The HP-65.

The SR-52: Another World's

To the casual observer, the HP-65 and the SR-52 look very similar; the big difference is in their logic systems.

J Bradley Flippin
5044 Park Rim Dr
San Diego CA 92117

On September 16 1975 Texas Instruments announced the latest entry in their series of sophisticated pocket calculators, the SR-52, exactly 20 months after Hewlett-Packard announced their HP-65 fully programmable unit. Richard Nelson described the HP-65 in the December 1975 issue of BYTE. The purpose of this article is to provide some additional information on the new SR-52 and to provide a comparison to the HP-65.

To the casual observer the two units look very similar. Both are of the hand held variety, packing a tremendous amount of logic and memory into a small package. The SR-52 weighs in at 12.3 ounces while the HP-65 weighs only 11 ounces. One of the big differences is the retail price. As of this writing, the SR-52 retails for $395 while the HP-65 retails for $795 (although it can sometimes be obtained for $695 if one looks hard enough).

Notation

Other than the differences in price and keyboard (which will be discussed later), the other big difference is in their logic systems. Hewlett-Packard uses Reverse Polish Notation (RPN) in their line of pocket calculators while Texas Instruments has stayed with the algebraic (or infix) notation used by the rest of the calculator industry. It is interesting to read the literature because each company sets forth a very convincing case for its own system. Texas Instruments put it this way in their SR-52 flyer:

> "And to make it more confusing, a good case can be made for both by the careful selection of sample problems. In truth there is no ultimate answer. Either system can be operated with ease by the experienced owner. And either can be a boon to the simple solution of the most complex problems. Many practiced users of RPN now swear by it. But owners of algebraic machines can find RPN awkward and confusing. It boils down to individual preference."

Hewlett-Packard has an excellent paper comparing the two systems from their point of view; it's entitled "ENTER vs EQUALS" (Publication 5952-6035). There is no answer, as Texas Instruments has pointed out, because both systems get the job done. As a result, this is sure to be one of those topics that will keep their respective advocates trying to "convince" each other for years to come.

What are the main differences between the two logic systems? For the benefit of the reader, it might be well to spend a few moments describing the two systems (some say there are actually three systems because the algebraic system has two variations).

Algebraic (Infix) Notation

The simplest system for the novice is the algebraic system because it is the notation the average person has been taught in school. It is also known as infix notation. It is based on solving a problem in exactly the way it is

Using the parenthesis form of algebraic notation, the calculator's hardwired software analyzes the problem as stated in a "natural" form.

Infix notation has operators written *in between* two operands.

Postfix notation finds an operator following notation of two operands.

Smallest

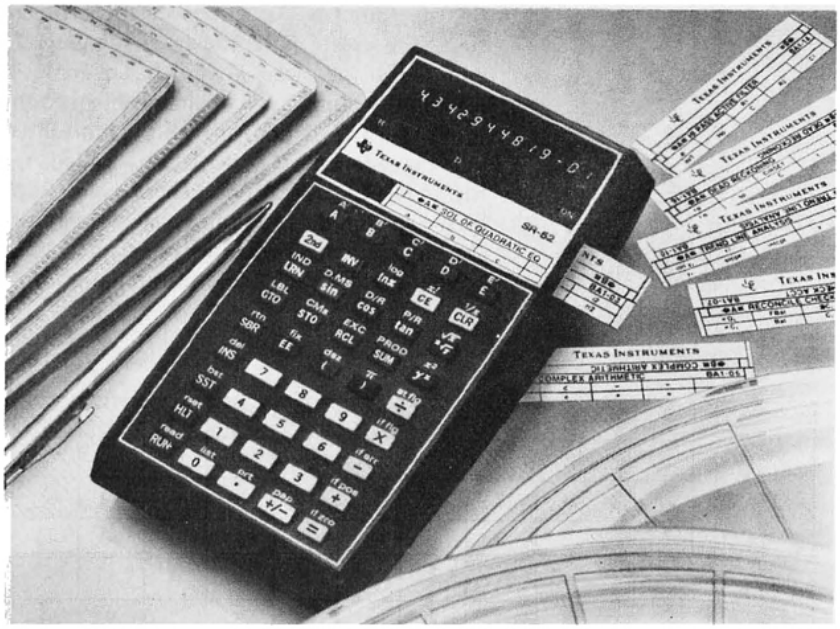

Photo 1: The SR-52 Calculator. Like its cousins in the microcomputer and large computer world, the SR-52 has magnetic recording features allowing the user to purchase and build a library of software.

commonly written. For example, let's solve the problem 5 + 4 = ?. The operator keys the data in exactly the way it is shown. He pushes "five", "plus", "four" and then hits the "equal" key and the answer appears in the display register. In reality the calculator's program maintains two internal registers, one of which is displayed. Pressing the first (five) key enters the data into the display register. Pressing the second (plus) key transfers the data to the second register which is also known as the accumulator. At this point the number "five" is in both registers and an internal switch has been set telling the logic that the next number is to be *added* to the accumulator when the time comes for the next arithmetic operation. The second number (four) is now entered into the display register. At this point the accumulator contains a "five"; the display register contains a "four" and the "plus" logic is set. To get the answer the operator simply presses the "equal" key which tells the logic circuits to perform the pending operation on the data in the accumulator, place the results in the display register and clear the preset function logic. Chain manipulations are possible by simply pressing another function key. Thus, the process can go on indefinitely. For example: 5 + 4 = 9 x 2 = 18 – 8 = 10 (etc.).

The operator can accomplish the same task more easily by eliminating the "equal" key each time. After "five", "plus", and "four", the operator could press the "plus" key directly. The same sequence described above will take place, except that the "plus" logic will again be set (actually, any function could be pressed). Thus, chain manipulations are possible in this way, also. For example: 5 + 4 x 2 – 8 = will provide the same results (10), but the number of steps needed is cut by two.

This is simple algebraic logic in that it does not use parenthesis to determine the precedence order from the entered data. A modification to the algebraic notation can be made by adding parentheses to help define the problem. For example: The problem 5 x 3 + 4 using simple chain manipulations will result in an answer of 19. However, if the operator wants to perform the summation first, then he would have to set it apart by parenthesis in this manner: 5 x (3 + 4). Now the answer is 35. Note also that the same result could have been obtained by rearranging the problem to read: 3 + 4 x 5. Now simple algebraic chain manipulation will result in an answer of 35 because the summation is performed first.

Parenthesis processing requires additional internal registers to hold the intermediate results. However, the process is similar to that described for the simple algebraic system. The only difference is that when the operator keys in the parenthesis, it tells the logic to "hold off" on the chain manipulation process until the matching parenthesis is found.

One might wonder why parentheses are needed because simple inspection of the problem will tell the operator that he must

A key to economical storage of programs in these small programmable machines is use of merged operations – two keystrokes which are stored as one location in memory.

These programmable calculators must surely be the ultimate in compactness and ingenuity in packaging.

do the "internal" portions of the problem first. The use of parentheses to some extent eliminates the need for such an analysis, leaving the breakdown of the problem to the internal logic. It does this through an internal precedence of calculations called the hierarchy of operations. There is no hierarchy in simple algebraic systems, because the logic simply processes the data as they are entered through chain manipulation.

Stack Content	T								
	Z					12	12		
	Y		3	3		12	5	5	12	
	X	3	3	4	12	5	5	6	30	42
Key		3	↑	4	x	5	↑	6	x	+
Step #		1	2	3	4	5	6	7	8	9

Figure 1: An example of the HP-65 operations stack in use. This chart shows numerical contents of the stack elements X, Y, Z, and T while calculating the problem (3 x 4) + (5 x 6). The HP-65 uses Reverse Polish Notation to calculate the result, using the keystrokes shown.

Table 1: Detailed Comparisons. This table shows specific comparisons between the HP-65 and the SR-52 in areas of programming capability, calculating capability and operating characteristics.

Programming Capability	SR-52	HP-65
Program steps	224	100
Merged prefixes	all merged	stack and comparison
Merged store and recall instruction codes	no	yes
Program read/write	yes	yes
User-defined function keys	10	5
Possible labels	72	15
Absolute addressing	yes	no
Subroutine capability	yes	yes
Subroutine levels	2	1
Program flags	5	2
Unconditional branching	yes	yes
Conditional branching decisions	10	7
Indirect branching	yes	no
Editing		
Single-step	yes	yes
Back-step	yes	no
Insert	yes	yes
Delete	yes	yes
Single-step program execution	yes	yes
Optional lock-in printer	yes	no

A question one always asks when parentheses are encountered is how deeply may they be nested? (Remember, each parenthesized level in the problem requires additional internal storage for intermediate results). Some systems go four or five levels deep. The SR-52 is capable of nesting to nine levels. The following example illustrates the maximum capability of the SR-52 in this area:

6 x (9 + (6 x (12 ÷ (3 x (8 x (2 x (6 ÷ (6 x (6 + 2)))))))))).

Straight chain manipulation, disregarding parentheses, will yield a result of 3458, which is incorrect. The correct answer is 4.5. It should be noted that the parenthesis count does not include expressions that have already been terminated. For example: 6 x ((5 + 7) ÷ (6 x 9)) is not three deep, but only two deep because the first expression (5 + 7) was terminated by the first right parenthesis, ")", prior to encountering the second expression (6 x 9). Internally, each level of parentheses is like using one level of the stack in a Reverse Polish Notation machine.

Reverse Polish Notation

The second logic system is known as Reverse Polish Notation (RPN). The Polish mathematician Jan Lukasiewicz wrote a book published in 1951 on formal logic wherein he was the first to demonstrate that an arbitrary expression could be shown unambiguously without the use of parentheses by placing the operators immediately in front of or after their operands. For example, (a + b) x (c – d) could also be expressed as x + ab – cd (keep in mind that ab is a logical notation and does not mean multiplication but shows only the sequence of the data). This is a prefix type of notation. It could also be reversed to provide a postfix type of notation as follows: ab + cd – x. (Now you know why the algebraic system in common use today is called infix notation, since the operators are in the middle between operands). As a result of this discovery, both prefix and postfix notation have become widely known, respectively, as Polish and Reverse Polish Notation in Lukasiewicz's honor.

Reverse Polish Notation, as mentioned above, does not utilize parentheses. As a result, the solution to problems using this type of logic must be approached in exactly the same manner as with any computer program because it is the same logic used in all large (and nowadays, small) computer systems. The programmer moves his data around into various registers and then, once

it is where he wants it, he executes the required arithmetic operation.

Hewlett-Packard has arranged its working registers into what they call an operational stack. It consists of four registers designated X, Y, Z and T. A fifth register is called the LAST X register and is a recent addition to the HP line, although it is not directly a part of the stack itself. The LAST X register holds the last data entry in the event the operator wants to either see what it was, wants to use it again, or wants to extract it from the solution (i.e., entered the correct data but pushed the wrong function. The operator simply presses LAST X, the inverse of the previous function and then the correct function).

The four operational stack registers have special functions and operate as an integrated group. The X register is the data entry register and is the only one that is displayed (For those who have used the HP-9100/9810 series, they displayed the X, Y and Z registers simultaneously). All of the trigonometric and some of the transcendental functions are performed directly in the X register (i.e., 1n x). The Y register can be thought of as the accumulator. All mathematical operations are performed in this register. Those transcendental functions which require two registers use both the X and Y registers (i.e., y^x). The Z and T registers are temporary storage registers and no mathematical operations can occur in them. Data are moved to and from them as required during the solution.

If one uses a concept such as an operational stack, it sometimes proves necessary to move the data around within the stack. To perform this function, Hewlett-Packard has designated a special set of data movement keys which do not appear (nor are they required) on the SR-52. For example, ROLL UP and ROLL DN allow the contents of the stack to be shifted up or down one position (in much the same manner as a circular shift or rotate instruction). When shifted up, the data in the T register goes into the X register and vice versa. The ENTER (or UP) key literally pushes the data up. Thus, the contents of all registers are moved up one, with two exceptions: The original data stays in the X register and the data in the T register is destroyed by the data from the Z register. As a result, if the operator desires, the same number can be placed in all registers by pressing the sequence: (data)(UP)(UP)(UP).* Figure 1 is an example of the use of an operational stack for the problem (3 x 4) + (5 x 6). The ENTER (↑) key breaks up the solution by moving the intermediate data up into the stack where it is saved until needed. In addition, the HP-65 has a special feature of automatically inserting an UP function prior to any data entry that follows a functional operation. This can be seen between steps four and five in figure 1. Notice that the intermediate result (12) moved up automatically as the five was entered. The operator must, however, ensure that he does not move up more than three intermediate results, which is the HP-65's limit (without using the data storage registers). In comparison, the SR-52 can handle up to ten pending operations.

As a further assist in manipulating data in

Table 1 (continued):

Calculating Capability	SR-52	HP-65
log, ln x	yes	yes
10^x, e^x	yes	yes
x^2	yes	yes
$\sqrt{x}$	yes	yes
$\sqrt[x]{y}$	yes	no
y^x	yes	yes
1/x	yes	yes
x! (factorial)	yes	yes
Trigonometric functions	yes	yes
Degrees-minutes-seconds to decimal degrees conversion	yes	yes
Degree, minute, second arithmetic (+, −)	no	yes
Degree/radian conversion key	yes	no
Polar/rectangular conversion	yes	yes
Octal conversion	no	yes
Absolute value	no	yes
Integer, fraction part	no	yes
Built-in π value precision	12 digits	10 digits

Operating Characteristics		
Angular modes	2	3
Fixed-decimal option	yes	yes
Calculating digits	12	10
Digits displayed (mantissa + exponent)	10+2	10+2
Data memories	20	9
Memory arithmetic (+, −, x, ÷)	yes	yes
Exchange x with y	no	yes
Exchange x with data memory	yes	no
Entry mode	algebraic	RPN
Max. number of pending operations handled	10	3
Number of keys	45	35
Indirect memory addressing	yes	no

***NOTE: For examples of keystroke sequences, the key name or a description of input (such as "UP") is enclosed in parentheses.**

And Now, a Printer for the SR-52

On January 7 1976 Texas Instruments Inc, Calculator Products Division, announced the new PC-100 print cradle for the SR-52 calculator. The product is a desk top unit with a 20 character per line thermal printer using 2.5 inch (6.35 cm) thermal printing paper available in roll form. The PC-100 interfaces to the calculator and expands capabilities to include program listing and execution trace capabilities. The listing allows a permanent human readable record of the program to be made automatically; a trace documents each calculation step of a program as it is performed. An extra feature is that the calculator can be locked into the base provided by the printer, making the entire system less likely to be pocketed by unscrupulous individuals. All this function is available for only $295, and the unit will be sold through the usual TI calculator distribution channels (direct mail and retail stores).

For further information on the PC-100, contact Texas Instruments Inc PO Box 5012, Mail Station 84, Dallas TX 75222 (Attn: PC-100).

the stack, the HP-65 has an EXCHANGE X AND Y key ($x \leftrightarrows y$) which allows the operator to interchange the contents of these two important registers. This is handy when encountering operations where ordering of operands is vital, as in division. Using this operation, it is also possible to reverse the sequence of the data in the entire stack: ($x \leftrightarrows y$), (ROLL UP), (ROLL UP), and ($x \leftrightarrows y$); (ROLL DN) could have been used, if desired.

Function Selection

The next area of interest is the keyboard itself. The SR-52 has 45 keys where the HP-65 has only 35 keys. Both machines use the "second function" type of system which allows one key to have two meanings. The SR-52 uses the symbol "2nd" while HP uses "f". In addition, the HP-65 also has a "third function" key designated "g" which allows all of their keys to take on a third meaning. Both units have an inverse function key designated INV on the SR-52 and f^{-1} on the HP-65. This is handy for finding the ARC SIN of a number. The operator simply presses: (2nd) (SIN). A comparison of the photographs with this article will reveal other differences in the keyboard layouts.

Operation Codes

The key to economical use of storage in this type of machine is the application of merged operations which allow one storage location to accept a prefix, when required, along with an associated instruction code. This is possible in both units; however, only the HP-65 allows merged storage and recall instructions. A practical relationship between the two programming systems can be obtained by using the problem on page 74 in the *HP-65 Owner's Handbook* as a comparison. It is a financial interest problem containing 22 storage and recall instructions. The problem requires 68 memory locations in the HP-65, while in the SR-52 (neglecting any translation due to logic systems, which is left to the reader as an exercise), it would require 90 memory locations. The difference is due to the SR-52's lack of merged storage or recall instructions. This may be one of the reasons Texas Instruments made their program memory over twice as big as the Hewlett-Packard's.

Both units use a similar method of designating their decimal numeric instruction (or operation) codes. With the exception of the digit keys, the instruction codes for any key on either unit can be found by simply counting down the left column to that row and then counting across to the particular key. Thus, on the SR-52, the

instruction code for Enter Exponent (EE) is 52 while the same function on the HP-65 (EEX) is 43 (this can be verified from the two photographs). The digit keys retain their own values (i.e., one is 01, two is 02, etc.).

Comparisons?

There are so many features of both units that it is difficult to say any one of them is the *big* feature; however, the fact that they can both record and read programs on small magnetic cards certainly ranks high on the list. A detailed features comparison is found in table 1. Because of the SR-52's large memory, it requires two passes to read or write the card. The card is inserted in the A direction first and then turned around (not over, as the oxide must remain face down) and the B side is entered. Both units contain a recessed card holder between the display and the five special function keys which are labeled A through E. The cards have an area upon which the operator can write to designate the functions of the special keys for customized programs. In the case of pre-recorded programs, the data elements are also pre-printed on the cards as can be seen in the photograph.

Last, but not least, is the one big feature of the SR-52 which is not yet available with the HP-65 system. In early 1976 Texas Instruments intends to market the PC-100 which is an optional desk top lock in printer for use with the SR-52. It looks like a regular desk calculator with the typical adding machine type tape printing unit on the left and a space on the right for the SR-52. The unit includes a key lock so the calculator cannot be "lost." It will allow the user to list out entire programs, print the results of calculations, and advance the paper. These functions are already on the SR-52's keyboard as second functions (LIST, PRT, and PAP, respectively).

This short article has not covered all points of comparison between the two calculators. As can be seen from table 1, there are many areas that have not been discussed. The purpose has been to inform you about these two interesting computer systems. However, if you feel teased and want to investigate these fascinating machines further, the manufacturers would love to tell you where you can see them in your community. Both have toll free (WATS) numbers you can use: Texas Instruments (800) 527-4980 [in Texas (800) 492-4298], Hewlett-Packard (800) 538-7922, ext 1000 [in California (800) 662-9862]. These programmable calculators must surely be the ultimate in compactness and ingenuity in packaging.■