Chapter 49

The HP 9810/20/30 Series

A New Series of Programmable Calculators[1]

Richard M. Spangler

In recent years, programmable calculators have taken on a large portion of the computation jobs that were previously handled by computers. Calculators have several advantages that are responsible for this trend. Calculators are small, self-contained, and easily transported—they can be brought directly to the user's desk. They are quiet, and fit easily into a laboratory or office environment. No complicated turn-on procedure is required; the user merely turns on the power switch, and the calculator is ready. The most important advantage is a psychological one—calculators are "friendly." They are interactive, they provide immediate feedback and immediate answers, and they are dedicated to their user.

The 9800 Series is a new line of powerful programmable calculators and an extensive set of calculator peripherals. The series is designed to cover a broad range of applications. Important objectives of the new series are to provide the user with a choice of calculators that are flexible and expandable, and to support those calculators with comprehensive applications software and peripherals.

The new 9800 Series is the successor of the 9100A/B [Hewlett-Packard, September 1968], HP's first programmable calculators. These earlier calculators were as powerful as the limits of technology at the time of their conception would allow them to be. But with technological advances come better calculators, hence the 9800 Series.

Three Models

There are currently three calculators in the 9800 Series. Model 10 is a key-per-function calculator with a keyboard and language that are extensions of the HP 9100A/B. The display is a three-register numeric display like the 9100A/B's, but uses seven-segment light-emitting-diode characters rather than a cathode-ray tube.

Model 20 has a statement-oriented algebraic language. The user doesn't have to position his variables in special registers or keep track of temporary results. He can enter arithmetic expressions in the same order as he would read them, including parentheses. Model 20 even allows implied multiplication, something that's not allowed even in most high-level computer languages. Model 20 has a display of 16 alphanumeric characters that can display a whole statement at a time. The alphanumeric display can be used during program execution to display comments and instructions as well as numeric results. This capability enhances the interactiveness of this model.

Model 30 is even more interactive. The keyboard is alphanumeric, like a typewriter, rather than key-per-function. This complements the 32-character alphanumeric display by making it convenient to enter text and messages. The programming language of the Model 30 is BASIC, a well-known and easy-to-learn computer language that is designed for use in interactive environments.

The electronics of the 9800 Series is general in design and is common to all three calculators. The central processing unit is a microprogrammed, 16 bit serial processor that implements a general computer machine language (see article, p. 000). The three separate keyboard languages and the arithmetic routines are implemented by firmware routines stored in MOS read-only memory (ROM), and the user's programs are stored in MOS read-write memory. The input/output structure is a general purpose system which makes it possible to interface with a wide variety of peripherals (see article, p. 817).

Many Peripherals

Some of the more important peripherals that have been interfaced are:

9860A	Card Reader
9861A	Output Typewriter
9862A	X-Y Plotter
9863A	Mechanical Paper Tape Reader
9864A	Digitizer
9865A	Magnetic Tape Cassette
9866A	Thermal Line Printer
9869A	Hopper Fed Card Reader
2570A	Instrumentation Coupler
2748A	Paper Tape Reader
2895A	Paper Tape Punch

Several general purpose interface cards are also available to interface with other HP instruments, the new HP interface systems [Nelson and Ricci, 1972], and many peripherals from other manufacturers.

Flexible and Expandable

Flexibility and expandability of the keyboard and programming languages of 9800 Series calculators are provided through the use of add-on ROM modules. From the optional ROMs available, the

[1] *Hewlett-Packard Journal*, vol. 24, December 1972, pp. 2–4.

Comparing 9800 Series Calculators

	9100A/B	*9810A*	*9820A*	*9830A*
Language	Reverse Polish	Reverse Polish	Algebraic	BASIC
Keyboard	Key per function	Key per function	Key per function	Alphanumeric
ROM size (bytes)	4K	5K to 11K	8K to 14K	14K to 31K
RWM size (bytes) Available to user	128(A); 256(B)	908 to 2924	1384 to 3432	3520 to 7616
I/O structure	Special purpose	General	General	General
User definable keys or functions	None	Optional—single key subroutine	Optional—single key subroutine or function with parameters	Standard—subroutine or function with one parameter
Recording device	Magnetic card	Card with cassette optional	Card with cassette optional	Cassette standard
Display	3 register numeric CRT	3 register numeric LED	16 character alphanumeric LED	32 character alphanumeric LED
Primary Printer	Optional 16 column numeric	Optional 16 column alphanumeric	Standard 16 column alphanumeric	Optional 80 column alphanumeric

user can select the language features that are required by his particular discipline.

In Model 10, three ROM blocks of up to 2048 bytes each may be added to the calculator. The first block is used to define and implement the functions of a set of 15 keys on the keyboard. The second and third blocks are for control of internal and external peripherals.

In Model 20, three blocks may be added, each controlling one of three sets of ten keys on the keyboard.

In Model 30, eight blocks may be added, and since the Model 30 has an alphanumeric keyboard, no special keys are required. The ROMs are accessed through mnemonics which are entered as a sequence of alphabetic characters.

Different Models for Different Users

Each of the three calculators is general purpose, but each has features which make it more appealing to different sets of users. Model 10's advantages are its low cost, and its compatibility with the 9100A/B, which provides the basis for an extensive applications program library. For example, the Surveying and Statistics applications packages that were originally developed for the 9100A/B have been updated and expanded to make use of the new features of Model 10.

Its natural algebraic language and its many programming and editing features, such as program flags and relative addressing, make Model 20 ideal for users who want to do their own programming. These features are particularly appealing to research scientists and engineers. The peripheral control capabilities of the Model 20 also make it attractive for use as a controller in instrumentation systems [Nelson and Ricci, 1972].

Its larger memory, its array-variable capability, and its built-in tape cassette make Model 30 appealing to users with large programs and data bases, such as structural engineers and investment analysts. The alphanumeric keyboard, string-variable capability, and page-width printer appeal to users in fields outside the scientific, such as education and business. The programming language of the Model 30 appeals to a large number of users who already know BASIC as a time-sharing language. With an optional Terminal ROM, time-share users can transform the Model 30 into a versatile terminal with local as well as remote computation and storage capability.

With all three calculators, each user can specify a system of optional ROMs, peripherals, and read-write memory size to meet his own needs. This versatility is enhanced by user-definable keys, optional on the Models 10 and 20 and standard on Model 30. All three machines can also be expanded by the user of special machine language programs that can be loaded into read-write memory from a magnetic card or cassette. This capability can be

used, for example, to supply a software driver for a special peripheral.

The special features of each calculator along with the general purpose nature of the hardware are designed so that some combination of 9800 Series instruments will provide a solution to almost any programmable calculator application.

References

Hewlett-Packard [September 1968]; Nelson and Ricci [1972].

Model 10 Maintains Compatibility, Expands Capability[1]

Curtis D. Brown / Jack M. Walden

In keyboard language and appearance, Model 10 of the 9800 Series, or Model 9810A, is closely related to the 9100A/B Calculators, HP's first programmable calculators and the predecessors of the 9800 Series. Most of the 9810A keys are marked the same as the 9100A/B keys, and when used in the same way, perform the same operations. The same "reverse Polish" keyboard language is used. What's more, the keycodes stored in the program memory are the same when the keys are marked the same. This close similarity was maintained wherever possible to provide a useful carryover of the well established 9100A/B keyboard operations and associated programming techniques. However, 9100A/B operations are a *subset* of the 9810A's. Many new capabilities and features have been added in the new calculator.

In hardware implementation the 9810A bears no resemblance to the 9100A/B. Rather, it is similar to the other 9800-Series Calculators, Models 20 and 30. 9800-Series Calculators are all implemented from a common hardware base which is actually a 16-bit-word, general-purpose minicomputer. Individual calculator-model characteristics are obtained by internal ROM-stored machine-language programming. The unique hardware for each model consists primarily of the keyboard and display, which are tailored to the needs of the individual models.

The most important individual characteristics of the 9810A are:

1 A three-register (x, y, and z) light-emitting diode display
2 Separate memories for program and data storage
3 All-decimal addressing of program and data storage
4 Modular internal expansion of program and/or data storage
5 Indirect addressing for any register reference
6 Arithmetic operations (all four functions) into or from all data registers
7 Optional function blocks (ROM) to define the operation of the lefthand keyblock and other auxiliary functions, user-installable with ease.

[1] *Hewlett-Packard Journal*, vol. 24, December 1972, pp. 5–8.

9810A Hardware Features

The three-register LED display is one of the most conspicuous front-of-machine changes noticed by those familiar with the 9100A/B's CRT display. LED display was chosen because it is a bright, highly visible display whose brightness and size of characters are practically uninfluenced by line voltage, it fits in a small space, and its low supply voltages and signal levels interface directly to internal logic levels and supply voltages.

The magnetic card reader has a new feedthrough card path, allowing the use of longer cards than those used in the 9100A/B. The longer cards have greatly increased storage capacity, a necessity for making full use of the larger program and data memories of the 9810A and 9820A Calculators.

The 9810A and 9820A Calculators each have three sockets for plug-in read-only-memory (ROM) modules. These are direct extensions of the internal ROM. They allow expansion of operating features and redefinition of the lefthand keyblock. In the 9810A the lefthand keys may be defined at the user's option to provide standard mathematical functions (almost identical to the 9100A/B), statistical calculations, user-definable (keyboard language) functions, or programming aids.

The optional plotter, cassette, typewriter, and other peripherals are controlled by other plug-in blocks that are accessed by use of the FMT key on the keyboard. This plug-in block concept allows the user to configure and reconfigure the machine and peripheral-control facilities to suit the needs of the moment.

The thermal strip printer, housed within the calculator case, is controlled by internal programming. The basic 9810A provides for the printing of numeric values as they appear in the x register, or the listing of programs as they appear in the program-mode display. An optional plug-in block provides for message printing

and the addition of keycode mnemonics (functional abbreviations) to the program listings.

9810A Software Features

While the 9810A has a much larger bit-storage capacity than the 9100A/B, its memory is effectively made still larger by some new step-saving features. A major change from the 9100A/B is the decimal addressing structure for programs and data. The novice programmer adapts more quickly to decimal addressing, but a more important reason for its adoption is that it is necessary for the indirect addressing mode. An indirect reference to register *a*, say, will result in using the *value* of register *a* as a new register *address*. For example, the keystroke sequence x→, INDIRECT, *a*, will store the contents of the calculator's x register in the register whose address is given by the contents of *a*.

The indirect capability of the calculator can save many steps when a number of data registers are to undergo equivalent operations. One data register can be set aside as a pointer. The value of the pointer register designates one of the data registers to which the operations are to be applied. By incrementing the pointer register, a common subroutine can operate on the desired data registers in turn, saving many program steps over the direct reference method.

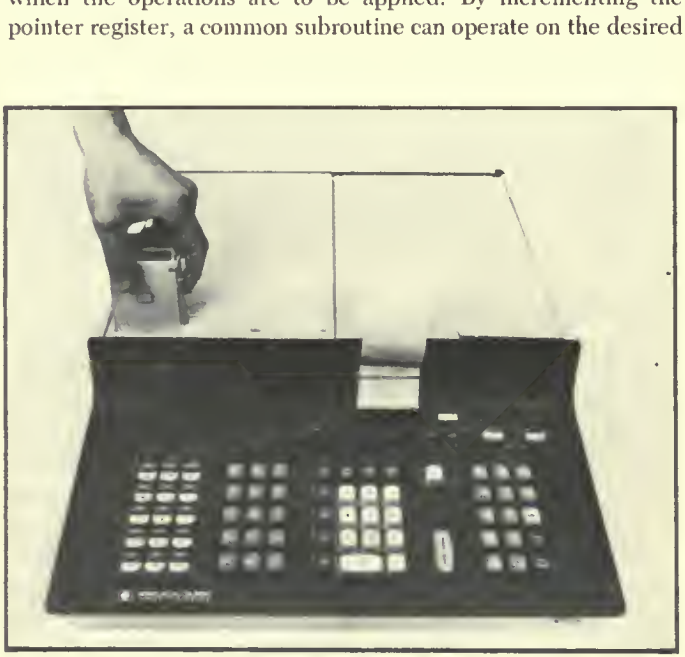


Fig. 1. Model 10 of the 9800 Series has all the capabilities of previous HP programmable calculators, plus many others. New features are the LED display, a larger internally expandable memory, decimal and Indirect addressing, arithmetic operations into or from all data registers, and optional plug-in function blocks to define the lefthand keyblock and other auxiliary functions.

Another important step-saving feature in the 9810A is the register arithmetic capability. Normally, if one wanted to add the x register to register 10, one would recall register 10 to the y register, add the x register, and store the results in register 10. With register arithmetic, however, a mathematical operator may be specified preceding the transfer address in a store or recall operation. One would then say x→, +, 10 to do the above operation.

This capability is bidirectional. Thus x←, +, 10 will add register 10 to the x register. Since any of the four arithmetic operations (+, −, ×, ÷) may be used, each register of the 9810A is in effect a powerful accumulator. This feature greatly increases programming flexibility by reducing the amount of shuffling of the x, y, and z registers.

Indirect addressing may be used with register arithmetic by including the INDIRECT key either before or after the arithmetic operator.

Because of the size of the 9810A memory, an improvement in programming and debugging ease over the 9100A/B was vital. Three new features attack this problem directly: label goto's; alphanumeric key mnemonics; and a printed record of entered keystrokes (keylogging.)

Labeled transfers are most useful during the program creation phase, where actual program step addresses either are not known or may change frequently as debugging progresses. Any step in the calculator program may be assigned a label by entering the LABEL key followed by any other keystroke. Control may later be transferred to that step by executing a GOTO or GOSUB LBL, then the same keystroke. A search is initiated, beginning at program step zero, and continuing through the program area until the label is found. Later, when the program is operating satisfactorily, absolute addresses may be substituted for each labeled GOTO; this gives a speed advantage by eliminating the search. (Model 9820A also has these capabilities.)

Alpha Printing

An alternative to remembering the numerical equivalents of all 64 keys is provided by an optional ROM, which generates three-letter mnemonics during listings on the thermal printer. These are useful for program debugging or documentation. This ROM also allows printing of messages during program execution, using the Format key to change the definition of the calculator keyboard. Two consecutive FMT keystrokes begin the character print mode. Keys that follow are interpreted as characters of the alphabet, rather than as their assigned function, and are printed a line at a time on the strip printer. Another FMT key terminates

the character print mode and restores normal calculator operation.

The keylog feature provides a printed record of all calculator keyboard operations. When the keylog mode is selected each key entered from the keyboard is automatically printed. If the calculator is in keyboard mode, the steps in a certain calculation may be verified. In program mode, the result is a full step-by-step listing of the program entered. With the optional plug-in ROM, mnemonics are printed along with the keycode.

Another feature that simplifies the operator's interaction with the 9810A is a backstep key. The backstep key decrements the calculator program step counter. It's helpful in examination of stored programs.

Flexibility through Plug-in ROMs

To one person, his 9810A may be an aid in statistics, to another it may be a purely mathematical or scientific machine, while to a third, it may be a peripheral controller. This changing nature of the 9810A is made possible by the plug-in ROM concept. The implementation of this concept posed a special problem for the machine coding. How can blocks yet to be conceived be slot-independent and still interface readily to the basic calculator?

The solution was to use the Format key to initiate a search through possible ROM block locations. The key following the FMT is compared with a special identifier word in each ROM, and when these codes match, the desired ROM has been found.

Many operations which could not be included in the keyboard directly are also implemented through the FMT key:

1) FMT ↑:	raise the plotter pen and move the coordinates given in the x and y registers.
2) FMT ↓:	lower the plotter pen and move to the coordinates given in the x and y registers.
3) FMT x→ x←	load or record data registers using the magnetic card reader.
4) FMT GOTO	load a magnetic card program at location zero, and begin execution at location zero (useful for chain-loading programs).
5) FMT CONTINUE	start the paper tape reader and prepare the calculator to accept information.

There routines are all contained in the basic 9810A.

Interactive Model 20 Speaks Algebraic Language[1]

Rex L. James / Francis J. Yockey

In Model 20 of the 9800 Series, or Model 9820A, the emphasis is on using the 9800 technology to provide a highly interactive calculator. Like the 9810A, the 9820A is a ROM-driven minicomputer. Its interactive nature stems mainly from its natural algebraic language and its built-in peripherals—keyboard, printer, and display.

Modularity provides another level of interactiveness by allowing the user to configure the 9820A to fit his application.

The display consists of a single register of sixteen alphanumeric characters. Each character is formed by a seven-row, five-column matrix of light-emitting diodes (LEDs). The printer and keyboard are similar mechanically to those of the 9810A (see box, page 808).

The 9820A's combination of fast LED display and quiet thermal strip printer allows a program to be run in an interactive mode unattainable before. The hard-copy results on the printer aren't cluttered with user instructions, since these appear in the LED display. User instructions appear instantaneously in the display; there's no need to wait for a printout.

When a key is pressed a mnemonic or character appears in the display to give instant visual feedback. The 16 characters are enough so that successive keystrokes can be seen in context. For example, the expression "if the square root of 6 equals A, go to line 17" would require the keystrokes

IF $\sqrt{6}$ = *A; GOTO 17*

and would appear in the display as

IF $\sqrt{6}$ = A; GTO 17

[1] *Hewlett-Packard Journal*, vol. 24, December 1972, pp. 8–13.

Natural Algebraic Language

Model 20 uses a powerful but natural instruction set to enable the user to solve complex mathematical problems quickly. The instruction set combines the features of the keyboard with the features of computer languages like ALGOL, FORTRAN, and BASIC. The result is a human-oriented, conversational approach to problem-solving, an approach that follows the structure of algebra in symbols and hierarchy.

A typical program for the 9820A is as follows. The program solves the quadratic equation $(-B\pm\sqrt{(BB-4AC)}/2A$.

```
 0: ENT "A VALUE",A
 1: PRT "A=",A
 2: ENT "B VALUE",B
 3: PRT "B=",B
 4: ENT "C VALUE",C
 5: PRT "C=",C
 6: IF 4AC>BB;GTO "IMAG"
 7: PRT "REAL ROOTS";SPC 1
 8: PRT (-B+√(BB-4AC))/2A;SPC 1
 9: PRT (-B-√(BB-4AC))/2A;SPC 9;JMP -9
10: "IMAG"
11: PRT "COMPLEX ROOTS";SPC 1
12: PRT "REAL", "IMAGINARY";SPC 2
13: PRT -B/2A,√(4AC-BB)/2A;SPC 1
14: PRT -B/2A, -√(4AC-BB)/2A;SPC 9; GTO 0
15: END
```

Notice in lines 8 and 9 of the program that the answer to the equation is programmed in the same way that the user would write it on paper. There are no artificial machine rules to remember. To maintain the structure of algebra, implied multiplication was implemented to avoid forcing the user to insert "*" between variables to be multiplied. Parentheses can be used and nested to any depth to change the order of evaluation of an algebraic expression.

Lines 0–5 demonstrate the interactiveness of the calculator. First the calculator stops and displays an alpha message of what is to be entered. The user then keys in the desired value. After RUN PROGRAM is pressed the calculator stores the value away and prints the label and the value of the entered data. In this way all three values A, B, and C may be entered. The roots then appear on the printout.

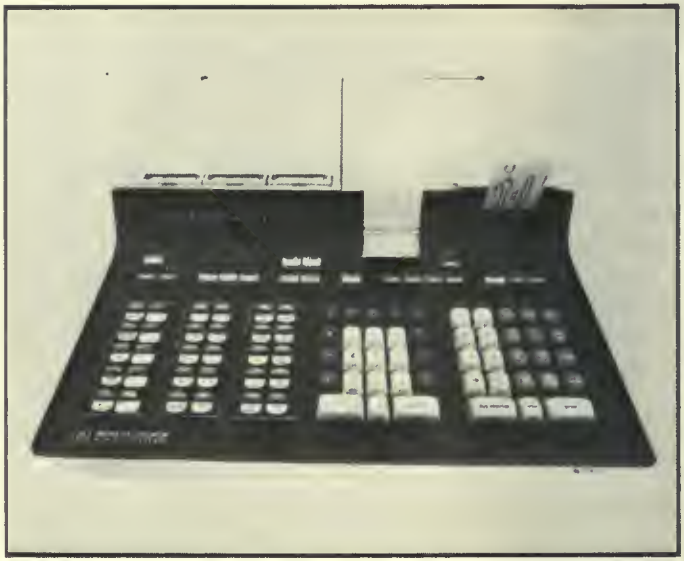


Fig. 2. Model 20 of the 9800 Series has a 16-character alphanumeric LED display that shows several keystrokes or program steps in context. Special features are a natural algebraic language, an interactive mode of operation, and plug-in modules that define the functions of the three lefthand keyblocks.

Editing

Convenient editing features have been included in the 9820A. To edit the above program to change to the absolute form of the GOTO, the user would key in *GOTO 6 RECALL.* Line 6 will be recalled to the display. Hitting the back key 7 times will give the following display:

```
6:IF 4AC>BB; GTO
```

To finish the edit, the user keys in *1 0 STORE* to form the new line of program. Since the label is no longer needed in line 10, it can be eliminated by keying in GTO *1 0 RECALL DELETE.*

The new listing is as follows:

```
 0: ENT "A VALUE",A
 1: PRT "A=",A
 2: Ent "B VALUE",B
 3: PRT "B=",B
```

```
 4: ENT "C VALUE",C
 5: PRT "C=", C
 6: IF 4AC>BB;GTO 10
 7: PRT "REAL ROOTS";SPC 1
 8: PRT (-B+√(BB-4AC))/2A;SPC 1
 9: PRT (-B-√(BB-4AC))/2A;SPC 9;JMP -9
10: PRT "COMPLEX ROOTS";SPC 1
11: PRT "REAL", "IMAGINARY";SPC 2
12: PRT -B/2A,√(4AC-BB)/2A;SPC 1
13: PRT -B/2A, -√(4AC-BB)/2A;SPC 9;GTO 0
14: END
```

Notice that the label has been deleted and all the lines below it have been moved up and renumbered.

With the editing keys, the user can REPLACE, INSERT, or DELETE any line or character. The user observes all of the changes as they take place, by watching the alphanumeric display.

Machine Language

An algebraic language is easy for humans to understand and use, but is difficult for a machine to understand and execute. Take the example:

$$A*B + C*D \qquad (1)$$

When operators appear between operands as in equation 1, the precedence of the operators becomes important in the sequence of execution. Since multiply is normally assigned a higher precedence than addition, those operations associated with multiplication are performed before addition.

Equation 1 can be rewritten so that operations are executed as they are encountered:

$$A\ B\ *\ C\ D\ *\ + \qquad (2)$$

This notation is known as "Polish," or more correctly, "reverse Polish" notation.

When equation 2 is executed, operands are passed directly to a stack, which is a temporary holding area organized so that the first item into the stack will be the last item out. When a binary operator is encountered, it is applied to the top two values of the stack. The specified operation is performed and a single result is returned to the stack. Several forms of Polish notation are widely used by most desk calculators today, including the 9810A. Its main advantages are that it is easy to implement, it has fast execution speed, and it allows compact storage of programs. Its main disadvantage is that it isn't natural to the untrained user.

9820A Has Compiler

To take advantage of both the naturalness of the algebraic language and the speed and compactness of Polish notation, the 9820A's algebraic language is compiled into a machine language similar to the reverse Polish notation shown in equation 2. The compiler flowchart is shown in Fig. 3, along with an example showing the process of compiling the equation.

$$A+B\uparrow(C*D/(D+F)*G)$$

As a string of algebraic codes is input, the compiler forms a string of machine language codes. During the compiling operation a stack is used to hold the operators and establish their order of appearance in the compiled string. All operands are passed directly to the output string while operators are put into the stack.

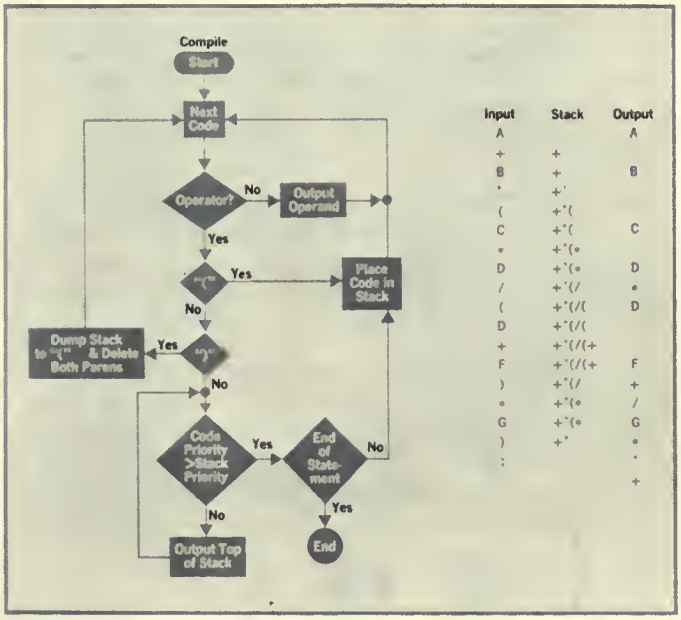


Fig. 3. Model 20 compiles algebraic-language user programs into a faster-executing machine language. Only the compile version is stored.

Printer and Keyboard for Models 10 and 20

The 9810A and 9820A Calculators use the same thermal printer and keyboard design.

The printer (Fig. 1) prints lines of sixteen alphanumeric characters on heat-sensitive paper. A five-by-seven dot matrix is used to form each character. The printer has a row of print elements distributed linearly across its printing head. Each print element is an electrical resistor of the right size and shape to produce a dot on heat-sensitive paper moved at a right angle to the line of print elements. Dots are formed in the conventional manner by exciting a resistor element with a pulse of electrical current, which raises its temperature.

The printer produces each line of print by printing the top row of all sixteen characters, then stepping down to print the second row, and so on until all seven rows are printed. Three blank steps are then added to produce the space between lines.

The printer is quiet and adaptable and has a minimum number of moving parts—all in the paper advance.

The keyboard is a contactless unit made up of an array of printed circuit transformers (Fig. 2). The secondaries of all the coils are tied in series to form the sense line (Fig. 3). The primaries of the coils are arranged in pairs. Each pair is connected in series with opposite polarity. Every pair has a drive and sink line, which is selected and driven by the scanner.

Centered above each coil is a metal disc attached to the end of the key shaft. When a key is pressed the disc moves closer to the coil. The disc acts like a shorted turn, reducing the coupling of the coil and unbalancing the pair. This unbalance is amplified by the comparator when it is greater than the "on" bias. The comparator triggers the one-shot multivibrator, which turns off the scanner and lowers the "on" bias. The scanner remains in the same state, which corresponds to the drive and sink line of the key that was pressed. This state is the keycode of the key pressed.

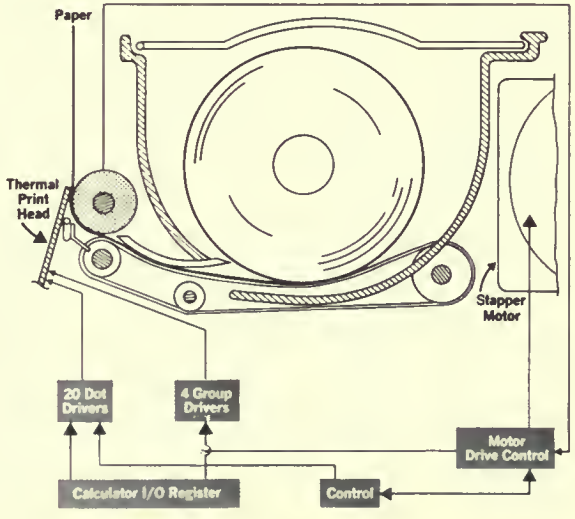


Fig. 1. *Thermal printer has few moving parts.*

When the key is released a spring retracts the key and disc. When the unbalance is less than the lowered "on" bias the comparator turns off and the scanner starts again, ready for a new keystroke. The two bias levels give the key mechanical hysteresis.

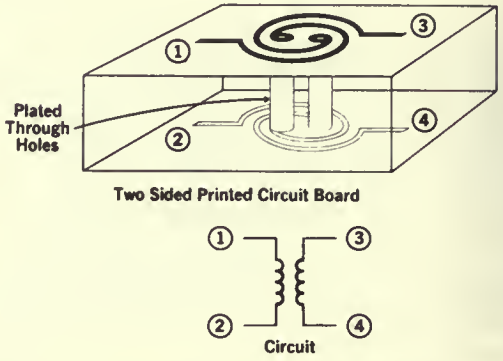


Fig. 2. *Printed-circuit transformers, one for each key, are used in the contactless keyboard.*

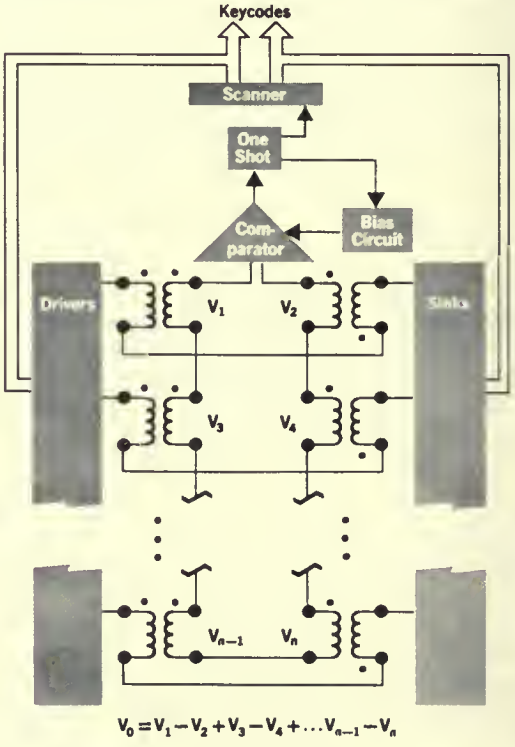


Fig. 3. *Pressing a key unbalances one of the pairs of transformers and causes a keycode to be transmitted by the scanner.*

Before an operator is placed into the stack, the stack is checked to see that all operators of greater or equal priority are first output.

Parentheses can be used to change the normal order of execution of an algebraic statement. The left parenthesis has the effect of temporarily resetting the compiler for the evaluation of the string of codes found inside the parentheses. The right parenthesis will then cause all operators in the stack to be output until a left parenthesis occurs. However, neither of the two parentheses are needed in the compiled code.

Compilers of this and more complex types have been used for years in computers. When changes to the program have to be made, the source cards or paper tape are changed accordingly and the program is recompiled. With a desk calculator, this operation is too severe a penalty to pay. Also, it isn't possible to store both the source code and the compiled code in the calculator memory at the same time. It's necessary, therefore, to reconstruct the code for editing and program listings.

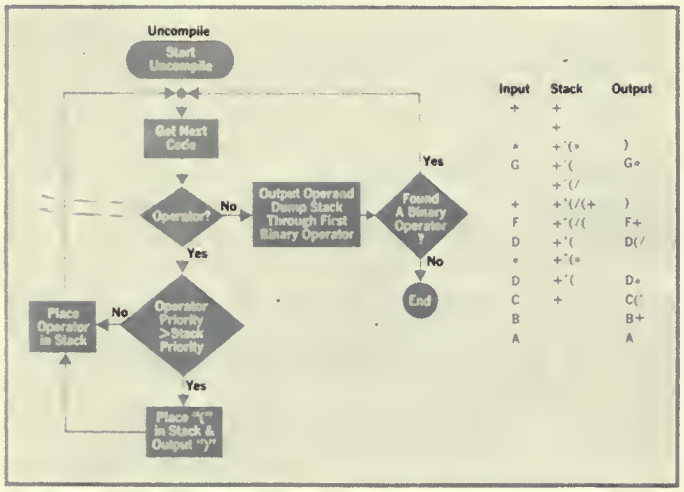


Fig. 4. An uncompiler reconstructs the original program for editing or listing.

Uncompiler

The solution to this problem is the concept of the uncompiler (see Fig. 4). With the uncompiler, it's possible to take the compiled code as input and reconstruct the original algebraic form. The code is scanned backwards. Parentheses are inserted where needed in the reconstructed code, and redundant parentheses are omitted. For example,

(A*B) + (C*D)

will be reconstructed as

A*B + C*D

after going through the compile/uncompile process.

With the compile/uncompile feature, the 9820A only has to store the compiled form of the code. In the 9820A, a line of program is the basic unit used for the compiler/uncompiler and editing features. As a line of program is entered, it is stored in a buffer area and displayed in its algebraic form. When the STORE key is hit, the line of program is compiled and stored away in the program area.

When the user chooses to edit a line of program, the program line is located in memory, uncompiled into a buffer area, and displayed in its reconstructed form. Now editing can be performed. When the editing is finished, the line is compiled and once again stored in memory. To the user, the compile/uncompile process is transparent except for a slight pause while storing or recalling a line of program.

Modularity

Another level of interactiveness is brought about by the modularity of the memory structure and the general-purpose minicomputer heart of the calculator. The 9820A can be configured to various applications in three ways: additional keyboard functions by read-only-memory additions, additional read/write memory, and addition of external peripherals.

There are thirty keys, arranged in three blocks of ten, available to the user to be defined for his special needs by means of plug-in ROM. Some plug-in modules provide mathematical functions, others give high-level-language control of external peripheral devices, and another allows users to define subroutines for their own special functions.

Because MOS RWM is used, additional user memory may easily be added to the basic machine. The fully loaded 9820A has 9K (9216) 16-bit words of memory: 7K is ROM and 2K is RWM.

The I/O structure provides four I/O slots on the back of the calculator to accept interface cards for peripheral devices. An I/O expander augments the four I/O slots of the calculator.

Another Program Example

A good example of Model 20's interactive nature is this Butterworth filter design program.

Display: NUMBER OF POLES?
Key in: 3
Display: CENTER FREQUENCY
Key in: 10000000
Display: BAND WIDTH?
Key in: 1000000
Display: RESISTANCE?
Key in: 50
Display: REALIZATION TYPE
Key in: 2

Realization Type 2 specifies the physical configuration of the circuit.

With this input the output shown in Fig. 5 comes from the printer.

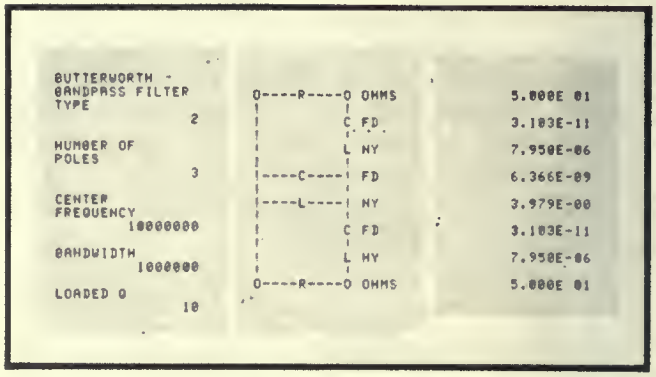


Fig. 5. Butterworth filter design program demonstrated interactiveness of calculator. Printer output is shown here; user instructions appear in display.

BASIC-Language Model 30 Can Be Calculator, Computer, or Terminal[1]

Richard M. Spangler

Model 9830A is the latest and most powerful calculator in the 9800 Series. Its keyboard design, programming language, memory size, I/O capability, and flexibility make the 9830A more like a desktop computer than a calculator. Yet it maintains the convenience and user interaction that makes a programmable calculator so easy to use. The user can still set the machine on his desk, turn the power on, type in 2+2, and see 4 on the display.

Like the 9810A and 9820A, the 9830A (Fig. 6) is a ROM-driven general-purpose minicomputer with specialized peripherals built in. In its minimum configuration, the 9830A contains 7½K words of read-only memory (ROM) and 2K words of read/write memory (RWM). The memory is expandable to 16K words of ROM and, initially, to 4K words of RWM. The display register contains 32 alphanumeric characters, and uses the same 5 × 7 LED dot matrix as the 9820A.

A built-in tape cassette unit is included in the mainframe in place of the magnetic card reader used in previous HP calculators. The I/O structure of the 9830A is identical to that of the 9810A and 9820A, so all the 9800-Series peripherals operate with the 9830A. The new 9866A page-width thermal line printer (see Fig. 1) is designed to be the primary output peripheral for the 9830A. It fits directly on top of the calculator. It can print 80-column lines at a rate of 250 lines per minute.

[1] *Hewlett-Packard Journal*, vol. 24, December 1972, pp. 14–18.

Fig. 6. Model 30 of the 9800 Series has an alphanumeric keyboard like a teleprinter. Its language is BASIC. It can be used as a desktop computer or a remote computer terminal, yet it maintains the convenience and user interaction of a programmable calculator.

Alphanumeric Keyboard

The keyboard of the 9830A represents its most significant departure from the traditional concept of a programmable calculator. It is not a key-per-function keyboard, but rather an alphanumeric keyboard like that of a typewriter or teleprinter terminal.

Besides the alpha section of the keyboard, there are three groups of special keys that facilitate use of the 9830A. The first group is a calculator section, which contains the digits and the

Fig. 7. Model 9866A Thermal Line Printer is the primary output peripheral for Model 30.

most commonly used arithmetic operators. The second group contains special control keys used in operating, editing, and debugging programs. The keys in the third group are definable by the user.

The use of an alphanumeric keyboard rather than a key-per-function keyboard removes the major restriction to programming language definition and language expandability. It is not necessary to add a new key to the keyboard whenever a new function is added to the language. Rather, a new function is assigned a mnemonic which can be entered as a sequence of alpha characters.

BASIC Language

In the development of the 9830A, it was decided not to define another new and unique programming language. The language of the 9830A is BASIC, which is well known among users of small computer and time-shared systems. All of the changes that have been made in BASIC in the 9830A are additions to standard BASIC, so programs written in versions of BASIC that are close to the standard version will run on the 9830A with little or no modification. This means that a tremendous program library is already available.

Besides being well known and used extensively, BASIC has several other characteristics which make it well suited for a programmable calculator. Since the language was originally designed for use in time-sharing environments, it is interactive and conversational. The 9830A fully exploits these characteristics by communicating through the 32-character alphanumeric display and the thermal line printer.

BASIC is easy to learn because the commands closely resemble English and there are very few tricky syntax rules to memorize. Each statement in a program is given a line number by the programmer, and the BASIC operating system automatically places the statements in order.

Program editing is easily accomplished simply by retyping any incorrect statement or assigning a line number between two existing lines to a statement to be inserted in a program. The 9830A has expanded on this editing capability by providing complete character-by-character editing. The user may recall a line of program to the display, edit the characters within that line, and store the corrected line without retyping the whole line. If an error is made while typing a statement, the incorrect line can be recalled and edited.

BASIC is also well suited for implementation by an interpreter rather than a compiler. With a compiler, the user's program is transformed before execution time into machine-language instructions, which are executed directly by the machine processor. With an interpreter, the user's program remains in memory in source form and an interpreting program examines the source program and calls on the appropriate execution routines. The main advantage of an interpreter in an interactive system like the 9830A is that only one copy of the user's program is needed for program editing and execution. With an interpreter, only minor additions are required to implement calculator functions such as TRACE or single STEP, or execution of statements directly from the keyboard.

Features of BASIC

The BASIC language has several important features which are new to programmable calculators. The most important is the type of variables that are allowed. Simple numeric variables, single- and double-subscripted array variables, and string variables may all be used. Array variables permit the analysis of large numbers of data items. String variables, which are strings of alphanumeric characters, permit such things as names and addresses to be analyzed and stored. Each variable is named by any letter of the alphabet, so the user can use R for resistance, Q for quota and N for the number of elements in an array. The calculator interpreter reserves only enough memory space for the variables currently in use.

Another feature of the BASIC language is user-definable functions. Standard BASIC restricts these functions to single arithmetic expressions, and allows only one parameter.

Like standard BASIC, 9830A BASIC allows only one parameter. However, the definition of the function can consist of more than one statement. For example, a function to evaluate N factorial may be defined as follows:

```
100 DEF FNF (N)
110 N1 =1
120 FOR N2 = 1 TO N
130 N1 = N1 * N2
140 NEXT N2
150 RETURN N1
```

To assign a function to a user-definable key, the user presses the key labeled FETCH and then presses any of the ten definable keys. This puts the calculator into the key-definition mode. The user then enters his function and presses the END key. Now whenever he presses that particular key, the calculator responds with the name of the function—for example FNF for the factorial function. The user can then enter the argument, 5 for example, followed by the EXECUTE key. The calculator responds with the answer 120. Functions assigned to keys can be called either from the keyboard or from a program.

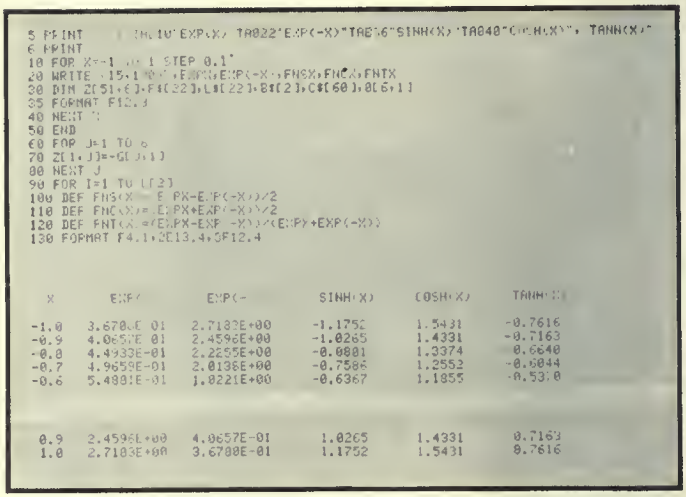
Single- or multiple-line functions are one of three categories of operations that can be assigned to the user-definable keys. These keys can also be used to store entire programs.

The user may also assign typing aids to his user-definable keys. A typing aid is simply a string of alphanumeric characters. Whenever a typing aid key is pressed, the characters that are assigned to that key are entered into the display just as if those characters had been entered individually from the keyboard. For instance, each of the keys could be assigned a mnemonic such as PRINT, INPUT, READ, and so on. These keys could then be used in typing BASIC programs. These functions of the user-definable keys make the 9830A act more like a calculator.

Output Formatting

A severe limitation of BASIC is its restricted formatting capability. In the 9830A, four new statements have been added to make output formatting more flexible. Two statements, FIXED and FLOAT, allow the user to specify the format for the numeric output in his PRINT or MAT PRINT statements. Two other statements called WRITE and FORMAT give the user formatting capability similar to FORTRAN. Fig. 8 illustrates the 9830A's formatting ability.

A series of tape operating commands has also been added to 9830A BASIC to control the built-in tape cassette. A command called MARK is used to initialize a cassette and set up a structure of fixed length files. These files can then be accessed randomly by file number. Three types of information can be stored and recalled from the cassette: user programs, numeric and string data, and sets of user definable keys. The command structure is simple yet flexible.



```
5 PRINT       TAB(10"EXP(X)"TAB22"EXP(-X)"TAB36"SINH(X)"TAB48"COSH(X)"; TANH(X)"
6 PRINT
10 FOR X=-1 TO 1 STEP 0.1
20 WRITE USING 15:1 X,EXP(X),EXP(-X),FNS(X),FNC(X),FNT(X
30 DIM Z(51,6),F$(22),L$(22),B$(2),C$(60),B(6,1)
35 FORMAT F12.4
40 NEXT X
50 END
60 FOR J=1 TO 6
70 Z(1,J)=-B(J,1)
80 NEXT J
90 FOR I=1 TO LEN(Z)
100 DEF FNS(X)=(EXP(X)-EXP(-X))/2
110 DEF FNC(X)=(EXP(X)+EXP(-X))/2
120 DEF FNT(X)=(EXP(X)-EXP(-X))/(EXP(X)+EXP(-X))
130 FORMAT F4.1,2E13.4,3F12.4


   X        EXP(X)        EXP(-X)        SINH(X)       COSH(X)       TANH(X)

-1.0    3.6788E-01    2.7183E+00    -1.1752       1.5431       -0.7616
-0.9    4.0657E-01    2.4596E+00    -1.0265       1.4331       -0.7163
-0.8    4.4932E-01    2.2255E+00    -0.8881       1.3374        0.6640
-0.7    4.9659E-01    2.0138E+00    -0.7586       1.2552       -0.6044
-0.6    5.4881E-01    1.8221E+00    -0.6367       1.1855       -0.5370


 0.9    2.4596E+00    4.0657E-01     1.0265       1.4331        0.7163
 1.0    2.7183E+00    3.6788E-01     1.1752       1.5431        0.7616
```

Fig. 8. Sample 9886A printout shows the extended formatting capability of the 9830A.

Add-on ROM

The most unique feature in the 9830A BASIC interpreter is its modularity. Each statement or function is accessed through a series of tables in ROM. Tables can be accessed on as many as eight optional add-on ROM modules, or even in the read-write memory. These add-on ROM modules, each containing 1024 words, are available both in small plastic cases and as printed-circuit modules that can be plugged into the 9830A. After a ROM is in place, the calculator can understand the commands implemented by that ROM and the interpreter can jump to the execution routines stored in it.

Five add-on ROMs are now available. The Matrix and String Variable ROMs include commands that are part of many BASIC systems, but are not needed by all BASIC users. The MAT commands on the Matrix ROM allow initialization of an array to all ones, all zeroes, or the identity matrix, or reading of the values for an array from DATA statements. Matrix arithmetic functions—addition, subtraction, multiplication, and multiplication by a scalar—are easily called for, and functions to take the inverse and transpose are also included. This ROM also includes two commands that are not common in BASIC. They are a REDIM statement to redimension an array without changing the values of any elements, and a DET function for taking the determinant of a matrix.

The String Variable ROM allows the BASIC program to handle strings of alphanumeric characters. The program can initialize, change, examine, and test these strings, and it can ask the operator to input character strings through the keyboard. The simplest example of the value of string variables is an operator

typing "YES" or "NO" in response to a question posed by the program. This add-on ROM makes the 9830A truly conversational.

The Plotter ROM adds several new statements to the 9830A language and provides the drivers needed to control the 9862A X-Y Plotter. Some of the most significant capabilities added are automatic scaling, convenient axis drawing, absolute and incremental plotting, and plotting relative to any origin. Labeling plots and axes has been made simple by a LABEL statement. This statement allows the user to draw alphanumeric characters of any height and width, at any angle of rotation.

The Extended I/O ROM adds statements and functions which provide convenience and flexibility in controlling input and output peripherals. The two most important features in this block are an ENTER statement that is used to input data from a peripheral in either free field or formatted form, and an automatic code conversion capability which allows the 9830A to communicate with peripherals using character codes other than ASCII. The Extended I/O ROM communicates through the standard interface scheme of the 9800 Series, and uses the standard interface cards.

Terminal Capability

The fifth ROM that is presently available with the 9830A is a Terminal ROM. This ROM gives the 9830A the unusual capability to act as a computer terminal. It can communicate with a time-sharing service through a modem at any speed from 3 to 300 baud. This optional ROM overrides the standard keyboard input routine and bypasses the syntax routines so that lines of free text can be stored in memory. For example, a FORTRAN program may be entered into the 9830A, edited, and saved on cassette. After a program has been entered and edited, the user can call up his time-sharing service and have the 9830A transmit the program automatically. The user may also have his time-sharing service transmit a program to be saved in the memory of the 9830A. The editing, execution and storage capabilities of the 9830A make it a very powerful computer terminal.

Modular Firmware

Underlying the modularity and expandability of 9830A BASIC is the modular structure of the firmware (machine-language programs) stored in ROM in the 9830A. Fig. 9 is an overall block diagram. Each shaded block can accept optional ROMs to expand its capabilities.

The keyboard input routine and keyboard monitor perform the user interaction and editing functions. The syntax routines, the pre-execution processing routines and the statement execution routines are the essential elements of the BASIC interpreter. There is a separate syntax routine and execution routine for each different statement type.

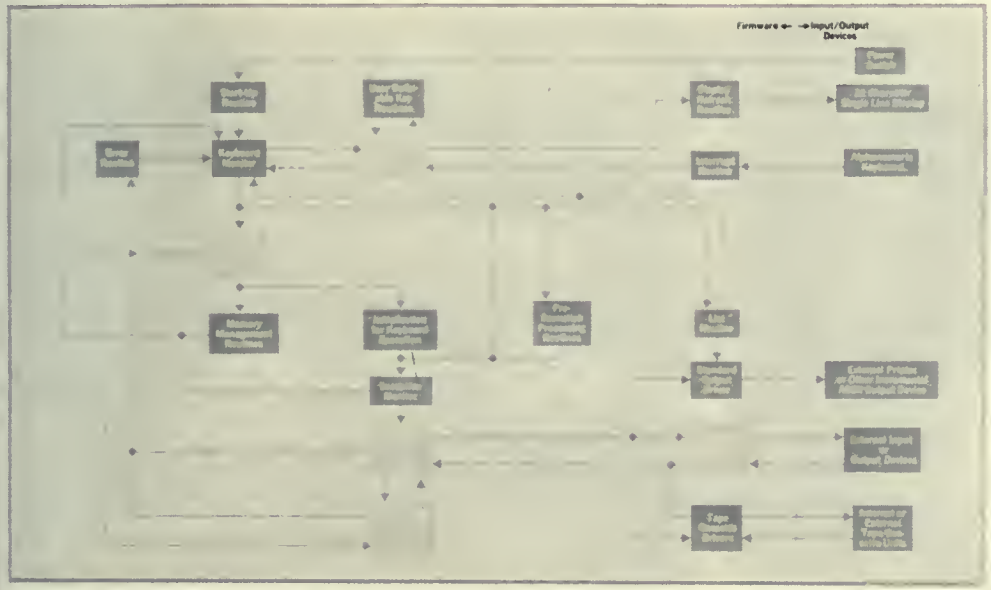


Fig. 9. 9830A BASIC is implemented by firmware routines stored in read-only memory. Modularity makes it easy to expand the language by adding plug-in ROM. The shaded blocks are the expandable modules.

The syntax routines accept an input record from the keyboard routines. This record is a string of the characters that were entered at the keyboard. The syntax routines examine the input record, character by character, checking for proper statement syntax, and transform the string of characters into a series of operation and operand codes that can be more easily used by the execution routines.

The key to the modularity of the syntax firmware is a table search routine which scans a series of name tables on each of the option-block ROMs and the main system ROM. The table search routine searches for a match between the characters in the input record and the characters in the tables in ROM. If a match is found, two codes are stored in the translated format of the input record, a code for the command, and a code for the ROM block where the command is located.

Two codes of information are also stored for each operand, a five-bit code for the letter naming the operand, and a five-bit code for the variable type. The type code is used to distinguish, for example, A, A(1), A1, and A$, which all have the same variable name. During syntax analysis, numeric constants are converted to a floating point format, and line numbers are converted to 16-bit integers. After syntax analysis, control is passed to the memory management routines to store the statement in program memory, or to the initialization routine, which prepares the calculator to execute the statement directly.

The interpreter uses a symbol table to keep track of the variables that are currently in use. It is the job of the pre-execution processing routines to set up this symbol table at the start of program execution. When the RUN command is given, old symbols are deleted, and then the current program is scanned. Any array variable names and their dimensions are saved in the new symbol table.

After the program scan, storage is reserved for each array and a pointer giving the starting memory address for each array is saved in the symbol table. A special key causes the pre-execution processing phase to be performed without the execution phase. This allows the user to set up his symbol table for array variables so they can be used from the keyboard.

Symbol table entries for simple variables and user-defined functions are made during the execution phase. This allows simple variables to be defined by a statement executed directly from the keyboard. The calculator user needn't be aware that a symbol table is being used. Any variable he wants to use is available immediately.

The execution monitor uses the code stored with each statement to locate the proper block of memory and branches to the execution routine for that statement. The execution routines examine the operation and operand codes and call upon subroutines to perform the arithmetic execution. The statement execution routines also call upon driver routines for control of the cassette, printer, display and any external input or output devices.

9800 Processor Incorporates 8-MHz Microprocessor[1]

Henry J. Kohoutek

The processing unit for HP 9800-Series calculators is a microprogrammed 16-bit serial processor that is capable of executing 75 basic machine-language instructions. The processor

- controls the data flow between memory and working registers,
- performs logical and binary or decimal arithmetic operations on data in the working registers,
- performs logical decisions (branching) based on the states of 16 qualifiers (carry/borrow, operation codes contained in machine-language instructions, etc.),
- controls the internal clock for variable-cycle-time microprogram steps, and
- transfers control to the I/O controller for input and output instruction execution.

The processing unit is implemented with MSI bipolar logic circuitry with strong emphasis on read-only memories. Central control of the processor, memory, and I/O unit is nested in microprograms stored in these ROMs in the microprocessor section of the processor (see Fig. 10). The microprocessor executes machine-language instructions in cycles by following these microprograms.

It's important to note that there are two levels of ROM in 9800-Series Calculators. Keystrokes or user program statements initiate sequences of *machine-language instructions.* These sequences are stored in MOS ROMs that are part of the memory system. For each of the 75 machine-language instructions there is a sequence of microinstructions stored in the microprocessor's bipolar ROM. The ROM modules that plug into 9800-Series

[1] *Hewlett-Packard Journal*, vol. 24, December 1972, pp. 19–22.

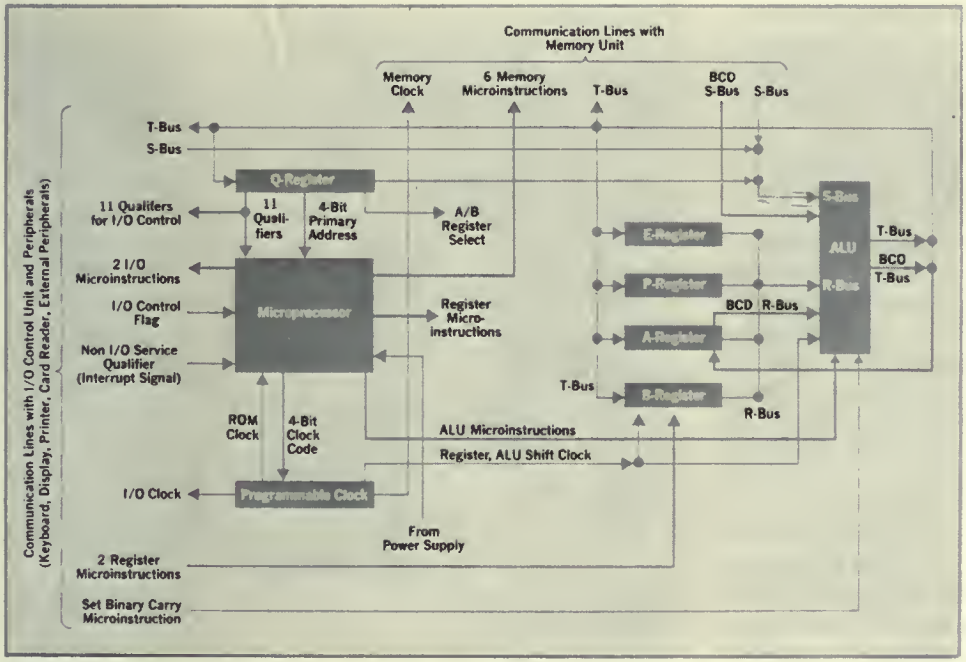


Fig. 10. Processor organization features three buses, five working registers, microprocessor, and arithmetic/logic unit (ALU).

Calculators expand the higher-level MOS ROM, not the microprocessor ROM, which is the same in all models.

The microprocessor ROM, which holds the microprogrammed execution routines for individual machine-language instructions, consists of a block of seven bipolar read-only memories organized in 256 words of 28 bits. Fast routine execution times, based on an internal clock frequency of 8 MHz, help speed up all keyboard functions.

Fig. 11 is a block diagram of the 9800 processor, showing its organization and its relationships with the memory and input/output control unit. The processor has an R-S-T bus configuration. Two buses, R and S, carry data to the arithmetic/logic unit (ALU), and the third bus, T, carries the ALU output.

There are five principal working registers which communicate via the bus system and the ALU, under control of the microprocessor's instruction logic and the number of shift clock pulses that have occurred.

P-register is the calculator's program counter. By going through a step-by-step counting sequence, it causes successive instructions to be read out of the memory. The sequential stepping can be altered by execution of skip or jump instructions, thus causing the program to continue at a different memory address. During

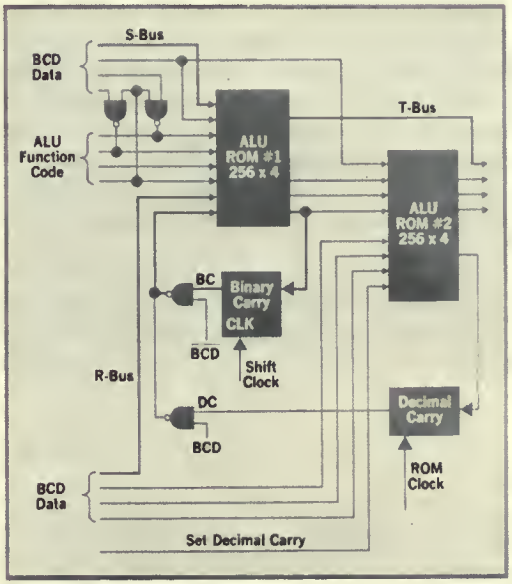


Fig. 11. Arithmetic/logic unit (ALU) performs binary arithmetic and logic operations and binary-coded-decimal operations.

execution of some instructions, the P-register contains a special binary word that is used to simplify digit and word counting.

A-register is one of the calculator's two accumulators. It is capable of accepting results of both binary and decimal arithmetic operations. When a decimal operation is performed, the four-bit result is temporarily received in bits A_3–A_0 of the A register.

B-register is another accumulator. It has the same capabilities as the A-register except for decimal arithmetic.

E-register represents a flexible four-bit extension of all other registers. It's used for left and right shifts with binary-coded-decimal data occupying several memory locations.

Q-register contains the program instruction currently being executed. Its individual bits can be tested as qualifiers to perform microprogram branching according to the instruction code. In the final part of microprogram routines when the instruction code has been fully recognized, the Q-register is used for temporary storage of internal processor information.

The programmable clock contains the system clock generator, along with logic which, by decoding the clock field of the microinstruction, causes the correct number of shift pulses to be issued to the working registers and the ALU. This scheme makes it possible to have variable cycle time for each state of the microprogram, and results in a substantial saving in microprocessor ROM. A ROM clock pulse occurs once for each microprogram state and is applied to the ROM address flip-flops.

The binary/BCD arithmetic logic unit (ALU) performs one-bit binary logic and arithmetic operations, as well as four-bit binary-coded-decimal arithmetic operations. Coded results for all logic and arithmetic operations are nested in a form of special look-up table on two bipolar 1024-bit ROM's. Data from working registers and the carry flip-flop, together with the microinstruction to be executed serve as ALU inputs. These inputs define a unique ROM address where the proper result is encoded, and gates are enabled to place this result on the ALU output lines. The states of two ALU carry flip-flops are communicated to the microprocessor where they are recognized as special qualifiers. The ALU organization is shown in Fig. 11.

The processor communicates with the memory unit and the I/O control unit via the T-bus and the S-bus and by special groups of memory and I/O microinstructions. The processor's clock circuitry synchronizes all units by generating memory clock pulses and I/O clock pulses.

Microprocessor

Detailed structure of the microprocessor is shown in Fig. 12.

The primary and secondary address flip-flops form a microprogram counter, which selects the memory location where the microinstruction to be executed is stored. Each microinstruction is 28 bits wide and contains information to control the data flow in the system by enabling appropriate gates and generating the proper number of shift clock pulses.

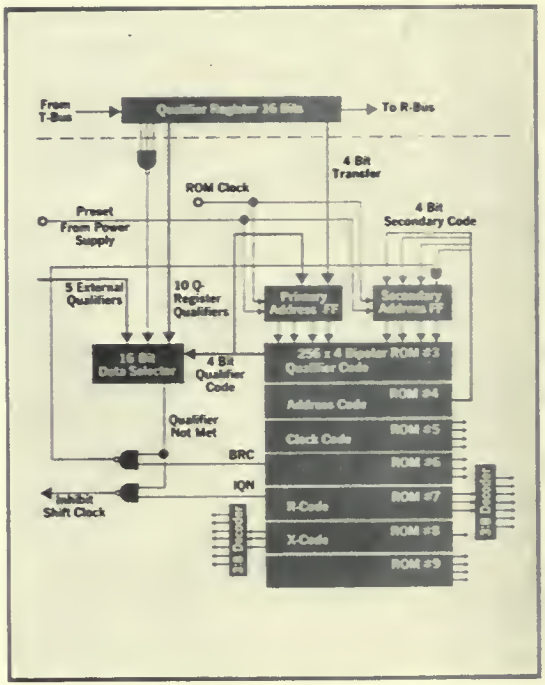


Fig. 12. Microprocessor read-only memories contain sequences of microinstructions that control execution of keyboard instructions.

Also included is information to define the ROM address of the next microinstruction. Thus instead of being limited to a fixed address sequence, the microprogram may in effect execute almost a random walk through the ROM addresses.

The microinstruction format is shown in Fig. 13. The four-bit qualifier code in each microinstruction serves a dual purpose. If branching is desired, the microinstruction BRC must be programmed. If the preceding microinstruction is BRC, the four-bit qualifier code selects the proper qualifier to be tested and the primary address of the next microinstruction is the same as the current one. If the preceding microinstruction is not BRC the qualifier code defines the primary address of the next microinstruction.

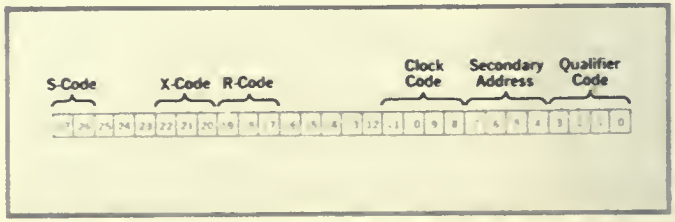


Fig. 13. Microinstruction format.

A single-chip 16-bit data selector permits any one of the 16 qualifiers to be tested according to the qualifier code. If branching is to occur, the microinstruction BRC, along with a signal from the data selector, defines the least significant bit of the secondary address of the next microinstruction, according to the result of the qualifier test.

A special microinstruction, IQN, inhibits all shift clock pulses from the clock decoder in case the selected qualifier condition was not met. This in effect prevents execution of microinstructions in that ROM state.

To minimize the microinstruction width the operation codes for clock decoder, ALU, bus-gate control, and so on, are coded into groups and decoded by hardware into individual signals.

Besides the 75 basic machine-language instructions, the system can also perform indirect memory calls, interrupts, I/O calls, and a simple resident diagnostic of its own performance in start-up conditions.

Testing in Production and Service

The microinstructions that control the calculator's processor and memory define the lowest language that can control the machine hardware. Therefore, for testing 9800 Series Calculators on the production line and in the field, a tester was designed to execute machine diagnostics on the microprocessor level, following a "start-small" strategy.

The tester organization is very similar to that of the microprocessor, but instead of machine-language execution routines, a system of tests is nested in a group of ROM's. Virtually identical organization, hardware, and timing of microprocessor and tester assures similarity between working and testing conditions from a physical and an electrical point of view. This means that the diagnostic information represents a realistic picture of the state of the tested machine.

The tester hardware also contains logic for convenient manual operations, simple aids for troubleshooting in case an error is detected, and circuitry for computer interface.

Test routines are organized in a sequence. There is a pretest, a series of 22 tests, and a posttest to check magnetic-card-reader mechanics. The pretest is a manually controlled resident microdiagnostic routine designed to test the tester's hardware. The start-small strategy is reflected in the test sequence, which begins with very simple tests of binary logic functions of the ALU, and continues through register tests to complex tests of the entire ALU and memory. Each test uses only successfully tested parts of the machine hardware and evaluates only a small new part of the hardware. This makes it easy to locate failures when an error is discovered.

Versatile Input/Output Structure Welcomes Peripheral Variety[1]

Gary L. Egan

The input output structure of 9800-Series Calculators, which links the calculator with its peripherals, is designed to be versatile and easy to use. It is flexible enough so a user can easily interface his calculator with a variety of HP peripherals as well as with many standard units and others of his own design.

I/O Processor

The input/output processor is a self-contained microprocessor composed of commerically available TTL logic circuits which generate the microinstructions necessary to implement the ten input-output instructions. The I/O processor is fully synchronous with the system clock and main processor, receiving starting control from the main processor whenever an input-output instruction is read from memory. While the I/O processor is in control, the main processor remains in a two-state waiting loop until the input-output instruction has been implemented, whereupon control is returned to the main processor. (See Fig. 14.)

The input/output instructions require six to twelve microseconds to execute. There are I/O instructions for setting or clearing flip-flops, for testing the state of flip-flops, and for moving data between registers in the main processor and the input/output register.

I/O Register

The I/O register is a 16-bit universal (parallel in/out, serial in/out) data register that is connected to the main processor by the serial bus system. Data contained in the I/O register is sent bit-serial into the main processor via the S-bus. Conversely, bit-serial data is received from the main processor by the I/O register via the T-bus.

[1] *Hewlett-Packard Journal*, vol. 24, December 1972, pp. 24–27.

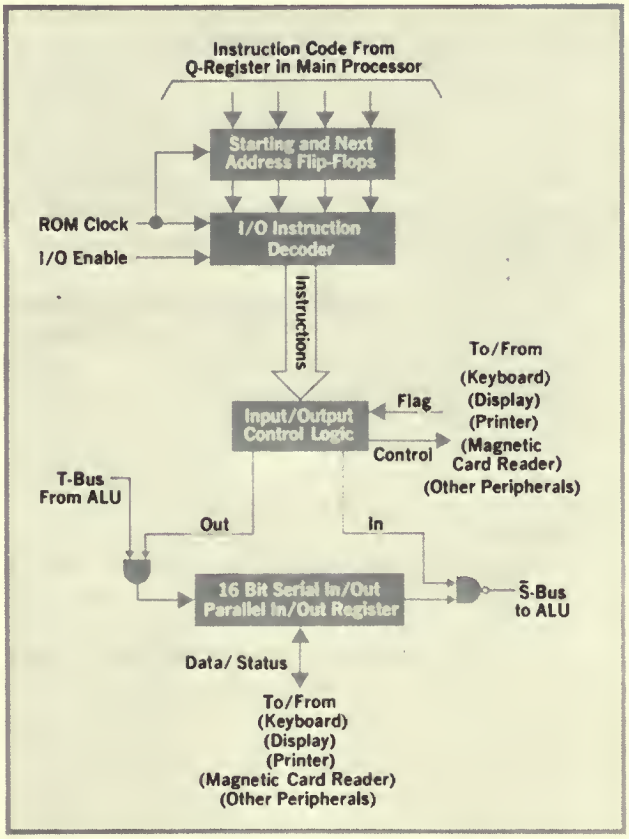


Fig. 14. Input/output processor is a self-contained microprocessor that implements the ten I/O instructions.

The I/O register's 16 parallel outputs provide the source for an output information bus structure which is common to all connecting peripherals. Parallel input information is received via an input information bus structure terminated by the twelve least-significant parallel inputs of the I/O register. Input information may be loaded into the I/O register by interrupt request or upon demand from the calculator.

All data communication between individual peripherals and the calculator makes use of a "handshaking" operation. Data is placed on the bus lines by the transmitter and then a signal indicating data ready is sent. The receiver acknowledges this and returns a signal noting that data has been accepted.

Associated with the I/O register are control circuits that implement this "handshaking" operation. The control circuitry consists of gates and flip-flops which are controlled by the I/O processor.

Internal Peripherals

A group of peripherals which may be contained within the calculator are called internal peripherals and are distinguished from a group called external peripherals by the fact that they are directly addressed as a part of the input/output instruction. This group of internal peripherals includes keyboard, display, magnetic-card storage, thermal printer, and I/O register.

Each internal peripheral has associated with it a driver contained in read-only memory in the basic calculator, plus supporting control hardware. The I/O register is included as an internal peripheral since it is directly addressable from the I/O instruction set and it functions as a holding and passing register for all peripherals. Fig. 15 shows the relationship between the internal peripherals and the I/O structure.

External Peripherals

External peripherals are connected to the calculator by an external signal cable. They are addressed indirectly from the I/O register. In general the driver for any external peripheral is contained in a plug-in ROM which may be unique to a certain peripheral (e.g., a typewriter) or may contain a general-purpose driver which communicates in bit-parallel, character-serial ASCII. Fig. 15 also shows the relationship between external peripherals and the I/O structure.

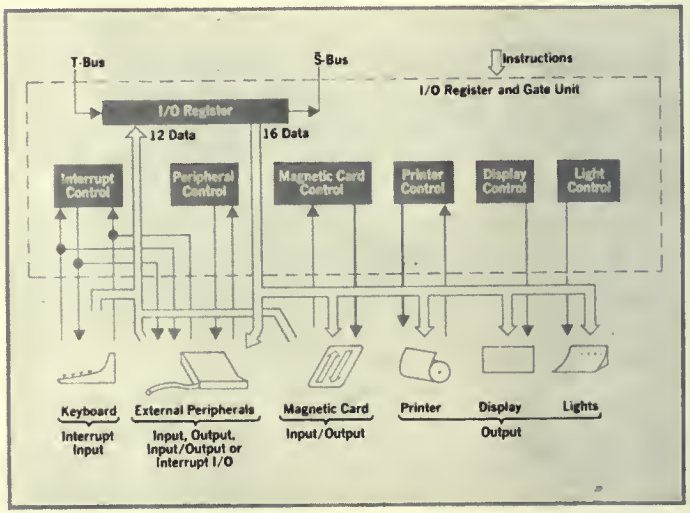


Fig. 15. I/O structure is designed to accommodate a variety of internal and external peripherals.

Peripheral Communication

All internal peripherals are addressed by the I/O instructions. Therefore, the receiving peripherals have access to the full 16-bit field of the I/O register. In addition each internal peripheral has its own control and flag logic by which "handshaking" takes place.

Communication with an external peripheral requires that a 16-bit word be formed in the processor. This word consists of a four-bit address in the four most significant bit positions, a four-bit status word in the four next-most significant bit positions, and eight data bits in the eight least significant bit positions. This 16-bit word is sent to the I/O register, where the parallel outputs of the I/O register place the word on the bus structure. After this has been accomplished a control signal is placed on the control line which, with the decoded four-bit address, causes the desired peripheral to take action. A receiving peripheral acknowledges the receipt of data by returning a flag signal. A transmitting peripheral places its data and status on the twelve input lines and sends a data-ready signal to the calculator.

The kinds of external peripherals are unlimited. The addressing scheme of 9800 Series Calculators provides for a maximum of 15 different addresses. Of these, addresses 10 through 15 are fixed and are reserved for unique drivers. Addresses 1 through 9 are variable and may be selected on a peripheral's interface card by means of jumper wires or switches. The bus structure makes the peripheral interfaces slot-independent, that is, they may be

System Information Series 9800 Programmable Calculators

Model 10		
Basic Model 10 Calculator, including 51 registers and 500 program steps	9810A	$2475
Factory Installed Options		
111 Total Data Registers	Opt. 001	$ 400
1012 Total Program Steps, or	Opt. 002	$ 500
2036 Total Program Steps	Opt. 003	$ 850
Printer	Opt. 004	$ 675
Carrying Handle	Opt. 015	$ 25
Field Installable Options		
111 Total Data Registers	11216A	$ 440*
1012 Total Program Steps, or	11217A	$ 540*
2036 Total Program Steps	11218A	$ 890*
Field Installed Printer	11219A	$ 715*
Plug-In Function Blocks		
Mathematics	11210A	$ 485
Printer Alpha	11211A	$ 485
Typewriter	11212A	$ 225
User Definable	11213A	$ 485
Statistics	11214A	$ 485
Plotter	11215A	$ 485
Plotter/Printer Alpha Comb.	11261A	$ 800
Peripheral/Cassette Comb.	11262A	$ 625
Peripheral	11264A	$ 485
Cassette Memory	11265A	$ 225
Peripheral/Printer Alpha Comb.	11266A	$ 800
Typewriter/Cassette Comb.	11267A	$ 450
Model 20		
Basic Model 20 Calculator, including 173 registers	9820A	$4975
Factory Installed Option		
429 Total Registers	Opt. 001	$1250
Carrying Handle	Opt. 015	$ 25
Field Installable Option		
429 Total Data Registers	11228A	$1490*
Plug-In Function Blocks		
Peripheral Control I	11220A	$ 485
Mathematics	11221A	$ 485
User Definable	11222A	$ 485
Cassette	11223A	$ 225
Peripheral Control II	11224A	$ 485
Model 30		
Basic Model 30 Calculator, including 3520 Bytes (1760 words) read/write memory	9830A	$5975
Factory Installed Option		
7616 Total Bytes (3808 words) read/write memory	Opt. 275	$1475
Plug-In Function Blocks		
Matrix Operations	11270B	$ 485
Plotter Control	11271B	$ 485
Extended I/O	11272B	$ 485
String Variables	11274B	$ 485
Terminal I	11277B	$ 485
9800 Series Peripherals		
Mark-Sense Card Reader	9860A	$ 850
Typewriter Output	9861A	$2250
Plotter	9862A	$2675
Paper Tape Reader	9863A	$1470
Digitizer	9864A	$5900
Tape Cassette	9865A	$1750
I/O Expander	9868A	$ 975
General I/O Interface	11202A	$ 200
BCD Interface	11203A	$ 300
Thermal Printer	9866A	$2975
Hopper-Fed Card Reader	9869A	†

Hewlett-Packard Manufacturing Division: Calculator Products Division.

*Plus field installation charges

†Not available.

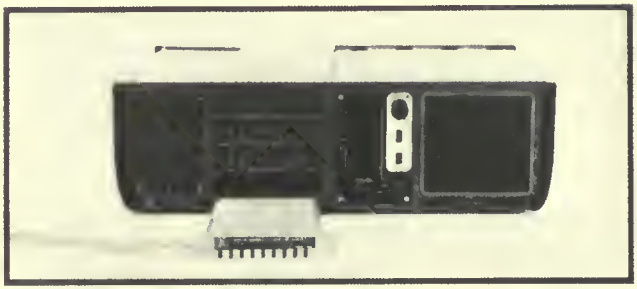


Fig. 16. Basic calculator has four I/0 slots for external peripherals. Expanders add nine more slots each.

connected to any calculator I/O slot (Fig. 16). The basic calculator has 4 slots for peripherals, and this can be expanded to as many as 40 by means of I/O expanders.

Reliability

The I/O system uses time-proven TTL circuits with known good reliability characteristics. Supplementing the hardware reliability, the handshaking operation assures reliable data transmission over cables up to 10 feet long. Although 10 feet is the maximum recommended length, longer cables have been used successfully.

APPENDIX 9800 MICROPROGRAMMED PROCESSOR

Instruction Set of the Central Processor

The instruction set, as alluded to before, is strongly related to the HP 2114 and HP 2116 minicomputer instruction set. There are 75 basic instructions. The memory reference, skip, alter, complement, and shift-rotate classes of instructions are essentially identical to the Binary Processor Chip (BPC) used in the HP 9845 (see Chap. 31). The full set of 28 ROM outputs with their associated microinstructions is shown in Table 1. The list of 16 qualifiers and assigned codes is contained in Table 2.

The 9810/20/30 Series processors did not have separate processors for I/O and extended arithmetic as the HP 9845 does. Thus instructions beyond those in the BPC were required to deal with I/O and decimal arithmetic. These additional instructions are outlined in Table 3.

Table 1 Microinstruction Set—9000 A

	Positive true	*Negative true outputs*				
Control field	*ROM output*	*Decoded μ-instruction output*				*Function*
General	1. IQN					Inhibit shift clock if qualifier not met.
	2. BRC					Branch: inhibits S00 if qualifier not met.
	3. $\overline{\text{TTT}}$					T BUS→ T REG
	4. $\overline{\text{TTM}}$					T BUS→ M REG
	5. $\overline{\text{XTR}}$					A/B REG→ R BUS
				$\overline{SC1}$	$\overline{SC0}$	
S-code	6. SC1	ZTS		0	0	ZERO→ S BUS
	7. SC0	MTS		0	1	M REG→ S BUS
		TTS		1	0	T REG→ S BUS
		UTS		1	1	ONE→ S BUS
			$\overline{RC2}$	$\overline{RC1}$	$\overline{RC0}$	
R-code	8. RC2	UTR	0	0	0	ONE→ R BUS
	9. RC1	PTR	0	0	1	P REG→ R BUS
	10. RC0	TRE	0	1	0	T REG→ E REG → R BUS
		WTM · ZTR	0	1	1	Store contents T REG → memory
		TQ6 · 2TR	1	0	0	T BUS→ Q REG (bit 6)
		QTR	1	0	1	Q REG→ R BUS
		RDM · ZTR	1	1	0	READ MEMORY <M> → T REG
		ZTR	1	1	1	ZERO→ R BUS

Table 1 (*Cont'd.*)

Control field	*Positive true ROM output*	*Negative true outputs: Decoded μ-instruction output*	*XC2*	*XC1*	*XC0*	*Function*
X-code	11. XC2	TTQ	0	0	0	T BUS→ Q REG
	12. XC1	QAB	0	0	1	Q REG (bit 11) → AB flip-flop
	13. XC0	BCD	0	1	0	BCD arithmetic mode of ALU
		TBE	0	1	1	T BUS→ E REG → R BUS
		CAB	1	0	0	Complement the AB flip-flop
		TTP	1	0	1	T BUS→ P REG
		TTX	1	1	0	T BUS→ A/B REG
		NOP	1	1	1	None of the above
		Decoded in ALU	*AC2*	*AC1*	*AC0*	
ALU	14. AC2	XOR	0	0	0	Exclusive OR ..... R⊕S → T BUS
	15. AC1	AND	0	0	1	Logical AND ..... R · S → T BUS
	16. AC0	IOR	0	1	0	Inclusive OR ..... R + S → T BUS
		ZTT	0	1	1	ZERO→ T BUS
		ZTT · CBC	1	0	0	ZERO→ T BUS, clear binary carry
		IOR · CBC	1	0	1	Inclusive OR, clear binary carry
		IOR · SBC	1	1	0	Inclusive OR, set binary carry
		ADD	1	1	1	Binary add
Clock	17. CC1 18. CC2 19. CC4 20. CC8					This 4-bit code initializes a preset-table down counter to generate any number of shift clocks from 1 through 16. Shift is inhibited by IQN if qualifier not met.
Qualifier	21. QC3 22. QC2 23. QC1 24. QC0					This 4-bit code performs *two* functions: (1) addressing the data selector to select one of sixteen qualifier inputs; (2) providing complement code to *primary* flip-flops.
Secondary address	25. S03 26. S02 27. S01 28. S00					This 4-bit code provides complement code to the sec*ondary* flip-flops. If BRC is given and qualifier is not met, the S00 bit is inhibited

Special microinstructions

TQR = UTR · XTR — transfers Q-Register to primary address as shown:
- Q14 · Q11→ P04
- Q12→ P05
- Q13→ P06
- Q14→ P07

IOS = PTR · XTR — (a) initiates transfer of control to I/O if Q10 = 1
(b) sets "single service" FF in I/O via $\overline{\text{SRA}}$ if Q10 = 0

Special operations

BCD sum → A<0:3> = BCD · $\overline{\text{UTR}}$ · ROM CLOCK
Clear decimal carry = QAB · ROM CLOCK
Set decimal carry = UTR · BCD · ROM CLOCK
Decimal add = BCD · ZTT . . . T<0:3> + A<0:3> → Q<0:3>

Table 2 Qualifier Set—9000A

Qualifier code					
QC3	*QC2*	*QC1*	*QC0*	*Mnemonic*	*Function*
0	0	0	0	Q00	Shift/skip one bit
0	0	0	1	Q01	Shift/skip two bits
0	0	1	0	Q02	Shift/skip four bits
0	0	1	1	Q03	Shift/skip eight bits
0	1	0	0	Q04	Fast square root qualifier
0	1	0	1	Q05	Set bit in A/S group; FDV qualifier
0	1	1	0	Q06	T-bus qualifier via TQ6
0	1	1	1	QBC	Binary carry from ALU
1	0	0	0	QP0	P Register bit 0, for BCD counting
1	0	0	1	Q15	Indirect address, clear bit in A/S group
1	0	1	0	QMR	Memory reference qualifier
1	0	1	1	Q10	Current page qualifier, FXA qualifier
1	1	0	0	QNR	Non-service request qualifier
1	1	0	1	Q08	FMP qualifier
1	1	1	0	QDC	Decimal carry from ALU
1	1	1	1	QRD*	ROM disable (normally zero)

POP will preset ROM address flip-flops at turn-on.

*QRD may be used with IQN to insure zero shift except when in I/O loop.

Table 3 I/O and Decimal Arithmetic Instructions of the HP9810/20/30 Central Processors

Execute-DMA group

These three special-purpose instructions were chosen to speed up printing and extended memory operations.

EXA/B Execute A. The contents of the A register are treated as the current instruction, and executed in the normal manner.

DMA Direct Memory Access. The DMA control in Extended Memory is enabled.

Input/output group (IOG)

The eleven IOG instructions, when given with a select code, are used for the purpose of checking flags, setting or clearing flag and control flip-flops, and transferring data between the A/B registers and the I/O register.

STF <SC> Set the flag flip-flop of the channel indicated by select code <SC>.

CLF <SC> Clear the flag flip-flop.

SFC <SC> Skip if flag clear.

SFS <SC> H/C Skip if flag set. H/C indicates if the flag flip-flop should be held or cleared after executing SFS.

CLC <SC> H/C Clear control. H/C indicates if the flag flip-flop should be held or cleared after executing CLC.

STC <SC> H/C Set Control. Set the control flip-flop in the channel indicated by <SC>. H/C indicates if the flag flip-flop should be held or cleared after executing STC.

OT* <SC> H/C Output A or B. Sixteen bits from the A/B register are output to the I/O register. H/C allows

Input/output group (IOG)

holding or clearing the flag flop after execution of OT*. The different select codes allow different functions to take place after loading the I/O register.

SC=00 Data from the A or B register is output eight bits at a time for each OT* instruction given. The A or B register is rotated right eight bits.

SC=01 The I/O register is loaded with 16 bits from the A/B registers.

SC=02 Data from the A/B register is output one bit at a time for each OT* instruction for the purpose of giving data to the Magnetic Card Reader. The I/O register is unchanged.

SC=04 The I/O register is loaded with 16 bits from the A/B register and the control flip flop for the printer is then set.

SC=08 The I/O register is loaded with 16 bits from the A/B register and the control flip flop for the display is then set.

SC=16 The I/O register is loaded with 16 bits from the A/B register and then data in the I/O register is transferred to the switch latches.

LI* <01> H/C Load into A or B. Load 16 bits of data into the A/B register from the I/O register. H/C allows

Table 3 (*Cont'd.*)

Input/output group (IOG)

	holding or clearing the flag flop after L1* has been executed.
LI* <00>	The least significant 8 bits of the I/O register are loaded into the most significant locations in the A or B register.
MI* <01> H/C	Merge into A or B. Merge 16 bits of data into the A/B register from the I/O register by "inclusive or." H/C allows holding or clearing the flag flop after MI* has been executed.
MI* <00>	The least significant 8 bits of the I/O register are combined by inclusive OR with the least significant 8 bits of the A or B register, and rotated to the most significant bit locations of the A or B register.

MAC instruction group

A total of 16 MAC instructions are available for operation (a) with the whole floating-point data (transfer, shifts, etc.) or (b) with two floating-point data words to speed up digit and word loops in arithmetic routines.

Note: A<0:3> means: contents of A-register bit 0 to 3.

AR 1 is a mnemonic for arithmetic pseudo-register located in R/W memory on addresses 1744 to 1747 (octal).

AR 2 is a mnemonic for arithmetic pseudo-register located in R/W memory on addresses 1754 to 1757 (octal).

D_i means: mantissas' i-th decimal digit; most significant digit is D1; least significant digit is D12; decimal point is located between D1 and D2.

Every operation with mantissa means BCD-coded decimal operation.

RET Return
16-bit number stored at highest occupied address in stack is transferred to the program counter. Stack pointer is decremented by one.

MOV Move overflow
The contents of E-register are transferred to A<0:3>. Rest of A-register and E-register filled by zeros.

CLR Clear a floating-point data register in R/W memory on location <A>
ZERO<A>, <A>+1, <A>+2, <A>+3

MAC instruction group

XFR Floating-point data transfer in R/W memory from location <A> to location <B>.

MRX AR1 mantissa is shifted to right n times. Exponent word remains unchanged.

MRY AR2 mantissa is shifted to right n times.

MLS AR2 mantissa is shifted to left once. Exponent word remains unchanged.

DRS AR1 mantissa is shifted to right once. Exponent word remains unchanged.

DLS AR1 mantissa is shifted to left once. Exponent word remains unchanged.

FXA Fixed-point addition
Mantissas in pseudo-registers AR2 and AR1 are added together and result is placed into AR2. Both exponent words remain unchanged. When overflow occurs "0001" is set into E-reg.; in opposite case<E> will be zero.

FMP Fast multiply
Mantissas in pseudo-registers AR2 and AR1 are added together B<0:3>-times and result is placed into AR2. Total decimal overflow is placed to A<0:3>. Both exponent words remain unchanged.

FDV Fast divide
Mantissas in pseudo-registers AR2 and AR1 are added together repeatedly until first decimal overflow occurs. Result is placed in AR2. Both exponent words remain unchanged. Each addition without overflow causes +1 increment of <B>.

CMX 10's complement of AR1 mantissa is placed back in AR1, and ZERO is set in E-register. Exponent word remains unchanged.

CMY 10's complement of AR2 mantissa.

MDI Mantissa decimal increment
Mantissa on location <A> is incremented by decimal ONE on D12 level, result is placed back in the same location, and zero is set in E-reg. Exponent word is unchanged.

NRM Normalization
Mantissa is pseudo-register AR2 is rotated to the left to get D1 ≠ 0. Number of these 4-bit left shifts is stored in B<0:3> binary form (B<4:15>=0). Exponent word remains unchanged.