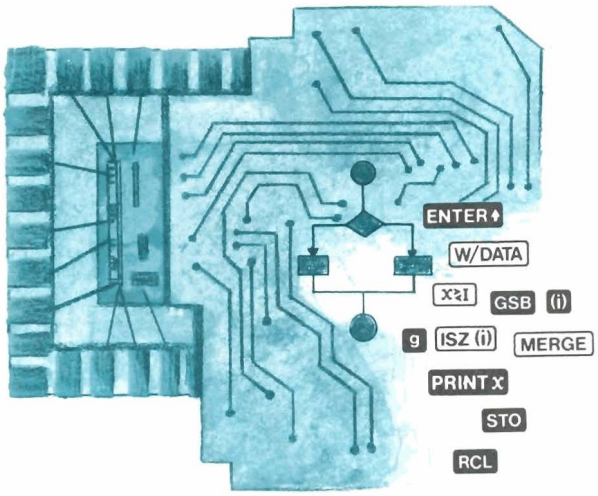


Computer Museum

Featuring this issue:

May 1979 Vol. 3 No. 2

HEWLETT hp PACKARD

HP Key Notes

2½ Years Later

"Where is Corvallis? What is it like?"
"What does the factory look like?"
"How large is it? How many people?"

Those are questions we see over and over in your letters. Therefore, since we haven't said much about the "home" of your calculators since Volume 1 Number 1, we decided to answer those questions for all of you.

Corvallis is at latitude 44°34′ north and longitude 123°17′ west. Or, if that doesn't help, we are located 35 miles (56 km) south of Salem (state capital), 11 miles (18 km) west of Albany, 40 miles (64 km) north of Eugene, and 53 miles (85 km) east of Newport, which is on the Pacific Coast. Portland is just 81 miles (130 km) north of us.

Corvallis lies in the beautiful Willamette Valley, which is bordered on the east by the Cascade Range mountains and on the west by the Coast Range mountains. The Willamette River flows along the east border of the city and then north to Portland, where it empties into the Columbia River.

Oregon State University (OSU) is located here, and it averages about 15 to 17 thousand students. Total population of Corvallis is about 40 thousand.

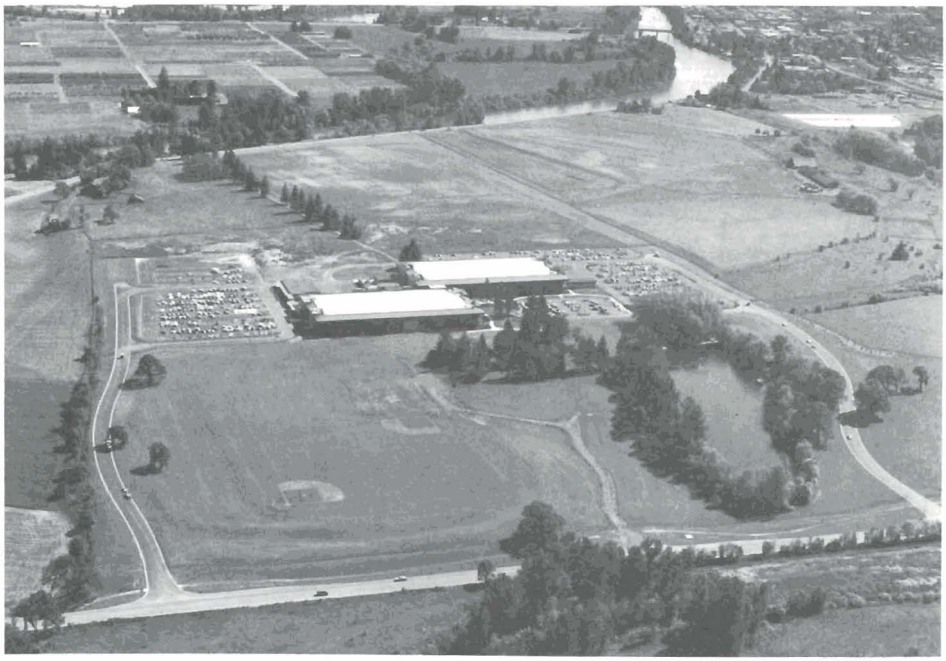
It rains a lot, but it's green and clean here, and we can easily see Mt. Hood (well, most of the time!), which is about 100 miles (161 km) northeast. All in all, it is a nice place in which to live and to make calculators.

As you can see in the photo, there are two main buildings at the factory, which is on a 140-acre site just northeast of the city. Each building totals 152,000 square feet of area, and there is a 60,000 square foot storage area in the basement of the building in the foreground. Total square footage at the site is 416,000.

The building in the back houses the calculator assembly lines, service department, shops, personnel, marketing department, accounting, and all other administrative functions. The other building houses a highly sophisticated and up-to-date integrated-circuit (IC) facility plus the research and development functions. Approximately 1200 people work at the factory.

At right, center, there is a 5-acre natural lake. Clean waste water used in the IC processing area is recycled by pumping it to the lake and then onto the landscaped grounds. It then returns, in part, through the earth's water table to the lake. And, of course, the lake serves as a source of water for fire and other emergencies.

Like all Hewlett-Packard sites, it is very well-kept and is a beautiful place in which to work. We are as proud of our factory as we are of our calculators!



A Pleasant Surprise!

Are you tired of seeing prices going up, up, up? Well, here's some good news for a change. Effective May 21, prices of the HP-31E, HP-32E, and HP-33E were reduced as follows:

HP-31E Scientific Calculator. Was $60, now $50.* This is a basic scientific model in the tradition of the original HP-35 and the popular HP-21.

HP-32E Advanced Scientific Calculator. Was $80, now $70.* This calculator also includes statistical functions and 15 addressable storage registers.

HP-33E Programmable Scientific Calculator. Was $100, now $90.* This is a keystroke programmable model with 49 lines of program memory and many more features.

Stop in to see these bargains at your local HP dealer's store. It might seem early to buy a back-to-school present, but how often—these days—do you see prices going *down*?

** U.S. dollars. See note at bottom edge of cover.*

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Library Corner

As you can imagine, there are quite a few HP-65, HP-67, and HP-97 calculators being used all over the world by people like yourselves. So it takes a lot of paper to print enough catalog addendums to satisfy this elite group. Unfortunately, the northwest has recently suffered some problems in the paper industry and so we've had to postpone a scheduled addendum until we were positive we could get enough paper to print it.

However, we are happy to report that *Catalog Addendum #4* will be distributed during the month of June. It will contain 500 new programs and will include a greatly expanded Application Category Table to help you find programs in fields that interest *you*. The table was revised to reflect the broader range of fields and subjects extant in this day and age. Thus, we can now highlight such things as solar energy, computer science, home computing, dentistry, nutrition, education, etc. Programmable calculators are helping to change our world, so we must change with it. We think you'll find the new categories a great asset.

Now, before you jump the gun, remember that *Addendum #4* will be mailed via third-class bulk mail, so allow at least until the end of July before you check with us. Also, make sure your address is up to date; **the addendum will not be forwarded if you have moved.**

ORDERING PROGRAMS

None of the individual programs in this issue are available in Europe at this time. They will probably be added to the European Library Catalog at a later date. And, don't forget, if you live in the U.S.A., you can order Library programs you see here in KEY NOTES by calling the toll-free number on the Order Blank.

Library programs are available in two forms: A set of the program listings and instructions (software) is $3*, and the fee is $5* for a set of software *and* a recorded magnetic card.

NEW HP-67/97 PROGRAMS

Here is another "special" program. It is *complete* . . . right down to program flow diagram, nomenclature list, and terrific documentation. There are 448 steps, 2 cards, and 30 pages. The name is: **Compressible Flow #67000-99987,** and the price is $13.00.* It is the exacting work of **Gerard A. A. Westen** of Newark, Delaware (whose other "special," *Orifices for ME's*, appeared on page 2 of Vol. 2 No. 3). Here is the abstract:

This program solves adiabatic and isothermal compressible fluid flow problems in conduits, and will solve pressure drop under many conditions. It is applicable for the subsonic, sonic, and supersonic flow region and includes two program cards, one for English units and one for SI units. A calculator printout tape can be attached to the compressible fluid calculation sheet for permanent record purposes. This program will free engineers from laborious complex calculations and charts.

(Congratulations–once again–Mr. Westen. I know engineers will love this one! A very fine job! Ed.)

The next offering for this issue is, again, a "special" program. It consists of 1291 steps, 7 cards, and 30 pages. The program is designed to fit electrolytic conductance/concentration data to the Fuoss-Onsager-Skinner equation. Inputs are the viscosity and dielectric constant of the solvent, temperature, and the concentration-conductance data. Both associated and non-associated electrolytes are treated.

The curve-fitting is done by the Gauss-Newton regression method. While the equations require a starting approximation, this is not a problem, because convergence will occur for almost any approximation. Details of the Fuoss-Onsager-Skinner equation are given in an attached supplement, along with additional references.

Outputs are: limiting conductance, the L parameter, ionic radius, and, for associated electrolytes, the ion association constant.

The name of this program is: **Curve Fit for Electrolytic Conductance #67000-99977,** and the price is $15.50.* This Herculean effort is the work of **Bruce W. Clare** of Coolbellup, Western Australia, who has done a remarkable job of documenting his program.

For a change of pace—and direction—here is a recent program that makes use of some former programs so that accountants, and others, can have a so-called "single source" for this type of computation. The author said it was actually a joint venture with **Harold Scheinhaus** (Wheaton, Maryland) because the program incorporates some "perpetual calendar" data contributed earlier by Mr. Scheinhaus.

67/97 Interest Calculator Between Any Two Precise Dates (#03608D)

This program calculates interest between two precise dates (day/month/year), using a 365/360 interest rate factor and then cumulates interest if there are multiple transactions between borrower and lender. It uses a modified version of *Perpetual Calendar* (02198D). It is somewhat similar to program #00227D but uses a different method of days calculation, which works for any two dates since the adoption of our calendar. (138 steps, 5 pages)

Author: **Steven Guralnick**
Daly City, California

Now, believe it or not, a *third* "special" program, and this one is from **Bernard L. Golding,** who is a professional engineer and Chief, Water Resources (South), for Howard, Needles, Tammen & Bergendoff of Orlando, Florida. His program is titled, **Dissolved Oxygen Concentration in Rivers #67000-99978,** and the price is $9.50.*

Mr. Golding has written and documented an impressive program. It is 410 steps long, is contained on 2 cards, and includes 21 pages of typed, easy-to-understand documentation. His abstract does a good job of describing his program, so here it is:

The program consists of a two-part mathematical model that simulates in rivers and canals the steady-state dissolved oxygen concentration resulting from both carbonaceous and nitrogenous biological oxygen demands (CBOD and NBOD). In the first part, BLEND, mass balances are performed to determine inputs to a river reach. In the second part, DOREM, the dissolved oxygen remaining at the end of the river reach or at one mile intervals along the river reach are computed.

Seven pages of the documentation are devoted to a small textbook describing the program, presenting the equations, and defining the variables. The three-page sample problem shows all keystrokes and outputs in an easy-to-follow format, and detailed, clear User Instructions are given in seven pages. The final four pages give the program listings, with extensive comments for the two cards.

(An excellent example of proper documentation, Mr. Golding; we congratulate you for carefully and diligently recording your work. Ed.)

We Get Letters

It is nothing new to readers of KEY NOTES that our calculators are sometimes used for some very unusual tasks, or in some very unusual places. Therefore, to make sure we keep you well informed about the many things you can do with your calculator, we present the following letter.

Gentlemen:

This is to show that an HP-67 calculator may have other uses besides mathematical calculations.

Convince yourself and your friends that the human eye has a blind spot: Turn the HP-67 on and press DSP 0. Press h REG and look with your *right* eye at the zeroes blinking at the left side of the display as the register contents are displayed. If you slowly move the calculator back and forth around a distance of approximately 8 inches from your eye, you will find a distance at which the register numbers on the right will not be visible! You just located the blind spot on your right eye. (Your left eye should be closed or covered.)

Conversely, looking with your *left* eye (covering the right) at the register numbers blinking on the right side of the display, you will find a blind spot there, too, when you cannot see the zeroes displayed on the left side.

Sincerely,
Carlos R. Schwantes, Monroeville, Pennsylvania

(Thanks for the letter, Mr. Schwantes. I'll have to admit I was skeptical, but it does work. Ed.)

**U.S. dollars. See note at bottom edge of cover.*

825 Flags . . . Continued

Did you notice the blank space at the bottom of the center column on page 5 of the last issue? Unfortunately, it was not left there on purpose. There should have been a paragraph, in blue ink, as follows:

"Starting the column this month is a real breakthrough in programming. We've seen programs that can provide up to 750 software flags, but this one, from **Cass R. Lewart** of Holmdel, New Jersey, can provide up to 825 flags! The flag program was jointly developed by Mr. Lewart and his 16-year-old son, Dan."

Through some unexplained error during the printing of the newsletter, the paragraph disappeared, so we asked Mr. Lewart to forgive us and to take us up on an offer to print his picture along with the missing paragraph. So here it is. And please accept our apologies, Mr. Lewart.

Dan and Cass Lewart. Dan has won several state-wide competitions in mathematics, and both father and son are active members of PPC, the calculator user's club. (A neighbor, **Dankwart Koehler,** took the photo, processed the film, and made this print.)

The 825-flag routine by the Lewart's elicited a few questions, so here's a continuation of the subject.

825 Flags—A Base b Encoding/ Decoding Programming Technique

In our last issue of KEY NOTES, we had a 66-step routine for the HP-67/97 that provides control of up to 825 flags. The routine used a programming technique that may be described as, "Base b Encoding/Decoding." Using a base of 2, the routine stored 33 flags in one HP-67/97 data register. This article describes the techniques used in the routine and shows how other combinations of numbers may be packed into an HP-67/97 register using other bases.

To understand Base b encoding/decoding, we must understand what is the base of a number system. In our day-to-day dealings with numbers, we use the numbers—or digits—0 thru 9. So the base of our number system is 10. When we count, we start with 0 and proceed to 9. When we get to one less than the base, i.e., 9, we carry one count to the next place and repeat the count from zero. Ten follows 9, 100 follows 99, etc. We observe that 123 is a number that may be described as 1 in the hundreds place plus 2 in the tens place plus 3 in the units place. This may be illustrated:

$$\underline{1}\ 100\text{'s} + \underline{2}\ 10\text{'s} + \underline{3}\ 1\text{'s}$$

This may also be expressed using the place value and the base of the number system being used.

$$\underline{1} \times 10^2 + \underline{2} \times 10^1 + \underline{3} \times 10^0$$

Table 1 further illustrates the idea.

Table 1. Place Numbers and Values for 123

Place	Third	Second	First
Place Value	100	10	1
Place Count	2	1	0
Number	0	0	0
	0	0	1
	0	0	2
	0	0	3
	.	.	.
	.	.	.
	.	.	.
	1	2	3

Now, let's apply these ideas to the Lewart's flag routine in the February 1979 issue of KEY NOTES. As you know, a flag has two states, set (1), and clear (0). So a base of 2 is suitable for this application. Table 2 shows base 2 numbers in the same format as table 1.

Table 2. Binary Values for Base 10—0 thru 16

Place		5th	4th	3rd	2nd	1st
Base 10 Place Value		16	8	4	2	1
Place Count (flag #)		4	3	2	1	0
Base 10 Number:	0	0	0	0	0	0
	1	0	0	0	0	1
	2	0	0	0	1	0
	3	0	0	0	1	1
	4	0	0	1	0	0
	5	0	0	1	0	1
	6	0	0	1	1	0
	7	0	0	1	1	1
	8	0	1	0	0	0
	9	0	1	0	0	1
	10	0	1	0	1	0
	11	0	1	0	1	1
	12	0	1	1	0	0
	13	0	1	1	0	1
	14	0	1	1	1	0
	15	0	1	1	1	1
	16	1	0	0	0	0

In base 2 a number place is called a BIT for *BI*nary digi*T*. Five bits are shown. The number 7 in base 2 (binary) is 111, and 3 bits are required to represent it. The base 10 value for binary 111 may be obtained by adding the base 10 place values as shown below.

$$\underline{1} \times 2^2 + \underline{1} \times 2^1 + \underline{1} \times 2^0 = 4 + 2 + 1 = 7$$

Storing 33 flags in one register is accomplished by representing the flags as a binary number stored in base 10 form.

If each place can represent three flags, then ten places can represent 33 flags. Each place cannot represent four flags (bits) because not all combinations of set and clear, 1's and 0's, can be represented by the highest digit 9, even though it is a four-bit number (1001). The upper limit of flags is better seen by considering that $2^{33} - 1$ is 8,589,934,591, which is the highest all-flags-set, base-10-value less than 11 digits. The next full set of flags is $2^{33} - 1 =$ 17,179,869,183, which is too large.

Study the simplified routines below to follow the Test, Set, and Clear process. Labels A-Test, B-Set, and C-Clear function the same as in the Lewart's routine. The error-correcting and housekeeping steps that direct each group of 33 flags to successively higher registers have been omitted for illustration and study. Routine A tests a given flag by dividing the register contents by 2 raised to the flag number power, then taking the integer part and testing for odd or even by dividing by 2. An even result displays a zero, for clear, by the last step, FRAC. A remainder (0.5) indicates the flag is set. (Key DSP, 0 to show "1.") The ABS step in routine A is used to store 2^n in LAST X (n is the flag number) and it is used by routines C and B if required.

Test Flag n	Clear Flag n	Set Flag n
*LBL A	*LBL B	*LBL C
2	GSB A	GSB A
X≷Y	X=0?	X≠0?
Y^x	RTN	RTN
ABS	X≷Y	X≷Y
RCL 0	ST- 0	ST+ 0
LSTX	RTN	RTN
÷		
INT		
2		
÷		
FRC		
RTN		

Routine B uses routine A to test the flag. If it is clear (0), no further action is taken and program execution stops at the first RTN. If the x=0? test shows the remainder (0.5) because the flag was set, the value of 2^n remaining in the stack is subtracted from the register after the x≷y step.

To better visualize the process, refer to table 2 and follow the process using the more familiar base 10 values. For example, suppose flags

0, 1, 2, and 3 are set. (Table 2 only shows all combinations of four flags.) All the other flags are clear. Under these conditions, R_0 would contain 15. Now suppose we wish to clear flag 2. In base 2, the first four flags set condition of 1111 would become 1011 when flag 2 is cleared. This corresponds to a base 10 value of 11. When subroutine B determines that the flag is set, it subtracts 2^n, which is 2^2 in this example. Considering only flag 2, the binary value is 0100, which is 4 in base 10. Subtracting from 15 gives the desired result:

Base 2	Base 10
1111	15
-0100	-4
1011	11

Routine C is identical to routine B except it does nothing if the fraction part of subroutine A is not zero, but adds the value of 2^n if the flag is clear. It is just the reverse of the example above.

The simplified routines above are fast and short, but have a limit of 30 flags per register. Because the routine uses the fractional part in testing a flag, the large numbers result in round-off error when flag numbers larger than 29 are used. The Lewart's additional 39 steps avoid the round-off error and provide for automatic addressing of the registers. If the full storage capability of the registers is used, it is possible to store up to 41 flags in each register. (Two more by using the signs of the characteristic and mantissa, and six more by using the characteristic to gain the additional eight flags.)

The principles described above can be generalized to increase register number capacity. In the more general sense, only two routines are required, one to store and one to recall the number. However, an additional input is required: the location in the register must also be entered.

In the Flags examples above, the register was divided into parts identified as flags 0-29. To identify the location within a register, the positions will also be numbered 0, 1, 2, 3 etc., up to the limit of the 10-digit register capacity. The number to be stored is an integer, i, to be stored into position p. Almost *any* base may be used. The base is stored in register B. If you use these routines in a program, you can free register B by having the program contain the base in each step that recalls register B. The following instructions apply.

1. Store base b in register B.
2. To store number i in position p, key p, ENTER, i, and press E.
3. To recall number i from position p, key p and press A.

Recall Position p	Store i in Position p
*LBL A	*LBL E
RCL B	STO E
x≷y	x≷y
y^x	GSB A
ABS	x
RCL 0	ST-0
LST x	x≷y
÷	RCL E
INT	x
LST x	ST+0
RCL B	RTN
÷	
INT	
RCL B	
x	
−	
RTN	

Table 3 gives the limits of a few of the more common bases used in programs that require data packing to more fully utilize the storage capacity of the HP-67/97. Other methods have been used; for example, decimal point encoding, and logarithmic encoding. Of all the methods used, however, the most amazing is Base b encoding/decoding using base 2 to represent flags.

Table 3. Base b Encoding/Decoding Limits

Number Range, i	Positions	Base
0-1	0-29 (30)	2
0-2	0-18 (19)	3
0-4	0-12 (13)	5
0-9	0-9 (10)	10
0-20	0-6 (7)	21
0-99	0-4 (5)	100
0-99999	0-1 (2)	10^5

CRYPTOLOGIA is a quarterly journal (100 pages per issue, 8½ by 11-inch format) with January, April, July, and October issues. It is the only journal dedicated to a broad, yet in-depth, coverage of cryptology and related fields. Among the subjects discussed are: Computer Security, Military Science, History, Mathematics, Literature, and Ancient Languages. Regular features include: Crypt analyst's Corner, Devices and Machines, Course in Cryptology, Book Reviews, and a column with bits about sources and happenings. If you are interested in this sort of thing, write for more information to: *CRYPTOLOGIA; Albion College; Albion, Michigan 49224. A year's subscription costs $20.** Also, back issues and bound volumes of past years are available.

(Thank you, Dr. Winkel, for sending us information about this service. Ed.)

* *U.S. dollars. See note at bottom edge of cover.*

Too Many Inputs/Outputs?

We recently received a very interesting letter from **Cass R. Lewart** (Holmdel, New Jersey) and, since we have had questions about this same function, we decided to share it with you.

What to Do When Your Program Has Over 10 Inputs/Outputs.

A good practice that produces an easy-to-use program is to assign only the ten non-numerical labels A-E and a-e to individual program inputs or outputs. However, it frequently happens that a program requires more than 10 inputs or outputs. Assigning numerical labels 0-9 necessitates pressing three keys (f GSB n) before entering a value or obtaining a result and should be avoided. Another straightforward method, namely keying GTO .nnn followed by R/S is even more cumbersome. What follows are several alternate methods of providing additional inputs or outputs, which you may find quite convenient when you run out of non-numerical labels. Each method has some advantages and disadvantages; the choice will depend on your particular program and personal preference.

1. Key in 0 through 9 followed by a non-numerical label. Good for up to 10 additional inputs or outputs.
2. Key in a non-numerical label by itself or press *any* numerical key followed by a non-numerical label. This method is frequently used for sharing the same label for input or output. Make sure that F3 is clear when you record on the magnetic card.
3. Repeated keying of the same non-numerical label. **This and the following methods have the disadvantage that inputs or outputs must follow in the same sequence.** Use of a prompting routine is recommended.
4. Alternate use of the same non-numerical label.
5. Use of a non-numerical label followed by repeated R/S for inputs, PAUSE or -x- for outputs. Several outputs can then be displayed in succession. This method is particularly suited for programs with many inputs or outputs.
6. Use of a non-numerical label followed by SST's, followed by R/S. This method saves on steps but does not allow for prompting; therefore it should be used as a last resort.

The following examples illustrate the six methods. An asterisk (*) shows suggested locations for a call to a prompting routine. The main program should reset the counter of the prompting routine R_i to 0.

1.	*2.*	*3.*	*4.*	*5.*	*6.*
LBL A	LBL A	LBL A	LBL A	LBL A	LBL A
STO I	F3?	*	F2?	*	R/S
GSB i	GTO 1	.	GTO 1	.	STO/RCL 1
RTN	.	.	*	.	STO/RCL 2
LBL 1	.	RTN	.	R/S or	.
.	RTN	LBL A	.	PAUSE	.
.	LBL 1	*	SF 2	*	RTN
RTN	.	.	RTN	.	*Prompting*
LBL 2	.	.	LBL 1	.	*Routine:*
.	RTN	RTN	*	R/S	LBL 9
.			.	*	ISZ I
RTN			.	.	DSPO
			RTN	RTN	RCL I
					PAUSE
					DSP 5
					RTN

HP-97 For Basketball?

Well, maybe not for basketball *per se,* but definitely for basketball *coaches!*

Dr. Aaron N. Moen, presently a professor and research scientist at Cornell University, has written a most remarkable book for coaches who are serious about evaluating team statistics, and for college professors in athletic theory who want to introduce students to a most up-to-date approach to evaluating team statistics. The book, *Basketball Performance Profiles,* describes ways to profile games *in progress,* and to evaluate seasonal trends from game to game. The first part of this book describes ways to do this with a clipboard and a handheld calculator. The techniques described in the second part of the book are based on fast, programmed electronic computing. Efficiency ratios are calculated during the game and as game summaries, providing insight into the floor functions that were strong, and those that were weak. Over 200 evaluations are completed during a game (26 evaluations per profile), and game summaries provide 78 additional evaluations. All of these are completed within 5 minutes after the game is over, using the instructions, program steps, and equipment described in this 175-page book.

If you live in the U.S. and want to obtain this book, send a check or money order for $10.00* to Corner Brook Press; Box 106; Lansing, New York 14882. If you live outside the U.S. or just want more information, write to the above address.

And if your local high school, college, or company basketball team is losing too many games, perhaps you should tell the coach about this newfangled science. And, just think: All along you thought the HP-97 was just another superlative scientific calculator for serious number-crunching in such fields as mathematics, medicine, statistics, and so on. As it turns out, it also happens to be the most powerful coaching aid ever available at courtside.

** U.S. dollars. See note at bottom edge of cover.*

The Ultimate in Merging Proficiency

That title may seem a bit strong, but after you read the following letter from **James H. LeMay** of Houston, Texas, perhaps you'll agree. At any rate, we think you will enjoy his contribution to KEY NOTES.

Dear HP KEY NOTES:

I agree with **A. G. Burns** (in Vol. 3 No. 1) that one of the most useful attributes of the HP-67/97 is the ability to accept programs longer than 224 steps, but I think I have found a simpler technique to accomplish unlimited merging and yet maintain all the advantages of his method for programming as well as data storage.

The technique for programming takes advantage of two aspects of the HP hardware. One is the execution of step 001 immediately following step 224. The other is that, when a second card is merged, the calculator accepts from the merging card only the number of steps that would fill up 224 steps of programming in the calculator.

For the first card in a long series to be merged, the program would be:

001	PSE	220	2
002	GTO i	221	STO I
003	*LBL A	222	PRT X
		223	*LBL 2
	.	224	MRG
	.		
	.		

Now the second card, card 2, can be programmed to be:

001	GTO i	219	3
002	*LBL 2	220	STO I
	.	221	PRT X
	.	222	*LBL 3
	.	223	MRG
		224	R/S

Notice that card 2 can have a maximum of only 223 steps.

Starting with step 220 on card 1, the "2" command serves three purposes. First, for the HP-97 only, the number of the next card to be used can be printed on tape to keep track of the card in use. Secondly, as the calculator begins the merge loop, the number of the next card to be merged is displayed during the pause as a visual reminder of the next card to be used. (Not all programs use cards in consecutive order.) Thirdly, as the calculator executes the GTO i statement, it returns back to the merge loop at LBL2 and sits in a holding pattern as it flashes the number of the next card each time the pause function is executed.

As card 2 is called and input into the calculator, the programming will now look like:

001	PSE	220	3
002	GTO i	221	STO I
003	*LBL 2	222	PRT X
	.	223	*LBL 3
	.	224	MRG
	.		

The only command left from card 1 is PSE, in step 001. Steps 001 to 223 on card 2 are now steps 002 to 224 in the calculator. Immediately after merging card 2 from the PSE command at step 001, the GTOi searches for LBL 2 because "2" was stored in the I-register back on card 1. But, because the program memory was rewritten with card 2, LBL 2 is now immediately after the GTOi statement in step 003, and it automatically continues the program execution. As the body of the program has finished execution, the new programming added from card 2 allows card 3 to be merged the same way. This time, 3 is displayed and stored in I and LBL 3 is the address of the new merge loop. The next cards will look like:

CARD 3		CARD 4		LAST CARD
001	GTO i	001	GTO i	001 *LBL A
002	*LBL 3	002	*LBL 4	.
	.		.	.
	.		.	.
	.		.	RTN
219	4	219	5	
220	STO I	220	STO I	
221	PRT X	221	PRT X	
222	*LBL 4	222	*LBL 5	
223	MRG	223	MRG	

The last card needs only a program body.

Merging cards (programs) in this manner can continue *ad infinitum.* It is convenient to number the merge loops in increasing order from card to card. For the program above, the merge loop was LBL 2 on card 1, LBL 3 on card 2, LBL4 on card 3, etc., up to LBLe and back to LBL 0 for card 20, but it does not have to be in order. The address can simply alternate between any two numbers, say 2 and 3, as more cards are merged. Card 3 (above) could have used LBL 2 for the merge loop, but it requires the command "4" after STOI to indicate the number of the next card to be used. For convenience and shorter programming, it is easier to leave it in order.

In the way of suggestions . . . if there is data stored in the I-register from the body of the program and it is necessary for it to be kept for future execution on other cards, the value can be easily recalled to the stack and juggled while a new card is being merged and then it can be replaced.

Also, if the total number of steps in the program does not add up to 223 and it is desired to merge future cards, the remainder can be filled with superfluous R/S commands any place convenient in the body of the program.

Finally, the PRT x command on the HP-67 is not necessary, but be sure the program retains the proper number of steps.

The advantages of this technique are:

1. Looped automatic merging (no keys to press).
2. Unlimited number of cards that can be merged.
3. Maximum programming space.
4. Total preservation of Set Status (no flags are used).
5. Visual indication of card to be merged.
6. Unlimited number of coffee (tea?) breaks between calculations.

What more could a programmer ask?

The technique for merging data cards automatically is less involved but more versatile. For example, in the course of programming, storage registers 0 to 13 need data from the corresponding registers on the data card, card 4. The program would look like:

.	4	F3?
.	PRT X	F3?
.	CF3	GTO 0
1	*LBL 0	.
3	MRG	.
STO I	PSE	.

As the program stores the number of the last register to be merged, the card number is printed on the HP-97 (a step not necessary on the HP-67) and continues to be displayed in the merge loop as long as no data is entered. Once the data is entered from both sides of the card, flag 3 is set and the GTO 0 is skipped, so execution of the body of the program can continue automatically.

This also allows two more options. The merge loop can be defeated by simply pressing any integer key on the keyboard while the display is flashing. Then the merging of data will be completely skipped and the data will be untouched (except that the stack was raised).

For the second option, the programmer can store just secondary registers. For example, suppose it was necessary to merge only secondary registers 0 to 3. The above program is identical except, to warn the programmer, the card indicator "4" could be changed to "4.2" meaning card 4, side 2. Then, when the calculator displays "4.2," side 2 of the data card should be entered, and only storage registers 10, 11, 12, and 13 have been added to memory. The calculator now displays **Crd.** Press any key to erase the display and the calculator will automatically continue execution.

Again, the advantages are:

1. Looped automatic merging.
2. Variable data entry.
3. Visual indication of the card and side to be merged.

I hope this has shown the power of the HP-67/97 merge capability and simplicity to its fullest.

(Indeed it has, Mr. LeMay. Many thanks for a fine job and for sharing your contribution with our readers. Ed.)

Spring is "Seed" Time!

John Craig, of Anaconda, Montana, contributed the following ideas on seeds. If you use seeds at all, this should be of interest to you.

Problem:

1. Seed given by HP for random number is hard to remember.
2. Some programs—like, games—need different starting sequences every once in a while.
3. It's hard to test other seeds to make sure they are as good as the one given.

Solution:

I experimented on our HP 9825A Desktop Computer and discovered several facts:

1. Seeds do recycle at exactly 500,000 as mentioned in the manual.
2. The 10 seeds formed from .n123123 (where n is any one of the 10 digits) form 10 starting seeds spaced exactly 50,000 seeds apart in the same tested sequence.
3. The 100 seeds formed from .ab23123 (where "ab" is 00 thru 99) form 100 starting seeds, each spaced exactly 5,000 apart, again in the tested sequence suggested by HP.

As an example of use, after loading a game, perhaps the first instructions would be to key in a two-digit integer between 0 and 100. The starting seed could then be formed by the program by tacking-on the "23123" and shifting the decimal point. This would guarantee a good sequence and would provide new seed sequences.

"25 Words" (More or Less!)

It is apparent from the mail we receive that most of you enjoy this column and read and/or use it. So you will notice that the column has been growing not only in size but also in scope. And it will continue to do so, as long as space, time, and inputs permit. We are only sorry that we cannot print all of the inputs we get. Anyway, here are some more interesting tidbits we know you will enjoy.

First entry this issue is a trick from **Frank A. D'Amico** (Pascagoula, Mississippi). It makes one of the *HP-67/97 Standard Pac* programs easier to use.

The following procedure will permit the use of both sides of the *HP-67/97 Standard Pac* program SD-12B ("English-SI Conversions") without having to reprogram each time you switch sides of the card.

Merge both sides of the card onto a blank card (169 steps) then insert "LBL 1" before step 001 and "LBL 2" after what is now step 081. CAUTION: To make sure you're in the proper place, GTO1 **must** be keyed *each time* before using LBLA-E or a-e on side 1, and GTO2 **must** be keyed *each time* before using LBLA-E or a-e on side 2.

Now, what's new in Wahiawa, Hawaii? That's easy; it's a neat and very simple plotting routine from **Dale P. Weber.**

I have come up with a Nonfunction Plotting Routine that can be used on both the HP-67 and HP-97. It uses one card for both the program and storage of the "print constant."

001	*LBLB	017	RCLA
002	STOA	018	-
003	-	019	RCLB
004	9	020	÷
005	÷	021	RND
006	STOB	022	10^x
007	RCLA	023	STOC
008	RCLB	024	RCLD
009	2	025	X≷Y
010	÷	026	÷
011	-	027	FRC
012	STOA	028	RCLC
013	RCLB	029	x
014	R/S	030	PRTX
015	*LBLA	031	R/S
016	DSP0		

The number (constant), 987654321.0, is stored in register D. To store it on the card, clear all registers except D, and put any instruction (say, CHS) in program line 113. (This line will not be used or recorded. Now load the program on side 1, then **Crd** will be displayed. Press CLx and then W/DATA. Now load side 2. Side 1 will contain the program and side 2 the constant. To use the plotting routine, enter the maximum (say, 250) and press ENTER↑. Then enter the minimum and press B. Now, just key in your values and press A after each one (with printer set for MAN). HP-97 users will obtain a graph and HP-67 users will be able to construct one by looking at the leftmost number.

(It works well, Mr. Weber, but it really is more convenient on the HP-97, with an already-plotted output. Also, the narrower the range the better the plot. Ed.)

This next routine was the result of a son's request to check his long division problems from school. The answers required the remainder to be shown, so **David L. Smith** (Burlington, Massachusetts) wrote, for his son:

Check Long division and Show Remainder.

001	*LBLA	010	x
002	÷	011	EEX
003	LSTX	012	4
004	X≷Y	013	÷
005	ENT↑	014	R↑
006	FRC	015	+
007	X≷Y	016	DSP4
008	INT	017	RTN
009	R↓	018	R/S

To use the routine, key in the dividend and press ENTER↑, then key in the divisor and press A. For example:

4359 ENTER↑ , 57 A --- 76.0027
Quotient = 76, Remainder = 27

(Now none of you has an excuse for not checking your son's or daughter's homework! Ed.)

For a change of pace—and geography—here is a 32-step routine from **Win Acton** of Vancouver, Washington, that is guaranteed to make your life easier if you insist on expressing length measurements in feet and inches with binary fractions, instead of succumbing to metrication.

Input format is ffii.nndd, where ff is integral feet, ii is integral inches, nn is the fraction's numerator, and dd is its denominator. For example, 6 feet $3^5/_8$ inches is keyed 603.0508. The entire fraction may be zero but a zero denominator following a non-zero numerator won't work. No storage registers are used. The previous value in X is saved in Y (and Z and T), so chain calculations are facilitated.

001	*LBLC	017	*LBL0
002	INT	018	+
003	LSTX	019	EEX
004	FRC	020	2
005	X=0?	021	÷
006	GTO0	022	INT
007	EEX	023	LSTX
008	2	024	FRC
009	x	025	EEX
010	INT	026	2
011	LSTX	027	x
012	FRC	028	1
013	÷	029	2
014	EEX	030	÷
015	2	031	+
016	÷	032	RTN

(Continued)

HP-67 and HP-97 Accessories

Description	Model Number	Price
DC Recharger/Adapter (HP-67)	82054A★	$ 35.00
Reserve Power Pack (HP-67)	82004A★	$ 20.00
Reserve Power Pack (HP-97)	82037A	$ 35.00
Security Cradle (HP-67)	82015A★	$ 30.00
Security Cable (HP-97)	82044A	$ 10.00
Hard Leather Case (HP-67)	82016A★	$ 35.00
AC Recharger/Adapter (HP-67) 110/220 Vac., Switchable	82002A★	$ 20.00
(HP-97) 110 Vac.	82059A	$ 12.50
(HP-97) Euro. 220 Vac.	82066A	$ 16.00
Battery Pack (HP-67)	82001A★	$ 10.00
Battery Pack (HP-97)	82033A	$ 18.00
Soft Case, Black Leather (HP-67)	82017A★	$ 10.00
Soft Case, Synthetic (HP-67)	82053A★	$ 7.00
Soft Case, Synthetic (HP-97)	82035A	$ 10.00
Thermal Printing Paper, 6 rolls (HP-97)	82045A	$ 6.00
Program Pad (HP-67 & HP-97)	00097-13154★	$ 4.00
3 Program Card Holders (HP-67/97)	00097-13142★	$ 10.00
Blank Program Cards (HP-67/97) 40-Card Pack with Holder	00097-13141★	$ 20.00
120-Card Pack with Holders	00097-13143★	$ 45.00
1000-Card Pac	00097-13206★	$195.00

HP-67/97 Application Pacs Each $35.00

HP-67 Standard Pac	00067-13101	Games Pac	00097-13185
HP-97 Standard Pac	00097-13101	Math Pac I	00097-13121
Business Decisions Pac	00097-13144	ME Pac I	00097-13155
Civil Engineering Pac	00097-13195	Navigation Pac	00097-13205
Clinical Lab & Nuclear Medicine Pac	00097-13165	Stat Pac I	00097-13111
EE Pac I	00097-13131	Survey Pac I	00097-13175

HP-67/97 Users' Library Solutions Books Each $10.00

BUSINESS:		**MEDICAL:**	
Options/Technical Stock Analysis	00097-14009	Medical Practitioner	00097-14005
Portfolio Management/Bonds & Notes	00097-14010	~~Anesthesia~~	~~00097-14019~~✱
Real Estate Investments	00097-14012	Cardiac	00097-14018
Taxes **(UPDATE FOR 1978)**	00097-14004	Pulmonary	00097-14037
Home Construction Estimating	00097-14033	**PHYSICAL/LIFE SCIENCES:**	
Marketing/Sales	00097-14032	Chemistry	00097-14006
Home Management	00097-14031	Optics	00097-14016
Small Business	00097-14039	Physics	00097-14015
ENGINEERING:		~~Earth Sciences~~	~~00097-14017★~~✱
Antennas	00097-14021	Energy Conservation	00097-14029
Butterworth & Chebyshev Filters	00097-14003	Space Science	00097-14028
Thermal & Transport Sciences	00097-14023	Forestry	00097-14034
EE (Lab)	00097-14025	Biology	00097-14040
Industrial Engineering	00097-14035	**OTHER:**	
~~Aeronautical Engineering~~	~~00097-14036★~~✱	Games	00097-14013
Beams & Columns	00097-14027	Games of Chance	00097-14038
Control Systems	00097-14026	Aircraft Operation	00097-14001
COMPUTATION:		Avigation	00097-14002
High-Level Math	00097-14011	Calendars	00097-14024
Test Statistics	00097-14008	Photo Dark Room	00097-14022
Geometry	00097-14007	COGO/Surveying	00097-14020
Reliability/Quality Assurance	00097-14030	~~Astrology~~	~~00097-14014★~~✱

★ Also usable on the HP-65.

✱ *No longer available.*

HEWLETT-PACKARD ORDER FORM

HP CODE NO. 300ZC

Model Number or Program Number	Description	Qty.	Price Each	Total Price

Please enter my one-year subscription	☐ For US, Puerto Rico or Virgin Islands	**$15.00**	
to the HP-67/97 Users' Library	☐ For countries outside US, PR, V.I., and Europe	**$20.00**	

SHIPPING CHARGE★ _____

STATE AND LOCAL TAX _____

TOTAL PRICE★★ _____

You may order any of the HP-67 and HP-97 accessories and software shown by calling our toll-free number, 800-648-4711 (in Nevada 800-992-5710), and asking for the nearest Hewlett-Packard Accessory Dealer. (Toll-free number not available in Alaska or Hawaii.)
Or you may use the convenient order form on the reverse side and mail to: Hewlett-Packard Co., 1000 NE Circle Blvd., Corvallis, Oregon 97330.

NOTE: All Users' Library Programs mentioned in Key Notes can be ordered on this order form. Simply note the appropriate Program Number and fill in the description and price. All programs with pre-recorded program cards are available at $5.00 each.

Hewlett-Packard reserves the right to make changes in materials, specifications, and prices without notice.

Orders cannot be shipped to any European countries.

★ For all orders shipped outside US please add $5.00 shipping charge.
★★ Paid in US dollars drawn on a US bank.

PAYMENT OPTION (Please check one)

☐ **CASH PAYMENT.** Check or money order enclosed in US dollars drawn on US bank. (Please be sure to include your state and local sales taxes.)

☐ **BILL MY CREDIT CARD.** To use our seven-day 24-hour charge-by-phone service, call us Toll Free at (800) 648-4711, Ext. 1000; in Nevada call (800) 992-5710. Toll-free number not available in Alaska or Hawaii. You can use the deferred payment plan available through your Master Charge or Visa/BankAmericard account simply by filling in the information below.

CHARGE TO: ☐ American Express ☐ VISA/BankAmericard ☐ Master Charge—Include 4-digit number appearing on card just above your name:

Your credit card account will be billed when your order is shipped.

MY COMPLETE CARD NO. IS: EXPIRATION DATA: _____

If using a credit card and the billing address does not correspond to that shown for shipping, please indicate correct billing address.

_____
NAME APPEARING ON CREDIT CARD

_____
STREET ADDRESS city, state, zip

_____
SIGNATURE TITLE BUSINESS PHONE NO. (Include area code)

SHIP TO: _____
NAME

_____
ADDRESS

_____
CITY STATE ZIP

To find the volume of a box 3 feet $0\text{-}^{47}/_{64}$ inch wide, 4 feet $11\text{-}^{9}/_{16}$ inches long, and $9\text{-}^{3}/_{4}$ inches high:

Key	Press	Display
	DSP3	0.000
300.4764	C	3.061
411.0916	C,x	15.194
9.034	C,x	12.345

[illegible] registers? Here it is, thanks to Curtis J. Caldwell of Jacksonville, Florida.

Occasionally it is desirable to perform the HP-67 "h REG" function slowly to allow the copying of register contents, or to interrupt such a review by halting execution while retaining the ability to resume the review from that point, retaining the sequence compatability with the "h REG" function for rapid register review. The I register is used to step through memory, and thus does not assume the importance which it might when the "h REG" function is used. Three variations are:

STEP	1	2	3	Remarks
001	*LBLA	*LBLB	*LBLC	
002	CLX	CLX	CLX	Zero the I register.
003	STOI	STOI	STOI	
004	*LBL1	*LBL2	*LBL3	
005	1	1	1	Skip secondary registers.
006	0	0	0	
007	RCLI	RCLI	RCLI	
008	X=Y?	X=Y?	X=Y?	
009	+	+	+	
010	STOI	STOI	STOI	
011	2	PSE	PSE	Begin display routine, testing for the I register as the indicator to quit (#25), displaying the register number about to be reviewed, and then displaying the contents of the register of interest.
012	5	RCL i	RCL i	
013	X≷Y	PRTX	X=Y?	
014	X=Y?	ISZI	R/S	
015	RTN	GTO2	PRTX	
016	PSE		ISZI	
017	RCL i		GTO3	
018	PRTX			
019	ISZI			
020	GTO1			

Routine #1 is useful when a return is required from the output routine to continue some other task. It is the most suitable of the three for programs created for use by non-programmers and casual users in that no special user instructions are required and no "unnerving" messages are displayed.

Routine #2 is useful when data output is the last program task, when program space is critical, and the user is not distrubed by halting with "Error."

Routine #3 is a substitute for routine #2 when the user is uncomfortable with an "Error" message. The user is burdened with pushing "R/S" whenever the register number equals its contents.

I would be interested in other routines that perform this function; those that do not use storage registers, those that require fewer program steps, or those that use not more than one extra step yet have a shorter execution time.

(Okay, Mr. Caldwell, we'll be the first to respond. Try this routine. Maybe you'll see more in the next issue. Ed.)

001	*LBLA	006	→P
002	3	007	X≷Y
003	6	008	X<0?
004	0	009	+
005	→R	010	RTN

[illegible] new approach [illegible] Guy R. Morris [illegible]

Enclosed is a very simple technique for conversion from Spherical to Rectangular coordinate systems and vice versa. The simplicity of the technique lies in its use of the calculator's built-in conversion from Polar to Rectangular and Rectangular to Polar Functions. Anyone familiar with the coordinate transformation equations (shown below) for both systems will realize the savings in keystrokes and memory. The technique is particularly useful for quick conversions from the key pad and excellent for gaming and scientific programs.

Spherical to Rectangular

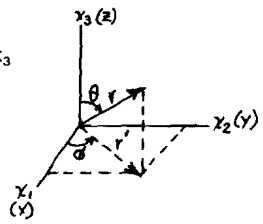
$x_1 = r \sin\theta \cos\phi$
$x_2 = r \sin\theta \sin\phi$
$x_3 = r \cos\theta$

75.00 ENT↑ → θ
15.00 → R → r
3.88 *** → x_3
CLX
90.00 x ≷ y → ϕ
14.49 *** → r′
→ R
0.00 *** → x_1
x ≷ y
14.49 *** → x_2

Rectangular to Spherical

$r = \sqrt{x^2 + y^2 + z^2}$
$\phi = \arctan y/x$
$\theta = \arctan \sqrt{x^2 + y^2}/z$

2.00 ENT↑ → x_2
12.00 → P → x_1
12.17 *** → r′
x ≷ y
9.46 *** → ϕ
CLX
15.00 → P → x_3
19.31 *** → r
x ≷ y
39.04 *** → θ
for r = 15
θ = 75
ϕ = 90



the rectangular values are:
x_1 = 0.00
x_2 = 14.49
x_3 = 3.88
(rounded to two places).

[illegible] navigation

In navigation problems it is desirable to have solution angles bounded between 000 and 360. It is not uncommon for angles to be bounded by −540 to 720. The following routine performs the desired conversion where the original angle is conditioned only by an upper bound of 720. Programming considerations were: (1) data storage space is critical, and (2) run time is not as important as saving program storage space.

STEP	1	Remarks
001	*LBLA	
002	3	Establish bounding constant.
003	6	
004	0	
005	−	Ensure solution angle is below upper bound of 360.
006	*LBL5	See if solution angle is above lower bound of 000.
007	X>0?	LST X contains 360.
008	RTN	
009	LSTX	
010	+	
011	GTO5	

(Continued)

Now, somewhat in the same vein, here is a nice trick from **Ward W. Miller** of Lagrangeville, New York.

While calculating some celestial navigation problems recently (with DSP 4), I found that 7.3499 hours were converted via g → H.MS to 7.2060 (7h 20m 60s) when I wanted 7.2100 (7h 21m 00s) displayed. Well, the following sequence produces the expected rounded display of hours (or degrees), minutes, and seconds, assuming DSP4 and that hours (or degrees) are originally in decimal forms.

KEY ENTRY	DISPLAY	MEANING
	7.3499	7.3499h
g → H.MS	7.2060	7h 20m 60s
f RND	7.2060	7h 20m 60s
f H←	7.3500	7.3500h
g → H.MS	7.2100	7h 21m 00s

(Thanks, Mr. Miller, especially for the sentence I didn't *print; those were very kind words about KEY NOTES. Ed.)*

Next, let's see what **Dr. Hans Stocklmair** is up to over in Klagenfurt, Australia.

As a contribution to the "25 Words or Less" column, I would like to suggest the following solution:

In many business programs, it proves useful to get results displayed/printed optionally

(a) In currency units;
(b) In thousands of currency units (divided by 10^3); and
(c) In millions of currency units (divided by 10^6).

This can easily be achieved by inserting the three steps RCL 0, 10^x, ÷ at the end of the calculation, with the exponents 0, 3, and 6 alternately set in R_0. All you need, then, is a short and comfortable routine to change the contents of R_0, perpetually, just by pressing a single program key (similar to the usual way of setting/clearing the print flag).

There are many ways to achieve this result; the shortest routines I have found are given below. Version I changes the contents of R_0 in the seqquence 0 → 3 → 6→ 0 every time key "C" is pressed, while version II works in the sequence 0 → 6 → 3 → 0 (which may be of greater comfort in case the 10^6 transformation is more frequently used).

VERSION I

```
001  *LBLc
002      3
003   ST+0
004      9
005   RCL0
006   X=Y?
007    CLX
008   STO0
009    RTN
```

VERSION II

```
001  *LBLc
002      6
003   ST+0
004   RCL0
005   X≠Y?
006    RTN
007      9
008   ST-0
009   RCL0
010    RTN
```

Our next contributor has not only two inputs for this column but also a short article elsewhere! Congratulations to **Harden Schaeffer** of Goldthwaite, Texas.

This routine is for plotting, with 13-increment resolution, a curve on the HP-97 printer. It is called by label 0 and uses three labels. It also uses the I-register and may destroy the contents of the stack.

```
001  *LBL0   011   DSP;
002     FIX  012   PRTX
003       1  013   RTN
004       0  014  *LBL1
005    X≠Y?  015   CLX
006    GTO1  016     3
007    X≷Y   017     -
008  *LBL2   018   SCI
009    STOI  019   GTO2
010       0  020   R/S
```

These two routines, used together, permit storing and listing 10 x,y data pairs per card, with each card being numbered automatically. The first x-value is stored in R_9, the first y-value in R_{59} and so on, down to the 10th x-value in R_0 and the 10th y-value in R_{50}. The number of each card in the series is stored in R_A. Registers B thru E are available to store the data. The y-value of each pair is listed first, then the x-value. (To reverse this, move the first P ⇌ S instruction in the listing routine from step 12 to right after step 17.)

TO STORE DATA: Load the storing routine and press [A]. Key in the y-value and press [ENTER↑], then key in the x-value and press [R/S]. Keep entering the data pairs and pressing [R/S] until the display indicates **Crd,** then "write" a data card. Enter more data pairs and keep writing cards until you run out of data pairs . . . or blank cards.

```
001  *LBLA   013   R/S
002       2  014   STO;
003       0  015   P≷S
004    STOI  016   X≷Y
005    ISZ;  017   STO;
006       1  018   P≷S
007       0  019   F0?
008    STOI  020   GTO0
009  *LBL0   021   WDTA
010     CF0  022   GTOA
011    DSZI  023   R/S
012     SF0
```

TO LIST DATA: Load the listing routine, read a data card, then press [A]. The calculator will print the number of the card, then 10 data pairs, then pause with a flashing display until you read in another data card, at which time it will repeat the process. Do not press a keyboard key during pauses; this will cause the calculator to list the last card again.

```
001  *LBLA 011    SF0 021    SPC
002    RCLA 012    P≷S 022    SPC
003    PRTX 013  RCL; 023   *LBL1
004     SPC 014   PRTX 024    CF3
005       1 015    P≷S 025    PSE
006       0 016  RCL; 026    F3?
007    STOI 017   PRTX 027   GTOA
008  *LBL0 018    SPC 028   GTO1
009     CF0 019    F0? 029    R/S
010    DSZI 020   GTO0
```

Because the stock market is always big news, here is a neat conversion trick from **Fortson H. Williams** of Cincinnati, Ohio.

For stock market calculations, the following routine (label A) allows entry of the stock prices in eighths, which are then converted to decimals. This can save many keystrokes if there is much of this data to be entered.

```
001  *LBLA   006       8
002    STO1  007       ÷
003     FRC  008    RCL1
004    ST-1  009       +
005       .  010     RTN
```

Label B allows entry of two stock price quotations in eighths, such as the high and the low, converts them both to decimals, and averages them. This routine is a combined eighths-to-decimals conversion which is much shorter than if each of the two prices had been converted separately by means of the routine of label A.

```
001  *LBLB   009       .
002    STO1  010       8
003     FRC  011       ÷
004    X≷Y   012    RCL1
005    ST+1  013       +
006     FRC  014       2
007       +  015       ÷
008    ST-1  016     RTN
```

Do you remember **Nicomachus**? If so, you must have studied mathematics pretty well. If not, see the note following this letter from **Peter Luschny** of München (Munich), Germany.

Nicomachus vs Euclid. There is a beautiful, simple algorithm—thanks to Nicomachus—for calculating the greatest common divisor (gcd). He stated that the general method is to "continually take the less from the greater until reaching a unit or two equal numbers." This algorithm is not as fast as Euclid's, but is is considerably simpler (no division, no multiplication).

The following eight-step routine uses only one label and no storage registers. The two values for which the gcd is to be found must be in the X and Y stack registers.

```
001  *LBLA   005     ABS
002       -  006    X≠Y?
003    LSTX  007    GTOA
004    X≷Y   008     R/S
```

It is easy to extend the routine so that it will also find the least common multiple (lcm). Again, no storage registers are used.

```
001  *LBLA   010    LSTX
002    ENT↑  011    X≷Y
003    ENT↑  012     ABS
004      R↑  013    X≠Y?
005       x  014    GTO0
006    LSTX  015      R↓
007      R↑  016       ÷
008  *LBL0   017      R↑
009       -  018     R/S
```

The gcd and lcm are in the X and Y stack registers, respectively.

Sometimes it might be useful not to destroy the input. In such cases, use the following routine:

```
001  *LBLA  014   R↓
002   ENT↑  015   R↓
003   ENT↑  016  ENT↑
004    R↑   017   R↓
005   ENT↑  018  X≷Y
006    R↓   019   ×
007  *LBLB  020  LSTX
008    -    021   R↓
009   LSTX  022  X≷Y
010   X≷Y   023   ÷
011   ABS   024  LSTX
012  X≠Y?   025   R/S
013   GTOB
```

After running this routine the stack registers contain:

T = a
Z = b
Y = lcm (a, b)
X = gcd (a, b)

and here is a sample:

```
     40902.00   T
     24140.00   Z
  29040420.00   Y
        34.00   X
```

(Nicomachus of Gerasa was a Neo-Pythagorean philosopher and mathematician who wrote Arithmetike eisagoge, *which was translated in 1926 to* Introduction to Arithmetic, *and it was the first work to treat arithmetic as a discipline independent of geometry. Considered a standard authority for 1000 years, it sets out the elementary theory and properties of numbers and contains the earliest known Greek multiplication table. In his book, numbers are no longer denoted by lines, as in Euclid, but are written in the Arabic numerals. General principles are stated with particular numbers taken as illustrations. A Latin translation by Apuleius of Madaurus—c. AD 125—is lost, but Boethius' version survived and was used as a school book up to the Rennaissance. Ed.)*

Let's move now to Mattstetten, Switzerland, for an input from **Hans Marbet.**

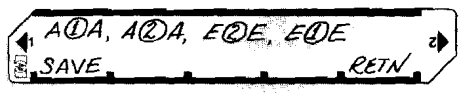
I have noted with interest the "Ersatz Memory" articles and follow-on comments in past issues of KEY NOTES. And I agree with the comments that the method is only applicable *when any program in execution is terminated.* However, I hardly can believe that the battery run-out time allows the user to consult the program description. Thus, I should like to present an alternative and easier-to-handle program which, furthermore, does the same job with less writing/reading of cards.

The 62-step program saves all register and stack contents on *a single magnetic card.*

Save data contents as follows:
1. Load the program.
2. Press [A], enter side 1 of data card, press [A].
3. Press [A], enter side 2 of data card, press [A].
4. Turn off the calculator.

Return the data as follows:
1. Turn on the calculator.
2. Load the program.
3. Press [E], enter side 2 of data card, press [E].
4. Press [E], enter side 1 of data card, press [E].

Caution: Ignore any **Crd** displays that might occur. Just follow the straightforward instructions above, pressing the [A] or [E] keys only when any current section of the program is terminated. The [A] or [E] keystrokes before and after each card-entering can easily be remembered by labeling the program card as follows:




```
001  *LBLA  022   RCL4  043   P≷S
002   WDTA  023    +    044   STO0
003   R/S   024   RCL5  045   P≷S
004  *LBLA  025    +    046   R/S
005   STOA  026   RCL6  047  *LBLE
006    R↓   027    +    048   P≷S
007   STOB  028   RCL7  049   RCL0
008    R↓   029    +    050   ST-0
009   STOC  030   RCL8  051   X=0?
010    R↓   031    +    052   R/S
011   STOD  032   RCL9  053  *LBLE
012   LSTX  033    +    054    0
013   STOE  034   X=0?  055   RCLE
014   P≷S   035   R/S   056    +
015   RCL0  036  *LBLA  057   RCLD
016   RCL1  037   WDTA  058   RCLC
017    +    038   R/S   059   RCLB
018   RCL2  039  *LBLA  060   RCLA
019    +    040   R/S   061   R/S
020   RCL3  041  *LBLE  062  *LBLE
021    +    042    2    063   R/S
```

(Since KEY NOTES is distributed in Europe at least one month later than in the U.S., Mr. Marbet did not see James P. H. Hirst's solution to "Ersatz Memory" in Vol. 3 No. 1 before he wrote the above program. Nonetheless, I thought it worthwhile to add Mr. Marbet's contribution to the "Ersatz Memory" puzzle. Ed.)

Use the Right Paper!

Lately, some of the HP-97 (and HP-91 and HP-92) calculators that have been sent back to our Service Department for repair have had the wrong thermal paper in them. Please remember that **HP thermal paper is the only approved paper for your calculator.** (Refer to page 285 in the *HP-97 Owner's Handbook.*) **Other types of thermal paper might work in your HP calculator, but those papers will cause premature printhead failure, and possibly cause repairs at your expense.**

Editorial

That *ENTER greater than EQUALS* T-shirt on **Gary M. Tenzer** on the cover of the last issue caused quite a few inquiries. So I tried to find the original source. The shirt was originated by **Jim Virgin** of the Stanford University Bookstore and was made by Shoreline Sportswear of Santa Barbara, California. However, I am sorry to report that they are no longer available.

I received a letter last month from **Robert Schild, Jr.** of Dover AFB, Delaware, wherein he suggested we add the KEY NOTES issue numbers at the bottom edge of pages so you can more easily locate specific material when paging through bound copies of the newsletter. So you can thank him for persuading me to do it.

Mr. Schild also pointed out a small error that not many people saw in the last issue. Refer to **Neal Neuburger's** routine near the bottom of the first column on page 10 of Vol. 3 No. 1. When running the example, "52" should be added to the third line, after ENTER.

Mr. Schild and many others have asked why we no longer punch holes in KEY NOTES so that it can be inserted in ring binders. Well, as the newsletter has grown and grown many times, it has become necessary to print it by more economical means. That is why you do not see staples in it and three holes drilled in the binding edge. We presently run the newsletter on a very large web press that does the entire process in one pass, and to drill holes as a separate, manual, process would cause a heavy additional expense. Instead of doing that, we put the money into extra pages for you, and we feel sure you'd rather have more information than three small holes in each copy.

If you've written to me and haven't received an answer, don't give up hope! Mail has been very heavy lately, and the situations that delayed this issue have put an unusual strain on your editor's available time. Letters to the editor should be addressed to:

Henry Horn, Editor
HP KEY NOTES
Hewlett-Packard Co.
1000 N.E. Circle Boulevard
Corvallis, Oregon 97330

We cannot guarantee a reply to *every* letter, but we will guarantee that every letter received will be read by the editor, and as many as possible will be answered either in KEY NOTES or in a personal response. Please be sure to put your return address on the face of your letter. Letters sometimes get separated from envelopes!

And, as a final word, don't forget to notify us of address changes. You won't be able to read your KEY NOTES if we don't know where to send it!

Wood You Help a Forester?

You all know that coal and oil are not renewable energy resources. But wood is. So we decided to print the following letter and see what our far-flung audience can do to help —maybe even from other areas of the world.

Gentlemen:

I am a forester with an HP-97 and am looking for forestry programs; that is, in addition to those listed in the HP-67/97 Users' Library. It seems, also, that most of your past contributors are westerners, and I hope to find some southern "67/97" foresters.

Please publish this letter and my address in the next KEY NOTES. This may lead to a greatly enriched forestry section of the Library.

Sincerely,

Colin Bagwell, R.F./A.C.F.

1601 Sun Valley Road, Huntsville, AL 35801

(Glad to help, Mr. Bagwell. Let us know what response you get from our readers. Ed.)

When $2.00 Equals $20.00!

In appendix B of *Catalog Addendum 3*, dated November 1978, we had a significant typographical error. The President wants us to keep prices down, but *this* price was ridiculous!

Thanks to **Robert Schild, Jr.** (Dover AFB, Delaware), who notified us of the mistake, we can inform you about it. On the page of "Supplies & Accessories for Your Programming Needs," the 00097-13141 40-Card Pac with Holder should be $20.00* instead of $2.00

On that same page, mark these other corrections:

1. On the first line, mark "(HP-67)" after "DC Recharger/Adapter."
2. The correct model number for the HP-97 110 Vac Recharger/Adapter should be 82059A (not 82058A).
3. On the fourth to last line, the correct number for "3 Program Card Holders" is 00097-13142 (not -13143).

Also, although it is not stated on the page, the asterisk after the model number indicates the supply or accessory can be used for the HP-65.

** U.S. dollars. See not at bottom edge of cover.*

"SST-ing" Thru Subroutines

Works for HP 19c also

Here is a rather clever tip you might not have thought of just because it never occurred to you. The following letter is from **Harden Schaeffer** of Goldthwaite, Texas.

This small item might be of interest to readers of KEY NOTES. It is a technique I discovered for single-stepping through a subroutine while debugging a program on my HP-97.

When, while single-stepping, a GSB x instruction is encountered, do the following:

1. While holding down the SST key, press the GTO key (or any other key in the same row).
2. While holding down the GTO key, release the SST key.
3. Release the GTO key.
4. Press GTO x.
5. The calculator will now single-step through the subroutine and will return to the proper place when it encounters a RTN instruction.

I don't know for sure if this will work with the HP-67, since its keyboard is electrically very different from the wiring of the HP-97's keyboard. And while the above procedure appears rather complicated "on paper," it's really quite simple to do in actual practice.

This procedure was developed during the course of some very helpful and enjoyable correspondence with one of the people in your Service Department.

Better Late Than Never!

If your KEY NOTES arrived late, we apologize. Because of recent strikes and some other setbacks, there was a delay in both printing and distribution. However, we know you enjoy getting the newsletter regularly, so we want to assure you that the August issue *will* be on schedule again.

Base Conversion Routine

Although a Library program will do the same thing the following routine will do, we think this contribution from **Cass R. Lewart** (Holmdel, New Jersey) is well worth printing in KEY NOTES.

Here is my program that converts an integer argument from *any* base to *any* base in only 49 steps. I saw a description of a similar program in the Users' Library (00409D) but it required 135 steps. I think that many KEY NOTES readers will be interested in this one. For example:

$(013\ 015\ 199)_{215} = (?)_{16}$

Argument ENTER↑ Old base ENTER↑ New base A.

13015199 ENTER↑ 215 ENTER↑ 16 A.

Answer: 90308113 = 938BD hex.

001	*LBL A	026	ENT↑
002	STO 0	027	INT
003	R↓	028	STO 3
004	ENT↑	029	−
005	GSB 3	030	RCL 1
006	GSB 0	031	×
007	RCL 0	032	RCL 5
008	GSB 3	033	×
009	RCL 0	034	ST+ 4
010	*LBL 0	035	RCL 2
011	STO 1	036	ST× 5
012	R↓	037	GTO 1
013	STO 2	038	*LBL 2
014	R↓	039	RCL 4
015	STO 3	040	RTN
016	0	041	*LBL 3
017	STO 4	042	1
018	1	043	−
019	STO 5	044	LOG
020	*LBL 1	045	INT
021	RCL 3	046	1
022	X=0?	047	+
023	GTO 2	048	10^x
024	RCL 1	049	RTN
025	÷		

HP KEY NOTES

May 1979 Vol. 3 No. 2

Programming and operating tips, answers to questions, and information about new programs and developments. Published periodically for owners of Hewlett-Packard fully programmable personal calculators. *Reader comments or contributions are welcomed. Please send them to one of the following addresses.*

Hewlett-Packard Company
Users' Library
1000 N.E. Circle Boulevard
Corvallis, Oregon 97330 USA

Hewlett-Packard SA
USERS' CLUB EUROPE
7, Rue du Bois-du Lan
P.O. Box, CH-1217 Meyrin 2
Geneva-Switzerland

Address Correction Requested
Return Postage Guaranteed

BULK RATE
U.S. POSTAGE
PAID
PERMIT NO. 814
PORTLAND, OR

Printed in U.S.A.