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.

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).

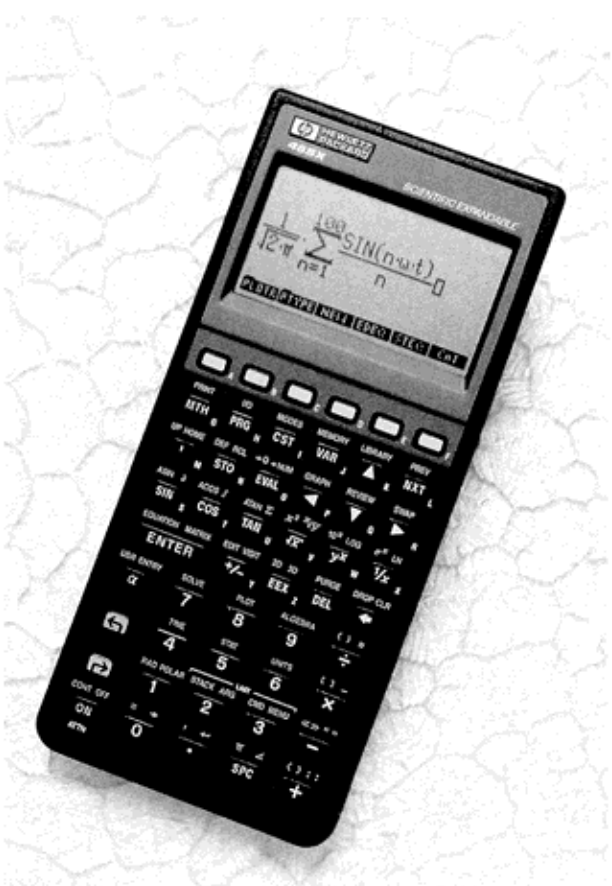


Fig. 1. *HP 48SX scientific expandable calculator, showing the EquationWriter application.*

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-

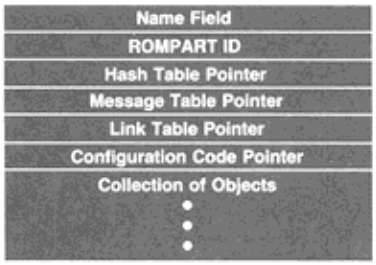


Fig. 2. *Structure of a ROMPART.*

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

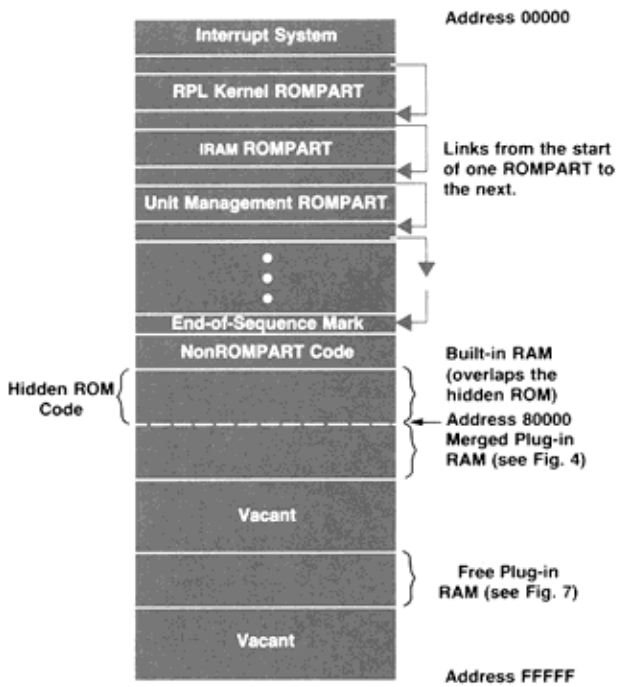


Fig. 3. *Overview of the address space layout of the HP 48SX calculator with one merged and one free (unmerged) RAM card.*

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

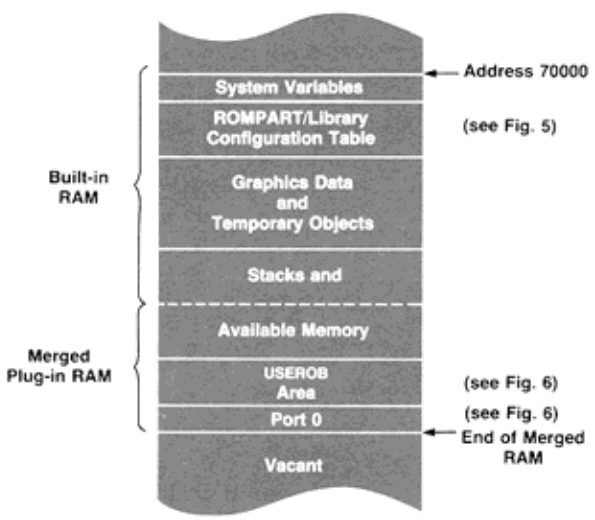


Fig. 4. *Overview of the layout of HP 48SX main RAM with one merged RAM card.*

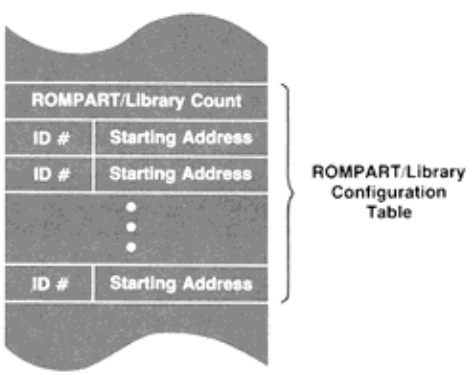


Fig. 5. *The ROMPART/library configuration table. Whenever the HP 48SX turns on, all ROMPARTs and libraries in ROM, port 0, and other unmerged plug-ins are registered here.*

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

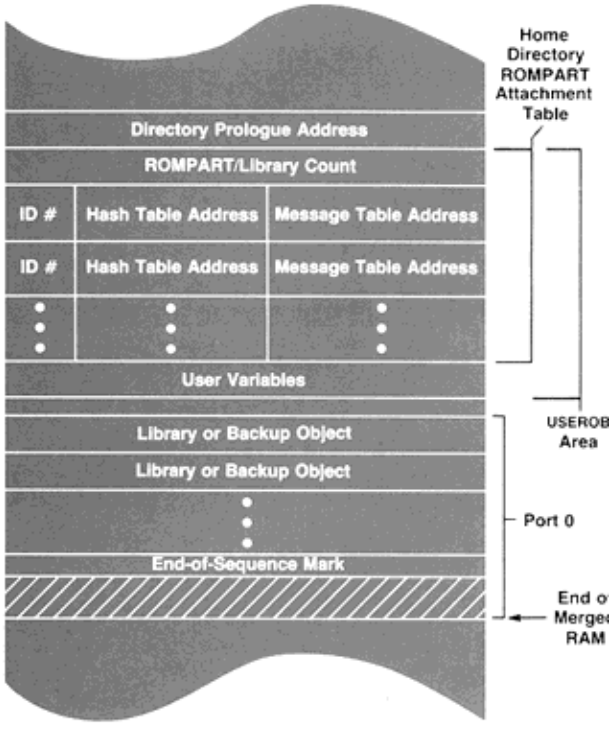


Fig. 6. *Layout of the* USEROB *area and port 0. A library or ROMPART attached to the home directory ROMPART attachment table, either by the* ATTACH *command or during the ROM poll, will have its keywords recognized by the system and can have both its keywords and its messages localized (for example,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

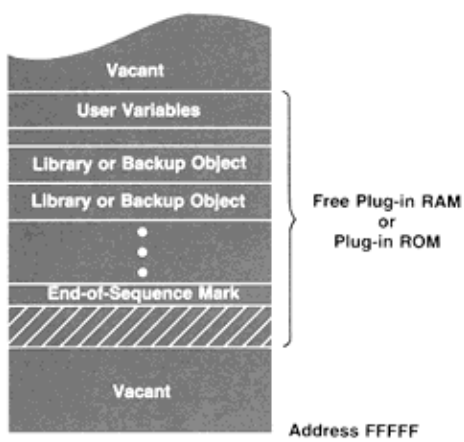


Fig. 7. *Layout of an unmerged (free) plug-in RAM card within the HP 48SX address space. An unmerged plug-in card is either port 1 or port 2 and has the same structure as port 0.*

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:

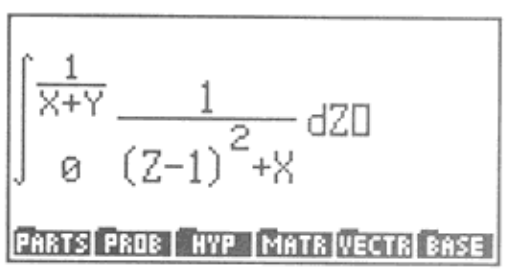
- Numerators above denominators, separated by a horizontal line
- Exponents written as superscripts, in a smaller font
- Parentheses of adjustable height
- The use of standard symbols for integral, summation, etc.

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

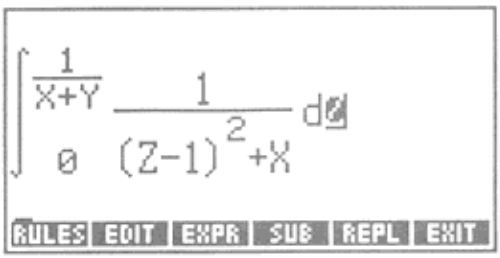
$$\int(0,1/(X+Y),1/(1+((Z-1)\hat{}2+X)),Z)$$

is terribly tedious to read and understand, compared to

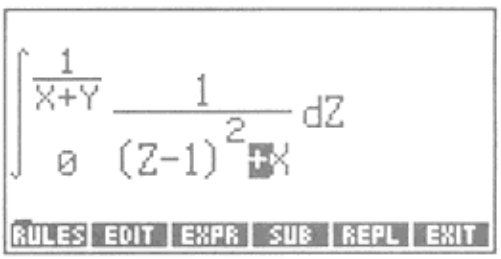


This problem was especially onerous for the HP 28S FORM interface (renamed RULES in the HP 48SX), which applies operations like commutation, association, and distribution to subexpressions of a given expression. Locating

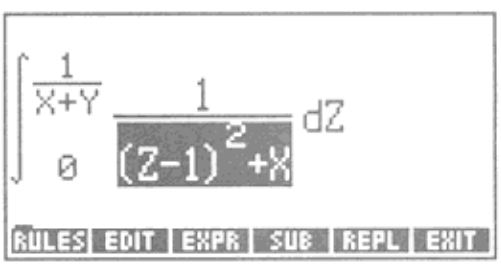
the desired subexpression (which very likely did not even fit on the screen) amid the plethora of parentheses, and recognizing its relation to the likewise messy and stretched-out result, was too much for many users to bother with. In the HP 48SX, the RULES interface is a subsystem of the EquationWriter, 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:



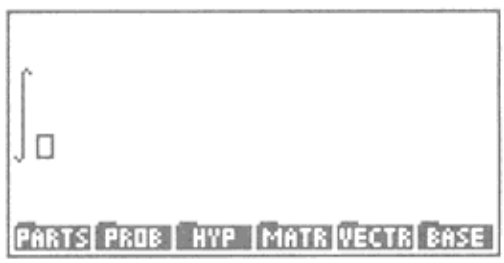
The subexpression selected for an operation is that which is included in the range of a highlighted operator (if a variable or number is highlighted, the subexpression simply consists of that object alone). A menu key toggles between highlighting the individual object and the selected subexpression.



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

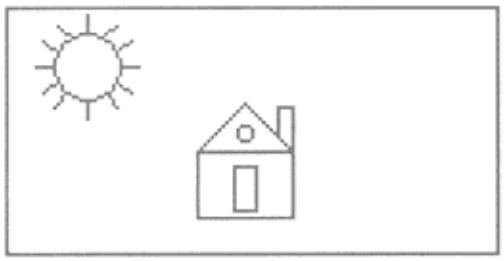
Scientists and engineers use graphics for many aspects of their work: describing problems, studying functions, working out solutions, presenting data, and so on. Our main goal for the HP 48SX graphics and plotting system was to offer plotting tools beyond those of the HP 28S, which provided function 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:

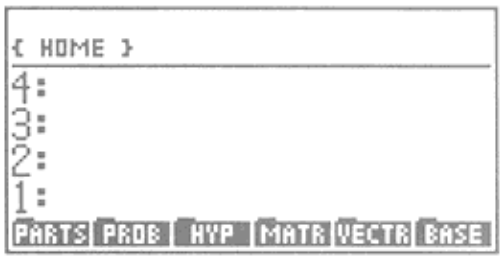
The REPL (replace) command takes a GROB from the stack and pastes it into the PICT GROB. The next example includes a label made from a string (by the →GROB command) and a picture imported from a computer.

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.

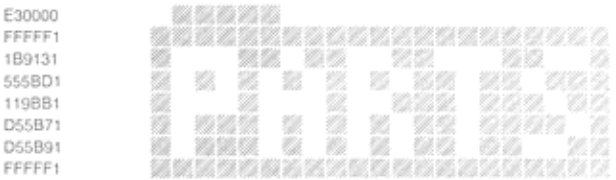
An example of the internal structure of a GROB can be seen in the PARTS menu label in the MATH menu. The calculator creates a small GROB for this label and then pastes that GROB onto the larger GROB for the whole display.



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:

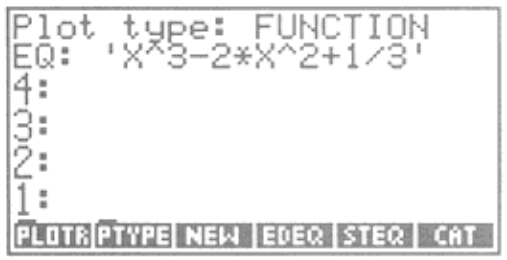


Each hexadecimal digit represents a horizontal sequence of four pixels, with the least-significant bit representing the leftmost pixel. For example, the hexadecimal digit E, written as 1110 in base two, represents four pixels: off, on, on, on.

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.

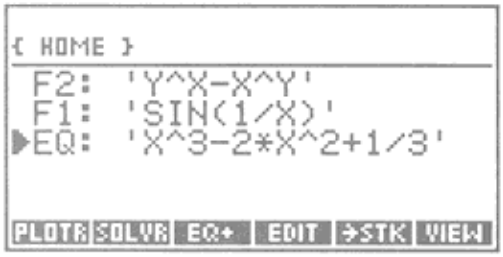
Function Plots

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):



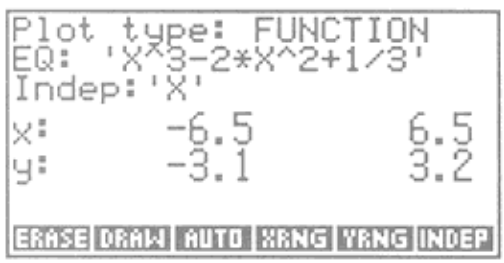
We first demonstrate the plot type FUNCTION, which is an enhanced form of the HP 28S plot type. Later we will show the results from other plot types.

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.

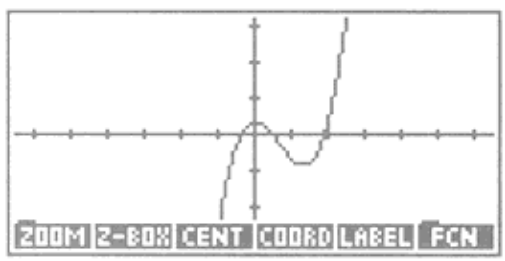
After selecting the equation, the user can press PTYPE to change the plot type. Other plot parameters control the plot's scale and the placement and labeling of the axes. To change the other plot parameters, the user presses PLOTR:



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.

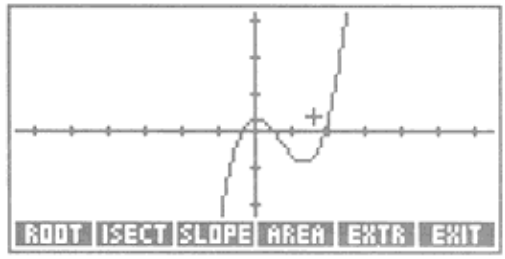
When the plot parameters are set, the user can press ERASE to clear PICT, or skip this step to superimpose the plot on the current PICT. The plotting is started by pressing DRAW or AUTO; the latter attempts to scale one or both axes automatically, according to the plot type.

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

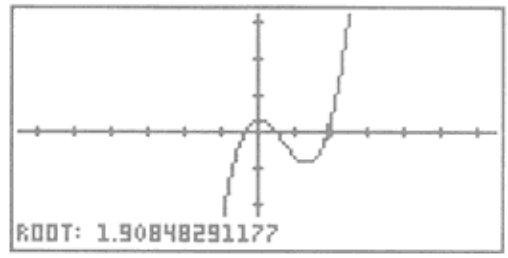


In the center of the display is a cross-shaped cursor, which the user moves by pressing the arrow keys. The cursor is used to specify locations for a variety of plotting and graphics operations. Pressing COORD causes a display of the cursor coordinates to replace the menu labels. If the PICT GROB is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

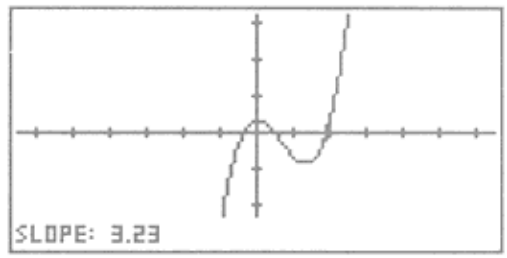
If the plot needs adjusting, the user can zoom in or out along either or both axes, or define a new center. If the plot is satisfactory, the user can press FCN to show a menu of mathematical tools (applicable only to the FUNCTION plot type):



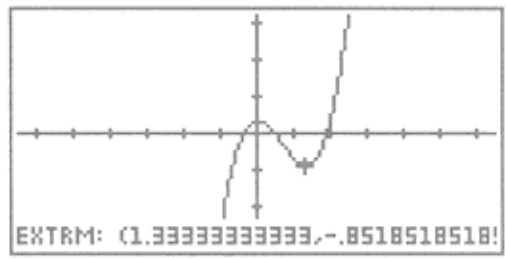
With these tools the user can analyze the function without leaving the interactive graphics environment. For example, pressing ROOT invokes the solver to find the nearest root (the cursor moves to the root):



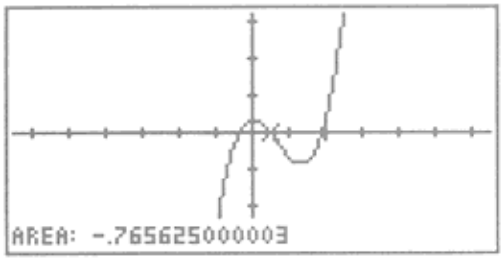
Pressing SLOPE invokes differentiation to find the derivative at the cursor's location:



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.

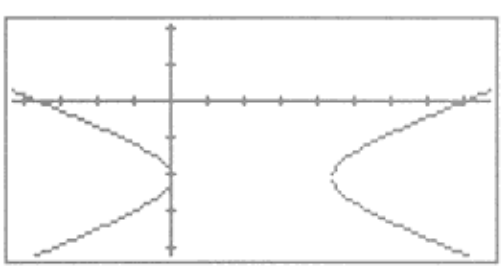
Other Plot Types

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
```

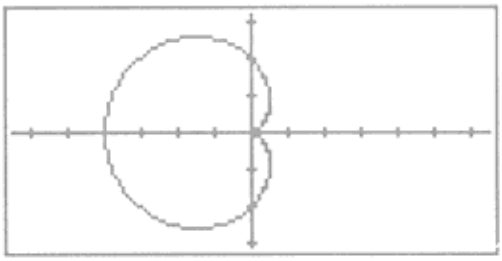
is plotted as:



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

```
2•(1-COS(X))
```

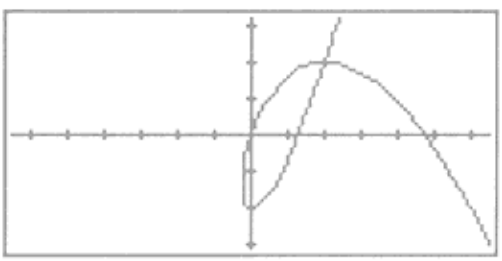
is plotted as:



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)
```

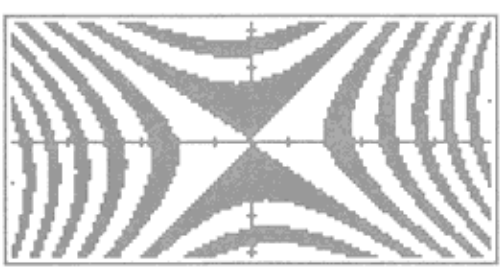
is plotted as:



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) MOD 16>8
```

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

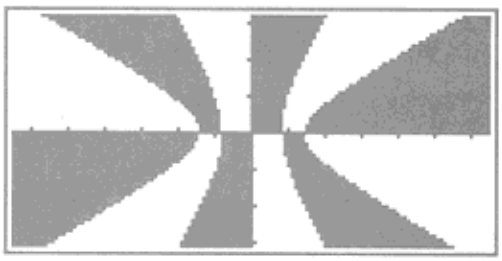


A second example is a complex function. Plotting the expression

```
SIGN (RE(Z^3-2•Z))==SIGN (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

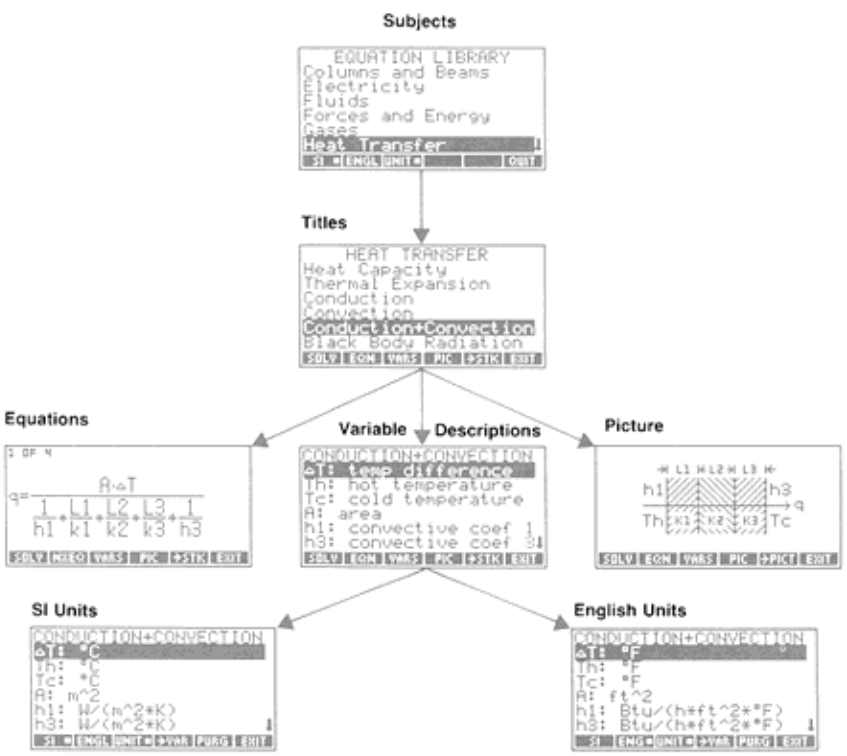


Fig. 1. *Equation Library user interface.*

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.

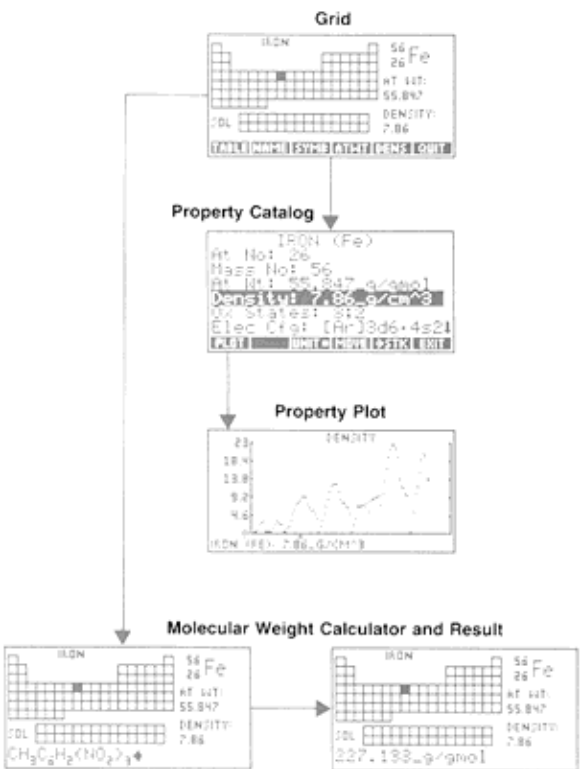


Fig. 2. *Periodic Table user interface.*

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-

$$x = x0 + v0 \cdot t + \frac{1}{2} \cdot a \cdot t^2 \qquad v = v0 + a \cdot t$$

$$x = x0 + v \cdot t - \frac{1}{2} \cdot a \cdot t^2$$

$$x = x0 + \frac{1}{2} \cdot (v0 + v) \cdot t$$

(a) Linear Motion

$$V = I \cdot R \qquad P = V \cdot I \qquad P = I^2 \cdot R \qquad P = \frac{V^2}{R}$$

(b) Ohm's Law and Power

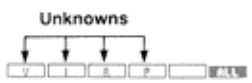
Fig. 3. *Examples of common equation sets.*

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:

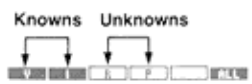
- Use the known variables to select an equation containing only one unknown.
- Solve for the unknown.
- Add the variable just calculated to the set of known variables.
- Use the combined set of known and calculated variables to select another equation containing only one unknown.
- Solve for the unknown.
- 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.

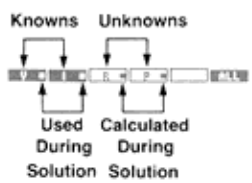
The barrier to proper functioning of the MES was identifying whether a variable is known or unknown. The existence of the variable alone is not sufficient—after a solution has been determined, all the variables exist. The key underlying principle is that the state of a variable (known or unknown) is 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.



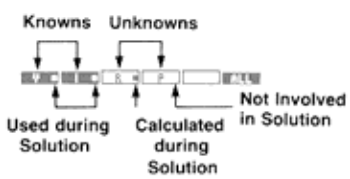
(a) Initial menu for Ohm's Law and Power



(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

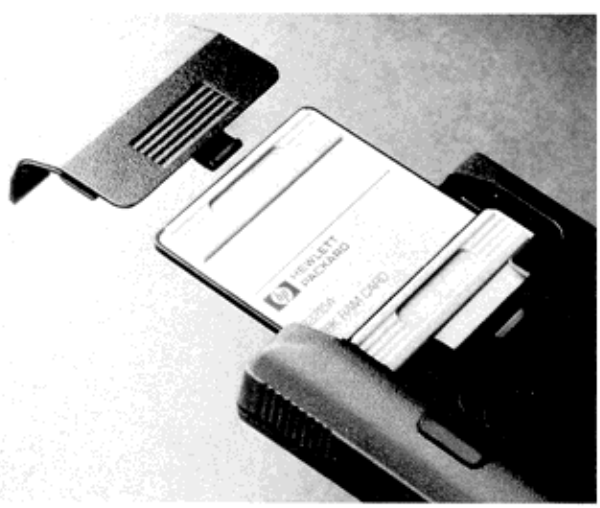


Fig. 1. *Two expansion ports accept ROM or battery-backed RAM cards.*

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)

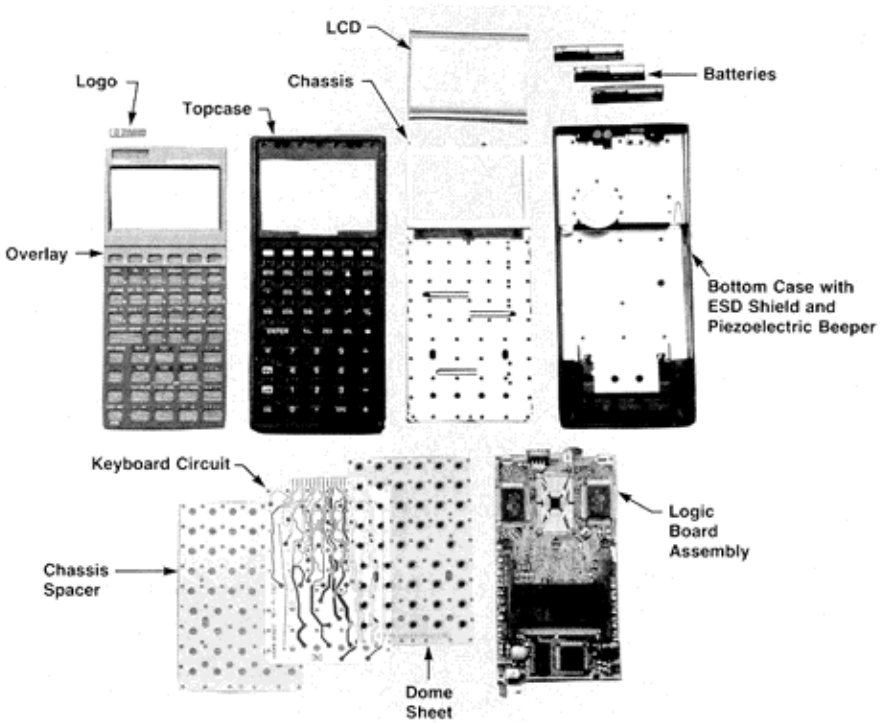


Fig. 2. *The three HP 48SX subassemblies are the topcase, the printed circuit assembly, and the bottom case. They are built from 24 mechanical parts and various ICs, discrete components, and surface mount devices.*

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

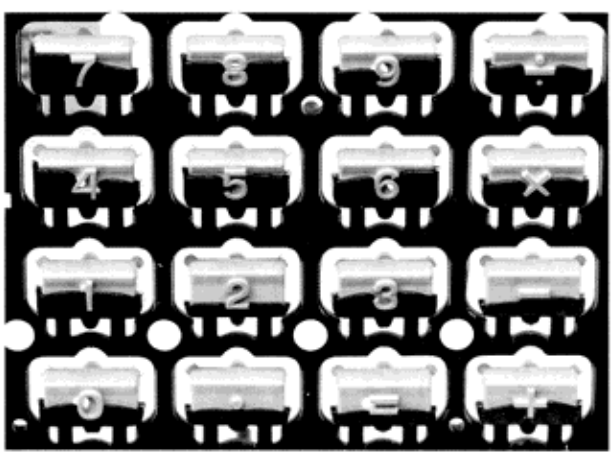


Fig. 3. *A "short-shot" photo of the HP 48SX keys. A separate molded part for each key produces the key nomenclature. The topcase is molded around the key nomenclature. Notice the hole in the* **9** *key, which is formed by a moving core pin. This hole allows the second-shot plastic (dark) to flow into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

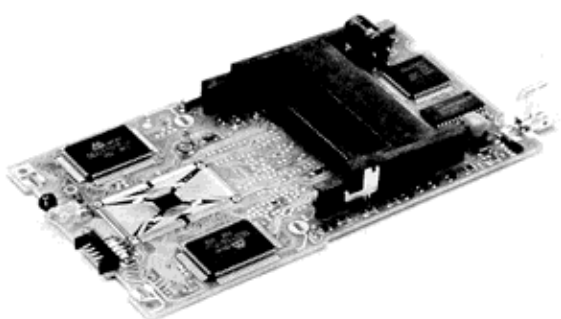


Fig. 4. *HP 48SX logic printed circuit assembly.*

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.

Mechanical Design of the HP 48SX Memory Card and Memory Card Connector

The mechanical implementation of the HP 48SX memory card and memory card connector addresses the goal of design and manufacturing leverage. For the HP 48SX, R&D was faced with the challenge of incorporating plug-in modules into a calculator for the first time since the HP 71B (introduced in February 1984). The product required either development of a custom module such as those for the HP 41C, HP 75C, and HP 71B, or the purchase of an OEM module. The decision was made to source and qualify a module produced by an outside company to minimize HP design effort. As the search progressed, it became apparent that credit-card-size digital memory modules known as memory cards were becoming a loosely defined standard among electronic equipment manufacturers. By choosing this memory card form factor, we were able to use the expertise and resources of numerous manufacturers that were already in high-volume production. Other objectives, such as second-sourcing options and possible adherence to a future industry standard, could also be addressed.

Selection

The selection of one memory card manufacturer from a list of a dozen was assisted by other HP divisions. During initial supplier research, some of the manufacturers commented that the Rohnert Park, Spokane, Vancouver, Boise, San Diego, and McMinnville sites were all investigating memory cards. Information about card contact plating and connector soldering was obtained from Rohnert Park, while Spokane provided comparative data regarding reliability and price. After reviewing design specifications, reliability data, and quotations, R&D managers visited four manufacturers. One was selected to produce memory cards for the HP 48SX. The chosen design offers electrostatic discharge protection, mechanical strength, and long insertion life.

All of the suppliers offered memory card connectors designed for wave soldering and for strain relief using screws. To meet the goal of manufacturing leverage, the connector for the HP 48SX had to be surface mounted, be suitable for infrared reflow soldering, and be strain relieved using heatstakes. Taking into consideration the added design requirements of low height and accommodation of two memory cards, the need to tool a custom connector became apparent. The selected card manufacturer gave permission to start engineering discussion directly with the connector company that produced the standard connector designed for use with the card. By developing the new connector with the existing supplier, we gained access to data regarding design criteria, materials, and reliability testing without extensive investigation.

Card Design

Basically, the memory card is an expansion board packaged to provide protection from environmental factors. It consists of a printed circuit board, a frame, and steel panels. The card design chosen uses an edge-type contact in which 40 gold-plated fingers on a 0.017-inch-thick FR-4 printed circuit board are located on a 0.050-inch pitch. To maintain thinness, surface mount discrete devices and chip-on-board integrated circuits are mounted on the printed circuit board. For rigidity, the card employs a sandwich construction. The printed circuit board is laminated to an injection-molded PBT frame on the noncontact side. Two 0.010-inch full hard stainless steel panels are laminated to the frame on one side and to the printed circuit board on the other. A 0.058-inch-diameter compression coil spring mounted through the printed circuit board electrically ties the two panels together to provide ESD protection. The contact-side panel traps a 0.005-inch-thick stainless steel shutter and two 0.039-inch-diameter tension coil springs. The springs keep the shutter closed over the contact fingers when the card is unplugged, providing protection from ESD and contamination. When the card is inserted into a connector, two formed tabs that are part of the shutter are pushed back to expose the contacts.

While the ROM card and SRAM card are similar in construction, the SRAM card incorporates several additonal parts. Two 0.004-inch-thick phosphor bronze flat contacts for a CR2016 lithium coin battery are heatstaked to the frame and make pressure contact to the printed circuit board. An injection-molded battery tray slides between the two panels and allows installation and replacement of the coin cell. A write protect switch consisting of a molded button and a flat spring is also trapped between the panels. The position of the tray and switch on the top edge of the card allows access while the SRAM card is plugged into the HP 48SX. This is necessary because the SRAM card must be powered by the calculator during battery replacement for memory retention.

The dimensions of the card are identical to those of a credit card at 3.370 inches by 2.126 inches. Overall thickness is 0.094 inch, which allows the HP 48SX to accommodate two stacked cards in the same thickness as one HP 41C module. The SRAM card weighs less than nine tenths of an ounce including the battery.

Although produced on the manufacturer's standard assembly line, memory cards for the HP 48SX incorporate several custom details for improved performance. The gold plating on the fingers is thicker for increased contact reliability during insertion. The ground finger is longer than the standard printed circuit board's to provide clean shutdown and startup of the SRAM card during removal and insertion. An injection-molded ABS grip is applied by the manufacturer to the front panel. The grip, which facilitates card removal from the calculator, was tooled and color matched by the manufacturer to HP specifications. The panels are offset printed using specified artwork films and colors.

The memory card chosen thus leverages the manufacturer's expertise in providing mechanical integrity and ESD protection. Custom design details are incorporated which, although available through resources already in place at the manufacturer, add significantly to customer satisfaction.

Connector Design and Assembly

The memory card connector serves as the interface between two cards and the HP 48SX logic printed circuit board. It consists of a body, contact pins, ground pins, and an alignment comb. The body requires an injection mold with four slides from which air is evacuated before injection of the heat-resistant 40% glass-filled PPS to facilitate filling such details as the 0.012-inch-thick ribs that fit between each contact pin. The body holds two memory cards in stacked configuration, offset 0.390 inch to expose both card grips. Four heatstake-pin/reinforcing-rib pairs fit into matching printed circuit board slots to hold the connector against the printed circuit board during the 437°F infrared reflow process and to provide strain relief for the solder joints when cards are inserted. The 80 contact pins are arranged in two rows of 40 each. The full hard phosphor bronze pins, each 0.010 inch wide by 0.010 inch thick, are press fit into holes in the body on a

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:

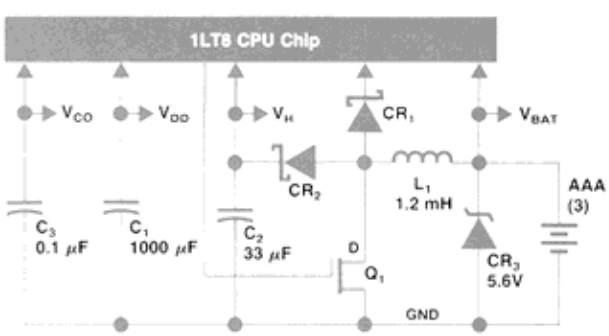


Fig. 5. *HP 48SX power supply schematic diagram.*

- 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-

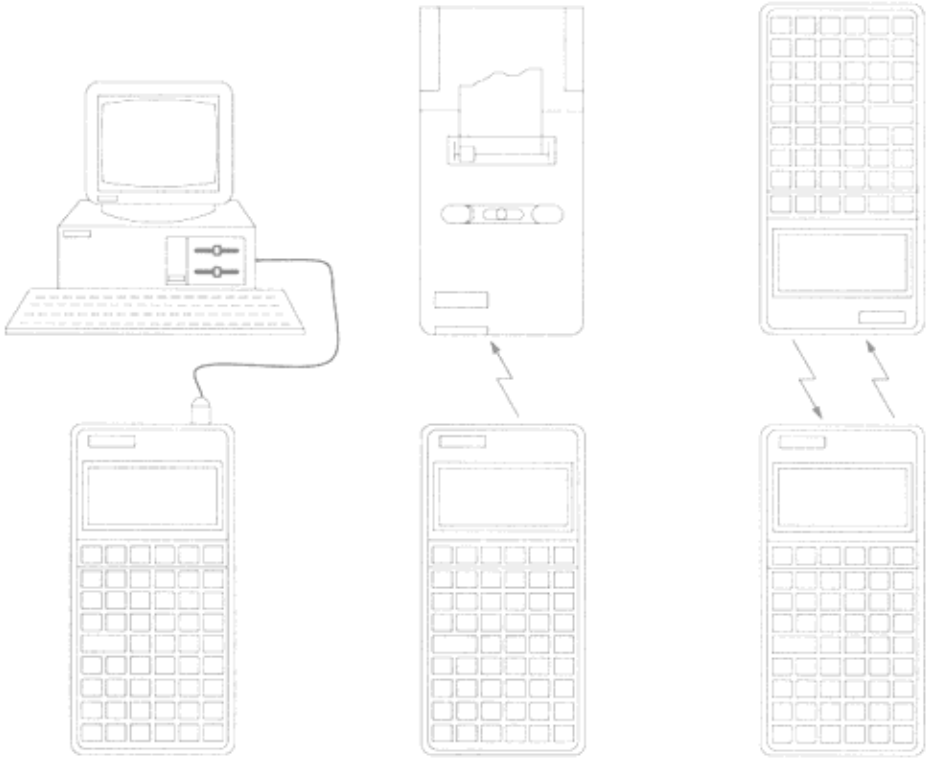


Fig. 1. *The HP 48SX calculator can communicate with personal computers or other RS-232 devices, with the HP 82240 portable infrared printer, and with another HP 48SX via a short-range two-way infrared link.*

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

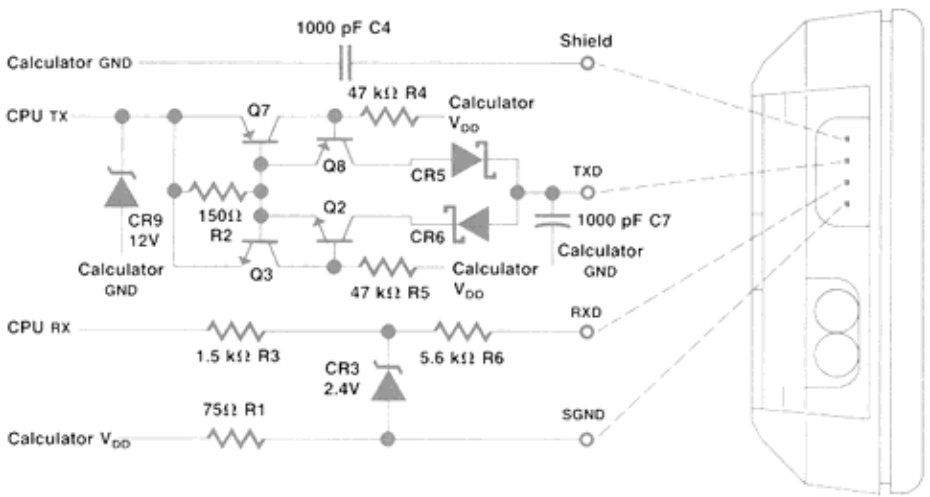


Fig. 2. *The discrete circuitry for the HP 48SX serial port provides protection for the calculator and compatibility with RS-232.*

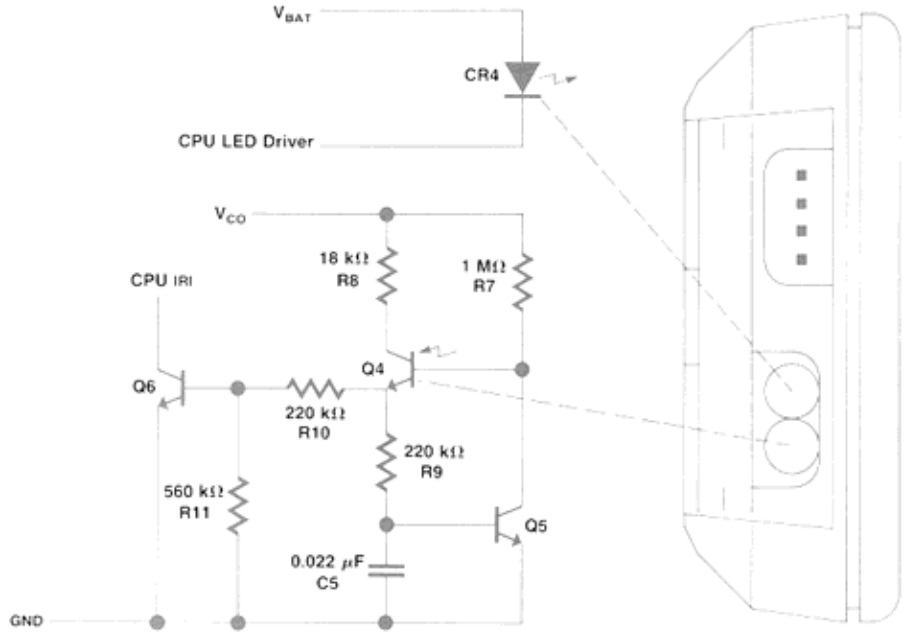


Fig. 3. *The infrared circuitry transmits to the portable infrared printer and provides two-way communication with another HP 48SX.*

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.

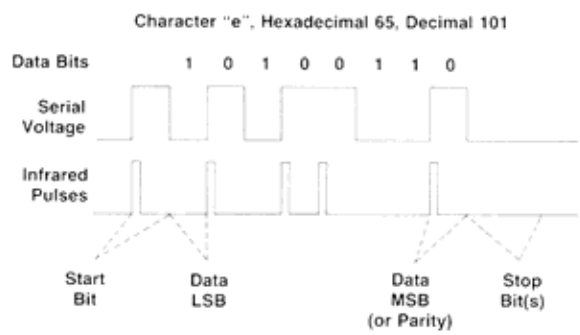


Fig. 4. *Coding of data in the two-way infrared protocol is similar to that in the serial link. A logic zero has a pulse at the beginning of the bit time; a logic one bit has no pulse.*

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.

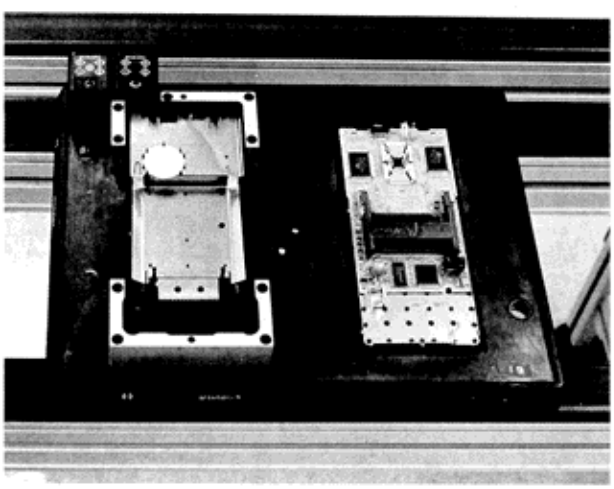


Fig. 1. *Assemblies are carried on pallets that ride on conveyor belts. The line has seven robot stations.*

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

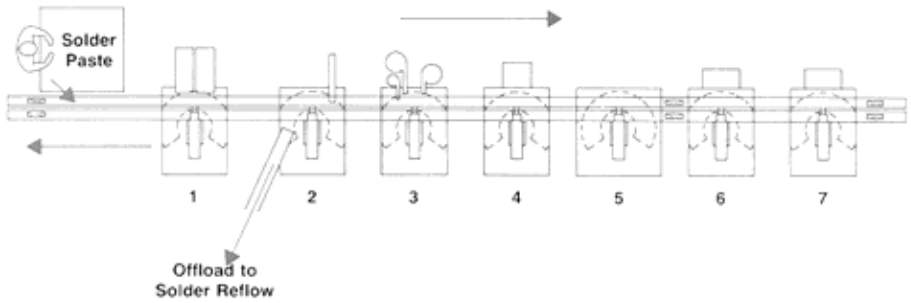


Fig. 2. *HP 48SX printed circuit assembly production line layout.*

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.

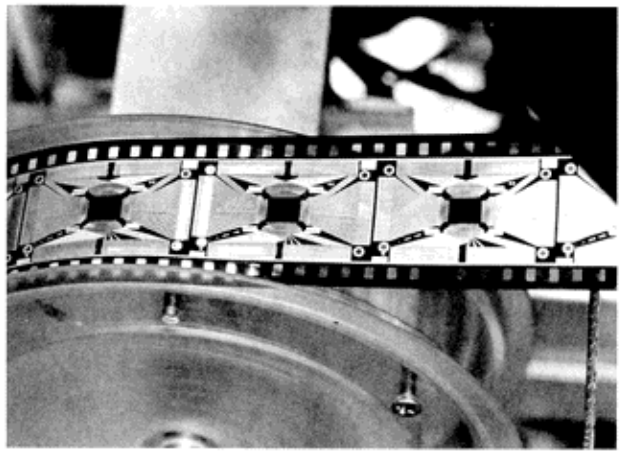


Fig. 3. *1LT8 ICs are packaged for tape automated bonding (TAB) and can be handled like movie film.*

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,

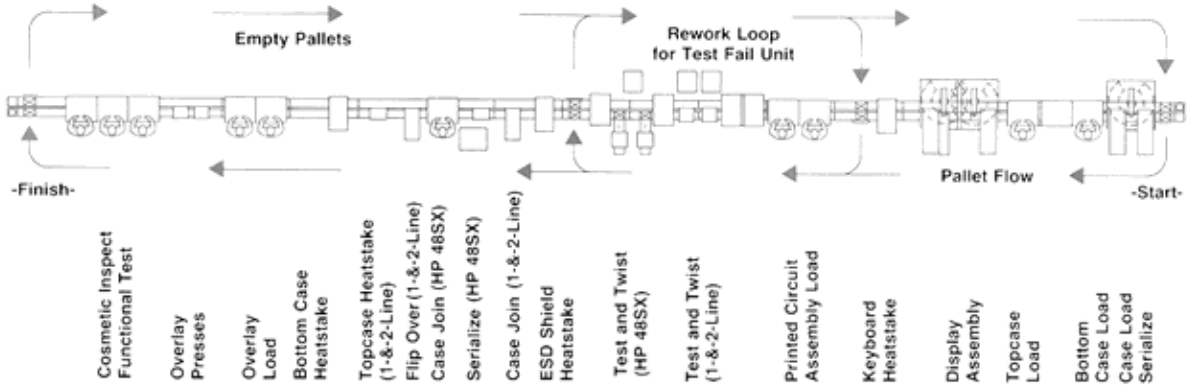


Fig. 4. *HP 48SX final assembly line layout.*

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.

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.

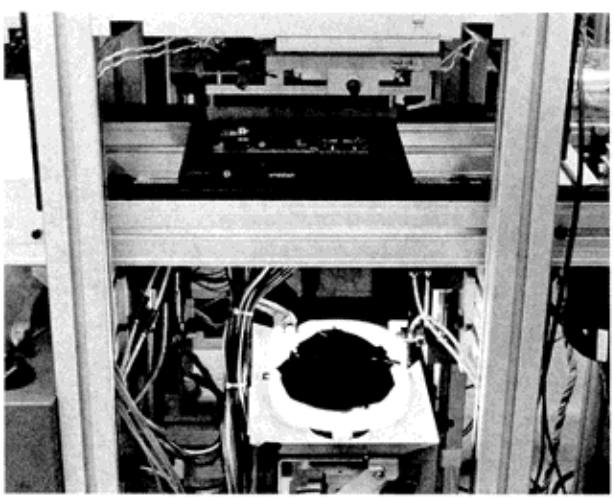


Fig. 5. *This tester uses two cameras and an automated vision system to inspect the calculator's display.*