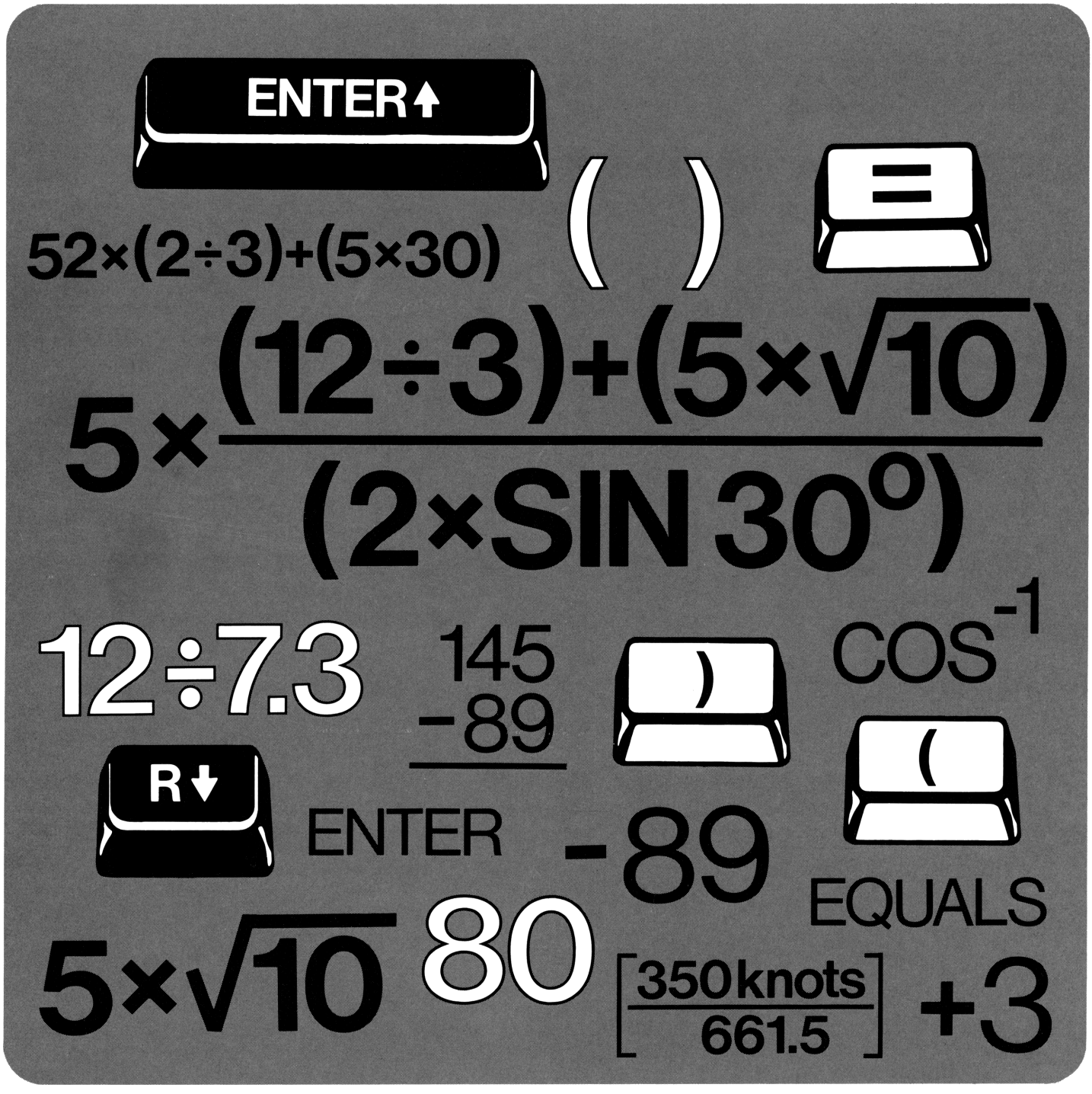
HEWLETT hp PACKARD

Advanced Calculator Logic
HP RPN/Algebraic
A Comparative Analysis



Advanced Calculator Logic
HP RPN/Algebraic
A Comparative Analysis

Introduction

This booklet discusses in depth the logic systems available on advanced hand-held calculators in use today. Logic systems have a great effect on the overall effectiveness of calculators: the logic system of a particular calculator determines directly how quickly, accurately, and simply the user can obtain answers.

The RPN system as implemented on Hewlett Packard calculators (hereafter referred to as HP RPN) is discussed first, followed by the various algebraic systems as offered on most competing brands. HP RPN is then contrasted with the most sophisticated of the algebraic systems, and benefits of each are demonstrated through the use of example problems.

No attempt is made to evaluate each system's effectiveness in programming. Rather the discussion is limited to the basic system feaures as encountered in manual keystroke solutions. However, the benefits of a particular system in manual solutions (e.g. efficiency, ease-of-use) generally apply to programming effectiveness as well.

This booklet will be of particular interest to educators who are involved with teaching calculator usage or assisting students in gaining the most from their calculators. It will also be of interest to users in any field who are either evaluating alternatives for purchase or striving to increase their effectiveness in utilizing their existing machines through greater understanding of the logic system.

Hewlett-Packard RPN

Primary Components

The basic elements of HP RPN are:

- A number entry convention which provides for a consistent approach to all types of operations.
- Four special memories logically arranged in a "stack" which operates automatically to manage the storage and recall of numbers involved in calculations.
- A few special features such as stack manipulation operations and an additional register to provide flexibility to the user to manipulate and review the numbers in his calculations and to recover from errors he may have made.

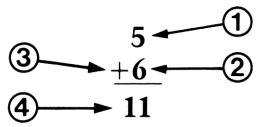
Number Entry Convention

At the heart of the HP RPN system, the number entry convention follows the rules of "Reverse Polish Notation" (RPN) as developed in 1949 by Polish mathematician Jan Lukasiewicz. RPN allows the placement of the operator immediately after the operand(s), eliminating all ambiguity regarding the order of execution in compound expressions. Thus, algebraic hierarchy rules and inserted parentheses are not needed to clarify the order of execution.

Consider how the following expression is evaluated by hand:

5 + 6

The first number is written and then the second number. Finally, the add operation is performed, and the answer is obtained:



In effect, the following sequence was followed, reading from left to right:

5 6 +

This, in essence, is the RPN convention. The argument (in this case two arguments) precedes the operator — this is called "postfix" notation.

In an HP RPN calculator, the key sequence follows directly:

■ Key in the first argument	5
■ Press the ENTER↑ key to separate the first argument from the second	ENTER↑
■ Key in the second argument	6
■ Press the + key	+

The result, 11, immediately appears in the display when the + key is pressed. The only new element in this key sequence is the ENTER used to separate the two arguments. The ENTER key is used only for separating the first from the second number in any operation requiring keyboard entry of two numbers in sequence. That is the only function of the ENTER key.

As problems become more complex, the RPN convention works in the same way:

$$16 + 30 - 11 + 17 - 14$$

Press	Display	Comments
16 ENTER↑	16.00	
30 +	46.00	(16 + 30)
11 −	35.00	(16 + 30 − 11)
17 +	52.00	(16 + 30 − 11 + 17)
14 −	38.00	(16 + 30 − 11 + 17 − 14)

ENTER is pressed only when putting two numbers into the calculator in sequence; therefore, it was only needed to separate 16 from 30. From there on, each number is simply keyed in, followed by its operator; and each intermediate answer immediately appears in the display. These intermediate answers or subtotals allow the user to "follow" the calculation procedures, providing feedback on each operation as it is keyed in.

Another characteristic of the RPN convention is its consistency for all problem types. The operation **always** follows the argument (or arguments) to be operated on. Previously, it was shown how this works with simple two-number operations, but the order is the same for single-number operations. Functions such as $\sqrt{\ }$, sin, cos, tan, ln and log all operate on a single number, or argument. For these functions, the basic keystroke sequence is the same as before. For example, to find the square root of 5:

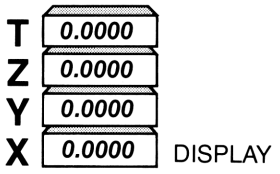
Press	Display
5	5
$\sqrt{x}$	2.2361

No ENTER was necessary because only one number was operated on. As a point of contrast, it is important to note that algebraic systems operate in exactly this way for single-number operations, but for two-number operations the function is placed between the numbers and must be followed by an = key. The number entry methods of most algebraic machines are inconsistent in that they use a combination of algebraic (two-number functions) and RPN (single-number functions).

The Automatic Memory Stack

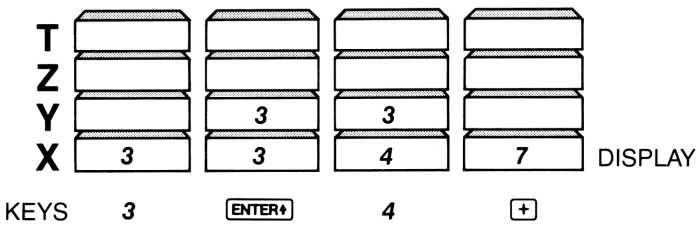
The HP RPN logic system combines an "automatic memory stack" with the RPN number entry convention just discussed. The stack provides an automatic mechanism for temporarily storing arguments and intermediate results and making them accessible at the appropriate point in a calculation sequence. In essence, the stack is responsible for the "housekeeping" or management of arguments awaiting their operations.

The stack consists of four storage registers, logically arranged to provide a last-in, first-out storage and retrieval of arguments in a calculation. This can best be illustrated if the registers are pictured one above the other:

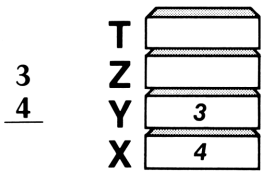


In the illustration there are the four registers, each one with a letter name that is used to refer to that particular register. Note also that the X register is the display register; that is, the number you see in the calculator's display is **always** the number in the X register.

To illustrate, a simple problem will be worked: 3 + 4 = 7. In the following diagram, the problem progresses from left to right with the contents of the stack registers shown as they would be immediately after each key is pressed.



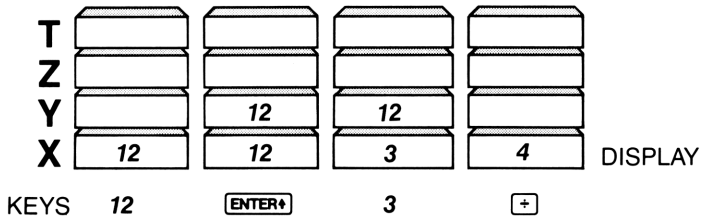
Notice that just before the + key is pressed, the two numbers are logically arranged in the stack the same way as they would be if the problem were being worked by hand:



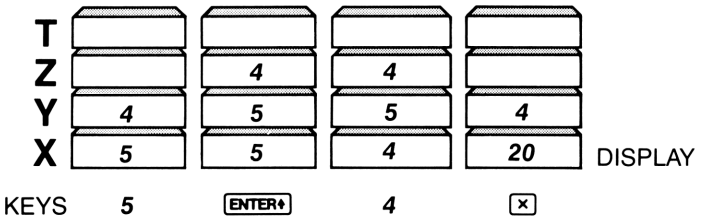
The effect of the ENTER key was to copy the number 3 from the X register into the Y register, leaving the X register ready to receive the second number. When the + key is pressed, the numbers in the X and Y registers are added together, and the result appears in the X register. Consider a more complex problem:

$$5 \times \frac{(12 \div 3) + (5 \times 4)}{(2 \times 3)}$$

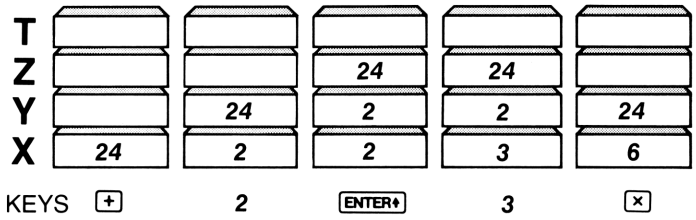
The problem is attacked as it would be by hand, calculating the expressions in parentheses first:



This is the first intermediate result, $(12 \div 3) = 4$. This intermediate result is automatically kept by the stack while the next intermediate result (5×4), is calculated.



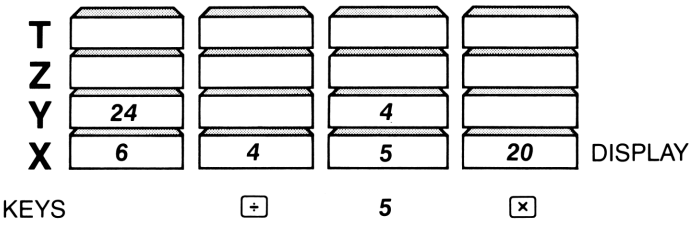
This is the second intermediate result, $(5 \times 4) = 20$. Above it, in the Y register, is the previous intermediate result, $(12 \div 3) = 4$. Both intermediate results are just in the correct positions, the X and Y registers, to be added together to get $(12 \div 3) + (5 \times 4)$. Continuing with the problem, this addition is performed followed by the calculation of $(2 \times 3) = 6$ in the denominator.



Again, the two intermediate results $(12 \div 3) + (5 \times 4) = 24$ and $(2 \times 3) = 6$ are positioned in the X and Y registers for the next step in the calculation:

$$\frac{(12 \div 3) + (5 \times 4)}{(2 \times 3)} = \frac{24}{6}$$

The quotient is now calculated by pressing the ÷ key, and the final multiplication performed to get the answer:



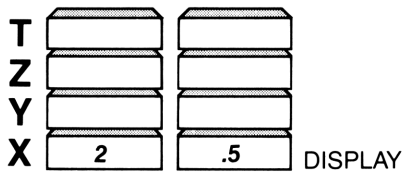
Notice that throughout the entire calculation, no parentheses were entered into the calculator. Parentheses are necessary in the written form of many expressions in order to remove ambiguity and ensure that the operations are performed in the proper sequence. HP RPN provides for this sequencing by following the order the problem is attacked and automatically storing the intermediate results. The stack allows the user to attack the problem in an order consistent with the structure of the written expression. This eliminates the need to enter and keep track of parentheses while working the problem.

Notice in the previous example that the T register was never used. Despite what looked like a fairly complex problem, only the X, Y, and Z registers were needed. Because of the problem solving approach provided in this system, the number of intermediate results rarely exceeds three levels of the four-level stack. The very complex problem worked later in this section illustrates this point.

The problems worked to this point have all dealt with two-argument operations. Single-argument operations such as x^2, $1/x$, $\sqrt{\ }$, and log and trig functions

work only on the contents of the X register and the answers return to the X register without affecting the other stack registers.

For example, with 2 in the X register the 1/x operation is performed as follows:

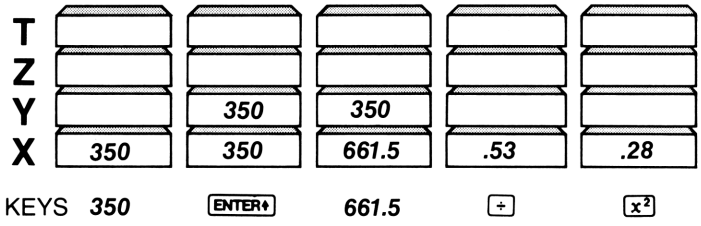


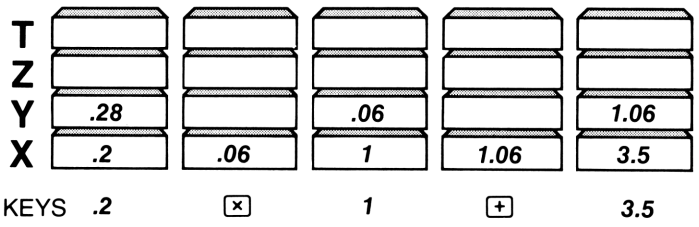
To illustrate further the operation of the stack in the HP RPN system, consider the following problem which calculates the Mach number M for an object traveling at velocity v, in knots, at altitude h, in feet:

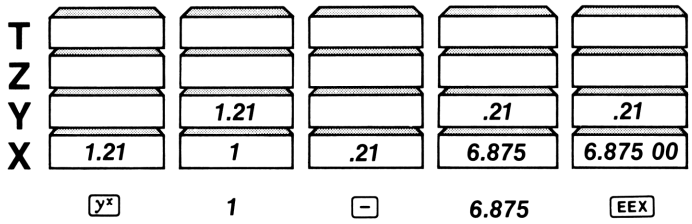
$$M = \sqrt{5\left[\left(\left\{\left[\left(1+0.2\left[\frac{v}{661.5}\right]^{2}\right)^{3.5}-1\right]\left[1-(6.875\times10^{-6})\,h\right]^{-5.2656}\right\}+1\right)^{0.286}-1\right]}$$

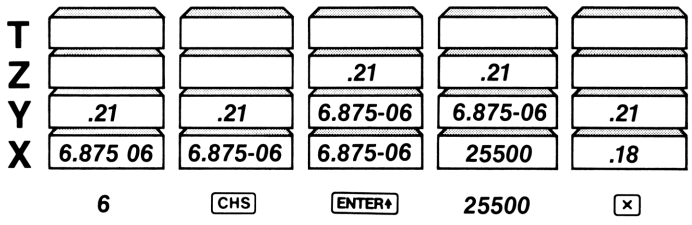
The problem will be worked for an object traveling at 350 knots at an altitude of 25,500 feet. The problem is attacked from the innermost set of parentheses, thus beginning with:

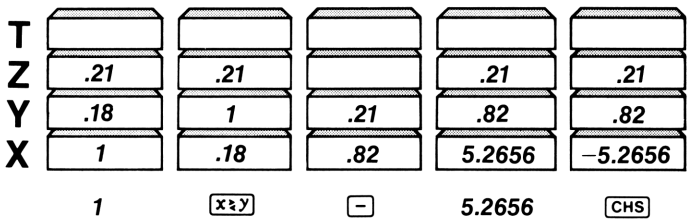
$$\left[\frac{350 \text{ knots}}{661.5}\right]^{2}$$

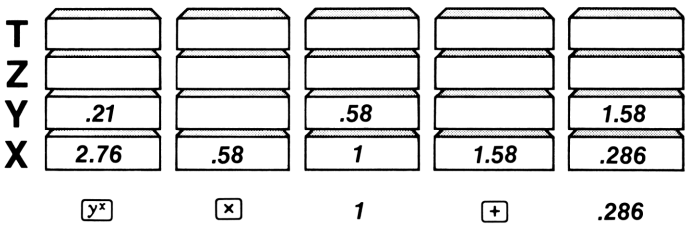


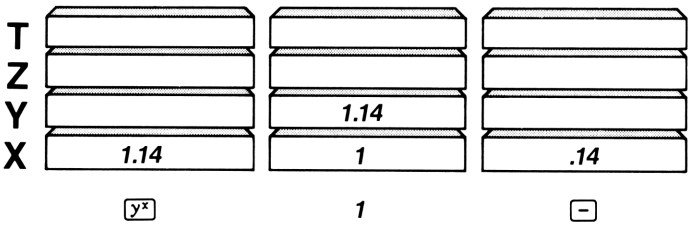


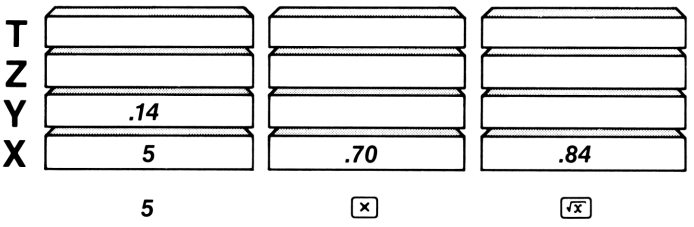












The HP RPN system enabled this calculation to be attacked and worked in the same order as if it were worked longhand with pencil and paper.

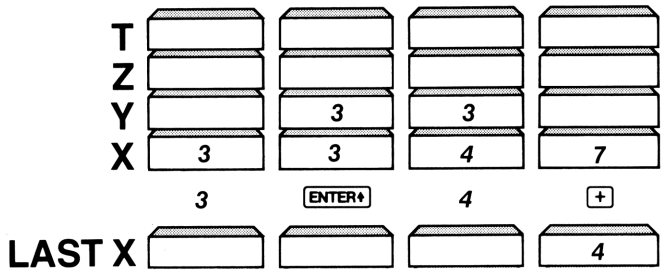
As each function key was pressed, the result of that operation was immediately displayed in the X register, providing feedback of the intermediate results.

Other Features of HP RPN

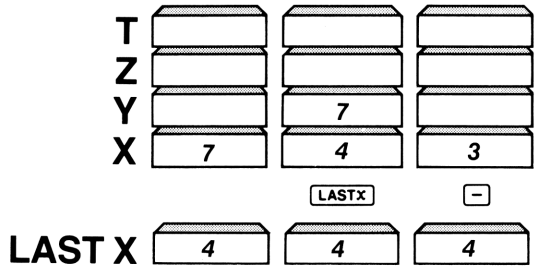
There are a few additional features of the HP RPN system which contribute to its utility in solving problems. These features deal with the ability to manipulate data in the stack registers. Since operations are not stored and held pending in this system, this ability can be valuable in rearranging or duplicating arguments prior to actual execution of the next operation to be performed. This may be especially helpful in error recovery.

LAST X Register

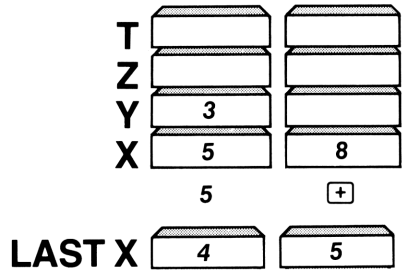
In the following diagrams, notice the addition of a fifth register, called LAST X. Essentially, every time a user presses a function key that puts a new number in the X register, the previous number goes into the LAST X register. To illustrate, consider the problem 3+4=7:



When + was pressed adding 3 and 4 to produce 7, the 4 from the X register was put into the LAST X register. This argument is now available for recall to be used in reversing an incorrect entry or tracing back one step to verify the last operation performed. To illustrate how the feature could be used in error recovery, assume that the problem above should have been 3+5=8, rather than 3+4=7. The LAST X register is used to recall the incorrect argument, 4, and the operation is then reversed by subtracting the 4. The correct argument, 5, is then keyed in and added:



The add operation has now been reversed.

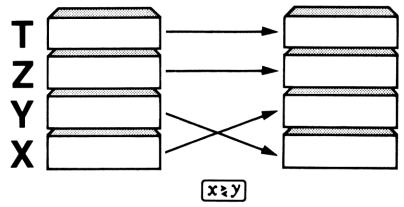


In a similar manner, if the wrong **function** key was pressed, a user can recall LAST X, reverse the function, and then proceed with the correct operation.

Stack Manipulation Keys

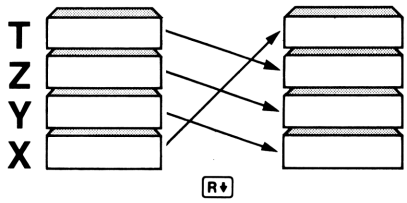
Besides the ENTER key, two other keys are provided which offer a means to rearrange arguments in the stack. These are the X-exchange-Y key and the roll-down key.

X ⇄ Y simply exchanges the contents of these two registers without affecting the other stack contents:



This capability is useful in divide and subtract operations, where the order of the X and Y contents is meaningful.

The R ↓ key "rotates" the contents of all four stack registers as shown:



This feature provides for review of all current stack register contents. When used with X ⇄ Y, it provides a means for rearranging the four arguments in any fashion.

The usefulness of these functions is most apparent in complex operations where the ability to review arguments, reverse operations, and rearrange the stack could save the user from having to restart from scratch.

Algebraic Systems

Primary Approaches

Algebraic systems available today vary widely in sophistication and implementation. However, all are meant to provide a simple means of entering problems by allowing entry of the arguments and functions in the same order as they appear in written form. This is, in fact, the case with very simple problems involving two arguments and a single function. When problems become more complex, some means of determining the order of operations must be established. How this is accomplished varies: following directly the order of entry; obeying the order rules of a built-in function hierarchy; keyboard entry of parentheses to remove ambiguity;...

Several implementations follow, using as an example the problem $5 \times 6 + 2 \times 5$.

Simple Algebraic

The problem is calculated in the order it is entered without regard to hierarchy.

$$5 \times 6 + 2 \times 5 = 160$$

Algebraic with Hierarchy

A hierarchy is built into the logic; typically, multiply and divide must be executed prior to add and subtract. All other functions (such as trig, logs, y^x, %, $\sqrt{\ }$, etc.) must also have defined places resulting in several more levels within the hierarchy.

$$5 \times 6 + 2 \times 5 = 40$$

Algebraic with Parentheses

Parentheses may be entered to define the order of execution. Within parentheses, no hierarchy exists.

$$5 \times 6 + 2 \times 5 = 160$$
$$5 \times (6 + 2) \times 5 = 200$$
$$5 \times (6 + 2 \times 5) = 200$$

Algebraic with Hierarchy and Parentheses

Parentheses may be entered, and a hierarchy is built in.

$$5 \times 6 + 2 \times 5 = 40$$
$$5 \times (6 + 2) \times 5 = 200$$
$$5 \times (6 + 2 \times 5) = 80$$

Number Entry Conventions

In an idealized algebraic system, a problem could be entered exactly in its written form. Thus, the following somewhat complex problem could be entered without change:

$$\frac{(5 + \sqrt{6})^2 + \sin 40^\circ}{3 + \cos 35^\circ}$$

Some computer languages such as BASIC come closest to this idealized goal by allowing the entire expression to be entered once it has been rewritten on one line with parentheses inserted to remove ambiguity:

$$[(5 + \sqrt{6})^2 + \sin 40^\circ] / (3 + \cos 35^\circ)$$

In this case, two new sets of parentheses were required to ensure the integrity of the original expression. Typically, the entire expression would be displayed on the computer's output device so that there would be a visual feedback to aid the user in checking proper placement of parentheses, etc. The entire expression is then interpreted by the computer prior to any function execution, and the ordering of operations is made to follow the written expression.

However, this level of sophistication is not available on hand-held calculators today. These algebraic systems are limited in several ways. First, the calculator displays do not display the non-numeric characters such as $\sqrt{\ }$ and ( ). They display instead just one number at a time, showing none of the operational relationships of the expression. Secondly, single-argument operations such as $\sqrt{\ }$ sin, and ln, are entered in the reverse order from the written expression.

And finally, functions are executed at the moment that ambiguity is removed, i.e., at the first instant that further entries will have no effect on the operations currently held pending. This last feature has the effect that at any given instant some operations will have been executed while others are being stored pending further entries.

The following section compares the operation of the most sophisticated algebraic system with the HP RPN system to illustrate the ramifications of each system's features to a user.

Algebraic systems compared with the HP RPN System

For these comparisons, the algebraic system which includes a built-in hierarchy and parentheses is used. The systems are compared to illustrate differences in consistency of operation, display feedback, keystroke efficiency, and error recovery.

Consistency of Operation

As discussed previously, one of the features of HP RPN is the consistency a user experiences in the ordering of number and function entry. Algebraic systems lack this consistency as shown in this example:

$$\frac{2 \times \text{Cos } 72^\circ}{\ln 2} + 17$$

In HP RPN, the order of key entry is as follows:

KEY	OPERATION PERFORMED
72 [COS]	Cos 72° is calculated
2 [×]	2 × Cos 72° is calculated
2 [ln]	ln 2 is calculated
[÷]	$\frac{2 \times \text{Cos } 72^\circ}{\ln 2}$ is calculated
17 [+]	17 is added to $\frac{2 \times \text{Cos } 72^\circ}{\ln 2}$

The function keys always follow directly the numbers to be operated on, whether the function is a single-argument function (Cos, ln) or a two-argument function (×, ÷, +).

In algebraic, the order of key entry is as follows:

KEY	OPERATION PERFORMED
2 [×]	Multiply is entered after the 2, awaiting the second argument.
72 [COS]	72 is entered prior to cosine, and the cosine is calculated immediately.
[÷]	Divide is entered, awaiting the second argument for divide.
2 [ln]	2 is entered prior to ln, and the ln is calculated immediately.
[+]	Plus is entered, awaiting the second argument.
17	The second argument for the plus is entered.
[=]	Equals is entered, signaling the end of the calculation.

Note how for two-argument operations (x, ÷, +) the operator is entered **between** the arguments; but for one argument operations, (Cos, ln) the operator is entered **after** the argument. This characteristic is a departure from the idealized algebraic system covered in the last section and results in an inconsistency that must be understood and managed by the user.

Display Feedback

Consider how a calculator can "talk back" (through the display) during operation. In a lengthy calculation, the display could show something that related to each step as it is performed; or it could show something that is meaningless, out of sequence, or which takes considerable understanding of the logic system to interpret.

Consider the following example, comparing the display feedback for HP RPN and Algebraic:

$$\frac{6 \times \sqrt{22}}{17} + \text{Sin } 37^\circ$$

HP RPN			ALGEBRAIC		
Key	Display	Comments	Key	Display	Comments
22.	22.		6	6	
[√x]	4.69	Result of square root.	[×]	6.	Multiply held pending until multiplicand is entered and calculated.
6.	6.		22	22	
[×]	28.14	Result of multiplication.	[√x]	4.69	Result of square root.
17.	17.		[÷]	28.14	Result of previous multiplication, divide held pending.
[÷]	1.66	Result of division.	17	17	
37.	37.		[+]	1.66	Result of previous division, add held pending.
[SIN]	0.60	Result of sine.	37	37	
[+]	2.26	Result of add. Final answer.	[SIN]	0.60	Result of sine.
			[=]	2.26	Result of previous addition. Final answer.

In the HP RPN system, the display always shows the results (intermediate answers) of the function just pressed. This allows the user to check for resonableness all the way through a calculation or discover where a computational problem occurs. The calculator **always** executes a function the instant it is pressed, it does not require storing operations for later execution (pending operations).

In this algebraic system, the display sometimes shows results of a previous operation when a later operation is pressed and sometimes shows the results of the operation last pressed. It is a characteristic of the

algebraic system that there is inconsistency in the relationship of the display contents and the function order.

When parentheses are included, this characteristic is more dramatically illustrated. In HP RPN, a problem with parentheses is worked beginning at the innermost set of parentheses, as it would be done by hand. In algebraic, it is worked from left to right.

Consider the following problem:

$$5 \times (4 + (3 \times (2 - 6)))$$

HP RPN			ALGEBRAIC		
Key	Display	Comments	Key	Display	Comments
2.	2.		5	5	
ENTER↑	2.00		×	5.	Multiply held pending.
6.	6.		(	5.	
−	4.00	Result of the subtract.	4	4	
3.	3.		+	4.	Add held pending.
×	−12.00	Result of the multiply.	(	4.	
4.	4.		3	3	
+	8.00	Result of the add.	×	3.	Multiply held pending.
5.	5.		(	3.	
×	−40.00	Result of the final multiply.	2	2	
			−	2.	Subtract held pending.
			6	6	
			)	−4.	Result of subtract 2 steps back.
			)	−12.	Result of multiply 6 steps back.
			)	−8.	Result of add 10 steps back.
			=	−40.	Result of multiply 14 steps back.

In this algebraic system, the display contents have lost virtually any usefulness to the user. In essence, the man/machine dialogue has become a man-to-machine monologue until the last = key is pressed. The result is that the calculator does not aid the user in following through his calculation, thus increasing his own responsibility to do so and adding to the probability of making an error.

Keystroke Efficiency

In the problem previously worked: $5 \times (4 + (3 \times (2 - 6)))$, note the number of keystrokes required to work the problem in the two systems. In HP RPN, 10 keystrokes were required; in algebraic, 16 keystrokes were required.

The HP RPN system does not require the entry of parentheses into the calculator, resulting in fewer keystrokes. It is widely accepted that in problems with even moderate complexity, algebraic systems will require a greater number of keystrokes due to the necessity to enter parentheses. Although parentheses are required to remove ambiguity in written expressions, they are not required in the operation of the HP RPN system.

This feature of HP RPN is also important in programmable calculators because it conserves program memory space.

Error Correction

Some of the capability of the HP RPN system to recover from entry errors has already been covered in an earlier section. The fact that there are no pending operations and that the stack and LAST X contents can be accessed and manipulated give the user a flexibility to "back-up" or even reverse an operation performed by mistake. The following examples point out typical error situations and show how an algebraic system deals with them versus the HP RPN system.

Consider the following problem:

$$(5 + 3) \times 6$$

Assume the user inadvertantly pressed + instead of ×, and then realized his mistake:

$$(5 + 3) + \ldots \text{ oops!}$$

To recover in the algebraic system, the user could attempt to first **add** the 6, then **subtract** it, then go back and multiply it as desired:

KEY	COMMENTS
(	
5	
+	
3	
)	
+	'Plus' mistakenly pressed.
6	Try to recover by first adding, then subtracting the 6.
−	
6	
×	Now multiply as originally intended.
6	
=	Incorrect answer is displayed as −22.

This procedure failed to correct the error. The problem occurs because of the built-in hierarchy which causes execution as shown below:

$$(5+3)+6-6\times6$$

or

$$8+6-36=-22$$

Another approach would have been to key in zero (instead of six) once the user realized he had mistakenly pressed the plus key. This would seem to cancel the effect of the mistake, then the multiply could be performed. However, the answer will again be wrong, but different than before:

$$(5+3)+0\times6$$

or

$$8+0=8$$

One method which will work is to prematurely complete the calculation by pressing = , thus completing all pending operations and allowing a new calculation to begin.

$$(5+3)+0=8$$

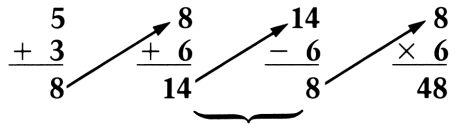
$$8\times6=48$$

Of course, if this simple expression were imbedded in a longer expression, pressing the [=] would cause **all** open parentheses to be closed and **all** pending operations to be executed. The result may bear no relationship to the desired calculation.

On the HP RPN system, the error would be corrected as follows:

KEY	COMMENT
5	
ENTER↑	
3	
+	
6	
+	'Plus' mistakenly pressed.
LASTx	Bring the last number (6) back into X. (Using LAST X is not necessary, just convenient.)
−	Subtract it, thus reversing the incorrect addition.
LASTx	Bring it back again.
×	Multiply it as intended and get correct answer of 48.

Logically, what has been done here is as follows:



mistake is completely cancelled.

Because intermediate results are calculated as each operation is entered, then operations can be reversed. Further, the LAST X function ensures accurate recall of the number incorrectly operated on.

Another type of error that can occur is incorrect entry of parentheses.

For example, consider the problem:

$$\frac{(17-5)^2}{16-4}$$

Assume that the numerator was evaluated correctly; but when the divide operation was performed, the user left out the parentheses for the denominator as follows:

$$(17-5)^2\div16-4=4$$

The result is incorrect, of course, because the divide operation utilized 16 as the denominator, rather than 12, (16 −4). The correct answer is 12.

To recover, the user may try to reverse the sequence, but the built-in hierarchy will work against him:

$$(17-5)^2 \div 16 \underbrace{-4+4\times 16}$$

attempt to reverse the sequence

When this is attempted, the final multiply is executed prior to the final add, resulting in the following:

$$(17-5)^2 \div 16 - 4 + 4 \times 16$$

or

$$144 \div 16 - 4 + 64$$

or

$$9 + 60 = 69$$

Because the HP RPN system user need not remember to add parentheses around the denominator and insert them into the key sequence, he is less likely to make this error in the first place.

After calculating the numerator with HP RPN, the user keys in the value 16, and **then** he makes the decision as to what operation to perform. Each operation executes immediately on the arguments the user has already entered or calculated; there is no need to 'look ahead' or anticipate the arguments for an operation already keyed in.

Assume, however, that the HP RPN user made the exact same mistake, logically, as the algebraic user; i.e.,

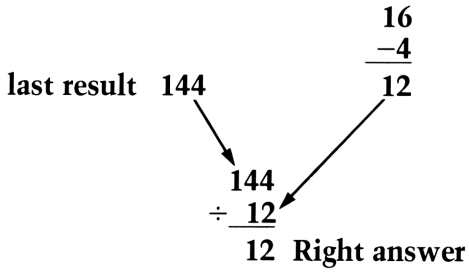
$$(17-5)^2 \div 16 - 4$$

At this point, the value 5 would be in the X register (display). To recover, the user reverses the order of the incorrect calculations, then re-enters them properly. Logically, the sequence is as follows:

Original calculation

Reverse to where the error occured

Re-enter correctly



The error recovery capabilities of a calculator are important especially for lengthy problems. The prospect of beginning a problem over after many keystrokes have already been entered is not a pleasant one for a user. Recovery capabilities should yield a higher confidence and a more efficient use of the user's time.

Summary

This booklet has explored the features of the two types of logic systems in use for sophisticated handheld calculators today. In the final analysis, these features must be evaluated as to their effectiveness in providing answers to the type of problems a user will be encountering in his work. For very basic operations, the choice is largely a matter of whether the user prefers to enter problems as he usually sees them printed (algebraic) or in the order he would use to solve the problem by pencil and paper (HP RPN).

For problems of even moderate complexity, however, the relative effectiveness of the systems becomes much more apparent. The criteria for evaluating this "effectiveness" is derived from how the system helps the user obtain his answers with specific regard to:

1. The user's confidence in the accuracy of results.
2. Simplicity of operation.
3. Efficiency of the user's time and energies.

Confidence in Accuracy of Results

Several features of the HP RPN system contribute to a greater user confidence in results than with algebraic systems. The consistent approach to number and function entry results in a lower likelihood of errors. Because parentheses are never entered into the HP RPN calculator, there is no possibility of mislocating them or of forgetting to include them. The display of intermediate results allows the user to "follow" the calculation, thus aiding him in keeping track of his place in complex expressions and in checking reasonableness of intermediate results.

Simplicity of Operation

As was mentioned, either system is quite straightforward for very basic operations, such as simple arithmetic with two arguments. As expressions become more complex, the consistent entry convention of the HP RPN system demands less from the user. Whereas, all operations are approached the same in HP RPN, the user must learn a variety of rules and conventions for his algebraic machine; e.g., some operators must be placed between arguments, some must follow their arguments, built-in hierarchies must be learned and remembered, and parentheses must be carefully entered to remove ambiguity. Once learned, the HP RPN system allows the user to solve problems of greater and greater complexity without adjusting his approach, whereby the algebraic user may need to learn and remember additional structures of the system.

Efficiency

It is generally accepted that HP RPN systems require fewer keystrokes for other than very basic problems. This is due to the basic structures of RPN which eliminates the need for entering parentheses. However, errors also effect the users efficiency in operating his calculator. The probability of making errors on the HP RPN system is lower, and error recovery is straightforward. Errors are less likely because of the consistency of the entry convention, the elimination of parentheses,and the display feedback which aids the user in tracking a complex problem. Error recovery, often impossible in an algebraic system, is straightforward in HP RPN, eliminating the need for restarting lengthy calculations.

Overall, the HP RPN system offers superior operating characteristics for calculator users who deal with problems of moderate or greater complexity. It is a system that was designed to accept more of the problem-solving burden in the calculator, while minimizing the burden on the user.

5953-1930

Printed in U.S.A.