An Advanced Scientific Graphing Calculator

The HP 48G/GX combines an easy-to-learn graphical user interface with advanced mathematics and engineering functionality, expanded memory capability, and seven new plot types.

by Diana K. Byrne, Charles M. Patton, David Arnett, Ted W. Beers, and Paul J. McClellan

The HP 48G/GX, Fig. 1, is a state-of-the-art graphing calculator that combines an easy-to-learn graphical user interface with advanced mathematics and engineering functionality. It is a continuation of the HP 28S[1] and HP 48S/SX[2] series of calculators, which are designed for high power, extendability, and customizability.

The HP 48G/GX includes improvements to address the needs of both novice and advanced users of scientific and graphing calculators. For the new user or the user who does not use certain functionality very often, the calculator has a dialog-box-style, fill-in-the-blanks user interface.

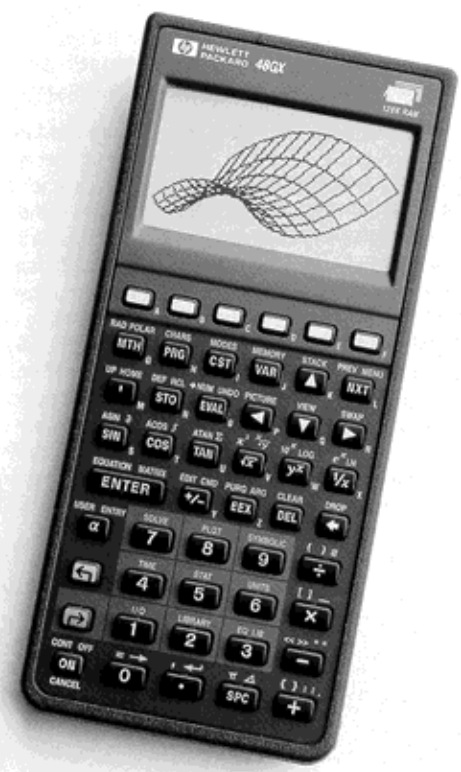

Fig. 1. HP 48GX scientific graphing calculator.

For the user who needs to do advanced problem solving, the calculator offers the following features:

- Differential equation solvers
- Polynomial root finder
- Financial problem solver
- Library of engineering equations and constants
- Fourier transforms
- Matrix manipulations
- Linear algebra operations.

For the user who needs more memory and extendability, the GX version has 128K bytes of built-in RAM, compared to 32K bytes in the S and G versions. The only other difference between the G and the GX is the two memory card slots in the HP 48GX. The second memory card slot accepts up to 4M bytes of RAM or ROM.

The graphing capability has been expanded with the addition of seven new plot types, for a total of fifteen. The HP 48S/SX has function, polar, parametric, conic, truth, histogram, bar, and scatter plots and the HP 48G/GX has these plus differential equation, slope field, wireframe, parametric surface, grid map, pseudocontour, and y-slice cross-section plots.

The HP 48S/SX functionality has been retained in the HP 48G/GX. The new calculator has twice the ROM and retains much of the original code. The HP 48S/SX and HP 48G/GX both have the following features:[2]

- Unit management
- MatrixWriter
- EquationWriter
- HP Solve numeric solver
- RPN-style stack calculation
- Symbolic mathematics
- Time and alarms
- Statistics operations
- Variables and directories for data storage
- User-definable keyboard and custom menus
- RPL programming language
- Two-way infrared communications link
- RS-232 serial cable connector.

Education Trends

The creation of the HP 48G/GX can be traced to the American Mathematical Society (AMS) meeting in January 1992. We had been closely following the trend of using technology in the mathematics classroom because our software team

includes former mathematics educators and a calculus textbook author, and because we visit mathematics, engineering, and education conferences and talk to educators both to promote existing products and to find out what teachers and students would like to see in future products.

We had watched the interest in graphing calculators grow steadily each year, and had been involved in workshops for teachers using technology in the classroom through an HP grants program. Although the HP 48S/SX was becoming a standard for engineering students and professional engineers, it was just the first step in meeting the needs of the education community, and we continued to hear that it was too difficult to use and too expensive for classroom use. By January 1992, when we talked to educators attending the AMS meeting about their needs and about the calculators that they were considering for use in the classroom, it became obvious that we needed to have a new product for the education market no later than the 1993 back-to-school period. This resulted in the formation of an education advisory committee, a group of six mathematics professors who would help us design the calculator to fit the needs of the education community, and then give us feedback on our implementation.

Design Objectives

Our number one objective for the new calculator was ease of learning. The users of the HP 48S/SX told us that they appreciated its power, but its complexity made it difficult for both novice users and experienced users. The novices tended to be intimidated by the extensive owner's manual and the difficulty of mastering so many new operations, while the experienced users had a hard time remembering how to use some operations that they did not use frequently.

Creating a state-of-the art graphing calculator was our second objective. We needed to add some graphing capabilities, such as tracing along a graph and shading between graphs, just to maintain parity with other graphing calculators, but we also wanted to go far beyond the competition with features such as 3D graphing and animation.

Our third objective was to enhance the high-end mathematics capability of the HP 48S/SX with the addition of features such as differential equation solving, polynomial root finding, more matrix operations, and Fourier transforms, thereby strengthening our position as the most powerful technical calculator in the world.

By offering two models, the HP 48G and the HP 48GX, we intended to please customers at both ends of the education spectrum. The HP 48G has a list price that represents a substantial decrease from the list price of the HP 48SX or HP 48S. This makes the HP 48G competitive with other graphing calculators and makes it appealing even at the high school level. The HP 48GX has more appeal for college students, with four times as much built-in memory and two plug-in card ports that are expandable to 4M bytes of memory.

Operating System

A calculator or computer operating system is primarily a set of conventions for memory organization, data structures, and resource allocation together with a set of software tools to aid in performing operations in accordance with those conventions. In contrast, an application is software built using the resources and conventions of the operating system. As new hardware resources become necessary and available, the operating system must grow to manage those resources effectively and as transparently as possible to the applications built on the system.

The operating system (and system programming language) in the HP 48G/GX is the RPL operating system, first used in the HP 18C and HP 28C and subsequently in a number of other machines including the HP 28S, HP 48S/SX, and now, with extensions, in the HP 48G/GX.

HP 48G/GX Fundamentals

The key concept underlying the operation of the calculator is the idea of objects on the stack. A stack is a data structure that is similar to a stack of cafeteria trays. The clean trays are added to the top of the stack, and as trays are needed, they are removed from the top of the stack. This type of last in, first out ordering characterizes the HP 48G/GX stack. All operations take their arguments (if any) from the stack and return their results (if any) to the stack.

There is only one data stack in the HP 48G/GX. This resource is shared by the user and the system RPL programmer, who must take great care to make sure that any objects that belong to the user are preserved through the operation of system RPL programs. For example, the user may have a few numbers sitting on the stack, then decide to plot the graph of a function. The system RPL program that runs when the DRAW key is pressed does many operations that require the use of the stack, such as recalling the plotting parameters, checking that they are valid, calculating the range over which to plot, evaluating the user's function, and converting the function values to pixel coordinates. After the graph is complete (or if the drawing of the graph is interrupted by the user), when the user sees the stack again, the same numbers that were there to begin with should not have been disturbed.

Instead of trays, users may collect various types of numeric, symbolic, and graphic objects on the HP 48G/GX stack. The types of objects available in the HP 48G/GX include real and complex numbers, real and complex arrays, binary integers, names, characters, strings, tagged objets, algebraic objects, unit objects, and graphic objects. There are also backup objects, library objects, directories, programs, and lists. (HP 48 object types are discussed in more detail in reference 2.)

In a key-per-function calculator, there is a single key that the user needs to press to get the machine to perform any operation, such as cosine. The HP 48G/GX has many more operations than the 49 keys on the keyboard, so there needs to be a way to access all the functionality without assigning one operation to each key on the keyboard. This is accomplished through the use of menus and softkeys. The top row of keys on the keyboard do not have anything printed on them because they correspond to menu labels that appear along the bottom of the screen. These keys are called softkeys, and their meaning changes whenever the corresponding labels on the screen are changed by the software.

HP 48S/SX Memory Controller Configurations

We will now discuss the memory controller configurations used in the HP 48S/SX and how these are used in implementing the various types of expanded address modes developed for these products. The next section outlines the differences

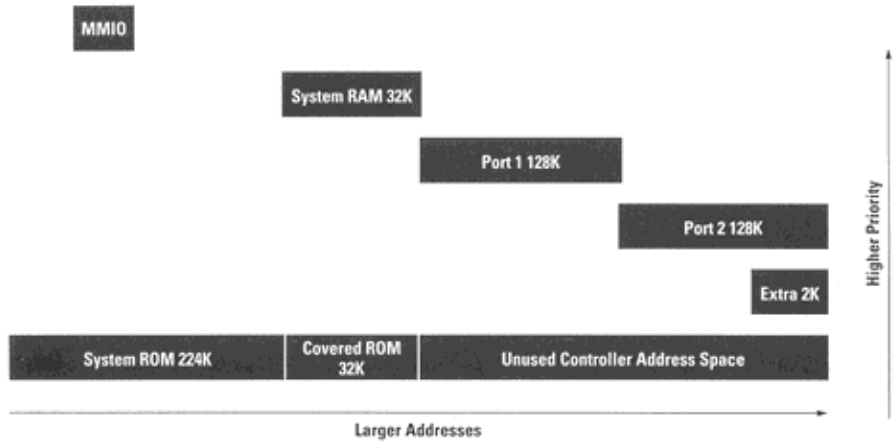

Fig. 2. Standard memory controller configuration for the HP 48S/SX calculator. Memory sizes are in bytes.

in configuration between the HP 48S/SX and the HP 48G/GX and discusses how these differences are used to extend and refine the expanded address technology to provide access to a total of 4.75M bytes of code and data as transparently as possible.

The CPU bus architecture first developed for the HP 71 and used in all HP calculators since that time has several useful features. One of the nicest is its address configuration capabilities. All chips attached to the bus are required to be able to change, on command of the bus, the range of addresses that evoke a response from the chips. Such a system eliminates, once and for all, the inconvenience and headache of configuring jumper switches on cards designed to plug into the machine. For a consumer product like a calculator this is not only a nicety, it is a necessity.

In the early days of the architecture (HP 71 to HP 28C), the CPU bus lines were actually routed around the circuit board and any RAM, ROM, or memory mapped I/O that was attached to the bus had to be custom-made with the bus interface attached. This had the advantage of allowing an arbitrary number of parts to be added to the system with assurance that the system would be capable of handling all of them in one way or another. It had the grave disadvantage of putting a price premium on such essential items as ROM and RAM.

In the second-generation CPU chip, a fixed number of memory controllers were included onboard the CPU. The CPU bus was then, for all practical purposes, completely hidden within the CPU itself. The combination of external standard RAM or ROM together with one of the internal memory controllers was then equivalent (so far as the CPU bus is concerned) to a standard bus device.

In the standard device implementations, the size of the device (that is, the address space occupied by the device) is designed into the device. In the second-generation chip, the size of the controllers was mask programmed at the time of manufacture since we knew exactly what size each controlled device would be.

With the advent of plug-ins for the HP 48S/SX, the configuration capabilities of the memory controllers had to be expanded to include varying the apparent size of the memory controller to conform with the device being plugged in. This is one of the many advanced features in the third-generation, HP 48S/SX implementation of the architecture. This resizing feature, in addition to allowing plug-ins of various sizes, also presented the opportunity to explore expanded address modes, which we have come to call the "covered" technology, for reasons that will be apparent shortly.

The third-generation CPU chip has six memory controllers. In the HP 48SX, these are allocated to memory mapped I/O, system RAM, port 1, port 2, and system ROM, and there is one extra controller. Their configuration in the usual state is shown in Fig. 2. The memory controllers are shown with their sizes and locations in the address space (00000h to FFFFFh). They are also pictured as having a vertical location in "priority space." In the CPU bus definition the devices are chained, with the result that devices closest to the CPU on

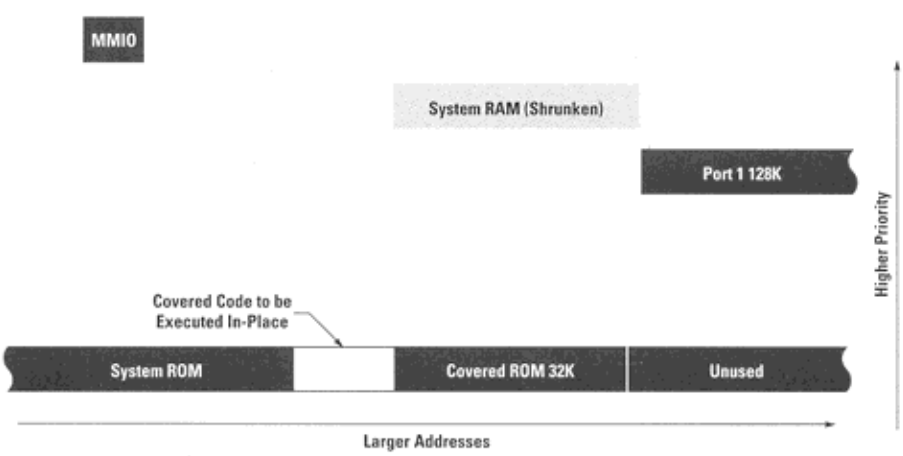

Fig. 3. Execute-in-place configuration for HP 48S/SX covered code.

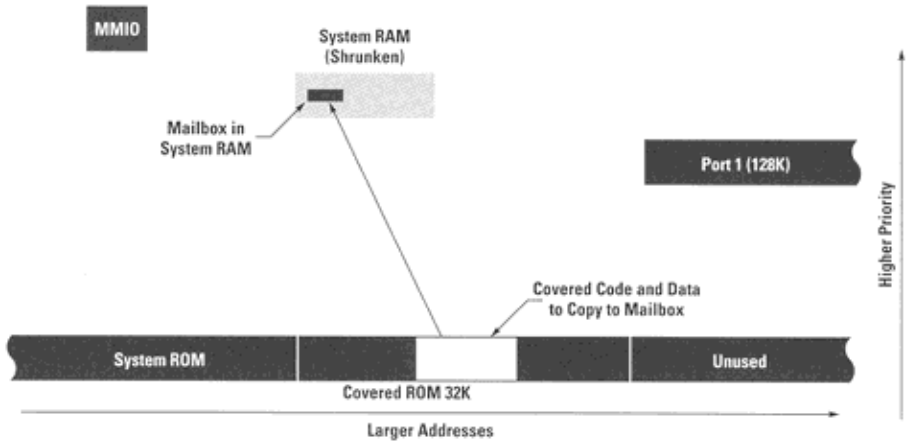

Fig. 4. Copy-to-mailbox configuration for HP 48S/SX covered data.

the chain have the first opportunity to respond to bus requests. In consequence, if two devices are configured with overlapping address ranges, the one closer to the CPU on the chain effectively hides the more distant one. In Figs. 2 to 12, higher priority can be interpreted as "closer to the CPU" or "hides those below."

As shown in Fig. 2, the memory controller for system RAM hides the section of ROM shown as covered. This is the reason for the name "covered" technology.

Fig. 3 shows more detail of the covered ROM and the first way in which it is used. In one section of the covered ROM there is assembly language code (mostly math routines) that requires no RAM resources outside the CPU for execution. This code is executed in-place in the covered ROM by shrinking and/or moving the memory controller for system RAM so that the relevant section of code is temporarily uncovered. When the routine finishes execution, system RAM is returned to its normal configuration.

A second set of routines, all of which only need access to a fixed set of locations within system RAM, can execute with system RAM in any one of 16 locations, as long as they themselves are not currently covered by system RAM.

Fig. 4 shows a second way in which the covered ROM is used. In this case, code and data (mostly data) are copied from covered ROM to a mailbox at a fixed location in system RAM. After the copy is completed, system RAM is returned to its normal configuration and the code and data are available to the rest of the system. Coders using this data must remain aware that it is volatile and can be destroyed by another fetch of data from covered ROM. In this sense, this method is not transparent.

Another way in which covered ROM is used is shown in Fig. 5. It is as transparent as the execute-in-place method but entails fewer restrictions on the code and data that can be included. In the HP 48SX code, this system is usually tied to the execution of ROMPTRs. Recall that ROMPTRs are RPL objects that substitute for hard addresses of objects whose precise location is not known in advance (and in fact might not even be present.) They are midway between hard addresses that only change at compile/link time and identifiers whose corresponding objects may move between subsequent calls at run time.

If, during the conversion of a ROMPTR to an address, it is determined that the corresponding object lives in covered ROM, the object is copied from covered ROM, through the mailbox, to the TEMPOB (temporary object) area. The address of its new location in the TEMPOB area is then returned. Fig. 6 shows a comparison of a named ROM word (keyword or command) as it would exist in covered ROM and as copied to the TEMPOB area. Although we'll refer back to Fig. 6 later, for now notice that in addition to the object itself, an additional piece is added to the image in the TEMPOB area. This piece is a ROMPTR preceding the object itself. This allows

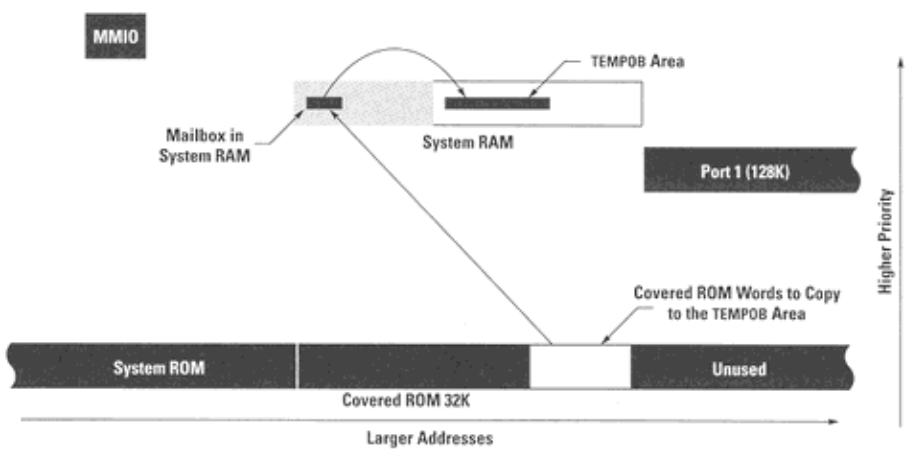

Fig. 5. Copy-to-TEMPOB configuration for HP 48S/SX covered ROM words.

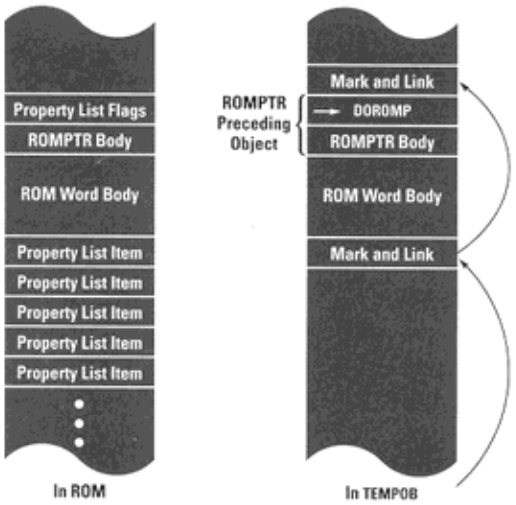

Fig. 6. Comparing the structure of a ROM word as resident in ROM and when copied to TEMPOB using covered technology.

the routine converting the ROMPTR to an address to check whether the object in question is a copy of one residing elsewhere. This method of covered ROM access, which we call "covered ROM word access," will be especially relevant to our discussion of the HP 48G/GX.

Preexisting design elements of the RPL system contributed greatly to the practicality and transparency of covered ROM word access, including:

- Encapsulation of code and data into RPL objects that are of determinable size
- Indifference of RPL object execution to object location in RAM or ROM
- Equivalence of direct and indirect execution of RPL objects, which allows (noncircular) structures to be stored and used in the same format.

HP 48G/GX Memory Controller Configurations

The HP 48GX has a number of important features including:

- Up to 128K bytes of built-in system RAM
- One plug-in port electrically equivalent to the HP 48SX ports
- Access to 512K bytes of system ROM
- Access to 4M bytes of RAM or ROM at a second port using industry-standard parts.

These features required increasing the usable address space from 0.5M bytes to 4.75M bytes, an 850% increase over previous machines.

While the HP 48G/GX has CPU functionally equivalent to the third-generation CPU discussed above and thus has six memory controllers, these controllers are configured and used differently. Fig. 7 shows the standard HP 48GX configuration. The controller previously allocated to port 2 is now used as a bank switch control, and the extra controller is now allocated to port 2. Furthermore, there are now as many as 34 layers over the last 128K bytes of address space.

Eliminated in this configuration is the HP 48S/SX covered ROM. This means that all of the functionality included in the HP 48S/SX can be accessed more quickly. Two things that are visibly enhanced are plotting (since the math routines are not covered) and screen update (since the font bitmaps are not covered.) Since there are a great many more covered places to access, however, there are many more "temporary" configurations to keep track of while working with the covered data.

To simplify the system, we use only a single covered technique, namely, covered ROM word access, with appropriate modifications. Without this simplification, the number of access method and configuration combinations would be unmanageable. Moreover, this is the only feasible method of covered access to code written for the HP 48S/SX or not expressly written for the the new configuration.

Fig. 8 shows the configuration while copying an object from a bank of port 2 to the TEMPOB area. Port 1 is unconfigured. In the unconfigured state, the controller responds to only a handful of bus commands and acts as if it weren't there for data access.

Fig. 9 shows the configuration while copying an object from the second half of the upper system ROM. In this case, both ports are unconfigured.

Fig. 10 shows the configuration while copying an object from the first half of the upper system ROM. Since a controller move or resize operation takes many more CPU resources than configure or unconfigure, it is often necessary to copy objects from this section, through a mailbox, and then into the TEMPOB area.

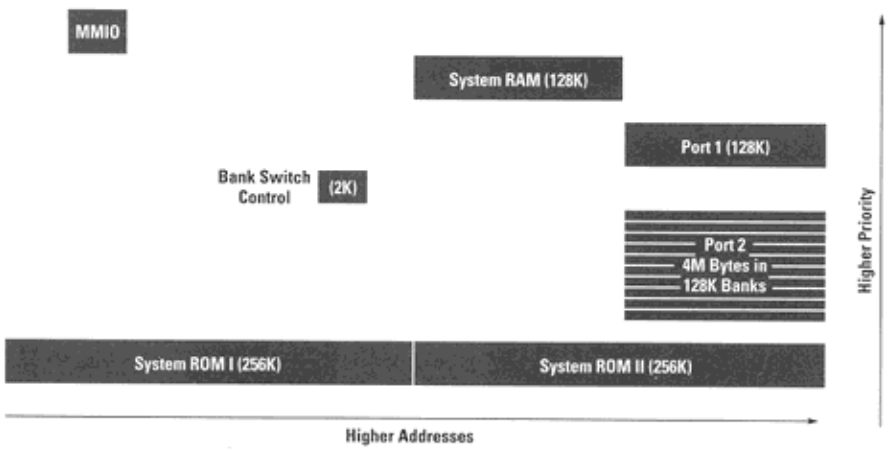

Fig. 7. HP 48GX standard memory controller configuration.

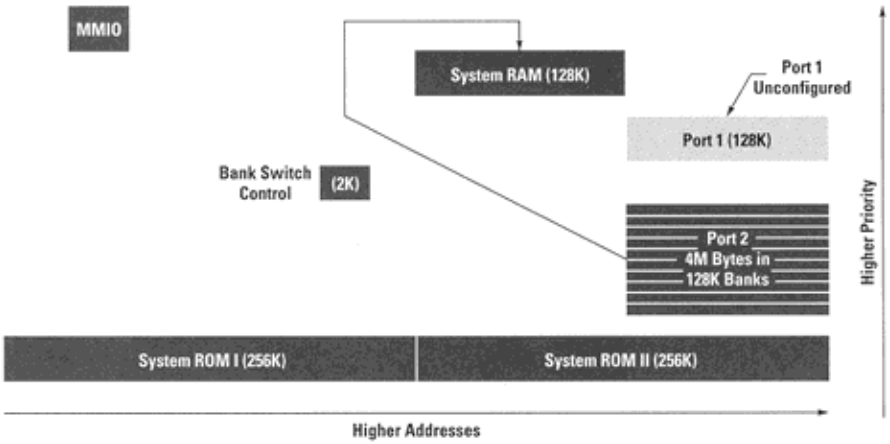

Fig. 8. HP 48GX configuration for copying an object from a bank of port 2 to TEMPOB.

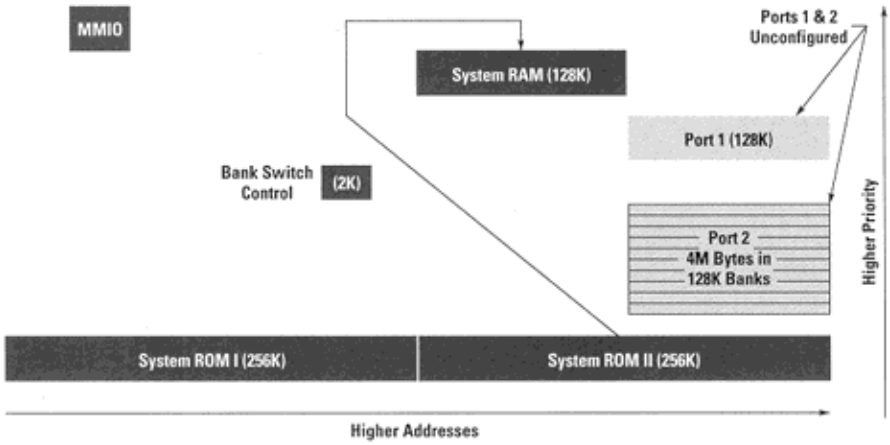

Fig. 9. HP 48GX configuration for copying an object to TEMPOB from the upper half of system ROM.

Fig. 11 shows the configuration when it is determined that nothing is plugged in at all. In this case, the only covered access is to the first half of the upper system ROM. Again, it is likely to be necessary to copy this material through a mailbox. Otherwise, all the ROM words can be executed in-place.

Fig. 12 shows the standard HP 48G configuration, which is identical to Fig. 11 except for the smaller size of system RAM. While it is not strictly necessary to use this configuration, which matches one of the HP 48GX configurations, there are advantages. First, it allows maximal code sharing between the two machines. In fact, the code can be identical in this case. Second, it gains the advantage of faster access to the base functionality, providing a more responsive implementation.

Hardware Design

The heart of the HP 48G/GX is a fourth-generation CPU chip. This custom ASIC is built around the original HP 71 processor, and its development was key to the creation of the HP 48G/GX. This chip has four advantages over the third-generation chip used in the HP 48S/SX. First, it is produced

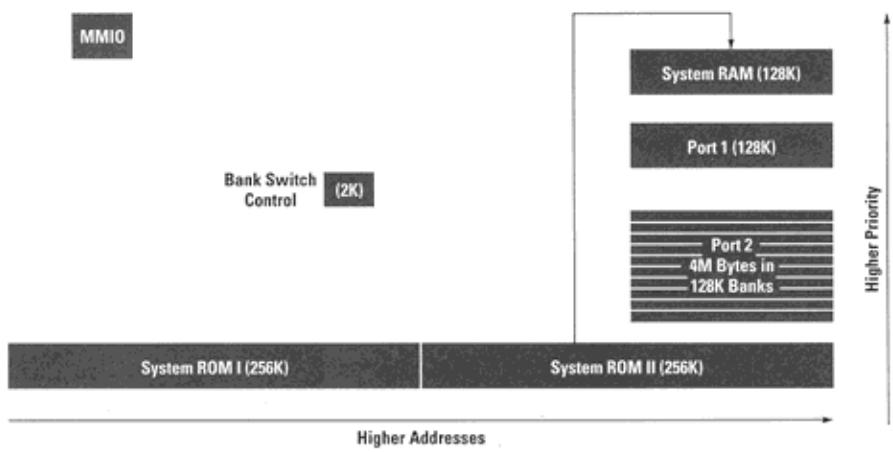

Fig. 10. HP 48GX configuration for copying an object to TEMPOB through a mailbox.

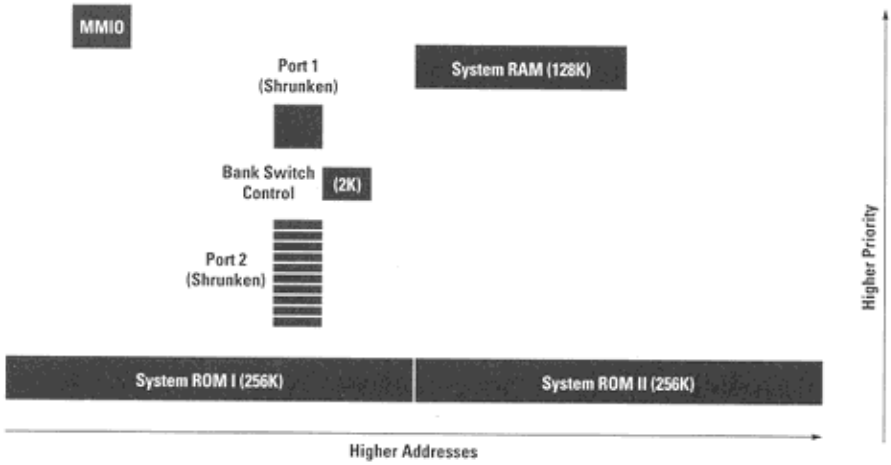

Fig. 11. HP 48GX all-ports-empty configuration.

using a different CMOS process, allowing better stability with onboard voltage regulation circuitry. Second, these improved voltage characteristics and several low-level optimizations allow the new CPU to operate at twice the speed of its predecessor. This speed increase gives it a 4-MHz bus rate. Third, the new CPU is packaged in a 160-pin quad flatpack, improving the manufacturability of the HP 48G/GX. Fourth, with all these improvements, the final cost is lower, increasing the budget for other hardware improvements to the calculator.

The faster processing speed of the HP 48G/GX CPU gave the software team incentive to improve the user interface, implementing graphical routines that would not have been acceptable at the slower processing rate. This added functionality required an increase in data storage space, so we boosted the size of ROM and RAM. We also decided to add the facilities to bank-switch a data card plugged into card port two.

The HP 48G/GX circuitry, with its additional components, had to fit in the same physical space as in the HP 48SX. The product plan and schedule did not allow changes to production tooling or plastic parts except for those that were absolutely necessary. At times we felt like poets trying to write crossword puzzles. The HP 48SX circuit board design was optimized such that it did not leave us much free space. These space constraints affected many of the HP 48G/GX hardware design choices.

The RAM increased from 32K bytes in the HP 48SX to 128K bytes in the HP 48GX, while the HP 48G retained the original 32K-byte chip. This difference between the G and the GX offers two advantages. First, it provides more differentiation between the functions and cost of the G and the GX, increasing the product family's market appeal. Second, the difference in RAM size provides a way for the calculator to know whether it is a G or a GX. If the calculator scans the RAM and finds only 32K bytes, then there will never be a plug-in data card installed. With this information the covered memory options become much simpler. The RAM memory size becomes an internal product type identifier, and several software routines are optimized for faster performance on the HP 48G.

ROM Changes

The HP 48G and GX share a common ROM code set. They also share a common circuit board. While this simplifies documentation, manufacturing, and stock control, it also complicates some areas. The HP 48GX RAM chip is wider and longer than the chip used in the HP 48G: the 32K RAM is in a 28-pin small-outline package (SOP), and the 128K device is a 32-pin SOP. Both conform to the JEDEC pinout standard. A 128K device was chosen that has an extra chip select line at pin 30. This chip select is tied high, allowing pins 1 through 28 of the smaller RAM to overlay pins 3 through 30 of the larger device. The extra chip select of the HP 48GX RAM

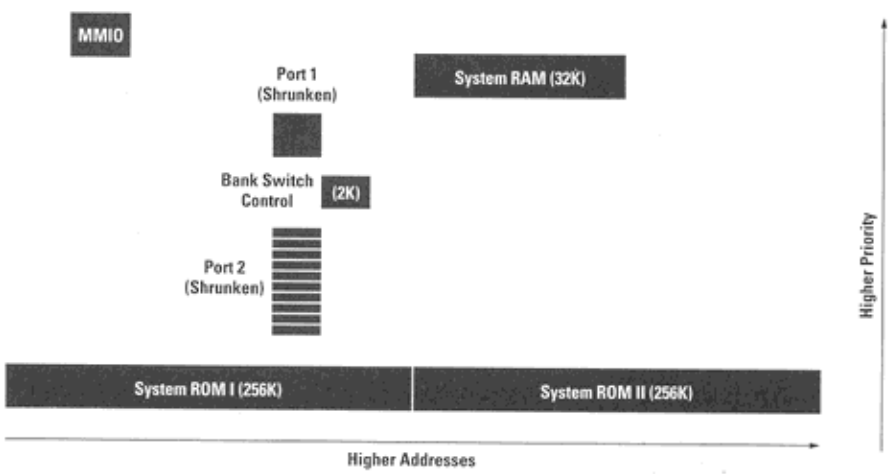

Fig. 12. HP 48G standard configuration.

matches the V_{dd} line of the HP 48G RAM chip, and all of the other lines are pinout-compatible.

The difference in physical package width also posed a problem. The foil patterns on the circuit board had to be modified to accept RAM chips with different lead spacings across the package. The immediate response was simply to stretch the oval-shaped patterns. However, this resulted in the foil extending well under the body of the 128K chip, a situation that could have led to solder bridging where the solder paste contacted the part body. This is avoided by using two different solder stencils on the manufacturing line. A paste of solder is laid on the blank circuit board before parts are loaded onto the board. A metal stencil defines the pattern of the solder paste, just as a silk screen controls the pattern of ink on a shirt. By using a different stencil pattern for the G and GX circuit boards, we control the original location of the solder paste and keep it out from under the 128K-byte RAM body. Once all the components are loaded onto the circuit board and the solder is heated to a molten state, a danger might again exist for the solder to flow under the part body. Fortunately, the nature of the mechanical contact between the RAM lead and the circuit board foil tends to cause the solder to pool or wick to the lead rather than spreading across the elongated foil pad.

The packaging of the ROM chip was also changed between the HP 48SX and the HP 48GX. The SX used a square 52-pin quad flatpack for its 256K bytes of program data. The code size of the HP 48GX is doubled to 512K bytes. Its package is a 32-pin SOP like the 128K-byte RAM chip. Their common package configuration allowed us to conserve space in the placement of the two chips and in the routing of signal wires between them.

Use of a standard SOP ROM chip also allowed us to use one-time programmable (OTP) ROMs in prototype calculators. An OTP uses the same semiconductor core as a UV-erasable EPROM. To get the semiconductor chip into an SOP, however, the manufacturer omits the familiar glass window in the chip, covering the device in opaque plastic. The resulting ROM is no longer erasable.

Typically, a product schedule requires months between code release and the start of production so that ROMs can be built with the software code built-in. The use of OTPs on this project cut the required time from months to days. For one prototype run, the time between code availability and product build was only a few hours.

The HP 48G/GX CPU multiplexes the highest address bit, A18, with an additional chip enable line, CE3. The original idea was to allow future expansion of the HP 48 family, either to use a larger ROM chip or to include an additional memory mapped device. By the time the HP 48G/GX design was complete, we had decided to do both. We doubled ROM to 2^{19} bytes, and we added bank switching to card port 2. Two small HCMOS chips were added to the board to demultiplex these signals. The two chips are a quad NAND chip and a hex D flip-flop, similar to the standard TTL devices. The multiplexing is accomplished by simply toggling a control bit inside the HP 48G/GX CPU. To demultiplex the A18 and CE3 signals, we developed a protocol for mirroring the state of the internal bit to one of the external D flip-flops. The NAND gates handle signal demultiplexing, and the remaining five flip-flops form a register for the card port 2 bank address.

Other Hardware Changes

The card ports of the HP 48SX were designed for Epson memory cards. Several unused lines were adapted to provide external video signals to drive an enlarged display for classroom use. On the HP 48GX, the video lines are retained only on card port 1. On card port 2, the video lines are replaced by five additional address lines. The system software allows the card in port 2 to be subdivided into 128K-byte sections, with each section treated as a virtual plug-in card. Five bank select address lines permit up to 32 virtual plug-ins in card port 2, yielding a maximum card size of 4M bytes in the plug-in port. With the ROM, RAM, and plug-in options, an HP 48GX can access 4,980,736 bytes of onboard data.

Since the inception of the HP 48 family of calculators, liquid crystal display technology has progressed significantly. The display in the HP 48G/GX provides improved visibility by improvements in pixel contrast. The display is thinner than before. This change in glass thickness reduces the parallax between the pixel within the display and its shadow on the rear face of the display. In the HP 48SX, the pixel contrast was lower and the shadow was not dark enough to cause problems, but in testing the new HP 48G/GX display under various light conditions, we found that shadow effects made the display hard to read. With the thinner glass now used, the pixel and its shadow appear as one image, and the shadow now enhances the appearance of the pixel.

Changes to plastic parts were not permitted, except where necessary. The back case of the HP 48G/GX required changes. The changes were all accomplished by making mold inserts. Where text on the mold needed changes or additions, the affected area of the mold was milled away. A piece of steel was placed into the hole to make a perfect fit, and the face of this new piece was etched or inscribed with the new textures and features. The new back case helps identify the differences between card ports 1 and 2, updates the copyright information, adds a mark indicating that the HP 48G/GX complies with Mexico's importation laws, and adds an area for a customized nameplate. The customized nameplate is a piece of metal with adhesive on one side. The customer's name can be engraved on the plate and attached to an inset area of the back case. This is the same nameplate used on HP's palmtop computer family.

The result of these changes is a computer platform that is more powerful than its predecessor, is well-suited to the enhanced user interface developed by the software team, is more versatile for both the user and the design engineer, and is less expensive to produce. It started as a processor upgrade and became a major product improvement.

User Interface

With ease of learning and ease of use the primary goals for the new HP 48G/GX calculator, the user interface and many built-in applications have been largely redesigned.

Input forms provide the common starting point for the new and rewritten applications in the HP 48G/GX. Looking

Fig. 13. A typical input form.

much like dialog boxes in an Apple Macintosh or Microsoft® Windows PC, input forms provide a fill-in-the-blanks guide to the input needed for a task, plus application-specific menu keys for acting on that input.

For selecting an application in a particular topic and for picking an input from among several choices we developed *choose boxes*, a type of pop-up menu that suggests alternatives and narrows the input focus.

We designed *message boxes* to make feedback to the user more manageable within our increasingly crowded display space. Message boxes appear on top of whatever the user is working on and provide more flexibility for formatted messages and icons than the two-line, fixed-location error messages they replace. They also preserve the context that can otherwise be lost when something surprising happens within an application.

Input Forms

An input form provides both a means to enter data pertinent to an application and operations that permit the user to direct actions.

Visually, an input form consists of (see Fig. 13):

- A title suggesting the form's purpose
- One or more fields, typically with explanatory labels, which are used to gather and display user input
- A help line that details the input expected in the selected field
- Menu keys that provide more options for working within or exiting the input form.

Each input form field can be one of four types. Most input forms, such as the Set Alarm and I/O Transfer input forms, contain several or all types of fields. *Text fields* are used to enter arbitrary HP 48G/GX objects like real numbers and matrices; the object types allowed are specific to each text field. In Fig. 14, a text field is used to enter an alarm message in the Set Alarm input form.

When a single choice among several is required, *list fields* are used to eliminate invalid input and to help focus user actions. To select an entry in a list field, a choose box is displayed. In Fig. 15, a list field is used to specify the transfer format in the I/O Transfer input form.

Sometimes only a simple yes-no, do-or-don't type of choice is needed. For this we use *check fields*. Fig. 16 shows how the overwrite (OVWR) field is used to specify whether or not an existing variable should be overwritten.

Finally, when arbitrary input is possible but logical choices are also available, *combined text/list fields* are employed. In

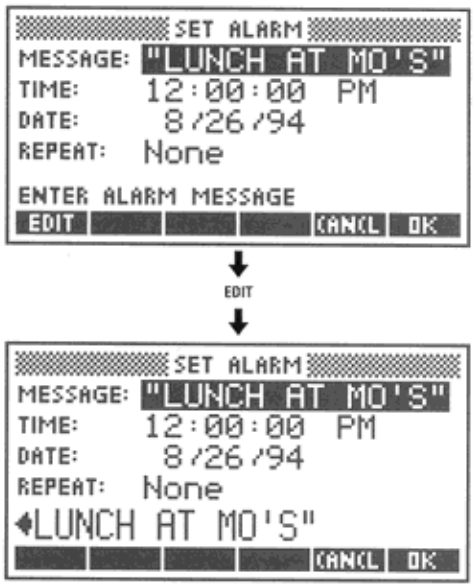

Fig. 14. Using a text field in an input form.

the Transfer input form, the Name field is a combined field that permits new names to be entered or the names of existing HP 48G/GX variables or PC files to be selected (see Fig. 17).

As the figures illustrate, each of the three base field types has associated with it a dedicated menu key that triggers the unique feature of that field type. This feature is an important part of how we maintained a calculator key-per-function-style interface within the constraints of a small display and with no pointing device. In other graphical user interfaces, visual elements such as list arrows are activated by mouse clicks to elicit different behaviors from fields. In the HP 48G/GX, the user's finger acts as the pointing device, triggering the desired behavior by pressing the appropriate action button for each field. Consistent location of the three types

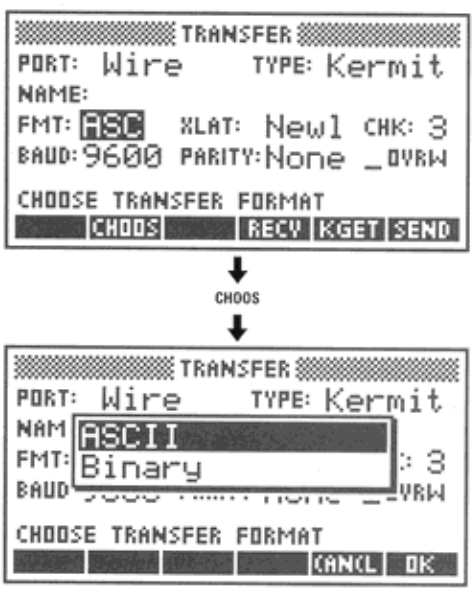

Fig. 15. Using a list field in an input form.

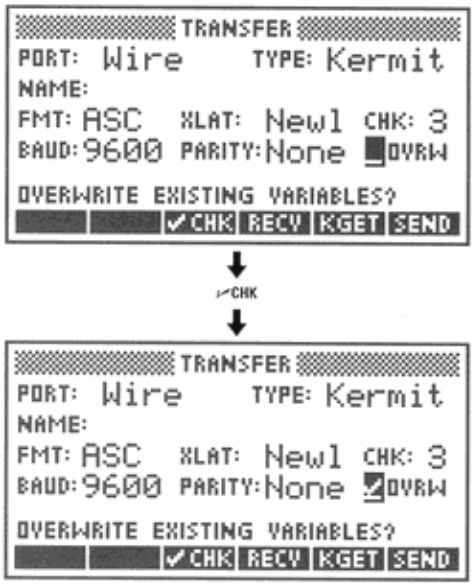

Fig. 16. Using a check field in an input form.

of action buttons helps the user navigate an input form confidently.

Some input form menu keys perform application-specific operations—for example, DRAW in Plotting. In the second row of the input form menu are more advanced input form operations for resetting a field or the entire form, displaying the object types allowed in a field, and temporarily accessing the user stack to calculate or modify a field value.

Choose Boxes

Choose boxes are used to make a choice in an input form list field. They are also used in most subject areas to choose a specific application from among several. Fig. 18 shows the choose box that is displayed when the STAT key is pressed to perform statistical calculations.

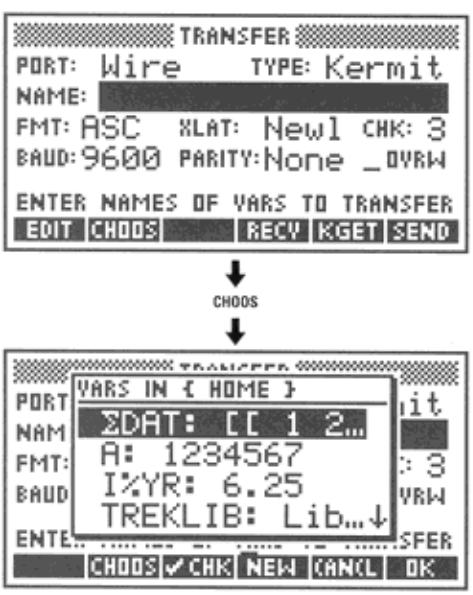

Fig. 17. Using a combined text/list field in an input form.

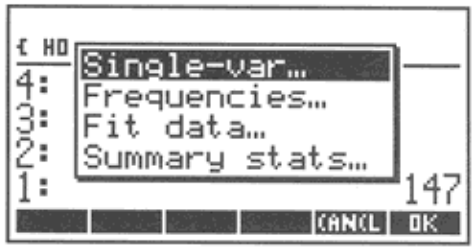

Fig. 18. A typical choose box.

When circumstances require, choose boxes can include any or all of several advanced features. The Memory Browser application, for example, is actually a maximum-size choose box embellished with a title, multichoice capability, and a custom menu (see Fig. 19).

Message Boxes

Message boxes are used primarily for reporting errors that require attention before proceeding. For example, if the user attempts to enter a vector in the EXPR field of the Integrate input form, a message box appears to inform the user of the problem (see Fig. 20).

Some applications also use message boxes to give additional information. For example, in the Solve Equation input form, the user can press INFO any time after a solution has been found to review the solution and determine how it was calculated (see Fig. 21).

Input Form Implementation

For the HP 48S/SX, we developed an RPL tool called the parameterized outer loop[3] to speed development of new interfaces such as the MatrixWriter by automating routine key and error handling and display management. The input forms in the new HP 48G/GX embrace this concept—in fact, the input forms engine is a parameterized outer loop application—and take it one step farther to automate routine matters of application input entry and selection of options. The input forms engine brings a uniform interface to all new HP 48G/GX applications.

While narrowly focusing the task of application development by managing command input tasks, the input forms engine also leaves much room for the customization that helps optimize the HP 48G/GX for ease of use. Since an important measure of our progress towards our goals for the calculator was to be feedback from typical users throughout the development cycle, we designed the input forms engine from the ground up to be highly customizable. This was accomplished in a programmer-friendly manner by including

Fig. 19. The Memory Browser: a complex choose box.

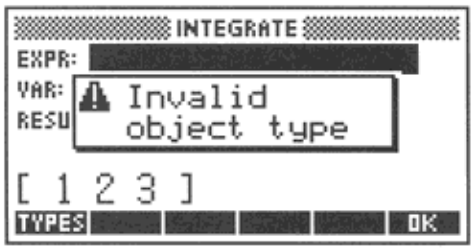

Fig. 20. A typical message box.

over fifty hooks into the input forms engine's responses to external and internal *events*. *External events* are triggered by users and include low-level events such as key presses and high-level events such as completion of a field entry. *Internal events* are usually activated by external events, such as formatting a completed field entry for proper display. A single external event can trigger a half dozen or more internal events, all of which are customizable.

Input form applications can customize any or all form-level events such as title display or field events such as displaying a help line. Each field has a *field procedure* associated with it, and the entire form has a *form procedure* associated with it. Whenever an event occurs, the appropriate field or form procedure is called with an identifying event number and perhaps additional information. If the procedure does not customize the event, it returns FALSE to the input forms engine. If it does customize the event, the procedure performs the custom behavior and returns TRUE. In this manner, every event first queries the proper form or field procedure to determine if custom behavior is needed, then handles the event normally only if it isn't customized. If a form or field has no custom behavior, it specifies a default procedure that quickly responds FALSE to all event queries.

The reason for a form procedure and multiple field procedures is to spread the burden of customization throughout the form. Since each field procedure only checks for the events that pertain to it, and since the form procedure only checks for form-level events, no single event processing is slowed by a highly customized form that would otherwise have to compare the event number against a lengthy list of event and field combinations.

For the HP 48G/GX project we needed another layer of regularity not enforced by the input forms engine. Because we sought and reacted to usability feedback almost until the code was released to production, the user interface details for each subject area were subject to constant change. It was imperative, therefore, that we maintain a strict and formal

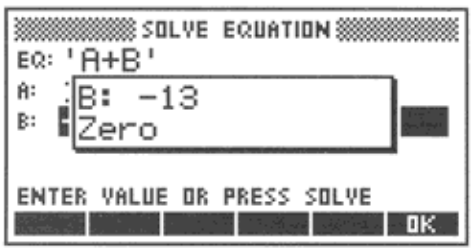

Fig. 21. The Solve Equation INFO message box.

division between unchanging and well-understood tasks—such as getting and saving problem domain information and calculating results—and the user interface details that were changing regularly. We developed a set of conventions that were embodied in what we called *translation files*. We used naming rules and constrained responsibilities to greatly mitigate the effects of user interface changes on the underlying problem-solving functionality. For example, one RPL word in the plotting translation file has the simple task of reading the current horizontal plot range from calculator memory. Since the word has no presumptions about how and when it will be called, references to it could be (and often were) changed around as the fields populating the plotting input form were worked out.

Choose Box Implementation

The choose box engine is very much like the input forms engine. For customization, the programmer can supply a *choose procedure* that responds to 26 messages.

A feature of choose boxes that simplifies their use is the option—heavily used by the built-in applications—of items that encapsulate both display and evaluation data. For example, when an angle measure—degrees, radians, or grads—is to be chosen in certain input forms, the choose box engine displays plain descriptions but returns an RPL program that sets the selected angle measure. This circumvents the need for branching according to the returned object and simplifies the extension of choices.

Results: Benefits and Costs

Initial feedback from the educational advisory committee and user reviews suggests that the use of input forms and other graphical user interface elements has greatly improved the ease of use of the HP 48G/GX over the HP 48S/SX. However, the path we took to this accomplishment was more challenging than we planned.

Event customization, originally conceived as a means to extend the functionality of input forms in unforeseen ways, turned out to be a key component of our ability to prototype new user interface ideas rapidly. As their name may imply, the original intent of input forms was very modest compared to the role they now play. We designed input forms to be the standard means by which applications gather data for a task. One or more input forms would be displayed as necessary within the context of another, undefined, application context. This original concept is applied successfully throughout the calculator. For example, in the Memory Browser, when NEW is pressed to create a new variable, an input form is used to get the information required (see Fig. 22). In this context, the user can do only three things in the input form: enter data, cancel the form, or accept (OK) the form. This simple but effective behavior was the model used for the original input form design.

As the project developed, however, it became apparent that an input form could serve not only as a information gatherer but also as an action director. Input forms thus graduated from simple dialog boxes to full-fledged application environments.

Fig. 22. Memory Browser NEW input form.

Interestingly, no major changes to the input forms engine were necessary or even desirable to support their new role. Instead, the essence of input form functionality remained always data management, and the events customization was applied selectively where needed to enhance application forms.

In a similar manner, the event-driven choose box engine was eventually pressed into service as a powerful base for list-style applications like the Memory Browser.

The combination of lean, focused, standard feature sets for input forms and choose boxes and high customizability proved invaluable during the calculator design refinement. Throughout the middle portion of the project, when the basics had been settled but many user interface details were still unclear, we were able to prototype new ideas quickly and realistically by customizing event responses.

Translation files were another development effort that helped us keep the design and implementation moving forward. However, we learned over time that their overhead caused some duplication of code and inefficiency to creep into the interface between the input forms and the calculator mainframe. We addressed this issue where possible by making simple and safe code substitutions while leaving the interface concepts intact to enable high-confidence code defect fixes late in the project. In effect, we made a choice between maintainability and high performance that still remains a controversial topic among the HP 48G/GX developers.

3D Plotting

The functionality described in this section is a suite of 3D graphing and viewing utilities for the HP 48G/GX. We had several requirements to consider in creating these routines. Our aims were that they be psychologically effective and require only a small amount of code.

In exploring visualization techniques on a variety of machines we found that increasing "realism" (ray-traced, Phong-shaded, hidden-line, etc.) in the graphical presentation of functions of two variables did not necessarily correlate with increasing ease of comprehension. The HP 48G/GX routines represent the results of some of these experiments (including time-to-completion as an important factor).

All of the 3D plotting routines are intended as seamless extensions of the other built-in plotting utilities. In particular, they share the same standard user interface and are selected as alternative plot types. The 3D plotting routines are SLOPEFIELD, WIREFRAME, YSLICE, PCONTOUR, GRIDMAP, and PARSURFACE.

Like the other plotting routines, all the 3D plotting routines assume that the function of interest is stored in EQ.[3] Further, they assume, by default, that the function is represented as an expression in the variables X and Y—for example, u,v → sin(u+v) is represented as SIN(X+Y) in EQ. The use of other variable names is provided for by input form options or by the INDEP and DEPEND keywords.

While this section is titled "3D Plotting," a better name would be "visualization techniques for functions of two variables." This would cover the perspective view of the graph of a scalar function of two variables (WIREFRAME), the slicing view of a scalar function of two variables (YSLICE), the contour-map view of a scalar function of two variables (PCONTOUR), the slope interpretation of a scalar function of two variables (SLOPEFIELD), the mapping grid visualization of a two-vector-valued function of two variables (GRIDMAP), and the image graph of a three-vector-valued function of two variables (PARSURFACE).

Given this unity of purpose, there is considerable overlap in the global parameters (options) used in these routines. These plotting parameters are stored in the variable VPAR, analogous to PPAR.[3] The main data structure stored in VPAR describes the *view volume*, a region in abstract three-dimensional space in which most of the visualizations occur (see Fig. 23).

VPAR quantities controlling the view volume are:

- X_{left} and X_{right}, controlling the width of the view volume
- Y_{far} and Y_{near}, controlling the depth of the view volume
- Z_{low} and Z_{high}, controlling the height of the view volume
- X_e, Y_e, and Z_e, the coordinates of the *eye point*.

In addition to these, VPAR contains other quantities used by some of the routines. These are:

- XX_{left} and XX_{right}, an alternative X input range, used for GRIDMAP and PARSURFACE
- YY_{far} and YY_{near}, an alternative Y input range, used for GRIDMAP and PARSURFACE (note that this differs from the current Suite3D interpretation)
- N_x and N_y, the number of X and Y increments desired, used in all of the routines instead of or in combination with RES.

SLOPEFIELD

The SLOPEFIELD plot type draws a lattice of line segments whose slopes represent the function value at their center point. Using SLOPEFIELD to plot f(x,y) allows your eye to pick out integral curves of the differential equation dy/dx = f(x,y). It is quite useful in understanding the arbitrary constant in antiderivatives.

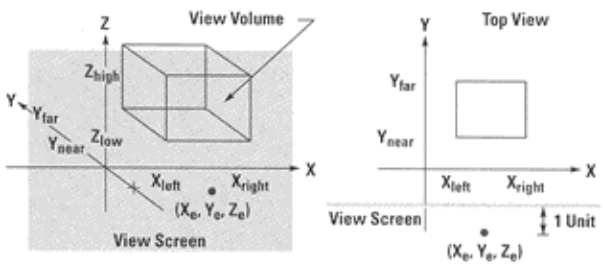

Fig. 23. VPAR parameters in relation to the view volume.

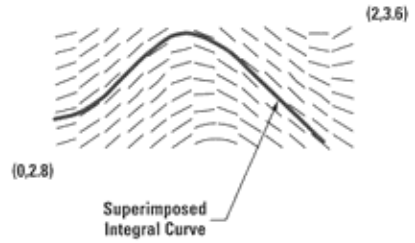

Fig. 24. SLOPEFIELD plot of dx/dt = sin(xt).

The number of lattice points per row is determined by N_x and the number of lattice points per column is determined by N_y. The input region sampled is given by $X_{left} < X < X_{right}$ and $Y_{near} < Y < Y_{far}$.

The input form in this case allows the user to:

- Choose or enter the defining expression for the function to be plotted
- Choose the names of the two variables (identical to INDEP and DEPEND)
- Choose X_{left} and X_{right} (default to their current value, or XRNG if no current value)
- Choose Y_{near} and Y_{far} (default to their current value, or YRNG if no current value)
- Choose N_x and N_y (default to their current value or 13 and 8 if no current value)
- Verify and/or choose RADIANS, DEGREES, or GRADS mode.

In trace mode for SLOPEFIELD, the arrow keys jump the cursor from sample point to sample point indicating both the coordinates of the sample point and the value of the slope at that point.

Example Problem: Determine graphically whether all solutions of the differential equation dx/dt = sin(xt) with initial conditions $3.0 < x(0) < 3.1$ satisfy $2.8 < x(t) < 3.6$ for all t in [0,2].

Solution: Choose SLOPEFIELD plot type and enter SIN(X*T) as the current equation. Choose T as the independent variable and X as the dependent variable. Choose 0 as X_{left}, 2 as X_{right}, 2.8 as Y_{near}, and 3.6 as Y_{far}. Verify RADIANS mode, and draw the result. As seen in Fig. 24, almost all of the integral curves in this region leave the window either through the top or the bottom. Therefore, not all the integral curves satisfy $2.8 < x(t) < 3.6$ for t in [0,2].

WIREFRAME

The WIREFRAME plot type draws an oblique-view, perspective, 3D plot of a wireframe model of the surface determined by z = f(x,y). The function determined by the current equation is sampled in a grid with N_x samples in each row and N_y samples in each column. Each sample is perspective-projected onto the view screen along the line connecting the sample and the eye point (see Fig. 25).

Neighboring samples are connected by straight lines. The sampled region is determined by the base of the view volume (X_{left}, X_{right}, Y_{near}, Y_{far}). The region of the view screen represented in the PICT GROB (graphics object[3]) and hence on the display is determined by the projection of the view volume on the view screen (see Fig. 26).

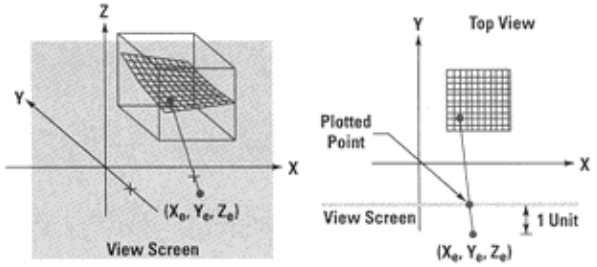

Fig. 25. Perspective projection of a point in the view volume onto the view screen.

The input form in this case allows the user to:

- Choose or enter the defining expression for the function to be plotted
- Choose the names of the two variables (identical to INDEP and DEPEND)
- Choose X_{left} and X_{right} (default to their current value, or XRNG if no current value)
- Choose Y_{near} and Y_{far} (default to their current value, or YRNG if no current value)
- Choose Z_{low} and Z_{high} (default to their current value, or default YRNG if no current value)
- Choose X_e, Y_e, and Z_e (default to their current value, or 0, −1, 0 if no current value)
- Choose N_x and N_y (default to their current value or 13 and 8 if no current value)
- Verify and/or choose RADIANS, DEGREES, or GRADS mode.

In trace mode for WIREFRAME, the arrow keys jump the cursor from sample point to sample point and the display indicates all three coordinates of the sample point.

Example Problem: Determine graphically whether the surface defined by $z = x^4 - 4x^2y^2 + y^4$ is, at the origin, concave up, concave down, or neither.

Solution: Choose WIREFRAME plot type and enter X^4−4*X^2*Y^2+Y^4 as the current equation. Choose X and Y as the independent and dependent variables. Choose −1 for X_{left}, 1 for X_{right}, −1 for Y_{near}, 1 for Y_{far}, −1 for Z_{low}, and 1 for Z_{high} so that the view volume surrounds the origin. Choose 4 for X_e, −10 for Y_e, and 3 for Z_e to give a distant, oblique view of the graph. As seen in Fig. 27, the graph displays a "monkey saddle" which is neither convex nor concave at the origin.

New Interactive Features

The picture environment, which is invoked automatically when graphs are drawn or by pressing the PICTURE key, allows the user to interact with a graph. The user can move

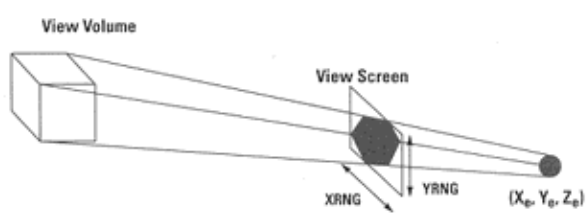

Fig. 26. Relationship of view volume and eye point to XRNG and YRNG.

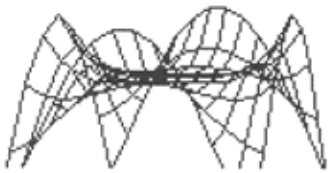

Fig. 27. WIREFRAME plot of the surface determined by $z = x^4 - 4x^2y^2 + y^4$ with view volume $[-1,1]\times[-1,1]\times[-1,1]$ and eye point $(4,-10,3)$.

the cross hairs around using the arrow keys, trace along the graph, add picture elements such as dots, lines, and circles, or do interactive calculus operations such as finding the derivative at the cross hairs location.

Trace, Faster Cross Hairs

The HP 48S/SX cross-hair-moving code was rewritten for the HP 48G/GX. The cross hairs needed to be faster, to fade less as they moved, and to accommodate added functionality such as tracing and shading. The cross hair code originally came from the HP 28 and has been maintained and modified over the years. In the HP 28, most of the cross hair code was written in high-speed assembly language and actually contained a routine called SLOW which was buried deep within RPL subroutine calls. SLOW was needed to slow the cross hairs down to an acceptable speed. The use of this word was discovered during the process of porting the code from the HP 28 to run on the bigger HP 48S/SX display. There were many occasions during the HP 48G/GX project when we were trying to speed up various operations and wished we could just find the word SLOW and take it out!

The fading of the cross hairs as they moved was improved by changing the code so that the time between turning the cross hairs off at one pixel and turning them on at the next pixel is minimized. To do this, all the calculations required for moving are now done before turning the cross hairs off. Unfortunately, the new display on the HP 48G/GX trades off response time for contrast, so although it is brighter and has higher contrast than the HP 48S/SX display, it takes longer for pixels to turn dark. Thus, much of the work to reduce fade in moving cross hairs was canceled out by the new screen characteristics. The user can darken the display by holding down the **ON** key and pressing the **+** key a few times, and this will make the moving cross hairs easier to see.

Tracing along a graph with the cross hairs presents a challenge because the user's function must be evaluated at every point, so in effect the system RPL programmer must turn control over to the user at each point of the graph. This required careful attention to error handling and to managing the data stack, which is a shared resource. The procedures for tracing vary with the different plot types. The procedures are kept in the property list associated with each plot type, and then the appropriate procedure is passed in and evaluated when trace mode is turned on. It required quite a bit of rewriting to implement this object-oriented, extensible approach because much of the existing cross hair code had previously undergone rewriting and optimizing for speed and code size.

Animation

The ANIMATE command is a program that was easy to write and that the user could have written in user-RPL programming language, but we added it for the sake of convenience. Also, it is used as part of Y-slice 3D plotting. It sets up a loop that repeatedly puts graphics objects into the PICT display area.

A quick way to get started with animation is to press PICTURE to go to the interactive graphics environment, where you will be able to create some pictures to animate. Press EDIT, then DOT+ to turn on the etch-a-sketch-style drawing mode in which pixels are turned on wherever you move the cross hairs. Using the up, down, left, and right arrow keys, sketch something, then press STO to send a copy of your picture to the stack. Continue sketching, press STO again, and repeat this procedure, continuing to add to your sketch until you have a handful of pictures on the stack, say six of them. Press CANCEL to leave the PICTURE environment, and you will see the the picture objects sitting on the stack. They are called GROBs, which is short for graphics objects.[3] To use the ANIMATE command, all you have to do is enter the number of GROBs (for example, press **6** then **ENTER** if you created six pictures), then press the ANIMATE key, which you will find in the GROB submenu of the PRG menu. Your series of sketches will come to life as the ANIMATE command flips through them.

Mathematics

Several new mathematical features were added to meet the needs of the educational market and to match or exceed corresponding features recently introduced by our competition. Design trade-offs made for and inherited from earlier, less capable platforms were reconsidered, and relevant software developed for earlier machines but not used in the HP 48S/SX was used wherever appropriate.

Design and Implementation Issues

The HP 48G/GX is targeted at the college-level mathematics, science, and engineering educational market. We hoped, also, to achieve more success in high school advanced placement courses. In these environments calculators are used as pedagogical tools, illustrating mathematical and modeling concepts introduced in the courses.

As a pedagogical tool, the calculator's accuracy and reliability are paramount design goals. Speed of execution is important but secondary to the validity of the computed results. To achieve high accuracy and reliability the computational methods needed to be more numerically sophisticated than typical textbook methods. This greater complexity is hidden from the casual user wherever possible, but made available to the sophisticated user so the methods can be tuned to their needs.

One means of achieving maximum accuracy and reliability is to read the current literature and consult with expert specialists to obtain the best methods, then implement those methods from scratch. We have employed this approach in the

User Versions of Interface Tools

Although the primary focus of the new user interface for the HP 48G/GX was to enhance our built-in applications, it became apparent as the project progressed that calculator owners who program would want access to the same capabilities to enhance their efforts. For the choose box, message box, and especially the input form tools, the biggest challenge involved scaling back the numerous features to produce simple user commands that still offer customization potential.

The message box command, MSGBOX, was designed to display pop-up messages with a minimum of fuss. Thus, it takes just one argument—the message string—and produces a word-wrapped normal-sized message box.

The choose box command, CHOOSE, is slightly more complicated. To enable but not require the same object-oriented use of choose boxes as the built-in applications, the CHOOSE command accepts a list of items in two formats. In the simplest format, an item is specified by a single object, which is displayed and returned if chosen. In the alternate format, an item is specified by a two-element list object. The first element is displayed in the choose box, and the second element is returned if the item is chosen.

For simplicity of the user interface, CHOOSE displays a normal-sized choose box without the multiple-choice capability used by some built-in applications.

The MSGBOX and CHOOSE commands largely follow the same interface specification methods as their system-level counterparts. This differs markedly from the input form user command, INFORM. To maintain complete flexibility over all elements of form layout and behavior, the input forms engine takes three arguments for each label and thirteen arguments for each field, specifying such details as exact location and size, display format, and so on. Added to that are global arguments for the form procedure and form title and some other details. All together, an input form with four labeled fields requires 68 arguments. While this amount of information is justified for the varied needs of built-in applications, it is an unnecessary burden for programmers just wanting to get some simple input from the user.

For the INFORM command, therefore, we developed an automatic form layout scheme that serves most needs, with options for further detailing. Basically, the INFORM input form is viewed as a grid that is filled with fields starting in the upper-left corner and proceeding from left to right and top to bottom. The number of columns in the grid is specified as one of INFORM's arguments, and each field's width is determined by the width of its label and by the user-supplied tab width, which places invisible tab stops within each column to help align fields vertically. A field can span multiple columns with a special field-expander specification. Help text and object type restrictions can be included for any field, but aren't required.

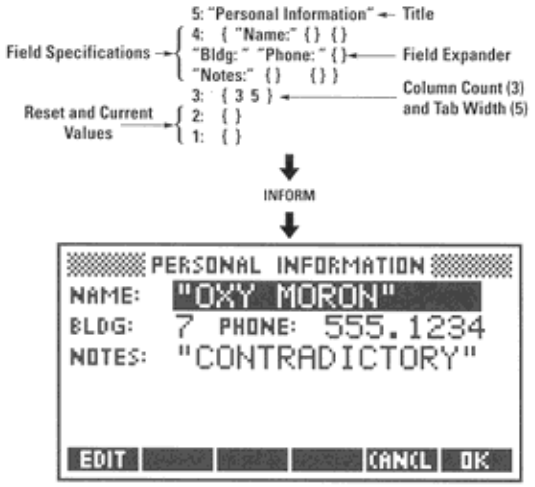

Fig. 1. A custom input form created by INFORM.

Fig. 1 shows an example of a custom input form created by INFORM. Notice that, despite the relative simplicity of the input arguments, an input form with aligned fields of varying widths is presented. This technique for building input forms proved so valuable that it was used to create the Solve Equation input form, which changes according to the number and names of variables in the equation to be solved.

past with success, but it can be time-consuming, expensive, and risky.

Another approach sometimes available is to consult standard computational libraries used by the professional scientific community. Several such public-domain libraries are available that represent the current state of the art. In some development environments these libraries can be used directly. In others, they can at least provide high-quality methods and implementations that when judiciously used facilitate meeting tight development schedules at low cost. We found the LAPACK library[4] of FORTRAN 77 numerical linear algebra subroutines particularly helpful in this regard.

As usual, code was reused whenever possible to achieve timely and reliable implementations. In addition to the source code for the HP 48S/SX and its Equation Library card, we had implementations dating from the HP 71 Math Pac that were revised for the HP 48S/SX but didn't find ROM space in that product.

While reusing code, we took advantage of the HP 48G/GX CPU clock speedup and larger RAM environment over the HP 48S/SX to reconsider some of our previous implementation trade-offs in an effort to achieve greater accuracy. In some cases we decided to employ more computational effort and to store intermediate values in higher precision to achieve more accurate results.

New Mathematical Features

The HP 48G/GX includes many new mathematical features over those provided by the HP 48S/SX. These are array manipulations, additional linear algebra operations, a polynomial root finder and related operations, two differential equations solvers and associated solution plotters, discrete Fourier transforms, and financial loan computations.

The array manipulation commands are primarily pedagogical tools. These include a random array generator and commands to add or delete rows or columns of matrices or elements of vectors, decompose matrices into or create matrices from row or column vectors, extract diagonal elements from a matrix or create a matrix from its diagonal elements, perform elementary row and column operations, and compute the row-reduced echelon form of a matrix.

We significantly improved and expanded the linear algebra functionality of the HP 48G/GX over the HP 48S/SX. The determinant, linear system solver, and matrix inverter were

revised to be more accurate through additional computation and by storing all intermediate values in extended precision. We added a command to compute a condition number of a square matrix, which can be used to measure the sensitivity of numerical linear algebra computations to rounding errors, a command to compute a solution to an underdetermined or overdetermined linear system by the method of least squares, commands to compute eigenvalues and eigenvectors of a square matrix, commands to compute the singular value decomposition of a general matrix, and commands to compute related matrix factorizations and functions. These linear algebra commands accept both real and complex arguments and perform all intermediate computation and storage in extended precision.

The HP 48G/GX has commands to compute all roots of a real or complex polynomial, to construct a monic polynomial from its roots, and to evaluate a polynomial at a point. The polynomial root finder is a modification of the HP 71 Math Pac's PROOT command, extended to handle complex coefficients. It uses the Laguerre method with deflation for fast convergence and constrained step size and an alternate initial search strategy for reliability.

The HP 48G/GX has commands to compute the discrete Fourier transform or the inverse discrete Fourier transform of real or complex data. These commands were leveraged from the HP 71 Math Pac's FFT and IFFT commands, requiring the data lengths to be a nonzero power of 2, and were modified slightly to match the customary definitions of these transformations.

Finally, we included time-value-of-money commands. These commands have appeared in our financial calculators and were available on the HP 48SX Equation Library card. Since engineering feasibility studies must include at least rudimentary time-value-of-money computations it seemed useful to include these commands in the HP 48G/GX.

Differential Equation Plotting

The HP 48G/GX contains two differential equation solvers and solution plotters. These solvers and solution plotters can be accessed via their input forms or invoked programmatically via commands. We provide a programmatic interface to the differential equation solvers and their subtasks so the user can use them with the calculator's general solver feature to determine when a computed differential equation solution satisfies some condition, or to implement custom differential equation solvers from their subtasks.

In implementing the differential equation solution plots, one challenge was to identify and implement good solution methods. Another challenge was to merge this new plot type with the new 3D plot types described earlier and with the existing HP 48SX plot environment in a backward-compatible manner.

The HP 48G/GX specifically solves the initial value problem, consisting of finding the solution y(t) to the first-order equation $y'(t) = f(t,y)$ with the initial condition $y(t_0) = y_0$. Here $y'(t)$ denotes the first derivative of a scalar-valued or vector-valued solution y with respect to a scalar-valued parameter t. Higher-order differential equations can be expressed as a first-order system, so this problem is more general than it might at first appear.

Many solution methods have been developed over the years to solve the initial value problem. We decided to implement two methods, a Runge-Kutta-Fehlberg method for simplicity and speed of execution and a Rosenbrock method for reliability. The first method is easier to use, requiring less information from the user, but can fail on stiff problems.* The Rosenbrock method requires more information from the user, but can solve a wider selection of initial value problems. Both initial value problem solution methods require the user to provide the function f(t,y), the initial conditions, the final value of t, and an absolute error tolerance. The Rosenbrock method also requires the derivative of f(t,y) with respect to y (FYY) and the derivative of f(t,y) with respect to t (FYT).

All plot types use the contents of the variable EQ, typically to specify the function to be plotted. If the user selects the stiff (Rosenbrock) method the extra functions are passed to the solver by binding EQ to a list of functions f(t,y), FYY, and FYT. Otherwise, EQ is bound to the function f(t,y) needed by the Runge-Kutta-Fehlberg method.

Both methods solve the initial value problem by computing a series of solution steps from the initial conditions towards the final value, by default taking steps as large as possible subject to maintaining the specified error tolerance. The solution plotter plots the computed values and by default draws straight lines between the plotted points. However, although the computed steps may be accurate, the line segments drawn between the step endpoints may poorly represent the solution between those points. The plot parameter RES is used by many plot types to control the plot resolution. If RES is zero the initial value problem solution plotter imposes no additional limits on the step sizes. If RES is nonzero the plotter limits each step to have maximum size RES.

For the scalar-valued initial value problem it is typical to plot the computed solution y(t) on the vertical axis and the parameter t on the horizontal axis. However, in the vector-valued case the choice of what is to be plotted is not as clear. The user may wish a particular component of the computed solution plotted versus t or may wish two components plotted versus each other. The HP 48G/GX allows the user to specify the computed scalar solution, any component of the computed vector solution, or the parameter t to be plotted on either axis. This flexibility was introduced into the plot environment by expanding the AXES plot parameter. Previously, this parameter specified the coordinates of the axes origin. This parameter was expanded so that an optional form is a list specifying the origin and the horizontal and vertical plot components.

By judiciously expanding the meaning of the various plot parameters we were able to accommodate the differential equation solution plot type while maintaining backward compatibility with previous plot types.

* Stiff problems typically have solution components with large differences in time scale. More information is needed by a solver to compute a solution efficiently.

Acknowledgments

We would like to acknowledge the rest of the software development team: Jim Donnelly, Gabe Eisenstein, Max Jones, and Bob Worsley. Bill Wickes also contributed software. Clain Anderson and Ron Brooks from the marketing department were involved in the day-to-day design process and kept us informed about user needs. Dennis York oversaw both the R&D and marketing aspects of the project, which helped generate synergy between R&D and marketing. We would like to thank Dan Coffin, our manual writer, and John Loux from the technical support group for going out of their way to participate in the design work and for providing many valuable ideas.

References

1. W.C. Wickes, "An Evolutionary RPN Calculator for Technical Professionals," *Hewlett-Packard Journal*, Vol. 38, no. 8, August 1987, pp. 11-17.

2. W.C. Wickes and C.M. Patton, "The HP 48SX Scientific Expandable Calculator: Innovation and Evolution," *Hewlett-Packard Journal*, Vol. 42, no. 3, June 1991 pp. 6-12.

3. T.W. Beers, et al, "HP 48SX Interfaces and Applications," *Hewlett-Packard Journal*, Vol. 42, no. 3, June 1991 pp. 13-21.

4. E. Anderson, et al, *LAPACK Users' Guide*, SIAM, Philadelphia, 1992.