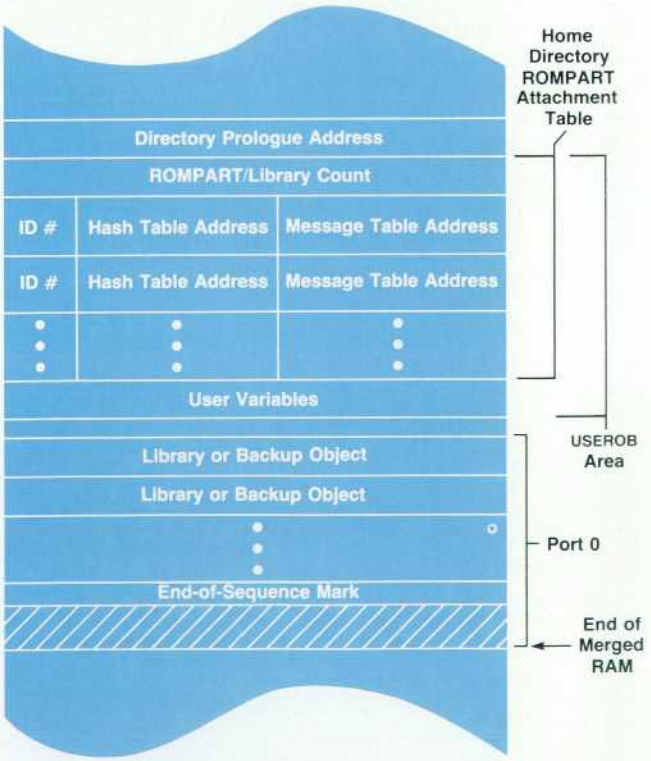


Fig. 6. *Layout of the* USEROB *area and port 0. A library or ROMPART attached to the home directory ROMPART attachment table, either by the* ATTACH *command or during the ROM poll, will have its keywords recognized by the system and can have both its keywords and its messages localized (for example, translated into other languages).*

shifted to new positions as necessary. This process is known as merging RAM. After RAM has been merged, it is not possible to unplug the module without endangering the system integrity. When a merged RAM is pulled, a large area of the system and user variables could go with it.

To make it possible to have ROMPARTs without read-only devices, an area within system RAM is set aside and given the same structure as a plug-in ROM, that is, a sequence of ROMPARTs. This RAM-based ROMPART area is also searched during the configuration of ROMPARTs. This model for plug-in structure provides almost all available services automatically and requires only five user-level commands: to prepare the system for removal of a RAM module (FREE) by reversing the procedure used to merge the RAM module, to attach and detach a given library from a given directory, to include a ROMPART (given in some object-coded form) in the RAM-based ROMPART area, and to remove such a ROMPART.

Design Changes and Challenges

In the evolution of the HP 48SX design, it became apparent that RAM modules needed to be used as mass-storage devices as well as system RAM. By analogy to flexible disks, one would expect that such a mass-storage RAM module could be removed without first informing the system. These two tenets had significant impact on the HP 48SX plug-in management design. Other factors that affected the design were that the RAM modules have a switch that allows them to act as ROM, that there is no effective way to determine the size of a ROM module, and that the system cannot reliably detect the removal of a plug-in as it is happening.

Backup Objects and Libraries. To use a RAM card as mass storage, we need to be able to store name/object pairs in the RAM card much as they are stored in variables in the main RAM. In addition, we need to be able to verify that the data on the card has not been corrupted in some way. This verification stage must be fast because it must happen at configuration time, that is, between the time the machine is turned on and the time the machine is available for use. Since no stand-alone object consisting of a name/object pair existed in the original RPL system, a new object type, the backup object, was invented for the purpose. A backup object, in addition to its prologue and length, consists of a name, an object, and a checksum.

Since ROMPARTs can coexist with backup objects in a plug-in, they are also encapsulated with a prologue and a checksum to become library objects.

The organization of the data in a ROM plug-in is largely dictated by the fact that the system can only determine the beginning of a ROM and not the end. This means that any data structure within the ROM must start at the beginning address and extend from there. The original RPL configuration assumed just such a structure, so that converting the configuration from a linked sequence of ROMPARTs to a sequence of backup and library objects was relatively straightforward. The system determines the end of the sequence when it finds either an end-of-sequence mark, an object that is not a backup or library object, or an invalid checksum. In either of the last two cases, the user is warned of Invalid Card Data, but no further remedial action is taken.

Since RAM plug-in cards can be converted to the equivalent of ROM cards by simply changing a switch setting on the card, we decided that the structure of an unmerged plug-in RAM card should be the same as a ROM card, that is, a sequence of library and backup objects with the sequence starting at the lowest address of the card.

Ports. The RPL directory structure is one of the most tightly integrated aspects of the system. Having been conceived of as semipermanent storage which could dominate the use of free memory in a memory-limited system, it is implemented as a self-contained unit containing no pointers that need to be changed as other parts of memory change. In addition, it is relegated to the high-address end of free memory.

This highly integrated structure with no provision for referring outside itself precludes the inclusion of unmerged RAM cards as virtual subdirectories of the home directory. We decided to extend the mass storage analogy further and have separate data storage space locations analogous to flexible disk drives. Instead of drives A, B, and C, we have ports 1, 2, and 0. Ports 1 and 2 refer to the data contained in cards plugged into the corresponding plug-in slots. Port 0 refers to an area in built-in RAM that acts like a permanently plugged-in card (see Fig. 6). Unlike personal computer mass storage, however, the current drive is never any of these. The port specification must be included in the information given to any operation involving the ports.

The normal STO, RCL, and PURGE commands, which normally store, recall, and delete variables in main memory, are extended to allow transfer of information to and from the ports. Tagging a name argument with :0:, :1:, or :2: indicates to these commands that a port operation is desired. For example, if ABC is the name of an object in port 0, then :0:ABC RCL will return the object to the stack.

Since the only kinds of objects allowed in a plug-in data area are backup and library objects, any other kind of object is first encapsulated as a backup object. Similarly, RCL of one of the backup objects will pry the object out of the capsule.

Directory Management Extensions. The fact that the current drive is always none of the ports has several conse-

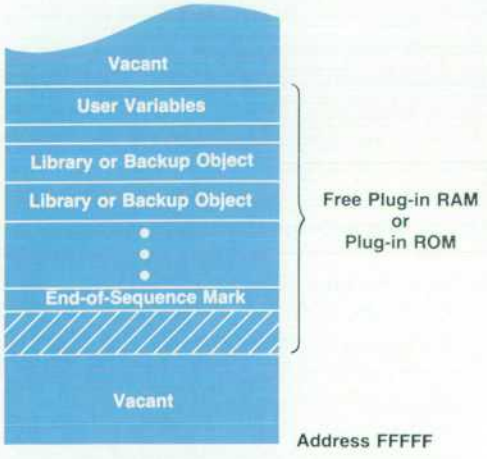


Fig. 7. *Layout of an unmerged (free) plug-in RAM card within the HP 48SX address space. An unmerged plug-in card is either port 1 or port 2 and has the same structure as port 0.*

quences. First, simply transferring a working set of related programs from a directory to a port will not result in a working set of programs in the port. This is because of the port-specific reference for names. If the program calls a subprogram by including its name, say SubProg1 EVAL, then only the directory is searched when the name is to be evaluated. On the other hand, if the subprogram is called by including a port-specific reference, say :0:SubProg1 EVAL, then only the specified port is searched. To compensate for this, we have included a "wild card" port specifier, :&:. The sequence :&:SubProg1 EVAL will search for the subprogram in all ports and then in the current directory before giving up. This calling sequence allows programs to be executed from directories or ports interchangeably.

A second consequence of there being no current drive is the lack of access to objects contained in directories stored in a port. Normally, the method of accessing an object in a directory is first to change context to the directory and then to use RCL or some other command. Since it is not possible to change context to a directory stored in a port, this method is not available without first copying the directory back to main RAM. This is solved by using a list as a complete path specifier for recalling a variable. For example, if A is a directory object stored in port 0 and it contains a variable B, then :0:{ A B } RCL will recall the contents of B to the stack. Similarly, it is possible to recall from any location without changing the context directory by using a list to specify the path completely.

Archive and Restore. With a method of mass storage available, it is natural to provide a means to archive the current contents of the calculator and later to restore this information. The operation ARCHIVE produces a copy of the entire home directory encapsulated as a backup object. It delivers this copy either to a specified port or to another machine.

Restoring from a backup copy of the home directory using the RESTORE operation reverses the archive process. Because of the potentially greater need for human intervention, RESTORE will not automatically restore from another machine. Instead, RESTORE will use a backup (or any other) copy of the home directory no matter how it was obtained.

RAM Recovery. With the large amounts of data that can be present in the machine, it is clear that additional data safeguards are necessary. One such safeguard is provided by the Recover RAM? operation.

Whenever it is found that the structural integrity of RAM has been violated, the user is given the opportunity to either start with a clean slate or attempt to salvage some data. If the user chooses to salvage data, the machine first searches through RAM, locating library or backup objects whose checksums are valid. It collects all of these into a new port 0.

It then searches for a directory object having the specific features of the home directory. If one is found, the RAM recovery operation verifies its structural integrity, and the operation is complete. To check the directory's structural integrity, the RAM recovery operation checks the structural integrity of each object within the directory (including recursively checking subdirectories) and removes any that are corrupt. If no home directory is found, the RAM recovery operation begins searching for ordinary directory objects. When it finds a directory it checks the structural integrity of each object within the directory (including recursively checking subdirectories) and removes any that are corrupt. The resulting corrected directories are named D.0, D.1, and so on, and are gathered together to form a new home directory, completing the recovery process.

Acknowledgments

The design, development, and testing of the HP 48SX firmware ultimately involved almost everyone in the Corvallis Division R&D department. In addition to the authors, those who contributed directly to the design and implementation were Ted Beers (application and interface management), Stan Blascow (BIOS and time management), Diana Byrne (plotting and EquationWriter), Gabe Eisenstein (EquationWriter), Grant Garner (decompile and ROM switching), Bill Johnson (MatrixWriter), Max Jones (menu system and editing), Paul McClellan (unit management), Pat Megowan (application interfaces), Nathan Meyers (I/O), and Bob Worsley (I/O and printing).

References

1. W.C. Wickes, "An Evolutionary RPN Calculator for Technical Professionals," *Hewlett-Packard Journal*, Vol. 38, no. 8, August 1987, pp. 11-16.
2. B.E. Musch, J.J. Wong, and D.R. Conklin, "Powerful Personal Calculator System Sets New Standards," *Hewlett-Packard Journal*, Vol. 31, no. 3, March 1980, pp. 3-12.
3. C.M. Patton, "Symbolic Computation for Handheld Calculators," *Hewlett-Packard Journal*, Vol. 38, no. 8, August 1987, pp. 21-25.

HP 48SX Interfaces and Applications

The HP 48SX scientific expandable calculator provides support for multiple applications, both built-in and externally developed, with customized user interfaces. The EquationWriter and interactive plotting are two of the built-in applications.

by Ted W. Beers, Diana K. Byrne, Gabe L. Eisenstein, Robert W. Jones, and Patrick J. Megowan

LIKE ITS PREDECESSOR THE HP 28S, the HP 48SX scientific expandable calculator is an RPN calculator designed as an electronic scratchpad for mathematical calculations. However, the simple user interface used in the HP 28S would have become overloaded if translated directly to the more capable HP 48SX. Consequently, the HP 48SX contains direct support for developing specialized user interfaces that can replace or extend the basic calculator interface. The support is used in the built-in applications such as the EquationWriter and interactive plotting, and is available for ordinary user programming and for externally developed applications. In this article, we will review the support mechanisms and give several illustrations of their use.

Managing Multiple Applications

Early in the development of the HP 48SX, it became apparent that without a common approach to application interface implementation, the calculator would not present a consistent methodology to the user. For example, when an application ends, it is important that the menu displayed before the application was started be restored. If one application restored the previous menu while another always displayed the MATH menu, user confusion would result.

Although a consistent approach to application interface design is important, so is the freedom of the designer to incorporate unique features that justify the need for a special interface. One of the challenges in developing the application interface engine for the HP 48SX was balancing consistency of operation with flexible design components. For the basic, stack-oriented, RPN operation of the HP 48SX, and for stack-oriented applications such as statistics, the user interface is handled by the built-in RPL outer loop. All other applications use an RPL tool called the parameterized outer loop, which is designed to customize a user interface.

The designer of an application can be expected to know how its interface should operate, but not necessarily to know or fully understand how the application should handle the application from which it was started or how to respond consistently to fatal error conditions and other unexpected events. The parameterized outer loop relieves the designer of these burdens while providing a common, robust method for handling application startup, application shutdown, and asynchronous event handling. The parameterized outer loop accomplishes this by handling the following major aspects of calculator operation:

- Saving the previous application's user interface
- Updating the application's display between key presses
- Waiting for and dispatching key presses, alarm interrupts, and unhandled errors
- Exiting the display and key handling loop
- Restoring the previous application's user interface.

Except for saving and restoring the previous application's user interface, the application-specific components of each step are specified by the application when it starts the parameterized outer loop.

Before the user can interact with an application such as the Equation Catalog, the application must set its user interface. The user interface is what makes the interaction with the application unique. For example, in the Equation Catalog, the familiar stack display is replaced by a list of equations, and the ▼ and ▲ keys no longer move the character cursor but instead move a list pointer around the equation list. An application sets these and other aspects of its interface when started by passing a set of user interface parameters to the parameterized outer loop. These parameters define how the application manages the HP 48SX display and keyboard and how the application interacts with the rest of the calculator environment.

Parameterized Outer Loop Operation

The operation of the parameterized outer loop can be summarized as follows:

```
Save the system or current application's user interface
If error in
   { Set the new application's user interface
     While exit condition object evaluates to FALSE
        { Evaluate display object
          If error in
             Read and evaluate a key
          Then
             Evaluate error handler object
        }
   }
Then
   Restore the saved user interface and error
Restore the saved user interface
```

The application specifies the unique operation components, such as the exit condition object and the display

object, when it starts the parameterized outer loop. This is how the application customizes the interface. The parameterized outer loop is responsible for the key-display loop, alarm interrupts, and low-level error handling.

Display Handling. There is no default display in the parameterized outer loop. The application is responsible for setting up the initial display and for updating it. The application display object is the method by which the application manages the HP 48SX display. This object, which is usually an executable program, can take advantage of the two main methods of displaying information that the HP 48SX supports: *passive* display update and *active* display update.

Passive Display Update. Passive display update involves using the display object to update any area of the display that needs to be changed after a key is handled. In this display handling model, each key is responsible for implicitly passing information to the display object regarding what areas of the display it hasn't changed. The display object then updates all other display areas.

Since the main outer loop itself uses this display update scheme, applications that use many standard keys, such as MatrixWriter, take advantage of the display update information passed by the standard keys to simplify their own display and key handling logic.

A major benefit of passive display update rests on the fact that the application programmer can make no display-related assertions at all in key handling, and still the display handling will work properly, albeit more slowly than necessary. As the application develops, the programmer can add assertions to those keys that do not affect certain display areas, thus saving time during display update. If the programmer misses a few combinations of key-display interaction, the application stills operates properly.

Active Display Update. The second method supported by the parameterized outer loop for handling display update is the more conventional active display update. In this model, each key that affects the display updates the display itself. With active display handling, the application display object can be reduced to a simple NOP (no operation). The major drawbacks of active display update are that all aspects of display handling must be considered by every key definition, and the implicit display update information required by other calculator resources must be determined whenever these resources are used by the application.

For consistency and robustness, most HP 48SX applications manage the display in the same manner as the main outer loop, namely with passive display update.

Hard Key Assignments. Any of the HP 48SX keys, in any of their six planes (unshifted, left-shifted, right-shifted, alpha-unshifted, alpha-left-shifted, and alpha-right-shifted) can be assigned for the duration of a parameterized outer loop application. The key object parameter specifies the keys to assign and their new assignments. In addition, there are two flag parameters that control how keys not assigned by the application are handled. If a key is not assigned by an application, and the allow default keys flag is TRUE, then standard or default key processing occurs, according to the do standard keys flag.

For example, if user keys mode is on and the key has a user key assignment, then the user key is processed if do standard keys is FALSE, or the standard key is processed if do standard keys is TRUE. If allow default keys is FALSE, then all nonapplication keys beep and do nothing else.

Menu Key Assignments. An application can specify any initial menu key assignments, in any of three planes (unshifted, left-shifted, and right-shifted), to be initialized when the parameterized outer loop is started. An outer loop parameter specifies the definition object for the application's menu, and may indicate that the current menu is to be left intact. When the outer loop is exited, the previous menu is restored automatically.

Since hard key assignments have priority over menu key assignments, it is possible to define more exotic behavior for the menu keys. To date, no parameterized outer loop application does so, however, since the menu key handling is very flexible and customizable itself.

Preventing Suspended Environments. Many applications need to allow arbitrary commands and user objects to be evaluated, but may not want the current environment to be suspended by the HALT and PROMPT commands. A parameterized outer loop flag specifies whether any command that would suspend the environment instead generates a HALT Not Allowed error. Since both HALT and PROMPT actually restart the main outer loop, which leaves the application suspended indefinitely without protection for its global resources, all current applications disallow suspension.

Nesting Applications. One of the powerful features of the HP 48SX is its ability to stack or nest multiple application user interfaces, effectively allowing an application to run within another application. For example, while working within MatrixWriter, one can press ↑ STK to start the interactive stack application to copy a value from the stack into MatrixWriter. Conversely, within the interactive stack, one can select a matrix and press VIEW to start MatrixWriter. In both cases, when the second application is finished, the first resumes where it left off. The parameterized outer loop makes sure all the details are sorted out.

Application Examples

The interactive stack is an HP 48SX application with which one can browse through the HP 48SX data stack and perform a set of stack-related operations based on the selected stack levels. Since the interactive stack is designed for stack operations only (including some object editing operations), it maintains strict control over the keyboard and display. This is accomplished with its key handling and menu objects. Unlike most applications, the interactive stack presents a different menu depending on how it is started. When an edit line does not exist, a full menu of operations is displayed. When an edit line does exist, the interactive stack displays a more restrictive menu, reflecting the fewer operations available. To implement this difference, the interactive stack passes one of two menu objects to the parameterized outer loop as its menu specification.

MatrixWriter is an HP 48SX application that simplifies the entry of matrix objects. Like the interactive stack, MatrixWriter controls certain keys that are redefined for its environment, such as ▲. Unlike the interactive stack, however, MatrixWriter allows all undefined keys to operate normally, since many standard key definitions, such as +, are useful in MatrixWriter.

Both the interactive stack and MatrixWriter use the passive display update method for managing their output. In the case of MatrixWriter, this is especially useful and important, since the standard edit line interface is used extensively within the MatrixWriter environment.

Customization by the User

The standard keyboard and display of the HP 48SX are designed for general use, offering direct access to numerical computation and indirect access to other features. For users who want direct access to features of their own choosing (or creation), the HP 48SX has a number of tools for customizing the user interface. The user can redefine keys, define a custom menu, customize how key definitions are executed, and maintain a variety of customized environments.

In the HP 28S, key definitions are objects of a special form. For easier customization, we changed the HP 48SX so that any object can be a key definition. For example, the user can assign the string 5 and the function + to keys, and those keys will act the same as the normal **5** and + keys.

For each key, the user can assign an object in one of six key planes: keys can be unshifted, left-shifted, or right-shifted with alpha on or off. If key assignments are viewed as yet another shift, this makes 12 key planes in all. The user can enable or disable the current assignments by pressing **↩USR**, or by setting or clearing a flag. The assignments can be recalled as a list of alternating key codes and objects, and such a list can be used to make assignments.

Another way to define keys is the custom menu. After storing a list of objects in a variable named CST, the user can press **CST** to put the first six objects in the menu, **NXT** to put the next six objects in the menu, and so on. This method doesn't involve the key assignments described above; rather, it uses the standard key definitions that make the menu system work.

The objects in the custom menu are given the same shifted interpretations as in built-in menus. For example, a name is executed, stored into, or recalled, depending on whether the key is unshifted, left-shifted, or right-shifted, just as in the VAR menu. Units are multiplied, converted, or divided, just as in the UNITS menu. Alternatively, the user can specify separate objects for the menu label and for unshifted, left-shifted, and right-shifted actions.

The most radical customization is called vectored ENTER. When the user presses a key in normal operation, the corresponding object is either written to the command line or executed. In the latter case, the text already in the command line must be parsed and executed first, and then the key-definition object is executed. The user can customize two steps in this process by storing programs in variables αENTER and βENTER.

The program in αENTER takes over parse-and-execute responsibilities. Such a program might either (1) print the command-line text and then execute OBJ→, which parses and executes as usual, (2) modify the text and then execute OBJ→, or (3) parse the text itself.

After the key-definition object is executed, its text form is given to βENTER as an argument. Such a program might print the text and the contents of the stack, drop the text and modify the results on the stack, or display status information or otherwise prepare for the next input from the user.

Vectored ENTER is enabled by setting both its own flag and the flag for user key assignments. The latter condition allows the user to disable vectored ENTER from the keyboard. This safety feature is important, since faulty customization routines can totally disrupt calculator operation.

Finally, the user can maintain a variety of interfaces customized for different purposes. Since the custom menu and vectored ENTER are defined by variables, switching directories can cause the interface to change accordingly. On the other hand, key assignments are independent of the directory. By assigning directory-switching programs to keys, the user can readily switch from one interface to another.

The EquationWriter

The primary design objective of the EquationWriter was to overcome several factors limiting the ease of use of existing calculators. The EquationWriter is the first application to emerge from advances in display technology, both hardware and software, compared with the HP 28S. The general result of these advances can be seen in the inclusion of the graphic object data type and the virtual screen of the HP 48SX.

The basic idea of the EquationWriter is to show mathematical expressions as they appear in textbooks or as normally written by hand—for example:

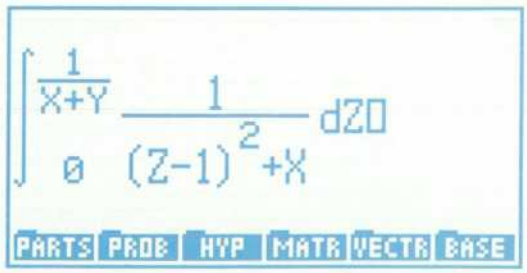
- Numerators above denominators, separated by a horizontal line
- Exponents written as superscripts, in a smaller font
- Parentheses of adjustable height
- The use of standard symbols for integral, summation, etc.

This in itself is novel only in the calculator world. However, the main challenge, which was felt not to have been met even by existing desktop systems, was to come up with a consistent and intuitive way of producing and modifying these formatted displays as the user enters the expression, symbol by symbol.

The most obvious limitation on the entry and display of mathematical expressions in the standard linear format is that when they get even moderately large, it becomes extremely difficult to survey the subexpression groupings visually and sort out all the parentheses. An expression like

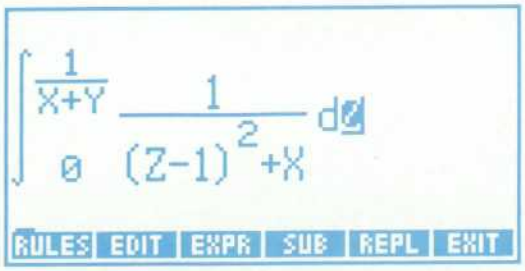
$$\int(0,1/(X+Y),1/(1+((Z-1)\hat{}2+X)),Z)$$

is terribly tedious to read and understand, compared to

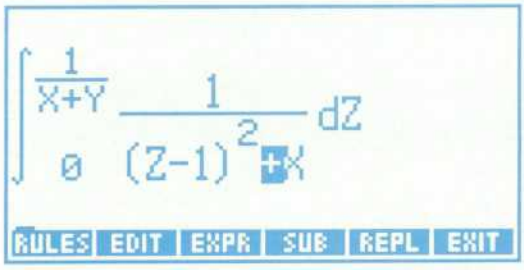


This problem was especially onerous for the HP 28S FORM interface (renamed RULES in the HP 48SX), which applies operations like commutation, association, and distribution to subexpressions of a given expression. Locating

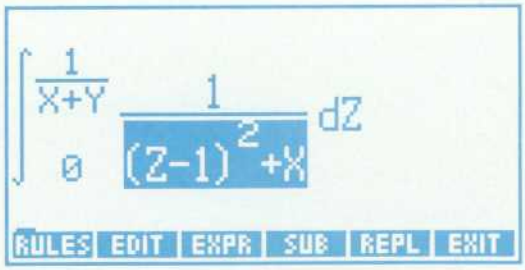
the desired subexpression (which very likely did not even fit on the screen) amid the plethora of parentheses, and recognizing its relation to the likewise messy and stretched-out result, was too much for many users to bother with. In the HP 48SX, the RULES interface is a subsystem of the EquationWriter, which can be entered any time the expression typed so far is complete (a+b is complete, a+ is not) by pressing the ◀ key. This sends the cursor back to the rightmost object (number, variable, or operator) in the expression, appearing as an inverse-video highlight of that object.



From here the highlight-cursor can be moved around with the arrow keys. Pressing ◀, ▼ results in:



The subexpression selected for an operation is that which is included in the range of a highlighted operator (if a variable or number is highlighted, the subexpression simply consists of that object alone). A menu key toggles between highlighting the individual object and the selected subexpression.



This removes any remaining uncertainty about which subexpression is selected (although it is not usually needed because the grouping is so much more apparent, and more of the expression tends to fit on the screen).

Since the subexpression selection mechanism was included for the RULES interface, it made sense to allow editing of subexpressions as well. Once a subexpression is selected for editing, a command line is brought up in which to modify the subexpression in its normal string form. This is mainly useful for changing the spelling of a name or number. Other means of modifying and combining expressions are provided by the SUB (substitute), REPL (replace) and RCL (recall) functions: the first sends a copy of the selected subexpression out to the data stack, the second replaces the selected subexpression with the algebraic object on the data stack, and the third inserts the object from the stack in the cursor position when in entry mode (rectangular cursor showing). Of course, it is possible to back up with the normal backspace key (⌫).

Another problem with the standard linear format is remembering the meaning and order of multiple parameters. A frequent complaint about the HP 28S was that no one could remember how to type in the parameters to the INTEGRAL function. Did the lower or upper bound come first? Did one have to type the d that goes with the variable of integration? In the EquationWriter there can be no such confusion. Upon pressing the ∫ key, one immediately sees an integral sign, with the cursor in the position of the lower bound.

Any expression can be entered as the lower bound; it is terminated by the ▶ key, which is the general means of terminating any syntactic piece (exponent, numerator, denominator, etc.). The cursor then moves to the upper bound position.

If one of the subexpressions grows vertically (e.g., when entering a quotient for the upper bound), the integral sign stretches to accommodate it.

After the upper bound and integrand are terminated in turn, a d appears with the cursor to its right, making it obvious that a variable name is now required.

This is a good place to mention another significant feature of the EquationWriter, which is its real-time syntax checking. With the cursor in the variable of integration position, the system will not accept any input except a legal variable name. Pressing + here will immediately result in the Invalid Syntax message, with the cursor returned to the left of the d. Similar behavior results from following a prefix function immediately with an infix function, and so forth. The EquationWriter uses the same internal parsing engine that is used to parse algebraic expressions typed into the command line. All graphical events in the EquationWriter (putting up a new symbol, inserting punctuation, altering the sizes of parts of the picture, and repositioning the cursor) are triggered by transitions across syntactic boundaries, as interpreted by the internal parser. The ▶ key always has the meaning of "go to the next syntactic position", so that it not only terminates exponents, denominators, and so on, but also results in the insertion of any required token following the current position, such as a closing parenthesis, or a comma if you are in the first argument of a function requiring two or more arguments. If the current syntactic piece is not legally completed, the cursor will not advance.

This general use of the ▶ key was actually quite controversial during the early phases of development, and this illustrates the challenge of coming up with an intuitive entry procedure, as mentioned above. One objection was that the ▶ key is an unnecessary nuisance when entering a typical polynomial: after typing the 2 in an expression like ax^2+bx+c, why can't I just type + ? There was no problem in adopting such a rule, based on operator precedence, but the result would be that the hated parentheses would start sprouting any time the exponent, numerator, denominator, or other expression was not typical (i.e., simple), and the user would have to remember to type the parenthesis before starting the subexpression (just like in the old linear format). In the end it was decided to provide both methods as modes that can be toggled by the user. A similar problem with respect to division was solved by providing, in effect, two division operators: one prefix (press ▲ to initiate a complex numerator) and one infix (press ÷ to draw a line under the preceding subexpression, going back to an operator of lower precedence than division). However, the uniformity of the ▶ key has proven to be a contribution to an intuitive interface. Not only do all built-in operators work similarly, but also all future operators, with their own distinctive graphical properties defined by users who write libraries, will also have the same feel.

The ideal of displaying expressions just as they appear in textbooks turned out to be unattainable, mainly because textbooks were found to follow different rules and ill-defined conventions. In the expression ax^2+bx+c, everyone assumes that a and b are coefficients, but in general, variable names cannot be limited to one character, nor should there be something special about the letter x. Thus we gave up the idea of incorporating implied multiplication in the display. However, it is present in the entry rules: typing 2A automatically creates the display 2*A, and similarly, any sequence of two contiguous tokens functioning as operands results in the insertion of the multiplication symbol. The display is unambiguous, but the typing is simplified.

Graphics and Plotting

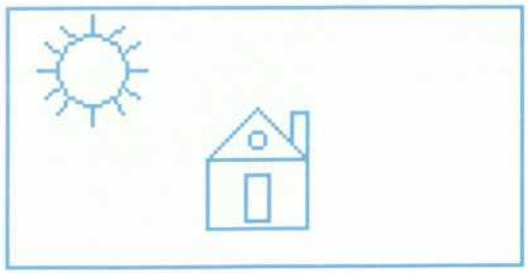
Scientists and engineers use graphics for many aspects of their work: describing problems, studying functions, working out solutions, presenting data, and so on. Our main goal for the HP 48SX graphics and plotting system was to offer plotting tools beyond those of the HP 28S, which provided function plots and statistical scatter plots. We also wanted to contribute to the overall goal of making the calculator easier to use. A third goal was better integration of graphics with other capabilities of the machine.

Our design choices were based on feedback from HP 28S users, guidelines provided by the National Council of Teachers of Mathematics, consultations with mathematics educators, and the experience of team members as college instructors, mathematicians, physicists, and engineers. The result is the HP 48SX graphics and plotting system, which has the following new elements:

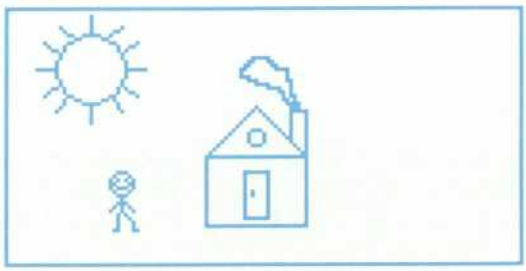
- A new RPL object type called a graphics object, or GROB, along with commands to create and modify GROBs
- An enhanced interactive environment for plotting and graphics
- Four new mathematical plot types and two new statistical plot types.

The HP 48SX uses a new object type called a graphics object, or GROB, to represent graphical images. Like all objects, GROBs can be placed on the stack, included in programs, stored in variables, and exchanged with other calculators. Most graphics commands act on the GROB stored in a special display region called PICT. The HP 28S used one area of memory for all displays. The addition of a separate graphics display area in the HP 48SX simplifies mixing graphics and stack operations. Both display areas are expandable, with scrolling available to view GROBs larger than the display.

Commands are available to draw geometric shapes, sketch freehand, or do cut-and-paste operations with smaller GROBs. Most of these commands are available in both interactive and programmable forms. For example, geometric shapes include boxes, circles, and lines:



Freehand sketches include arbitrary curves and individual pixels:



The REPL (replace) command takes a GROB from the stack and pastes it into the PICT GROB. The next example includes a label made from a string (by the →GROB command) and a picture imported from a computer.

Adding a modify step to cut-and-paste leads to a rudimentary but entertaining form of animation.

The calculator itself uses GROBs for all display-related tasks. Graphic applications such as plotting use GROBs, of course, but text must also be converted to pixels before it can be displayed. For example, the components of the normal display (status information, stack objects, command line, menu labels) are created as individual GROBs and then pasted onto the GROB in the stack display area. Applications such as EquationWriter and MatrixWriter similarly construct and combine GROBs.

Much of the benefit of GROBs, like other object types, is in having standard tools for standard objects used by both the system and the user. This uniformity leads to smaller code with fewer defects.

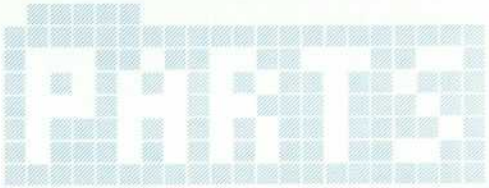
An example of the internal structure of a GROB can be seen in the PARTS menu label in the MATH menu. The calculator creates a small GROB for this label and then pastes that GROB onto the larger GROB for the whole display.

The command-line form of this GROB is:

```
GROB 21 8 E30000FFFFF11B9131555BD1119BB1D55B71D55B91FFFFF1
```

where 21 and 8 are the width and height in pixels, and the hexadecimal digits represent the graphical data, starting left to right across the top row:

```
E30000
FFFFF1
1B9131
555BD1
119BB1
D55B71
D55B91
FFFFF1
```



Each hexadecimal digit represents a horizontal sequence of four pixels, with the least-significant bit representing the leftmost pixel. For example, the hexadecimal digit E, written as 1110 in base two, represents four pixels: off, on, on, on.

Each row is represented by an even number of hexadecimal digits because the display hardware reads one byte (two hexadecimal digits) at a time. This requires up to seven bits of padding at the end of each row; in this example, three bits of padding are required.

Function Plots

Like the HP 28S, the HP 48SX uses the variables EQ (equation) and PPAR (plot parameters) to control plotting. The user can maintain multiple plotting environments by creating multiple directories, each with its own EQ and PPAR. When the user presses ↰ PLOT, the plot application first shows the current equation (EQ) and plot type (one of the plot parameters):

We first demonstrate the plot type FUNCTION, which is an enhanced form of the HP 28S plot type. Later we will show the results from other plot types.

The user can press NEW to name a new equation, EDEQ to edit the current equation, or CAT to show a catalog of equations:

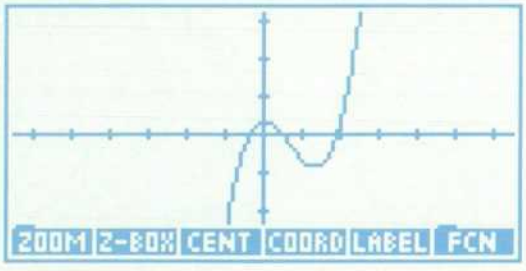
The equation specified by EQ can be an expression, an equation, a program that computes values, or the name of a variable that contains one of these. Multiple equations can be plotted simultaneously by combining them into a list and storing that list in EQ.

After selecting the equation, the user can press PTYPE to change the plot type. Other plot parameters control the plot's scale and the placement and labeling of the axes. To change the other plot parameters, the user presses PLOTR:

Like the initial plot menu, the PLOTR menu displays the current values of relevant variables. To avoid interference with normal stack activity, these displays are maintained only as long as the commands in the menu are being used interactively.

When the plot parameters are set, the user can press ERASE to clear PICT, or skip this step to superimpose the plot on the current PICT. The plotting is started by pressing DRAW or AUTO; the latter attempts to scale one or both axes automatically, according to the plot type.

When the plot is completed, a menu of interactive operations appears:

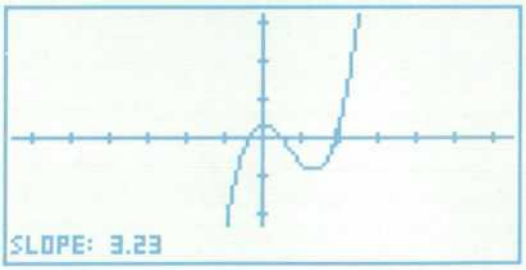


In the center of the display is a cross-shaped cursor, which the user moves by pressing the arrow keys. The cursor is used to specify locations for a variety of plotting and graphics operations. Pressing COORD causes a display of the cursor coordinates to replace the menu labels. If the PICT GROB is larger than the display, the user can force the display to scroll by moving the cursor off the edge of the display.

If the plot needs adjusting, the user can zoom in or out along either or both axes, or define a new center. If the plot is satisfactory, the user can press FCN to show a menu of mathematical tools (applicable only to the FUNCTION plot type):

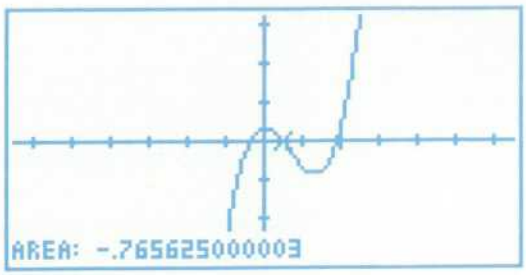
With these tools the user can analyze the function without leaving the interactive graphics environment. For example, pressing ROOT invokes the solver to find the nearest root (the cursor moves to the root):

Pressing SLOPE invokes differentiation to find the derivative at the cursor's location:



Pressing EXTR invokes differentiation and the solver to find the nearest extremum (the cursor moves to the extremum):

Pressing AREA (twice, with the cursor at each limit) invokes numerical integration:



When EQ contains of list of equations, the user can apply these tools to any individual equation, or use ISECT to find the intersection of any pair of neighbors in EQ.

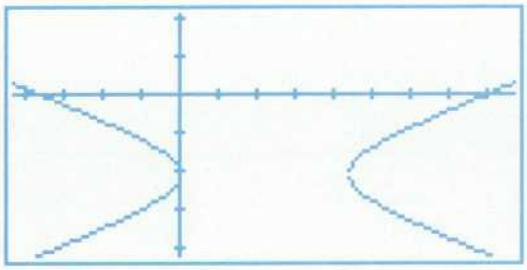
Other Plot Types

To the HP 28S plot types FUNCTION and SCATTER the HP 48SX adds mathematical plot types CONIC, POLAR, PARAMETRIC, and TRUTH, as well as the statistical plot types HISTOGRAM and BAR. The mathematical plot types share code for the basic steps: setting up the plotting environment, assigning successive values to the independent variable, evaluating EQ, plotting the corresponding points, cleaning up the environment, starting the interactive phase, and handling errors. Each mathematical plot type requires its own code to process EQ once at the start and to process each result of evaluation.

CONIC plots handle circles, ellipses, parabolas, and hyperbolas. This type turned out to be a simple combination of existing tools: the code underlying the command QUAD is used to turn EQ into two branches. Then the code in the FUNCTION plot type that plots both sides of an equation is used to plot both branches of EQ. For example, the equation

$$4*X^2-9*Y^2-24*X-90*Y-225$$

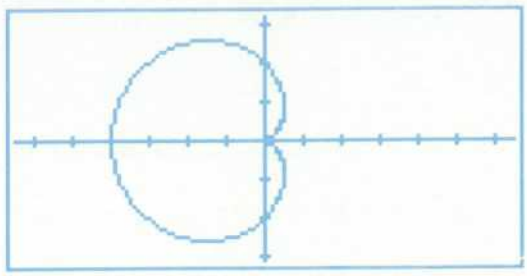
is plotted as:



POLAR plots show the independent variable as a polar angle and the dependent variable as the radius. For example, the equation

$$2*(1-\mathrm{COS}(X))$$

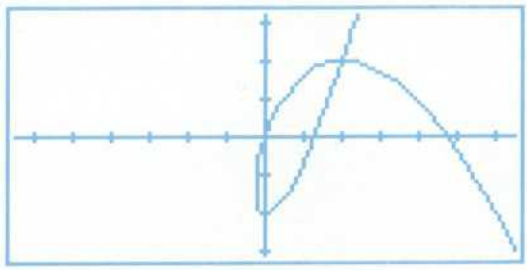
is plotted as:



PARAMETRIC plots show a complex-valued function of one real variable, where the real and imaginary parts are functions of the independent variable. For each point, the horizontal coordinate is given by the real part and the vertical coordinate by the imaginary part. For example, the equation

$$T^2-T+i*(T^3-3*T)$$

is plotted as:



TRUTH plots show truth-valued functions of two real variables. The location of each pixel represents the domain, and the value of the pixel represents the function value. Often the truth-valued function is the compostion of a function of interest, such as a real function of two variables or a complex function, and a projection function that maps function values to truth values. For example, consider a two-argument function. Plotting the expression

$$(2*X^2-3*Y^2+X*Y)\ \mathrm{MOD}\ 16>8$$

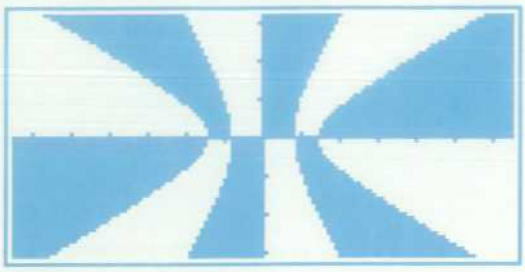
produces a contour plot of the polynomial with contour intervals of 8:

A second example is a complex function. Plotting the expression

$$\mathrm{SIGN}\ (\mathrm{RE}(Z^3-2*Z))==\mathrm{SIGN}\ (\mathrm{IM}(Z^3-2*Z))$$

where Z is defined to be $X+i*Y$ produces a quadrant plot of

the polynomial. The black regions are the points mapped to the first or third quadrants of the complex plane and the white regions are points mapped to the second or fourth quadrants.



HP Solve Equation Library Application Card

The card contains a library of 315 equations, the periodic table of the elements, a constants library, a multiple equation solver, a finance application, and engineering utilities.

by Eric L. Vogel

HISTORICALLY, EVERY HP programmable calculator has had programs available in some form. There are application books with printed programs and keystroke sequences for machines like the HP 55, 25, 33E, 11C, and 32S, application pacs with programs on magnetic cards for the HP 65 and 67, and pacs with programs in plug-in modules for the HP 41 and 71. These books and pacs focus on computation-intensive (as opposed to data-intensive) problems in specific science or engineering disciplines. Program size and capability are limited primarily by available memory and the single-line calculator display.

The HP Solve Equation Library application card provides this capability for the HP 48SX scientific expandable calculator, but without the limitations of previous pacs. The focus on computation-intensive solutions is preserved, but for a wider range of disciplines than in individual pacs in the past. An additional focus on data-intensive applications has been added in the form of on-line, electronic reference information (two thirds of the card contains data). The 128K-byte memory capacity of the card makes these two emphases possible, and the large display allows improved user interfaces for the interactive applications.

The card contains six major applications:

- Equation Library (Fig. 1). This is the primary application for which the card was named: a collection of 315 equations organized in a catalog of 15 different subjects, each containing a catalog of equation titles. For each title, the user can examine the equations and catalogs of names, descriptions, and SI or English units for its variables. A key contribution is pictures that describe the physical situations represented by the equations. Our goal was that the subject, title, and reference information would help a user select an equation to use with the HP Solve

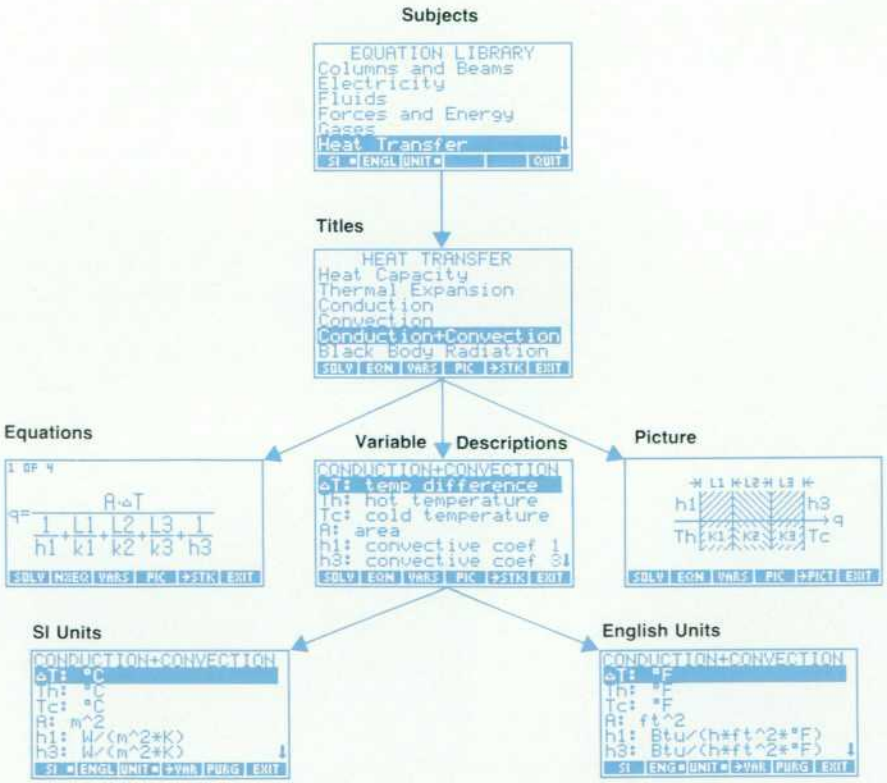


Fig. 1. *Equation Library user interface.*

application or the card's Multiple Equation Solver (discussed below).

- Periodic Table (Fig. 2). This application contains all the chemical data (such as atomic weight and density) that appears on a standard periodic table of the elements. The primary user interface is the universally recognized grid of elements. The user can move a highlight block to see any element and its most-used properties on the grid. There is also a catalog of 23 properties available for each of the 106 elements. Properties can be plotted versus atomic number to reinforce the relationship between property and atomic structure. A molecular weight calculator allows typing chemical formulas and quickly calculating their molecular weights.
- Constants Library. This is a collection of 39 commonly-used physical constants. These appear in catalogs of symbols, descriptive names, values, and SI or English units.
- Multiple Equation Solver. This is a collection of commands that make it possible to use the Multiple Equation Solver to interact with the user's own equations as a group, rather than just the groups of equations from the Equation Library.
- Finance. This application duplicates the basic calculations performed by HP financial calculators: time value of money (the relationship between the number of payments, interest rate, present value, payment, and future value) and amortization.
- Engineering Utilities. These are engineering functions that support the computational needs of some of the equations in the Equation Library.

These applications come in two forms: interactive for working with the application and its data, and noninteractive for programmatic calculations and access to the on-line data.

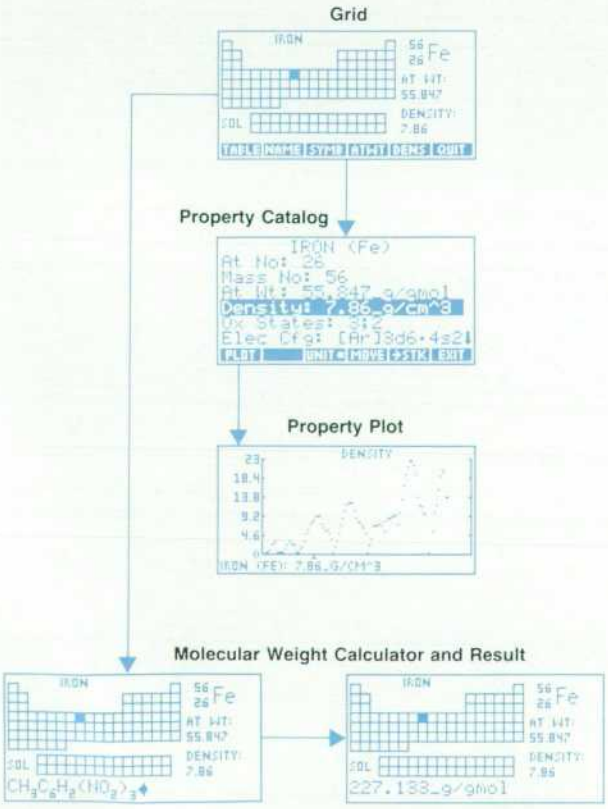


Fig. 2. *Periodic Table user interface.*

Equation Library Evolution

The Equation Library concept stemmed from three observations. First, students need a wide variety of solutions because of the number of classes they take. Because application pacs are limited to specific areas, students often need several pacs to cover the different disciplines they study simultaneously. Second, most of the engineering applications for the HP 41 and its predecessors are programs that simply solve an equation for a specific variable. More sophisticated programs of this type allow interchangeable solutions in which most or all of the unknown variables can be calculated as long as they can be isolated algebraically in the equation. Some programs use iterative techniques to find a solution when an algebraic isolation is not possible. Later application pacs attempt to allow the user to select different units for the different variables.

Third, the HP Solve application in the HP 48SX takes an equation and makes it into a small, self-contained application. It solves for any variable given the others, allows units to be specified for each variable, handles unit conversions automatically, and provides a consistent, straightforward user interface for interacting with all the variables.

From these three observations, we realized that we could create a collection of small applications in most of the science and engineering disciplines of previous application pacs by combining a collection of equations with HP Solve. The HP Solve user interface allows each application to work the same way, and the ability to solve for any variable and automate unit conversions makes these applications more versatile than in previous application pacs.

Interacting with Groups of Equations

As the equation selection proceeded, we found that related equations were usually needed as a group, rather than independently (Fig. 3). While there are certainly instances where only one of the equations is needed, more often the entire set is used to find the value for a particular variable. To provide simplified access to related equations, we group them together under a single title, such as Linear Motion or Ohm's Law and Power, rather than forcing the user to return to the Equation Library to select each equation individually.

When we examined how a user typically interacts with a group of equations, we realized that the solution proce-

$$x = x_0 + v_0 \cdot t + \frac{1}{2} \cdot a \cdot t^2 \qquad V = I \cdot R$$

$$x = x_0 + v \cdot t - \frac{1}{2} \cdot a \cdot t^2 \qquad P = V \cdot I$$

$$x = x_0 + \frac{1}{2} \cdot (v_0 + v) \cdot t \qquad P = I^2 \cdot R$$

$$v = v_0 + a \cdot t \qquad P = \frac{V^2}{R}$$

(a) Linear Motion (b) Ohm's Law and Power

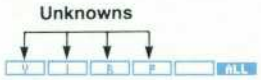
Fig. 3. *Examples of common equation sets.*

dure is straightforward, but gets tedious as the number of equations and variables gets large. Here is the manual process for finding all the variables for a set of equations given that some of them are known:

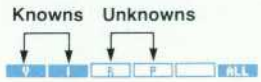
- Use the known variables to select an equation containing only one unknown.
- Solve for the unknown.
- Add the variable just calculated to the set of known variables.
- Use the combined set of known and calculated variables to select another equation containing only one unknown.
- Solve for the unknown.
- Repeat this process until either all the unknowns have been found or as many unknowns as possible have been found from the given set of knowns.

To make using the equations more straightforward, we have automated this manual select-and-solve process by developing an extension to HP Solve called the Multiple Equation Solver (MES). The MES selects the appropriate equation to solve based on whether it has one remaining unknown, and then solves for that unknown using the same numerical root finder used by the HP Solve application. It tracks the variables that have been solved for, and uses different equations to calculate other unknowns as soon as there is enough information available.

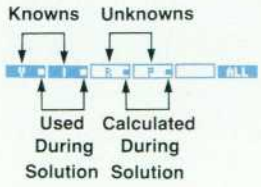
The barrier to proper functioning of the MES was identifying whether a variable is known or unknown. The existence of the variable alone is not sufficient—after a solution has been determined, all the variables exist. The key underlying principle is that the state of a variable (known or unknown) is independent of the value of the variable. The MES uses this state information to select the equations to be solved and the order in which to solve them.



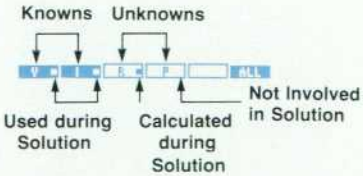
(a) Initial menu for Ohm's Law and Power



(b) After entering values for V and I



(c) After solving for all unknowns



(d) After solving for R only

Fig. 4. *Multiple Equation Solver menu key appearance.*

Displaying Variable States

The MES user interface is similar to that of HP Solve. A menu of variable names is displayed in the menu key area at the bottom of the display. The appearance of the menu keys is used to distinguish the MES state information. An extra key, ALL, appears at the end of the menu.

Initially all the menu keys are white with black letters (like HP Solve), indicating that all of the variables are unknown (Fig. 4a). Typing a value and pressing a menu key stores the value in that variable and changes the key to black with white letters, indicating that the variable is known (Fig. 4b).

Pressing the ALL key solves for all remaining variables, or as many as can be found from the given set of knowns.

Messages appear during the solution identifying which variable is being solved for and its resulting value. After the solution has completed, each variable retains its initial state. Correspondingly, each menu key retains its initial appearance. Black keys (knowns) remain black, and white keys (unknowns) remain white. This simplifies solving a problem using the same knowns and unknowns but with different values.

Indicating Variable Relationships

After a solution, some of the menu keys will have a small block in them to indicate the roles their variables played in the solution (Fig. 4c). A block in a black key (known) indicates that the variable was used to find an unknown in a particular equation. A block in a white key (unknown) indicates a value was calculated for that variable during the solution. This represents a unique state for a variable—it is an unknown, yet it has a calculated value. The next time this variable is solved for, this calculated value will be used as its initial guess.

Pressing the shift key followed by a menu key solves for that specific variable, regardless of whether it is black or white (known or unknown). After a variable is solved for, its menu key is shown in white with a block to indicate an unknown that was solved for with a calculated value (Fig. 4d). Other menu keys may have blocks in them based on the roles their variables played in the solution.

Solution Summary

A summary of the solution procedure is available by pressing the shift key followed by the ALL key (Fig. 5). This summary shows a catalog of each unknown that was found,

(a) Values Calculated during Solution

(b) Equations Used during Solution

Fig. 5. *Solution summary.*

its calculated value, and the equation that was solved to find it, in the order that the unknowns were determined by the solution procedure.

Acknowledgments

I would like to thank the rest of the development team and the people who made exceptional contributions to the product: Jim Donnelly and Megha Shyam for making it all happen, Dennis York for the initial vision, Lynn Winter for a catalog engine that was an enduring foundation, Chris Bunsen for critical design advice and testing for the Multiple Equation Solver, Diana Byrne for spearheading the software testing effort, Hank Schroeder for an owner's manual in a remarkably short time, and Ray Depew, whose thoroughness testing the Periodic Table was critical to both the card and the HP 48SX.

Hardware Design of the HP 48SX Scientific Expandable Calculator

Leveraging an earlier design resulted in prototypes with 90% production tooled parts only nine months after the start of the project. The HP 48SX includes an 8-line-by-22-character super-twisted nematic liquid crystal display, two expansion ports for ROM or battery-backed RAM cards, and two I/O ports: RS-232 and infrared.

by Mark A. Smith, Lester S. Moore, Preston D. Brown, James P. Dickie, David L. Smith, Thomas B. Lindberg, and M. Jack Muranami

THE NEEDS OF HP HANDHELD CALCULATOR customers—engineering, math, and science students and business and scientific professionals—have radically changed over the years, as the personal computer has become the standard tool of technical students and professionals. However, the need for convenient, handheld computation has not disappeared, but simply evolved.

Surveys of our high-end technical calculator customers reveal that nearly all own or have access to a personal computer, that they use their calculator daily, that professionals make up about half of the customers for high-end technical calculators, and that they need a powerful handheld calculator with graphics and I/O capabilities. The HP 48SX scientific expandable calculator (see Fig. 1 on page 6) is designed to meet their needs for more advanced computation, graphics, customizability, expandability, and the ability to link to their personal computers.

The 64-row-by-131-column STN LCD (super-twisted nematic liquid crystal display) provides the means for presenting the power of HP 48SX graphics, matrix manipulations, and equation entry. Two plug-in card slots provide RAM expansion (in 32K-byte or 128K-byte increments) and customization (plug-in application ROMs such as HP's Equation Library ROM card described in the article on page 22). A serial I/O port provides a means to link to an IBM PC-compatible or Apple Macintosh computer, and an IR (infrared) port allows sharing of solutions between HP 48SX calculators and provides a link to the HP 82440B infrared printer.

Previous Series Leveraged

One of the primary project objectives for the HP 48SX was extensive leverage from the design and manufacturing

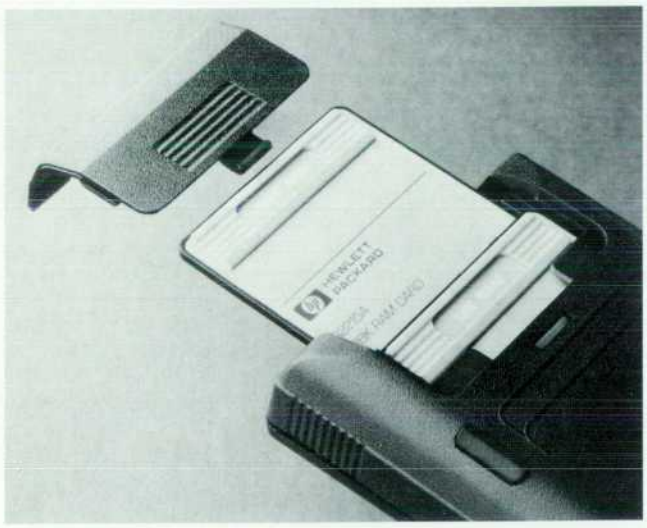


Fig. 1. *Two expansion ports accept ROM or battery-backed RAM cards.*

development of our current calculator family to minimize development time and cost. Design leverage was a primary consideration in every HP 48SX part design. This emphasis on design leverage resulted in laboratory prototype units built with 90% production tooled parts only nine months after the start of the project.

The HP 48SX is a direct descendant of the series of low-cost, high-volume, vertical-format calculators introduced in January 1988. This series, consisting of the HP 10, 14, 17, 20, 21, 22, 27, 32, and 42 calculators, achieved low cost and minimal part count through the use of a few highly integrated designs. These calculators were designed from the outset to be built on a highly automated assembly line at a rate of several calculators per minute.

The designers of the HP 48SX leveraged many of the processes, materials, and even design details from the previous series. As a result, the HP 48SX is able to enjoy the benefits of an automated assembly line that could not have been justified by HP 48SX volume alone (see article, page 40). Sixteen out of seventeen major assembly processes are common to both product lines. Only four parts are shared, but all raw materials and many suppliers are common to both families. A subtle benefit resulted from leveraging design details, since many design decisions were preordained, saving time during the investigation phase. By sharing details, the HP 48SX designers were able to proceed more confidently, capitalizing on the knowledge gained and improvements made as the earlier series of products entered production.

It took some time for the designers to accept leveraging completely, because they were forced to compromise what they envisioned as optimum. Once the commitment was made, however, many unknowns were eliminated and it became clear that, in this case, leveraging was the right way to meet the product and division objectives.

Design Features

The topcase of the HP 48SX is a four-color, injection molded part. Its 49 keys are an integral part of the topcase; each key is attached and guided by two small cantilevers. This design yields a low-profile key with an excellent controlled feel. The keyboard is made of Mylar with tuned-resistance carbon graphite traces. This allows a low profile, reduces cost, and improves reliability because of the inertness of graphite. The battery contacts are engineered to eliminate fretting corrosion and the loss of memory that would inevitably result. The liquid crystal display is a super-twisted nematic (STN) design for improved contrast and viewing angle. Two expansion ports accept ROM or battery-backed RAM cards (Fig. 1). These cards are conveniently sized to be handled easily and have a thin outline.

Throughout the machine, from the generous radius of the bottom case that allows it to fit comfortably in the hand, to the seven small openings in the battery compartment that pick off electrostatic discharges, details are incorporated into the design of the HP 48SX to achieve small size, reliability, and overall customer satisfaction.

Initial Design

Leveraging the design from the previous calculator design resulted in less than optimal tooling and design options.

For the simpler products, the layout of the single display driver IC on the opposite side of the printed circuit board from the display made sense. When applied to the HP 48SX, this constraint forced the 202 outputs from three display drivers to feed through to the display side of the printed circuit board. This wasted a large amount of board area and caused the printed circuit board to be the most expensive non-IC component in the product.

Initially, the mechanical layout was done around the same small button cell batteries that were used in the previous series of calculators. However, electrical system simulations done early in the design cycle showed battery life to be unacceptable and the HP 48SX was redesigned around higher-capacity AAA batteries.

Several other iterations occurred. A printed display window was thrown out in favor of a special hard polarizer optically bonded to the top surface of the LCD. This reduced cost, eliminated the window, which is easily scratched, and resolved a long-standing problem of particles trapped between the window and display. The biggest benefit of the windowless design is the 67% reduction in glare.

For a time we considered incorporating only one plug-in port. The two-port design was eventually selected to allow both ROM and RAM cards to be plugged in at the same time.

Final Design

The HP 48SX is manufactured from three subassemblies: the topcase, the printed circuit assembly, and the bottom case. These assemblies are built from 24 mechanical parts, four surface mount ICs, a 170-pin TAB (tape automated bonding) IC, five discrete leaded components, and 31 surface mount devices (see Fig. 2).

Topcase Assembly

The topcase assembly is designed to be as free as possible of product-specific detail. This was done to allow the basic topcase assembly to be incorporated into future products.

The top surface of the product consists primarily of a 0.012-inch-thick, seven-color, printed aluminum overlay. The purpose of the overlay is threefold: (1) to provide a pleasing surface finish and shape, (2) to carry the nomenclature for three shifted functions as well as the product name and number, and (3) to make the overlay the first line of defense in a carefully designed ESD (electrostatic discharge) protection system. Adding to the cosmetic appearance is an electroformed copper logo which, along with the overlay, is applied to the topcase using pressure-sensitive adhesive.

The four-color molded topcase design represents a major engineering and process achievement. The HP 48SX key nomenclature is actually a separate molded part for each key around which the topcase is shot (see Fig. 3). To create the cavities that form the nomenclature, the legend artwork was digitized on a CAD/CAM system. Cutter paths are generated from this data. The cutter paths are used to cut a positive image of the nomenclature in a carbon electrode. The cutters used are special ground cutters as small as 0.001 inch in diameter at the tip. The electrodes are used for electrodischarge machining of the extremely fine detail into the first-shot cavities. The tolerances must be carefully

Industrial Design of the HP 48SX Calculator

The expressed needs and wants of high-end technical calculator users contributed to the design objectives for the HP 48SX scientific expandable calculator. Needs analysis and concept testing were conducted at the onset of the HP 48SX program, and stated requirements included:

- Handheld product shape (traditional vertical format)
- Large display (8-line desirable)
- Customizable and expandable through plug-in media
- Data I/O for mass storage, printing, programming, etc.
- High-quality tactile keyboard.

In addition to these stated requirements, the industrial design objectives for the HP 48SX included:

- Easily learned accessibility to functions and features through an organized keyboard and display interface
- Direct and easy access to expansion and customization media
- A visual and tactile experience consistent with the product's "next-generation" technological leadership.

Package Design

A softened, vertical-format package allows comfortable handheld use of the HP 48SX. Even the molded rubber feet are sculpted to match the case contour for minimal tactile intrusion. Overall size was determined by the handheld requirement for width, keyboard and display size for length, and component sizes for thickness. Components are located to allow a low leading edge for the keyboard, resulting in a three-degree wedge profile. The keys and keyshapes are leveraged from the previous series of low-cost, high-volume calculators. They are designed to provide comfortable yet positive tactile response, minimal entry errors, and high reliability. User access to batteries and plug-in media is provided through covers at the back of the calculator. Both the infrared and RS-232 I/O ports are at the back, directed away from the user. A molded arrow-shape indicator on the topcase, above the logo, communicates the position of the direction-sensitive infrared I/O feature, and the lens area of the cover over the infrared components is polished to communicate its light-transmissive function. Next to it is the RS-232 port, which accepts a small, custom 4-pin plug designed to match the visual theme and scale of the HP 48SX.

The goal of customization and expansion is accomplished through two plug-in slots for credit-card-size memory modules, available as RAM or ROM applications. The slots are located under a removable cover and infrared lens at the rear of the bottom case. Because of cost and size constraints, an eject mechanism was not feasible for user removal of the cards. Instead, user access is provided by offset-stacking the cards to expose custom, molded plastic grips, which are attached to the cards by the vendor (see Fig. 1 on page 25). These contoured, textured grips are molded with a hole in the center to provide visual access to a title graphic on the printed card surface while the card is inserted in the calculator. The constructed grip texture is consistent in scale and appearance with other grip textures used on the HP 48SX, and enables the user to remove the cards from their friction-fit edge connectors with a single finger or thumb.

A custom soft case was designed to be included with the HP 48SX. Its objectives are to protect the product in normal transport, to provide storage for a quick-reference guide and additional memory cards, and to reinforce the message of product superiority and quality. The final design is a fine-weave pouch. The case is lined and padded, and has a zipper closure. An internal pocket is provided for the quick reference guide and memory cards.

(continued on page 28)

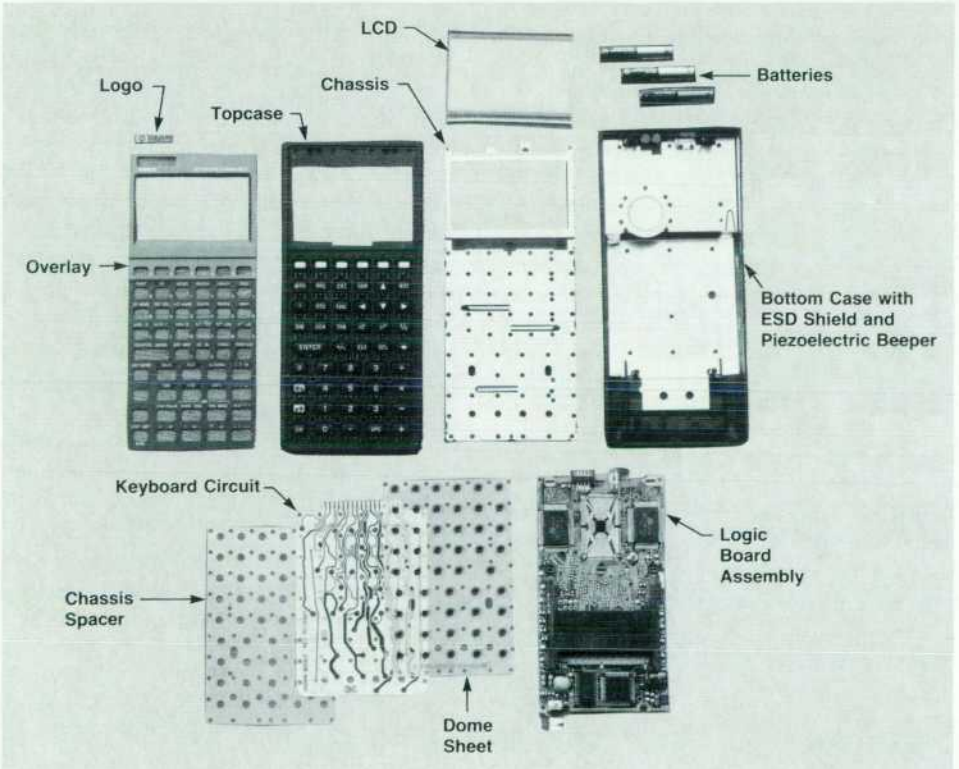


Fig. 2. *The three HP 48SX subassemblies are the topcase, the printed circuit assembly, and the bottom case. They are built from 24 mechanical parts and various ICs, discrete components, and surface mount devices.*

User Interface Hardware Design

Although the HP 48SX approaches computer power and sophistication, it operates primarily as an application with a dedicated keyboard and display interface. The most significant industrial design challenge was to provide visually ordered access to the 2100 functions using only forty-nine keys plus display menus.

The user interface surface of the HP 48SX is divided into two primary zones: the display and display bezel and the keyboard and keyboard overlay (see Fig. 1 on page 6). The windowless 8-line liquid crystal display is framed by a formed aluminum overlay, which "cascades" down to the keyboard in two steps. Located in the middle step is a row of six keys which operate in conjunction with menus presented on the bottom row of the display. The overlay color surrounding these keys is the same as the display bezel to reinforce the association of keys with display menus. The six menu keys are differentiated in size and color from all other keys to distinguish them as special. At the base of the second overlay cascade is the remainder of the keyboard, which provides quick, direct access to alphanumeric entry, math operations, cursor control, and function sets presented as display menus. The total of 49 keys was chosen based on the minimum number required to provide discrete alphanumeric entry. Keys are grouped and sized according to function and relative frequency of use to order the keyboard visually.

The majority of the function markings are printed on the aluminum overlay. Unlike preceding models, including the HP 41C, there was no opportunity to place a second set of function markings on the keys. This is because of the leveraged keyboard design, which is limited to a single, integrally molded function marking. This presented a graphic challenge because the keys access up to four major functions each. As a result, the keyboard overlay is designed with up to two shifted functions above each key, positioned side by side and accessed by color-coded shift keys. The shift keys are also coded with arrows indicating left or right for the relative position of the shifted nomenclature. These symbols are used in the documentation in place of color to reduce printing costs. Twenty-six keys also have an alpha character at the lower right corner, accessed by an alpha shift key. Shifted functions that call up screen menus are distinguished by a black field behind the text and are grouped in two areas of the keyboard. The overlay has a total of six silk-screened colors and one tint. The keyboard is designed to accept snap-in custom overlays for user-programmed keys, custom application requirements, and so on.

Colors

A very dark, high-chroma brown called mercedes black was chosen for the two major case parts for its consistency with the new calculator line and strong customer preference in earlier appearance tests. Because of the integral design of the hinged keys, they share the case color. A warm gray metallic color is used on the display bezel to feature the display area visually, soften the transition to the liquid crystal display, and add richness to the overall product appearance. The background color of the keyboard overlay is a lighter version of mercedes black called mercedes medium, which provides a subtle contrast to give the keys visual prominence.

The most challenging colors to determine were the light blue and coral shift colors on the keyboard overlay. The shift colors on all HP calculators are intended to contribute an accent color on otherwise neutral platforms, and to code the product visually into a product category (business, scientific, or RPN scientific). The HP 48SX is in the scientific category, but functions in both algebraic and RPN modes and is really in a category of its own. The two shift colors selected were originally used for the two scientific categories, but are lightened for readability. In addition to their both being "scientific" colors, they were selected because of their easily distinguishable hues and their balanced values and chroma, which help alleviate a spotty appearance. All other function markings are in legend light gray, selected for its neutrality and readability on the background colors.

Corporate identity is provided by a nickel-plated electroformed logo, selected for its high-quality, three-dimensional appearance. The product name is designed for consistency in location and fonts with other current HP calculators, and is printed in a nickel metal tint to match the logo.

Michael Derocher
Senior Industrial Designer
Corvallis Division

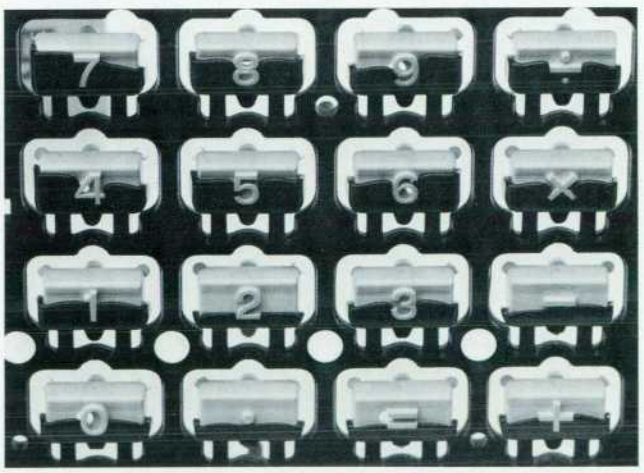


Fig. 3. *A "short-shot" photo of the HP 48SX keys. A separate molded part for each key produces the key nomenclature. The topcase is molded around the key nomenclature. Notice the hole in the* **9** *key, which is formed by a moving core pin. This hole allows the second-shot plastic (dark) to flow into the center of the 9.*

controlled because the curved steel surface of the second-shot cavity must "shut off" against the corresponding curved plastic nomenclature that was formed in the first-shot cavity. The raised-plastic nomenclature ribs must be just touched by the second-shot cavity. The interference between the two must be large enough to prevent the nomenclature from being pushed around by the injection pressure of the second-shot plastic but not so large that the nomenclature plastic is crushed and distorted by the clamping force, which can be as high as 400 tons.

The 49 keys are integrated into the topcase design to simplify assembly and eliminate the possibility of misloading a key. To achieve good tactile feel, each key is suspended on two cantilevers, each 0.016 inch thick by 0.116 inch long. These small beams serve three purposes. They guide the key through a well-controlled arc, they permanently attach the keytops to the topcase, and they serve as the paths through which the second-shot plastic flows. To fill the keytops through these small beams, each key has its own gate, for a total of 60 gates. An ordinary part of this size would have only one gate.

The Triax material used for the topcase is a Nylon-ABS

alloy and was specifically selected for its toughness and processing characteristics. The most critical aspect of the topcase design is the small cantilever beams. The beams had to be tough enough to maintain their mechanical integrity through environmental and life testing, yet supple enough not to degrade the force-deflection characteristics engineered into the snap domes. The Triax plastic was found to flow well, which allows a thin beam, and its toughness was verified in testing over 2.5 million key cycles without failure.

The tactile feel of the keys is a result of painstaking design, testing, and quality assurance efforts. The 0.004-inch-thick Mylar dome sheet contains 49 details, which provide the snap feel as each key is depressed. Each key has an actuator which presses against the top center of the dome. As the dome is pressed it deflects at its base into a dome spacer recess. At this stage the force is building linearly with deflection. As the force builds to the trip point the dome buckles, causing a sudden drop in force. Momentum and the resulting imbalance of force between the finger pushing on the key and the key no longer pushing back causes the inside of the dome to impact firmly against the keyboard contacts. The inside of the dome is printed with a pad of conductive carbon graphite. Upon switch closure the dome pad shorts the interdigitated carbon graphite contact fingers of the keyboard. Below each key switch is another recess created by a 0.003-inch-thick chassis spacer. This recess allows the keyboard to wrap around the depressed dome, resulting in a much more reliable area contact instead of the point contact of a dome against a flat plane.

The keyboard is a 0.003-inch-thick Mylar sheet with screen-printed carbon graphite traces and contacts. The feedthroughs of the two-layer keyboard employ triple redundancy to improve yield and reliability. The keyboard lines are multiplexed with the address lines. This scheme reduces the pin count on the IC and saves printed circuit board area, both of which lower costs. However, it requires that key line resistance be carefully controlled. The exact sheet resistance of the carbon ink was measured. This information was used to hand-tune the area of each of the 98 key line traces.

The backbone of the HP 48SX is the nickel-plated 0.018-inch-thick stainless steel chassis. The chassis serves several functions, the most obvious of which is to support the keyboard and add rigidity to the topcase. Sixty-two heatstakes clamp the keyboard between the topcase and the chassis. The large number of heatstakes provides an even preload, which guarantees a consistent key feel.

The chassis also furnishes the negative battery contact. Incorporating the negative battery contact as an integral part of the chassis eliminated the need for a separate contact and the operations to attach it and its wire. The chassis serves as the connection between the overlay and the ESD shield located in the bottom case. Redundant cantilever contacts are employed in both cases to provide the lowest-impedance contact under all conditions. Seven snaps around the perimeter of the chassis maintain a constant preload between the two case halves. This ensures a good cosmetic fit along the 0.040-inch-wide seam. It also resists shear, greatly enhancing the torsional rigidity of the product. A closed-cell urethane foam pressure pad provides a constant force to maintain the connection between the key lines and the printed circuit board. The chassis provides the topcase with two snap details centered on the key line connections. These prevent the topcase from creeping over time under the constant force of the pressure pad in this area. A special detail is designed to stiffen the chassis around the liquid crystal display. Basically, the chassis forms a frame around the glass display to protect it from impact and bending that occur as a result of the product's being dropped.

Two narrow strips of double-sided adhesive tape are used to secure the display in the chassis. The 202 0.013-inch-wide LCD contact pads must be positioned within ±0.003 inch to make proper connection to the printed circuit board. Two zebra connectors are used to make connection between the LCD and the printed circuit board. The zebra connectors consist of two strips of silicon rubber supporting a thin row of alternating conductive (carbon loaded silicon) and insulating materials. The conductors are on a 0.004-inch pitch so that each display line connection is made with a minimum of two conductors. Two precision punched holes, which are optically registered to the printed circuit board traces, align the chassis/LCD assembly. Tabs on the chassis are inserted into the zero-clearance printed circuit board holes, permanently alligning the printed circuit board and LCD. Six twist tabs secure the assembly, providing a constant preload for the zebra connectors.

The eight-line, twenty-two character graphics display is one of the HP 48SX's most valued features. Attempting to prevent display breakage when the product was drop tested from up to two meters was one of the more interesting challenges for the designers.

The glass from which liquid crystal displays are manufactured is fragile. Glass failures are always in tension. The glass has a theoretical tensile strength of 100,000 psi. However, flaws within the glass lower its design limit to 10,000 psi. Edge defects, a result of scribing and breaking the glass to size, cause stress risers which further reduce its usable strength to only 1,000 psi. Once the design of the display mounting had been optimized, designers turned to reducing the variability in the strength of the display itself. Improving the surface finish at the scribed edge proved to be the most attractive solution. The display in the HP 48SX has a special polished finish on the glass edge where the tensile stress is highest during a front drop. This results in measurably better and more consistent drop test performance.

Printed Circuit Assembly

The HP 48SX printed circuit assembly is a collection of proven technology and innovation. To be built on the existing surface mount printed circuit assembly line, both of the new HP 48SX connectors had to be designed as robot-loadable surface mount devices.

Dominating the printed circuit assembly is the large 80-pin plug-in card connector. This custom connector accepts two cards in a staggered formation so that the label on each card can be seen. The card connector is molded with 40% glass filled PPS to survive the 225°C (437°F) infrared soldering process. Heatstakes are used to secure the card connec-

HP 48SX Custom Integrated Circuit

The HP 1LT8 IC is the single custom chip in the HP 48SX calculator. The 1LT8 IC is divided into the following functional areas:

- A 4-bit CPU with an 8-MHz clock. The 1LT8 CPU is identical to the CPU used in the HP 28S, which is a leveraged redesign of the 1LK7 CPU. The 1LT8 CPU provides faster instruction execution times but still maintains full compatibility with the 1LF2/1LK7 and the HP 71B bus architecture. The CPU internal and external data paths are 4 bits wide. Memory is accessed in 4-bit quantities (referred to as nibbles) using 20-bit addresses, yielding a physical address space of 512K bytes. The CPU internal word size is 64 bits. Operations are performed on data strings up to 16 nibbles in length.
- A memory controller capable of interfacing to five commercial byte-wide RAMs, ROMs, or plug-in ports. The purpose of the memory interface is to allow the 1LT8 chip to drive commercial RAM and ROM ICs. Since commercial memory parts are byte-wide, this requires some careful interfacing to the 4-bit world of the CPU. The same lines that go to the memory address are also used to scan the keyboard. The keyboard is connected through series resistors to the I/O lines while the memory is connected directly. This allows commercial memory ICs to be connected to the 1LT8 chip with a minimum of added pads. Each memory device can be configured by software to the required size and placement in the address space.
- A liquid crystal display (LCD) controller capable of driving a 64-way multiplexed display. This controller halts the CPU to access data in main RAM and then formats it for the LCD. It is capable of handling the softkeys in a separate memory area and scrolling the main display.
- A collection of memory mapped control registers including a 32-bit quartz-crystal-controlled timer, a 1200-to-9600-baud full-duplex UART capable of driving RS-232 or infrared-level signals, and a cyclic redundancy check (CRC) generator.
- A collection of analog circuitry including a dual power supply (4.3 volts for the system and 8.5 volts for the display), a low-battery indicator, a power-on/reset circuit, a crystal oscillator, a frequency multiplier (which generates the 8-MHz CPU clock from the 32-kHz crystal), and a display voltage generator.

The 1LT8 IC is manufactured at the Northwest IC Division of Hewlett-Packard.

Preston D. Brown
Development Engineer
Corvallis Division

tor to the printed circuit board and to take the preload exerted by the 80 cantilevered contacts. The contacts consist of two rows of cantilevered phosphor bronze beams which are angled into the solder paste to form a butt joint. A butt joint is used because it wicks the solder up high into a fillet, preventing solder bridging between adjacent leads and allowing easier inspection. The nominal preload between the leads and the plane of the printed circuit board allows for some nonplanarity in the leads, ensuring that each lead penetrates the 0.004-inch-thick solder paste and yields a well-wetted solder joint.

The four-pin serial connector uses a similar butt type solder joint design. Like the card connector, it is a custom design using heatstakes to fixture the part during infrared soldering and to prevent the solder joints from being overstressed during use. Cantilevered arms are formed with the body of the connector to provide a slight snap upon insertion of the plug and a retention force to resist removal.

The most intensively engineered component on the printed circuit board is the 170-pin TAB-package IC. The capabilities of the IC itself are very impressive, but equally impressive is the way in which the IC is connected. Gold bumps are deposited on the IC at each contact pad. A circuit consisting of 0.003-inch-thick polyimide film with 0.003-inch-wide traces is positioned on top of the IC. The layout of the circuit is such that the traces at the inner portion of the tape actually cantilever off the polyimide substrate. These tin-plated copper beams are then aligned with the IC and simultaneously bonded. 170 inner lead bonds with a 0.005-inch pitch are made in one operation. The advantages of TAB are that its pin count is high, that it can be gang bonded, and that the polyimide substrate is punched with holes for 35-mm sprockets just like 35-mm move film (see Fig. 3 on page 42). The last feature is used to place hundreds of parts with no human intervention. The reels of burned-in and tested TAB packages are loaded onto an HP-developed TAB dispensing tool. This tool trims and forms the outer leads and presents the excised parts to a robot for placement on the printed circuit board.

The pitch of the outer leads is 0.020 inch, so accurate positioning of the TAB package on the printed circuit board is important. The position of the printed circuit board traces is determined by reading a target on the board with an optical sensor mounted on the robot. The TAB package is positioned by an annular ring of copper trace which overhangs a hole in the polyimide. The hole is exposed and etched at the same time as the 170 leads that are being positioned. This allows very accurate positioning of the leads using a simple mechanical detail. The leads are formed so they are preloaded into the solder paste. They have some compliance but the force they exert requires a TAB clamp to maintain the leads in intimate contact with the paste during the reflow operation. The steep angle at which the leads enter the paste is designed to hold the solder up high into a fillet, easing inspection and preventing solder bridging between pads.

Much design and testing went into the development of the TAB process. The results from a technical point of view far exceed all expectations.

Bottom Case Assembly

Most of the details unique to the HP 48SX are contained within the bottom case, making it possible for future products to use the HP 48SX topcase assembly.

The HP 48SX's plug-in cards are guided into the card connector through the rear of the bottom case. A box is created within the bottom case to guide the cards and prevent damage to internal components. An infrared-transmissive polycarbonate card door covers this box, keeping the cards from dislodging during a drop and forming a window

over the LED and phototransistor used for infrared I/O.

Critical to a CMOS product is maintaining power to avoid losing memory. Particular attention was paid to designing the battery compartment so power could not be interrupted. The battery compartment holds three AAA batteries. The compartment is designed so that if battery leakage occurs from the vents at the negative terminals, no openings allow it into the circuit board. The battery contacts are designed to prevent fretting corrosion, a type of corrosion caused by micromotions at the contact point, which results in oxide buildup and eventual loss of contact. Extensive testing was undertaken at the very start of the project to characterize this phenomenon. Custom electrical measurement tools were employed that could detect the very onset of fretting corrosion. Batteries from U.S., European, and Japanese suppliers were exhaustively tested for compatibility with the contacts. Even the battery door was designed to prevent loss of power. The door employs a multistage snap, which prevents the batteries from becoming dislodged in drops up to two meters. The result is a design that ensures extremely reliable power even under the most extreme conditions.

A 0.006-inch-thick 3003 H18 aluminum ESD shield lines the inside of the bottom case. The shield contacts the chassis with a large-area contact. This forms a two-dimensional Faraday cage, protecting the circuitry from exposure to electric fields. The HP 48SX is designed to withstand repeated discharges with potentials up to 25 thousand volts while running without hard failures, and 15 thousand volts without any disruption of its operation. The ESD shield also provides the ground connection for the piezoelectric beeper. A single spring makes the positive connection and provides the force for the ground connection to the ESD shield. Heatstakes keep the beeper and ESD shield in place. The conical beeper spring is designed with three closed coils at the top and bottom. These coils are wider than the pitch between the open coils, preventing the springs from tangling so they can be barrel plated and vibratory-bowl-fed to a robot for totally automated placement directly on the printed circuit board.

Completing the HP 48SX package are four molded feet, which are sculptured to conform to the bottom case radius. The feet are press fit into the bottom case while it is still warm at the molding machine. A small bump on each foot provides compliance so that the HP 48SX will resist rocking even if the surface is slightly uneven.

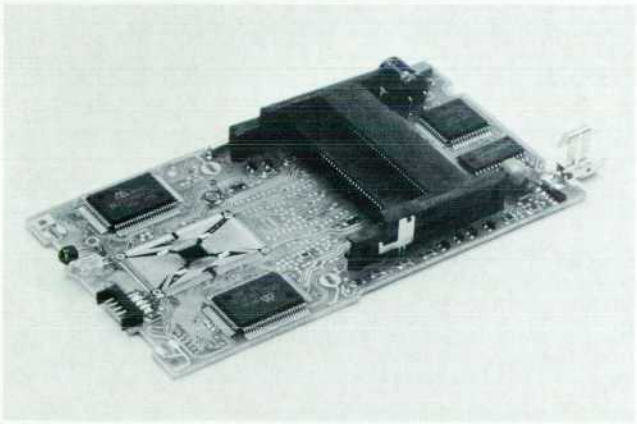


Fig. 4. *HP 48SX logic printed circuit assembly.*

Electrical System

The HP 48SX electrical system is considerably more powerful and more complex than those of previous HP handheld calculators. With its infrared and RS-232 I/O and plug-in ports, it is also more versatile. The heart of the electrical system is the 1LT8 chip, a 170-pin HP-developed IC (see box, page 30).

The HP 48SX contains two printed circuit boards. The main logic board is sandwiched between the topcase assembly and the bottom case. The Mylar domed keyboard with carbon graphite traces is housed in the topcase assembly.

The 49-key keyboard is scanned by the 1LT8 chip via multiplexed RAM and ROM address lines. Address lines A9 to A17 scan the keyboard while A0 to A5 are inputs to the 1LT8. The keyboard is read asynchronously every millisecond when the CPU drives its output register lines, A9 to A17, all high and reads its input register lines, A0 to A5. When a key is pressed, contact is made between an input register line and an output register line, putting a high level on the input register line. This high level generates an interrupt, causing software to scan the keyboard to determine which key is pressed. The **ON** key is not scanned but is wired to V_{DD}. This allows the system to be turned on while in deep sleep. The **ON** key is the only key capable of generating an interrupt and waking the system up. All key lines are isolated from the main system address lines by built-in 4-kΩ carbon graphite resistors.

The logic board contains five ICs: a 256K-byte ROM, a 32K-byte RAM, two column drivers and the 1LT8, which contains the CPU, an LCD driver controller, a memory controller, and a UART for RS-232 and IR I/O control. Also on the logic board are 36 discrete components, two 40-pin card connectors (for plug-in cards), one four-pin RS-232 connector, 202 pads for connection to the display, and 17 pads for keyboard contact. All of this is on a printed circuit board that measures 5.1 inches by 2.75 inches (see Fig. 4).

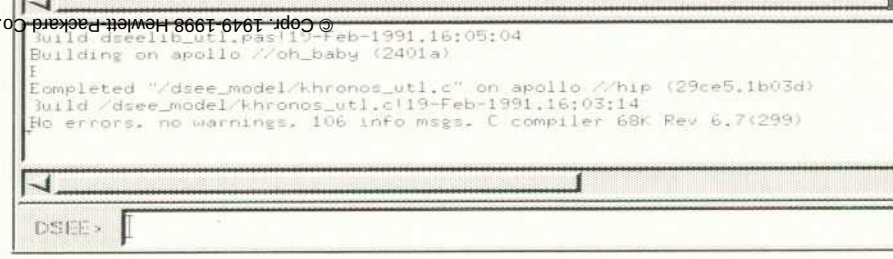
During product development, three logic boards were designed: the 256K ROM version that was released to production, a 32K OTP* EPROM version that contained self-test code used for initial shakedown and environmental testing, and a 256K EPROM version that was used for software development, debugging, and quality assurance.

For production testing, 77 logic board traces have dual test points. These test points are probed by a special test block that is connected to an HP 3065 test system. The HP 3065 tests all discrete components and ICs before the unit goes to final assembly.

The system is powered by three AAA batteries and has three power supplies, which are controlled by the 1LT8 chip. The V_H (8.1 to 8.9V) supply is used for the LCD display and RS-232 voltage swings. V_{DD} (4.1 to 4.5V) is the main logic supply. The V_{CO} (4.1 to 4.5V) supply is derived from the V_{DD} supply and is used to power the ROM and plug-in cards.

The power supply requires only two discrete diodes, an inductor, an n-channel power MOSFET, and three filter

*OTP: One-time programmable.



ical. If a critical dependency changes, DSEE will rebuild the module. If a noncritical dependency changes, DSEE will record what compiler option or source version was used, but it will not rebuild the module. There are also override mechanisms through which the user can force DSEE to rebuild even though the bound configuration thread doesn't call for a build, or not to build when DSEE thinks it has to.

Before DSEE starts building it sequences the required builds according to their interdependencies. As the user's translation script executes, it reads the desired version of each DSEE element directly out of its library. After the program is built and debugged, the user can make a release, or permanent snapshot of the bound configuration threads, binaries, sources, tools, and system model that make up the program. With this feature, when customers report a bug two years later, the engineer repairing the code can rebuild the same bits, plus the fix.

Version 3 of DSEE added support for parallel building on more than one Apollo computer at the same time. DSEE tracks the CPU utilization of candidate build computers, and starts builds on the most lightly loaded machines. Fig. 6 shows parallel builds running on several machines. The performance improvement provided by parallel building can be substantial. For example, the time to run the Ada validation test suite drops from 80 hours to 17 hours when it is built in parallel.

If an executable is compiled with an arbitrary collection of stdio.hs, it might take weeks to track down a problem caused by inconsistency in the include file. DSEE ensures that all builds use the same set of tools and system include files by building relative to a reference directory.

Heterogeneous Configuration Management

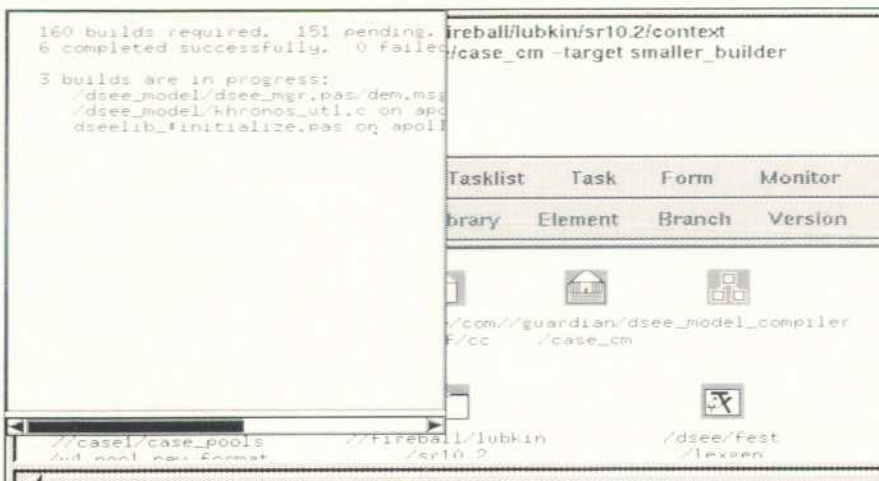
Many users want to use DSEE to develop software for non-Apollo platfor
figuration manager
tion management. V
that are based on th
the setup for heter

DSEE can be us
targets as well; it'
rule is an arbitrary
a non-UNIX target
back into a pool.

Host Types. To ha
agement, each syst

*UNIX is a registered trade
countries.

Fig. 7. *The ar
ment.*



0.050-inch pitch. Towards the memory card, the pins are gold-plated and cantilevered to provide a contact force of 30 grams to 70 grams each. Towards the HP 48SX printed circuit board, the contacts are solder-plated and terminate in either a vertical or 35°-off-vertical surface mount butt joint. Since the two memory cards can share all but five pins, 35 pairs of pins are soldered to one printed circuit board pad each. The two copper ground pins, which are press fit into the body, push open the memory card shutter and provide a path for ESD from the card panels directly to the printed circuit board ground plane. The 40% glass PPS alignment comb is press fit onto the body after all 80 pins are inserted and formed, providing alignment of the pin tips onto the printed circuit board pads.

Overall, the connector measures 2.378 inches wide by 1.555 inches long and stands 0.556 inch off the printed circuit board. Although two cards are held stacked, the custom connector is less than 1.5 times the thickness of the manufacturer's original single-card connector.

While design details pertaining to body strength, pin geometry, and pin plating were adapted from the original connector, the heatstaking process to which the HP 48SX memory card connector had to conform was a challenge. Basically, after insertion through the printed circuit board, the heatstake pins are flattened using temperature and pressure, thus preventing removal. The flattened portion, called the heatstake "head", consists of very rigid and brittle glass fibers in a matrix of remelted PPS. The head must not deform under the force of the 80 preloaded contact pins during reflow soldering, or solder defects will result from lifted pin tips. In addition, the head must be tough enough to endure repeated shear stress during card insertion testing and repeated tensile stress during drop testing. To complicate matters further, the heater block on the printed circuit board assembly line needed to heatstake the memory card connector, serial connector, and TAB clamp simultaneously to minimize cycle time. To establish optimum heatstake head geometry, heater block pressure, heater temperature, and melt time, repeated staking and subsequent drop testing were conducted. Initially, ten stakes of 0.062-inch diameter were used but proved to be brittle in drop testing. Next, the stake diameter was increased to 0.118 inch but the resulting force imbalance on the heater block caused the serial connector heads to be loose. After much trial and error, the combination of four 0.118-inch-diameter heatstakes, a 0.147-inch head diameter, a 440°F heater temperature, and a 13-second melt time produced the toughest heatstake heads.

The memory card connector thus implemented combines the benefit of the manufacturer's proven design with custom details incorporated to produce the highest-performance part using predetermined assembly processes.

Performance

Tests proved the memory card and connector for the HP 48SX tested be a reliable combination. No functional problems developed after 20,000 cycles of insertion/removal life testing consisting of 1000-insertion sequences alternating with 24-hour periods in a supersoak chamber. The card survived one-meter drop testing, both alone and plugged into the calculator, as did the connector. The card also experienced no mechanical damage when exposed to 25,000-volt ESD, both alone and plugged in.

Conclusion

The goals of design and manufacturing leverage were met by using an OEM memory card and incorporating off-the-shelf connector design details where applicable. Custom features were added, however, when increased reliability and manufacturability were deemed necessary. Thus the best solution for both customer and project was implemented in the HP 48SX memory card and connector.

M. Jack Muranami
Mechanical Engineer
Corvallis Division

capacitors (see Fig. 5). This is a boost-switching power supply in which the 1LT8 chip controls the current in an inductor, which is connected to the batteries, via the MOSFET. When one of the supplies (V_H or V_{DD}) is low, the 1LT8 pulses the MOSFET at a 122.84-kHz rate, increasing the inductor current. The current from the inductor is then dumped through one of the diodes, charging its filter capacitor. If both supplies are low the 1LT8 switches the charge between them at a 30.72-kHz rate.

To conserve battery life, the power supplies (and the product) have three modes of operation:

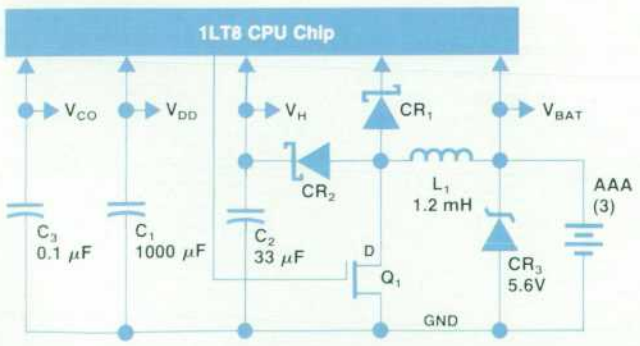


Fig. 5. *HP 48SX power supply schematic diagram.*

- Running. The 1LT8, column drivers, RAM and ROM, power supplies, and plug-in ports are all powered. Battery current is 9 mA.
- Light sleep. In this mode the 1LT8 turns off and battery current drops to 4 mA. This mode is entered whenever the CPU is inactive and a key is not being pressed. The 1LT8's display controller accesses memory every 244 μs to update the display. When the update is complete, the address lines switch and check to see if a key is pressed. Pressing any key will turn the CPU on.
- Deep sleep. All supplies are turned off and battery current drops to 12 μA. The V_{DD} supply floats to the battery voltage (V_{BAT}) which supplies power to the **ON** key, 1LT8, and RAM. This mode is entered when the CPU has been inactive for ten minutes or the unit has been turned off. To wake the system, the **ON** key must be pressed.

The 1LT8 chip also monitors the battery voltage. When the voltage falls to between 3.4 and 3.0 volts, the low-battery annunciator is turned on. If the batteries are not changed and the battery voltage falls below 1.5 volts, the system turns off. A 1000-μF capacitor maintains the V_{DD} supply for several minutes while the batteries are being changed.

The 64-row-by-131-column STN LCD is driven by two commercial column drivers, each driving 64 columns, and the 1LT8 which drives 64 rows, 3 columns, and 7 annun-

ciator lines. The column drivers receive their data, timing and control signals, and voltage levels from the 1LT8. One of the problems with the commercial column drivers is that they require a negative voltage. To overcome this, we connect their +V line to our V_H (+8.5V) supply, their GND to V_{DD} (+4.4V) and their negative supply to GND. This requires all data and control signals received from the 1LT8 to swing from 4.4V to 8.5V. Display data is stored in system RAM, and the 1LT8 display controller interrupts the CPU for 22 to 23 μs every 244 μs to access it. As display data is received, it is serially shifted to the column drivers. When the column drivers have received 128 bits of data, they store it and output it to the display synchronously with a row driver output from the 1LT8.

For ease of expanding the HP 48SX's capabilities, dual 40-pin connectors are installed on the logic board. These connectors will accept credit-card-size plug-in RAM or ROM cards. Each connector has its own chip select line but all address and data lines are common to the internal ICs. Each connector can accept up to 128K bytes of memory.

The 1LT8 tests the connectors to determine if a card is present and if it is write protected. It does this by checking the card's write protect output. If the write protect signal is high, a card is plugged in and can be written to (RAM). If the output is low, a card is present and is write protected (RAM or ROM). If the line is floating, no card is present.

RAM cards have their own lithium keep-alive batteries. When the HP 48SX goes into deep sleep, the power supply to the cards (V_{CO} supply) is turned off. When the supply drops to between 3.9 and 3.5V, the RAM switches to its internal battery. The lithium voltage is sampled by the 1LT8, and when it drops to between 2.5 and 2.2V, a low-battery annunciator is turned on.

Acknowledgments

The design of the HP 48SX would not have been possible without the efforts of a number of people. Horst Irmscher contributed to the plastic tooling. Lonnie Byers did the printed circuit design. Pam Burkhalter and Deborah Cover were project coordinators. Recognition must also be given to the many model makers who did prototype work, Bob Livengood who gave direction and managed the early part of the project, Bonnie Miller and Donita Baron and the whole production team who helped build the prototypes, John Allen and the quality assurance team, procurement engineering, and all of the other people who contributed to the success of the Hewlett-Packard 48SX.

A project is very satisfying to the team when the objectives are met and customer response is enthusiastic. It is additionally satisfying when recognition is bestowed upon the product by related professional publications and organizations. The HP 48SX has received five awards for its innovative design including *EDN's* Innovation of the Year for computers and peripherals, and was one of ten products to receive *Electronic Products'* Product of the Year award. These honors reward the many people in HP who contributed to making the HP 48SX a successful product.

The HP 48SX Calculator Input/Output System

An RS-232 link allows communication with personal computers. An infrared link provides for printing and for two-way calculator-to-calculator communication.

by Steven L. Harper and Robert S. Worsley

WHEN HANDHELD CALCULATORS were first introduced in the early 1970s, they provided portable computation without much memory. The limited keyboard and one-line numeric display provided just about all the input/output needed to use this capability effectively. As the incorporation of advancing memory technology made possible the storage of larger amounts of data and even programs created by the user in these handy little machines, the keyboard and display became an intolerable bottleneck. The need to enter a thousand or so keystrokes manually to use a program that someone else has written is quite an effective barrier.

The first solution to this problem was to use small magnetic cards to store the data and programs. There were some difficulties with power consumption, physical size, mechanical wear and tear, and cleanliness, but this mode of input/output was the accepted standard for some time. Eventually, however, larger and larger memories began to outstrip the capacity of the cards. This, combined with a need to communicate with other types of devices that did not use magnetic cards, necessitated a new approach.

The HP-IL (Hewlett-Packard Interface Link) was an electronic interfacing system that was designed with the needs of calculators in mind. It allowed systems with several devices to be configured automatically and controlled by a calculator. These devices included printers, mass storage, adapters to other interface systems, and even instruments. In many ways, it was superior to existing electronic interfaces, but this was not sufficient to overcome the inertia of the massive installed base of these other devices. Prices of calculators continued to drop and patterns of use changed as personal computers became ubiquitous. For these and other largely nontechnical reasons, HP-IL was no longer the ideal input/output medium for the majority of HP calculator users.

One area of serious complaint with calculators and electronic interfaces had been the cost and inconvenience of the cables. In addition, some calculators were so thin that even the HP-IL connectors, designed for small size, were unacceptably large. The most pressing need for these machines was to provide some form of hard-copy output to a low-cost portable printer. Drawing upon technology similar to that used in infrared remote control of TVs and VCRs, HP introduced a printer and a line of calculators that met the basic need with an interface that gave customers what they wanted: no cables at all. The only disadvantage was that this infrared connection was output-only.

Input/Output Needs of the HP 48SX

Market research indicated that an overwhelming majority of high-end calculator users either owned or had access to a personal computer. Clearly, it would be an advantage to be able to perform the data-entry-intensive tasks with the large keyboard and display of the personal computer and the computation-oriented work with the simple and powerful applications resident in the calculator. In other cases, the portability of the calculator was essential for part of the job and the speed and capability of the personal computer and its associated peripheral devices satisfied the rest of the need. These considerations forced the primary objective for the input/output capability of the HP 48SX scientific expandable calculator: an easy-to-use connection to existing personal computers.

In addition, there were two secondary objectives. While most users would choose to satisfy their needs for hard copy with the printer connected to their personal computer, there would be some applications where portable printing was essential, and perhaps some users who needed simple hard copy without access to a personal computer system. Also, our experience with the HP 41 calculators had shown that users of this class of machines do a lot of sharing of programs and data. In consequence, it was felt necessary that the HP 48SX be compatible with the existing infrared printer and that there be a convenient way to do input/output directly between calculators. Fig. 1 illustrates the desired input/output connections.

The most common interface on personal computers is the RS-232 serial port. It would be much too costly and inconvenient to require the customer to buy a special card for the personal computer, such as an HP-IL or infrared interface, to connect to the calculator. Development of these cards would be costly, too, since there are several different types of personal computers. These considerations quickly led to the decision to design into the HP 48SX the capability to connect directly to the serial port already in nearly all personal computers. This required a custom connector and cable, since the traditional DB9 or DB25 connectors used with RS-232 were much too large.

The requirement for program and data sharing between calculators could also be satisfied with RS-232, but the need to have the cable and/or some special calculator-to-calculator adapter was a severe drawback in this situation. A one-way infrared interface was already needed to maintain compatibility with the infrared printer. Would some enhancement of this interface fill the need without requir-

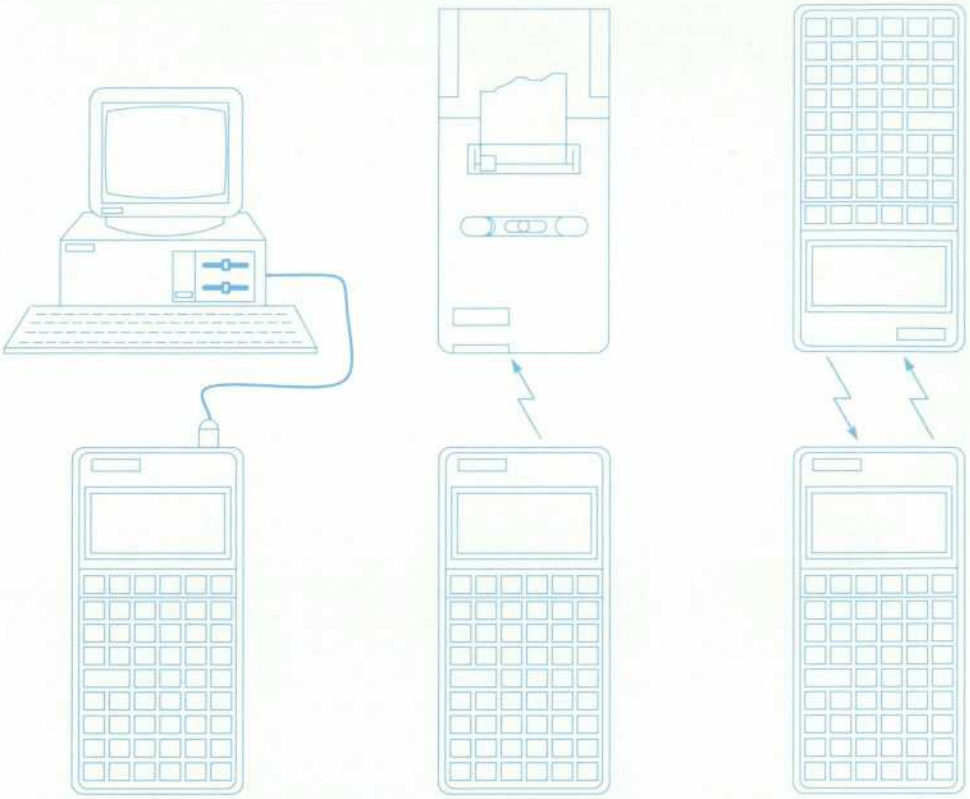


Fig. 1. *The HP 48SX calculator can communicate with personal computers or other RS-232 devices, with the HP 82240 portable infrared printer, and with another HP 48SX via a short-range two-way infrared link.*

ing the user to carry around a cable? Some investigation showed that a simple infrared receiver circuit for very short distances could indeed be included with minimum impact on the design.

The input/output solution for the HP 48SX is really two solutions, each optimized to particular requirements: the serial port for the important connection to personal computers, and a new two-way infrared interface for compatibility with the portable printer and for calculator-to-calculator communication. The plug-in memory cards, which are essential for memory expansion, might also be considered part of the solution. The RAM cards with built-in battery can be used for archiving and backup purposes and for program and data sharing between calculators as well. This provides only a partial solution, however, and the relatively high cost of the cards makes it very desirable to provide these functions in other ways also.

The input/output capability consists of not only the electronics, but the protocol and software to allow the user to move data easily. Here again, what was widely available for the personal computer was the important consideration. The Kermit protocol[1] was chosen because it provides automatic retransmission to correct errors, is widely used, and is available essentially free of charge for all the major types of personal computers. It was decided to include the Kermit protocol as one of the built-in applications in the HP 48SX. The Kermit protocol is applied to both the serial interface and the infrared interface, so the user only needs to learn one simple application for most input/output needs.

The Serial Interface

RS-232 uses bipolar signaling, that is, different logic states are indicated by voltages that swing both above and below the ground reference level. The HP 48SX does not otherwise need to generate a negative voltage, and it would seriously complicate the system design to have to do so. While integrated circuits are available to perform this function, their cost and power requirements are prohibitive in the calculator. One possible solution was to use unipolar signaling. While this is not really RS-232, most receiver circuits change the indicated logic state at about one volt positive, and will therefore function acceptably with a unipolar input. There are a few serial interfaces on personal computers, however, with input thresholds that really are set at zero volts. These would not work reliably with unipolar signals, and worse yet, customers have no easy way to tell which type of interface they have.

The solution to the problem was found in clever use of the voltages the HP 48SX generates to drive the LCD display. The logic in the calculator works between zero volts and the V_{DD} supply, about 4.3 volts. The display works between zero volts and the V_H supply, about 8.4 volts. Because the calculator has no external ground connection to any other device, it does not matter what voltage level we use as the ground reference for the serial port. By using the V_{DD} supply as the signal ground, we can drive the TXD (transmit data) line to calculator ground and it will be at -4.3 volts as viewed by the receiving device. Likewise, driving the line to V_H will make it appear as $+4.1$ volts ($V_H - V_{DD}$) at the serial port. The result is a bipolar signal that will work with all serial ports without the additional expense and power of special integrated circuits.

Technically, an RS-232 device is required to swing at least 5.0 volts above and below signal ground. After various circuit losses, the HP 48SX voltage swing is only a little more than three volts positive and negative under worst-

case conditions. It would have been convenient if the power supplies were at a slightly higher voltage, but this was not possible since the CPU integrated circuit has a maximum voltage limit of about nine volts. Even though this does not strictly comply with the requirement, it works quite well with short cables since an RS-232 receiver is required to indicate a logic zero for +3.0 volts or more and a logic one for −3.0 volts or less. The slightly reduced noise margin has had no noticeable effect.

It would have been convenient, though slightly more costly, to use a standard driver/receiver integrated circuit to provide the necessary short-circuit protection and comply with other interface requirements. Unfortunately, these parts are not specified at the low voltages used by the HP 48SX and their higher voltage drop would have caused the output to be less than what was needed. Strict limitations on the amount of current that the two power supplies in the HP 48SX can source or sink also mandated a discrete circuit to satisfy all the needs.

Considerable experimentation resulted in the circuit of Fig. 2. When the TX line from the CPU is driven high (V_H), current flows through R2, Q8, and CR5 to the TXD pin and the load in the receiver of the other serial device. If this current exceeds about 4 milliamperes, the voltage drop across R2 will turn on Q7. This will cause the voltage at the base of Q8 to rise, tending to turn Q8 off. Thus the current that will flow is limited regardless of the voltage on the TXD pin. Because the circuit is symmetrical, precisely the same action occurs with Q2 and Q3 when TX is driven low (calculator ground). The Schottky diodes, chosen for their low forward voltage drop, are necessary to prevent reverse conduction through whichever side of the circuit is off. The capacitor on the TXD pin slows the rise and fall times of the signal to minimize electrical noise generation. The 12-volt Zener diode on the TX line provides additional protection to the CPU from electrostatic discharges. Capacitor C4 is needed to provide dc isolation between the calculator shield, which is at calculator ground potential, and the shield of the other device, which is likely connected to signal ground, which is connected to the calculator V_{DD} supply. Without this capacitor, the calculator V_{DD} power supply would be short-circuited.

Receiving RS-232 signals in the calculator is a relatively simple matter. When the RXD (receive data) line swings positive, the 2.4V Zener diode allows the voltage to rise above the logic threshold of the RX input on the CPU chip (about one volt above V_{DD}), but clamps it below the CPU V_H power supply so that excessive current cannot flow. A similar situation occurs for negative swings as the diode conducts in the forward direction. The 5.6-kilohm resistor limits current and presents the proper impedance for an RS-232 input. The 1.5-kilohm resistor adds additional short-circuit and ESD protection for the CPU RX input. The 75-ohm resistor merely provides a protective current limit between signal ground and the V_{DD} supply.

The Infrared Interface

Transmitting infrared light is relatively simple from the viewpoint of the hardware. Fig. 3 shows the schematic diagram for the infrared transmitter and receiver circuits. Because of the high current pulses needed to drive the infrared LED, it is connected directly to the batteries, greatly reducing peak demand on the calculator power supply. The other end of the LED is driven low by a special constant-current driver circuit integrated on the CPU chip. Pulse duration and timing are controlled by a combination of hardware in the CPU and firmware.

A phototransistor was chosen as the receiving element. While much slower than a photodiode, it is fast enough for the data rate of 2400 bits per second, and the sensitivity is much greater. Since high ambient light levels can cause relatively large currents to flow in the phototransistor, Q4, some means had to be found to reduce or stabilize the bias current. It would not be acceptable for battery life to be considerably shorter in sunlight than in the office. For this reason, the phototransistor chosen is one that has all three terminals accessible. The combination of this feature and transistor Q5 provides the needed stabilization. If the quiescent current of Q4 increases, this increase goes into the base of Q5, which in turn conducts more current away from Q4's base terminal, tending to reduce its quiescent current. In the office, the total current resulting from the infrared receive circuit is only about seven microamperes. In direct sunlight, this value rises to nearly 200 microamperes. This

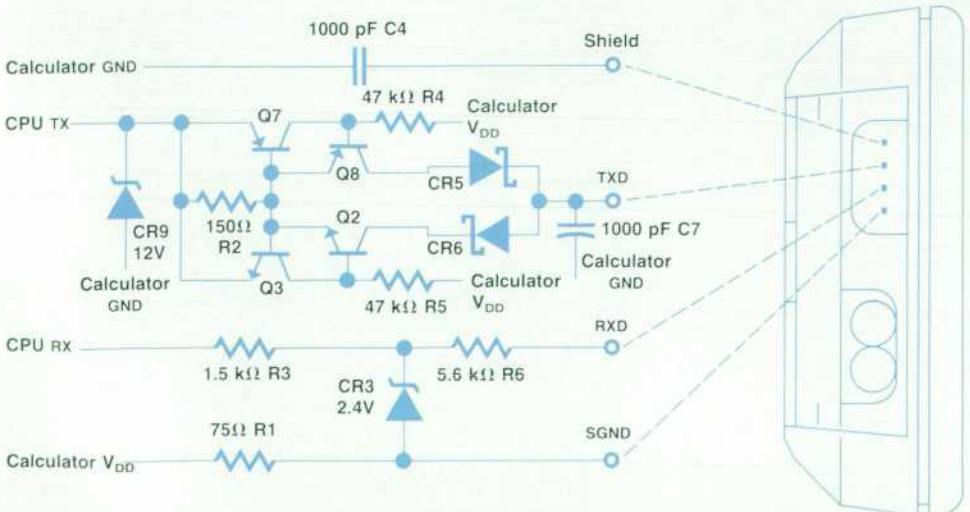


Fig. 2. *The discrete circuitry for the HP 48SX serial port provides protection for the calculator and compatibility with RS-232.*

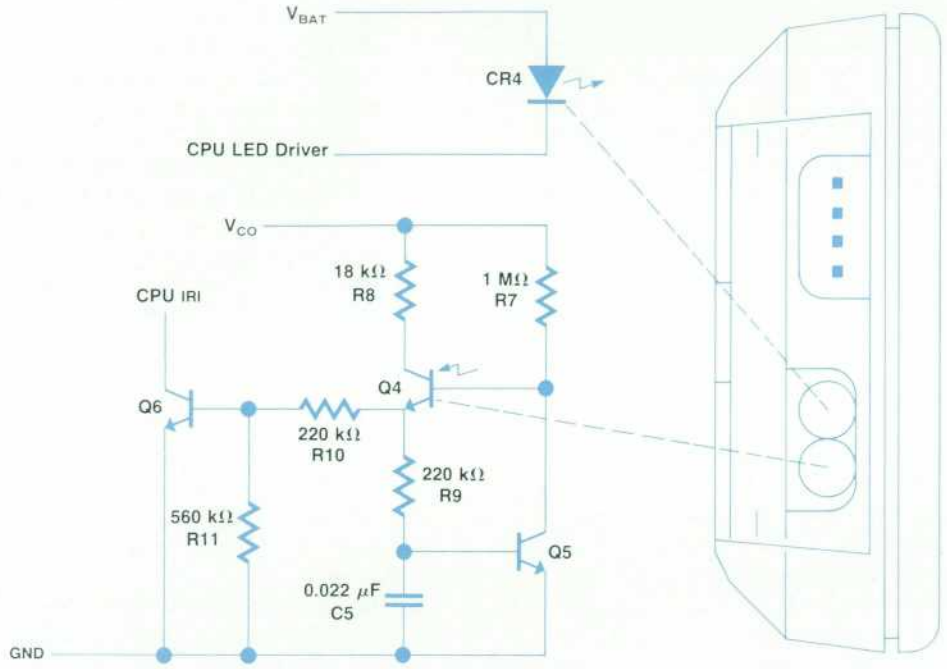


Fig. 3. *The infrared circuitry transmits to the portable infrared printer and provides two-way communication with another HP 48SX.*

seems like a large change, but without the stabilization circuit the sunlight value would be several milliamperes, and the circuit would saturate and not function.

Capacitor C5 is necessary to bypass the stabilization circuit at the signal frequency. R10 and R11 form a bias network and threshold-setting mechanism for Q6, the output amplifier and buffer. Q6 drives the IRI pin on the CPU, which includes a pull-up device. If the circuit were to draw current even while the calculator is turned off, high ambient light would result in a significant reduction of battery life. For this reason, the infrared receiver is powered by V_{CO}, a gated version of the V_{DD} supply which is switched off when the calculator is off.

While this system is limited to a range of only two or three inches, it is low in cost and provides the user a very convenient means of sharing programs and data.

The infrared coding for the printer has been discussed previously.[2] Coding for the two-way infrared data for calculator-to-calculator communication is very similar to coding of data from the serial port. First is a start bit (logical zero), which triggers the asynchronous receiving circuitry. Then follows the least-significant bit of data and so on to the most-significant data bit, for a total of eight data bits. If parity is enabled, the eighth data bit is replaced by the parity bit. The HP 48SX sends slightly more than two stop bits (logical one bits), but only requires one stop bit when receiving. Fig. 4 shows the bit coding for both the serial link and the two-way infrared link. A logical zero is represented by a single pulse of infrared light about 50 microseconds long. A logical one bit has no light pulse. When these pulses are received, a latch-and-sample circuit in the CPU converts this format into logic levels, which can then be routed into hardware shared with the serial port. Note that two-way infrared data is only sent at 2400 bits per second, whereas serial data can be sent at 1200, 2400, 4800, or 9600 bits per second.

The User's Point of View

As anyone who has used RS-232 knows, plugging the cables together is only the beginning. Even for RS-232 ports that don't require a response on the hardware handshake lines, the transmit data and receive data lines may need to be reversed. After getting the cable wiring right, the baud rate, parity, and software handshaking such as XON/XOFF must be set identically for both the transmitter and the receiver.

The HP 48SX design attempts to minimize cabling problems by using only the minimum three-wire RS-232 connection (transmit data, receive data, and signal ground) and matching the cable pinout to the most commonly used RS-232 ports. A setup menu is included in the first row of the I/O menu to allow the user to see and modify the current setting for baud rate, parity, and other parameters. Since the HP 48SX uses XON/XOFF handshaking only for low-level I/O commands, the XON/XOFF setting isn't included in the setup menu. Most of the parameters in the setup menu are stored in a variable named IOPAR, which also contains separate XON/XOFF handshaking flags for receive and transmit.

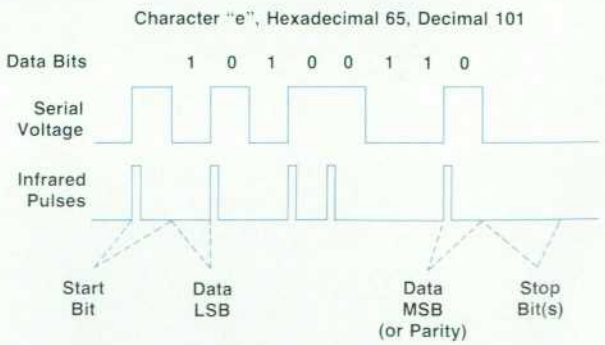


Fig. 4. *Coding of data in the two-way infrared protocol is similar to that in the serial link. A logic zero has a pulse at the beginning of the bit time; a logic one bit has no pulse.*

The higher-level I/O commands don't use XON/XOFF handshaking since they follow the Kermit protocol, which has packets of at most 96 bytes.

The simplest I/O interface for the user is the two-way infrared interface, which has no wires to mix up, doesn't use XON/XOFF handshaking, and is fixed at 2400 baud and no parity.

All I/O commands on the HP 48SX automatically turn on and initialize the appropriate I/O port (RS-232 or infrared) based on the data from IOPAR. Once the port is ready, the HP 48SX interrupt system receives bytes for either RS-232 or two-way infrared I/O. The HP 48SX can respond quickly enough to incoming bytes so that even at 9600 baud the interrupt system will read the first byte before it is overwritten by the next one. Incoming bytes are placed in a 255-byte input buffer. The interrupt system waits for about four byte times before concluding that the incoming byte stream has ended. Then, before exiting, it checks for other sources of interrupts such as cursor blink, clock ticks (if the clock is ticking in the display), alarms, cards pulled out or added, or keys down or up.

All of these checks can make the time required to exit the interrupt system and then reenter long enough that incoming bytes may be missed. For this reason, alarms, key presses, and the clock display should be avoided during I/O.

Low-Level I/O Commands

The HP 48SX provides a triplet of low-level I/O commands: XMIT (transmit), BUFLEN (buffer length), and SRECV (serial receive). They are intended for use in programs, so instead of stopping the program when an I/O error occurs, they simply return a succeed/fail flag to level 1 of the stack. This allows the programmer the option of handling the error without having to use IFERR. XMIT transmits a string from the stack, using XON/XOFF handshaking if transmit pacing is enabled. BUFLEN tells how many bytes are in the input buffer. SRECV reads a specified number of bytes from the input buffer (waiting for more to come in if necessary) and returns them as a string to the stack.

Supporting commands are STIME (serial timeout), SBRK (serial break), OPENIO (open I/O port), and CLOSEIO (close I/O port). STIME sets the length of time that XMIT (XOFF received, transmit pacing enabled) or SRECV will wait before giving up if the data flow is interrupted. SBRK sends a serial break. OPENIO and CLOSEIO turn the I/O port on and off.

As pointed out earlier, bytes can be lost during I/O transmissions. If the I/O connection is noisy, bytes can be garbled. It is also possible that some communication channels will remove certain bytes (usually control characters) from the data stream. The low-level I/O commands have no protection from this type of data corruption except for parity checking. The Kermit protocol chosen for higher-level I/O commands overcomes these problems to give more reliable communication for transferring programs and data.

The Kermit Protocol

The Kermit protocol encodes control characters as sequences of ordinary printable characters so they can pass safely through to the destination. Garbled data is detected by the checksums on each packet and lost packets are detected by the sequence number on each packet. A Kermit protocol packet consists of a mark byte to mark the start of a packet, a length byte, a sequence number byte, a type byte, data bytes, and finally one to three checksum bytes. The type byte defines the type of the packet.

A typical Kermit protocol transmitter sends the following packet sequence. After sending each packet it waits for the receiver to send either an ACK packet (type Y) to indicate successful receipt of the packet or a NAK packet (type N) to indicate a garbled packet. If the receiver doesn't respond, the transmitter can time out and resend.

Sequence Number	Packet Type	Description
0	S	Negotiates parameters such as maximum packet length, time-out, and control character encoding.
1	F	Contains the name of the file to be sent.
2	D	Data (as many packets as necessary).
.	D	
.	D	
n	D	
n+1	Z	Marks the end of this file. May be followed by another F,D,...,D,Z sequence or by B.
n+2	B	Marks the end of the whole transfer.

The packet sequence numbers wrap back to 0 after packet 63. If either the transmitter or the receiver encounters a fatal error, it can send an error (type E) packet to tell the other unit to give up also.

The Kermit protocol has another mode setting in addition to parity and baud rate: text versus binary. Computers sending text files are required to end each line of text with a carriage return character followed by a linefeed character. If a computer normally uses some other line terminator, such as a single linefeed, it must transform linefeed to carriage return plus linefeed when sending text files, and must translate carriage return plus linefeed to linefeed when receiving text files. If a computer normally terminates text lines with carriage return plus linefeed, no transformations are needed. This works fine for text files but not for binary files like compiled programs unless the transmitter and receiver both do the transformations or both don't do them. Therefore, to send a binary file, a computer that has text and binary modes must be set to binary.

The HP 48SX greatly extends this concept of text (ASCII) versus binary files by automatically converting an object to string form when sending in ASCII mode, thus converting a binary object into its text form. In addition, the HP 48SX has a 256-character character set based on the ISO 8859 Latin 1 character set. This is not compatible with many current PC character sets, so translation modes were added to ASCII transmission to convert the new character codes to sequences of normal ASCII characters. To ensure the accurate interpretation of the automatically generated text form of objects, a header string is prepended to the object. This string contains the modes in effect when the text form of the object was created. The modes are the

translate mode (to allow proper "untranslation"), angle mode (in case the object contains a complex number in polar form), and fraction mark (to distinguish decimal points and argument separators). The header string also allows a receiving HP 48SX to know that it should receive this object in ASCII mode. A short header is also prepended to binary objects to instruct a receiving HP 48SX to receive in binary mode. At the cost of this small amount of extra overhead, a receiving HP 48SX can correctly interpret an incoming file even if its modes are set differently than would be required for that file.

Acknowledgments

Preston Brown and Les Moore did the investigation and design of the IR receiver circuit. Tom Lindberg designed the HP 48SX serial connector and cable. A special thanks goes to the inventors and implementors of the Kermit protocol. The wide availability of this protocol on many machines greatly enhances its value in the HP 48SX.

References

1. F. da Cruz, *Kermit—A File Transfer Protocol*, Digital Press, 1987.
2. S.L. Harper, R.S. Worsley, and B.A. Stephens, "An Infrared Link for Low-Cost Calculators and Printers," *Hewlett-Packard Journal*, Vol. 38, no. 10, October 1987, pp. 16-21.

Manufacturing the HP 48SX Calculator

Sharing manufacturing processes with earlier, simpler calculators shortened development time and improves manufacturing efficiency. The HP 48SX and the simpler calculators also share the same production line at the same time—a concept known as coproduction.

by Richard W. Riper

THE HEWLETT-PACKARD 48SX is an advanced scientific calculator that reaches new levels of capability and performance. Rather than start with a clean sheet, the design team looked to simplify HP's calculator line when developing the HP 48SX, in particular by making use of common manufacturing processes. This reduced the time to develop the calculator. Sharing common assembly techniques with other HP calculators has also led to improved production efficiency and increased flexibility.

Common Processes

When work started on the HP 48SX, design engineers were given the goal to use as many of the design concepts from a 1-and-2-line display family of calculators (HP 10, 14, 17, 20, 21, 22, 27, 32, and 42) as possible. This was done to reduce the amount of new development needed and to allow the new machine to be built on the same production line as its simpler cousins. The 1-and-2-line family set new standards in HP for calculator manufacturing efficiency and we wanted to extend this efficiency to the HP 48SX. Using proven designs shortened the typical design/build/test/redesign cycle, shortening the whole design process. It also meant that there were no new manufacturing processes to develop. This reduced the cost and delivery time for the assembly tooling. Since these tools were for familiar tasks, experience gained from the first set of tools led to better, more refined tools for the HP 48SX.

Mixed Production

The major difference between the 1-and-2-line family of calculators and the HP 48SX is in the complexity of the electronics. Most of the assembly steps needed to build the HP 48SX happen as quickly as they do for the simpler calculators. However, the HP 48SX's complexity requires inspection and test times three to four times those of the simpler machines. Since the HP 48SX is designed to be built on the same production line as the simpler models, this could have led to some major line balancing problems.

One solution to this problem would be to duplicate the inspection and test stations so that the whole line could run at the same high rate as it does for the simpler calculators. This would cost a great deal for duplicate test equipment. Also, more people would be required when building the HP 48SX, who would not be needed when the simpler product was being produced.

Another solution would be to run the whole line at a slower rate to match the longer test and inspection times. This would lead to lower efficiency, since many of the faster operations would sit idle through part of the longer cycle time.

To solve this problem, we decided to build both the HP 48SX and the simpler calculators at the same time using a process we call coproduction. One of the 1-or-2-line calculator models is built at the same time and on the same assembly line as the HP 48SX. The products are mixed

uniformly, in the desired quantity ratio. The assembly steps shared by both machines run at the higher rate of the original 1-or-2-line assembly line. Since the test and inspection stations are peculiar to each product, the more complex HP 48SX can spend the longer time needed for testing, while several of the simpler products get tested at the same time.

The overall result is a line that builds almost as many combined units as the simpler line did by itself. It can run this mix constantly, rather than building one model of calculator for a time and then switching to the other. This eliminates large changes in the number of people required on the line and leads to better use of expensive equipment. The mix ratio is not fixed, so changes in the number of each product produced can be made to suit changes in orders.

Conveyor Production

The production of the HP 48SX is divided into two main areas: printed circuit assembly, where the circuit board is loaded, soldered, and tested, and final assembly, where the loaded circuit board is brought together with the case parts, display, and keyboard. In both areas a pallet conveyor is used to move the unit from station to station. The pallets are rectangular steel plates surrounded by plastic frames (Fig. 1). The plates have details to hold the parts in position.

The pallets ride on plastic belts, which never stop moving. At each station a stop mechanism halts the pallet while the flat belts continue to slide by underneath. At stations where accurate part location is required, the pallet is lifted off the belts and registered by the precision bushings that hold the pallet together.

Each conveyor system is made up of two sections of belt side by side. The assembly steps are done on the side nearest the operator, while the conveyor on the back side is used to move empty pallets back to the head of the line. The pallets follow this flat, rectangular loop and never need to be removed from the conveyor.

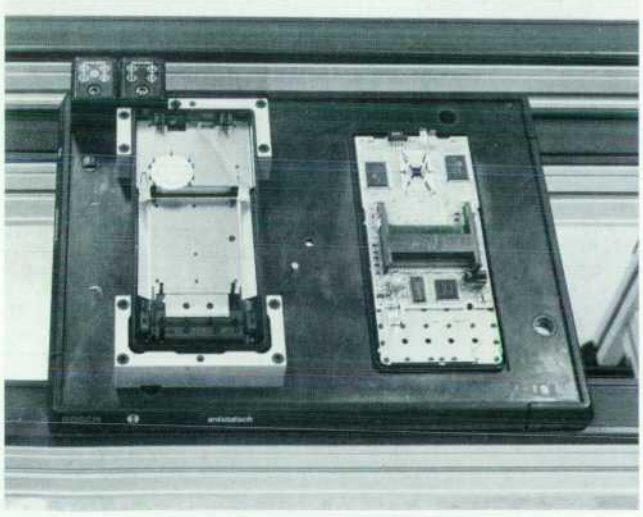


Fig. 1. *Assemblies are carried on pallets that ride on conveyor belts. The line has seven robot stations.*

Printed Circuit Assembly

Printed circuit assembly starts with an operator applying solder paste to the bare circuit board. This paste is made up of small balls of solder held in a solvent base. A printer pushes the paste out though a stencil (0.004-inch-thick brass) onto the pads for the component leads. The operator then loads the board onto an empty pallet on the conveyor (Fig. 2). All of the pallets on this conveyor are the same, and can hold the HP 48SX boards as well as those for the simpler products. The pallet then moves into the first of seven robot stations. Each station has a sensor to determine if the board on the pallet is an HP 48SX board or a board for a 1-line or 2-line calculator, so that the robot will load the correct parts.

The first robot loads the 1LT8 CPU (central processing unit) onto the HP 48SX board. Instead of using the traditional integrated circuit package consisting of a plastic body with leads, the CPU is packaged on a continuous tape (Fig. 3). This is known as TAB (tape automated bonding). Not only is this package style thinner than plastic bodies, but part handling is reduced. The ICs are simply rolled up on this reel of tape like 35-mm movie film. This tape is fed through a machine that shears the leads out from the tape and reforms the lead ends to the optimum shape for soldering. The robot first uses a reflective optical sensor in its gripper to locate the gold-plated traces and solder pads accurately on the printed circuit board. It then picks up the IC from the feeder and places it on the board, correcting its position based on the optical search.

At the second robot, connectors for the plug-in cards and input/output port are loaded. Also loaded is a TAB clamp, which will hold the leads on the TAB IC down while the solder paste is reflowed, ensuring a high-quality solder joint. After this robot the board moves into a heated press that forms rivet-type heads on plastic bosses, which will hold the connectors and the TAB clamp tightly to the board. At the third robot, the random-access memory (RAM) is loaded, along with a small coil spring that will make contact with a piezoelectric beeper. This spring eliminates the need for hand-soldered wires to the beeper. The fourth robot doesn't load any parts on the HP 48SX board, but is used on some of the simpler products.

At this point the simpler boards are completely loaded. The pallets carrying these boards are moved to the back belt, where they travel up to the second robot's area. There the second robot takes the board off the pallet and places it on a simple flat belt conveyor to move it to the solder reflow machine.

The pallets holding boards for the HP 48SX continue down to robots 5, 6, and 7, where the rest of the components are loaded. Since only a fraction of the total pallets continue on, the cycle time at these last three robots can be longer. This is a good example of where the coproduction scheme pays off. These robots can load many more parts than if they were limited to the shorter cycle time of the first four robots. After the final robot, the pallets move to the back belt, where they travel up to the second robot. As in the simpler product, the HP 48SX boards are off-loaded to solder reflow. The empty pallets continue to the head of the conveyor line, ready to repeat the process.

After loading, the boards pass down a conveyor belt to

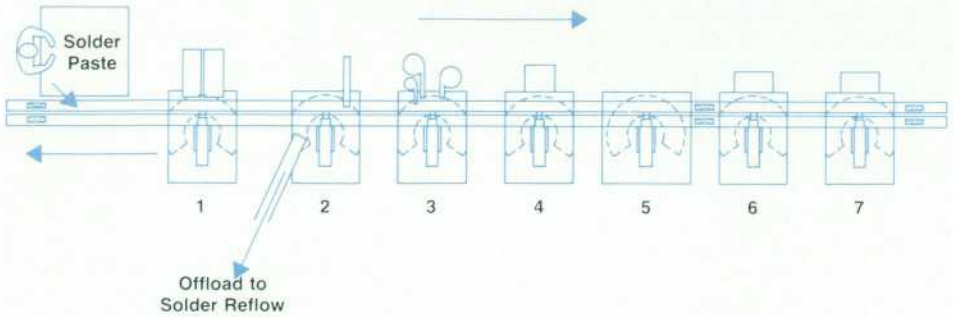


Fig. 2. *HP 48SX printed circuit assembly production line layout.*

an infrared reflow machine. The boards are then inspected under a microscope for solder joint quality and component alignment. Next the TAB clamps are removed, because there isn't room for them in the final product. A few components that are not surface mounted are added to the board by hand, and then the boards are washed to remove solder flux and other contaminants. The boards are next tested on an HP 3065 board test system, then passed on to final assembly.

Final Assembly

The final assembly line is similar to the conveyor used for printed circuit assembly, but the pallets are larger. Also, different pallets are needed for the HP 48SX and the simpler 1-and-2-line products. Sensors at each station read the pallet type and act accordingly. Instead of being totally automated, the final assembly line is a mix of manual stations with robots and other automated stations (Fig. 4).

The first station is a robot that loads plastic top and bottom cases into the pallet for the simpler calculators. This station is not used for the HP 48SX because the case parts are larger and could not be fed to the robot in the same way. At the next two stations, operators place case parts for the HP 48SX into the pallet and load keyboard parts for both products.

The next station is a pair of robots that work together to align and load the liquid crystal display (LCD) into the metal chassis and then load the chassis into the topcase, which is waiting on the pallet. One robot picks the LCD up from its shipping tray and places thin strips of double-stick tape along both long edges. It then passes the LCD off to the other robot. This second robot locates the LCD's contact pads under a reflective sensor and uses this search information to place the LCD accurately into the metal chassis. Locating details on the chassis accurately carry this alignment to the pads on the circuit board.

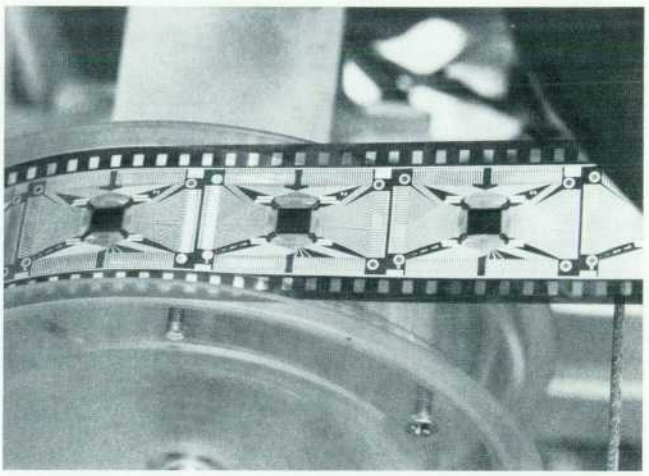


Fig. 3. *1LT8 ICs are packaged for tape automated bonding (TAB) and can be handled like movie film.*

The pallet then travels into the first of four heated presses used to hold the calculator together; no screws are used. Three of these four heatstakers work on both product types, so the product type is sensed and the top portion of the tool automatically shifts the correct heater block into place. The fourth heatstaker is only used on the 1-and-2-line products, so the HP 48SX pallets simply pass through. The first heatstaker forms rivet-style heads on bosses in the topcase,

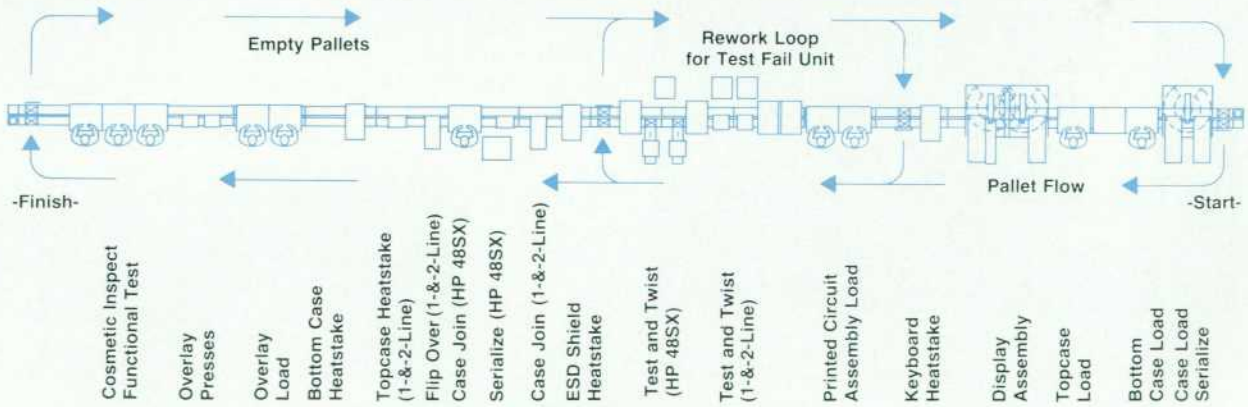


Fig. 4. *HP 48SX final assembly line layout.*

holding the chassis and keyboard parts tightly together.

The pallet now moves into a set of manual stations where the printed circuit assemblies are added. Elastomeric connectors known as zebra strips (for the alternating black carbon conductor bars), which make the electrical connection between the LCD and the printed circuit assembly, are loaded onto the LCD, and then the printed circuit assembly is added.

Automated Vision and Functional Tests

The pallet then moves to a series of testers. The testers for the 1-and-2-line family are located on the conveyor, and the HP 48SX pallets pass through these without stopping. The testers for the HP 48SX are located to the side of the main conveyor line just past the simpler testers. The HP 48SX pallets are automatically moved off the main line into the testers. This allows them to take longer to test without holding up movement of the simpler products, which test much more quickly. After testing, the HP 48SX pallets are moved back onto the main line to continue to the next station.

In each tester, the pallet is raised up against a test block (Fig. 5). This block contains spring-loaded probes which make electrical contact to test pads on the circuit board. Small air-powered cylinders can press on each key on the keyboard of the upside-down calculator, both to test the keyboard and to start a number of self-tests that are built into each calculator. Cameras mounted at the bottom of the tester look up at the display and an automated vision system checks to ensure that the whole display "lights up" as the calculator runs through its self tests. HP instruments check current levels and measure the operating frequency.

Once the calculator passes these tests, metal tabs formed as part of the chassis are automatically twisted to hold the printed circuit assembly, zebra strips, and LCD together. If the unit fails the test, a mechanical test failed code is set on a code block mounted on the pallet. Following the test stations is a mechanism to move failed pallets to the back belt, where they will loop back to the stations where the printed circuit assembly was loaded for evaluation. To aid rework, an error reporting system using an HP Vectra personal computer lets the operator at that station know which test the unit failed.

The pallets then pass through a heatstaker to stake in an aluminum electrostatic discharge shield and the piezoelectric beeper. At the next manual station, additional shielding parts are added by an operator and the cases on the HP 48SX are joined together. The cases are joined on the simpler products automatically. The calculators then pass through a pair of heatstakers to join the cases together permanently. Again, no screws are used.

Next, an operator places a printed overlay over the keyboard, which is pressed on at the next station. At the last set of stations, operators load the batteries and doors, and perform cosmetic and functional tests. The calculators are now ready to be packaged with the instruction manual in the box for shipment to the dealer.

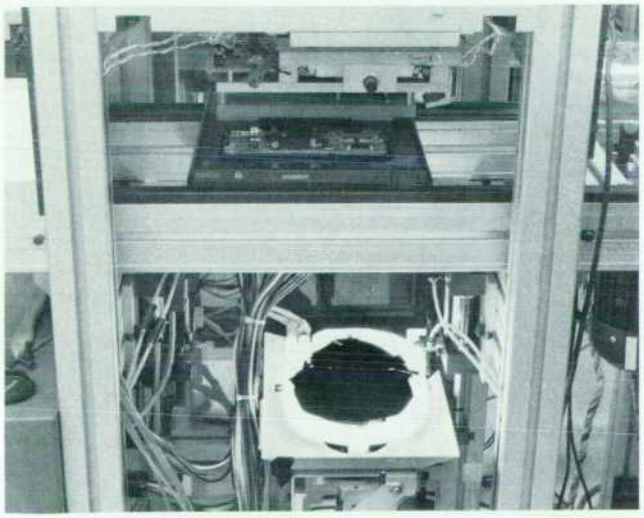


Fig. 5. *This tester uses two cameras and an automated vision system to inspect the calculator's display.*

Conclusion

As described above, many of the assembly steps for the HP 48SX are shared with similar steps for the simpler 1-and-2-line series of calculators. By basing the design of the HP 48SX on this earlier product line, the same production line can be used to build both type of products. By using the coproduction techniques described above, both products can be built at the same time, leading to better production efficiency and people balancing.

Acknowledgments

The other members of the manufacturing engineering team included Randy Brooks, Holly Drew, Ken Frazier, Carl Johnson, Mike Monroe, Colin Powers, Tom Revere, Ralph Sebers, Steve Terhune, and Bob Walsh. Thanks also to Donita Baron, Laureen Mangan, Bonnie Miller, Janet Souza, and the rest of the people in production who helped out during the prototype builds and to get the HP 48SX into production.

A 10-Hz-to-150-MHz Spectrum Analyzer with a Digital IF Section

The HP 3588A's digital resolution bandwidth filters offer better shape factors and can be swept four times faster than their analog counterparts. Narrowband zoom measurements using fast Fourier transform analysis can be hundreds of times faster. Extensive self-calibration, a help system with hypertext, and adaptive data acquisition also improve performance.

by Kirsten C. Carlson, James H. Cauthorn, Timothy L. Hillstrom, Roy L. Mason, Joseph F. Tarantino, Jay M. Wardle, and Eric J. Wicklund

THE HP 3588A SPECTRUM ANALYZER (Fig. 1) is a heterodyned, synthesized instrument with a tracking source. It is designed for spectrum and scalar network measurements from 10 Hz to 150 MHz. The key measurement contribution of the HP 3588A is fast, high-resolution, narrowband measurements. The HP 3588A is a pioneer in a new era in which spectrum analyzers will increasingly use digital signal processing technology to improve performance and features. By combining digital filtering and FFT (fast Fourier transform) analysis with traditional swept spectrum analysis techniques, the HP 3588A typically provides four times faster swept measurements and up to hundreds of times faster narrowband measurements than were previously available in a swept analyzer. The HP 3588A accomplishes this while achieving superior amplitude accuracy.

The HP 3588A brings digital signal processing technologies to RF spectrum analysis in two ways. For improved swept measurements, the HP 3588A is the first RF spectrum analyzer with an all-digital final IF (intermediate frequency) section. The selectable bandwidth of the IF section's digital filters determines the resolution bandwidth of the measurement. The filters have precise, predictable characteristics and performance that would be impractical

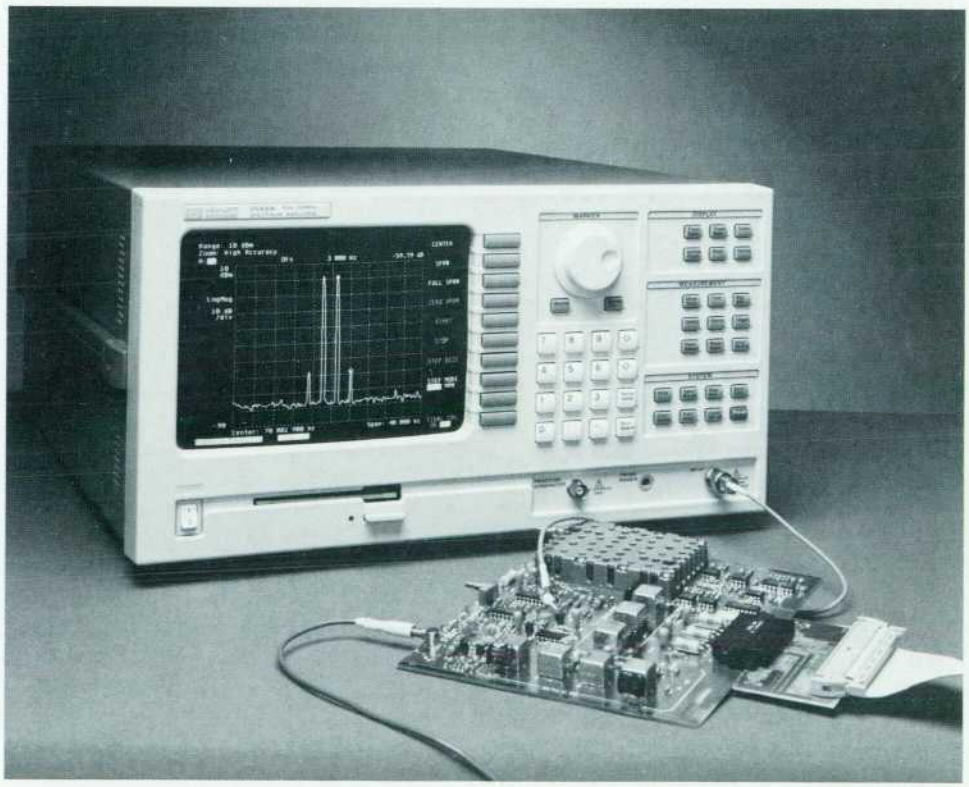


Fig. 1. *The HP 3588A is a new type of spectrum analyzer that greatly improves measurement speed in the frequency range of 10 Hz to 150 MHz. Resolution bandwidth is selectable from 17 kHz down to 0.0045 Hz. The new narrowband zoom mode allows fast Fourier transform analysis of spans up to 40 kHz wide anywhere in the frequency range.*

to achieve using analog technologies. Signal detection for swept measurements is also done digitally. FFT signal processing is used for a narrowband zoom mode. For measurement spans less than 40 kHz, narrowband zoom mode provides all the speed and resolution to 150 MHz that previously were only available to users of FFT-based analyzers.

For swept measurements, the digital resolution bandwidth filters provide shape factors of less than 4:1 with the same inherent accuracy obtained using more traditional IF filters with typical shape factors of 11:1. Since these filters are implemented digitally, their response is completely predictable and repeatable, and they can be swept faster than the traditional rate of one half the square of the resolution bandwidth. Any effects of the filters on the response are known and are calibrated out of the measurement. The result is a swept measurement that is typically four times faster and has better than two times narrower shape factor than analyzers with analog resolution bandwidth filters. It is important to note that these speed improvements are achieved together with the improved shape factors. If speed comparisons are made for equivalent 60-dB bandwidths, even more dramatic improvements are seen. Speed improvements under these conditions can be in excess of 16 times for swept measurements and 1000 times for narrowband zoom measurements.

FFT analysis for fast narrowband zoom measurements is available at any center frequency from 10 Hz to 150 MHz. Signals from the last analog IF section are digitized and subsequently filtered. Frequency spans from 40 kHz down to 1.2 Hz are available in this mode, and signals can be resolved with 4.5-mHz resolution at frequencies to 150 MHz. An ovenized reference oscillator is available to provide the frequency stability necessary to make these narrowband measurements.

All instrument functions are available over the HP-IB (IEEE 488.2, IEC 625), and the instrument can act as an HP-IB system controller as well. Optional HP Instrument BASIC is available for programming and automated testing. A 3.5-inch disk drive is an option for storage of instrument states, data, and Instrument BASIC programs.

Principal applications for the HP 3588A include general spectrum and scalar network analysis to characterize circuits and systems in laboratory and automated production environments. Examples of applications that benefit from some of the new features of this instrument include:

- Analysis of nonstationary or short-duration signals such as vibration-induced sidebands on oscillators
- Radio surveillance and measurements of moving sidebands on modulated carriers
- Circuit adjustments where fast feedback of results saves time
- Automated testing using the instrument's internal Instrument BASIC programming to control repetitive tasks (system controller capability over the HP-IB allows system integration without the need of a separate controller in ATE applications)
- Phase noise and broadband noise measurements (because of the detection scheme used, a true rms level is given without correction factors).

Development Challenges

Throughout the development of this product, many technical obstacles were overcome in integrating the digital IF and FFT technologies into the instrument:

- Custom gate arrays were developed for the digital filters.
- Because the instrument measures to 10 Hz, a special mixer design and a local oscillator (LO) feedthrough cancellation circuit were developed to reduce local oscillator signal levels in the IF path at low frequencies.
- To achieve good amplitude accuracy in FFT-based measurements, a special technique was developed to calibrate out analog IF frequency response effects.
- Residual spurious signals within the IF passband that would never have been resolved in a swept-only analyzer were initially seen using an FFT, requiring special attention in development to the reduction of spurious and residual signals.
- An autoranging input that operates over the full 150-MHz frequency range is provided.
- A multiple-loop local oscillator that includes a fractional-N loop as an interpolation oscillator was designed to provide the synthesizer performance needed for narrowband measurements.[1]

To speed development, the HP 3588A incorporates several assemblies that were previously used by a low-frequency dynamic signal analyzer. Among these assemblies are the main processor board (including the disk drive), an auxiliary memory board, the front panel, the chassis, the power supply, and the display. It was quite a challenge adapting many of these parts to meet the needs of a high-frequency analyzer. An instrument-specific card nest was added to house the HP 3588A circuitry and provide shielding. Many chassis improvements were required to allow the HP 3588A to meet regulatory RFI requirements and to provide adequate internal electromagnetic compatibility to meet performance requirements. It was also necessary to synchronize the processor clocks (which indirectly control the display clocks) and the power supply switching frequency to the HP 3588A reference to prevent interference in the IF sections.

Receiver Section

Fig. 2 is a hardware block diagram of the HP 3588A spectrum analyzer.

The receiver section applies gain, filtering, and several frequency conversions to condition the input signal. The receiver contains circuitry to detect the incoming signal level and can choose the optimum input attenuator setting automatically. The HP 3588A also provides a low-distortion mode of operation in which an additional 10 dB of attenuation is applied to reduce any distortion products contributed by the instrument. Scaling, reference level tracking, and calibration are unchanged by this mode of operation. At the end of the receiver chain are the analog-to-digital converter (ADC), resolution bandwidth filter, detector, and video filter. All of these filtering and detection operations are implemented digitally in the HP 3588A and will be discussed in more detail later.

Since the HP 3588A is designed to be used at both RF and baseband, the input impedance is selectable—50 ohms or one megohm. Power is provided for a high-impedance

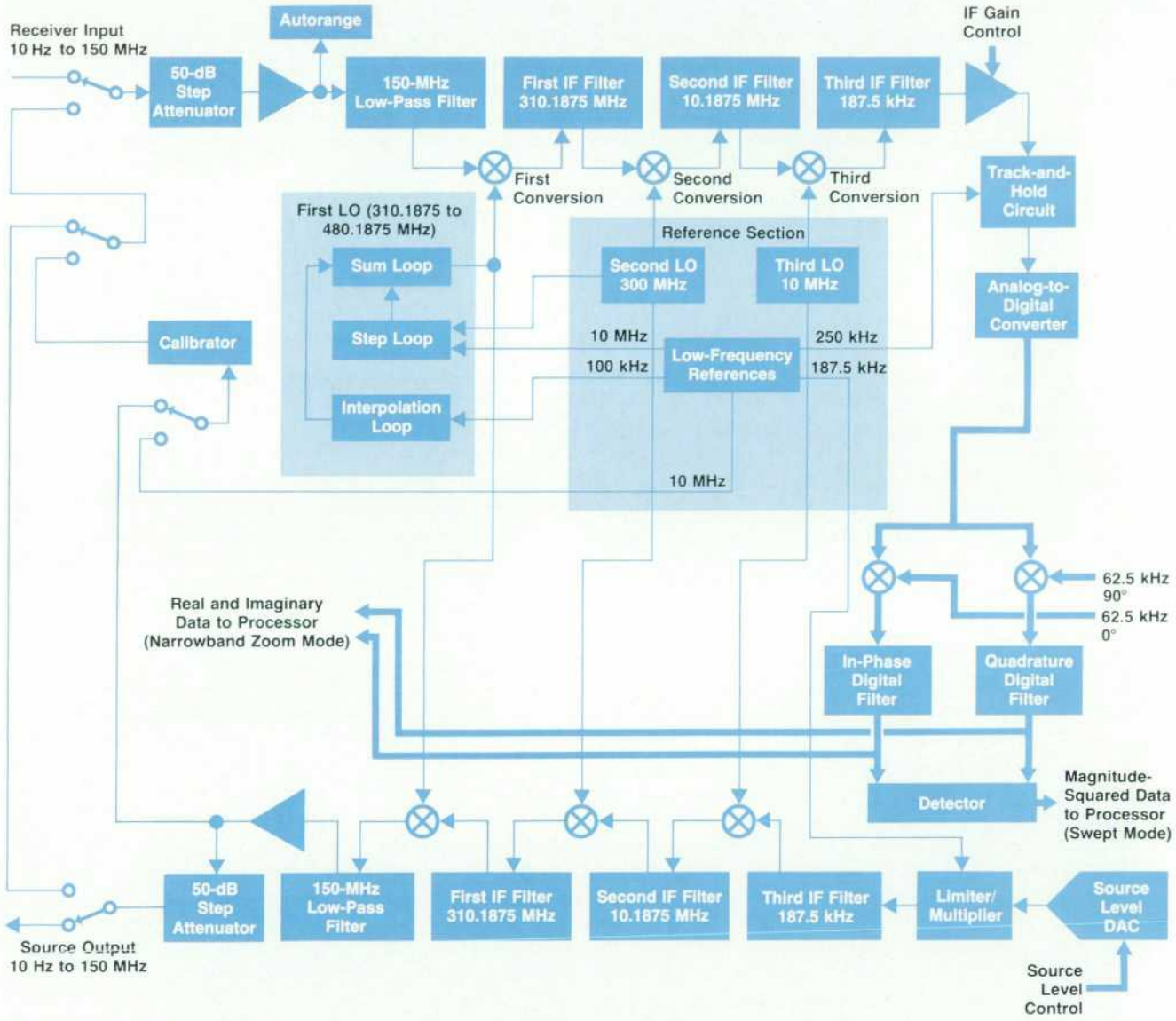


Fig. 2. *HP 3588A spectrum analyzer hardware block diagram.*

active probe. In the 50-ohm configuration, the signal level is monitored by an overload detector. In the event of an overload, the detector sets the input to the one-megohm impedance setting. This prevents any damage to the 50-ohm measurement path. In this situation a message is placed on the instrument's display informing the operator of the overload condition and telling how to reconnect the input. In the 50-ohm measurement path, a step attenuator provides 50 dB of attenuation in 10-dB steps so the signal level can be set to obtain the best dynamic range. The minimum and maximum input range settings are −20 dBm and +20 dBm. The HP 3588A also has the ability to pick the best input range setting automatically. A high-input-impedance buffer amplifier in the one-megohm measurement path conditions the signal and provides a 50-ohm output impedance to drive subsequent circuitry.

Following the step attenuator is an amplifier stage with a 50-ohm input impedance. This input amplifier is implemented largely with surface mount components and is shielded to prevent interference from externally generated signals. The amplifier provides 5 dB of gain, isolates the input from the rest of the instrument, and to a large extent determines many of the instrument specifications (such as distortion and noise). The isolation prevents the input return loss of the instrument from being affected by the input low-pass filter or the first mixer. It also prevents the first local oscillator signal from appearing at the instrument input. The amplifier also provides a separate signal to drive the autorange circuitry.

The autorange detection circuit is composed of a wide-band amplifier, a peak detector, and a main processor interface. The amplifier must respond to signals from 10 Hz to greater than 150 MHz if the optimum input range is to be chosen. The peak detector is designed for minimum temperature drift. The detected signal undergoes amplification and drives two comparators, which can interrupt the main processor if an out-of-range signal is detected. Since the signal levels are 10 dB lower when the instrument is con-

(continued on page 48)

Spectrum Analyzer Self-Calibration

The HP 3588A contains extensive self-calibration capability that allows it to achieve very good amplitude accuracy and LO feedthrough specifications compared to those previously achievable. On power-up and periodically thereafter, the HP 3588A performs self-calibrations of the frequency flatness for all input ranges, the IF level and shape, LO feedthrough, and the source amplitude.

Internal Hardware Calibrator

The internal hardware calibrator provides a precision amplitude reference level of −20 dBm ±0.02 dB from 200 kHz to 150 MHz. This signal can be switched internally to the receiver input for automatic range flatness and IF level calibration. Fig. 1 shows the calibrator block diagram.

The limiter conditions the input signal and desensitizes the output level to input level, allowing up to 20 dB of variation at the input with negligible output variation. The dc servo around the differential switcher provides precise control of the sum of the collector currents, which indirectly controls the average (dc value) of the individual collector currents. If the input to the switcher is reasonably symmetrical (i.e., low even-order harmonic content), then the fundamental of the switcher is exactly related to the dc value by

$$\text{fundamental rms amplitude} = (\text{dc amplitude})\sqrt{8}/\pi.$$

Thus the servo loop can, in theory, exactly control the level of the fundamental of the output. Practically, second-order effects such as finite switching times and base-to-collector feedforward exist. However these effects can be well-controlled with a simple frequency compensation network. The impact of employing this calibrator in the HP 3588A is reflected in the typical absolute amplitude accuracy specification of 0.2 dB from 30 kHz to 150 MHz.

First LO Feedthrough Nulling

It is important to minimize LO feedthrough to minimize low-frequency residuals and to reduce the problem of multiple tones in the IF at low frequencies. The HP 3588A incorporates a circuit to reduce the feedthrough level to at least 20 dB below range under all conditions.

LO feedthrough nulling is accomplished by a circuit which feeds a small amount of the main (swept) LO signal around the first mixer. The in-phase and quadrature components of this signal are adjusted by the instrument's main processor while monitoring the response when tuned to zero Hz. A very simple gradient search algorithm is used to find the minimum feedthrough level. The tuning is done by means of two "electronic potentiometers" which adjust the resistive and reactive impedance components of the feedforward circuit. The resistive component is adjusted by varying the current through a pin diode, while the reactive component is adjusted by a half-bridge composed of two VVC (voltage variable capacitor) diodes (see Fig. 2).

IF Filter Shape Calibration

The overall cascaded IF shape must be measured so that the contribution of the IF response away from its center frequency does not degrade the performance for narrowband zoom measurements. This is accomplished by sweeping past an internal fixed-frequency signal so that the IF frequency response is traced out in reverse. This response is then reversed and interpolated as needed to correct any subsequent narrowband zoom measurements.

Source Calibration

The HP 3588A source output amplitude is calibrated using the previously calibrated receiver. The output amplitude is measured during self-calibration at two amplitude settings using a direct internal path between the source output and receiver input. From

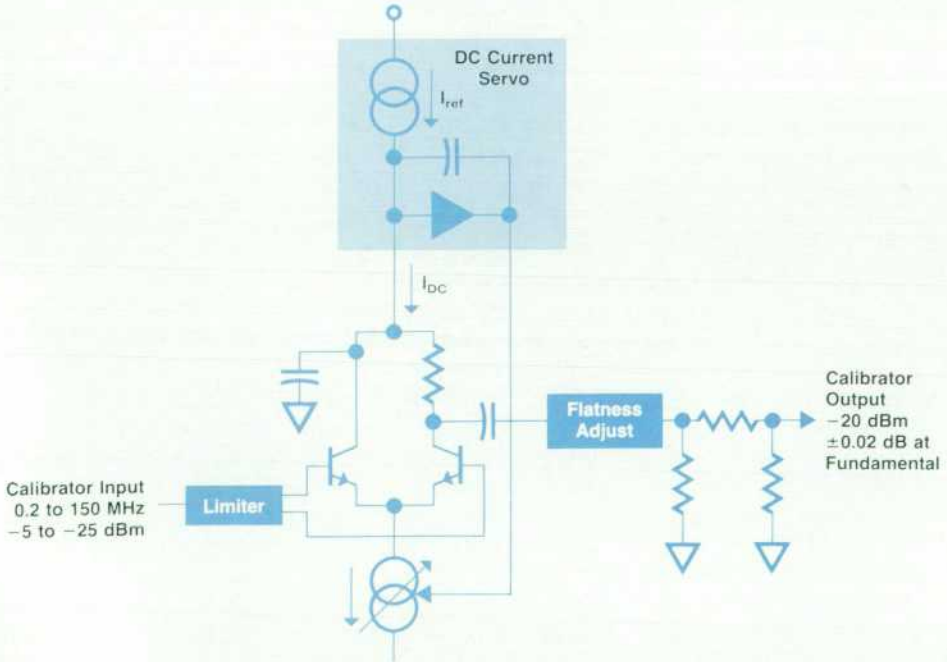


Fig. 1. *HP 3588A Calibrator.*

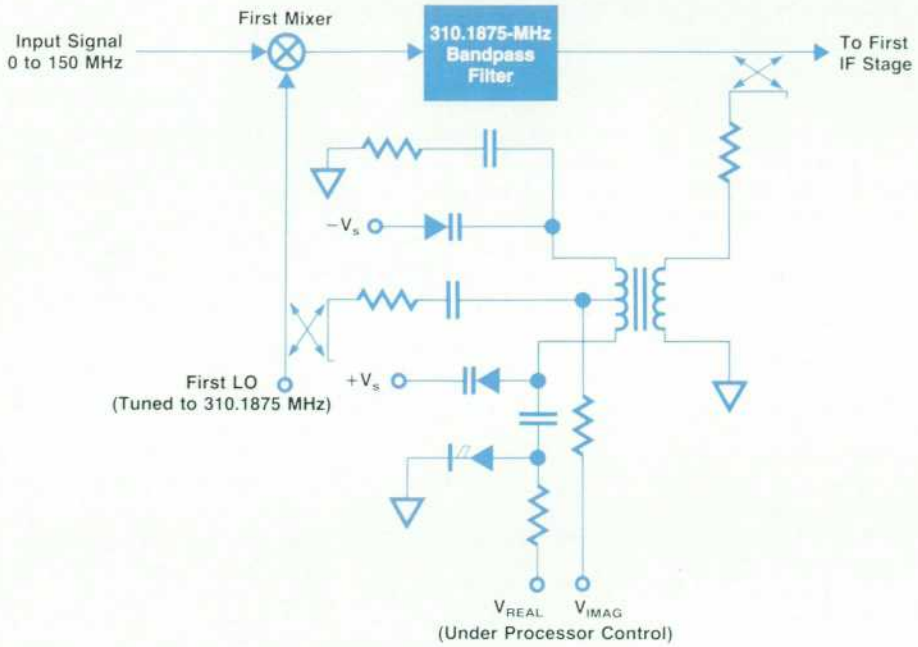


Fig. 2. *Local oscillator feedthrough nulling circuit.*

these measurements, constants are calculated that correct for gain errors in the source IF and amplitude control circuitry and for offset errors in the amplitude control circuitry. A manual amplitude adjustment is not required.

Timothy L. Hillstrom
Joseph F. Tarantino
Development Engineers
Lake Stevens Instrument Division

figured in its low-distortion mode, the lower autorange threshold must also be reduced by 10 dB in this mode. Detection of an underrange condition at this level is difficult, so the autorange algorithm is defeated in this case. However, a single autorange can still be performed, allowing a one-time optimization of the signal level.

The main signal path includes a 150-MHz low-pass filter between the input amplifier and the first conversion to prevent out-of-band signals from appearing at the mixer input. This filter is a ninth-order Chebyshev design with an integral amplitude equalizer. The filter is implemented with printed circuit inductors and surface mount capacitors and consistently demonstrates passband frequency response flatness of 0.4 dB. Special care was taken to shield this filter so that section-to-section crosstalk does not compromise stopband performance. Because this filter is also used in the source section of the instrument, development time was reduced. We were also able to reduce manufacturing cost since all of the parts are placed by machine, and we were able to reduce the time for final test since no adjustments are required.

The first conversion follows the low-pass filter and translates the input signal to a fixed first intermediate frequency of 310.1875 MHz. The first local oscillator is synthesized and covers a frequency range from 310.1875 MHz to 460.1875 MHz. The first LO will be discussed in detail later. As in any receiver that must accept a broadband input, spurious products, harmonic and intermodulation distortion products, and frequency flatness all require careful attention. It is primarily the first mixer that sets the harmonic distortion and residual spurious performance for the instrument. Since the design criteria to satisfy these goals were in conflict to some extent, achieving the desired first mixer performance represented a significant design effort. To achieve good distortion performance it is necessary to operate the mixer at high LO drive levels while maintaining symmetry of the drive waveform. However, any mixer has finite isolation between its ports, so higher LO levels result in large amounts of LO signal at the output of the mixer. If these signals are not prevented from reaching subsequent conversions they can produce residual (false) responses. Careful filtering and shielding were required to minimize these residual responses. In addition, a small amount of the first LO signal is fed around the first mixer to cancel the mixer's intrinsic feedthrough (see "Spectrum Analyzer Self-Calibration," page 47).

The 310.1875-MHz first IF signal is filtered by a four-section coupled helical-resonator bandpass filter having a bandwidth of approximately 3 MHz. Performance and ease of manufacture were important in the development of this filter. The helix assemblies within the filter are wound on supporting cores molded of a very low-loss fluoroplastic to minimize the reduction in resonator Q caused by dissipation loss. The resonators are aperture-coupled and the input and output connections are tapped directly onto the helix assemblies. Tuning screws with self-locking pellets make tuning of the filter easier than if lock nuts were used. The helical-resonator filter mounts directly into a cutout at the edge of a printed circuit board and the whole assembly slides into a card slot as a unit.

The second conversion translates the signal from 310.1875 MHz to 10.1875 MHz. The local oscillator for this conversion (the second LO), is a phase-locked 300-MHz signal also used as an offset in the main (swept) LO. A

seven-section LC filter and several gain stages make up the second IF section. To achieve the required instrument amplitude accuracy, the behavior of all of the IF amplifiers over temperature needed careful attention. Since operating frequencies and amplitudes are different for each IF section, various approaches were involved, but each amplifier stage received individual attention.

The third conversion is accomplished with an active mixer, providing gain as well as frequency translation. The third LO is a 10-MHz signal phase-locked to the instrument's common reference. This conversion translates the signal to 187.5 kHz. Following the conversion, the signal is further amplified and filtered, yielding a final IF bandwidth of approximately 40 kHz. An attenuator (settable in 1-dB steps under main processor control) is set so the signal is applied to the analog-to-digital converter at the optimum level. Conflicting performance goals made the design of the third IF challenging. To prevent degradation of swept measurement data, the transient response of the filter must be well-behaved. However, for good narrowband zoom performance, the frequency response of the filter should be reasonably flat over the entire 40-kHz IF bandwidth, and a filter designed for flat amplitude response often has poor transient response. The filter circuit was simulated and refined until good transient response at high sweep rates was achieved, and the flatness variation was kept to a level that can easily be calibrated out.

The analog-to-digital converter section consists of a track-and-hold circuit operating at 250 kHz followed by a two-pass converter. This entire section was leveraged from a previous product.[2] The final resolution bandwidth filtering, detection, and video filtering are all accomplished digitally in the HP 3588A by two custom gate arrays.

Digital IF Section

One of the goals for the HP 3588A project was to reuse existing modules wherever practical to minimize development costs. Hewlett-Packard dynamic signal analyzers use a chip set of gate arrays that perform many signal processing steps: triggering, optional quadrature detection from an arbitrary center frequency, programmable information bandwidth reduction by decimation filtering, and accumulation of complete sample records for later block processing. The HP 3588A digital IF section needed to be compatible with the same analog-to-digital converter as this chip set and at similar sample rates. It also needed to provide programmable information bandwidth reduction and be fairly inexpensive to build. Modification of the dynamic signal analyzer signal processing chip set seemed to be a good approach for the HP 3588A digital IF section, since a rapid development cycle was desired.

This approach to development paid off quite well. The digital IF section of the HP 3588A consists of two gate arrays (the filter and the detector), and three RAM chips. The dynamic signal analyzer chip set consists of five gate arrays and five RAM chips. The decrease in parts count was achieved by removing unnecessary functionality and by integrating more functions.

Many modifications were made because of the different needs of a swept, high-frequency analyzer and the block-record-oriented dynamic signal analyzer. The most important difference is that the swept analyzer applies signals with constantly changing frequency to the resolution filter. The resolution filter must react to these transient signals properly. The usual implementation for resolution filters is an n-pole synchronously tuned filter, which is often modeled as a Gaussian frequency response. The main requirements are a smooth step response with no ringing or overshoot, and good symmetry between the rising and falling edges of the response to a swept transient input.

The filter in the dynamic signal analyzer chip set implements a recursively applied decimation algorithm to reduce the information bandwidth to the user's desired span. Each application of the algorithm acts as a filter that halves the signal bandwidth and a resampler that then halves the sample rate. The decimation filter removes any out-of-band signals that would otherwise alias into the final information bandwidth because of undersampling. The frequency response of this filter has steep skirts and a flat passband, which is well-suited to sample rate decimation. Unfortunately, it has very poor transient response. The filter gate array for the HP 3588A uses the decimation architecture of the previous gate array, but internally cascades the output of the last decimation with a filter of appropriate transfer function and transient response for a swept analyzer.

This new filter is implemented as a finite impulse response (FIR) filter. It has a frequency response that is roughly Gaussian. Not only does it react much like traditional filters used in swept analyzers, but it also provides some additional benefits. It is digital and therefore has no analog adjustments. Environmental changes do not affect its response. It also has a much smaller shape factor than IF filters in most traditional swept analyzers. It has a 60-dB shape factor of less than 4:1. This means that the frequency response curve is less than four times as wide at 60-dB attenuation as it is at 3-dB attenuation. Traditional resolution filters typically specify a 60-dB shape factor ranging from 11:1 to 15:1. The 80-dB shape factor in the HP 3588A is less than 5:1! This narrower shape factor means that larger than normal resolution bandwidths can be used while resolving small signals that are very near large ones, thus improving measurement speed. Another characteristic of this FIR filter is its very predictable reaction to signals swept at faster than normal rates. This makes it possible to apply oversweep corrections, thereby providing four times faster sweep rates than traditional swept analyzers for a given resolution bandwidth without loss of accuracy. The concept of oversweep corrections is discussed later in this article.

The digital input to the signal processing chip set from the analog-to-digital converter is applied to two mixers simultaneously. The local oscillators for the two mixers are both at the same frequency, but are 90 degrees apart in phase. This provides two data streams, one representing the real part of the IF signal and the other representing the imaginary part. This quadrature detection is nearly identical to that done in the HP 3577A Network Analyzer.[3]

Swept spectrum analyzers detect the scalar magnitude of the input signals as they are swept through the analyzer's IF. The detector gate array converts the real and imaginary components of the signal into the magnitude of the signal. This magnitude is passed to the main processor for sub-

sequent processing and display. The complex signal samples can also be passed into the main processor for use in a fast Fourier transform process whose output can then be converted into magnitude and phase.

The detector computes the magnitude by squaring the real and imaginary components and then adding them together. This results in the square magnitude of the filtered signal, which is proportional to the square of the input voltage or to the input power.

Since the resolution filter bandwidth may be much narrower than a bin (display point) on the instrument, a peak detection function is also required. This is a standard swept spectrum analyzer function. If the peak detector were not included and the response were just sampled, a response narrower in frequency than the equivalent frequency spacing of adjacent pixels on the display could be missed entirely. For this reason, all swept spectrum analyzers have a peak detector. The peak detector gets reset whenever its output is sent to the main processor and thus to the display. Each magnitude sample received by the peak detector is compared to the peak detector's current value. If the new magnitude is larger than the current value then the peak detector value is updated to the new value; otherwise, the current value is retained.

Swept analyzer users expect to see the screen updated as the instrument sweeps, so acquisition of a complete sample record is inappropriate. This means that no record needs to be acquired, and detected data needs to be immediately available to the main processor. The detected result is passed to the main processor by an on-chip DMA interface. This saves several ICs over the previous approach, which used a FIFO buffer to acquire a complete sample record.

Traditional swept spectrum analyzers must have a video filter. This filter is typically placed after a full-wave detector and removes the component at twice the intermediate frequency resulting from the full-wave detection. The detector in the HP 3588A can be called a power detector since it merely takes the sum of the squares of the real and imaginary components. This detection technique has no resultant component at twice the intermediate frequency, so no video filter is needed for most measurements. However, a video filter is provided for occasions where noise smoothing is desired.

Filtering the video in the power domain has an advantage over the traditional approach. The traditional analog spectrum analyzer IF path has a logarithmic amplifier before the full-wave detector. This combination is calibrated for sinusoidal signals. However, those detectors respond differently to modulated signals and to nonsinusoidal signals such as noise. For instance, traditional detectors underrepresent the thermal noise power by 2.5 dB. The video filter in the HP 3588A does mean square filtering on the detected signal. It responds to the average power being measured and requires no corrections for different kinds of signals.

The detector gate array converts its input to a floating-point representation to avoid losing precision. The square of the 24-bit input would have to be 48 bits long if kept in a fixed-point representation. With the input normalized to a floating-point representation, the square of the 24-bit mantissa can be represented quite adequately with a 24-bit result. The exponent of the square is simply double the exponent of the input. The floating-point representation chosen is the IEEE single-precision format so that the results from this gate array can be used directly by the instrument firmware.

Local Oscillator

The local oscillator (LO) section of the HP 3588A provides a fully synthesized swept signal from 310.1875 MHz to 460.1875 MHz. It can be configured in two ways, depending on whether faster sweep speed or better phase noise and spurious performance is desired. In the single-loop mode the LO is configured so that it can sweep the entire range of frequencies very quickly. In the multiple-loop mode, a step loop and an interpolation loop are summed in a third loop, the sum loop, to produce the desired frequency (see Fig. 3). The choice of configuration is based on the instrument setup. If the measurement requires a large frequency span, single-loop mode is used. Setups involving narrower spans or resolution bandwidths use the multiple-loop mode. All narrowband zoom measurements use the multiple-loop mode. Six printed circuit assemblies make up the LO section, and many, but not all, are used in both modes of operation. The oscillators used for the step and sum loops are identical, and the resultant similarity of their tuning characteristics aids the overall LO performance. The oscillators are both voltage-tuned and are a negative-resistance design, with the base inductance realized by a trace on the printed circuit board. A fractional-N loop is used in both modes of operation to obtain fully synthesized sweeps and superior frequency resolution (better than 0.01 Hz).

In single-loop operation the sum loop oscillator output is divided by 10 to yield a frequency in the range of 31.01875 MHz to 46.01875 MHz. This signal is the input to the fractional-N divider circuit. The output of the divider is phase-detected using a 100-kHz signal from the reference section. The loop integrator sums the phase detector output current with the API (analog phase interpolation) correction currents, yielding a tuning voltage which is sampled and applied to the sum loop oscillator.

In multiple-loop operation the step loop output and interpolation loop output are summed to produce the main LO output. The step loop provides frequencies from 306 MHz to 450 MHz in 2-MHz or 5-MHz steps. The interpolation loop provides continuous coverage of the 2-MHz or 5-MHz bands between the step loop frequencies. The reference frequency for the step loop is derived from a 10-MHz signal supplied by the reference section. To simplify the design of the step loop divider and allow for some phase noise cancellation, a 300-MHz offset frequency is applied in the step loop (see the discussion of the HP 3588A reference section below). To maintain consistent loop bandwidth (and thus phase noise performance) the loop gain of the step loop is also adjusted as the divider value is changed.

The interpolation loop provides fine-resolution frequency coverage in multiple-loop mode. It uses the fractional-N divider, phase detector, and loop integrator/filter described earlier, but tunes a different VCO. The interpolation VCO is a single-transistor design tuned with VVC (voltage vari-

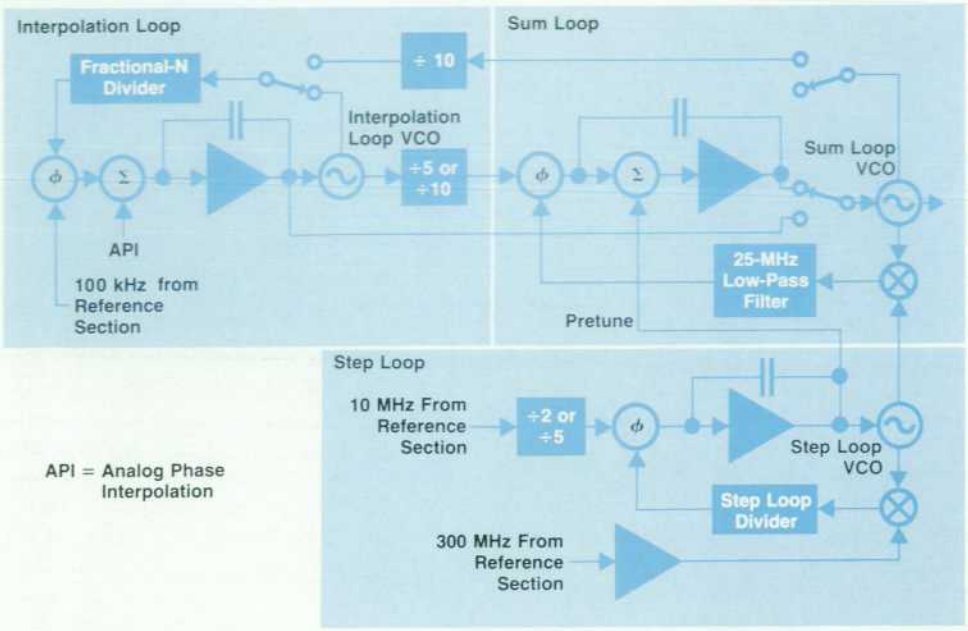


Fig. 3. *HP 3588A local oscillator (multiple-loop configuration).*

able capacitor) diodes. This design was leveraged from a previous product.[1] To fill in the steps for the two different step sizes used by the step loop, the interpolation loop VCO is followed by a circuit that divides by either 5 or 10.

The sum loop combines the step loop and interpolation loop frequencies by mixing the step loop frequency with the sum loop frequency and applying the resulting difference signal to the sum loop phase detector. The other input of the phase detector is the interpolation loop signal. The resulting sum loop frequency is $f_{sum} = f_{step} + f_{int}$. To reduce the capture time of the sum loop, a pretune voltage derived from the step loop's tuning voltage is summed into the sum loop integrator/filter. This pretune voltage guarantees that the sum loop's tuning voltage will produce an oscillation frequency near the step loop frequency. This in turn guarantees that the resultant difference frequency is within the range of the interpolation loop, so this loop will reliably achieve phase lock.

The phase noise of the HP 3588A LO section in multiple-loop mode is typically −115 dBc/Hz for offset frequencies

Adaptive Data Acquisition

The HP 3588A spectrum analyzer uses an adaptive data acquisition scheme to optimize system performance and responsiveness. At the beginning of a sweep, a DMA controller chip is set up to transfer 401 data points. As each new data point from the input channel becomes available, a custom gate array chip initiates the DMA transfer of the data point into main memory. The measurement sequencer process periodically checks the contents of the DMA memory buffer as it fills with new data and sends all newly arrived data as a block to the math coprocessor and the display process.

System data processing and display time can be modeled roughly as follows:

postprocessing time = math and display time
+ number of blocks × overhead per block

The math and display time is the total time required to process and display all 401 data points. This time is essentially fixed. However, transferring data to the math coprocessor and to the display controller chip also carries with it a fixed overhead per block, which is multiplied by the number of data blocks sent to postprocessing.

Clearly, the lowest overall processing time is achieved by sending all 401 data points at one time to math and display. On very fast sweeps, this is precisely what happens in the HP 3588A. When the measurement sequencer checks the DMA buffer, all of the sweep data has already arrived, so one 401-point block is sent on for postprocessing and display. Thus, the analyzer operates in its most efficient processing mode on the very fastest sweeps—when efficiency is most important.

When the sweep is very slow, however, the total measurement time is dominated not by the postprocessing time, but by the sweep time. The adaptive data acquisition scheme serves this case equally well. Here, the data is only trickling into the DMA buffer. The measurement sequencer may check the buffer many times before finding any new points to process. Still, when new points do arrive, they are processed and displayed. In this way, the user is provided with a "real-time" update of the display trace.

At moderate sweep rates, there may be many data blocks of ten or more points each. The adaptive data acquisition scheme again tends to optimize overall system performance. In this case, if the postprocessing is slowed because of competing system activity (like the front-panel keyboard), the next data block will be proportionally larger. This will tend to drive down the number of blocks per trace and thereby increase the postprocessing efficiency.

James H. Cauthorn
Development Engineer
Lake Stevens Instrument Division

greater than 100 Hz, and spurious sidebands are typically better than −80 dBc.

Source Section

The HP 3588A includes a source whose output frequency tracks the tuned receiver frequency and can be used as a stimulus for scalar network measurements. The source output frequency covers the full frequency range from 10 Hz to 150 MHz at any user-selected output amplitude between 10 dBm and −59.9 dBm with 0.1-dB resolution. It is implemented on two printed circuit assemblies. The source block diagram structure mirrors that of the HP 3588A receiver (see Fig. 2). Development time was saved with this approach because several functional blocks were developed to be used in both the source and the receiver. These blocks include the 150-MHz low-pass filter, the 310.1875-MHz helical-resonator IF filter, and the step attenuator. Because the source and receiver share common intermediate frequencies, care had to be taken that the source IF signal did not get into the receiver IF path and degrade performance.

A 187.5-kHz signal from the HP 3588A reference section is amplitude-limited and applied to the carrier input of a Gilbert cell multiplier. The second input to the multiplier is a dc control signal generated by a 12-bit digital-to-analog converter (DAC) and its associated transimpedance amplifier. The digital inputs to the DAC come from the main processor. The output of the multiplier is a square wave with an output amplitude controlled by the digital inputs of the DAC. This arrangement provides fine amplitude control steps that fill in between relatively coarse output attenuator steps. The user can access 19.9 dB of amplitude control range, and an additional 6 dB is available for source calibration, for a total of 25.9 dB of fine amplitude control range. The DAC provides adequate step accuracy over the 25.9-dB range. Harmonics of the multiplier output square wave fundamental frequency are removed with a 190-kHz low-pass filter. The 187.5-kHz signal is mixed with a 10-MHz signal from the reference section in an active mixer to produce a 10.1875-MHz IF signal. A crystal bandpass filter removes all undesired signals. The 10.1875-MHz signal is then mixed with a 300-MHz signal from the reference section to produce the 310.1875-MHz IF signal.

The 310.1875-MHz IF signal is filtered by a four-section, coupled helical-resonator filter identical to the one used in the receiver's first IF section (see description above). The 310.1875-MHz IF signal is then mixed with the local oscillator signal, which ranges from 310.1875 MHz to 460.1875 MHz, to produce the 10-Hz-to-150-MHz output signal. Both the 300-MHz mixer and the output mixer are standard diode ring mixers with an LO drive level of 7 dBm. A ninth-order 150-MHz low-pass filter removes all unwanted signals, such as mixer products, before the signal reaches the output amplifier.

The output amplifier design was leveraged from a previous product and consists of three sections.[3] All three sections are complementary, dc-coupled amplifiers with local bias stabilization circuitry. The first two are identical and provide 20 dB of signal gain each. Both series-shunt feedback and shunt-series feedback are used in these two amplifiers to provide gain accuracy, good flatness, and controlled impedance over the full frequency range. The final section uses series-shunt feedback only and provides 15 dB of gain at an output level as high as 11 dBm. A back-match resistor in series with the 15-dB amplifier output provides a 50Ω output impedance. A servo circuit in the output amplifier samples the output voltage of the 15-dB amplifier and injects a correction signal into the first 20-dB amplifier to maintain the output at zero volts dc. Power to all three amplifier sections is supplied through shunt Zener diode regulators to prevent the output amplifier from degrading receiver or LO performance by corrupting the power supply at low frequencies. The 20-dB amplifiers also have private regulators. Following the output amplifier is a 50-dB step attenuator consisting of 10-dB steps. The step attenuator uses precision surface mount resistors and RF relays that maintain a coaxial structure internally. Microstrip techniques are used to ensure good return loss over the full frequency range (the receiver step attenuator uses a very similar design). The overall flatness of the HP 3588A source is typically ±0.5 dB.

If an overload condition exists on the output connector that could damage the output amplifier and/or the step attenuator, an overload detector circuit disconnects the output connector from the source output. This source tripped condition is latched and can be reset by the main processor. The sense point for the overload detector is on the output connector side of the protection relay, and a reset will only be allowed if the overload condition no longer exists.

In addition to the main output, the HP 3588A source provides a signal path directly to the receiver for instrument self-test and source calibration. A portion of the source output signal is taken from the main output path ahead of the step attenuator to drive the swept calibrator. The HP 3588A source can be turned off when not in use. Turning the source off disables the 187.5-kHz limiter (which greatly reduces source IF levels) and disconnects the step attenuator from the output connector. This minimizes any signal interference that could corrupt a spectrum measurement. A 50-ohm load is connected to the output when the source is off or when the direct source-to-receiver path is being used (such as during calibration). This provides the proper termination impedance for any device the user may have connected.

Very few adjustments are required for the source circuits. This reduces test time and the time required for periodic instrument calibration. The only required adjustments are on the helical-resonator bandpass filter. The four helical-resonator cavities can be tuned using Dishal's method,[4] which is simple, accurate, and noniterative. Tuning is facilitated by jumpers on the printed circuit board that allow reconfiguration of the IF signal path so that a signal from a sense port on the filter input cavity is routed through the normal IF path. No external test equipment is required for the tuning procedure on the source or receiver.

Several features are included in the source circuitry to aid in fault isolation. A detector is available to sense 187.5-kHz signals and indicate their presence or absence to the main processor. An analog multiplexer allows main processor switching of the detector input to either the source 187.5-kHz input or the output of the Gilbert cell multiplier. The threshold of the detector is such that a functional check

Help System with Hypertext

HP's Lake Stevens Instrument Division (LSID) has included online help systems in its new analyzers for the past six years. All of these systems—including the HP 3588A's—are designed primarily as online key references: when the user presses a hardkey or a softkey, the analyzer displays a description of the key's function. However, the HP 3588A's help system also includes two new features that meet the needs of a wider variety of users:

- An index of help topics
- Hypertext links between related topics.

How It Works

The user enables the HP 3588A's online help system by pressing the **Help** hardkey and exits by pressing **0**. While help is enabled, most of the analyzer's screen is dedicated to displaying help text; the rest of the screen continues to display softkey labels.

The analyzer provides some functions that allow the user to move around in the help system. Most of these functions are assigned to hardkeys on the numeric keypad. A legend at the bottom of the screen shows these assignments so the user doesn't have to remember them. The page is changed by pressing 9 or 6. The index (see Fig. 1) is displayed by pressing 1. Hypertext links are navigated by turning the knob and pressing 4 or 7.

Hypertext Navigation

On a given screen full of help text, there may be several special words (or phrases) that are linked to related topics. The linked words are all underlined to distinguish them from the surrounding text (see Fig. 2). To select and display the topic that's linked to one of these words, the user turns the knob until the desired word is highlighted and then presses 4. After reading the contents of the new topic, the user can return to the previous topic by pressing 7.

The method used to display a linked topic is also used to display topics listed in the index. After pressing 1 to display the index, the user can turn the knob to highlight one of the listed topics and press 4 to display the contents of that topic.

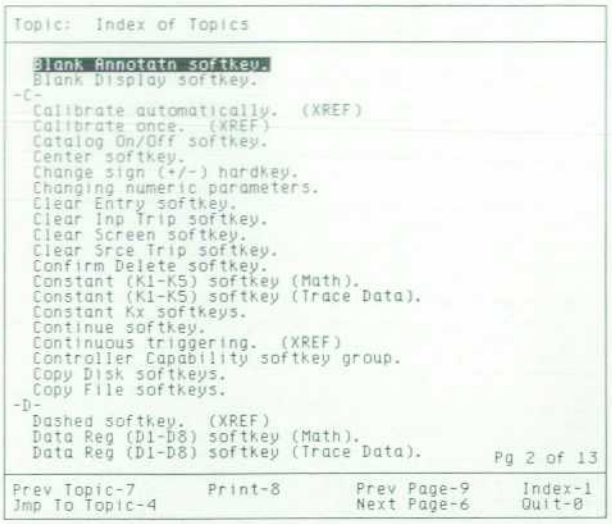


Fig. 1. *A portion of the HP 3588A spectrum analyzer's help index.*

Index Access

In previous LSID help systems, a user could only access a key's description by enabling help and then pressing the key. These designs assumed that the user had already navigated the menu structure and was pondering the use of a particular key. They worked well enough for experienced users and for new users who like to locate functions by poking around on front-panel keys. They worked less well for other users.

The index provides a more systematic way for new or infrequent users to locate information about an analyzer function—including the name of the key that controls the function and the location of that key in the menu structure.

For example, to find out how to turn off the display graticule, a user can enter the help system, display the index, and then look for an appropriate keyword, like "graticule." In this case,

of the amplitude control circuitry is made possible by using amplitude control settings above and below the detector threshold with the detector switched to the multiplier output. Additionally, the threshold of the source overload detector can be lowered by the main processor so that a 10-dBm signal will trip the overload detector. This is useful as an indication of source functionality and output amplifier bias faults.

Reference/Trigger Section

The reference/trigger section is partitioned among three printed circuit boards to minimize internal electromagnetic compatibility problems. The primary functions of this section are to provide the various UHF, VHF, and LF references used by other sections of the instrument, and to provide the instrument synchronization and trigger logic. Fig. 4 is a block diagram of the reference/trigger section.

The HP 3588A can free-run on its internal 80-MHz VCXO, lock to an optional 10-MHz oven reference, or lock to a user's reference at 10/N MHz, where N is an integer from 1 to 10. The 80-MHz VXCO is designed to have low temperature drift and good stability for users who do not require the high performance of the optional ovenized reference. The 80-MHz reference is used to provide quadrature 20-MHz clocks for the digital filters, using a Johnson counter on the same circuit board as the filters. 10-MHz references are derived from the 80-MHz VCXO and are used in various places throughout the instrument.

The 300-MHz reference is an LC oscillator that acts as the LO for the second conversion and provides an offset frequency in the LO step loop (see Fig. 3). With this architecture, the net phase noise caused by the 300-MHz reference is:

$$(\text{300-MHz phase noise}) \times (1 - H_{step}H_{sum}),$$

where H_{step} and H_{sum} are the closed-loop phase transfer functions of the step and sum loops, respectively (with the gain of the step loop normalized out).

This effectively high-pass filters the 300-MHz phase noise (in the offset frequency domain) and provides attenuation at low offset frequencies. Thus, the requirement for

```
Topic:  Swept Spectrum softkey.

Key Path:  (Meas Type)

   Press [SWEPT SPECTRUM] when you plan to use the
HP 3588A as a traditional swept-tuned spectrum analyzer.

   During swept spectrum measurements, the analyzer
gathers frequency-domain measurement data by mixing your
input signal with a sweeping local oscillator (LO) and
then detecting the difference frequency.  The analyzer's
display appears to sweep through the
band of frequencies you have specified.

   Swept spectrum measurements offer several measurement
options that are not available during narrow band zoom
measurements:

   * Wider spans (up to 150 MHz).
   * Manual sweeping.
   * Zero span analysis.
   * Noise level markers.

   However, narrow band zoom measurements provide much
shorter measurement times at spans below 40 kHz.

                                                  Pg 1 of 2

Prev Topic-7      Print-8      Prev Page-9      Index-1
Jmp To Topic-4                 Next Page-6      Quit-0
```

Fig. 2. *Words in help text that are linked to related topics are underlined.*

the user would find the entry "Graticule On/Off softkey." When the entry is selected, the analyzer displays text describing the key. The text also contains a key path, which indicates the location of the key in the menu structure.

Related Topics

In a complex product like a spectrum analyzer, one key's function is often closely related to another key's function. A user needs to understand such relationships to use the analyzer most effectively. Hypertext links make it easy to explore relationships by allowing a user to move directly between related topics.

For example, the help text that describes the **External Trigger** softkey points out that the analyzer must be armed before it can be triggered. In this text, the word "armed" is linked to the topic "Arm Auto/Man softkey," which describes arming. The user can move directly to the topic on arming by selecting the word "armed."

Non-Key Topics

Since the HP 3588A help system allows access to topics via the index and links, it is possible to include topics that are not bound to a particular hardkey or softkey. We have taken advantage of this new freedom to create independent topics that define commonly used terms. Any time we use one of these terms in the help text, we link the term to its definition. Defining terms in independent topics has two advantages:

- Users who don't know what a term means—typically novice users—can simply select the term and display the topic that defines it. This allows all readers to have a common understanding of the terms used in help topics.
- Users who already know what a term means—typically expert users—don't need to read its definition each time the term is used. This reduces the information clutter presented to these users, making it easier for them to find the information that they really need.

As a side benefit, this approach allows the help-text writer to define a term only once. This reduces both writing time and the storage space required in the analyzer for online help.

Mark M. Smith
Learning Products Developer
Lake Stevens Instrument Division

spectral purity of the 300-MHz reference at low offset frequencies is eased by this effect, allowing the use of an inexpensive LC oscillator rather than a more expensive low-noise oscillator employing a SAW (surface acoustic wave) resonator.

Low-frequency references are provided for the source (187.5 kHz), the ADC sample clock (250 kHz), and the interpolation loop reference (100 kHz). Synchronization is maintained between all these references and the Start_Meas signal so that data taking and the LO sweep are syn-

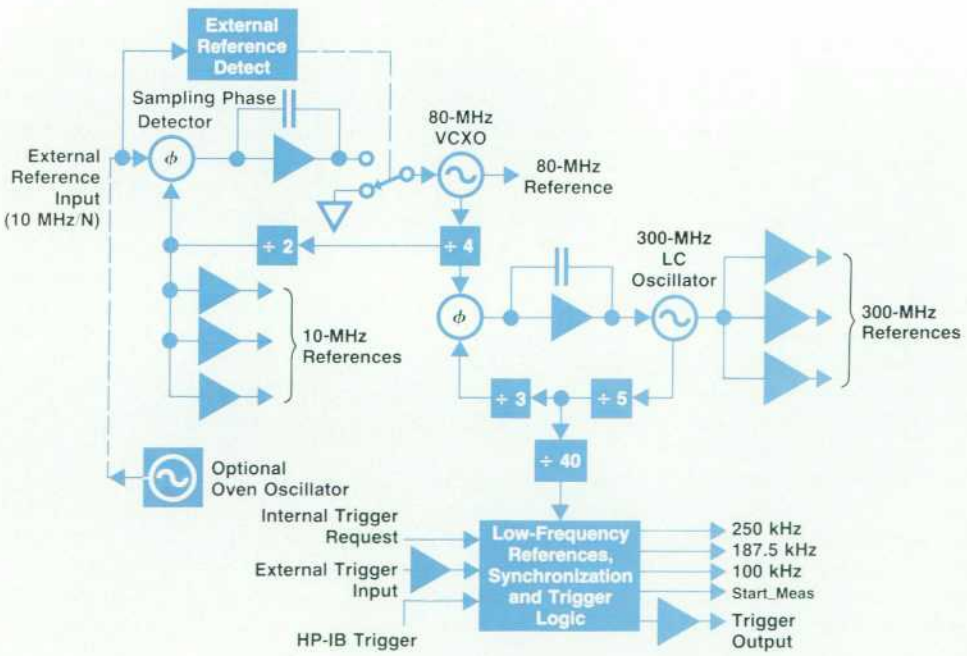


Fig. 4. *HP 3588A reference section.*

chronized and repeatable.

Further synchronization capability is provided so that several HP 3588A's can be configured to sweep synchronously, within a few microseconds uncertainty, using an IBASIC application program. This allows synchronized offset or harmonic sweeps, facilitating mixer or generalized distortion testing. To achieve this, the synchronizing circuit allows the low-frequency references and Start_Meas signal of all instruments to be synchronized through a **Trigger Out** signal provided on the rear panel.

Speed Advantages of FFT-based Spectrum Analyzers

For several decades, Hewlett-Packard has offered spectrum analyzers using FFT technology in the baseband frequency range up to 100 kHz. The FFT is used essentially as a bank of parallel bandpass filters. Since all the filters settle and measure simultaneously, the measurements can be much faster than with a swept filter of similar resolution. This is very useful for transient signals and time-varying signals, so this technique has been termed "dynamic signal analysis". Typical window functions designed for amplitude accuracy are three to four FFT frequency samples wide. A 512-point FFT thus provides over a hundred parallel filters at once and may provide measurements over 100 times faster than a single analog filter.

Swept analyzer measurement speed varies dramatically with different shape factors when the need to resolve close-in spurs is considered. For the purposes of this discussion, a close-in spur is a signal to be analyzed that is very small and very close to a much larger one. The intent, of course, is to resolve the spur as a spectral component, independent of the large signal. To resolve these spurs, a filter with a larger (worse) shape factor must be set to a much smaller nominal 3-dB resolution bandwidth than a filter with a better shape factor. Otherwise, the spur just disappears into the skirt of the large signal. In a swept spectrum analyzer, the sweep time variation must be inversely proportional to the square of the resolution bandwidth. If a filter with a poor shape factor has to have one-third the 3-dB bandwidth of a better filter to resolve the desired signal, the sweep time will be nine times longer.

The speed advantage of FFT spectrum analysis over swept filter analysis is even more apparent in the close-in spur case. The FFT windows typically used have shape factors of 2.6:1, which is much better than the typical swept resolution bandwidth filter specification, which ranges from 11:1 to 15:1. Thus, with both the basic FFT speed improvement and the speed improvement from the improved shape factor, an FFT analyzer can be many hundreds of times faster than a traditional wide-shape-factor analog swept analyzer for small, close-in signals.

FFT analyzers have been used in down-converted systems to gain the advantages of FFT measurements at higher frequencies. An example is the HP 3048A phase noise measurement system, which uses a dynamic signal analyzer to measure phase noise rapidly at offsets up to 100 kHz for carrier frequencies up to 18 GHz.

The narrowband zoom measurement mode of the HP 3588A uses FFT analysis to provide quick measurements for frequency spans up to 40 kHz. The center frequency can be anywhere in the instrument's frequency range. The instrument updates the 40-kHz span more than seven times per second, and provides frequency resolution better than 400 Hz. Many measurements that were not really possible before are now fairly easy.

For instance, it is possible to watch modulation sidebands move as the modulation frequency changes. During development of the HP 3588A, this capability was used to improve the 80-MHz crystal reference oscillator. One HP 3588A was vibrated on a shaker table that was being driven by a swept frequency ramp, and a second HP 3588A analyzed the output of the first. The real-time presentation of the vibration-induced oscillator sidebands allowed diagnosis of the chassis and printed circuit vibration modes and subsequent suppression of those modes. Previously, it was difficult to see the modulation sidebands move as the table stimulus changed.

Improved FFT Dynamic Range

Users of traditional swept analyzers often reduce the resolution bandwidth to lower the noise floor to analyze smaller signals. The noise sources in the input circuits of the instrument are typically the limiting factor in low-level signal detectability. If the resolution bandwidth can be lowered enough, thereby lowering the noise bandwidth, signals can be inspected that are over 100 dB below the input range of the instrument. The user trades off slower measurements for more dynamic range. A 32-bit FFT is used in the HP 3588A to allow this same trade-off. Reuse of the processor board from a previous design meant the reuse of the 16-bit math coprocessor (a Texas Instruments TMS320C25), and the reuse of much of its firmware. Firmware was available to do all the fundamental math operations. A well-characterized 16-bit FFT was also available. This certainly meant faster development, but the 16-bit FFT did not have enough dynamic range to reduce the noise floor adequately. Because of headroom and windowing, the 16-bit FFT was good to about 90 dB of dynamic range. A 32-bit word-length FFT was written specifically for the HP 3588A. This FFT provides spans of less than 100 Hz, at which the instrument's displayed noise floor is 120 dB or more below the input range. This allows the simultaneous analysis of a carrier at or below the range and a spur a few hertz from the carrier frequency at 120 dB below the range.

The computation of a 32-bit FFT is considerably slower than a 16-bit FFT. However, the dynamic range obtained with the longer-word-length FFT is deemed worth the resulting decrease in speed. The speed of FFT measurements with narrow spans is dominated by the time required to acquire the time record. Wide-span FFT measurements run at over seven updates per second.

Sweep Dynamics and Corrections

The resolution filters in the HP 3588A's final IF section can be swept at rates four times faster than traditional, synchronously tuned filters of the same resolution bandwidth. The HP 3588A automatically corrects amplitude and frequency errors resulting from these fast sweep rates. This technique allows the HP 3588A to make swept measurements much faster than previous swept spectrum analyzers.

The goal of the resolution filter is to select a narrow band of frequencies and reject all others. The energy in the

selected band is detected and passed to the main processor for subsequent display. The resolution filter is defined by its resolution bandwidth, which is its 3-dB bandwidth. When the analyzer is tuned to a signal, a heterodyned version of the signal is inside the resolution bandwidth and is detected. When the analyzer is tuned away from the signal, the heterodyned version falls outside the resolution bandwidth and should not be detected. However, the signals are actually moving in frequency as the local oscillator is swept. Therefore, the filter must respond quickly and accurately to transient signals. The filter should respond with an accuracy of 0.1 dB to all these swept signals, with no ringing or overshoot. The traditional approach to resolution filter implementation is to approximate a Gaussian frequency response using a synchronously tuned filter, which is a cascade of several identically tuned filter stages.

The result of sweeping traditional filters much faster than the traditional maximum rate (one half the square of the resolution bandwidth) is to distort the spectrum display in several ways (see Fig. 5). Two first-order distortions are expected since they appear when the theoretical Gaussian filter is analyzed. One first-order distortion is a monotonic decrease in response amplitude as the sweep rate is increased. The amplitude error rises dramatically and unacceptably for sweep rates faster than the traditional maximum sweep rate. However, the amplitude error is minimal for slower sweep rates. Another first-order distortion is a displayed frequency shift to the right. The perceived frequency shift is equal to the group delay of the IF filters times the sweep rate. Therefore, if the analyzer is swept four times as fast, the displayed signals shift four times as much.

Sweeping traditional filters too fast also has some second-order distortion effects (see Fig. 5). For example, the amplitude response becomes very asymmetrical about the peak of the response. Both edges may also exhibit notches and ringing rather than a smooth transition between the noise floor and the signal peak. These distortions are unacceptable because they may conceal spectral features of interest to the spectrum analyzer user, such as spurs or modulation on the large signal. The combination of all these spectral distortions has limited analyzers with traditional filters to the maximum sweep rate limit of one half the square of the resolution bandwidth.

Analysis of the response of an ideal Gaussian filter to swept-frequency input signals shows only the amplitude decrease and the frequency shift. Both of those distortions in the response are perfectly predictable. However, the analysis of the ideal Gaussian filter shows none of the second-order effects apparent in traditional implementations. Thus the accuracy of the traditional filter approximation is inadequate for faster than traditional sweep rates. However, there are advantages to sweeping faster than the traditional sweep rates.

It can be shown by analysis of the ideal Gaussian filter that for a fixed sweep rate, a range of resolution bandwidths narrower than that traditionally chosen for that sweep rate actually provides better frequency resolution. For a given sweep rate, there is a bandwidth that gives the finest resolution. This bandwidth is commonly known as the optimum resolution bandwidth.[5] For a fixed sweep rate, the signal-to-noise ratio is maximized when the resolution bandwidth is set to its optimum value. This signal-to-noise ratio is almost 2 dB better than at the traditional bandwidth for that sweep rate.[5]

The HP 3588A uses a digital, linear-phase, resolution bandwidth filter in its final IF section. Just as in traditional implementations, the amplitude response decreases and the detected trace shifts right as the sweep rate increases. However, the HP 3588A is able to correct these first-order distortions completely (see Fig. 6). The decrease in response amplitude is purely a function of the sweep rate and the filter bandwidth. The amplitude decrease is corrected by predicting the loss and applying a compensating gain. Unfortunately, the resolution bandwidth filters in the HP 3588A are not ideal Gaussian filters. In fact, their response is not close enough to allow use of theoretical Gaussian response calculations for correction values. Instead, the software holds a two-dimensional table of calibrated amplitude loss, indexed by sweep rate and resolution bandwidth. The expected decrease is interpolated from the table for each measurement setup and inserted into the composite correction trace for the measurement.

The right shift in the detected data is also predicted and corrected. Since both the group delay in the IF path and the sweep rate are known, the amount of frequency shift is easily calculated. This shift is corrected by shifting the detected data left by an equal amount.

Spectrum analyzers using traditional filters could conceivably correct the first-order distortions much as the HP 3588A does. Many users already do so to some extent. For instance, sophisticated users are aware of the frequency shift and mentally correct that themselves. There seems to be no way, however, to correct the asymmetry between the rising and falling edges and the ringing in the edges of the response. If those distortions are still present, there is nothing to be gained by correcting the first-order effects.

The resolution filters in the HP 3588A digital IF do not have the second-order distortions apparent in the tradi-

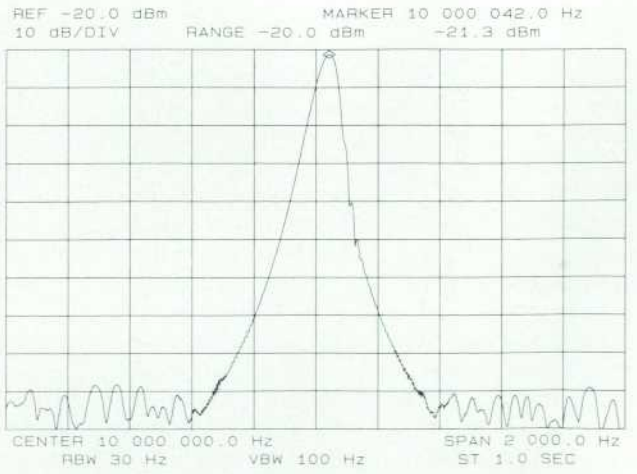


Fig. 5. *Results of sweeping a conventional resolution bandwidth filter four times faster than the traditional maximum rate. Note the asymmetry and ripples in the overswept response, especially in the falling edge. The input is a −20-dBm, 10-MHz signal.*

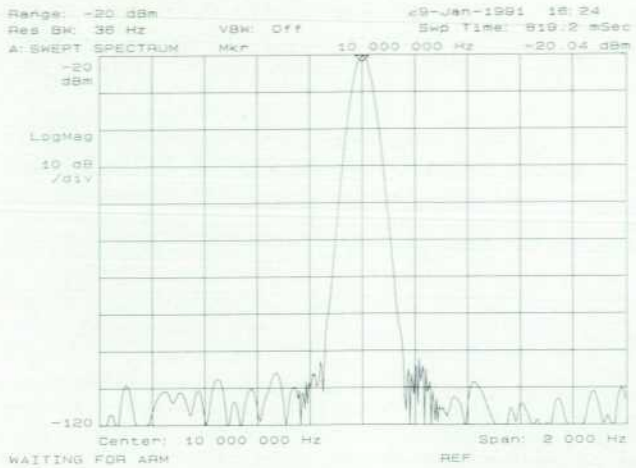


Fig. 6. *Corrected responses of the HP 3588A's digital resolution bandwidth filter swept four times faster than the traditional maximum rate. Note the symmetry of the rising and falling edges of the overswept response. The effects of oversweeping are predicted and corrected. The input is a −20-dbm, 10-MHz signal.*

tional implementations. Since they are FIR filters with perfectly linear phase, the falling edge of the response is just a mirror image of the rising edge of the response. Especially important is that the edges of the response are smooth, with no ringing (see Fig. 6).

Firmware Implementation

The firmware controlling the HP 3588A spectrum analyzer supports a rich set of features for the setup and control of spectrum analysis measurements. In addition, it offers a range of higher-level capabilities, including:

- Optional HP Instrument BASIC
- HP-IB controller capability
- Programmable limit testing
- Online help
- Programmable "user math" postprocessing operations
- HP Test and Measurement Systems Language (TMSL) compatibility*
- Internal file system for storage of traces, states, and programs
- Optional internal disk drive.

Some of these features have never before been available in a swept spectrum analyzer.

*The HP Test and Measurement Systems Language (TMSL) is the basis for the new Standard Commands for Programmable Instruments (SCPI) standard.

User Interface Compiler

The user interface compiler (uic) is a tool designed to increase the productivity of software engineers and to aid in the documentation of the user interface for the people who use, test, and implement it. On the HP3588A project, this tool helped to make the user interface more consistent. It also greatly reduced the time needed to make modifications and significantly reduced errors in documentation.

The uic software tool reads an input file containing a description of the user interface and generates several output files. The input file contains a sequence of keywords followed by relevant information in a fairly flexible format. The different keywords indicate definitions for softkeys, menus, and HP-IB commands and parameters. There are also fields for a key's functional description, the name of the engineer responsible for that description, the help text associated with the key, and the HP-IB command associated with the key.

The output files include the following information:

- C data structures that are downloaded into the analyzer
- The HP-IB language tree (hpibtree)
- The softkey menu tree (keytree)
- A list of all HP-IB commands for automated testing (hpibleaves)
- Help text that is downloaded into the analyzer
- Customer reference manuals for both the front panel and the HP-IB
- Internal technical references for both the front panel and the HP-IB.

The various descriptive paragraphs in the input file are targeted to a particular output file or files through the use of different keywords. It is also possible to include the help text descriptions in one or more of the generated documents.

The most notable output file contains the C data structures that represent the entire user interface. The contents of this file represent the menu structure, parameter information, configuration data, and HP-IB language for the analyzer. These data structures are used by a state machine called the front-panel kernel, which realizes the front-panel interface for the analyzer. An HP-IB parser implements the bus control for the analyzer, using the uic-generated data structure file. These files can be compiled and linked together with the rest of the analyzer firmware to create the analyzer executable code.

Several documentation files can be generated by the uic. Some of these files contain detailed descriptions of the commands; others contain summary information. The hpibtree and keytree files are examples of summary files. Hpibtree shows the entire HP-IB language understood by the analyzer in a form that portrays the hierarchical nature of the HP Test and Measurement Systems Language. Keytree shows all softkey menus and their relationships to each other as well as the equivalent HP-IB commands for each softkey.

Other documentation files contain more detailed information. The front-panel external reference specification (ERS) contains a description of each softkey and describes the parameters, the responsible engineer, and how the softkey interfaces to the rest of the analyzer. Traditionally, a similar document has been the only description of the user interface available until the technical writers finish the customer reference manuals. Another document, the HP-IB ERS, describes similar information for each of the HP-IB commands. These two files are considered internal information and are intended to be used by software and hardware engineers, test engineers, technical writers, and managers.

The uic also generates files that can be used as prototypes for front-panel and HP-IB customer reference manuals. These files contain refined descriptions of functionality and omit implementation details.

The hpibleaves uic output file has proved to be very valuable in testing the analyzer. This file includes only the HP-IB commands that have execution routines associated with them. Listed with

each command are the required and optional parameters, including the range of accepted values for each numeric parameter. Tests have been written that parse this file and send commands in either a random fashion or sequentially to verify that all of the analyzer's HP-IB commands function properly.

In addition to these output files, the uic provides an extensive set of diagnostic warnings and errors. These provide information about problems and potential problems with the user interface description.

The key to the user interface compiler's benefit is that it provides a single point of control for the user interface definition. The uic's input file contains all of the information required to generate the documents described above automatically. Thus, changes made in key labels, actions, or definitions during firmware development are immediately reflected in the front-panel and HP-IB user interfaces, in all versions of the documentation, and in the automated test documents. Having a single point of control, coupled with automatic generation of important files, means that the user interface and its documentation are more up-to-date and reliable throughout the product's development cycle. This translates into a considerable productivity gain.

As an example of this productivity gain, consider making a simple change to an analyzer's user interface definition during development, such as adding or deleting a softkey. Making such a change on our previous generation of analyzers would typically take two to three person-days to complete. The change process involved editing several files that variously contained information about the key's operation, its associated documentation, help text, and automated test files. Using the uic, a similar change takes only 15 to 20 minutes because all of the associated files are generated from the single uic input file. Given that our recent-generation analyzers have typically contained many hundreds of key and HP-IB mnemonic definitions, the potential for time savings is great.

Another problem with maintaining the user interfaces of previous analyzers throughout their development phase was that all changes required the attention of software engineers. This was because the majority of the information was buried in C source files. The uic allows a nonprogrammer to maintain the user interface. On one current project at the Lake Stevens Instrument Division (LSID), a technical writer and a software engineer share the responsibility of maintaining the uic's input file. Also, the uic makes it possible for marketing engineers or project managers—who are chiefly responsible for the analyzer's functional definition—to maintain the input file and hence the user interface.

Using the uic, an analyzer's entire user interface can be prototyped very quickly. The prototype of the user interface can be provided even before analyzer hardware is available. One LSID project has made an X Window System host version of the analyzer front panel, which allows a large portion of analyzer code to be developed on the host without having to use the target hardware. Because the uic assumes a generic interface to the analyzer state control code, it is not necessary for that code to be aware of the menu structure, except where there are variant menus that depend upon the current state of the analyzer.

Often the user interface data structures developed during the prototyping phase of a project have to be reworked and maintained constantly during the project's development cycle. With the uic, this effort is reduced to maintaining the uic's input file. All of the key user interface data structures are then automatically generated.

The user interface compiler was developed and evolved during the development of the HP 3588A. During that time, it matured into a tool whose output was relied upon by a diverse spectrum of project personnel. It saved hundreds of hours of development time on the HP 3588A project and helped to create a more reliable and consistent product. Today it is being reused on many projects at several Hewlett-Packard divisions.

Bryan P. Murray
Development Engineer
Lake Stevens Instrument Division

Implementing these features was not the only challenge to the design team. Another requirement was to reuse the main processor board, front panel, and display from an earlier low-frequency FFT-based analyzer. The processor board contains a dedicated TMS320C25 math coprocessor. This coprocessor, along with synthesized sweeping capability and a fully digital final IF stage, makes it possible for the HP 3588A to integrate traditional, broadband, high-frequency swept spectrum analysis with the excellent speed and resolution of FFT-based analysis. The math coprocessor also enabled us to include limit testing and extensive user math capability.

The reuse of the CPU board brought its own challenges, however. The board was basically designed to support the block-mode data processing that is characteristic of an FFT-based analyzer. The HP 3588A presents a serial data stream to the CPU board. It was therefore a challenge to use the available hardware to present sweeping data on the display.

In addition, the firmware system designed for the earlier, FFT-based analyzer could not meet the performance requirements of the HP 3588A. With the exception of some leveraged hardware drivers, the team essentially had to start the firmware development from scratch. As usual, this was all undertaken in the face of very rigid time-to-market constraints.

The code was written primarily in C on HP 9000 Series 300 workstations interconnected over a LAN. Each engineer had a prototype analyzer which was used for software development and testing. The target system (analyzer hardware) consisted of an MC68000 CPU, the TMS320C25 math coprocessor, and auxiliary processors used in controlling the front-panel keyboard, HP-IB, CRT display, DMA, internal disk drive, and interfaces to the various analyzer hardware boards. During development, the designers were able to download executable code from their workstations into the target system. Execution of the code in the analyzer was controllable through a source-level debugger running on the workstation. The final firmware system consists of approximately 107 thousand lines of source code, occupying approximately 1.2 megabytes of ROM.

One of the first key decisions was to use pSOS* as the kernel of our real-time, multitasking operating system. Most previous analyzer projects from our lab had invented their own operating system kernels. pSOS provides a small, reliable, and efficient real-time kernel. By picking an off-the-shelf product, we avoided most of the development, debugging, and quality assurance time we would have faced had we developed a new operating system kernel.

We also accelerated the project by adopting an existing SCPI-compatible HP-IB parser from Hewlett-Packard's

* pSOS is a trademark of Software Components Group, Inc.

Measurement Systems Operation. The HP 3588A has over 300 softkeys and recognizes over 200 HP-IB commands. To ease the coding burden of such a large system, one of the designers developed an automatic code generator that produced all of the necessary data structures and much of the documentation for the entire user interface.

High-level analyzer operation was broken out into five primary processes: user interface, measurement manager, measurement sequencer, display manager, and Instrument BASIC. Other processes were allocated for such purposes as parsing HP-IB input, trace plotting, and other system utility functions. Each designer was assigned responsibility for one of the primary processes and certain auxiliary tasks. Then, interprocess communication requirements were determined.

At this point, the design team had established a simple, modular design with clear requirements and division of responsibility. These factors contributed greatly to rapid development time by allowing work to proceed at different rates in each of the various project areas. During the course of firmware development, it was unusual for any designer to be held up waiting for another to complete some task.

Firmware Architecture

Fig. 7 shows a conceptual block diagram of the firmware organization. In the figure, the rectangular blocks represent pSOS processes, the oval blocks are major modules that do not run in a unique process, and the shaded blocks are major data stores. Interprocess communication takes place via pSOS message passing and process signaling mechanisms.

When the analyzer is not processing inputs from the front panel, HP-IB, or Instrument BASIC, only two of the system's processes are generally active: the measurement sequencer and the display manager. This condition may be thought of as the steady state. In this mode, interprocess communication and swapping are minimized. This helps system performance and keeps the firmware operation relatively simple.

During steady-state operation, the measurement sequencer manipulates hardware and oversees the data acquisition required during each measurement loop. It passes newly acquired data to the math manager module, which executes physical coprocessing within the TMS320C25 chip. The measurement sequencer process is then free to continue acquiring new data while the postprocessing and display of previous data are underway. Math operations include calibrated data correction, display scaling, user math operations, and limit checking. When the math coprocessor has completed its operations, it interrupts the CPU to awaken the display manager process, which then routes the processed data to the display.

When a front-panel key is pressed, or when HP-IB commands arrive from the external bus or from an Instrument BASIC program, the firmware goes into a dynamic change management mode. These inputs are first detected in interrupt service routines. These, in turn, activate the user interface process, which preempts the measurement sequencer and display manager processes. The key codes or HP-IB commands are then directed through the instrument state module.

The instrument state module performs parameter coupling and qualification before entering new values into the analyzer's master state data structure. After the master state is updated, the instrument state module informs the measurement manager and the display manager processes that a certain type of state change must occur. This interprocess communication occurs via messaging over a pSOS exchange. If the changes require updating the data scaling or calibration constants, then the math manager module is also informed. Once all the specific change command messages are sent, the user interface process terminates, allowing the measurement and display manager processes to become active. These processes effect the desired changes. When the hardware setup and display scaling and format are consistent with the master analyzer state, the measurement manager process becomes inactive, awaiting further change command inputs from the instrument state module.

The measurement sequencer process then reactivates, resuming data acquisition. The display manager process converts from a dynamic change management operating mode to a relatively quiescent mode of drawing trace data and annotation on the display. At this point, the analyzer has resumed its steady-state operating mode.

Firmware Quality Assurance and Reuse

Several factors considerably increased the effectiveness of our quality assurance effort on the firmware. First, we kept our design solution proportional to the size of the problem. This resulted in a reasonably simple, robust firmware system that was well-understood by the entire

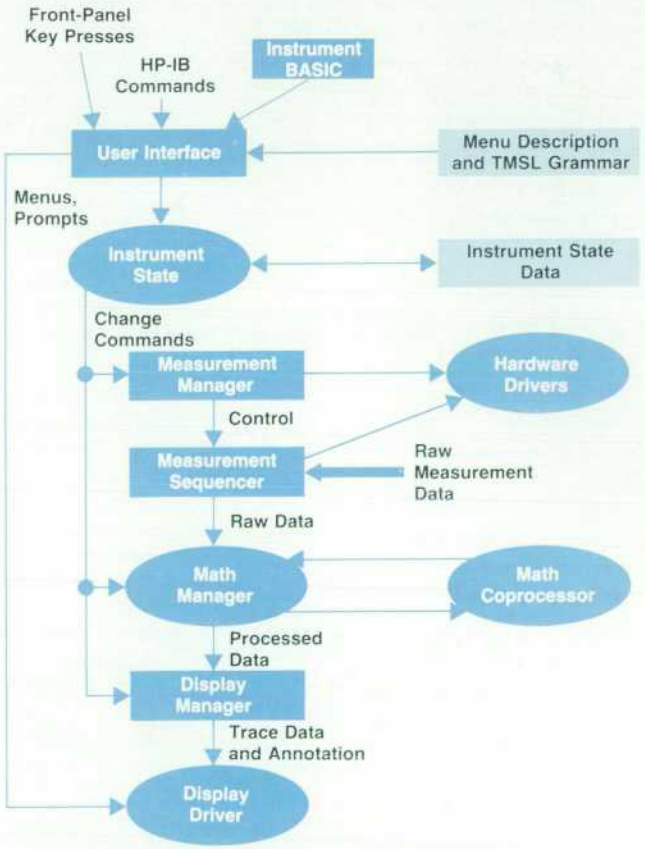


Fig. 7. *HP 3588A firmware block diagram.*

team.

Second, we implemented a very extensive automated testing system. One of the outputs of our user interface compiler (see "User Interface Compiler," page 57) was a list of every HP-IB mnemonic that the analyzer can recognize. This output file was used as a data base for a generalized computer-controlled random codes test, which attempted to cause the analyzer's firmware to fail. In past projects, the random codes test required months to design and hours per week to maintain. Our system was thorough, always up-to-date, and required virtually no maintenance.

Third, the HP 3588A's internal Instrument BASIC capability provided a very convenient way for the designers to develop test programs. These programs were kept on standard 3½-inch diskettes, which were easily moved between prototype analyzers. These programs served both as regression tests and as debugging and verification aids.

The firmware system developed for the HP 3588A has already been extensively reused on three projects at the Lake Stevens Instrument Division and one project at the Network Measurements Division. The HP-IB controller code has been reused by five local projects and by the Colorado Springs Division, the Spokane Division, and the Yokogawa-Hewlett-Packard Instruments Operation (YIO). The user interface compiler is in reuse on virtually all local projects and at YIO.

Acknowledgments

Many people contributed to the development of the HP 3588A. Jerry Daniels contributed to the original product definition, while Dave Bartle, Manfred Bartz, Bill Ginder, and Bob Witte helped with the original hardware implementation. Dave also helped with the firmware development later in the project. Rod Nemitz and Larry Sanders worked on much of the IF and ADC circuitry, and Yoshiyuki Yanagimoto helped with the LO implementation. Bob Kunz and Bill Spaulding managed the firmware development effort—the team included Ken Blue, Andy Cassino, Ben Mejia, Bryan Murray, Harry Plate, and Andy Purcell. Mechanical design was accomplished by Thatcher Harvey and Jay Miazga, and industrial design was done by Randy Eilert. The section manager for the project was Charlie Potter. Hans Mortelmans was the production engineer and had the challenging task of developing the methods and tests to verify the HP 3588A's performance. Many thanks to the team who developed the package and associated components for their continued support during the HP 3588A development. Special thanks also to the marketing team and to the people who produced a very complete documentation package for the instrument. The HP 3588A is among the first products developed by our division that employs a substantial amount of surface mount technology. For this reason a certain amount of process development occurred in parallel with the product development. Many thanks go to those people who brought the process up and assembled what must have seemed like an endless stream of prototype boards. Their efforts resulted in a very smooth transition into production.

References

1. D.D. Danielson and S.E. Froseth, "A Synthesized Signal Source with Function Generator Capabilities," *Hewlett-Packard Journal*, Vol. 30, no. 1, January 1979, pp. 18-26.
2. J.S. Epstein, et al, "Hardware Design for a Dynamic Signal Analyzer," *Hewlett-Packard Journal*, Vol. 35, no. 12, December 1984, pp. 12-17.
3. R.A. Witte and J.W. Daniels, "An Advanced 5-Hz-to-200-MHz Network Analyzer," *Hewlett-Packard Journal*, Vol. 35, no. 3, November 1984, pp. 4-16.
4. M. Dishal, "Alignment and Adjustment of Synchronously Tuned Multiple-Resonant-Circuit Filters," *Electrical Communication*, June 1952, pp. 154-164.
5. M. Engleson, *Modern Spectrum Analyzer Theory and Applications*, Artech House, 1984.

Authors

June 1991

6 HP 48SX Calculator

William C. Wickes

R&D project manager Bill Wickes has authored four books on HP calculators and numerous scientific papers on astrophysics, cosmology, and quantum mechanics. He was the software project manager during development of the HP 48SX scientific expandable calculator. Since joining HP's Corvallis Division in 1981, Bill has been the project manager for the HP 41 extended I/O ROM, the HP 75 math and advanced I/O ROMs, the HP 71B math and FORTH/Assembler ROMs, the HP 28C and HP 28S calculators, and the HP 82208B program development link. His professional interests include calculator design, mathematical software, and math education. He earned a BS degree (1967) in physics from the University of California at Los Angeles, and an MA degree (1969) in physics and a PhD degree (1972) in physics from Princeton University. Before joining HP, Bill was an assistant pro-

fessor of physics at Princeton University and the University of Maryland. Named an inventor in three patents on calculator design, he is a member of the American Mathematical Society, the American Astronomical Society, and the Mathematical Association of America. Bill coached the University of Maryland College Bowl team to a national championship in 1981. Born in Lynwood, California, he lives in Corvallis, Oregon, is married, has two children, is active in the Boy Scouts, and enjoys amateur astronomy, volleyball, and technical writing.

Charles M. Patton

Twistor theory, dynamical systems, symbolic computation, and visualization techniques are the professional interests of Charles Patton, who joined HP's Corvallis Division in 1982. Since then, he has worked on system and mathematical software design and programming for the HP 48SX scientific expandable calculator and the HP 28C and HP 28S calculators. He also helped develop the RPL operating system, and worked on math ROM software for the HP 75 and HP 71 calculators. He is the author of several articles in mathematical physics, and a coauthor of articles on representation theory and calculus incorporating the use of technology. He is currently coauthoring a new college textbook. A member of the American Mathematical Society, he is named as an inventor in patents for three techniques supporting symbolic computation in a ROM-based machine. Charles earned a BA degree in 1972 in mathematics and physics from Princeton University, and a PhD degree in 1977 in mathematics from the State University of New York at Stony Brook. Before joining HP, he was an instructor and visiting assistant professor in the Department of Mathematics at the University of Utah, and a mathematics researcher at the Institute for Advanced Studies.

13 HP 48SX Applications

Ted W. Beers

After receiving a BS degree in computer and electrical engineering from Purdue University in 1984, Ted Beers joined HP's Portable Computer Division as a customer support engineer. As a college co-op student, he developed an HP 41 calculator well log analysis module for a Houston oil consulting firm. As an HP software R&D engineer in HP's Corvallis Division, he helped develop the HP 28S scientific calculator. Working on the HP 48SX calculator, he designed the parameterized outer loop, interactive stack, key-handling system, and high-level display management. He also worked on the PC program development link to the HP 48SX calculator. Ted is an author of technical publications on the HP 41 operating system and games, and on Solvit, a calculator program to solve multiple-variable equations. Born in West Lafayette, Indiana, he lives in Corvallis, Oregon, is married, and enjoys hiking with his wife and two beagles.

Diana K. Byrne

Diana Byrne was responsible for the design of the HP 48SX calculator's plotting and graphics, and worked on the EquationWriter. She received a BS degree (1982) in mathematics from Portland State University, and MS degrees in mathematics (1987) and computer science (1988) from the University of Oregon. Before joining HP's Corvallis Division in 1988, she was a mathematics instructor at the University of Oregon and Portland State University. She is a member of the Mathematical Association of America and the Society of Women Engineers. A resident of Corvallis, Oregon, Diana has two sons and enjoys skiing, dancing, making beer, and swimming in the Willamette River.

Robert W. Jones

Software design engineer Max Jones designed the object-oriented menu and key handling systems for the HP 48SX scientific calculator. He also has written documentation for other calculators and computers at HP's Corvallis Division, which he joined in 1982. Max earned a BS degree in math from the University of Washington in 1976. Before joining HP, he was a partner in a tugboat in Seattle. Born in Mt. Vernon, Washington, he lives in Corvallis, Oregon, is married and has a son. He enjoys playing Prof. Longhair-style blues piano.

Gabe L. Eisenstein

During evening hours, Gabe Eisenstein teaches philosophy at the University of Oregon, Oregon State University, and Portland State University, and writes articles for philosophy journals. He incorporated VisiCalc in the HP 75 calculator, FORTH in the HP 71 calculator, operating systems in the HP 28 and HP 48SX calculators, and EquationWriter in the HP 48SX calculator. He joined HP's Corvallis Division in 1981 as a software engineer. His eclectic interests are reflected in his education—a BA degree (1974) in religious studies at the University of Michigan, and a PhD degree (1979) in philosophy and a BA degree (1981) in computer science at the University of Texas. He is named an inventor in a patent pending for the HP 48SX EquationWriter application and a coinventor in a patent on calculators with a user-accessible object stack. Born in Chicago, Illinois, Gabe lives in Corvallis, Oregon, and enjoys backyard astronomy, alternative music, and skiing.

Patrick J. Megowan

Making computing accessible, relevant, and rewarding to people is the professional goal of Pat Megowan, who joined HP's Corvallis Division in 1981. He designed and developed the solve, plot, stat, time, and I/O applications for the HP 48SX scientific calculator, and helped design its general application structure, keyboard, and screen layout. Pat also helped design the HP 41, 18C, 19B, 27S, 17B, 20S, 10B, and 21S calculators. Before joining HP, he developed pattern recognition software at the IBM Federal Systems Division. A 1979 graduate of the California Polytechnic State University at San Luis Obispo, Pat earned a BS degree in mathematics, and has done postgraduate work in computer science at the University of California at Santa Barbara and in physical geography at Oregon State University. Pat is active in community planning and land use issues around his hometown, Corvallis, Oregon. He is married, has two daughters and a son, and enjoys natural history, rock climbing, telemarking, and playing the violin, mandolin, and guitar. Pat competes in freestyle bike trials, collects and cultivates trees, and "moonlights" in architecture and landscape design.

22 HP Solve Equation Library

Eric L. Vogel

R&D software project manager Eric Vogel joined HP while he was still a student in 1976 when HP's Corvallis Division first opened. He earned a BS degree in mechanical engineering from Oregon State University in 1978, and was hired on a permanent basis as an application engineer in 1980. Eric developed application software for the HP 41 and HP 75 calculators, and served as a lead software engineer for the operating system and development tools for the HP 94 industrial computer. He also helped test and develop the HP 17B, HP 19B, HP 27S, and HP 20S calculators, and was a project leader and then project manager for development of the HP Solve Equation Library card for the HP 48SX scientific calculator. The author of owner and reference manuals for calculator products and a collection of petroleum reservoir engineering programs, Eric is named as a coinventor in patents pending for calculator unit management techniques and the Multiple Equation Solver. Born in Los Angeles, California, he lives in Corvallis, Oregon, is married, and

has two sons and a daughter. He enjoys golf, foreign travel, reading, woodworking, and remodeling his house.

25 HP 48SX Hardware Design

Mark A. Smith

Mark Smith joined HP's Corvallis Division in 1980, shortly after receiving his BS degree in mechanical engineering from the California Polytechnic State University at San Luis Obispo. As an R&D mechanical engineer, he helped develop the HP 48SX calculator and also worked on the HP 10C, 15C, 16C, 75D, 18C, 28C, 19B, and 28S calculators. A licensed professional engineer, he has authored articles on the mechanical design and hinge interconnect design of the HP 18C and HP 28C calculators. Born in Livermore, California, he lives in Corvallis, Oregon, is married, and has two sons. He enjoys skiing, dirt bike riding, and coaching a youth soccer team.

Lester S. Moore

Les Moore joined HP in 1959 after serving in the U.S. Air Force from 1955 to 1959. He graduated from San Jose City College in 1955. An electronic design engineer with HP since 1959, Les has worked on the HP 65, HP 45, HP 21, and HP 28 calculators, and produced the system design for the HP 48SX scientific calculator. Born in San Francisco, California, he lives in Corvallis, Oregon, is married, and has three grown children. Les enjoys gardening, growing orchids, woodworking, and growing koi fish in a 4,000-gallon fish pond.

James P. Dickie

Hardware project manager Jim Dickie joined HP's Corvallis Division shortly after receiving a BSEE degree in 1976 from Arizona State University. He earned an MSEE degree in 1983 from Oregon State University. Jim was the project manager for the electrical and mechanical systems for the HP 48SX calculator. He designed the microprocessor for the HP 71 calculator, and helped design the HP 29C calculator and the Series E/C calculator family. He is named an inventor in a patent on the architecture of the HP 71 microprocessor. Born in San Diego, California, Jim was raised in Phoenix, Arizona, and lives in Corvallis, Oregon. He is married, has three children, and enjoys running, playing softball, skiing, reading, and coaching youth soccer.

Preston D. Brown

Soon after graduating from the University of Florida with a BSEE degree in 1981, Preston Brown joined HP's Corvallis Division as an R&D engineer. He was responsible for custom IC and system design for the HP ThinkJet printer and the HP 41CV, 41CX, 18C, 28C, 19B, 28S, and 48SX calculators. A member of the IEEE and the AAAS, Preston is the author of conference articles on pattern matching and a display driver. Born in Pensacola, Florida, he lives in Eugene, Oregon, and enjoys downhill skiing.

David L. Smith

Plastic design is the professional interest of Dave Smith, a mechanical engineer who joined HP's Corvallis Division in 1982. He received a BS degree in mechanical engineering that same year from the California Polytechnic State University at San Luis Obispo. Dave did mechanical design and calculator R&D for the HP 48SX scientific calculator, and also worked on the HP 71B, 18C, and 94 calculators and the HP 82240A thermal printer. Born in Klamath Falls, Oregon, Dave lives in Corvallis, is married, and has a son. He enjoys woodworking, skiing, softball, and volleyball.

Thomas B. Lindberg

Tom Lindberg was an R&D and production engineer for the HP 48SX scientific calculator. He also was a project engineer during development of the HP 71B, 18C, 28C, 19B, and 28S calculators. Tom joined HP's Corvallis Division in 1981, shortly after receiving his BS degree in mechanical engineering and materials science from the University of California at Davis. He is a registered professional engineer. Born in Royal Oak, Michigan, he lives in Corvallis, Oregon, and enjoys electronic music composition.

M. Jack Muranami

Jack Muranami designed the memory card, memory card connector, and battery contacts for the HP 48SX scientific calculator. He joined HP's Corvallis Division shortly after graduating from Rice University in Houston, Texas, in 1985 with a BS degree in mechanical engineering. Jack also helped develop the HP 82240A printer and the HP 82242A IR module for the HP 41C calculator. Born in Tokyo, Japan, he lives in Corvallis, Oregon, is married, and has two children. Jack enjoys reading, cooking, gardening, traveling, and buying toys for his children during business trips to Japan.

35 HP 48SX I/O System

Steven L. Harper

Steve Harper was responsible for the design of the memory cards, display, and I/O circuitry for the HP 48SX scientific calculator. He has also worked on the HP 9551D instrument calibration system, the HP 01 calculator/watch, and the HP-IL system and devices. Steve joined HP's Automatic Measurement Division in 1972, shortly after graduating from Brigham Young University with BS and MSEE degrees. The coauthor of a book on the HP-IL system, Steve is named as an inventor in an HP-IL protocol patent. Born in Medford, Oregon, he lives in Corvallis, is married, has five children, and enjoys amateur radio, windsurfing, and singing in the church choir.

Robert S. Worsley

Responsible for I/O printing and firmware for the HP 48SX calculator, Bob Worsley joined HP's Corvallis Division in 1977. He also helped develop firmware for the HP 41 and HP 42S calculators. He received a BSEE degree in 1977 from San Jose State University and an MSEE degree in 1982 from Oregon State University. A resident of Corvallis, Oregon, Bob enjoys hiking and cross-country skiing.

40 HP 48SX Manufacturing

Richard W. Riper

Machine design and computer-controlled assembly machines are the major professional interests of Rick Riper, a manufacturing engineer who joined HP's Corvallis Division in 1982. He graduated from Oregon State University that year with a BS degree in mechanical engineering. Rick coordinated the manufacturing engineering and final assembly of the HP 48SX calculators. He also has worked on the HP 10 Series and HP 18C/28C families of calculators. A registered professional engineer in Oregon, he was born in Chester, Pennsylvania, lives in Corvallis, Oregon, is married, and is active as a Cub Scout den leader. He enjoys woodworking and photography.

44 HP 3588A Spectrum Analyzer

Eric J. Wicklund

Eric Wicklund was the hardware project manager for the HP 3588A spectrum analyzer. He also helped develop the voltmeter for the HP 3497A data acquisition and control unit, the HP 3326A two-channel synthesizer, and the HP 3561A and HP 3562A dynamic signal analyzers. He is named an inventor on a patent pending on IF calibration for the HP 3588A spectrum analyzer. Eric received a BSEE degree in 1977 from the University of Washington, and then joined HP's Lake Stevens Instrument Division that same year. He serves as a member of the Seattle Pacific University Electrical Engineering Advisory Council. Born in Seattle, he lives in Snohomish, Washington, is married, and has three children. He enjoys amateur radio, woodworking, and astronomy.

Joseph F. Tarantino

R&D engineer Joe Tarantino contributed to the design of the input and IF circuits and portions of the local oscillator for the HP 3588A spectrum analyzer. He also is a co-developer of the local oscillator and reference circuitry for the HP 3577A network analyzer. Joe earned a BSEE degree (1979) and an MSEE degree (1981) from Purdue University, and joined HP's Lake Stevens Instrument Division in 1981. He is named an inventor on two patents pending for IF calibration for the HP 3588A spectrum analyzer and the gated sweep capability in the HP 3585B spectrum analyzer. Born in Milwaukee, Wisconsin, he lives in Seattle, Washington. He enjoys hiking, bicycling, soccer, and archaeology.

James H. Cauthorn

Jim Cauthorn joined HP's Lake Stevens Instrument Division as a customer support engineer after he received his BSEE degree from the University of Arizona in 1984. After moving to the R&D lab in 1986, he began developing firmware systems for signal analyzer products, including the HP 3588A. His professional interests include analog and digital signal processing, communications, and real-time firmware system design. Before joining HP, he worked as a co-op student on read-channel electronics for an IBM magnetic tape data storage division. Born in New Orleans and raised in Arizona, he lives in Everett, Washington. Jim enjoys leisure time with his two sons, dancing, guitar music, movies, books, and hiking.

Kirsten C. Carlson

Production engineer Kirsten Carlson performed automated testing of the HP 3388A spectrum analyzer. She joined HP's Lake Stevens Instrument Division in 1982, a month after she received a BSEE degree in 1981 from the University of Washington. Since then she has worked in hardware and software development, product investigations, software quality, and production engineering for numerous HP products. As a production engineer, she was responsible for HP 3325A/B synthesizer/function generators, HP 3335A and HP 3336A/B/C synthesizer/level generators, and the HP 3312A function generator. Born in Vancouver, Washington, Kirsten lives in Snohomish and has two sons, one of whom she adopted from an orphanage in Cimpulung, Romania. She enjoys her vacation home at the beach, reading, walking, biking, and church activities.

Jay M. Wardle

Digital and analog signal processing are the professional interests of Jay Wardle, a design engineer who joined HP's Lake Stevens Instrument Division in 1984. He designed the digital IF, coded math coprocessor, and diagnostic software for the HP 3588A spectrum analyzer. Before that, he designed coprocessor and diagnostic software for the HP 35660A dynamic signal analyzer. Before joining HP, Jay performed hardware design for John Fluke Manufacturing Company and Advanced Technology Labs. He earned a BSEE degree in 1978 from Washington State University and an MSEE degree in 1980 from the University of Washington. Raised in Yakima, Washington, Jay lives in Seattle, is married, and has two daughters. He spends most of his leisure time with his young children, and also enjoys hiking, climbing, and spelunking.

Timothy L. Hillstrom

After joining HP's Lake Stevens Instrument Division in 1982, Tim Hillstrom designed and developed frequency reference, calibrator, and trigger circuits for the HP 3588A spectrum analyzer. His major professional interests include oscillators, frequency synthesizers, and phase noise. He is named an inventor on one patent and on another patent pending on spectrum analyzer measurement features and circuits. Tim also is the author of magazine and conference articles on oscillator theory and design. He earned a BSEE degree in 1982 at the University of Portland and an MSEE degree in 1986 in analog/RF design from the University of Washington. Tim and his wife recently returned from a successful nine-week journey to Romania, bringing back two children adopted from orphanages in Arad and Timisoara. A tennis fanatic who was born in Hollywood, California, Tim lives with his family in Everett, Washington.

Roy L. Mason

The receiver IF filter, source, and receiver input image filter of the HP 3588A spectrum analyzer were developed by Roy Mason. A 1984 graduate of the University of Nebraska at Lincoln with a BSEE degree, he joined HP's Lake Stevens Instrument Division that same year. He is now working toward an MSEE degree. Roy was in the U.S. Air Force from 1976 to 1980 and a member of the Air National Guard from 1980 to 1984. Born in Omaha, Nebraska, he lives in Lake Stevens, Washington, is married, has two children, and enjoys woodworking and machine metalworking (the latter as a result of his work with high-frequency filters).

65 HP GlancePlus

Rex Backman

Rex Backman is a software engineer in the performance technology center at HP's Application Support Division. He worked on the development of the HP GlancePlus/XL diagnostic performance tool. In the past, Rex worked on MPE XL operating system testing and the development of other performance products such as GlancePlus/V and CAPLAN/XL. Rex joined HP in 1982, shortly after receiving a BS degree in computer science from California State University at Chico. He authored an article on performance management for Interact magazine, and several conference papers presented at Interex. His professional interests include user interfaces and ease-of-use features. Born in Redwood City, California, Rex lives in Citrus Heights and enjoys skiing, travel, and southwestern art.

71 Product Development Process

Douglas Daetz

Quality management methods and the social implications of technology are the professional interests of Doug Daetz, who joined HP Laboratories in 1985. He was an early participant in the "design for manufacturability" and "factory modeling" projects in HP Laboratories' manufacturing research center. Now a quality engineer in HP's Corporate Quality Department, Doug has been promoting the use of

quality function deployment (QFD) in product and service development. Before joining HP, Doug was the director of quality at Shugart Corporation and an assistant professor of industrial engineering at Stanford University. He earned a BE degree in 1962 in electrical engineering at Yale University, an MSEE degree in 1963 and a PhD degree in electrical engineering and computer science in 1968 from the University of California at Berkeley. He is a member of the IEEE, the American Society for Quality Control, the Society of Manufacturing Engineers, and the American Society for Engineering Education. He is the author of articles on design for manufacturability and quality and quality function deployment, and originator of the multivariable display technique now called the radar chart. Born in Providence, Rhode Island, Doug lives in Sunnyvale, California, is married, has three sons, and enjoys stamp collecting, poetry, and life drawing.

William P. Carmichael

Bill Carmichael's professional interests include developing and applying robust statistical techniques to real-world problems. To accomplish this, he earned an AB degree in mathematics and statistics in 1973, and a PhD degree in statistics at the University of California at Berkeley in 1980. He joined HP's Stanford Park Division in 1982 as a statistician and implemented a total quality control (TQC) program. As a productivity manager, he implemented continuous process improvement techniques in R&D at HP's Stanford Park Division, and developed break even time (BET) software at HP's Corporate Engineering. Before joining HP, he was an assistant professor of statistics at the University of Wisconsin in Madison. Born in Red Bluff, California, he lives in San Francisco and enjoys hiking, biking, and restoring Victorian-era edifices.

Edith Wilson

As HP's manager of product definition and corporate manufacturing in Europe, Edith Wilson wears three hats. She is responsible for HP's product definition process, the worldwide benchmarking of product definition, and starting HP's product generation effort in Europe. Previously, Edith served as section manager for HP product definition at Corporate Engineering. After joining HP Laboratories in 1980 as a mechanical engineer, she worked on the electron beam machine project, designing portions of the wafer-stage mechanism. She then joined the Optoelectronics Division and developed a tape-and-reel LED process. As a project manager, she designed and developed HP's ASCII-addressable 5 × 7 dot matrix LED display family. As a section manager, she helped develop custom LED displays, printheads, and LED auto brake lights. Before joining HP, Edith was a mechanical engineer with Telesensory Systems, Inc., working on a hand scanner for the Optacon tactile imaging reader for blind people. She earned a BSE degree in biomedical engineering in 1977 from Duke University, an MS degree in mechanical engineering in 1979 from Stanford University, and an Engineer's Degree in mechanical engineering at Stanford University in 1990. The author of articles on data analysis, TQC, and product definition at HP, Edith's major professional interests include project management and design innovation. She is active in community projects, including the opening of the Silicon Valley Technology Museum. Born in New Haven, Connecticut, Edith was recently married, and now lives in London, England. She is currently remodeling a historically registered house, and she enjoys sports and gardening.

Spencer B. Graves

Spencer Graves joined HP's Santa Clara Division in 1982 as a statistician. In 1984, he transferred to the Computer Supplies Operation (now the Direct Marketing Division) where he served as a statistician and quality services manager for TQC planning, course development, and training, and developed the prototype for the HP TQC course. Spencer joined HP from the University of Wisconsin at Madison, where he received a PhD degree (1983) in statistics. He earned a BS degree (1967) in aerospace engineering at the University of Colorado in Boulder, an MS degree (1972) in industrial engineering at the University of Pittsburgh, and an MA degree (1975) in mathematics at the University of Missouri in Kansas City. Before joining HP, Spencer was an operations analyst at the Midwest Research Institute in Kansas City, an industrial engineer and office manager in the U.S. Air Force, and a volunteer teacher in Latin America. Spencer is now a consultant and is teaching Industrial Systems Engineering at San Jose State University. His specialties include improving strategic management and the product development process. He is a member of the American Society for Quality Control, the American Statistical Association, the Institute of Management Sciences, and the American Economics Association. He has published papers on strategic management and the application of improvement concepts to national and international problems. Born in Oberlin, Kansas, he lives in San Jose, California, is married and has a stepdaughter.

77 DSEE

David Lubkin

David Lubkin was the project leader for DSEE heterogeneous configuration management at HP's Apollo Systems Division for the last three years. Before joining Apollo, he worked on terabyte archival storage and supercomputer operating systems at Lawrence Livermore National Laboratory for five years. He received a BA degree (1978) in linguistics from the State University of New York at Stony Brook and an MS degree (1982) in computer science from Michigan State University, where he also did a year of doctoral work in artificial intelligence. He is a member of the Science Fiction Writers of America. David was born in New York but spent five years living in Israel (where he sampled the Yom Kippur War). He now lives in Nashua, New Hampshire, is married, and has a daughter. The eldest of 13 children, he is learning to fly.

84 Parallel Development via RCS

John W. Goodnow

Computer and system architecture, operating systems, and image processing are the professional interests of John Goodnow, a project manager at HP's Imaging Systems Division. In 1983, he joined HP's Andover Division, now the Imaging Systems Division, shortly after receiving a BS degree in electrical engineering from the University of Pennsylvania. He earned an MS degree in electrical engineering in 1988 from Stanford University through the Honors Coop program. After Joining HP, he worked on HP PageWriter electrocardiographs and ultrasound system software as a development engineer. John is now a project manager working on ultrasound software development. Born in York, Pennsylvania, he lives in Andover, Massachusetts, is married, and enjoys windsurfing, skiing, woodworking, and vacationing on Martha's Vineyard.

90 Integrated Project Support

Ronald F. Richardson

Ron Richardson began his career as a programmer analyst when he joined HP's Corporate Parts Center in 1979. As a quality process engineer in HP's Roseville Networks Division, he was the lead for the team developing the technical computing environment for cooperative computing. In the past, he has worked as an R&D software development engineer and a program manager at other HP divisions. He is now a process consultant working to enhance hardware and software productivity with new design tools and methods. He will receive a BS degree in computer science from California State University at Chico this year, and is concurrently registered in an MSCS program there. Born in Santa Cruz, California, he lives in Rocklin, is married, and has three children. He enjoys motorcycling, hunting, fishing, hiking, and camping with his family.

Easy-to-Use Peformance Tools with a Consistent User Interface Across HP Operating Systems

By involving customers in the product development process and incorporating their feedback into the product, HP GlancePlus has eliminated the mystique commonly associated with performance tools. Exception-based reporting displays only the interesting data.

by Rex A. Backman

ALL HP COMPUTER SYSTEM users should be able to use and understand their performance tools. This concept was the guiding principle behind the HP GlancePlus family of diagnostic performance tools created by the performance technology center of HP's Application Support Division. Typically, computer system performance tools have been directed towards the technically advanced user. The focus on tools that required a high degree of operating system knowledge left out novice users. Feedback from our customers regarding computer system diagnostic performance tools subtly addressed this issue by requesting performance tools that were consistent in look and feel across platforms (MPE V, MPE XL, and HP-UX operating systems), were easy to use, and were low in cost. To answer these customer needs, the HP GlancePlus family of diagnostic performance tools was created.

Performance diagnostic tools provide the customer with a means of seeing and understanding the current status of the machine at any time. These tools are typically used to diagnose machines when an abnormal situation is present. Abnormal situations can be a variety of events ranging from poor terminal response time for a specific end user to overall sluggishness in the machine. Whatever the symptom, the customer has deemed something to be out of the ordinary and the performance tool is executed to display real-time data about the machine so that appropriate action can be taken to correct the situation. HP GlancePlus is typically used to determine information such as who is the main CPU consumer, where are the record pointers within a file, who is accessing a certain file, or what disk drive is most heavily used. All of these actions are typical of the diagnostic quadrant of performance management, and keep the user abreast of the current status of the machine. The performance management quadrant and other types of performance tools are described on page 70.

Background

Historically, many HP performance tools have started as software laboratory prototypes or special tools. These software prototypes were usually trying to address a specific and immediate need in a project-development cycle. For example, new operating systems development work necessitated some form of a monitor to track modifications made in system level algorithms. Upon completion of the project-development cycle, the performance tools would sometimes evolve into a salable customer product. Unfortunately, these customer performance tools still retained the roughness associated with laboratory prototypes. While some effort was spent on improving the prototypes, this work usually concentrated on an improved user interface with no true thought given to the performance data that was displayed. The result of this quick transformation from development tool to customer product was a utility that provided too much data, provided data that was not applicable to the customer's needs, and was too expensive. As a result, performance management on our customer's machines was often relegated to technical experts. This caused an aura of mystique about the performance discipline because the thinking was that if you had to be an expert to use a tool, then the discipline must truly be difficult.

Removing the Mystique

The engineers at the performance technology center recognized, like many others, that performance management was not difficult nor should Hewlett-Packard customers find it so. Several goals were identified at the start of the product cycle to ensure that the mystique about performance management was eliminated. An aggressive development schedule, coupled with frequent and repeated customer feedback sessions, was part of the development methodology used to keep the product team focused on the goal of creating a useful, more informative performance tool for our customers. Customer feedback was the essential checkpoint vehicle used to ensure that the tool was providing valuable and pertinent data (see "Design Prototyping for HP GlancePlus" on page 69).

This customer feedback loop was important because the HP GlancePlus family is based on a different type of model than previous tools. To help remove the mystique of performance tools, HP GlancePlus incorporates a method of exception-based performance reporting. This approach allows tool designers to limit the data displayed to only

those few metrics that are considered interesting. This concept allows the end user to be concerned only with the interesting events on the machine. By focusing only on these interesting events, end users can ascertain what is wrong much faster.

In our diagnostic tools, information can be divided into various hierarchies such as global CPU use, global I/O activity, global memory allocation, and process-specific use. The first three of these hierarchies are condensed into single screens to provide information to the tool user. The process hierarchy presents a challenge because of the possible volume of data. To lower this volume, the exception-based methodology provides only information on processes that have some kind of activity during a specified time interval. Examples of activity for a process include CPU time consumption, disk reads or writes, and terminal transactions. Processes that have engaged in any of these activities during the specified time interval are considered exceptions and are classified as interesting. A process that does not have any activity is of no interest and therefore is not reported.

Additional benefits from the exception-based reporting method are that HP GlancePlus customers are not overwhelmed with useless data. The data that is displayed has direct impact on their ability to identify and understand the current performance problem. Exception-based reporting and optional color graphic displays result in a tool that is much more concise and informative than previous efforts. This change in data representation is the key to making HP GlancePlus performance tools easier to use across all levels of technical expertise.

As previously mentioned, earlier generations of performance tools provided too much in-depth reporting on system metrics. Many of these metrics were found hidden deep within the operating system kernel and were of no relevance to the typical end user. These additional metrics also made the situation worse by requiring unnecessary display screens, causing confusion when moving from screen to screen. One of the design goals for HP GlancePlus was to eliminate this problem. With the exception-based method of reporting, fewer display screens are used to display only valuable and pertinent metrics for performance diagnosis. Positive customer feedback to this approach was received almost immediately. Sites involved in the design prototyping phase almost always reported that less technically adept end users were finding the tools easy to use and understand. Most important, end users reported that the tools were providing a service. Time and effort to identify performance problems were lowered, and many sites repeatedly stated that through the use of the new tools they had easily identified performance problems and were able to take the appropriate action with much less anxiety than before.

Versions and Algorithms

Three versions exist in the HP GlancePlus family, one for each specific operating system platform. Represented platforms are the MPE V, MPE XL and HP-UX operating systems (including the workstation and HP 9000 Series 800 versions). The HP GlancePlus version for each environment or platform had identical design goals and each environment uses similar algorithms to retrieve the required performance metrics.

Differences in low-level hardware and software structures require each model to depend on different sources of data to feed the performance algorithms. These core differences are handled independently by each specific model. However, a common and similar user interface guarantees that customers have a consistent perspective of their performance metrics (see Figs. 1, 2, and 3). This consistent user interface across the HP platforms was viewed as way to bring a level of comfort to end users.

Previous tools did not offer any degree of consistency, and this added to the mystique that we were trying to remove. The comfort level was low in shops that were running in mixed modes. Data centers that included more than one operating system platform, such as MPE V and MPE XL, provided performance tools with different user interfaces. By contrast the user interface consistency offered by HP GlancePlus allows these users to switch from machine to machine with ease.

Performance algorithms are used by the HP GlancePlus

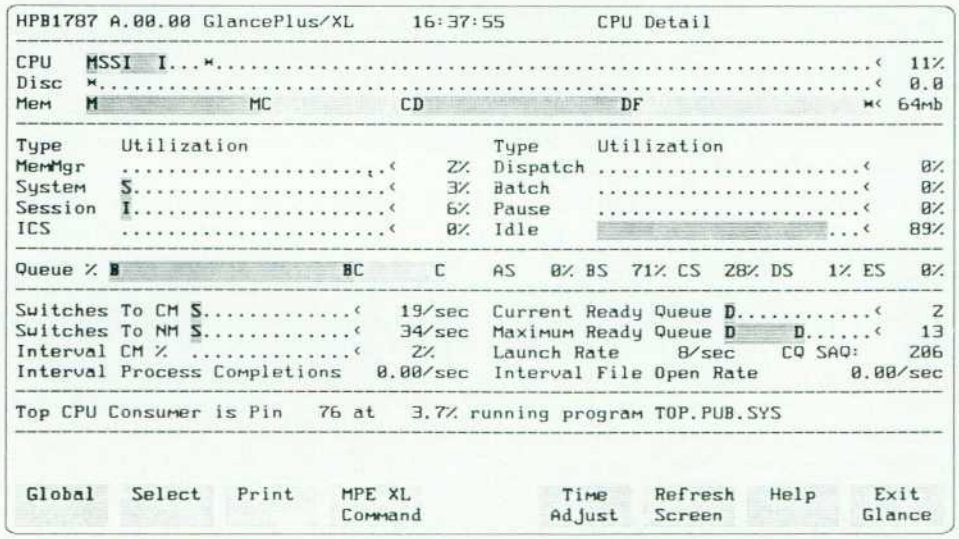


Fig. 1. *A typical HP GlancePlus/XL user interface screen for for HP MPE XL operating systems. This is a CPU detail screen which shows CPU use for a specified time interval. The data is broken down by type and various MPE XL performance metrics such as switches, launch rates, and so on.*

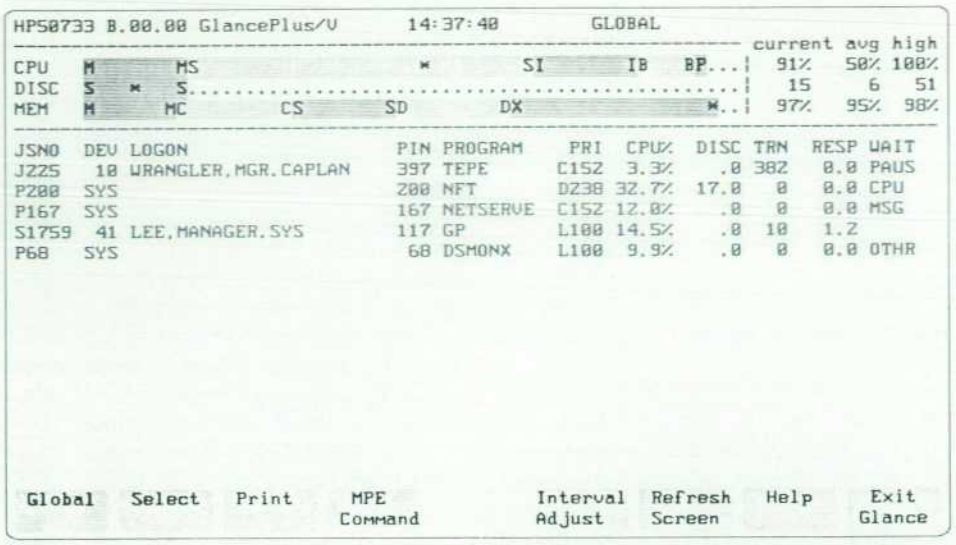


Fig. 2. *This screen shows the default global screen for HP GlancePlus/V for HP MPE V operating systems. The processes listed below the global section are the processes that were of interest during a particular time interval. The example shows that out of 400 or so processes on the machine only six were of interest. The global section shows the global perspective which allows the user to identify the possible performance bottleneck area quickly.*

tools to compute metrics for each specified time interval. The differences in software structures across the platforms mentioned above require different access methods to acquire the data. Once acquired, the algorithms are basic ones such as deriving the rate per minute of terminal transactions, disk use percent, and CPU use percent.

The HP GlancePlus tools are designed to be used interactively, or in a real-time mode, with the performance metrics displayed as rates, absolute values, or percentages. To derive these values two time points are required. The difference between the time points is referred to as the *interval*, and is used as the base for all metrics the user sees. The interval can be as small as five seconds or as large as several minutes. Usually, end users configure the interval between thirty and sixty seconds. As each interval expires, another time point is met, data points are collected, and performance metrics are computed. This refreshing of the performance metrics gives the end user a continuous perspective of what is happening on the machine. Examples of these metrics include system disk rates, which state the number of physical disk I/Os per second, main memory use, which states how much of main memory is being used in megabytes, and global CPU use, which shows the percentage of overall use of the CPU in the most current interval.

Data Sources

Data sources for the HP GlancePlus family differ from platform to platform (see Fig. 4). The underlying differences in the data sources are hidden from customers. However, these separate and unique data sources are the main contributors to the HP GlancePlus performance tools. Each platform has a subsystem associated with the base operating system called the measurement interface. Each measurement interface, regardless of its platform, is composed of two elements: instrumentation and counters. The instrumentation interface is where certain events within the operating systems are recorded. This is accomplished by adding code to specific modules of the operating system kernel that record the activities that the specified modules perform. The added code consists of high-performance statements that simply update the proper counters reflect-

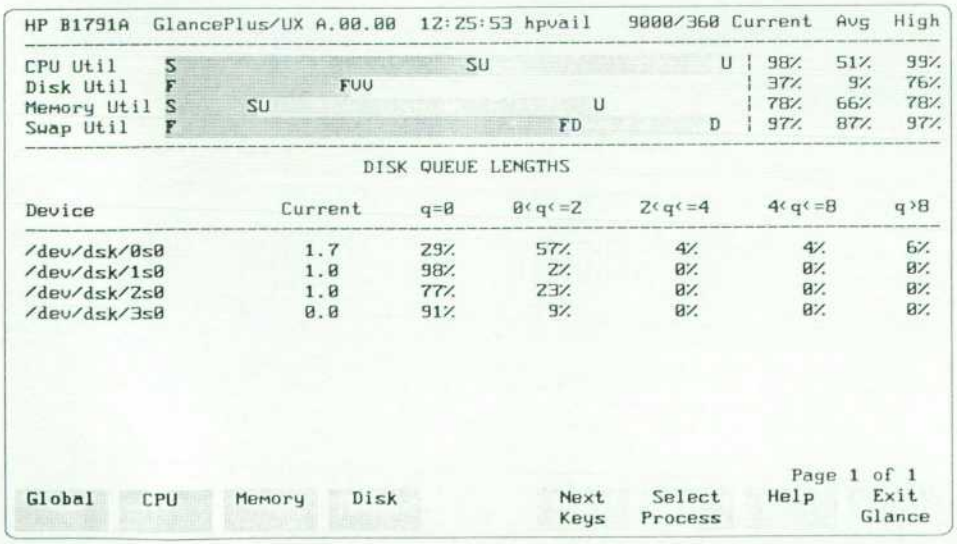


Fig. 3. *This is an HP GlancePlus/UX screen for HP-UX operating systems. This screen displays the disk queue length screen. It shows the queue length (how many processes are waiting to use the disk drives) of the drives present on the machine plus past queue history.*

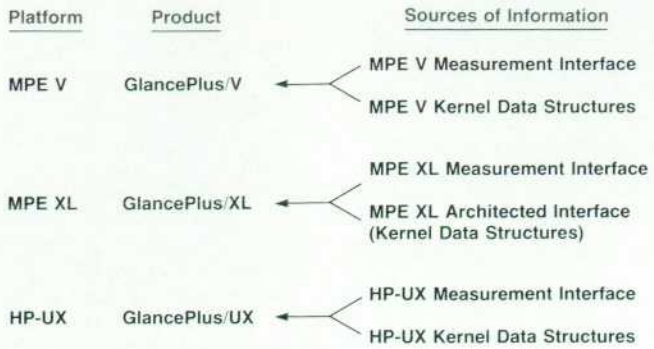


Fig. 4. *Data sources for HP GlancePlus.*

ing the action that has taken place. For example, the completion of a disk read is marked in the disk driver code. Counters are nothing more than accumulators for certain types of operating system events. Counters can be viewed as time stamps signifying the amount of time an event has taken.

Fig. 5 shows the logical map of the MPE XL measurement interface data structure. This structure shows the counters used by the HP GlancePlus/XL tool. While this structure is unique to the MPE XL operating system, it is common for measurement interface structures to organize data in a hierarchy of classes of global, process, and I/O events. In the case of the MPE XL operating system, the counters are organized into these three classes at the highest level. The subclass level represents an intermediate level of organization that provides the ability to identify objects of a similar type. For example, a subclass in the I/O class identifies each logical device present on the machine. Subclasses do not exist in the global class, but they do exist in the process class to identify each process. Each subclass is made up of groups of counters. The counters in each group are logically related by function. An example would be the file system counters for a process. This group of counters contains all the counters associated with file system activities for a specific process. To access a counter, the program must access the proper class (global, process, or I/O), select the proper subclass if applicable, and then access the group that contains the specific counter of interest.

While each operating system has a measurement interface, the sets of counters differ. This difference presented a challenge for the HP GlancePlus development team because we had to develop the appropriate measurement interface for each platform and still present to the end user similar metrics regardless of the platform. The challenge was overcome by mapping the disparate measurement interface counters to a common set of performance metrics that all models shared. While some metrics are unique to each platform, these are in the minority. The majority of the metrics did map to common data items for each version of HP GlancePlus.

Another source of data is nonperformance-related items residing in certain operating system structures such as process-specific information not contained in the platform's

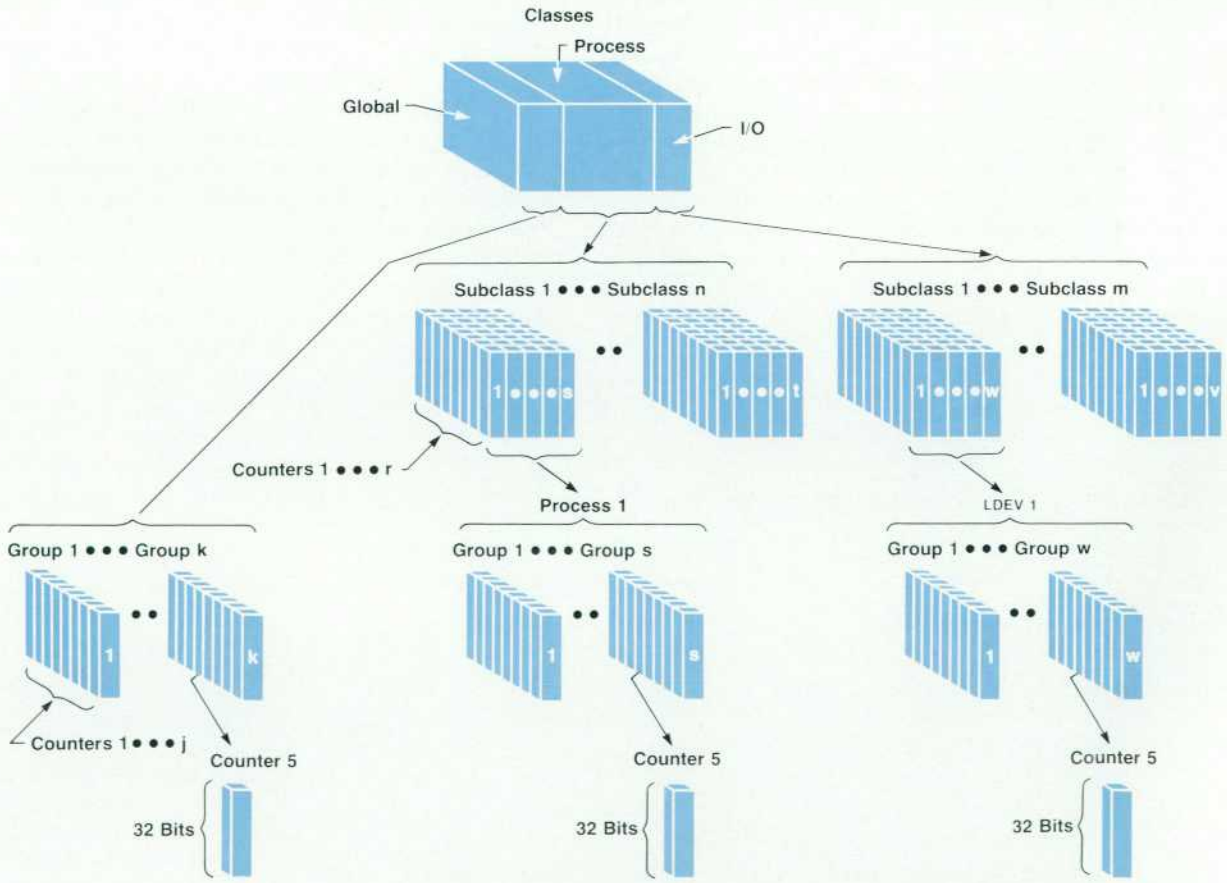


Fig. 5. *The measurement interface model for the HP MPE XL operating system.*

Design Prototyping for HP GlancePlus

Design prototyping is the process of placing a very early product prototype in the hands of a select number of prospective customers and allowing them to help design the product, test the functionality, and refine the product in production environments. This process requires that there be a continuous dialog with the selected customers and a quick response to their requests within days.

For HP GlancePlus we wanted this set of customers to give us a good representation of our target market. The eight customers selected ranged from naive to sophisticated, from single-person shops to large data processing shops, and from sites with only HP MPE V or HP MPE XL operating systems to sites with both operating systems. The overriding requirement was that the customers had a need to use the product.

One week after sending the prototype programs, we called each customer to find out how they had used the prototype and how it could be improved. Interviews were conducted directly with the customers so that we could completely understand what they needed. Anything that two or more of the customers suggested was implemented for the next prototype version. Opinions from the development team that disagreed with customer consensus were discarded since we were not developing a product to sell to ourselves. We then made the changes, transmitted them electronically to our prototype customers, and began the cycle again.

Four repetitions of this cycle were achieved within six weeks. When customers saw changes they had requested implemented within days, their general reaction was disbelief. Disbelief turned to high enthusiasm in providing feedback as they realized their ideas were not only being heard but acted upon immediately. This methodology allowed correction of a lot of irritations that often spur customers to ask "Didn't anyone use this product before it was released?" It also allowed the addition of needed functionality that normally is not recognized and added until customers submit enhancement requests. And most important, it created a user interface shaped by users.

Some valuable lessons were learned from design prototyping. First of all, we discovered that when users have access to updated versions of a piece of software rapidly, they can easily decide on their real needs. For example, users may have their specifications met exactly and then find out that it only helps them to understand what they really need. We also learned that prototyping can be done with third-generation programming languages. Finally, we learned that design prototyping can give a product the feel of a second- or third-release product at introduction. The benefit to both customer and development group are great in overall cost reduction and product usability.

Joe Thomas
Development Engineer
Performance Technology Center/
Application Support Division

measurement interface. Examples include process wait reasons, current process priority, and file system information related to the process. This information is used to provide discrete data on process or program related items. While not part of the measurement interface, these data items provide important information that can be used to aid in performance management. Maturity of the platforms dictated how this data was to be retrieved; MPE provides direct routines while HP-UX data is retrieved using a proprietary library of routines. These secondary data sources coupled with the platform-specific measurement interfaces provide each HP GlancePlus model with all required data items for computing performance metrics.

Binding the measurement interface as the primary data source with the secondary kernel data structures provides a simple and streamlined approach to the data gathering required for performance metrics. Tying these data structures with relatively straightforward timing algorithms allowed the design team to meet the goal of creating easy-to-use performance diagnostic tools.

Processing Logic

A look at the internal logic of the HP GlancePlus programs shows a simple and efficient design. By having a core technology available, each model was able to leverage code from previously written software modules. The core section is an area of software procedures and programs that existed for other HP software engineering tools or products. The functionality provided by these tools is used to collect data from the measurement interface and kernel data structures for systemwide global and process-specific performance metrics. Some of the software modules that were reused had been written for the HP LaserRX family of performance products, and with very slight modifications they fit into the HP GlancePlus product easily.

Performance data at the global level includes systemwide information that can provide some insight into systemwide bottleneck conditions. Overall disk throughput, mem-

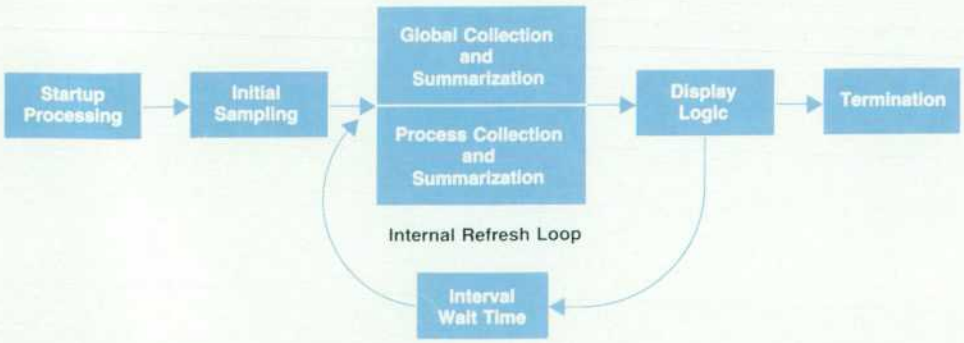


Fig. 6. *HP GlancePlus processing loop.*

ory manager use, and global file open rates are examples of global metrics. Process-specific data includes data items that are associated with a specific process for the current interval. This data provides a mechanism by which to view a process in detail and compare it to the bigger picture of global performance data. As shown in Fig. 6, the modules responsible for collecting and summarizing global and process-specific data are in a refresh loop in which they are continually sampling, processing, and sending the data to formatters for display.

Conclusion

Market acceptance of the HP GlancePlus family has been favorable. The paradigm that previously existed for performance diagnostic tools was one of expensive products that were difficult to use and understand. This has now changed as Hewlett-Packard customers can use these new inexpensive and easy-to-use offerings. A successful first step has been taken to provide intuitive performance diagnostic tools.

Acknowledgments

The HP GlancePlus products have been a true team enterprise from the beginning. The performance technology center's Gerry Wade and Paul Primmer were the original proponents of bringing performance tools to all levels of Hewlett-Packard customers. Technical assistance was provided from several sources in the PTC Lab. Software engineering contributions were made by Richard Santos, Jim Knoop, Doug Grumann, Terri Csete, and Marie Weston. Project management under Joe Thomas deserves a special mention; it was his focus on customer satisfaction and quality that kept the development team focused on providing quality tools for our customers.

The Performance Tool Quadrant

The HP performance technology center's development of performance tools is based on a core area surrounded by four distinct areas of performance management (see Fig. 1). This approach differs from other perspectives on performance products but allows HP to offer distinct offerings in each area, or quadrant, without sacrificing the focus or effectiveness of the specified tool.

The core area consists of common access routines that provide access to performance metrics regardless of the operating system platform. The commonality of data access provides a mechanism by which the separate tools from each of the quadrants shown in Fig. 1 can then access and manipulate data.

Each performance quadrant defines a different area of focus for performance management tools.

Application Optimization. This quadrant provides the tools necessary to fine-tune application programs. By analyzing where a program has spent its time, the user of application optimization tools may be able to identify flaws in logic which if corrected may result in more efficient code.

Diagnostic Tools. Tools in this quadrant provide the capability to see what is currently happening on the machine. The HP GlancePlus tools described in this article fit this criterion.

Resource Management. This quadrant provides tools that allow users to gain a larger perspective of how the system has been used by tracking multiple criteria over periods of time ranging from a few hours to several months. These criteria are items such as global CPU use, disk I/O use by disk drive, and process-specific information.

Capacity Planning. This quadrant provides tools that enable the performance planner to ask "what if" questions. Using sophisticated analytic modeling techniques, end users can project their future computing needs.

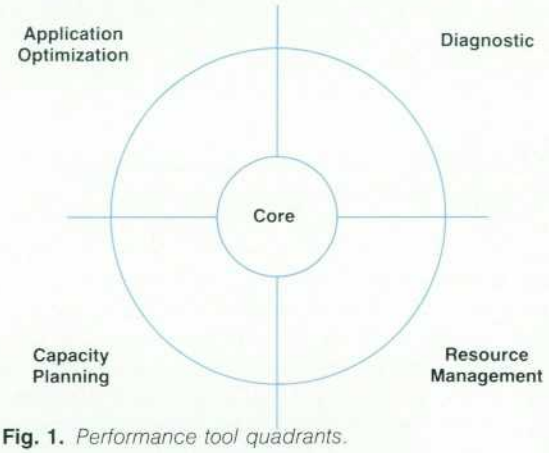


Fig. 1. *Performance tool quadrants.*

Using the core as a common technical focal point, each quadrant addresses a specific need. From each quadrant comes an offering of products that address that specific performance management need. The result is that Hewlett-Packard system users can pick and choose their performance toolset to fit their budget and functionality requirements.

Rex Backman
Development Engineer
Performance Technology Center
Application Support Division

Improving the Product Development Process

To define, design, and produce products and services that will be successful in the marketplace, it's necessary to understand the product development process and employ tools to measure and improve this process.

by Spencer B. Graves, William P. Carmichael, Douglas Daetz, and Edith Wilson

MANAGERS IN MARKETING, manufacturing, and especially research and development at HP are becoming more aware that they jointly manage a cross-functional process. Their people define and design a product and develop processes to manufacture and market that product (see Fig. 1).

Many HP divisions are working to improve this process. Their improvement efforts rely on concepts such as break-even time (BET), post-introduction product reviews, in-process project retrospective reviews, and quality function deployment (QFD). This paper briefly describes these techniques. Improvement efforts in the R&D departments of other organizations are described in references 1, 2, and 3.

Break-Even Time

Break-even time, or BET, has been identified within HP as a primary metric for the development process. A simple definition of BET is that it is the time from the initiation of a project to the point when all the design and startup costs have been recovered and the project is generating a net profit. Illustrative calculations are presented in Fig. 2. BET is a reasonable measure because it includes market acceptance, cost of production and sales, selling price, time value of money, and startup costs. Partial support for using a concept like BET in other companies can be found in references 4 and 5.

Measures such as break-even time may ultimately improve management's ability to predict profitability as early as initial product concept.* The data necessary to attempt to improve forecasts of profitability would be similar to PIMS (Profit Impact of Marketing Strategy).[7] The PIMS research program was initiated in 1972 for the specific purpose of determining how key dimensions of strategy affect profitability and growth. Since that time, 450 corporations have contributed data. PIMS data supports part of the famous Deming[8] chain reaction: higher quality tends to improve market share, to enhance ability to charge a higher price, and to decrease costs, all of which contribute to higher profit. By combining BET data with the results of post-introduction product reviews (see below), managers should be able to identify things they need to do to build on previous experience (like PIMs) and improve their ability to forecast profitability.

Break-even time is only one measure, and is evaluated in combination with other measures such as the estimated net present value of a subproduct line under various future investment assumptions. BET alone could push managers to do the wrong thing in some cases because it does not consider leverage or product success beyond the initial payback period. BET will not properly evaluate a product that is expected to open up a new market and whose ultimate success depends on whether future products are able to leverage that entry for higher profitability. BET also does not consider the cannibalism of a successful product by a follow-on. In some cases, the organization can protect market share by introducing a follow-on product early. In other cases the early introduction of a follow-on product may

*McHugh[6] describes how some engineers at HP's Roseville Network Division have compared actual with initial estimates of development time and have improved their ability to forecast development costs by using that information.

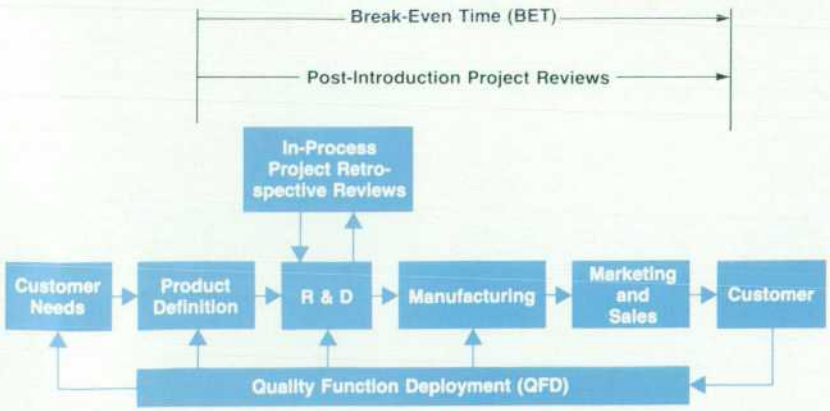


Fig. 1. *The product development cycle.*

	1986	1987	1988	1989
Investment in R&D ($K)				
By Year	40	200	40	0
Interest on Previous Investment (10%)	0	4	24.4	30.84
Balance	40	244	308.4	339.24
Net Cash Flow ($K) =				
(Revenue − Production Costs) by Year	0	100	200	200
Interest on Previous Income (10%)	0	0	10	31
Cumulative	0	100	310	541

BET = 3 Years

Fig. 2. *A simplified example of break-even time calculations. BET calculations are typically done in quarters. This figure presents annual figures.*

suggest a misdirection of R&D resources. No measure is perfect. For example, BET cannot be calculated if the project is canceled or if the product never recovers costs. To correct for this, some entities in HP compute a division-wide BET that includes canceled and unprofitable development efforts. Throughout HP, BET is being used to improve the R&D process within a division. Some of the possible causes of poor break-even time are summarized in Fig. 3.

To complement the BET metric, tools are needed for evaluating the strengths and weaknesses of the management of a project. Two such diagnostic tools that are intended to help evaluate and improve project management are post-introduction product reviews and in-process project retrospective reviews.

Post-Introduction Product Reviews

Kaoru Ishikawa[9] said, "Whenever I am asked to help introduce a total quality control program to a company, I choose for my case study one of the company's new product development projects that is full of problems." Ishikawa found that his clients learned a lot by studying their own experience in a structured way—and he provided the structure. The structure is essential to the learning. As Deming[10] explains: "Experience without theory teaches nothing. In fact, experience cannot even be recorded unless there is some theory, however crude, that leads to a hypothesis and a system by which to catalog observations."

Post-introduction product reviews and project retrospective reviews are vehicles for developing and improving the theoretical understanding that R&D managers use to guide their work. A study of this nature is being conducted at HP's Corporate Engineering. Ten factors have been identified that distinguish successful from unsuccessful projects (i.e., projects that met their initially stated objectives and those that didn't). Fig. 4 lists these ten factors and shows a comparison of six successful and six unsuccessful projects.

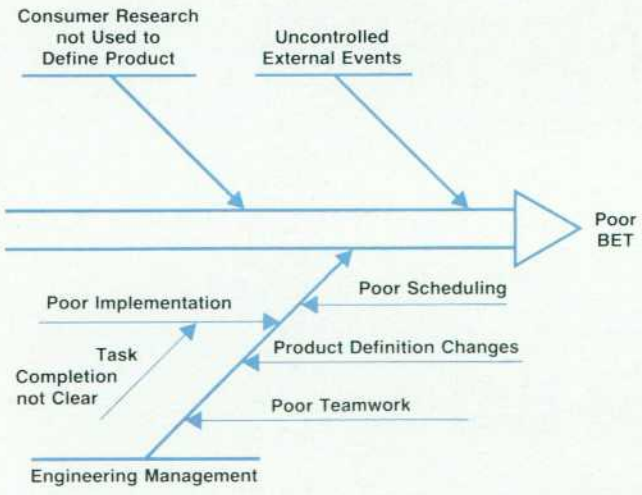


Fig. 3. *Possible causes of poor break-even time.*

From Fig. 4 it is obvious that in unsuccessful projects many factors were either omitted entirely or done inadequately. With information like that given in Fig. 4, engineering project teams and their management can ensure that the factors causing unsuccessful projects are addressed when creating a product definition. Of course this requires that a structure be in place that encourages early and complete consideration of these issues. Several success-versus-failure studies have been conducted. For example, researchers at the University of Sussex[11] paired 29 commercially successful innovations with an equal number of unsuccessful innovations. They identified five factors that distinguished successful from unsuccessful innovations:

- Understanding user needs
- Marketing
- Efficiency of the development effort
- Effective use of outside technology
- Project leaders with more seniority and authority.

By using any one of these five factors, they could identify the successful innovations in two-thirds of the pairs correctly, and by using the top three factors, they could classify all but two of the pairs correctly. Similar types of studies have been reported by other researchers.[12,13,14,15,16]

Among the projects summarized in Fig. 4, the most common problem was misunderstanding user needs. Although the sample size is small, it suggests that some benefit could be gained from more effective use of consumer research, and from using a technique such as quality function deployment to help manage the results of consumer research, competitive analyses, and the subsequent decisions.

The major point here is not the relative importance of the factors shown in Fig. 4. A similar study in another organization or at another time might identify different issues as being more important to that organization at that time. Each organization is unique. Data such as that reported here can be used to help an organization's members improve the way they manage their business. The hard part is not in collecting the data but in knowing what to collect and in becoming convinced that it is really worthwhile to regularly collect and use such data to make improvements. For this, top management support is criti-

	Successful Products						Unsuccessful Products					
	A	B	C	D	E	F	U	V	W	X	Y	Z
Understanding Users' Needs	X	X	X	X	X	X	○	X	○	○	○	X
Strategic Alignment and Charter Consistency	X	○	X	X	X	X	X	○	○	X	○	X
Competitive Analysis	X	X	X	X	X	X	○	○	X	X	○	X
Product Positioning	X	X	X	X	X	X	○	○	X	○	X	X
Technical Risk Assessment	X	X	X	X	X	X	X	○	X	○	X	○
Priority Decision Criteria List	X	X	X	X	X	X	X	X	X	○	○	○
Regulation Compliance	X	X	X	X	X	X	○	○	X	X	X	X
Product Channel Issues	X	X	X	X	X	X	○	○	X	X	X	X
Project Endorsement by Upper Management	X	X	X	X	X	X	X	X	○	X	X	X
Total Organizational Support	X	X	X	X	X	X	X	X	○	X	X	X

Legend:
X = Done Completely
○ = Done Inadequately or Omitted

Fig. 4 *Factors distinguishing successful from unsuccessful projects.*

cal. Experience with similar improvement efforts has established that people rarely have time for efforts like this without strong management support. The data collection and analysis is a minor expense when compared to the payback of bolstering the organization's performance in weak areas.

One discouraging aspect of post-introduction product reviews is that the cycle typically takes years. It is difficult to maintain the discipline to collect the data required when the analysis will take place so far in the future. Several HP divisions deal with this long-cycle-time issue by conducting similar evaluations at the end of each major phase in an R&D project. These are called in-process project retrospective reviews, or simply project retrospective reviews.

In-Process Project Retrospective Reviews

Experts discussing the management of product development have recommended reviews at various stages to help keep the project on track.[17,18] Some HP divisions use these reviews both to help keep projects on track and to improve the ways in which projects are managed. At one HP division, at the end of each major phase of an R&D project, each member of a cross-functional project team completes a new product development and introduction follow-up survey. The participants are asked to describe what went well, what went poorly, and what they would like to see different in the future. A sample questionnaire is shown in Fig. 5. The results are tabulated and summarized and presented to the R&D management. Fig. 6 summarizes the findings from four different projects.

This particular analysis was of great help to the R&D management. For example, the biggest problem in Fig. 6, deficient project planning and scheduling, motivated the management to make better use of its existing PERT (Program Evaluation and Review Technique) planning tools. PERT is a detailed project planning and scheduling tool. Until this survey was conducted, the R&D managers had little to tell them whether they had made an adequate PERT chart—or even needed one. With feedback in the form of Fig. 6, R&D managers learned why they needed to use PERT and how to know if they had done it correctly. For example, they found that many problems arose from misunderstandings in deciding when a task was completed. From this, R&D managers concluded that tasks should be defined in terms of specific deliverables. Similarly, they found that they tended to have more problems with larger rather than smaller tasks. In looking at this, the R&D managers decided that tasks on a PERT chart that required 80 hours or more should be decomposed into smaller tasks with intermediate

```
0) NAME (optional): _____
1) POSITION (during development/introduction):
   ___Engineer   ___Supervisor   ___Manager   ___Other
2) FUNCTIONAL AREA and DEPARTMENT (during development /
introduction):
   ___R&D  __Marketing
   ___Manufacturing   __Quality  __Finance
3) Based on your involvement, please rate each of the
following aspects of the _____ product development/
introduction effort on a scale of 1 to 5, where
        1---------2---------3---------4---------5
    very poor   poor    acceptable  good    very good

   ___How well were you informed of the overall
      development/introduction plan and the role that
      you were expected to play in it?
   ___How much of an opportunity did you have to influence
      the portion of the development/introduction plan
      that related directly to your role and
      responsibilities?
   ___How well did you understand which deliverables you
      were responsible for, who the customers were for
      these deliverables, and what were their specific
      needs?
   ___How adequate was the amount of time allotted to you
      to complete your activities?
   ___How adequate was the quantity of resources allotted
      to you to complete your activities?
   ___During the course of the development/introduction,
      how well were you informed of the status of other
      activities that could affect your ability to
      complete your own activities as planned?
   ___How effectively were risks and potential obstacles
      identified and dealt with over the course of the
      development / introduction?
   ___How well was the development/introduction effort
      coordinated and managed over time?
   ___Overall, how satisfying of an experience was your
      involvement in the _____ development/introduction
      effort?
   ___How useful was a reference such as a new product
      development manual over the course of your
      involvement?
4) What was the most positive aspect(s) of this new product
development / introduction for you?
   _____________________________________________________
   ...
5) What recommendation(s) do you have for building more
teamwork and personal satisfaction into the development /
introduction process?
   _____ ...
6) What was the most negative / frustrating aspect(s) of
this new product development / introduction for you?
   _____ ...
7) What change(s) to the new product development /
introduction process would have helped you to use your time
more effectively / efficiently?
   _____ ...
8) What additional comments, concerns and recommendations do
you have with regards to the new product development /
introduction process?
   _____ ...
```

Fig. 5. *Questionnaire for in-process project retrospective reviews.*

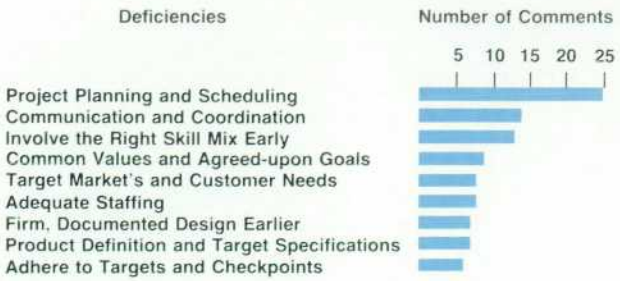


Fig. 6. *Deficiencies identified by in-process project retrospective reviews.*

deliverables and time estimates of less than 80 hours each.

After a while, the R&D managers had access to an informal library of PERT charts on completed projects which they could use as a starting point for new projects. This tended to improve the accuracy of planning while reducing the work of doing it. (This is an example of the general principle that higher quality often costs less.)

The product development process as practiced at this division has several distinct phases. A procedures manual explains each phase and what requirements must be met to proceed to the next phase. Without data like that shown in Fig. 6, it is hard to know when something in the procedures manual should be revised. The retrospective surveys gave the R&D managers data showing where the process was deficient and which sections of the manual should be updated. This data transformed the task of updating the manual from being thankless, vague, and infinite to being important, specific, and finite.

Some of the other HP divisions that have attempted similar use of project retrospective reviews are achieving similar results. It was also found that an essential element for the success and continued use of this type of tool is leadership and encouragement from top management.

Quality Function Deployment

Quality function deployment (QFD) is a methodology that was developed to ensure that the voice of the customer drives the definition, design, and production of products and services. The QFD approach requires cross-functional cooperation—principally among the marketing, R&D, and production functions—to collect adequate input on customer wants and competitive standing and to translate this information effectively into product design and production processes. This cross-functional cooperation is made more efficient and effective through the use of the so-called "seven management and planning tools."[19,20] These tools, which include special types of diagrams, charts, and matrices, facilitate communication and consensus.

One of these seven tools is a class of matrix diagrams that are used in QFD to guide the planning process and to store the results of judgments and calculations that have been made. An example is provided in Fig. 7. In this figure, the *importance of want* for each of the customer wants should be derived from customer research. These importance weights are multiplied by the corresponding coded relationship strengths in each column and summed vertically to give the column importance weights. For example, in the first column, we compute $10 \times 9 + 8 \times 1$ to get 98 as the column importance weight. The degree of technical difficulty entry records team judgment about the relative difficulty (including cost, time, need for breakthroughs, etc.) of achieving the different technical objectives. The roof on the "house of quality" is called the correlation matrix because it portrays correlations between the different technical parameters. For more information on QFD, see references 21, 22 or 23.

QFD matrices and tables provide explicit documentation of the link between customer wants, competitors' solutions, and the characteristics of the product under development. This explicit link extends through product specifications to the processes by which the product will be produced and tested. The QFD tables or matrices are easy to understand and easy to use to facilitate communication between members of a team. One result is that problems tend to be more effectively anticipated and resolved throughout product development. Design engineers have used the QFD documentation to trace a product specification to the words recorded from customers in the consumer research phase. This access to the voice of the customer has helped engineers make more intelligent trade-offs between alternative solutions to the design problem.[24]

In the first phase of QFD, the product planning phase, one key task is to obtain importance ratings for the various customer wants (see the importance of want column in Fig. 7). The importance ratings are usually obtained through a variety of sources, tools, and techniques including customer surveys, complaint letters, choice modeling, and other types of consumer research. Usually the project team clusters the customer wants into groups of related items before trying to establish the importance rating or ranking for the customer wants.

The QFD team then generates a list of measurable, controllable characteristics (often called engineering characteristics) that relate to the customer wants they seek to satisfy. The relationships between these controllable factors and the customer wants are documented in a matrix as in the upper center of Fig. 7. The project team places symbols or numbers representing the strength of each relationship in appropriate cells of the matrix (see R_{ij} in Fig. 7).

This information is combined with the results of competitive benchmarking, which is done from both the customer perception and technical product analysis standpoints. Numerical assessments of the relative importance of different engineering characteristics are made and a target value is determined for each characteristic. In essence, the target values for the engineering characteristics define the desired product.

At this point the project team focuses on design and, particularly, exploring alternatives to find a design that achieves the target values of the engineering characteristics. If trade-offs have to be made, precedence is given to achieving the target values for the engineering characteristics that the team identified from the QFD chart as most important.

In one of the more widely used QFD approaches in the U.S., the details of the winning design are elaborated in a cascading series of three additional QFD phases: design planning (parts deployment), process planning, and production planning.[25] In the design planning phase, the key

QFD matrix displays engineering characteristics versus the parts that the design specifies. For process planning, a matrix is developed that shows the relationships between parts and the processes that will be used to obtain them. Finally, in the production planning phase, a table or matrix is developed that shows how the processes will be controlled.

One QFD matrix or table can sometimes be used for several related products. If consumer research has identified different market segments with different priorities for the various customer wants in the different segments, then the same table can be used to define different products for different segments. It is a straightforward calculation to translate changes in importance of customer wants into changes in importance of engineering characteristics. Changes in the importance of engineering characteristics or desired target values affect the evaluation of each specific product concept.

QFD has been used in the design not only of hardware but also of software[23] and services.[26]

QFD does not eliminate the need for managers to use other techniques to improve their project management. Post-introduction product reviews and in-process project retrospective reviews, as described above, help managers charged with all phases of product development improve their understanding and effective use of techniques such as QFD and PERT. For example, managers need to know if they are adequately investing in consumer research. Post-introduction product reviews can help answer such questions.

Conclusion

The development of successful new products is a process that itself produces a product—namely, a design for a product and for its associated manufacturing process. This paper has suggested that the product development process can be improved by using certain tools and management

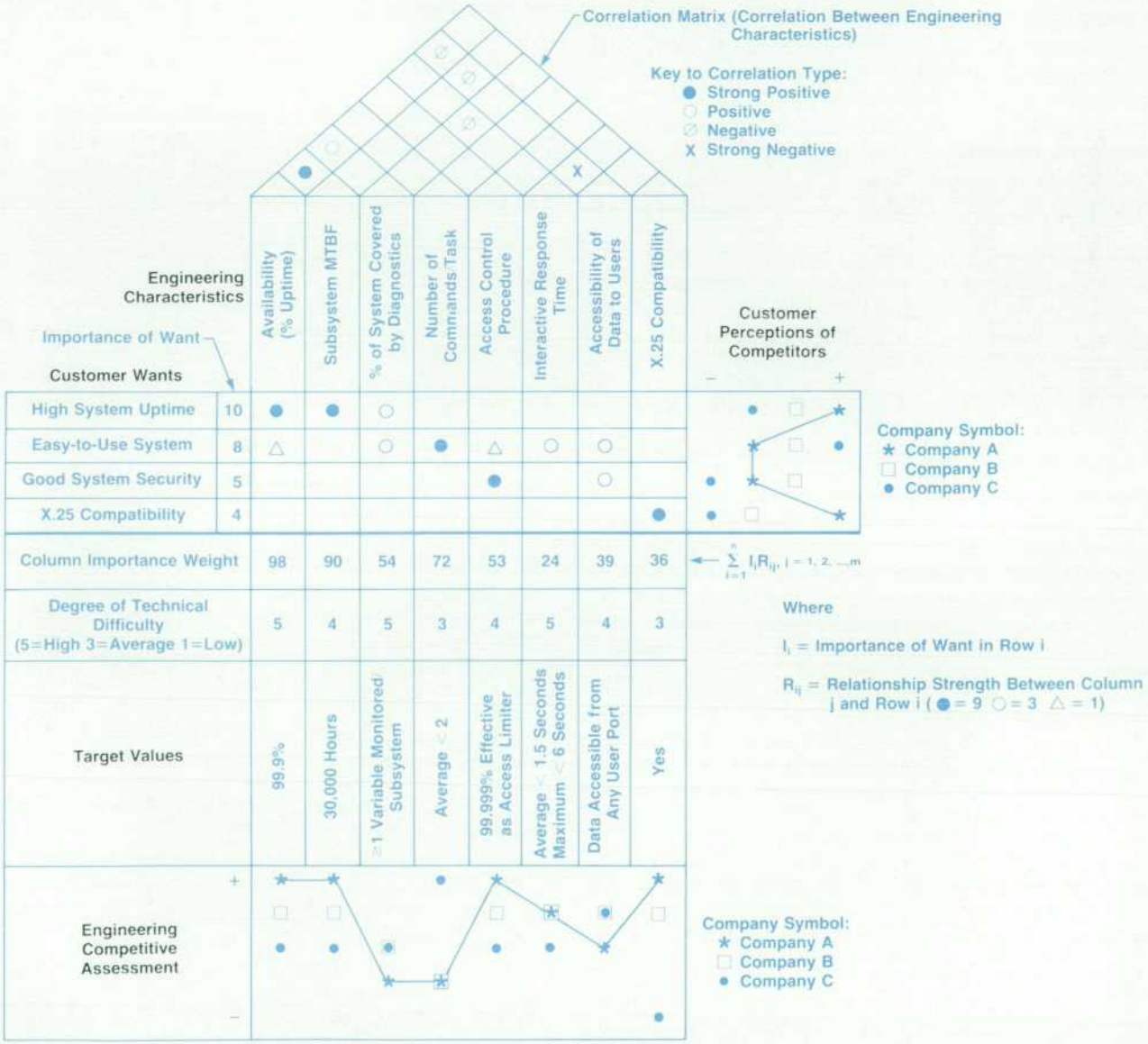


Fig. 7. *Example of a QFD house-of-quality matrix.*

processes for:

- Evaluating the performance of the process (break-even time)
- Studying the process and its results (post-introduction project reviews and in-process project retrospective reviews)
- Revising the process to achieve better cross-functional communication, a good understanding of customer wants and competition, and a clear view of interrelationships so that important factors or steps are not inadvertently left out (QFD).

The use of these processes and the data they generate has resulted in major improvements in R&D productivity in parts of HP. One crucial lesson from HP's experience is that top management leadership is vital to the successful use of the techniques described in this report. HP's top leadership has been effective in promoting the use of BET and QFD throughout the company. Division general managers and R&D managers have asked for and used project retrospective reviews. Similar efforts that lacked such support have not been successful.

In the future we expect that expanded use of techniques such as those described above will improve the competitive position and possibility of success for organizations that use them.

Acknowledgments

The authors wish to thank Val Nereo and Khushroo Shaikh for useful discussions contributing to this paper.

References

1. F. Takei, "Productivity Improvement in Engineering Work—the 'EPOC' Campaign in a Japanese Company," *Engineering Management International*, Vol. 1, 1981, pp. 23-28.
2. R. B. Cooper, et. al, "Applying Quality Assurance to R&D Projects," *Quality Progress*, Vol. 23, no. 7, July 1990, pp. 21-26.
3. G. W. Roberts, "Wipe out R&D Waste," *Quality Progress*, Vol. 22, no. 1, January 1989, pp. 54-57.
4. K. B. Clark and T. Fujimoto, "Lead Time in Automobile Product Development Explaining the Japanese Advantage," *Journal of Engineering and Technology Management*, Vol. 6, 1989, pp. 25-58.
5. Y. Kuwahara and Y. Takeda, "A Managerial Approach to Research and Development Cost-Effectiveness Evaluation," *IEEE Transactions on Engineering Management*, Vol. 37, no. 2, May 1990, pp. 134-138.
6. J. McHugh, "Can T, Q, C, R, & D Be Combined in a Meaningful Way?" (unpublished), 1989, Hewlett-Packard Roseville Networks Division.
7. R. D. Buzzell and B. T. Gale, *The PIMS Principles*, The Free Press, 1987, pp. vii and 81-87.
8. W. E. Deming, *Out of The Crisis*, Massachusetts Institute of Technology Center for Advanced Engineering Study, 1986.
9. K. Ishikawa, *What Is Total Quality Control*, Prentice-Hall, 1985, p.81.
10. W. E. Deming, *op. cit.*, p. 317.
11. *Success and Failure in Industrial Innovation: Report on Project Sappho by The Science Policy Research Unit*, University of Sussex Centre for the Study of Industrial Innovation, 1972, pp. 4-9.
12. M. L. Dertouzos, R. K. Lester, and Robert M. Solow, *Made in America*, MIT Press, 1989, pp. 70-72.
13. E. von Hippel, *Sources of Innovation*, Oxford University Press, 1988.
14. E. W. Larson and D. H. Gobeli, "Significance of Project Management Structure on Development Success," *IEEE Transactions on Engineering Management*, Vol. 36, no. 2, May 1989, pp. 119-125.
15. G. L. Lilien and Eunsang Yoon, "Determinants of New Industrial Product Performance: A Strategic Reexamination of the Empirical Literature," *IEEE Transactions on Engineering Management*, Vol. 36, no. 1, February 1989, pp. 3-10.
16. K. Brockhoff and A. K. Chakrabarti, "R&D/Marketing Linkage and Innovations Strategy: Some West German Experience," *IEEE Transactions on Engineering Management*, Vol. 35, no. 3, August 1988, pp. 167-174.
17. G. Vincent, *Managing New-Product Development*, Van Nostrand Reinhold, 1989, pp. 106-108.
18. J. A. Bradbury, *Product Innovation: Idea to Exploitation*, John Wiley & Sons, 1989, pp. 20-44, 149.
19. M. Brassard, *The Memory Jogger Plus*, Goal/QPC, 1989.
20. S. Mizuno, *Management for Quality Improvement*, Productivity Press, 1988.
21. L. P. Sullivan, "Quality Function Deployment," *Quality Progress*, Vol. 19, no. 6, June 1986, pp. 39-50.
22. B. King, *Better Designs in Half the Time*, Goal/QPC, 1987.
23. K. Shaikh, "Thrill Your Customer, Be a Winner," *HP Software Engineering Productivity Conference*, August 1990, pp. 393-402.
24. M. Thompson and K. Chao, "Quality Function Deployment and HP IVI," *Hewlett-Packard Journal*, Vol. 41, no. 5, October 1990, pp. 9-10.
25. R. Hauser and D. Clausing, "The House of Quality", *Harvard Business Review*, Vol. 66, no. 3, May-June 1988, pp. 63-73.
26. M. Ofuji, T. Noda, and J. Ogino, "Quality Deployment for the Service Industry," *Quality Deployment: A Series of Articles*, Goal/QPC, 1987.

DSEE: A Software Configuration Management Tool

HP Apollo provides a software tool that helps to manage development and maintenance of the many components that make up large-scale software systems.

by David C. Lubkin

THE DOMAIN SOFTWARE Engineering Environment (DSEE) is a configuration management system that is useful for managing large, complex software development and maintenance projects.* DSEE runs on the Apollo Domain operating system (Domain/OS) and is in use at over 6000 sites world-wide, including HP Apollo. We estimate that DSEE is responsible for managing over one billion lines of code. DSEE is also encapsulated in the new Softbench[1] technology.

DSEE is designed to deal with the particular problems of large-scale software development. Some of these problems are working on multiple development efforts in parallel, maintaining existing releases while working on new code, coordinating code and documentation, reducing the time it takes to build a program, and knowing precisely which tools and versions of source code were used to create a binary file. DSEE unifies development and maintenance for large-scale systems (the largest so far is 11,000,000 lines). It supports project development in a distributed computing environment, and works with any programming language or tool. DSEE is highly reliable, thanks to transactional file and database operations, store-and-forward message passing, audit trails, and time-proven fault recovery procedures. The latest version of DSEE, version 4, makes the configuration management capabilities of DSEE available to a variety of non-Apollo systems.

The major subsystems that make up DSEE include:

- History Manager. The history manager stores, controls and tracks source code evolution, and enforces concurrency control.
- Task Manager. The task manager plans and tracks the low-level steps taken toward achieving some high-level activity—for example, tracking the modules changed to implement a new feature.
- Monitor Manager. The monitor manager tracks people-oriented, rather than build-oriented dependencies. For example, it notifies users when alarms they set are triggered because components of interest are modified.
- Configuration Manager. The configuration manager is concerned with system building. It also manages derived modules such as binary files or formatted text.
- Release Manager. The release manager archives executables, and optionally, it can take a snapshot of the tools, elements, and files used. An element is a file under DSEE source control.

*DSEE is also useful for small and simple projects.

This article provides an overview of these subsystems. There is also a discussion of DSEE version 4 features that allow non-Apollo users to have access to DSEE capabilities.

Version Control

The history manager stores, controls and tracks source code evolution. It enforces concurrency control and security, and keeps a complete chronological audit trail of library modifications. DSEE views the evolution of a file as a directed acyclic graph (see Fig. 1). Any number of people may work on the same file concurrently by making new branches. Their work may be kept separate or merged together with a three-way merge command (Fig. 2). The trunk is also considered to be a branch, and it is usually referred to as the *mainline*.

In merging versions on two branches together, DSEE compares the two versions with their nearest common ancestor. If work in a small section of the file changed one branch

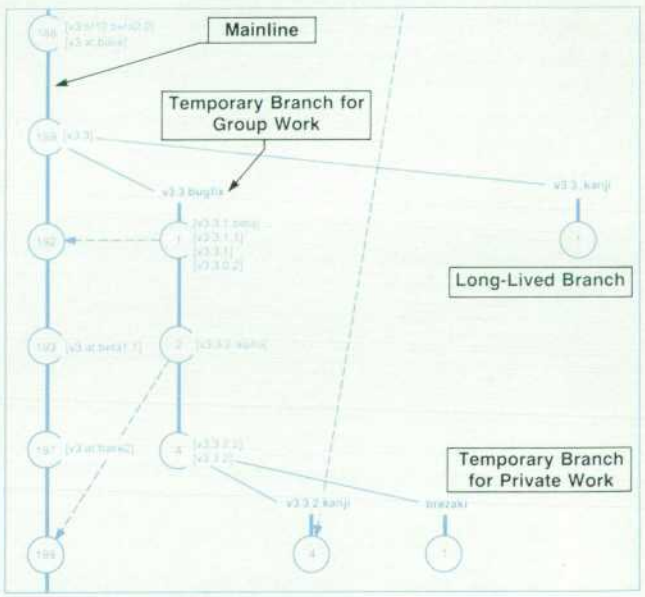


Fig. 1. *A portion of a graph showing the evolution of a typical module in DSEE. The numbers in the bubbles represent version numbers and the bracketed text represents version names. The double solid lines represent mainlines, the single solid lines represent branches, and the dashed lines represent merges. Without the callouts, this screen is provided in a window via the DSEE* show derivation *command.*

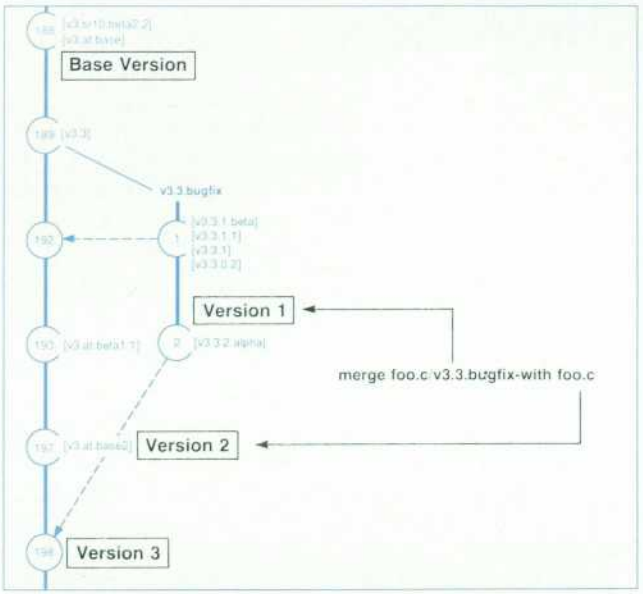


Fig. 2. *The execution of the three-way merge command results in merging version 1 and version 2 and creating version 3.*

relative to the ancestor—and didn't change the equivalent location in the other branch—DSEE automatically selects the new modifications. If both files have changed relative to the ancestor, DSEE presents both sets of changes in a window. Users may pick one or both sets of changes, which are then copied to a window containing the merged file. The user can further edit the file in the merged window. Fig. 3 shows the DSEE screen during a merge operation.

Branches have names like v3.3.bugfix or AddBuzzwords. Versions or generations on a branch have version numbers, but can also be given one or more names (e.g., version 1 of v3.3.bugfix in Fig. 2 has four names). The evolution of a module can get very complicated. That's why DSEE has a command called show derivation that allows the user to display graphically the structure of a module's evolution, including branch points. Figs. 1 and 2 without the callouts are examples of what the show derivation command provides. The show derivation command also has options to prune portions of the displayed structure.

Versions are stored together efficiently. DSEE's interleaved forward deltas and compression allow 50 to 200 versions of source code to be stored in the same space as two ordinary text copies. Binary files can be stored under DSEE, but they are not stored as compactly as text files. Any version of either text or binary files can be read in a single pass through a DSEE history file.

Transparent Access to Versions. Because of the Domain operating system's typed file system,** any DSEE element can be read directly, without checking it out from the library, either from an Apollo system or across a transparent remote file system like NFS, DECnet, or Domain/PCI. This lets ordinary applications like cc and troff operate on objects under DSEE control without modification.

DSEE files are type case_hm. When asked to read a pathname that ends with a DSEE file's name, the case_hm type manager returns the most recent version on the mainline. If the user wants to read some other arbitrary version, the version can be explicitly named or implicitly associated with the original file. If the pathname continues beyond a DSEE element, the case_hm manager interprets the remainder of the pathname as a sequence of branches and generations. This extended naming does not work across all transparent remote file systems. Some remote users have to use the DSEE fetch command, which refers to versions with a similar syntax. All users can use the read command, which puts any version into a read-only X11 window.

The following examples, which are derived from Figs. 1 and 2, illustrate the different types of pathnames that are used to access different versions of modules in DSEE.

- Most recent version of the main line of descent
  foo.c
- Specified version number on the main line of descent
  foo.c/193

**Under Domain/OS, each file has a type, and each type has an associated manager that performs I/O for that type.

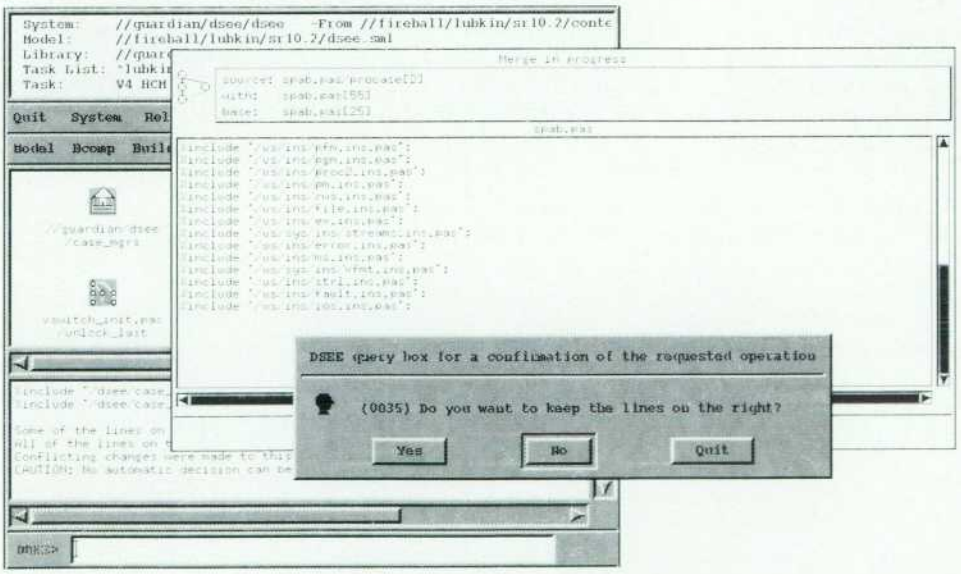


Fig. 3. *A screen showing how DSEE displays the status of a file while it is being merged.*

- Most recent version on the specified branch
  foo.c/v3.3bugfix/brezak1
- Specified version number on the specified branch
  foo.c/v3.3bugfix/2
- Specified version name
  foo.c/v3.3bugfix/[v3.3.2.alpha].

Users can also implicitly refer to a collection of versions, like those used to build some binary, or a specific named version of each of their files. In Domain/OS, the user can create a shell that is bound to that collection. Whenever files are read from that shell, the case_hm manager will access the particular versions that had been specified.

In the case where DSEE files are accessed from a non-Apollo computer, the mechanism that binds versions to shells does not work across most transparent remote file systems. Therefore, the user has to access the desired files through a special directory whose presence indicates the version the user wants. In the following example, vq6cro6b is such a directory. Requests to read below it are handled by the vswitch type manager, which reestablishes the versioning context and then hands the request on to the case_hm type manager.

Domain/OS pathname:

```
emacs //fireball/mylibrary/utils.c
```

HP-UX pathname:

```
emacs /apollo/zorro/sys/nodedata/tmp/vq6cro6b/mylibrary/utils.c
```

When emacs on HP-UX opens

```
apollo/zorro/sys/nodedata/tmp/vq6cro6b/mylibrary/utils.c,
```

the request is passed to the HP-UX operating system. HP-UX recognizes that /apollo is an NFS mount point and issues a series of NFS requests that should result in the desired file being opened. Eventually, the remainder of the pathname

```
zorro/sys/nodedata/tmp/vq6cro6b/mylibrary/utils.c
```

is passed to the Apollo NFS server which submits the request to the Domain/OS I/O system. When the I/O system encounters the object vq6cro6b, which is file type vswitch, it loads the vswitch type manager. The vswitch type manager uses the rest of the pathname and information stored in the vswitch object to establish the proper versioning context, and then passes the request to the case_hm type manager.

Note that on Domain/OS, the file system of any individual Apollo computer is accessible as a subdirectory of //. Similarly, NFS users can mount the entire Apollo network at one mount point (/apollo in our example). These two mechanisms for implicit reference to versions are integral parts of the configuration manager described later.

Project Management

Project management capabilities are provided in DSEE by the task manager and the monitor manager. The DSEE task manager plans and tracks the low-level steps taken toward achieving a high-level goal, like fix bug #1332 for Lockheed. It automatically records DSEE actions made on behalf of the goal, like builds or replaces. It can also be used to record non-DSEE tasks to do or that have been done, like review Chapter 5 of the new doc. For tracking recurring activities, new tasks can be created using a task template facility. A task template facility is a set of services in DSEE to manage task templates. A task template is a template for a new task. For example, suppose the task system is used for bug tracking. A task template can be created that contains the standard set of steps used when tracking a bug. When there is a new bug, the task can be initialized from the template.

In the short term, task records can be used by team members and managers to track progress toward goals. Later on, they provide a historical record that can be useful for resolving similar problems and for helping new team members learn what has been done and what steps were taken.

All of a user's tasks are collected on a personal tasklist. For example, the following tasks might be on tasklist foo.

```
7: Created 16-Jan-1990 10:50 in library
   "//helpless/case/from_dean/case_cm":
   NFS/vswitch performance improvements

6: Created 8-Jan-1990 12:27 in library
   "//helpless/case/from_dean/case_mgrs":
   V4 HCM bugs
```

The monitor manager tracks people-oriented dependencies. The user describes what DSEE elements to monitor, who can or cannot activate the monitor, and what should happen when the monitor is activated. The monitor then remains dormant until it is triggered by changes to the elements being monitored. When the monitor is triggered it may:

- Send a mail message to one or more users
- Add a new task to some user's tasklist
- Send an alarm window to anyone on the network
- Execute an arbitrary shell script.

One example of monitor use is that our technical writer keeps a monitor on the module that handles DSEE's command syntax. This way the writer does not need to rely on the DSEE developers to know when changes are made that affect the documentation.

Some customers use monitors to convert from RCS or SCCS to DSEE. Typically, customers are cautious and only convert one group at a time. If that group had been sharing their SCCS files with other groups, they could set monitors on all their DSEE elements. When a monitor is triggered by the creation of a new generation of a file, it would run a shell script that updates the corresponding SCCS file.

Configuration Management

The central concern of DSEE configuration management is to provide a reliable and complete permanent record of what sources, tools, and translator options were used to build a program. This is one of the primary characteristics that distinguishes DSEE from other configuration management tools. DSEE uses this precise description to improve personal and group productivity, but never at the expense of correctness. Users can share binaries with each other

when the descriptions of what they want to build match. Their work is kept isolated when they don't match. DSEE will minimize the number of compiles required to build a program and how long they take.

The configuration manager separates the descriptions of the structure of the user's program (i.e., the system model) and the specific versions the user wants to build at a particular time (i.e., the configuration thread).

System Models. A system model is a set of files that describe the static structure of a program or system. It is written in a special declarative and block-structured language. System model components may either be DSEE elements or external, nonelement files. For each component of the system, the model expresses:

- Source dependencies (i.e., what the source includes)
- Translation rules* (i.e., how to compile the component)
- All possible translation options (e.g., -debug, -opt)
- Tool dependencies (i.e., tools required for the translation)
- Subcomponents that must be built first.

The model language supports conditional constructs that may be triggered by target system characteristics.

A utility is provided with DSEE to help the user create a system model. It determines the include file interdependencies by looking at the source code and creating a skeletal system model. The user will still have to tell DSEE how to build the program and fine-tune the model. Fig. 4 shows a small portion of a system model description.

Threads. Configuration threads describe which versions and translator options to use when building each component in the system model (see Fig. 5). By changing threads, users can quickly switch among new development projects, fixing bugs in old releases, and making special versions for particular targets or customers.

Configuration threads may be:

- Explicit (e.g., use version 5)
- Dynamic (e.g., use the most recent version)
- Referential (e.g., use the same version used in rev2)
- Contextual (e.g., use x.h[7] for P, otherwise x.h[6]).

Fig. 5 illustrates how two engineers can build similar, but not identical configurations of the same system by using different configuration threads. Engineer A is using a thread that tells the configuration manager to follow these rules:

1. For the file alpha, use the most recent version on the mainline.
2. For every other file, use the version named V2 (version 3 from beta and version 6 from gamma).

The mainline for file gamma is reserved in engineer B's working directory. Engineer B uses a configuration thread that tells the configuration manager to follow these rules:

1. For each file used in the system, if it is reserved, use the version in the working directory.
2. Otherwise, use the version named V2 for each file (version 3 from beta and version 8 from alpha).

There is also a model thread that applies to system models. Model threads are usually stored within a DSEE library so that users have a precise description of the structure of a program at any time in its history. They are often composed of several files, or model fragments. The model thread describes which versions of model fragments and conditional compilation options (targets) to use when interpreting the system model. One use for targets is to switch between different hardware platforms or operating system versions. At Apollo, the system model for the Domain/OS operating system supports over a dozen different hardware variants.

Bound Configuration Threads. DSEE combines the system model and configuration thread to make a bound configuration thread (see Fig. 4). It is a permanent record of the options, versions of sources, and tools used to build a component. Bound configuration threads are kept along with derived objects and binaries in special directories called pools. When the user types build, DSEE looks in the pools for bound configuration threads that match the components the user asked to build. DSEE keeps a number of earlier builds in the pool and attempts to reuse them. If it can't, and the pool is full, the least recently used build of that component is purged to make room for a new one. This lets the users share binaries, thereby reducing the number of builds to perform. On the other hand, because the bound configuration thread gives such a clear picture of what makes up a particular binary, only binaries that match the users' needs are shared.

DSEE lets the user declare that some compiler options or source versions are critical and that others are noncrit-

*Translation rules are stylized shell scripts that describe how to build a system model component. They are exactly like shell scripts except that in place of many pathnames and compiler options, there are macros.

System Model		Configuration Thread		Bound Configuration Thread
model my_program : element lexerr.c = host_type hp-ux; depends_source x.h @ lib1; y.h @ lib2; translate cc -o %result %source %done end of lexerr.c; :	+	-reserved [beta] -when_exists -use_options -debug [release_3]	=	lexerr.c[19] x.h[7] y.h[38] -debug compiler v6.2 :

Fig. 4. *An excerpt from a system model description and its relationship to a configuration thread and the resulting bound configuration thread.*

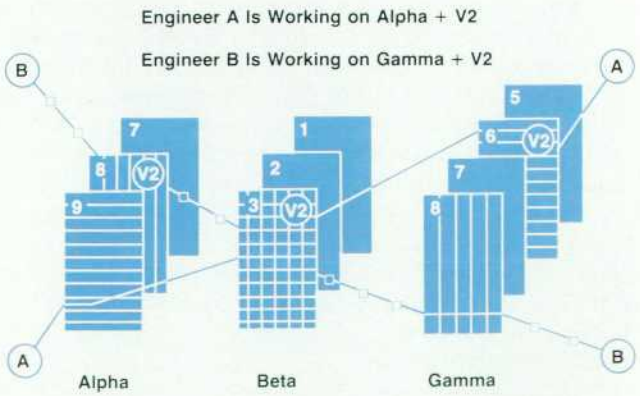


Fig. 5. *An example of configuration threads. Alpha, beta, and gamma are files and the numbers represent different versions of the files.*

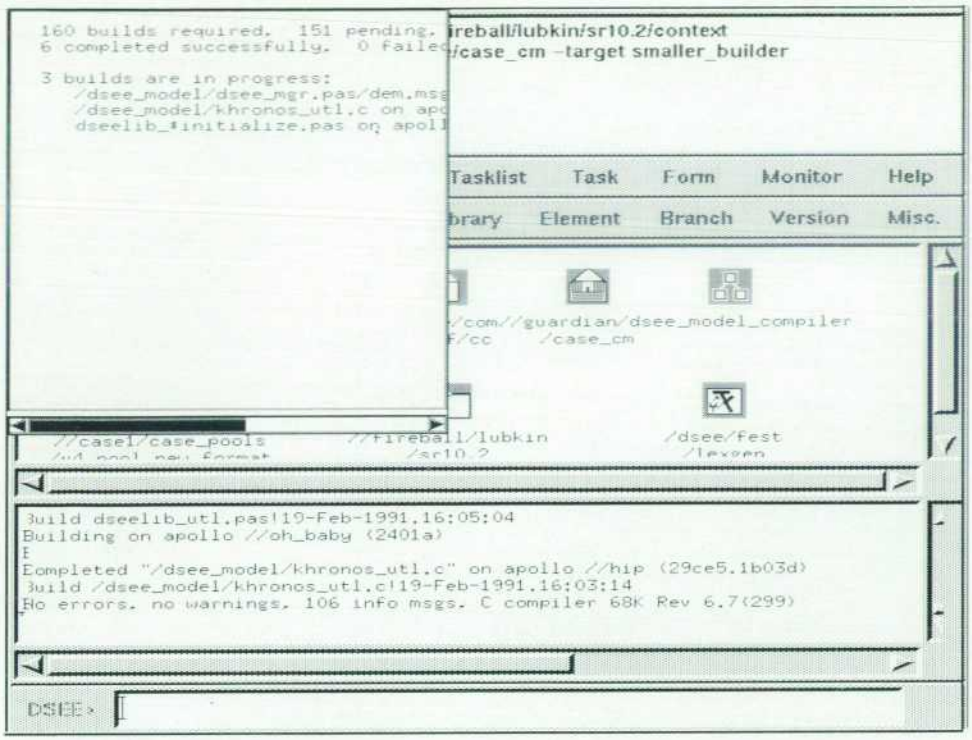


Fig. 6. *Display showing parallel builds in progress.*

ical. If a critical dependency changes, DSEE will rebuild the module. If a noncritical dependency changes, DSEE will record what compiler option or source version was used, but it will not rebuild the module. There are also override mechanisms through which the user can force DSEE to rebuild even though the bound configuration thread doesn't call for a build, or not to build when DSEE thinks it has to.

Before DSEE starts building it sequences the required builds according to their interdependencies. As the user's translation script executes, it reads the desired version of each DSEE element directly out of its library. After the program is built and debugged, the user can make a release, or permanent snapshot of the bound configuration threads, binaries, sources, tools, and system model that make up the program. With this feature, when customers report a bug two years later, the engineer repairing the code can rebuild the same bits, plus the fix.

Version 3 of DSEE added support for parallel building on more than one Apollo computer at the same time. DSEE tracks the CPU utilization of candidate build computers, and starts builds on the most lightly loaded machines. Fig. 6 shows parallel builds running on several machines. The performance improvement provided by parallel building can be substantial. For example, the time to run the Ada validation test suite drops from 80 hours to 17 hours when it is built in parallel.

If an executable is compiled with an arbitrary collection of stdio.hs, it might take weeks to track down a problem caused by inconsistency in the include file. DSEE ensures that all builds use the same set of tools and system include files by building relative to a reference directory.

Heterogeneous Configuration Management

Many users want to use DSEE to develop software for non-Apollo platforms. To fill this need we generalized configuration management to handle heterogeneous configuration management. Version 4 of DSEE supports other systems that are based on the UNIX* operating system. Fig. 7 shows the setup for heterogeneous configuration management.

DSEE can be used to develop software for non-UNIX targets as well; it's just more involved. Since a translate rule is an arbitrary shell script, it can copy sources over to a non-UNIX target, invoke a shell, and copy the results back into a pool.

Host Types. To handle heterogeneous configuration management, each system model component has a host type,

*UNIX is a registered trademark of UNIX System Laboratories in the U.S.A. and other countries.

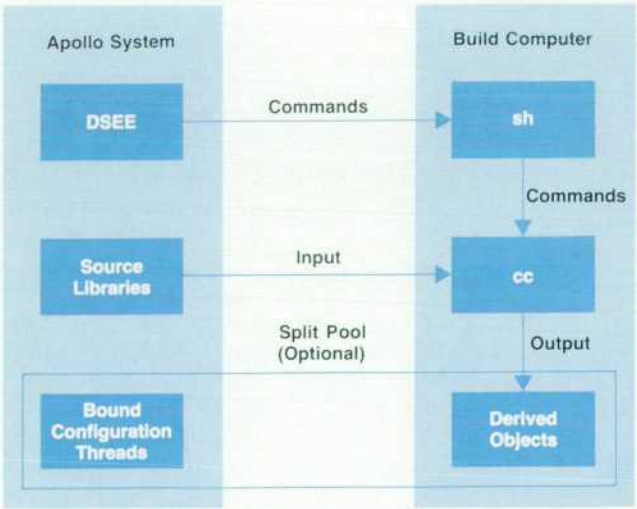


Fig. 7. *The architecture for DSEE in a heterogeneous environment.*

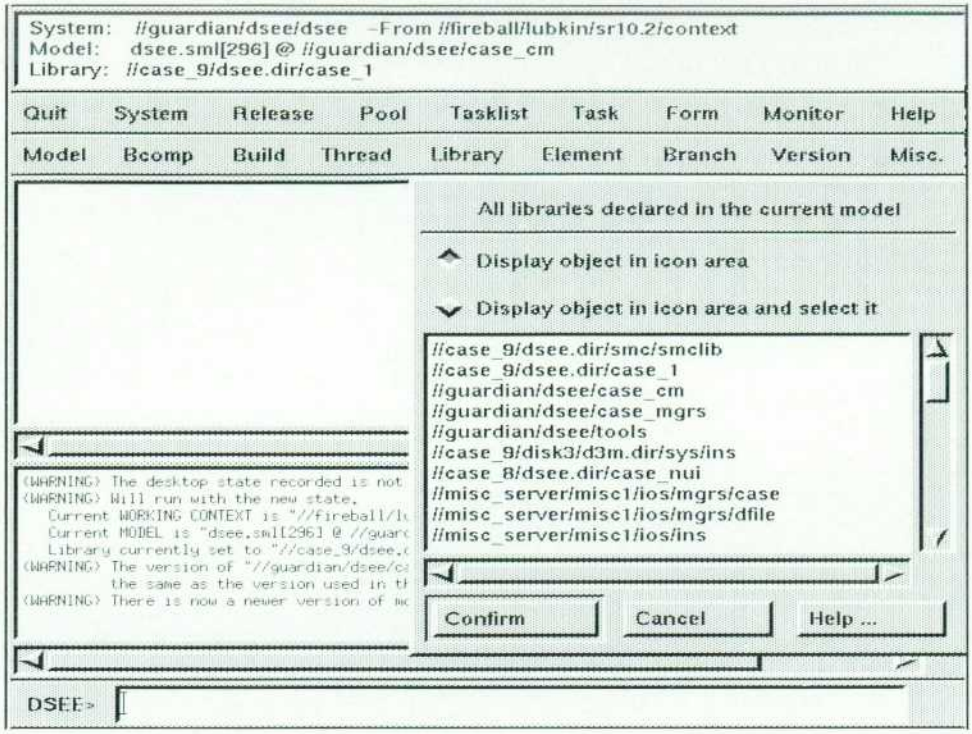


Fig. 8. *DSEE graphical user interface.*

which is a user-defined text string that describes the class of machine the translation should run on. It reflects the distinctions between machines that the user wants to use. Host types can be used to make very fine distinctions, like DN4500 with FPA board, or broad groupings like UNIX. The UNIX keyword might be used for translators that produce portable output, like yacc or troff. The only host type that is predefined is apollo. The following is an example of a user-defined host table.

```
#  /sys/dsee/dsee_config/hosts
#
#  host_type              OS manager          build manager
#
   apollo                 domain_os           dds
   prism                  domain_os           dds
   dn4500_w_fpa           domain_os           dds
   hp-ux                  posix               rsh
   sun386i                posix               rsh
   unix                   posix               rsh
```

Each host type has an associated OS manager, which is used to manage derived objects and translate pathnames, and a build manager, which is used to select build computers and start compilations. DSEE V4 provides OS managers for Domain/OS and POSIX, and build managers for Domain network services (dds) and rsh (remote shell). Each host type also has an associated list of build computers. DSEE picks one, and then starts the build on the foreign machine. If additional builds are required, they can proceed in parallel.

Split Pools. The bound configuration thread and derived object sections of a pool can optionally be split into separate directories. For non-Apollo users, this means that users can use their existing investment in non-Apollo disks to store binaries. Depending on the user's network configuration, this can also result in faster load (/bin/ld) times. When there are DSEE managers for non-UNIX operating systems, split pools may be essential, since many have file system attributes that have no equivalent on Domain/OS.

User interface

Version 4 also comes with a new user interface (see Fig. 8). It is based on the X Window System, and conforms to OSF/Motif standards.[2] As such, it can be used remotely from any machine that supports X. The user interface is object-oriented. Objects can be browsed, resulting in icons being copied to a desktop area. DSEE commands can be applied to selected icons through associated menus, picked from a menu bar, or entered from a textual command window. When DSEE needs more information, it pops up a dialog box. The boxes are preseeded with likely option values. Some of these dialog boxes support multiple iteration, which allows the same command to be issued repeatedly, each time using a different list of arguments (Fig. 9). All commands result in entries in a command history, from which they can be perused and reissued.

DSEE is still accessible through earlier user interfaces: a command line interface based on Domain/OS display manager pads and a serial I/O interface which is used when the input stream is not a window. There are also two programmatic interfaces that allow DSEE commands to be intermixed with C or shell commands. When the binary library /lib/dseelib is bound into a program, the program can issue DSEE commands. This callable interface was used to write dsee_server_c, an example program shipped with DSEE. A client program, ds, sends IPC messages containing DSEE commands to the server, which executes them through the procedure dsee_$cmd. The result is that shell

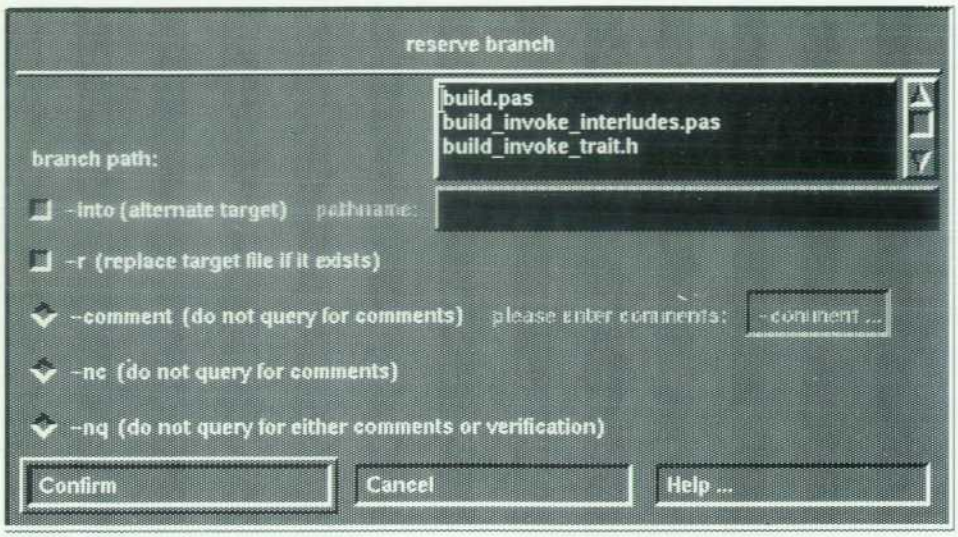


Fig. 9. *A dialog box that provides multiple iteration. The box contains a scrollable window containing element names. Users can select one or more of these names and DSEE will issue a separate reserve branch command for each selected element name.*

control constructs can be used to postprocess the results of DSEE commands.

Conclusion

DSEE is a powerful tool for version control, configuration management, and project management for software projects. While it is often used on small projects, it is clear that the product's particular strengths are in handling the complexities inherent in large projects developed by teams of programmers.

References

1. B. D. Fromme, "HP Encapsulator: Bridging the Generation Gap," *Hewlett-Packard Journal*, Vol. 41, no. 3, June 1990, pp. 59-68.
2. A. O. Deininger and C. V. Fernandez, "Making Computer Behavior Consistent: The HP OSF/Motif Graphical User Interface," *Hewlett-Packard Journal*, Vol. 41, no. 3, June 1990, pp. 6-12.

Bibliography

1. R. P. Chase, Jr. and H. Spilke, *Device for Managing Software Configurations in Parallel in a Network*, U.S. Patent No. 4,951,192.
2. P. J. Leach, P. Levine, B. Douros, J. Hamilton, D. Nelson, and B. Stumpf, "The Architecture of an Integrated Local Network," *IEEE Journal on Selected Areas in Communications*, November 1983.
3. D. B. Leblang and R. P. Chase, Jr., "Computer-Aided Software Engineering: a Distributed Workstation Environment," *ACM Software Engineering Notices and ACM SIGPLAN Notices*, May 1984, pp. 104-112.
4. D. B. Leblang and R. P. Chase, Jr., "The Domain Software Engineering Environment for Large-Scale Software Development Efforts," *Proceedings of the First IEEE International Conference on Computer Workstations*, 1985, pp 266-280.
5. D. B. Leblang and R. P. Chase, Jr., "Parallel Software Configuration Management in a Network Environment," *IEEE Software*, November 1987, pp. 28-35.
6. D. B. Leblang, G. McLean, Jr., H. Spilke, and R.P. Chase Jr., *Computer Device for Aiding in the Development of Software System*, U.S. Patent No. 4,809,170.
7. D. Lubkin, "Integrated Text Management," *International Workshop on UNIX-Based Software Development Environment*, Dallas, Texas, January 1991.
8. D. Lubkin, "Heterogeneous Configuration Management with DSEE," *3rd International Workshop on Software Configuration Management*, Trondheim, Norway, June 1991.
9. D. McCreary, "Networking Unties OS Development Knot," *ESD Magazine*, Feb. 1988.

A Mechanism to Support Parallel Development via RCS

HP's Imaging Systems Division uses the HP-UX revision control system utility, RCS, to implement a configuration management system that allows stable, released software to remain unchanged while modifications are made to some of its components.

by John W. Goodnow

AT VARIOUS INTERMEDIATE checkpoints during the product development life cycle, it usually becomes necessary to stabilize the software for events such as field trials and final functional verification. A conflicting, but equally important requirement is that development for future checkpoints and releases must be allowed to continue while baseline* software is kept stable. At HP's Imaging Systems Division (ISY), the need for this parallel development is also driven by continuous field trials and frequent software upgrades. ISY produces real-time embedded systems software for the control of ultrasound instruments (primarily diagnostic cardiac imaging). A typical instrument consists of several subsystems, which range in size from 700 source files totaling 60 KNCSS** to 1200 source files totaling 200 KNCSS.

This article describes a simple mechanism, based on the HP-UX platform and the RCS (revision control system) utility, for effectively achieving parallel development capabilities and implementing baseline configuration management. The mechanism described is reliable, robust, and simple to use and administrate. It is widely used by the software development groups at ISY.

Although the parallel development mechanism described here can be integrated with almost any software development environment in which RCS is used, the implementation at ISY is based on an in-house software development environment built over the last four years. This underlying environment, hereafter referred to as the ISY SDE, or just SDE, runs on a distributed network of HP 9000 Series 300 workstations and uses RCS and nmake tools to realize revision control and build management. To understand the workings of the parallel development mechanism, it is important first to understand the major underpinnings of the ISY SDE.

The ISY Software Development Environment

The ISY SDE is essentially an extensible directory structure and a rich set of support tools that provide the development group with revision control and build management. In addition, support is provided for independent private development, in which individual software engineers can link local copies of modules under development with stable baseline software.

Directory Structure. In general, a particular instance of the ISY SDE is rooted at a directory consisting of three logical components: the super root, the name, and the revision. For example, in the SDE rooted at /users/prism/dss/main, the super root is /users/prism, the name is dss, and the revision is main (see Fig. 1). Each instance of the ISY SDE has an area for source (and header) files under RCS control as well as a separate area in which individual software engineers can keep local, working versions of any of those source files (work directory in Fig. 1).

Revision Control Based on RCS. As stated above, the revision control aspects of the ISY SDE are handled by RCS. Without considering parallel development, the use of RCS in the SDE is quite straightforward. The SDE contains a large body of source files, all of which are under RCS control. At any given time, the latest revisions of these files make up the *mainline* configuration. For projects in the ISY development lab, the mainline usually corresponds to unreleased software in various stages of development.

During the course of development, software engineers can check out mainline files, modify these files, generate and test executable software based on these modifications,

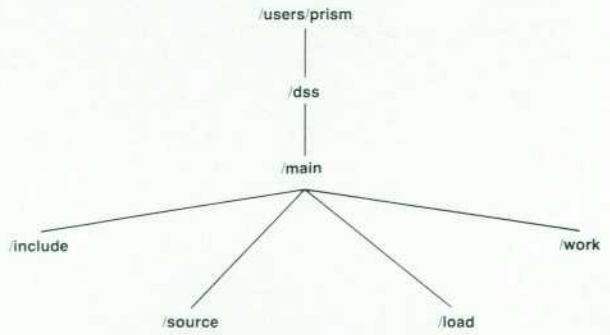


Fig. 1. *An example of an ISY software development environment directory structure. The* include *and* source *directories contain the standard, globally available software. All the files in* source *are compiled nightly to produce a standard daily executable, which is placed in the directory* /load. *The files in the* work *directory are completely private to the individual engineers.*

*In our environment the baseline contains all the components (source code, executables, documentation, etc.) from which the final product is created.

**KNCSS = thousand lines of noncomment source statements.

and check the files back in. These modified files become part of the mainline when they are checked back in. These operations are the standard checkout (co) and checkin (ci) RCS commands.

Build Management Based on Nmake. The ISY SDE uses the nmake program to manage the build process. Nmake is very similar to the standard make program, but it is substantially more flexible. For example, nmake automatically evaluates and maintains dependencies introduced by header files.

For a typical ISY project, the mainline is always built nightly, and is rarely, if ever, built during the day. This is because the build process takes several hours to complete. Building the mainline during work hours not only leaves the project's object files in an inconsistent state for long periods of time, but also introduces unacceptable loading on the workstations executing the build.

Standard Directory Types. The source files that make up the mainline of the project are stored in a hierarchical directory structure. This ensures that each source directory contains a set of functionally related files, and no directory contains an excessive number of files. Header files are kept in a global header directory or in individual source directories according to their scope.

Although the hierarchy and allocation of source files to directories are arbitrary and completely at the software engineers' discretion, the organization of an individual source directory is highly structured. This structure is imposed by the SDE, and it exists to strike a good balance between the capabilities provided and ease of use. A source directory with this SDE-imposed structure is called a standard directory.

Several different types of standard directories can be configured, and all standard directories are classified according to two attributes: major and minor directory type. The major directory type determines the combination of revision control and object file generation services provided, while the minor directory type selects the compiler family to be used for object file generation.

Each major directory type is tailored for certain requirements (see Fig. 2). A revision control standard directory is used to hold header files, which don't need any object file generation. A source standard directory is used to hold source files which must be under RCS control and require object file generation, such as compilation and assembly. A working standard directory provides object file generation services only, and is ideal for software engineers wishing to modify, compile, and test their own local versions of source files. The major directory types and their capabilities are summarized in Table I.

Table I
Major Directory Type and Capabilities

Major Type	Revision Control	Object File Generation
Source	X	X
Working		X
Revision Control	X	

Examples of different minor directory types include the native HP-UX compiler and assembler family and a 68000 cross compiler and assembler family. The HP-UX minor directory type is used for development of host software designed to run under the HP-UX operating system, while the 68000 cross compiler minor directory type is used for development of software targeted for execution in embedded systems. Moreover, the method of handling minor directory types is fully extensible, allowing plug-compatibility with additional minor directory types defined by end users.

One inherent feature of a standard directory that greatly enhances its ease of use, at the possible expense of end user flexibility, is that all files in a given standard directory with a given suffix are treated identically. This means, for example, that all C source files in a specific standard directory will be compiled with the same command line. Furthermore, generic makefiles with a standard set of rules are used for object file generation in all standard directories. This eliminates the need for software engineers to create or maintain makefiles.

RCS Features

Support for parallel development draws heavily on three advanced RCS features: branching capabilities, directory links, and symbolic names. Although all of these are fully supported and documented RCS features, each will be described briefly below. Refer to the HP-UX reference manual for a complete discussion of these features.

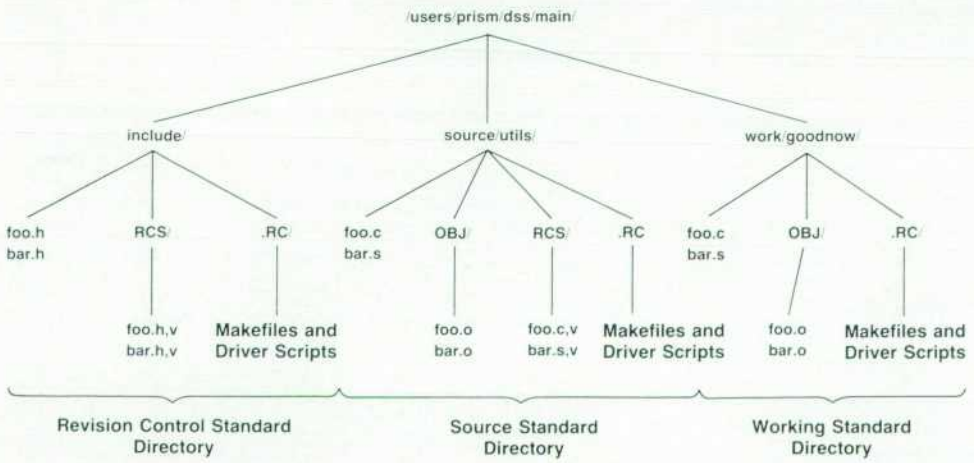


Fig. 2. *Standard directories.*

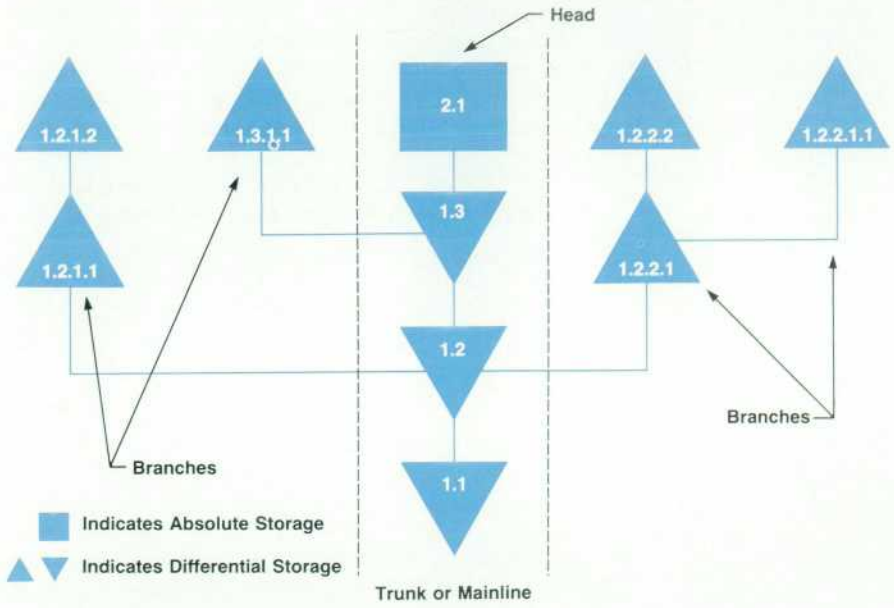


Fig. 3. *Branching structure of an RCS file. Absolute storage means that every line of the particular version is stored. In RCS the head is the only version stored absolutely. Differential storage means that only the changes from the previous version (i.e.,* diffs *in HP-UX nomenclature) are stored. Typically, this consists of a series of added and/or deleted lines and associated line numbers, which when applied to an absolute version results in a new absolute version.*

Branching Capabilities. Each RCS file (an RCS file is a file ending in ,v) has a trunk, and optionally, a number of branches. The trunk has a two-field revision number (e.g., 1.25, 2.1). Branches have a 2n-field ($n \geq 2$) revision number (e.g., 1.25.1.1, 2.1.1.3, 1.1.1.3.1.1). The latest revision on the trunk is called the head. Fig. 3 illustrates these branching relationships.

This structure closely corresponds to the ISY method of splitting off branches for releases, trade shows, clinical trials, and other checkpoints. The parallel development mechanism built into the SDE uses these branching capabilities to allow RCS to manage branching on a file-by-file basis. For a subsystem with multiple revisions, although there are multiple development trees, there is only one set of RCS files located in the mainline tree. Each RCS file in the mainline tree holds all possible configurations of its corresponding source file. Furthermore, since there is only one set of RCS files, it is unnecessary to copy any RCS files when a new tree is split for parallel development. Instead, pointers to RCS directories are copied. This is achieved via the directory link feature described next.

One final note about branches is that they are only created when necessary, which is when conflicting modifications are made to the same file during the course of parallel development. Extremely stable source files, which are rarely changed, are very likely not to have any branches at all, regardless of the number of parallel development trees.

Directory Links. RCS directory links are fully integrated into the ISY SDE. These directory links allow multiple directories to be created for checkout and compilation, all based on a single mainline RCS directory. RCS links are created by embedding a path to another RCS directory in a regular file with the name RCS. Fig. 4 shows a simple use of RCS directory links. Directory links may be chained, and RCS will follow a chain of up to ten directory links to reach the actual RCS directory. Directory links should not be confused with HP-UX symbolic links; the two are completely separate features.

Symbolic Revision Naming. The final RCS feature used to support parallel development is the ability to associate a symbolic name with a particular revision of an RCS file. Once a symbolic name is created, revisions can be selected, checked out, and checked in using the symbolic name rather than a specific revision number. An RCS file can have any number of symbolic names, and multiple symbolic names may also refer to the same revision.

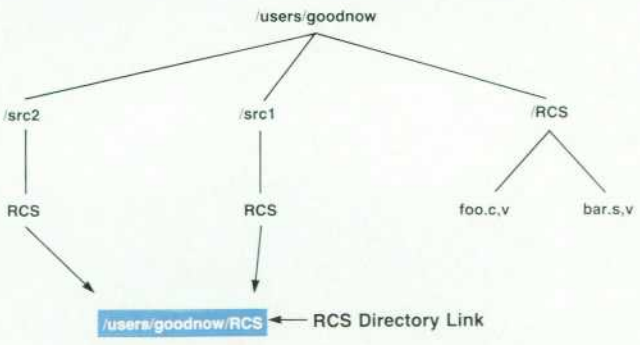


Fig. 4. *Use of RCS directory links. Only one copy of files* foo.c,v *and* bar.s,v *is needed. Other directories contain a file* RCS *which contains a path to the directory where these files exist.*

ISY Implementation

The RCS features described above are used to implement the ISY SDE configuration management system. Basically, this involves creating a split tree for parallel development. The split tree has the same super root and name as the original tree, but has a different revision. The overall directory structure of the split tree is the same as that of the original. Fig. 5 shows an example of a split tree based on the directory structure described earlier.

The support provided by the SDE provides as much transparency as possible, so that for most cases, individual software engineers need not be concerned with low-level details. For example, check in and checkout of RCS files in the split tree work the same way as they do in the main-

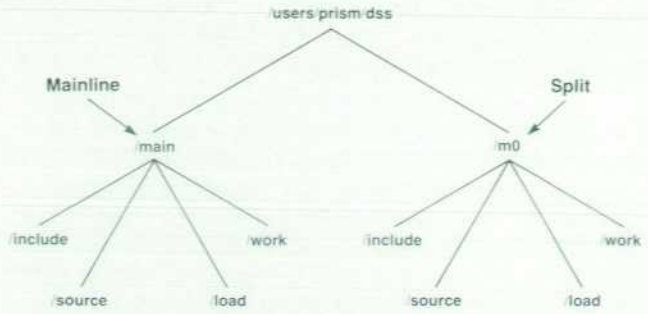


Fig. 5. *Directory structure for mainline and split trees.*

line tree with some infrequently occurring exceptions, which are described later. This is possible because the RCS options required to select the proper revisions for checkout and check in are supplied automatically by the SDE. Thus, this operation is transparent to the software engineer. The parallel development mechanism is recursive in that a split tree can be created from the mainline tree as well as from another split tree. Masterfiling* a split tree is fully supported. This is a common operation, as splits are frequently created for software release.

Splitting a Tree. To split a tree for parallel development, the following operations must occur.

- All user-contributed source files in the original tree must be under RCS control.
- All RCS files in the original tree, whether this is the mainline or a previously split tree, are given a symbolic name, which is defined as the name of the split. The name of the split will hereafter be referred to as split_name.
- The original tree is cloned over to a new tree rooted with the split_name. In our example directory structure given in Fig. 5, the /users/prism/dss/main tree would be cloned over to /users/prism/dss/m0 (m0 is the split_name). This is no ordinary clone, however. The directory structure of the original tree is cloned verbatim, but, in terms of RCS files, a virtual copy of the original tree is made, with RCS links being generated in the new split tree pointing back to corresponding RCS directories in the original tree. In other words, when a split tree is created, no RCS files are copied, but RCS directory links are created instead.
- The RCS checkout and check-in options for the split tree are modified to refer to the symbolic names (or split_name) created in the previous step.
- The new split tree is booted, which means it is turned into an independent environment which can be entered and used just like the environment containing the original source tree. The boot process consists of building all the system tools and configuration files in the new split environment.
- The new split tree is built. Since this step requires checking out and compiling all user source files, which is a lengthy process, it usually occurs overnight.
- The new split environment is now ready for use. It is entered as a separate development environment. Development can proceed independently in both the original and split environments.

*Masterfiling is the process of storing all files in an environment to tape so that the executable can be reconstructed offline on another system.

Checkout and Check In with Split Trees. In most cases, making a change to a file in a split environment is no different from making a change to the same file in the mainline. The commands used to check out (co) and check in (ci) the file are the same in either case. To provide this functionality, the user is not allowed direct access to the co and ci RCS commands. Instead, the SDE provides two interface scripts called envco and envci. The interface scripts intercept user command lines, add additional options to the user commands, and then pass the concatenated string onto RCS. For example, suppose it is necessary to modify the file foo.c in either the mainline or a split. Table II summarizes typical user command lines, and shows the resultant commands issued for both mainline and split trees.

Table II
User versus Actual Command Lines for Mainline and Split Trees

User Command Line	Actual Command Lines	
	Mainline	Split Tree
envco -l foo.c	co -l foo.c	co -l split_name foo.c
envci foo.c	ci foo.c	ci -N split_name foo.c
envrcs -l foo.c	rcs -l foo.c	rcs -l split_name foo.c

For the split tree case, by specifying a particular revision of a file when checking it out, we can randomly access any revision contained in the RCS file, not just the head. This is used in conjunction with the symbolic naming capability to check out the proper revision of a file for a given split.

Changes made in the mainline never affect the split. For example, consider the situation in Fig. 6 in which the file foo.c,v has three revisions: 1.1, 1.2, and 1.3. Revision 1.3 is symbolically named m0. Checking a change to foo.c,v into the mainline will result in a new revision, 1.4. This new revision becomes the new head of foo.c,v. Since the symbolic name m0 still points to revision 1.3, the revision of foo.c,v used in the m0 split is not affected.

For the checkin command in a split tree case, the symbolic

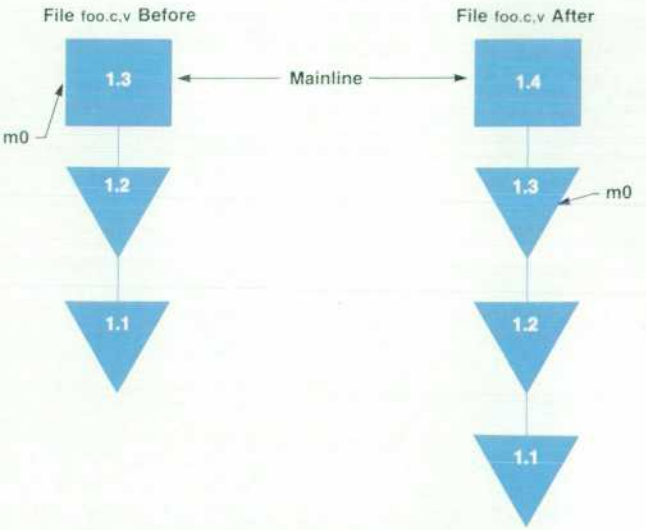


Fig. 6. *An example showing how a change to the mainline does not affect the split.*

name is updated to refer to the newly checked-in revision. Furthermore, when a file is checked into a split, an RCS branch may or may not be created. An RCS branch is created if the revision is not at the head of an existing branch (see Fig. 7). Otherwise, if the revision is at the head of an existing branch, including the trunk, then the new revision becomes the new head of that existing branch (see Fig 8). This functionality has some important ramifications.

When checking a file into a split, in most cases it is desirable to propagate the changes through to the parent tree, which is usually the mainline. This is possible, indeed automatic, if the revision of the file for the split is the same as the revision for the parent. In this case, checking the file into the split automatically checks the same revision into the parent. This functionality can be exploited in that many defect fixes can be made once in the split, then checked into both the split and the parent in one operation.

In the case where the revision of the file for the split is not the same as the revision of the file for the parent, the changes cannot be propagated to the parent. In this case, a new branch is either created, or an existing branch continued. If the changes are to be propagated to the parent, then the file must be checked out, modified, and checked back into the parent environment. The RCS command rcsmerge is often used to merge in the changes semiautomatically in this scenario. In practice, the automatic propagation case mentioned first occurs far more frequently than does the nonpropagation case, so there has not been much incentive to automate the case in which changes cannot be propagated to the parent.

Although it is usually desirable to attempt to propagate defect fixes automatically from a split to its parent, in some cases this is not desired at all. In these cases, the user can force an RCS branch during check in. Again, in practice, this occurs very infrequently.

Quality and Productivity Gains

Many quality and productivity gains have resulted from the mechanism described in this article. First, development on new projects can continue independently in the mainline while a split is created for software slated for release (i.e., baseline software). Final functional test, which often takes four to six weeks, is done on the split, stable, software. Changes to the stable software are limited to certain defect fixes and are extremely well-controlled. New development for future projects can take place simultaneously and independently in the mainline. In the past, new development (or at least file check-ins related to new development) had to cease during the several weeks of functional test, creating many bottlenecks, not to mention idle software engineers.

Next, in the past, files were simply copied from one place to another during the process of creating final (frozen) software. Once this was done, it was extremely difficult to propagate changes from one version of the software to the other reliably. This put an excessive burden on the software engineers charged with maintaining several completely separate versions of released software, and this generally manifested itself as a schedule slip during functional test as problems were uncovered. This problem has been reduced to manageable proportions by the parallel development mechanism described here. As discussed above, changes can in most cases be automatically propagated from a split to the mainline. Also, since all versions of the software are centralized in one set of RCS files, it is easy to track the status of all software versions. Reports can be generated (typically via the rlog and awk programs) to describe all the changes to a split since a certain date, time, second, and so on.

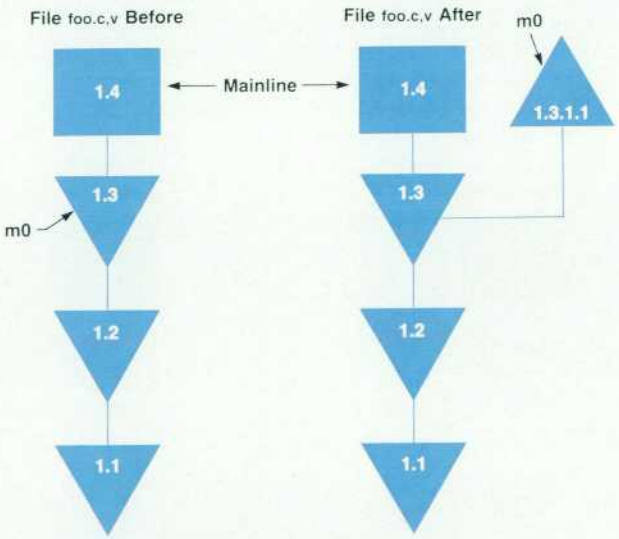


Fig. 7. *Creating an RCS branch when a revision is not at the head of an existing branch. In this case* foo.c,v *has four revisions: 1.1, 1.2, 1.3, and 1.4. Revision 1.3 is symbolically named* m0. *Checking a change to* foo.c,v *into the* m0 *split forces the creation of the RCS branch 1.3.1.1. The symbolic name* m0 *is updated to point to the new revision 1.3.1.1. Checking the change into the* m0 *split does not result in its being checked into the mainline.*

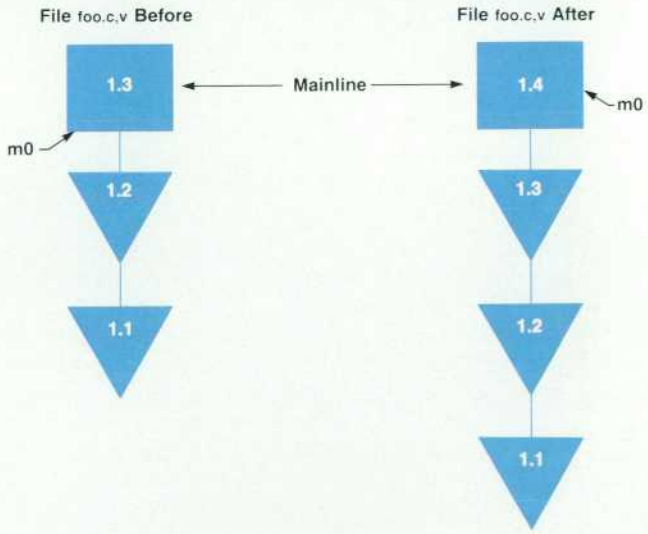


Fig. 8. *Making a change to a file when the revision is at the head of a branch. In this case* foo.c,v *has three revisions: 1.1, 1.2, and 1.3. Revision 1.3 is symbolically named with the* split_name m0. *Checking a change into* foo.c,v *results in a new revision, 1.4. The* split_name m0 *is updated to point to revision 1.4. Since 1.4 is now the head of* foo.c,v, *this new revision is used by the mainline as well.*

Third, custom software loads in which several files (either experimental or from the mainline) are added to a baseline of stable software (typically a split), are much easier to generate with the mechanism described here. Because all versions of the software are kept in a single set of RCS files, it is easy to identify and extract the proper revisions for the custom load. Furthermore, these customizations are very well-controlled because they are built on a baseline of stable software. This is very important for ISY because field trials are often conducted to evaluate a single feature, with all other functionality assumed to be unchanged from the given released software. In the past, generating custom software had the unpleasant side effect of incorporating changes to files unrelated to the features under test.

Finally, management is now able to assume that software splits are now not only possible, but reliable as well. This has allowed an additional degree of freedom in project planning which was not previously available.

Conclusion

The mechanism described here has been in use at ISY for approximately two years. During that time, the software environments for three major subsystems of the HP SONOS 500/1000 ultrasound system have used the split mechanism a dozen times for both formal software releases and field trial software. These three subsystems are made up of approximately 1000, 500, and 300 source files.

Although the use of RCS to implement parallel development capability has been highly successful within ISY, it is not perfect. First, there are some minor problems in the RCS toolset which require an awareness on some level, either by the software development environment administrator, or by the software engineers themselves. As an example, the RCS tools do not allow concurrent access to an RCS file, so two users cannot check out different revisions of the same file at the same time. Another problem is that although the mechanism described in this article is ideally suited for splitting off software revisions which then become relatively stable (e.g., software release), it is not very well-suited for long-term parallel development in which multiple revision threads are all active and all equally important. This gets back to the very structure of an RCS file, which is a single, dominant trunk with an arbitrary but inferior branch structure.

Overall however, the parallel development mechanism described here has been highly successful across multiple software development groups within ISY. It allows excellent control over stable software in conjunction with independent parallel development of new software. Best of all, the mechanism is simple and applicable to any software development environment in which RCS is used for revision control.

Acknowledgments

The author would like to acknowledge Dave Desormeaux (now at HP's Graphics Technology Division in Fort Collins, Colorado) for his work in conceiving, designing, and implementing the original ISY Software Development Environment over four years ago. Without his key contribution, building the parallel development extensions described here would have been immeasurably more difficult. In addition, the author would like to acknowledge Bill Harte of ISY for his many valuable design suggestions. Finally, the author would like to acknowledge the many ISY software engineers for their enhancement ideas.

Building and Managing an Integrated Project Support Environment

HP's Roseville Networks Division has developed an integrated, cost-effective computing environment that fosters cooperative computing and provides R&D engineers with easy access to the tools and methodologies for product development.

by Ronald F. Richardson

ONE OF THE WAYS to foster increased quality and productivity in an R&D environment is to provide consistently managed and maintained computing facilities that allow engineers to have easy access to the latest tools and methodologies. In August 1988 the software quality and productivity team at HP's Roseville Networks Division (RND) was assigned the tasks of managing the overall R&D HP-UX computing facility and developing a foundation from which to build a highly integrated project support environment. The primary goal of this effort was to provide a tool-supported software engineering process built on sound engineering platforms to facilitate creating better products faster. Our challenge was establishing a computing strategy and direction for laboratory-wide cooperative computing in an R&D organization that traditionally addressed their computing needs on a project-by-project basis.

This paper discusses cost-effective HP-UX computing and ways of reducing the effort of maintaining and administering this environment. The main areas discussed include financial aspects, cooperative computing, network architecture, and a management model for system administration.

Financial Aspects

The first hurdle we had to overcome was convincing senior management that workstations for software engineers made financial sense. The current environment was primarily based on central HP-UX timesharing systems. Most of our software engineers had three to four terminals in their office to emulate a windowing environment. This was required because in developing networking products many machines were used. Remote communication from one machine to another via data switching was extremely time-consuming and multitasking was not easily achieved. Hardware engineers were just beginning to use workstations within their areas. The financial situation required outfitting over 100 engineers with workstations while keeping the cost per engineering seat close to the cost of multiple terminals or personal computers.

Fortunately, at that time, HP's HP-UX development laboratory in Colorado had just finished developing the first version of HP-UX diskless workstations.[1] Diskless systems have the advantages of a timeshared environment while maintaining the performance of standalone workstations. In fact, depending on the configuration, performance exceeding that of a standalone system is possible. Advantages of diskless systems include:

- Centralized file system. The user sees a single file system from any workstation, even though the file system may be located far away on a file server.
- Lower total system cost. Peripherals such as disks and tape drives are only needed for the file server. This significantly reduces the cost per user.
- Lower cost of ownership. With fewer mechanical parts (disks, tape drives, printers, etc.) maintenance costs are reduced over those of standalone systems. Additionally, software support contract prices are also reduced.
- Easier system administration. Administration focuses on the file server and the cluster disk, and support of the diskless nodes is no longer an issue. Contrast this with a standalone network in which each machine is administered individually.

The financial model was not one of accounting genius but mainly common sense. R&D was evenly split between hardware and software engineers. The hardware engineers were currently purchasing standalone workstations with four disks. Using diskless workstations and amortizing the cost of the server over eight nodes per cluster, a hardware engineer could be outfitted with an HP 9000 Series 360C with 16M bytes of memory for 25% to 50% of the cost of a standalone system. As an additional checkpoint, our finance department used a standard accounting model to determine that a 7.2% productivity increase was required to achieve a 20% ROI (return on investment) on the workstations over a five-year period. Subjective measures led us to believe that this productivity increase could be achieved. Given the savings associated with diskless workstations, having an attainable productivity goal, and realizing that software engineers were undercapitalized relative to their hardware counterparts, funding was provided to purchase 110 workstations.

Architecture

A successful diskless implementation is based on a well-designed and maintained network. If a reliable network cannot be achieved, diskless is not the preferred distributed

computing environment. In this section we first look at the logical model of our cooperative computing environment to obtain an overview of workstation and server organization, then review the physical model of the network topology including the lower layers (physical, link, and network), which provide the foundation of the network.

Logical Model

The essence of workstation organization is a triad of systems supporting a project team. The first system is the boot server. This machine's primary functions are to provide HP-UX facilities to its configured diskless clients and swapping over the network. The second piece of the triad is the diskless clients. There is one of these diskless workstations in each engineer's office. The workstation is the engineer's user interface into the computing environment. Using X Windows, this is where most of the engineers perform their work ranging from schematic entry to C compiles. The final piece of the triad is the project server. It acts as a file server and compute server for project-specific needs. All three of these components need to be highly integrated over a fast, reliable network.

Physical Model

A local area network must move packets of data from one host to another as quickly, reliably, and efficiently as possible. The diskless configuration also requires the LAN to have minimal impact on the individual user (host). Not only should office connections to the network be unobtrusive, but maintenance and reconfiguration should be virtually unseen by the workstation user. If careful choices are not made, there is the risk of maintenance problems later.

Requirements. HP StarLAN 10* was chosen as the main transmission medium connecting the triad of machines supporting the project team. HP StarLAN 10 provides the same functionality as Ethernet, only over a 10-Mbyte/s twisted-pair cable with the added benefit that connections are like point-to-point. Administration of the current R&D laboratory HP ThinLAN network, from a hardware perspective, consisted of "install and forget until the network stops working." This approach could not be taken with the new HP StarLAN 10 network because we would be servicing over 100 diskless workstations. Our problems included reconfiguration, fault isolation, and traffic management.

- Reconfiguration. More hosts require continual expansion and reconfiguration of the network. These changes must have minimal effect on the current users.
- Fault isolation. Problems are inevitable—the more systems, the more faults. A failure, either hardware or software, should not affect the entire network.
- Traffic management. As the network grows, so does traffic. The ability to contain local traffic in one area becomes extremely important with a diskless system.

To address these issues we use HP StarLAN 10 hubs (multiport repeaters and wire centers) and bridges.

Cabling Plan. The engineers' offices were wired with four-pair twisted-pair cable. This eight-wire cable is split into two four-wire portions: two pairs for phone and two pairs for HP StarLAN 10. We use HP StarLAN 10 transceivers* to connect the workstations to the network. Unlike HP ThinLAN coaxial cable, which can be 185 meters in length and have 30 nodes attached to it, the HP StarLAN 10 cable has a 100-meter length restriction and can only have one node attached much like a terminal point-to-point connection. To bring together these multiple segments we use HP StarLAN 10 hubs to concentrate twelve segments at a single point. They are then interconnected via a standard HP ThinLAN backbone segment (see Fig. 1). In addition to providing interconnect convenience, the HP StarLAN 10 hubs provide electrical isolation of individual segments from the entire network. Should a cable become damaged, only the user on that segment is affected and troubleshooting is much easier.

Gateways. Like any repeater, HP StarLAN 10 hubs appear transparent to the interface on any host. Except for electrical problems on any given segment, repeaters do little to provide segment isolation. To control and monitor network

*The HP StarLAN products mentioned in this paper have been updated and are now called the Ethertwist [2,3] family of network products.

*HP StarLAN 10 transceivers are also called media access units (MAUs).

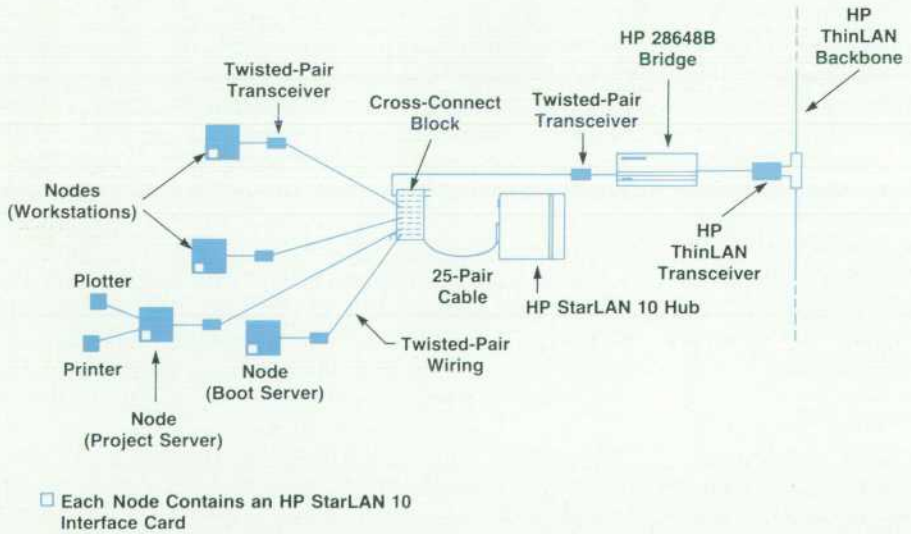


Fig. 1. *Multiple segments (workstations or servers) are concentrated at a single HP StarLAN 10 hub, which in turn is connected to an HP ThinLAN backbone. Up to 12 segments are allowed per hub.*

traffic a device is needed that deals with levels of networking beyond the Ethernet level (i.e., network layer of the Open Systems Interconnect (OSI) Reference Model).

Since our network is exclusively internet protocol (IP) based, the obvious choice for such a device is an IP gateway or router. Part of the internet protocol is the concept of logical subnets. This concept and its ramifications are outside the scope of this paper, with one exception: logical subnets provide a method of breaking apart a network into a local setting or a wide area network. Subnetting makes the division of local network traffic feasible.

Broadcasts prove to be the most troublesome traffic problem. Every interface that sees an address resolution protocol or user datagram protocol broadcast interrupts its host processor even if the packet is not required by the host. As the network grows, more CPU time is spent servicing these broadcasts. The IP gateway provides a logical boundary and constrains subnet traffic, including broadcasts, to a particular LAN segment.

We found that the organizational boundaries within our site suggest logical places for subnet partitions. Organizational partitions distribute the administrative responsibilities of each subnet to the appropriate entity. The gateway indirectly provides a psychological division that enables each group or entity to control its own network (subnet), yet retain connectivity with the entire corporate network. To the user, a neighboring host appears the same as a host in a different subnet, no matter where the subnet may be physically located.

Troubleshooting. Hardware problems are a constant source of worry for administrators of large networks. Certain tools are necessary to maintain and debug networks. When entire groups of hosts fail, the problem can usually be traced to a single segment failure, a hub failure, a bridge failure, or possibly a gateway failure. These types of problems are easy to find and correct. Since logical divisions are already in place, many of the more common problems are eliminated before a system administrator has to begin a binary host-to-host search for possible failures. The problems that cause the most difficulty are the host that occasionally transmits bad packets, or the segment that loses connectivity to certain other hosts on the subnet because of irregularities in cabling.

Conceptually we would like to know how well each host sees traffic (i.e., what percentage of incoming packets are bad). We also want to know how other hosts on a subnet see packets from a specific host. As a first-order solution, the internet control-message protocol echo (ping) can be used to provide some information on the performance of a subnet. The usual test is to send ping packets from one workstation to another and examine the error statistics on the returning packets, then proceed from machine to machine looking for a host or group of hosts with significantly more errors or a pattern of failed responses. A typical problem could be a host responding to only half the pings. This could indicate an electrical problem with the cable (e.g., the cable is too long).

The method works most effectively when the problem is fairly solid. However, experience has shown us that more often than not, users perceive an intermittent performance loss on the network as a failure. A quick check reveals nothing wrong. Solving intermittent problems requires some long-term statistics gathering. For this we use a network protocol analyzer and HP BridgeManager[4] (personal-computer-based network management software for bridge interrogation). With a statistics application package it is possible to obtain detailed information on any individual node as well as the network in general. We are able to gather such information as percent network use, error count, and collision count, in an understandable graphical format. Additionally, the protocol analyzer can provide these same statistics on any individual node or group of nodes. We have also used it effectively as a ping generator to check the performance of various hosts and gateways on the network. Given that the protocol analyzer is the most powerful device on the network in terms of frames per second, it can be dangerous and should only be used for stress testing during off-hours.

If the protocol analyzer has one fault, it is that it cannot examine how a particular interface sees the network. Often, one node will see the network with many errors while its neighbor only a few feet away sees few errors. Typically these errors include bad frame checksums or misaligned frames. Usually, bad cable connectors, a defective transceiver, or a bad tap are the causes of such problems. These types of errors may simply go undetected until a node with this problem attempts to connect to another node, whereupon the errors become so prevalent that reasonable data transfer is impossible.

To detect these sorts of problems before the user notices, an administrator needs to see how each node on the network "feels" about its interface. Since dragging the analyzer to each port on the net would be impractical, we log in to each machine remotely to execute diagnostic tools such as landiag and ping. This allows us to interrogate each system's view of the network.

Bringing It All Together

Our data center physical environment consists of a traditional raised-floor management information systems (MIS) machine room. The diskless computing equipment shares this space with the HP 3000 machines that support the rest of the division. We have 10 boot servers that provide 100 diskless clients with HP-UX facilities and swapping over the network. All boot servers have a standard configuration, allowing mutual backup in the event of a system failure. Each boot server is an HP 9000 Series 360C computer with 8M bytes and a single HP 7937H disk drive. All servers and their corresponding disks are rack-mounted in HP 19514A eight-pack cabinets. The console lines are run to multiplexer cards in another workstation in which each boot server has a dedicated console window. Ergonomics are done quite nicely.

In the mezzanine we have banks of HP StarLAN 10 hubs that support various areas of the office floor. Depending on project team location, their workstations will be connected to one or more of the hubs supporting that floor area. Each hub supports twelve ports and normally all members of a project team share one hub. The project server, which is also in the office area, is connected to port two of a team's hub. The boot server for the project team, located in the data center, is connected to port one of their

hub. The result is that all systems of the triad supporting the project team are physically on the same HP StarLAN 10 hub. All network file system (NFS) mounts and associated traffic are thus localized to one hub. In the event that the team generates too much network traffic, the hub is bridged from the HP ThinLAN backbone as shown in Fig. 1. Because we are using twisted-pair cable and cross-connect blocks to make the connections, patching between hubs, workstations, and servers is trivial (see Fig. 2). When engineers move, all they do is plug their workstations into the HP StarLAN 10 ports in their new offices. Their workstations will automatically boot from the correct server. Once the entire project team is situated, the boot server is patched to the appropriate hub supporting their new offices.

Today our new HP StarLAN 10 network and the old HP ThinLAN network, which is used for the remaining standalone systems, share the same subnet in the gateway. The HP StarLAN 10 hubs provide the network with a method of expansion and fault isolation within the user (office) environment and across the project team triad. Point-to-point functionality further increases fault isolation. The concept of subnets and the use of gateways add further flexibility to the network and their use will be expanded as traffic levels dictate. The twisted-pair cabling offered with HP StarLAN 10 makes reconfiguring the network to meet changing traffic patterns a trivial task.

A Management Paradigm for System Administration

Having effectively laid the computing foundation for future work toward building an integrated project support environment, the challenge of managing it arises. R&D computing facilities consist of many components, including the diskless environment just described. We were faced with developing a support paradigm that would merge the best practices of other organizations managing computer facilities while retaining much of the "hands on, no controls" mentality traditional to an R&D environment. The three major computing environments within HP were evaluated and their merits weighed.

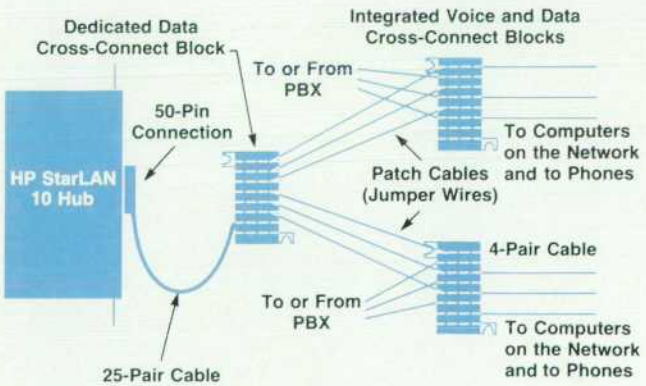


Fig. 2. *Connection of cross-connect blocks to an HP StarLan 10 hub.*

Computing Environments at Hewlett-Packard

It is necessary to understand three computing environments within HP to put the HP-UX support issue into perspective. These computing environments are R&D, manufacturing, and management information systems. Each environment is different in its focus, the metrics by which it is managed, and its hardware and software configurations.

R&D Computing. R&D computing is often managed and administered from an R&D productivity department. This department typically reports to the entity's R&D productivity manager. The department staff are engineers and system administrators (or they may be system administrators who are engineers). This department manages all aspects of R&D computing including installing applications, writing utilities, tuning performance, and designing and managing networks. Most of the users of these services are engineers, although cross-functional applications bring in other users. Managing CAD systems, design information, and CASE tools seem to be the primary concerns in R&D computing.

R&D computing is highly distributed and typically based on the HP-UX operating system running on HP 9000 systems. The highest premium in this environment is placed on productivity and ease of use of the various resources. Also, networking and connectivity are prime services to the R&D engineer. With the ever-shortening R&D cycles and increasing competition in HP's markets, any service allowing the R&D engineer to be more productive is highly valued. Security and control are acknowledged as important in this environment, but the trend has clearly been to emphasize openness and not to sacrifice productivity for security and control.

Information Systems Computing. Information systems computing is usually managed by an information systems department that reports to the entity or site information technology manager or entity controller. This department is responsible for all aspects of business computing and is accountable to entity management as well as HP corporate information systems. They may also be responsible for site communications and networks and the site data center. The department typically installs and supports all corporate-developed information systems that are used at the entity. A well-established internal audit process exists providing business control feedback on this group's performance.

Information systems computing has traditionally been centralized (few machines, many terminals) and primarily based on the MPE XL operating system and HP 3000 computers. The highest premium is placed on reliability, integrity, control, and security. Secondary importance is placed on the effective use of resources and on ease of use. Of course, productivity is the driving motivation behind all information systems. However, the trend in information systems has clearly been to emphasize security and control, sometimes at the expense of individual user productivity.

Manufacturing Computing. Manufacturing computing includes the elements found in R&D computing with two additional major areas of emphasis: process control and computer integrated manufacturing (CIM). Manufacturing computing is usually managed from a combination of the following.

- Information systems departments (business systems)

- R&D productivity departments (engineering systems and R&D links)
- Manufacturing systems departments (business systems and CIM)
- Manufacturing engineering departments (engineering systems and CIM).

The staff in these departments includes all types of information systems personnel and engineers.

The manufacturing computing environment encompasses both the R&D and MIS environments. In manufacturing there exists a large mix of computers and operating systems including HP 3000, HP 9000, HP 1000, and HP Vectra computers running the four major HP operating systems (MPE XL, HP-UX, RTE, and MS-DOS®). There are also non-HP computer systems that control non-HP process equipment. The move toward CIM has placed great emphasis on heterogeneous computer architectures in manufacturing because of the large mix of available process control equipment. The issues of productivity, security, and control have about equal importance in manufacturing.

Computing Support Model

The goal of the computing support model is to achieve a balance between the high value placed on productivity and ease of use by R&D and the concerns about reliability, integrity, control, and security paramount to the MIS community. To achieve this end, the model focuses not on the concerns stated above but on the data accessible and tasks performed on a given R&D HP-UX machine. From a security and control perspective, if a machine does not have data that could violate the rules of confidentially or harm the corporation by loss or disclosure, that machine does not pose a security risk unless it can facilitate "hacking" a machine or network that contains sensitive data. From a productivity and ease-of-use perspective, based on the diskless environment using X Windows, any auditable service is instantly available by opening a window to the secure machine providing that service. By layering services across many machines, as opposed to the traditional model of all services being on all machines, tight controls can be instituted on the machines and services that require them and more openness can be maintained on machines that do not house those services. This paradigm shift was reviewed with the internal audit team in 1988 and was viewed as a valid approach.

Fig. 3 shows an overview of our computing support model. Color codes are assigned to all the machines to indicate the level of service, the level of security and control, and the predominant use of the machine. Red and yellow machines are used for product development with the red machines designated for testing. The green machines have some development responsibility, but are primarily devoted to administrative and production-oriented engineering duties.

Through the model we are successful at stratifying our computing services through color codes so that the appropriate level of control, accountability, and access can be given to the right individuals without compromising the integrity of the overall computing environment. As an example, e-mail on HP-UX systems is viewed by the corporation as an auditable service on a par with HP Desk on MPE XL. The model therefore considers this service green. Mail then must be managed on a secure green machine and mail messages are only allowed to flow between other green machines on the network. If an engineer is working

MS-DOS is a U.S. registered trademark of Microsoft Corporation.

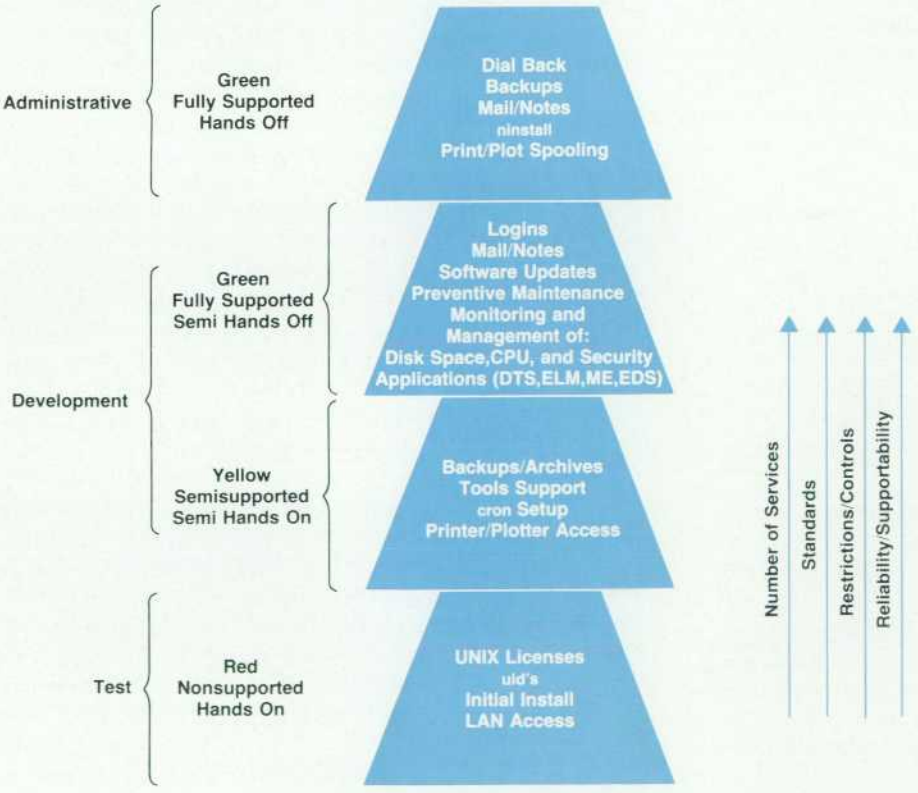
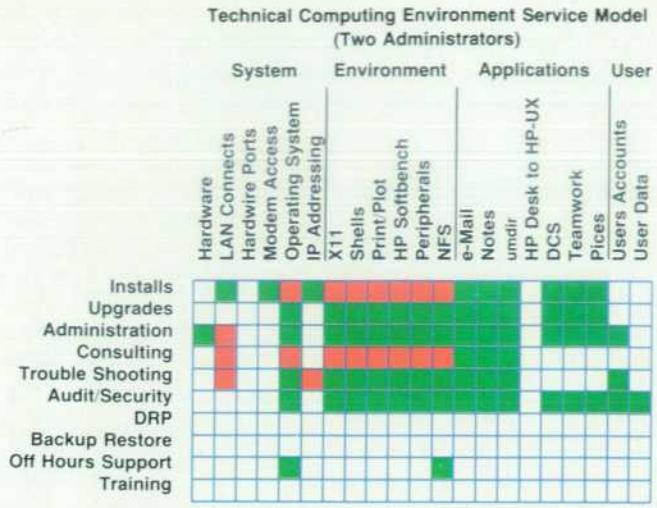
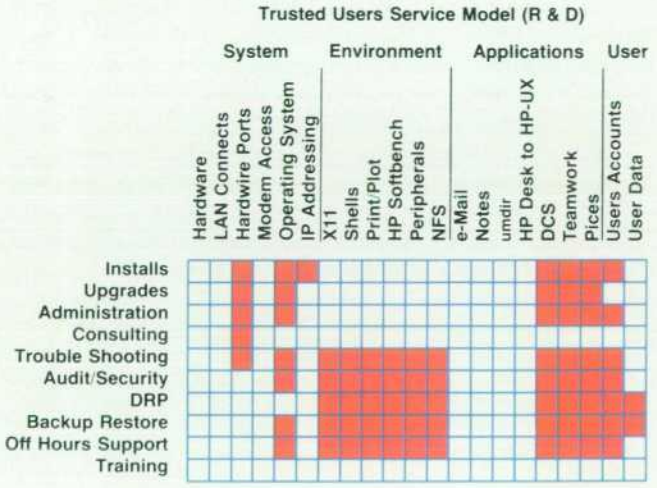


Fig. 3. *Computing support model.*





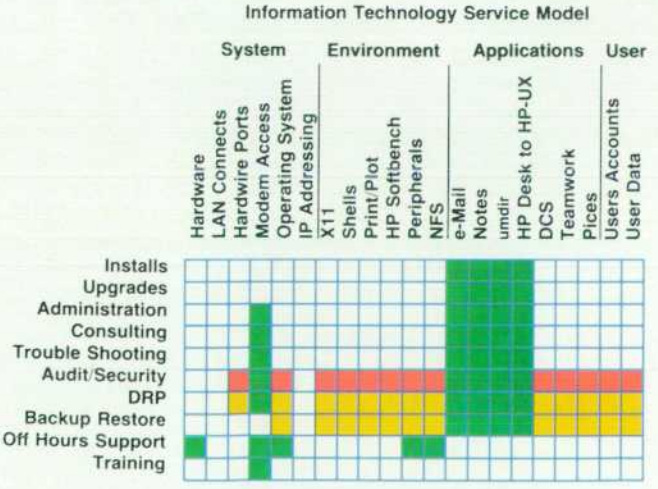
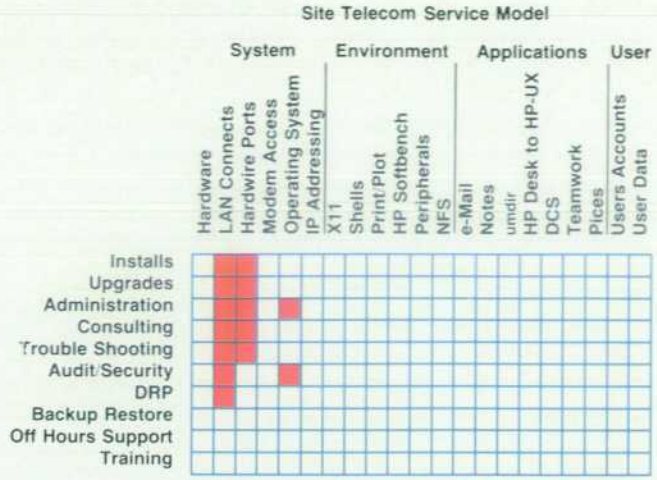
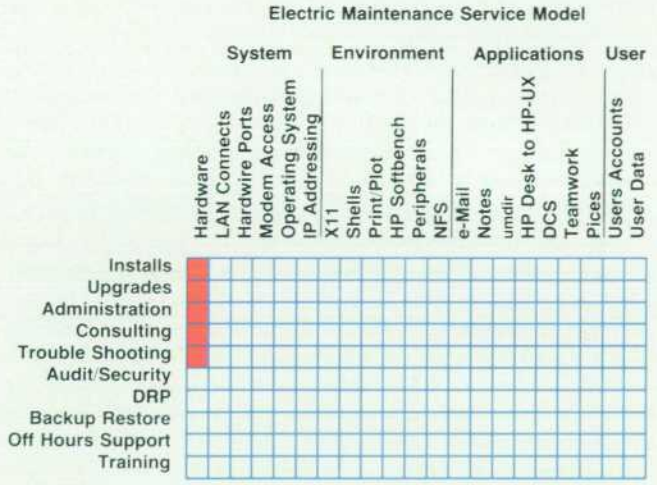


Fig. 4. *Service model matrices. These matrices define the services each organization is responsible for providing to the computing environment.*





on a red machine, which is typical of the testing environment, it is a minor inconvenience to open a window to a green machine offering mail. This approach ensures the integrity of mail confidentiality and administration with a minimal productivity impact.

Collaborative Resources

The service model matrices in Fig. 4 indicate that we have five organizations supporting the R&D computing environment. Based on the color assigned to each machine, each organization supports different services and varying levels of service. For example, the technical computing environment service team, which is made up of two system administrators, is responsible for most services on green systems (see the first matrix in Fig. 4). The trusted user team (second matrix in Fig. 4) is made up of R&D engineers who have the responsibility of providing HP-UX system administration and most other activities on red machines. These two groups meet monthly to work collaboratively on solving laboratory-wide productivity issues related to the computing environment. The remaining three organizations (information technology, site telecommunications, and electronic maintenance) are used like subcontractors for providing basic services such as backups, cabling, and preventive maintenance. By placing our services on ma-

trices and overlaying them we can see any holes in coverage. The support model has helped us stratify our support services and understand their requirements such that the annual business audit is effectively addressed. By building collaborative support teams with strong management support we are effectively bridging the gap between service providers and the end user. Because the trusted user team is effectively end users they help ensure that the customer's voice is incorporated into services provided.

Conclusion

Diskless systems have proven to be an outstanding first step toward implementing a cooperative computing environment. Each engineer has a CPU to perform general development tasks while the highly tuned project servers and boot servers execute very specialized requests. In terms of hardware acquisition and maintenance costs, diskless computing is a very cost-effective alternative to standalone workstations. Diskless performance has met and in many cases exceeded our expectations. This must be qualified by indicating that the diskless clusters must be tuned to specific types of workloads. An optimum configuration for us has been eight to ten nodes per cluster. Past this point diskless performance begins to degrade over that of a standalone system. Using diskless clusters has provided monumental strides in HP-UX system administration. We successfully support over one hundred diskless workstations with just two systems administrators. The old support ratios were one systems administrator for every ten to fifteen stand-alone machines. Diskless computing has helped us achieve economies of scale in many areas. HP StarLAN 10 has also proven itself extremely well by providing point-to-point functionality. If a node is unplugged, it has no effect on the rest of the network. Compare this with an open occurring on a standard HP ThinLAN network, which can have adverse effects on all nodes.

As well as it has worked for us, the diskless workstation concept has been a problem in a few cases. For example, hardware engineers can generate some huge files and swapping over the network can cause performance problems. Because of this we use local swap disks for most of our hardware engineers. We use HP 7958 132-Mbyte disks. We had many of these left over from converting stand-alone systems to diskless clusters. Therefore, these disks had little financial impact on the implementation.

Managing NFS mounts so that the correct file systems are coming from the correct servers for each project team is not something to be taken lightly. We have been successful at managing this aspect but it takes a fair amount of work. It especially gets challenging when more than one project team shares a boot server. At least two project servers are involved and they can occupy geographically different areas in the office environment. This has an impact on which HP StarLAN 10 hub they are connected to and how bridging is achieved. Again, we have been quite successful but we have room for improvement.

We have successfully laid the computing foundation to build an integrated product support environment for our R&D Laboratory. We still lack a solid integration framework and the methods and tools to support all phases of the product life cycle. We have begun to experiment with a fourth component to our cooperative computing platform, application servers. These machines house laboratory-wide tools that cross project team boundaries. We have successfully implemented two design capture system application servers for hardware design and an HP Teamwork application server for structured analysis and design for software engineers. We are in the process of implementing a laboratory-wide HI-LO® simulation application server and experimenting with SoftBench as our software engineering integration framework for CASE tools.

The computing support model, truly collaborative support teams, and strong management commitment have enabled us to achieve a score of eight on a scale of one to ten for our diskless environment according to those who manage the environment and those that use it.

HI-LO is a U.S. registered trademark of GenRad, Inc.

References

1. *Hewlett-Packard Journal*, Vol. 39, no. 5, October 1988, pp. 6-39.
2. *Technical Reference Guide for Workgroup LANs*, HP publication number 5091-0663E.
3. *Technical Reference Guide for Site and Multisite LANs*, HP publication number 5091-0666E.
4. A. S. Fraley and T. I. Perez, "HP OpenView BridgeManager: Network Management for HP LAN Bridges," *Hewlett-Packard Journal*, Vol. 41, no. 2, April 1990, pp. 66-70.

HEWLETT-PACKARD JOURNAL

June 1991 Volume 42 • Number 3

Technical Information from the Laboratories of Hewlett-Packard Company

Hewlett-Packard Company, 3200 Hillview Avenue
Palo Alto, California 94304 U.S.A.
Hewlett-Packard Marcom Operations Europe
P.O. Box 529
1180 AM Amstelveen, The Netherlands
Yokogawa-Hewlett-Packard Ltd., Suginami-Ku Tokyo 168 Japan
Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive, Mississauga, Ontario L4V 1M

HEWLETT PACKARD

5091-1595E