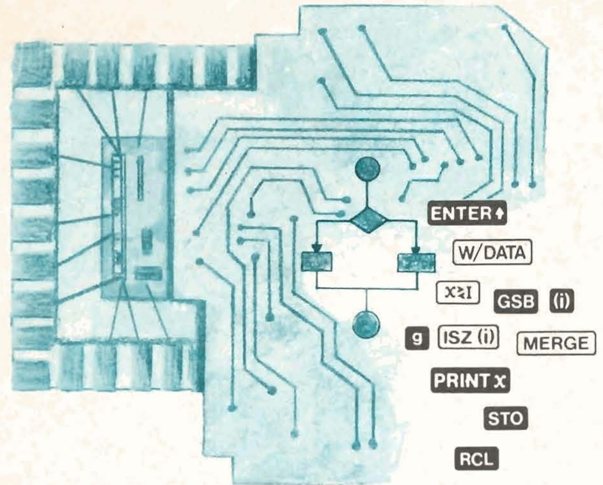


Featuring, this issue:

HEWLETT PACKARD

September-December 1982 Vol. 6 No. 4
ISSN 0730-2037

HP Key Notes

Published Quarterly   By Subscription: $5/yr. $2.00

HP Has the Right Tool for the Job

On November 3, Hewlett-Packard began a major media campaign to promote its full range of personal computation products. In such media as *The Wall Street Journal, Business Week, U.S. News & World Report, Time, Fortune,* and *Newsweek,* you can see advertisements of the new products now available for every facet and level of personal computation.

Because the Corvallis Facility is the "home" of the Series 10, 40, 70, and 80 families of personal computation products, we are proud to present a photo, used in the above-mentioned campaign, of HP's *complete* range of personal computation products to show you how prominently *your* personal machines figure in this splendid line-up.

In the foregound are two brand-new HP products: on the left is the HP 120 Personal Office Computer and on the right is the HP Series 200 Model 16 Personal Technical Computer, both produced by other HP Divisions. Information about these and other Series 100 and 200 computers can be obtained from your local HP Sales Office.

In the background, left to right, are the Series 80 Personal Computers from Corvallis: the *portable* HP-85, the *powerful* HP-87, and the *personal-priced* HP-86. And if you don't recognize it, that's the new and wonderful HP-75 at right, center; it's in a class by itself. If you haven't seen it yet, you are missing a real contribution to the world of personal computation.

Last, but by no means least — at left, center — are the super-portable, personal Series 10 calculators and the forerunner of handheld personal computers, the venerable HP-41.

If you can't locate an HP Sales Office near you, call our toll-free number 800-547-3400 (U.S. only). (In Oregon, Alaska, and Hawaii, call 503-758-1010.) TTY users with hearing or speech impairments, please dial 503-758-5566.

NEW...For You, Inside...

It seems fitting that we end this year with a lot of new things for you, just in time for your Christmas shopping. And you will notice right away that this issue is thicker and heavier than usual!

NEW HP-10C CALCULATOR

Turn to page 3 to see the latest in HP low-cost, high-quality, thin-line calculators. Yet, despite its low price, it is programmable, plus lots more.

NEW HP-41 GRAPHICS CAPABILITY

Turn to page 4 for another surprise from Hewlett-Packard. You'll find the new HP 7470A Plotter and the new HP 82184A Plotter Module. Combine these with your HP-IL system and you have a low-cost graphics system that is very, very hard to beat!

NEW HP-12C TRAINING GUIDE

Turn to page 9 to see the newest in aids for anyone who cannot unravel the mysteries and intricacies of financial functions and transactions. And with the present state of interest rates, mortgages, and investments, this new book would appear to be the answer to many present-day problems.

NEW HP-12C REAL ESTATE HANDBOOK

Turn to page 20 and you will find the new *HP-12C Real Estate Applications Handbook* that we ran off the presses just in time for Christmas. This handy handbook is not a dissertation in jargon and unintelligible formulas and algorithms. It *shows* you by example, how you can make decisions that will make your career profitable instead of profitless. If you are in any facet of real estate, don't miss this!

All prices in this newsletter are suggested retail prices excluding applicable state and local taxes—Continental U.S.A., Alaska, and Hawaii.

Library Corner

CORVALLIS USERS' LIBRARY NEWS

The new Corvallis Users' Library HP-41 "cassette-based" submittal contest is still underway. (And if you were a member of the Corvallis Users' Library, you would have heard about it before now!) The prospect of winning an HP-75C Portable Computer should convince even the most reluctant programmer to take pen in hand. Five Grand Prizes of HP-75C Portable Computers, plus Ten Series 10 (HP-11C/12C/15C/16C) calculators will be awarded to the authors of the fifteen programs judged most outstanding by our review staff. The contest rules are consistent with previous contests, except these programs must be HP-41/HP-IL cassette-based. This contest ends December 31, 1982, but don't wait until the last days to submit your entry. Send your entry early enough so that if the review staff should have to return your program for modification, you will have time to resubmit your program before the contest deadline.

You probably notice that only the June winners of our last Corvallis Library contest appear in this issue. Our review staff is working "day and night" but the response to our contest was tremendous; so . . . next issue will contain the July and August winners.

HP-41 SOLUTIONS BOOK UPDATE

Here's the rundown on what is new, soon to be in stock, and upcoming.

00041-90443 HP-41 Games II — in stock and shipping (and extremely popular!). $12.50*

00041-90441 Structural Design — available in December 1982. $35.00*

00041-90442 HP-41 Matrices and Complex Numbers — watch for announcement in next issue.

HP-75 USERS' LIBRARY NEWS (CORVALLIS)

With the introduction of the HP-75C also comes both an HP-75 Users' Library and eleven HP-75 Solutions Books. The Solutions Books are $35.00* each (including magnetic cards), and they are available in the following applications.

Math I	- 00075-13003
Math II	- 00075-13004
Math III	- 00075-13005
Games I	- 00075-13006
Games II	- 00075-13007
Electronics	- 00075-13008
Finance	- 00075-13009
Real Estate	- 00075-13010
Statistics	- 00075-13011
Test Statistics	- 00075-13012
I/O Utilities	- 00075-13013

FREE MEMBERSHIP — FOR A WHILE . . .

For a limited time, a complimentary HP-75 Corvallis Users' Library membership will be offered with the purchase of the HP-75. (At present, this offer is limited to only the U.S. It will probably be extended to include Europe and maybe elsewhere, but we will let you know in the next issue.) The membership includes a *Catalog*, documentation guide, submittal forms, and the program "EASYMEMO," which can be used to format your letters or memos. We encourage all new HP-75C owners to help launch our new Library by documenting any programs that may be of interest to others and by sending them to:

The HP-75 Corvallis Users' Library
1000 N.E. Circle Blvd.
Corvallis, OR 97330

***BEFORE YOU ORDER** any of the Solutions Books mentioned above, make sure you first check with your local authorized HP Dealer. Remember: All direct orders (whether to Corvallis or to the Corporate Parts Center) must include a $3.50 handling charge *per order.* If you live outside the U.S. and Canada, you should check with your local HP Sales Office or authorized HP Dealer for availability of these books. The above ordering and pricing information does NOT apply outside the U.S. and Canada. Also, all payments must be in U.S. dollars.

WINNING PROGRAMS

The June winners of the 1982 Users' Library Submittal Contest are featured below. Although the contest ended in August, as of the press date for this issue of KEY NOTES, the review staff has chosen the winners only through June. The contest brought in such a tremendous number of program submittals, we will have to feature the winners for July and August in the next issue of KEY NOTES.

The winning programs for June cover a broad range of interests. The authors of these programs each received a fine HP product for their fine contributions to the Users' Library.

JUNE WINNERS

(41) Electric Rate Analysis II #01983C (Price: $12*)

This is a general purpose electric rate analysis program to compare two rates. It supports Hopkinson and Wright type demand rates with up to five energy blocks, up to three demand blocks, a customer charge, and monthly minimum. The two rates need not be of the same type. Both rates and the results of comparisons are printed. *Required accessories: 3 Memory Modules and Printer.* (734 lines, 1515 bytes, 30 pages)

Author: **John Brown**
Boise, Idaho

(97) Prediction of Seasonal Heating Costs With a Heat Pump #04836D (Price $16*)

This set of programs uses weather data, heat pump capacity specifications, and building loss relations to calculate seasonal heating energy requirements and estimated costs. Comparison is made to similar installations heated with gas and oil. Analysis of life-cycle energy savings versus investment is included. *Required accessories: HP Library programs #00252D or #00564D.* (440 lines, 58 pages)

Author: **James Baskerville**
McLean, Virginia

(41) Design and Rating of Packed Columns #01962C (Price: $10*)

Using equations fit to pressure drop curves published by the Norton Company, the design diameter is calculated for a packed column at flooding or six other user-specified pressure drops. Given diameter, the program will rate the column (calculate the pressure drop). Properties required are: densities, liquid viscosity, flow rates, and packing type. *Required accessories: Two Memory Modules.* (447 lines, 1052 bytes, 22 pages)

Author: **Robert Wooley**
Midland, Michigan

(41) Pharmacokinetic Parameters From Serum Drug Concentrations #01878C (Price: $16*)

This program determines k and V from two drug levels, or k from an assumed V and one drug level, during maintenance dosing with any of four dosing schedules: intermittent, fixed interval; intermittent, two fixed intervals; intermittent, non-uniform does, non-uniform interval; continuous intravenous infusion/sustained release. A loading dose and/or one additional level taken before maintenance dosing begins can be taken into account for additional flexibility. Steady state concentrations are projected when dosage regimen and pharmacokinetic parameters are known. *Required accessories: Card Reader, Quad Memory Module.* (715 lines, 1585 bytes, 41 pages)

Author: **E. Maurice Jones**
Charleston, South Carolina

(41) Flow Computations For Various Open Channel Configurations #01957C (Price: $14*)

This program computes flow, given normal depth, or normal depth given flow,, and critical depth, for rectangular, triangular, trapeziodal, circular, or parabolic channels. Standard output is normal depth, top width, and flow. The user may select additional outputs including area, wetted perimeter, hydraulic radius, and/or average velocity. The program accepts S.I. or U.S. units. *Required accessories: 3 Memory Modules.* (554 lines, 1442 bytes, 35 pages)

Author: **Martin Hanson**
Park Falls, Wisconsin

(41) Time Domain Analysis for Nonlinear Networks #01909C (Price: $12)

This Calculator-Aided Design program can perform DC or Time Domain analysis for arbitrary networks. Components allowed are linear inductive, capacitive, resistance, independent current/voltage sources, current- or voltage-controlled current sources, and user-defined nonlinear devices. Modified companion models and Newton-Raphson algorithm are used. Outputs are all the node voltages. *Required accessories: One Memory Module.* (649 lines, 1106 bytes, 27 pages)

Author: **Nai Chi Lee**
Stony Brook, New York

(41) Queue #01896C (Price $18*)

"QUEUE" is a fully integrated program enabling the user to compute queueing statistics for 6 different queue types. The program has been written in a modular manner so the user may delete segments of the program that do not apply to his/her situation. This feature enables the user to run even the longest of the routines on an HP-41C with only 1 memory module! The types are as follows — type A: single server model with arbitrary service times; type B: multiserver model with Poissen arrivals and exponential service times; type C: basic single server model; type E: models with a finite calling population; type F: single server model with arbitrary service times and a priority queue discipline. *Required accessories: 3 Memory Modules.* (908 lines, 1610 bytes, 55 pages)

Author: **Lauren Hansman**
Kitchener, Canada

(41) Crane/Outrigger Reactions and Stability #01952C (Price: $8*)

This program computes outrigger float reactions or wheel tandem reactions for a truck crane, and internally checks stability for specified lift loads, horizontal boom angle, and operating radius. A valuable program for those concerned with truck crane installations such as construction sites, piers, and docks. A warning "TIPPING" is displayed if instability is detected. *Required accessories: 2 Memory Modules.* (473 lines, 959 bytes, 18 pages)

Author: **Charles Dinsmore**
Seattle, Washington

(41) Joint Venture Financing #01894C (Price: $12*)

Investment is recovered by a two-part levy on sales. The first levy period runs until capital plus compound interest is recovered. The second stage usually equals the first levy period; alternatively, the user can specify first or second periods. Program calculates levies, first levy period, money and real internal rates of return. *Required accessories: Three Memory Modules.* (715 lines, 1351 bytes, 25 pages)

Author: **Clive Goldman**
London, England

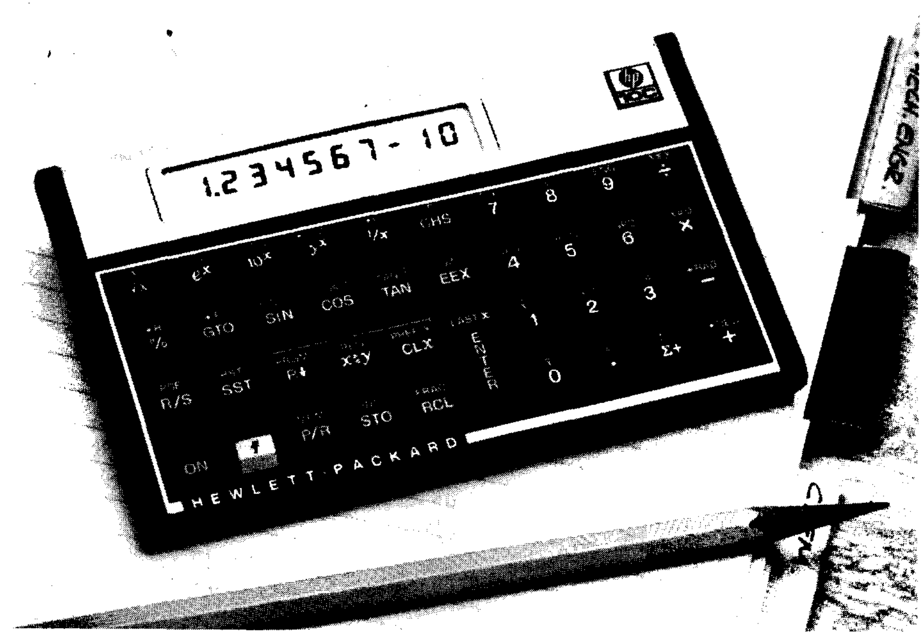
(41) Radar Plotting With Timer #01908C (Price: $8*)

This program plots, concurrently, five radar targets. It returns relative, truecourse, speed of targets, CPA times, distances, bearing. In the process, users ship may alter course or speed. In users ship simulated course, speed alterations, it forecasts plotted targets CPA times, distances, bearings for best course-of-action. *Required accessories: HP 82182A Time Module.* (339 lines, 563 bytes, 14 pages)

Author: **Matteo Kalcic**
Trieste, Italy

(The "big winners" for June were Mr. Wooley and Mr. Jones. They each chose a new Digital Cassette Drive, or a comparable prize. All of the other June winners received an Extended Functions/Memory Module or, again, a comparable prize — Ed.)

**U.S. dollars. Orders from anywhere outside the U.S. must include a negotiable check (or money order), in U.S. dollars, drawn on a U.S. bank. All orders from anywhere outside the U.S. and Canada must include an additional 10 percent fee for special handling and air mail postage. (For example, an order for two programs = $6 x 2 = $12 + $1.20 = $13.20 total.) If you live in Europe, you should order KEY NOTES Programs directly from the Geneva UPLE, but make certain you make payment as required by Users' Program Library Europe; the above $6 fee is good only for orders to the Corvallis Library.*



HP-10C: The New Low End, 10 Years After the HP-35A

Remember the HP-35A? A revolutionary concept in 1972. An "electronic slide rule" with one register, *no* programming, and — oh, yes — a nice, soft, black leather case. All for the "affordable" price of $395, factory list. And, it was as hard to find at dealers as it was to get delivery from HP.

Yet . . . had the electronic computing industry gone the way of inflation instead of each generation of calculator getting more powerful and cheaper, a calculator like the HP-35 would today cost $740.

But the programmable HP-10C — with 79 lines of program memory that can be traded-off for 10 registers — is *at least* 10 times more powerful than the HP-35. Just think: assuming the inflationary process mentioned above had become reality, and if the relationship between power and price were linear, the HP-10C would today cost over $7,400!

Instead of such a horrible fate, the HP-10C lists for a small percent of the HP-35's original price. So it is indeed the least expensive programmable *ever* produced by HP. Plus, it is a great *value*: it offers a virtually unbeatable combination of power and features, some of them unique in this price range.

One of the HP-10C's unique features is conditional branching. By virtue of this, the HP-10C is probably the least expensive "decision-making" machine in today's market. To this, the HP-10C adds other traditional HP features that enhance the power of a calculator and make programming and editing so much easier, such other features as merged keycodes, forward and backward program review, RPN, LAST X, pause, and the convenience of slim-line LCD design, long-life batteries, *Continuous Memory,* and automatic shut-off.

The HP-10C is the fifth model of the new Series 10 calculators. The HP-11C and HP-15C are also scientific programmables, with 203 and 448 lines of program memory, respectively. The HP-12C, also programmable, is designed for financial applications plus a large range of other special functions. The HP-16C is a virtually one-of-a-kind specialized calculator for computer science and digital electronics applications. This entire Series was created not only for maximum pocketability but also for durability. The HP-10C could very well be the most indestructible of the lot.

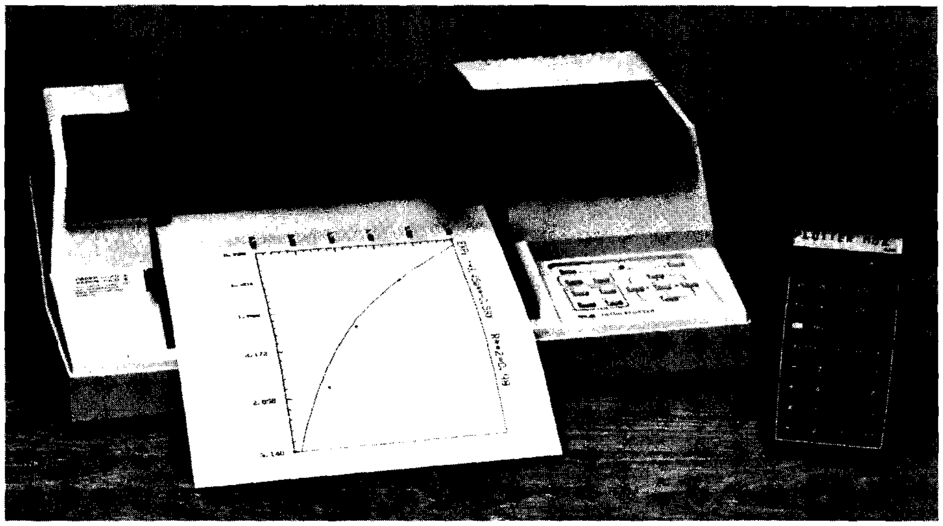
In addition to having a full complement of math, trig and log functions, the HP-10C keyboard includes rectangular/polar-coordinate conversions, degree/radian conversions, decimal hours/hours-minutes-seconds conversions, mean and standard deviation, linear regression and estimate, summations, correlation coefficient, and factorials. Also, numbers can be displayed in fixed-decimal, scientific, or engineering notation.

The HP-10C may be the least-expensive programmable calculator in the HP line, but you will find that it is way, way beyond the least-useful on the market. And with Christmas rapidly approaching, maybe now is the time to go see your local HP Dealer and get a head-start on what most of us refer to as "last-minute" Christmas buying.

"NANTUCKET" Coming Next Issue...

Because a lot of people are curious about how **Robert L. (Keystroke) Gardner** managed to get such a nice "painting" on his stock HP 82162A Thermal Printer, in the next issue we are going to tell you how he did it. Quite a few readers have written that they have driven themselves to distraction trying to emulate his "Nantucket painting."

Well, there's a lot more to the story than just the printer capability. We have discovered that Mr. Gardner is in a class all by himself, and we want you to read about him in the next issue. We'll bring you a story you will enjoy no end, and we'll let Mr. Gardner tell you in his own words how he managed to make that printer do some things that are seemingly not possible. So, until next February, don't waste your time trying to copy his "art," unless you've already discovered the "secret."



Announcing HP's Lowest- Cost Graphics System!

On November 1, Hewlett-Packard introduced two new products that, when combined with the HP-41 and HP-IL module, provide a complete system for generating graphics and bar code.

The **HP-41 Plotter Module (82184A)** is our newest software product for the HP-41. This 8KB plug-in module provides HP-41 language extensions that make plotting easy and fast. You can develop your own programs using the 52 plotter functions (including 10 bar code functions), or you can use the Utility Plotting Program to generate framed, labeled plots with a minimum of preparation. The Utility Plotting Program contains 5 plotting routines that use your inputs to automatically generate a plot of a math function or a series of points that you enter.

The Plotter Module enables an HP-41 to control HP plotting devices that operate in HP-IL and are compatible with the HP Graphics Language (HP-GL). HP's San Diego Division has recently announced a new version of their **HP 7470A Plotter** — with the HP-IL Interface. The HP 7470A (Opt. 003) will be available after November 1. The plotter accepts two standard media sizes: 8.5 x 11 inch (ANSI A) and 210 x 297 mm (ISO A4), and plots on paper and transparency film.

The HP 7470A uses low-inertia, microgrip technology. The medium is driven back and forth across the platen for plotting along the x-axis. Pen movement locates points along the y-axis. Movement of both paper and pens allows the HP 7470A to plot lines at speeds of up to 15 inches (38 cm) per second.

The HP-41 plotter system is easy to use and inexpensive to own. With the HP-41 Plotter Module and the HP 7470A Plotter, you can create multi-color line graphs, bar charts, and text pages to improve your presentations. Or use your HP-41 for data collection in engineering or statistical studies, then plot the results for better interpretation and analysis. Graphics makes complex data easier to understand and communicate.

An added feature of the Plotter Module is its bar code capability. Using the Plotter Module with the HP 7470A Plotter enables you to plot standard HP-41 bar code for the HP 82153A Optical Wand, as well as bar code for scanning devices used in other bar code systems. The module also enables you to print HP-41 bar code on the HP 82162A Thermal Printer.

The HP-41 Plotter Module and HP 7470A (Opt. 003) Plotter are available now at your local dealer.

PLOTTER EXAMPLE: EXPONENTIAL CURVE FIT

This program is a remake of the curve fit program in the "Standard Applications" PAC. It only fits data to an exponential curve but it could easily have been designed to fit any or all of the other types of curve fits. It uses "X?" and "Y?" from the plotter module to prompt the user for the X and Y coordinates of each point. When all data points have been entered the user simply presses [R/S] in response to the next request for an X coordinate.

The program automatically computes the correct scale for plotting and then plots a "*" at each data point. It then computes the equation for the exponential curve that best fits the data, and plots 20 points using this equation. Finally, it labels the graph with the computed equation in the form "EXP: Y=ae**bX" and the coefficient of determination in the form "R**2=".

The data points used in the example are the same as those used in the Standard PAC manual.

Xi: 0.72 1.31 1.95 2.58 3.14
Yi: 2.16 1.61 1.16 0.85 0.50

```
01♦LBL "EXP"
02 CLRG
03 "Y?"
04 ASTO 08
05 "X?"
06 ASTO 05
07 .0242
08 STO 02
09 505.00505
10 STO 03
11 STO 01
12 STO 07
13 XROM "PLINIT"
14 XROM "PLTUXY"
15 4221
16 STO 02
17 XROM "PLANOT"
18 XROM "PLTUXY"
19 ΣREG 12
20 CLΣ
21 RCL 08
22 1 E3
23 *
24 INT
25 1 E3
26 /
27 STO 05
28♦LBL 13
29 RCL IND 05
30 ISG 05
31 RCL IND 05
32 LN
33 X<>Y
34 Σ+
35 ISG 05
36 GTO 13
37 RCL 17
38 RCL 13
39 RCL 12
40 RCL 12
41 XEQ 09
42 STO 20
43 RCL 14
44 RCL 13
45 RCL 12
46 RCL 16
47 XEQ 09
48 RCL 20
49 /
50 STO 21
51 E↑X
52 STO 18
53 RCL 17
54 RCL 16
55 RCL 12
56 RCL 14
57 XEQ 09
58 RCL 20
59 /
60 STO 19
61 RCL 21
62 RCL 14
63 *
64 RCL 19
65 RCL 16
66 *
67 +
68 RCL 14
69 X↑2
70 RCL 17
71 /
72 STO 22
73 -
74 RCL 15
75 RCL 22
76 -
77 /
78 STO 22
79 "PLT"
80 FIX 2
81 ASTO 08
82 -19
83 STO 05
84 11
85 STO 02
86 XROM "PLTUXY"
87 1
88 PEN
89 LORG
90 5
91 CSIZE
92 SETGU
93 97
94 ENTER↑
95 30
96 MOVE
97 SF 17
98 "EXP: Y="
99 ARCL 18
100 "⊢e**"
101 ARCL 19
102 "⊢X"
103 LABEL
104 "   R**2="
105 ARCL 22
106 LABEL
107 CLX
108 PEN
109 STOP
110♦LBL 09
111 *
112 STO 23
113 RDN
114 *
115 RCL 23
116 -
117 RTN
118♦LBL "PLT"
119 RCL 19
120 *
121 E↑X
122 RCL 18
123 *
124 .END.
```

COMING IN THE NEXT ISSUE:

Richard J. Nelson, editor of the *PPC Calculator Journal* (see page 12 footnote) will contribute a technical article on the HP-41 system. Also, because he is just back from the PPC Conference in the United Kingdom, perhaps he will report on that and his experiences in other parts of Europe. Don't miss it!

Milliseconds Away

This letter, from **Patrick Schibli**, of Berneck, Switzerland, leads nicely into the subject of function timing with the HP 82182A Time Module.

As a matter of fact, computers add two numbers faster than they can multiply them. Therefore, I propose that everyone change their: 2; *; program sequences to: ST+ X (unless the LASTX register is used). Up to four "ST+ X" statements work faster than a simple multiplication by 2, 4, 8, or 16, respectively. However, using "ST+ X" to double a number more than once consumes extra bytes.

Mr. Shibli is right. Using the "ST+ X" function, rather than 2; *, to double a number, will save time. If you have a Time Module, it is easy for you to determine just how much time you do save. With only a Time Module connected to your HP-41, the time savings will be approximately 60 milliseconds. Now, the time it takes to execute a function is a "touchy" subject, because there are many variables that affect the operation speed of the HP-41 and some of those variables are beyond our control. Most functions have execution times that depend even upon the numbers involved. What follows, in this article, is an explanation of how you can use your Time Module to determine, for a given HP-41 system configuration, whether one programming method executes more quickly than another.

For a particular condition, timing a function, using the Time Module, is a relatively simple procedure. The general process requires that a time measurement be taken for multiple iterations of the function in question and, then, that the time measurement be translated into seconds per iteration, or whatever.

Take a look at the following routine.

```
01♦LBL "TFUNK"   11 RCLSW
02 .1            12 1 E2
03 STO 12        13 *
04 0             14 FRC
05 SETSW         15 LASTX
06 RUNSW         16 INT
07♦LBL 00        17 .6
08 ISG 12        18 *
09 GTO 00        19 +
10 STOPSW        20 END
```

This routine, "TFUNK," contains all of the fundamentals for timing a function. Lines 02 and 03 initialize register 12 (our loop control register) for 100 iterations. Lines 04, 05, and 06 initialize and start the time measurement. Lines 07, 08, and 09 are the iteration control loop. Line 10 stops the time measurement. Lines 11 through 19 display the measured time interval in seconds per iteration. This is just a skeleton routine. You still have to calibrate it and make some other adjustments (read on) in order to use it.

Now, key-in the routine, "TFUNK." Press [GTO] [.] [.] to pack it, and then execute TFUNK. FIX 4 and note the number in the display (it should be around .100 if nothing but the Time Module is connected to your HP-41). This is the average time, in seconds (100 ms), that it took each iteration of the first 100 iterations of the control loop. Press [R/S]. Note that the resulting average time for each of the second 100 iterations is less than that of the first 100 iterations. Succeeding runs result in a displayed time that varies by no more than a few tenths of a millisecond. The reason the first run gives a greater time is because of the somewhat lengthy compiling procedure that takes place during the first iteration of the loop. (See KEY NOTES V5N2p7c.) After making changes or PACKING a function-timing routine, the results of the first run should be disregarded.

Now, modify TFUNK so it gives a result that is close to zero. Just add, to the end, two lines that subtract the number that resulted when you ran the routine the second time in the last paragraph. They will be something like: .0995; −, or .0970; −. PACK the routine and run it twice to verify that the result you get is small (no bigger than a few tenths of a millisecond). With all this completed, we are ready to time some functions.

To time the function X<>Y, place the line X<>Y after line 07 in TFUNK. TFUNK will now look like this:

```
01♦LBL "TFUNK"   13 1 E2
02 .1            14 *
03 STO 12        15 FRC
04 0             16 LASTX
05 SETSW         17 INT
06 RUNSW         18 .6
07♦LBL 00        19 *
08 X<>Y          20 +
09 ISG 12        21 .0994
10 GTO 00        22 -
11 STOPSW        23 END
12 RCLSW
```

where the number in line 21 is the time that resulted when you ran the skeleton routine.

PACK the routine (XEQ "PACK") and run it (XEQ "TFUNK) twice. You will find that the X<>Y function requires about 10 ms. That is, "about 10 ms" with nothing but the Time Module connected to the HP-41, and with the batteries in fairly good condition, and at a temperature of about 295 degrees Kelvin*, etc.

You can now replace X<>Y with any function or sequence of functions whose execution times are not number-dependent. Such functions as R↑, RDN, ENTER, LASTX, etc. fall into this category.

Such functions as SIN, LN, FACT, etc. have execution times that depend on the parameters. There are tests and traps in the microcode of these functions that may cause the result for one parameter to be resolved quicker than the result for another parameter.

To time a function whose execution time is number-dependent, like SIN or LN, you must have a skeleton routine that sets-up the stack so that the function always operates on the same number. To time the SIN function, start with a routine like this:

```
01♦LBL "TFUNK"   13 1 E2
02 .1            14 *
03 STO 12        15 FRC
04 0             16 LASTX
05 SETSW         17 INT
06 RUNSW         18 .6
07♦LBL 00        19 *
08 RCL 00        20 +
09 ISG 12        21 .1210
10 GTO 00        22 -
11 STOPSW        23 END
12 RCLSW
```

Adjust the number in line 21 so that running this routine in your HP-41 gives a result close to zero (the same as we did above). Register 00 will store a constant parameter. Key-in SIN after the RCL 00 in line 08. XEQ "DEG." Store 30 in register 00. XEQ "PACK" [RTN] [R/S]. When it stops, run it again. The result will be the amount of time that your HP-41 takes to calculate the SIN of 30 degrees. Store 90 in register 00 and press [R/S]. The result this time will probably be smaller. The HP-41 uses a different path through the microcode to obtain the SIN of 90 degrees.

You can see that compiling a list of the *precise* execution times for all of the HP-41 functions would be an improbable task.

However, with your new Time Module, you now have the tools to determine relative execution times, and to approximate execution times for a given system configuration.

Methods for timing routines and functions such as the ones shown above are handy tools to help you shorten the execution times of lengthy or iterative programs. If, in a loop that iterates 100 times, you can replace the sequence: 2; *; with ST+ X, you trim about 6 seconds from the execution time. Now, that "doesn't sound like much, but it adds up."

**That's 22 degrees Celsius or 71.6 degrees Farenheit.*

Your Address . . . Revisited

In the last issue (V6N3p7c), we addressed the issue of keeping your address up to date. Unfortunately, there was some misunderstanding about where you should send address changes and corrections.

If you live in the United States or Canada, you should send your address changes and corrections to the Corvallis Users' Library. If you are a member of the Corvallis Users' Library and get KEY NOTES free as part of your membership, you must send address changes and corrections to the Corvallis Users' Library. If you live in any European countries and get KEY NOTES in an envelope that does NOT have a U.S. return address on it, DO NOT SEND YOUR ADDRESS CHANGES TO CORVALLIS. Your KEY NOTES came to you by way of the UPLE in Geneva, so send your changes and corrections to the Geneva UPLE.

For the rest of the world, if your KEY NOTES envelope does NOT have a U.S. return address on it, DO NOT SEND CHANGES AND CORRECTIONS TO CORVALLIS. Send them to the return address on the envelope in which you received your newsletter.

Back Issues . . . Revisited

ALL ORDERS FOR BACK ISSUES, IN PRINT, MUST BE SENT TO THE CORVALLIS USERS' LIBRARY, AND PAYMENT IN U.S. DOLLARS MUST ACCOMPANY YOUR ORDER. See page 19 for details and a list of available back issues still in print.

If you send your order for back issues to the Geneva UPLE you will cause a very long delay because it is time-consuming and expensive to transfer money and orders from Geneva to Corvallis. To keep your cost for back issues to an absolute minimum, they are stocked only in the Corvallis Library and mailed directly to you when you order.

For those who want a complete set of HP KEY NOTES back to V1N1 (Jan. 1977), or to replace a missing issue, The Corvallis Users' Library now offers a photocopy service at the prices listed below. An index will be furnished on request. **Remember that these are photocopies, not printed copies.** Also, the prices and any offers appearing in these or any other back issues are no longer valid.

V1N1 January 1977 (12 pages)
V1N2 June 1977 (12 pages)
V1N3 October 1977 (12 pages)
V2N1 February 1978 (12 pages)
V2N2 May 1978 (12 pages)
V2N3 August 1978 (12 pages)
V2N4 November 1978 (12 pages)
V3N1 February 1979 (12 pages)
V3N2 May 1979 (12 pages)

All prices include first-class or air mail. Payment must accompany your order and must be a check or money order in U.S. dollars, drawn on a U.S. bank. Or you may use your American Express, VISA, or MasterCard account, but be sure to include your account number and card expiration date. Make sure you mail your order and payment to the Corvallis Users' Library.

NO. OF ISSUES	U.S., MEXICO, CANADA	ALL OTHER COUNTRIES
1	$ 2.50	$ 3.00
2	4.50	5.50
3	7.00	8.00
4	9.00	10.50
5	11.00	13.00
6	13.00	15.50
7	15.50	18.00
8	17.50	20.00
9	19.50	22.00

These prices include photocopying, handling, envelope, and postage. Please allow a minimum of two weeks for delivery.

Local Label GTO's

In KEY NOTES, V5N2p6c, the "In the Key of HP" article discusses local-labels and the compiling (or "jump distance recording") of local-label GTO statements. **Paul Boltwood** of Stittsville, Ontario, Canada requested a more thorough discussion of compiled GTO statements. Thus, we have the following.

Upon the first execution of a local-label GTO statement, the HP-41 searches (line-by-line, starting at the GTO statement) for the local-label destination. Once found, the location of the local-label, relative to the calling GTO statement, is recorded with that GTO statement so that future executions will not require a line-by-line search. This process of recording local-label locations with their calling GTO's is known as *compiling*.

Whenever a program is PACKed to eliminate null bytes or expanded to allow for the addition of more lines, or whenever a program line is deleted, the local-label locations recorded with the GTO statements are "erased." This process is called *decompiling*.

When a program is packed, the calculator sets a bit (a "flag" called the *PACK-bit*) in the END statement of that program. Then, on future executions of GTO . . or PACK, the calculator checks the PACK-bit to determine if the program should be PACKed. If the PACK-bit is set, then that program will not be PACKed *or decompiled.* The PACK-bit and all compiled GTO's are maintained by *Continuous Memory,* whether the calculator is on or off. The PACK-bit is cleared whenever the program is altered.

Thus, a PACKed and compiled program will not be decompiled while in main memory unless it is altered.

When a compiled program is transfered to a mass storage medium, the compiled GTO's are recorded with the program. So, when the program is read back into the machine and executed, it will run just as quickly as it did before it was recorded; no compiling process is necessary. However, because the END statement, and thus the PACK-bit, is *not* recorded with the program, the first execution of PACK or GTO . . . will decompile the program.

Another Slice of PI

Okay, okay . . . now we all know that it is possible to access the PI function without touching the PI key. It all started in the last issue of KEY NOTES (V6N3p15a) when we published "PI A La Mode." There, we presented a somewhat challenging puzzle, and that puzzle read like this:

From a Master Clear state, and without using the 0 to 9 keys or the PI key, and with the fewest possible keystrokes, how do you get pi in the display?

The two solutions that we offered to this puzzle were [XEQ] RAD [COS-1] [STO] [+] [.] [X] and [XEQ] RAD [COS-1] [ENTER] [+]; or, 12 and 10 keystrokes, respectively. Now, the true "PI in the face" came when we began receiving letters that offered the five-keystroke solution: [XEQ] [ALPHA] PI [ALPHA]. As these letters claim, this solution does satisfy the given "rules." However, we do wish that those who sent these letters had included a napkin, with which we could wipe the meringue from our eyes.

Among the letters we received on this subject was this letter from **Herbert Gudehus,** of Hamburg, West Germany. Mr. Gudehus presents a logical argument for using his "recipe" for PI.

In V6N3p15 you published a nice puzzle, "PI A La Mode," and on page 16 of that issue you published two answers. Both of these answers must be supplemented by XEQ "DEG" (6 additional keystrokes) if DEGREE-mode has to be restored. Under this condition, a shorter solution is [COS-1] [XEQ] D-R [ENTER] [+] (11 keystrokes).

Another unique solution was sent to us by **Jean-Marc Delbos,** of Paris, France. Mr. Delbos' solution was this: [XEQ] RAD [Σ-] [R-P] [X<>Y] (11 keystrokes).

Thank you Mr. Gudehus and Mr. Delbos. And, thanks to everyone who sent in their solution. We only wish that we had the room to, at least, print all of your names.

Guest Article: About Program Files

Occasionally, one of your letters causes an antenna to go up to alert us that perhaps it could be the basis for an article that would be of benefit to all of our readers. *This* started as a letter, but we think it is now a very good *article*; we also think *you* will like it.

Our "guest writer" is **Jeffrey D. Smith,** a 24 years young (*his words!*) Systems Programmer from La Palma, California. He is currently employed by Southern California Edison (an electric utility) as a Systems Programmer specializing in APL (an interactive, high-level programming language) and Graphics (both interactive and batch processing). He was previously employed by I. P. Sharp Associates, Inc. (a Canadian-based APL time sharing vendor) as a Systems Programmer for APL development and maintenance. Both Mr. Smith and his spouse are native Southern Californians, and they have two daughters and a son.

In addition to APL, Mr. Smith also knows something about an HP-41. We also think he writes rather well. Here, then, is his "guest article."

While reading through several issues of KEY NOTES and other publications, it came to my attention that there are some HP-41 users who are not fully aware of the definition of "last program in memory," or of "program file." Plus, some HP-41 users do not fully understand the true behavior of certain program-loading functions, such as: RSUB, MRG, WNDLNK, etc. Yet, with the advent of the new HP-IL interface loop, precise and accurate knowledge of these terms and functions is *essential* for achieving a productive use of the HP-41 system.

First, HP-41 programs in main memory do not have names. Many users refer to the global line on line 01 as the program name, but this is not true. As the HP-41 owner's manual points out, programs in main memory are separated from each other by END statements. END statements serve to delimit "program files," and a program file is not required to have any global labels. Global labels serve only to locate program entry points and to enable the access of programs and subroutines from other programs in main memory. If a program file has two global labels, then what is its name? You may say that either name (as indicated by the labels) is valid. But, this is incorrect because the results obtained from the program usually depend on which global label is executed.

How, then, does a program file receive a name, if at all? A program file is given a name when it is stored in Extended Memory or on an HP-IL mass storage device (HP 82161A Digital Cassette Drive). The file name is entirely independent of any labels that may be inside the file, and it is used only for locating the file on the storage medium. Thus, many program files, with the same global labels, can coexist on the same storage medium. The labels only become effective when the program is loaded into main memory. In summary, global labels are used during execution, and program *names* are used for storage purposes. They are independent entities.

Now that we know how to identify and locate a program file, let's see what tools we have for loading and storing programs. There are three relatively inexpensive I/O devices: the Card Reader, the Optical Wand, and the Extended Functions/Memory Module.

CARD READER

This device allows the HP-41 to read and write unnamed program files via magnetic cards. Historically this was the first peripheral to provide mass storage. With the HP-41 positioned to a desired file in program mode, the file can be automatically recorded to cards. The Card Reader depends on the user to mark (or "name") and keep track of program files recorded on cards. The ability of the Card Reader to manipulate program files is limited, providing only two functions for loading programs — RSUB and MRG — and only one, non-programmable, function — WPRV — for writing a program file.

The WPRV (write private) function causes the current program file to be written to cards as a PRIVATE program. This prevents alteration and duplication of the program. Being non-programmable, the user must make a conscious decision to use WPRV, because its effect is irreversible.

The RSUB (read subroutine) function prompts the user for a card program that replaces the "last program file" in main memory. What is the "last program file" in main memory? It is the program file that is terminated by the permanent .END. statement. Note that after executing a GTO .., the "last program file" consists of a single statement — the permanent .END. — and every other program file is terminated by a normal END statement.

The program file that is executing RSUB (or to which the HP-41 is positioned when RSUB is executed manually) will not be replaced by the card program. If the "last program file" contains the executing RSUB, then the permanent .END. of this program will be converted to a normal END and the incoming program file will become the "last program file." Also, the conversion of the permanent .END. to a normal END will occur if RSUB is executed from the keyboard while the HP-41 is positioned to the "last program file." Therefore, the conversion of .END. to END will *always* occur if the HP-41 is positioned to the "last program file," regardless of whether RSUB is executed manually or from a running program. Usually, this is not a problem, but if you execute GTO .., and use RSUB to load a program, then a solitary END statement, that must be deleted manually, will be generated immediately preceding the new program.

Execution continues with the next statement after the RSUB function. In order to invoke the newly loaded program, the active program file must either call or branch to it via a global label. This presupposes that the newly loaded program is indeed the correct program; since card programs are not named by the calculator, there is no way for the Card Reader to validate the program. Indeed, even a successful call to a global label from the active program is no guarantee that the label is in the desired program (it could be that an older or even obsolete version of the intended program was mistakenly loaded). The problem of loading the incorrect program file is remedied by accurately marking each magnetic card. And, the solution to automatically invoking the new program lies in the MRG function.

The MRG (merge program) function causes a program file to be loaded into main memory immediately after the current instruction, replacing all program lines up to the permanent .END. statement. Notice that MRG works only for the "last program file." It cannot be used from Application Pac modules nor by private programs, because these types of program files cannot be altered, which is exactly what the MRG function is supposed to do. After the operation is complete, execution continues with the first line of the newly loaded program file. Hence, there's no need to know what global labels (if any) are needed. Because execution starts immediately after MRG (except when MRG is performed manually), the danger of loading the wrong program file becomes much more serious.

To load and automatically execute a card program, the following short routine could be used as the last program file:

```
01 LBL "RLNK" ; Read subroutine and link
02 MRG ;        Merge the card program
```

Each time another subroutine is needed, the main program can either call or branch to "RLNK" to read another program. Notice that the MRG function clears the pending return (RTN) stack. In order for the subroutine to return control to the main program, it must explicitly branch back to it via a global label. Execution will halt if RTN or .END. is encountered.

Although cards can be cumbersome, the RSUB and MRG functions provide a workable means of managing subroutines. However, both of these functions rely on the user to provide the correct program cards.

OPTICAL WAND

The Wand is an "input-only" device, as opposed to the Card Reader, which is an input and output device. The Wand provides two programmable functions — WNDSUB and WNDLNK — for loading bar-coded program files into main memory.

The WNDSUB (scan bar-coded subroutine) function operates identically to the Card Reader function RSUB, but uses bar code rather than magnetic tape as the storage medium.

The WNDLNK (scan bar-coded subroutine and link) function performs essentially the same task as the "RLNK" routine listed above. It causes the incoming program file to replace the last program file according to the same rules as WNDSUB and RSUB but, in addition, control is passed to the first line of the newly loaded program exactly as if it had been called via XEQ. There is no need to have a global label on line 01, because the HP-41 already knows where to begin execution. WNDLNK has two advantages over the "RLNK" routine: (1) the "RLNK" routine will be rendered useless if a private program file is read, whereas the WNDLNK function may be used to repeatedly load and execute PRIVATE, bar code program files; (2) the WNDLNK function preserves the pending RTN stack so that the subroutine will return control to the main program on the execution of a RTN or .END. statement.

The same problems that exist with unnamed card programs are still present with unnamed bar code programs. Both devices rely on the user to submit the correct file for loading, but otherwise they provide the same utility.

EXTENDED FUNCTIONS/ MEMORY MODULE

This new module provides many extensions to the HP-41, including Extended Memory. Extended Memory can be used by either the HP-41C or the HP-41CV, because it is slightly different than main memory.

Extended Memory is composed of *Continuous Memory* registers just like main memory, but you must use the functions that are provided by the module in order to gain access to that memory. The functions permit the creation, manipulation, and deletion of ASCII, data, and program files. All of these files must be given unique names. ASCII files are made up of characters, and

(Continued)

therefore are byte oriented. Data files are made up of complete registers (similar to data cards produced by the Card Reader, but more flexible). Program files are similar to card and bar code program files, but they must be referenced by name. The name is independent of any labels that may be present inside the program file.

There are two functions — GETP and GETSUB — provided for loading programs into main memory, and one function — SAVEP — for storing programs into Extended Memory. They are all programmable, and they represent a significant step ahead of their counterparts in the Card Reader and the Wand.

The SAVEP (save program) function is used to write a program file from main memory to Extended Memory. It requires a name for the target file in Extended Memory and some way of identifying the source program file in main memory. You can identify the source program file by specifying any global label that occurs within that program, or you can position the HP-41 to the program and then indicate that the current program file is to be saved. Whichever way you choose to indicate the source program file, you must specify a unique name for the target file. (If you specify the name of a program file that already exists in Extended Memory, it will be replaced by the source file.)

Incidentally, the SAVEP function will preserve the privacy status of the source program file in the target file. Thus, private program files may be transferred between main memory and Extended Memory. The HP-41 only prevents private program files from being transferred to an *external* storage medium.

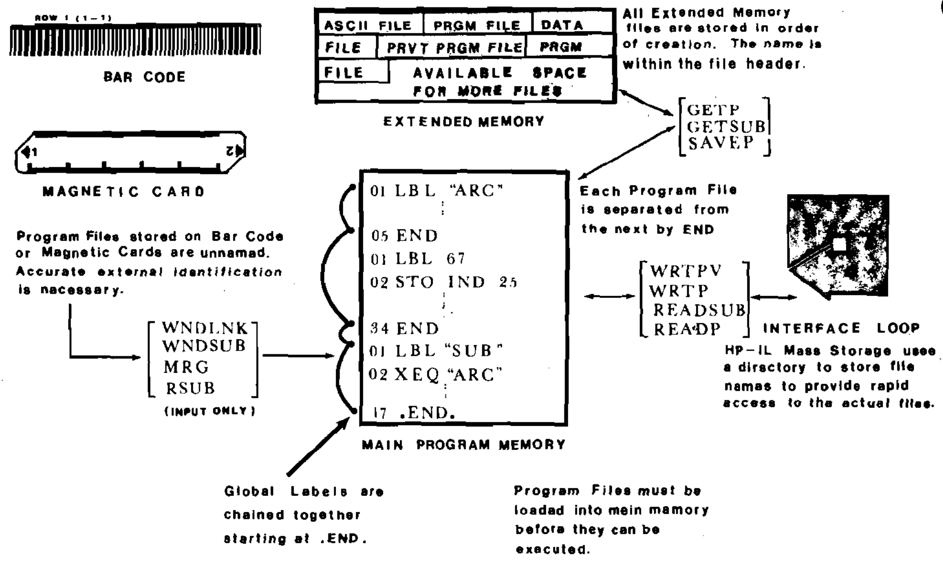
You may think that the GETSUB (get subroutine) function would be quite similar to RSUB and WNDSUB, but this is not so. The GETSUB function is actually more like the built-in HP-41 COPY function that is used to copy program files from Application Pacs into main memory. The GETSUB function always loads a program file into main memory *after* the last program file; it never overwrites the last program even if the last program contains only the permanent .END. statement. If the user isn't aware of this behavior, then repeated use of GETSUB could clutter main memory with many solitary END statements.

Why should GETSUB work differently from RSUB and WNDSUB? One reason might be that it was easier to design it that way. The GETSUB function might be "borrowing" some of the microcode used by the COPY function, thus saving the designers the trouble of reinventing the wheel. However, a closer look reveals that neither the Card Reader nor the Wand provide a means of appending programs; they only replace the last program file (with the minor exceptions mentioned earlier).

The GETP (get program) function is like WNDSUB or READSUB, and WNDLNK mixed together. The program that is loaded by GETP always replaces the last program file, even if it is the one executing GETP. If the active program is not the last program file, then it continues to run. This is the same as the RSUB and WNDSUB functions. However, if it is the last program file, then it is entirely replaced by the new program file and execution continues from the first line of the new program. In this respect, it is similar to the "RLNK" routine or the WNDLNK function and, like WNDLNK, it does not alter the pending RTN stack in any way.

HEWLETT-PACKARD HP-41C/V

Program File Storage


ROW 1 (1-1)
BAR CODE
MAGNETIC CARD
Program Files stored on Bar Code or Magnetic Cards are unnamed. Accurate external identification is necessary.
WNDLNK
WNDSUB
MRG
RSUB
(INPUT ONLY)
ASCII FILE
PRGM FILE
DATA FILE
PRVT PRGM FILE
PRGM FILE
AVAILABLE SPACE FOR MORE FILES
EXTENDED MEMORY
All Extended Memory files are stored in order of creation. The name is within the file header.
GETP
GETSUB
SAVEP
01 LBL "ARC"
05 END
01 LBL 67
02 STO IND 25
34 END
01 LBL "SUB"
02 XEQ "ARC"
17 .END.
MAIN PROGRAM MEMORY
Each Program File is separated from the next by END
WRTPV
WRTP
READSUB
READP
INTERFACE LOOP
HP-IL Mass Storage uses a directory to store file names to provide rapid access to the actual files.
Global Labels are chained together starting at .END.
Program Files must be loaded into main memory before they can be executed.

Thus, Extended Memory provides an excellent means of dynamically storing and loading program files. In fact, a programmer can now design efficient overlay structures that can execute independently of user intervention (such as loading magnetic cards). A "supervisor" program can be developed that will load a program into main memory and then transfer control to the new program. When the new program terminates with RTN or .END. (or erases itself with PCLPS), control is passed back to the supervisor program, which then determines the next routine to be loaded into memory (replacing the previous program). Now, we have entered the era of "disposable" code, memory sharing, and time sharing. Attaching a mass storage device increases the possibilities.

If you are a little wary of using GETP for such a supervisor program, then you can use the GETSUB function during the initial staging of the supervisor to load an empty program file. This will ensure that a normal END exists somewhere between the supervisor and the last program file. From then on, the supervisor can use GETP without ever overlaying itself. (An empty program file can be created by keying GTO .., and then executing SAVEP, after placing a file name such as ",NULLPGM" in the ALPHA-register. The initial comma in the ALPHA-register indicates that the current file is to be saved in Extended Memory under the name "NULLPGM.")

Many advantages can be realized by dividing large, monolithic application programs into several small subprograms. For instance, the program region size can be greatly reduced, because only the portion of the application that is executing needs to be in main memory. Also, all of the advantages of "modular" programming become available. Each program file or "module" of an application package ideally performs a single, unambiguous function. If the application needs to be changed, then only the affected modules need to be edited and replaced. This eliminates having to load the entire application package into main memory (which may not be possible).

The Extended Functions/Memory Module represents a significant advance over the Card Reader and the Wand (although, not a replacement). The WNDSUB, or READSUB, and WNDLNK functions have been condensed into GETP, and a new function GETSUB has been added to permit appending program files in main memory. Also, the problem of incorrect user intervention has been eliminated.

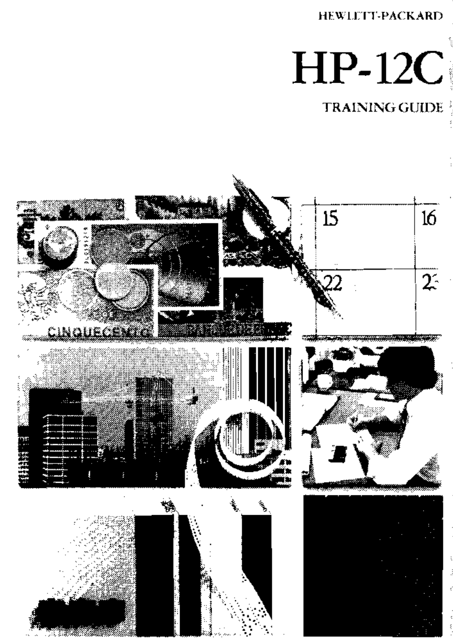
DID YOU KNOW?...

Did you know that *all* copies of KEY NOTES are printed in the U.S.A.? However, if you live in Europe or anywhere outside the U.S. and Canada, your copy actually comes from the place that shows as the return address on your envelope. So if you have problems or questions about your KEY NOTES, always direct your inquiry to whoever sent the copy to you. See page 5, lower corner, for more details.

New HP-12C Training Guide Released

Have you considered buying the new HP-12C Advanced Financial Programmable Calculator but were afraid you couldn't understand all those financial functions on the keyboard? Well, fear no more! The *HP-12C Training Guide* quickly puts the power of the HP-12C into the hands of the beginning calculator user. It very quickly dispels fears of financial calculations and programming, and it helps the novice to develop the skills that permit anyone to solve the toughest financial problems.

As a tutorial workbook, the *HP-12C Training Guide* emphasizes learning by doing. It is written in a friendly, self-paced format. Divided into 12 chapters, the training guide provides example solutions and follows each topic with review questions to ensure thorough understanding.



With over 150 pages, the *HP-12C Training Guide* takes the user from the features and basic operation of the HP-12C to financial concepts and then on to elementary programming. By making it possible to understand how to use and apply the HP-12C, the training guide not only makes the powerful financial features of the HP-12C readily available to beginners, it also serves as a refresher course for experts.

So if you are involved in financial and business calculations, yet afraid or intimidated by complex computations and equipment, then the *HP-12C Training Guide* is for you. Plus, it also makes an excellent, affordable Christmas present, and it's available now at your local authorized HP Dealer. The order number is 00012-90022. We'll also bet that it is tax-deductible, too; check with your income tax rules or your tax consultant.

Care and Feeding of Cassettes

The HP 82161A Digital Cassette Drive is a rugged, but sophisticated electronic device. As such, it is very capable and durable, but a few common-sense rules should be followed to prolong the life of the drive and the tapes it uses. The main area of concern is tape wear, but there are also some general-care procedures that will extend the life of your equipment. For example, avoid unnecessary abuse of your equipment; it is an investment that will continue to serve you for many years if common sense is applied during its use.

WEAR

Tape wear is the greatest cause of malfunction on the cassette drive. Excessive directory accesses are the main cause of early tape failure. The tape will tolerate a minimum of 500 accesses, but 1500 accesses for any particular tape is more typical. Following are a few procedures that will help extend the life of a cassette.

1. Minimize directory accesses. The directory is the most-used section of tape. Such functions as WRTP, WRTA, and DIR use the directory every time the function is executed. Continued rewriting of program files during editing is usually unnecessary and should be avoided. It is best to rewrite at the completion of a session, if possible.
2. Avoid partial data file writes. Register-by-register writes should be avoided. It is better to use WRTRX where X > 32. This will assure that you are storing at least one record at a time, thus minimizing the wear on any particular segment of tape.

GENERAL CARE

The tips in this section will aid in understanding the critical areas of your tapes and drive. Again, use common sense when handling tapes or using the drive.

1. Always rewind the tape. When you have completed a session, it is recommended to rewind the tape, even if it is left in the drive. This will protect the storage areas of tape from accidental mechanical damage.
2. Avoid touching the tape. The tape recording surface is very sensitive to any contact by foreign materials. Fingers or cleaning material will damage the tape. Rewinding will help avoid accidental contact with the tape.
3. Remove slack before using a tape. Tapes kept in the storage compartment will sometimes partially unwind. Tensioning the tape before inserting will avoid harsh movements by the drive. Insert an instrument, such as a pencil or pen, into the hub of the cassette, and rotate the hub until the tape appears taught inside.

CAUTION DO NOT OVER-TENSION THE TAPE.

4. Avoid temperature extremes. The inside of an automobile on a hot summer day can exceed the recommended storage temperatures. Also, sudden temperature changes can cause condensation to occur, and expansion or shrinkage of the tape.
5. Clean the head frequently. Use isopropyl alcohol, in accordance with manual instructions, to clean the head at least once a month under normal operating conditions. Be careful not to touch the tape viewing window with the cotton swab, as it will cause permanent stains.
6. Keep the drive clean. Dirt, dust, and liquids will damage your drive if they reach the mechanism inside the tape access door. Do not allow foreign matter to enter this area!
7. Avoid magnetic fields. The drive is protected, as far as possible, from radio and magnetic interference; however, placing the drive in close proximity to a strong magnetic field will cause disruption of the drive. For example, keep the drive as far as possible from a T.V. or video monitor.
8. Avoid operation with the BAT light on. The low-battery indicator is telling you to stop and recharge the battery. Follow instructions in the user's manual for the drive. Failure to do so could cause loss of data on the tape.
9. The tape drive is not a toy! Do not use the cassette door as release for nervous energy. Repeated unnecessary opening and closing of the cassette door will cause premature failure of the head cable.
10. General care of exterior. The exterior of your cassette drive is made of high-quality material that should last a lifetime if it is properly maintained. Cleaning with soap and water and a damp cloth is acceptable, as is the use of a mild plastic cleaner/polish such as "Snap." Do not use harsh household cleaners or abrasives. Cleaners containing fluorinated, or chlorinated hydrocarbons also will cause damage to the surfaces.

Again, the primary "rule of thumb" is to always use common sense when handling tapes or using the drive.

"Pioneering" With HP-41 Peripherals

The HP-41 is a real pioneer in the hand-held computer market! As such, it has its own set of nuances that often need some explanation to enable both the experienced computer buff and the neophyte to feel comfortable with its operation. The implementation of the printer functions in the HP 82160A HP-IL Interface Module fall into this category. During the design phase, it was decided that the HP-IL version should be "backward-compatible" with the dedicated HP-41 printer as much as possible. This meant that the character sets had to match to allow previously written programs to function without modification. Another very important objective was to have an ASCII-compatible printer that would allow future mainframe computers, such as the HP-75, to function properly without any modification or interpolation. As you might imagine, these two objectives are mutually exclusive, so a compromise had to be struck.

The solution entailed creation of an alternate character set (a modified HP-41 set) that would allow backward-compatibility. As stated, it is a modified set, as it was also necessary to maintain

(Continued)

the ASCII compatibility in the area of providing a carriage return and line-feed command at their respective positions of decimal 10 and 13. This cost two character locations in the set, and the chosen path to make it as transparent to the user as possible was to use character 0 to replace character 10 (it was a duplicate anyway), and put character 13 at location 124, causing, in-turn, the loss of character 124 (a|, or Sheffer, sign). Not a big loss, considering it can be easily created using either the BLDSPEC or ACCOL commands. Now, whenever a character 10 or 13 is sent, the HP-41 (actually the Interface Module) goes to a look-up table that will in-turn send either a 0 or 124, respectively, to the 82162A printer to enable the corresponding character. Additionally, if an HP 82163A Video Monitor is used, character 126 (the Σ sign) is not translated, as it does not exist; and the ~ at that location in the video interface (and the ASCII set) was deemed not an acceptable substitute, therefore a blank was substituted.

Summary: For the HP 82163A Video Interface, and the HP 82162A Thermal Printer, the following responses to HP-41 commands will occur. (SF 17 on the HP-41 prior to attempting verification of this table.)

COMMAND \\\\\\ CHAR CODE	DEVICE	ACCHR	ACA	OUTA	PRA
10	PR	◆	NO RESP	NO RESP	NO RESP
	VIDEO	NO RESP	LF	LF	EXT LF
13	PR	ⱥ	ⱥ	CR	ⱥ
	VIDEO	|	|	CR	EXT CR
124	PR	ⱥ	ⱥ	ⱥ	ⱥ
	VIDEO	|	|	|	|
126	PR	Σ	Σ	Σ	Σ
	VIDEO	NO RESP	NO RESP	~	NO RESP

82162A Thermal Printer Applications: Since it is likely that some of you would want to print character 124, the | sign, here are two routines that will accomplish the task.

1. Using BLDSPEC

```
CLX        Clear the "X" register.
ENTER      Clear the "Y" register.
BLDSPEC
0
BLDSPEC
0
BLDSPEC    Build the character.
127
BLDSPEC
STO 00     Store it in reg. 00 for later use.
ACSPEC     Accumulate it into print buffer.
PRBUF      PRINT IT!!!
```

2. Using ACCOL

```
3
SKPCOL     Move column pointer to 4th
           column.
127
ACCOL      Create a full column of dots at
           4th dot column.
PRBUF      PRINT IT!!!
```

Feedback

This column contains reader feedback about articles or routines that appeared in previous issues. Though much of the information presented here is useful on its own, you will find that it is a good idea to have your library of KEY NOTES handy while reading this column.

As the production deadline for each issue of KEY NOTES approaches, those of us on the KEY NOTES staff begin pushing things aside in favor of getting KEY NOTES to the printer. This letter was pushed aside during the production of the last issue, and it shouldn't have been. The letter and routine are from **Mike Edwards**, of Urbana, Illinois.

(V6N3p13a) I regret that I sent you an inaccurate routine (FLEFT) a few days ago. It doesn't allow for the overhead bytes contained in ASCII files. This revised routine has been tested, and works under the following assumptions (see page 24 of the *Extended Functions/Memory Module Owner's Manual*): (1) there is one unusable byte in every ASCII Extended Memory file, plus (2) one unusable byte for each record within the file. I suppose (1) is an end-of-file pointer and (2) is a record length indicator. This routine calculates the number of available bytes within an ASCII Extended Memory file.

```
01◆LBL "FLEFT"      21 FC?C 25
02 FIX 0            22 GTO 01
03 CF 29            23 ALENG
04 "FILE NAME?"     24 -
05 AON              25 FS? 17
06 STOP             26 GTO 00
07 AOFF             27 1
08 0                28 -
09 SEEKPTA          29 GTO 00
10 FLSIZE           30◆LBL 01
11 ASTO Y           31 CLA
12 ASHF             32 ARCL X
13 ASTO Z           33 "⊢ BYTES LEFT: "
14 7                34 ARCL Y
15 *                35 ARCL Z
16 1                36 AVIEW
17 -                37 FIX 4
18◆LBL 00           38 SF 29
19 SF 25            39 END
20 GETREC
```

Here's a "Feedback" letter that brings back many memories to long-time KEY NOTES readers. Because of that and because we do not want to again start a flood of factorial routines, we are publishing it with the warning that there are many, many ways to compute factorials of numbers greater than 69 . . . and we've probably seen all of them . . .. This routine was contributed by two people, **Kent Krumvieda** and **Mike Daniels**, of Boulder Colorado.

(V2N3p9c) Recently, we were going through our old KEY NOTES and we noticed the HP-97 routine, written by **James Grandstaff,** that computes factorials of numbers larger than 69. We decided to rewrite it using the capabilities of the HP-41. The following is a listing of this revision.

```
01◆LBL "FAC"    19 LOG
02 STO 01       20 ST+ 00
03 70           21 DSE 01
04 X<=Y?        22 GTO 00
05 GTO 01       23◆LBL 02
06 X<>Y         24 RCL 00
07 FACT         25 INT
08 RTN          26 LASTX
09◆LBL 01       27 FRC
10 .069         28 10↑X
11 ST+ 01       29 CLA
12 69           30 ARCL X
13 FACT         31 "⊢ E"
14 LOG          32 FIX 0
15 STO 00       33 ARCL Y
16◆LBL 00       34 AVIEW
17 RCL 01       35 FIX 5
18 INT          36 END
```

Kongsberg is a town of about 20,000 people, just south of Drammen in Norway. It is the home of **John Erik Setsaas,** and this is his tip.

(V5N3p14b) Sometimes you need a routine that changes a number into a corresponding character. For example: **Fred Scheifele's** routine requires that 26 characters are stored, one in each register. This occupies 182 bytes of program memory. If you have a printer available, there is a much shorter way to do this.

```
01◆LBL "D-C"
02 0
03 X<>Y
04 BLDSPEC
05 END
```

Just put the number of the character (according to the list on page 37 in the printer handbook) into the X-register, and execute "D-C." The character will then be in the X-register.

Next, here's a nice letter from **Axel Harvey**, of La Macaza, Canada.

(V6N2p14b) In the last issue your editor suggests assigning the functions of the top two rows on the HP-41 to themselves: X<>Y to the X<>Y key, RDN to the RDN key, and so on. It's well worth repeating. This is the best way to avoid waiting forever for a RDN when your program pointer is in the middle of an undocumented 850-byte masterpiece (you wish to inspect the stack without losing your place), and it's 2 A.M. (At 2 A.M. the 4-second wait is forever!)

Of course, a simpler way is to get out of USER mode. The trick here is to remember to get back into it after you have seen the stack, so that your next keystroke won't be a disaster. This is not always possible at 2 A.M.

In the last issue of KEY NOTES (V6N3p13a), we printed what we claimed to be the shortest sequence of program lines to round-up in absolute value any number. This sequence was — INT; LAST X; FRC; X=0?; SIGN; +. Since then, we have received several inputs claiming to be shorter versions of this sequence, but these only made us realize that we hadn't done a very good job of explaining what we meant by "rounding-up a number." What we meant is this: if an integer (whole number) is input, the sequence will return that integer; otherwise, if the input is not an integer, it will be rounded to the next (greater in absolute value) integer. Thus, an input of 0 will return 0, an input of 3 will return 3, and an input of −3 will return −3. And, an input of .0004 will return 1, an input of −.002 will return −1, an input of 3.67 will return 4, and an input of −4.033 will return −5. Though we thank everyone who sent in a "shorter version," we have yet to receive a shorter version — than ours — that fulfills the above description of "round-up."

Now, we bring you some adaptably useful feedback from **Lawrence Vassallo**, who lives in Rochester, Michigan.

(V6N3p12c) In response to **Dr. Keith Bernstein's** submission to "Routines, Techniques, Tips, Etc . . .," I have enclosed my own HP-41 subroutine "PRC" for producing column-formatted printer output without use of the HP 82180A Extended Functions/Memory Module.

```
01♦LBL "PRC"     16 SKPCHR
02 RND           17 RDN
03 CF 00         18 CLA
04 X<0?          19 ARCL X
05 SF 00         20 FS?C 00
06 ABS           21 GTO 00
07 ENTER↑        22 "⊦ "
08 X≠0?          23 ACA
09 LOG           24 RTN
10 INT           25♦LBL 00
11 X>0?          26 CHS
12 ST- Z         27 "⊦-"
13 X<> Z         28 ACA
14 1             29 END
15 -
```

To call "PRC," the main program must first set the proper number of decimal places (FIX #) and clear flag 29. Then, the main program must place the number of digits to the left of the decimal point in the Y-register, the number to be output in the X-register, and XEQ "PRC." The value in the X-register will be accumulated into the print buffer with a trailing sign. PRC may be called multiple times before printing the formatted line with PRBUF or ADV. The method used for formatting is not iterative and requires only the stack registers, the ALPHA-register, and temporary use of flag 00. The X-register contents are preserved.

Here are some comments about the last issue of HP KEY NOTES from **Jeffrey Smith**, of La Palma, California.

(V6N3) Richard• **Partridge's** "HP-35" display simulator (V6N3p11c) fails for values that overflow or underflow into scientific or engineering notation (or if the calculator is already set to SCI or ENG). It's limitations should be clearly understood by anyone who wishes to use it to obtain "suppressed zero" formatting.

Ed Keefe's method of establishing FIX/ENG notation (V6N3p12a) has two flaws: (1) The flag range of 41.043 should have only two significant figures in the fractional portion (that is: 41.43), and (2) since only flag 41 needs to be set, a value of just 41 is sufficient for a range specification.

(The modified sequence would be: ENG n; RCLFLAG; FIX n; 41; STOFLAG — Ed.)

Routines, Techniques, Tips, Etc . . .

The routines and techniques furnished in this column are contributed by people from all walks of life and with various levels of mathematical and programming skills. While the routines might not be the ultimate in programming, they do present new ideas and solutions that others have found for their applications. *You might have to modify them to fit your personal application.*

The Extended Functions/Memory Module has quickly become a strikingly popular addition to the HP-41 system. In Houston, Texas, **Bill Rudersdorf** is using his Extended-Functions/Memory Module to make life just a little bit easier for all of us.

(41) Here's a quick and useful routine for finding the number of bytes in a program. It illustrates a few uses of the X-Functions Module. I've been using an HP-41 for almost three years now.

```
01♦LBL "BYTES"   16 FC?C 25
02 FIX 0         17 GTO 00
03 SF 21         18♦LBL 02
04 CF 29         19 SF 25
05♦LBL 00        20 RCLPTA
06 CF 00         21 FC?C 25
07 "NAME?"       22 GTO 01
08 AON           23 FS?C 00
09 PROMPT        24 PURFL
10 AOFF          25 "⊦:"
11 GTO 02        26 ARCL X
12♦LBL 01        27 "⊦ BYTES"
13 SF 00         28 AVIEW
14 SF 25         29 GTO 00
15 SAVEP         30 END
```

Near the border of South Australia and New South Wales, on the banks of the Murray River, lies the small town of Renmark. And, in this small town, we know of at least one person, **Chris Tolley**, who owns an HP-41, a printer, and an Extended Functions/Memory Module. Here's Mr. Tolley's contribution.

(41) Here is a routine to print the contents of an ASCII file matched, record to register, to the contents of a data file. The output will appear on one line, assuming the total number of characters in the ASCII record and the data register does not exceed 21 (10 if flag 12 is set). Execution will halt when the END of either file is reached.

```
01♦LBL "MATCH"   20 CLA
02 0             21 ARCL 01
03 STO 00        22 RCL 00
04 STO 01        23 SEEKPTA
05 STO 02        24 GETREC
06 "FILE MATCH"  25 ACA
07 PRA           26 2
08 ADV           27 SKPCHR
09 "ASCII FILE?" 28 CLA
10 AON           29 ARCL 02
11 PROMPT        30 RCL 00
12 ASTO 01       31 SEEKPTA
13 AOFF          32 GETX
14 "DATA FILE?"  33 ACX
15 AON           34 ADV
16 PROMPT        35 ISG 00
17 ASTO 02       36 X<> X
18 AOFF          37 GTO 01
19♦LBL 01        38 END
```

Here's another impressive routine that requires the Extended Functions Module. It was contributed by **Harold Schumann**, of Munster, Germany.

(41) The new Extended Function/Memory Module is really an incredibly powerful means for advanced programming. Recently, I have written a short routine that replaces single characters in the ALPHA-register by other single characters. It can easily be seen that the application of Extended Functions facilitates this task.

```
01♦LBL "REPA"    11 XTOA
02♦LBL 01        12 X<> Z
03 AVIEW         13 1
04 ENTER↑        14 +
05 POSA          15 CHS
06 X<0?          16 AROT
07 RTN           17 RDN
08 AROT          18 GTO 01
09 ATOX          19 END
10 R↑
```

How to use REPA:

1. Place the equivalent code of the character to be replaced into the X-register and the code of the replacing character into the Y-register.

(Continued)

2. Place any string into the ALPHA-register.

3. XEQ "REPA"

The "old" character will be replaced by the "new" one at each occurrence in the string.

(If, instead of the numerical character codes, you would rather ASTO the respective ALPHA characters in the X- and Y-registers, this routine will still work. You'll notice, Mr. Schumann, that I replaced the global GTO "REPA" that you had at the end of this routine with a local GTO 01 and LBL 01 at line 02. This conserves bytes and search time — Ed.)

Not far from Munster, Germany, on the north coast of Germany, is a town called Cuxhaven. Cuxhaven is the home of **Gunter Merten,** another owner of an Extended Functions/Memory Module.

(41) Following the operation GETKEY, I needed the number 0, 1, 2,..., 8, 9 in the X-register instead of the key code for the keys 0 to 9. I found two short, quick routines to implement this. The first (labeled KEYNUM) returns a −1 to the X-register if any key besides keys 0 through 9 is pressed. The second (labeled KEYNM) is shorter but it doesn't account for the user pressing a "non-number" key.

```
01♦LBL "KEYNUM" 01♦LBL "KEYNM"
02 "RHIJ>?"     02 GETKEY
03 64           03 45
04 XTOA         04 MOD
05 "⊢456"       05 13
06 GETKEY       06 MOD
07 POSA         07 11
08 END          08 MOD
                09 END
```

(The "goose" will pause in the display during the execution of these routines. It is just waiting for you to press a key — Ed.)

This next routine is being printed with the permission of **Richard Nelson** of PPC*. The routine was published on the last page of the *PPC Calculator Journal* V9N4, and it struck us as being pretty clever.

(41) Recall sigma is a routine to recall the SIGMA-X and SIGMA-Y values from the statistical registers when the location of these registers is unknown. Lines 02 through 05 can be omitted if n is still in the X-register from a previous SIGMA+ or SIGMA−.

```
01♦LBL "RΣ"     06 MEAN
02 CLST         07 LASTX
03 Σ+           08 ST* Z
04 CLST         09 *
05 Σ-           10 END
```

**Founded in June 1974 by Richard J. Nelson, PPC is the world's first and largest organization dedicated to Personal Programmable Calculators. The Club is a volunteer, non-profit, loosely organized, independent, worldwide group of Hewlett-Packard personal programmable calculator users. PPC is not sponsored, nor in any way officially sanctioned, by Hewlett-Packard.*

For more information about PPC and a sample issue of the Club's newsletter, send a self-addressed, large (folded) envelope (9 X 12 inches; 23.8 X 30.5 cm) with first-class postage for 2 ounces (56.7 grams) to: PPC Calculator Journal; 2545 W. Camden Place; Santa Ana, California 92704 U.S.A. If you live outside the U.S., make sure you include a legible address label and international postal coupons for 56.7 grams (2 ounces). A letter is not necessary and will only slow the response.

What is the biggest number you can think of? What is the biggest number your HP-41 can think of? Is it around 10^{100}? If "yes" is your answer, **Thomas Excelsior Valere,** of Paris, France, may beg to differ.

(41) Sometimes you need to calculate values for large exponents like 41^{67}, but $41^{67} > 10^{100}$, so the HP-41 cannot compute it with the Y^X function. So, try this lifesaving routine based on the formula $Y^X = 10^{Y \text{ LOG } X}$. It works just like the Y^X function, except that x is not stored in the LASTX-register. The resulting mantissa will be in the X-register, and the respective exponent in the Y-register.

```
01♦LBL "Y**X"   06 LASTX
02 X<>Y         07 FRC
03 LOG          08 10↑X
04 *            09 END
05 INT
```

(This also works for negative exponents. For example, 86^{-85}. But the true question is: excepting the human imagination, is there anything in the universe that demands a number greater than 10^{100} or less than 10^{-100}? And, if there is, perhaps we should switch to an HP-75C, which flaunts a numerical range from greater than 10^{-500} to less than 10^{500} — Ed.)

Peter Rushworth, who lives in Lakewood, Colorado, contributed this routine to KEY NOTES. It is a "sort of humorous" application of the Time Module.

(41) Here is an HP-41 routine for use with the Time Module. This program will display the increasing labor costs of a business meeting, given the number of people and their labor rate in dollars per hour. The routine requires a minimum SIZE of 003.

```
01♦LBL "MT"        13 GTO 01
02 0               14 "RATE?"
03 STO 00          15 PROMPT
04 STO 01          16 *
05 SF 27           17 ST+ 01
06 CF 21           18 GTO 00
07 FIX 0           19♦LBL 01
08♦LBL 00          20 FIX 2
09 CF 22           21 "    READY"
10 "NO. PEOPLE?"   22 AVIEW
11 PROMPT          23♦LBL 02
12 FC?C 22         24 STOP

25 GTO 02          34 HR
26♦LBL A           35 RCL 01
27 CF 27           36 *
28 TIME            37 "    $"
29 STO 02          38 ARCL X
30♦LBL 10          39 AVIEW
31 TIME            40 GTO 10
32 RCL 02          41 END
33 HMS-
```

Example Problem:

Number of People	Labor Rate ($/hr)
3	8.50
5	12.75
1	17.50

Solution:

Input	Display
XEQ "MT"	NO. PEOPLE?
3 [R/S]	RATE?
8.50 [R/S]	NO. PEOPLE?
5 [R/S]	RATE?
12.75 [R/S]	NO. PEOPLE?
1 [R/S]	RATE?
17.50 [R/S]	NO. PEOPLE?
[R/S]	READY

At the start of the meeting, press [A] and the cost of the meeting will accumulate in the display.

C. Lamar Williams lives in San Jose, California. Lately, Mr. Williams has been thinking of ways that we can save money.

(41) Budget minded grocery-shoppers must repeatedly choose the least costly (lowest unit cost) of equivalent items. The classic decision is exemplified by: "13 ounces at $1.63 for brand A, or 17 ounces at $2.15 for brand B — which is cheaper?" And, it is easily solved by the routine "SHOP." Just execute "SHOP," at the prompt: A=?, ENTER the cost (1.63 [ENTER]), key-in the quantity (13), and press R/S. Do the same at the prompt: B=?. The appropriate answer (A IS CHEAPER, B IS CHEAPER, OR A = B) will be displayed. Try it! Note that A IS CHEAPER for the above example.

```
01♦LBL "SHOP"        10 AVIEW
02 "A"               11 RTN
03 XEQ 00            12♦LBL 00
04 XEQ 00            13 "⊢=? $↑QUAN"
05 X>Y?              14 PROMPT
06 "A"               15 /
07 "⊢ IS CHEAPER"    16 "B"
08 X=Y?              17 END
09 "A = B"
```

Many HP-41 owners have an interest in music. This routine was submitted to KEY NOTES by one of these music-loving readers. **Paulo de Salles Mourao** lives in Belo Horizonte, Brazil and this is his routine.

(41) In my spare time (and when not programming), I like to play the violin (for internal consumption, for that matter). Today I was considering adjusting my tempo, when I realized that my daughter had borrowed my metronome some days ago. But, I am an HP-41 owner, and this type of person has many resources. Why not write a metronome program?

```
01♦LBL "METR"    10♦LBL 02
02 "INDEX"       11 DSE 01
03 PROMPT        12 GTO 02
04 X=0?          13 TONE 9
05 GTO 03        14 GTO 01
06 STO 00        15♦LBL 03
07♦LBL 01        16 TONE 9
08 RCL 00        17 GTO 03
09 STO 01        18 END
```

On XEQ "METR", you have access to the "INDEX" entry. Zero is *vivace,* 1 is *vivace non troppo,* 4 is *allegro,* 7 is *adante,* and 10 is *moderato* (incidentally, the goose flies backwards), but the nuances I leave to the *maestros.* The routine is stopped manually ([R/S]). It is another unusual application of the HP-41, *n'est-ce pas?*

Along these same lines, we have this routine that was contributed by **Tom Nguyen,** of Issaquah, Washington. This routine can be used on any HP-41.

(41) This routine is used when transposing music from one key to another according to the interval of half-steps specified by the user. The interval is specified only once for successive notes. The notation used by the program is best explained by referring to the piano keyboard. The white keys are represented by capital letters (C, D, E, F, G, A, B,) and the black keys are represented by the small letter that corresponds to the flat (not sharp) note of that key. Thus, D-flat, E-flat, G-flat, A-flat, and B-flat are represented by d, e, g, a, and b respectively. (The ALPHA character "9" is used for a small g.)

```
01♦LBL "TR"      23 ASTO 11
02 "A"           24 "a"
03 ASTO 01       25 ASTO 12
04 "b"           26♦LBL 05
05 ASTO 02       27 12
06 "B"           28 "INTERVAL"
07 ASTO 03       29 PROMPT
08 "C"           30 X<=Y?
09 ASTO 04       31 GTO 04
10 "d"           32 GTO 05
11 ASTO 05       33♦LBL 04
12 "D"           34 STO 00
13 ASTO 06       35♦LBL 03
14 "e"           36 "NOTE:"
15 ASTO 07       37 AON
16 "E"           38 STOP
17 ASTO 08       39 ASTO 14
18 "F"           40 AOFF
19 ASTO 09       41 1.012
20 "9"           42 STO 13
21 ASTO 10       43♦LBL 01
22 "G"           44 RCL 14

45 RCL IND 13    54 RCL 00
46 X=Y?          55 +
47 GTO 02        56 12
48 ISG 13        57 X<Y?
49 GTO 01        58 ST- Y
50 GTO 03        59 RCL IND Y
51♦LBL 02        60 PSE
52 RCL 13        61 GTO 03
53 INT           62 END
```

The specified interval always must be positive. For example, to transpose from the key of G to the key of C, specify an interval of 5 (the number of half-steps between the roots of the two scales). At the prompt "NOTE:," key in the letter that corresponds to the note you wish to shift. Shifting a note down an interval X is the same as shifting it up an interval 12 - X.

Jim Boardman, of Tucson, Arizona, sent us this next routine and with his letter he included the following P.S.: "I am an avid reader of HP KEY NOTES and I feel as though I do a lot of *taking* of information from its pages. Here, hopefully, I have done a small amount of *giving.*" Yes you have, Mr. Boardman, and here is your contribution.

(41) Occasionally, I find a need for saving the current date and time. The Time Module is very handy for providing this data. Normally, two registers are required for saving the information. But, by making use of two other Time Module functions, the data can be packed into one register and later re-expanded into two registers.

```
01♦LBL "PAKDTM"  13 RTN
02♦LBL 00        14♦LBL "RESDTM"
03 23.5958       15 FRC
04 TIME          16 1.01198
05 X>Y?          17 LASTX
06 GTO 00        18 INT
07 1 E2          19 DATE+
08 /             20 X<>Y
09 1.01198       21 1 E2
10 DATE          22 *
11 DDAYS         23 END
12 +
```

"PAKDTM" will pack the current date and time into the X-register. A routine calling "RESDTM" with such a packed number in X will receive back the original date in Y (in the form MM.DDYYYY or DD.MMYYYY) and the original time in X (in the form HH.MMSS). If you are not concerned about the date changing at midnight while "PAKDTM" is executing, lines 02, 04, and 05 of that routine may be deleted.

(You'll note, Mr. Boardman, that we replaced that 8-byte GTO "PAKDTM" with a GTO 00 (2 bytes) and we added a LBL 00 (1 byte) after LBL "PAKDTM", thus, conserving 5 bytes. For more information on this, see "Add a Local Label," KEY NOTES V6N1p7b — Ed.)

Aston, Pennsylvania, is the home of **Vasant Patel,** who sent us this next contribution. The routine will be of interest to those who are developing programs that solve systems of linear equations.

(41) Here is a short, efficient subroutine for solving n simultaneous equations using the Gauss-Jordan method. Prior to accessing this routine and starting at register 21, the n coefficients are to be stored row-by-row, with each row followed by its corresponding constant. You can change the starting register by changing line 04. The value of n must be contained in register 00. At the end of execution, the constant vector is replaced by the solution. Note that in addition to the matrix elements, registers 01 through 08 are altered.

```
01♦LBL "LIN"     31 RCL 02
02 RCL 00        32 +
03 STO 07        33 STO 03
04 21            34 RCL 01
05 STO 08        35 X=Y?
06 +             36 GTO 15
07 STO 04        37♦LBL 14
08♦LBL 11        38 RCL IND 01
09 RCL 04        39 RCL IND 02
10 RCL 07        40 *
11 -             41 ST- IND 03
12 RCL 04        42 1
13 3             43 ST- 01
14 10↑X          44 ST- 03
15 /             45 DSE 05
16 +             46 GTO 14
17 RCL IND X     47♦LBL 15
18♦LBL 12        48 RCL 00
19 ST/ IND Y     49 1
20 ISG Y         50 +
21 GTO 12        51 ST+ 02
22 RCL 08        52 DSE 06
23 STO 02        53 GTO 13
24 RCL 00        54 ST+ 04
25 STO 06        55 1
26♦LBL 13        56 ST+ 08
27 RCL 04        57 DSE 07
28 STO 01        58 GTO 11
29 RCL 07        59 END
30 STO 05
```

Now, here's a tip that was sent to us by **Paul Murteira,** of Lisboa, the capital of Portugal. This is a work-conserving suggestion.

(41 and Card Reader) I have a suggestion that might help users to find how many tracks will be needed to record a program onto cards. (1) Assign PACK and WPRV to HP-41 keys. (2) Key-in the program and leave the HP-41 in program mode. (3) Press [PACK] and then [WPRV] in user mode, and the display will show "RDY kk of nn." (4) If you don't want to record in private, simply press the backarrow key, then insert the cards to record your program.

(Continued)

If you want to reduce the number of tracks required, then shorten a long ALPHA message or shorten any long ALPHA labels, and repeat from step (3). This procedure will conserve your batteries because you won't have to insert a card to find how many tracks are needed.

For example, solve the two equations:

2x + 6y = 27

5x + 9y = 13

Store, in registers 21 through 26, the numbers 2, 6, 27, 5, 9, and 13 in that order. Also, store 2 in register 00. After executing the routine, the solution to x will be found in register 23 and the solution to y will be found in register 26.

(This routine is, just as Mr. Patel says, short and efficient. It does not rearrange the rows of the matrix to obtain the largest absolute pivot values and, thus, it does not minimize rounding error. It does not test for singularities or estimate the numerical condition of the matrix. So, any program that calls this as a subroutine should take these things into account, beforehand — Ed.)

Now, we have a fine contribution from **John Chaffer**, of Seattle, Washington. Mr. Chaffer presents a great idea that many of you will be able to build on.

(41) Here is a short routine I wrote to print a single byte of "WNDSCN"-type barcode on the HP 82143A (non-HP-IL) printer. Decimal integers from zero to 255 (with no checksum) can be printed in barcode. Black printer paper is needed, and the darker settings seem to increase the reading reliability. The barcode is printed so that the stop-bits come before the start-bits, but the Wand doesn't care. To view the results, use the routine "SCN."

```
01♦LBL "BC1"    21 2
02♦LBL 00       22 *
03 1.008        23 X≠0?
04 STO 00       24 SF 00
05 ADV          25 XEQ 01
06 SF 12        26 ISG 00
07 CF 00        27 GTO 02
08 "NUMBER?"    28 XEQ 01
09 PROMPT       29 XEQ 01
10 ENTER↑       30 GTO 00
11 XEQ 01       31♦LBL 01
12 SF 00        32 127
13 ENTER↑       33 ACCOL
14 XEQ 01       34 FS?C 00
15♦LBL 02       35 ACCOL
16 2            36 1
17 /            37 SKPCOL
18 INT          38 RCL T
19 LASTX        39 RTN
20 FRC          40 END
```

```
01♦LBL "SCN"    05 VIEW 01
02♦LBL 00       06 PSE
03 CLD          07 GTO 00
04 WNDSCN       08 END
```

Surely, the esthetics of the next routine will brighten your day. It makes good use of the "input during pause" feature of the HP-41. This routine was contributed by **Jean-Luc Marechal**, of Liege, Belgium.

(41) This idea could help some KEY NOTES readers. It is a short routine that can be used in place of PROMPT or STOP. This routine tests to see if the input is to be an ALPHA or a numerical input (checks ALPHA-mode flag 48). Then, after pausing for the input, it returns to the main program.

```
01♦LBL "IN"     08♦LBL 00
02 23           09 PSE
03 FC? 48       10 FC? IND L
04 22           11 GTO 00
05 STO L        12 AOFF
06 CF IND L     13 END
07 AVIEW
```

New Orleans, Louisiana, is a city known for its high-energy festivities. And no doubt, part of the celebrant atmosphere found in this city is attributable to the elated HP-41 owners who live there. **Edward Scheinuk** is an HP-41 owner who lives in New Orleans, and this is his contribution.

(41) Having gone through the experience of having someone erase a brilliantly conceived program — "I just pressed a few keys, honest" — that was not yet recorded on cards, I am fully aware of the need for "guard" routines as mentioned in V6N3p11. Thus, I would like to contribute my own version. It requires an HP 82182A Time Module.

```
01♦LBL "Z"      01♦LBL "ZZ"
02 SF 11        02 DATE
03 "↑↑ZZ"       03 DOW
04 CLST         04 X≠Y?
05 .0004        05 GTO "Z"
06 OFF          06 END
07 TIME
08 +
09 XYZALM
10 END
```

After "Z" has been executed to turn-off the HP-41, the user must key-in the current DOW (day of week) within a short time after pressing the [ON] key or the HP-41 will not remain ON. The time interval can be adjusted by changing line 05 (I suggest starting with a 4-second time interval). There are only three methods (that I know of) of circumventing the workings of the routine and all three would be obvious only to someone who is experienced in the use of the HP-41.

Fredrikstad, in southern Norway, is the origin of this routine by **Hans Aspenberg**. The subject addressed by this routine is "sort-of" popular.

(41) Here is a cute sorting routine for the HP-41. Key-in the block of data-registers to be sorted using the format bbb.eee, where bbb is the begin-address and eee is the end-address of the block. Then XEQ "SORT." Note that bbb >= 3 because the routine uses registers 00, 01, and 02 for pointers. SORT is short, using only 49 bytes, and it is surprisingly fast!

```
01♦LBL "SORT"   13 GTO 02
02 STO 00       14 RCL 01
03♦LBL 00       15 STO 02
04 RCL 00       16♦LBL 02
05 STO 02       17 ISG 01
06 1.001        18 GTO 01
07 +            19 RCL IND 00
08 STO 01       20 X<> IND 02
09♦LBL 01       21 STO IND 00
10 RCL IND 02   22 ISG 00
11 RCL IND 01   23 GTO 00
12 X<=Y?        24 END
```

(The bubble sort — the method used by this routine — is the traditionally chosen method used on computers for sorting data. An alternative method that is rarely used in practice, utilizes, for each successive element, a binary-search to locate the position of that element in the previously ordered elements. This binary-search method greatly reduces the number of comparisons involved. However, on most computers the additional matrix-index management required to insert an unordered element into its proper position in the previously ordered elements is extensive. It is for this reason that the binary-search method has been neglected. But now the REGMOVE function in the Extended Functions/Memory Module makes it easy to insert a value anywhere in a block of registers. Thus, at the risk of dropping a hint, perhaps the REGMOVE function makes the binary-search method a viable method for sorting data on the HP-41 — Ed.)

Language Name Delayed . . .

Many of you are on the edge of your seat just waiting to see what name we choose for the HP-41's language and to see who won the contest (V6N3p16a), but we have some bad news for you. Because some of the most popular names we chose were already being used by other companies, and because we want to make sure that there is no infringement on trade marks or trade names, we are delaying the announcement of a winner until the next issue of KEY NOTES, which will be mailed the third week of February 1983. So hang onto your seat until then!

NEW HP-75C PAGES

Turn to page 16 for the first edition of HP-75 KEY NOTES. There are four extra pages just for this latest star in the HP sky.

Book Reviews

Books are reviewed or announced in KEY NOTES only as a service to our readers. A review here does **not** represent an endorsement by Hewlett-Packard. If you are unsure about the contents or usefulness of a book, we suggest you first check with a local bookstore; if that fails, write to the publisher, not to KEY NOTES. Availability problems also should be addressed to the publisher, not to KEY NOTES.

CALCULATE BASIC STATISTICS, by **Mark Finkelstein** and **George McCarty,** is a new 352-page softbound book published in October. ISBN number is 0-936356-01-04 and the size is 5.4 by 8.5 inches (13.7 by 21.6 cm).

This book represents a friendly new calculator method for learning statistics, even if you are a novice with the subject or studying the subject in school. It features interesting, realistic examples that you follow step-by-step — instead of a lot of algebra. There are straight-to-the-point explanations for such topics as "confidence intervals" and "contingency tables" that are so confusing in ordinary textbooks. Also, any calculator or "home" or "personal" computer will do fine to make the calculations.

It's all here, including Mean and Standard Deviation, Sample Means, Inference and Hypothesis Testing, Chi-Square Distribution — the whole gamut of statistics. Plus, there are two bonuses at the end: a new type of table simplifies frequently used tests, and a BASIC program is given for multiple linear regression.

The authors are both faculty members at the University of California, Irvine. And you will remember that another of George McCarty's books, *Calculator Calculus,* appeared before in KEY NOTES.

This new book should soon be in book stores and is available now from the publisher, below. You also should check with your authorized HP Dealer; many of them are now stocking books reviewed in KEY NOTES — or they will order them for you. The price is $14.95 plus any state or local taxes. Add $2.50 per book for air mail in the U.S.; $3.50 for air mail to Canada and Mexico; and $6.00 for air mail to all other countries. The publisher is:

EduCALC Mail Store
27953 Cabot Road
Laguna Niguel, CA 92677 U.S.A.

HP-41 SYNTHETIC PROGRAMMING MADE EASY, by **Keith Jarett**, is another new book just off the press in October. It is spiral-bound, 190 pages, and 6.75 by 8.5 inches (17.2 by 21.6 cm) — roughly the same size as the HP-41 owner's handbook.

This is the *second* "synthetic programming" book to appear in KEY NOTES, the first being *Synthetic Programming on the HP-41,* by **Dr. William C. Wickes** (see V4N3p8 and V6N1p7c).*

For the uninitiated, here is the definition of synthetic programming — or SP — from Jarett's new book: "Synthetic programming encompasses the creation and use of synthetic instructions. Synthetic instructions cannot be keyed into program memory by normal means, but must be created by splicing normal instructions. Synthetic programming will work on any HP-41, does not require any modification of the calculator, and will not harm it. Thousands of HP-41 owners have used synthetic functions to obtain extended key assignment capability, 21 additional display characters, over 100 additional TONEs, extra 'scratchpad' registers, improved control of lowercase printer characters, and much more."

Because a great many of you are interested in SP, and because the Corvallis Library now accepts *limited* SP in programs (see V6N1p7c), it is very likely that some of you will be interested in this new book. It provides an introduction to SP, using recently developed techniques that make SP even easier than before. It is full of examples and step-by-step instructions. Also included are the latest "byte-grabbing" and "byte-loading" techniques for simple, fast creation of synthetic codes. Plus, you will find applications and utility programs for the Extended Functions and Time Modules. Also included in the book is a separate, handy, three-color plastic Quick-Reference Card for Synthetic Programming, a $3 value. This card contains a byte equivalence table that is the "Rosetta Stone" of SP. It also shows all 56 flag functions and incorporates a quick-reference summary of how to construct each type of synthetic instruction.

Keith Jarett is an inveterate "key-puncher," a steadfast member of the PPC (see footnote on page 12), and is, in his words, "addicted to HP calculators since I bought an HP-45 in 1973 while an undergraduate in Electrical Engineering." He graduated from Culver Military Academy in 1972 and received his B.S. degree in Electrical Engineering from Cornell in 1975, his M.S. degree in Electrical Engineering from Stanford in 1975, and his Ph.D. in EE from Stanford in 1979. He's currently a Systems Engineer at Hughes Aircraft. This book was started in January of this year, and here is how you can obtain a copy; write to:

SYNTHETIX
1540 Matthews Ave.
Manhattan Beach
CA 90266 U.S.A.

or you can check at your local bookstore or authorized HP Dealer.

Prices, according to actual mailing costs for various places throughout the world, are:

$16.95 to U.S., shipped fourth class.
$18.45 to U.S., priority mail or UPS.
$18.95 to Canada, Mexico, or Central America, air mail.
$20.45 to Europe or South America, air mail.
$21.95 elsewhere, air mail.

Be sure to add state or local taxes (California residents add $1.10 tax), and checks or money orders must be payable through a U.S. bank.

**Hewlett-Packard does not "support" synthetic programming. The HP-41 series hand-held Computers are only guaranteed to operate in accordance with the instructions and information provided in the* HP-41C/CV Owner's Handbook and Programming Guide. *Do not refer questions about synthetic programming to the factory. References for SP can be found in V6N1p8b.*

Editorial

Although this column has been missing for a while, your "Ed." is still alive and kicking. Some recent letters have asked why the editorial staff is not listed on the back cover, so we changed that to satisfy those who would like to see our names.

Because we have to make a "trade name search" to make sure that the name we pick as the winner of the "Name That Language" contest is not an infringement on the registered name used by someone else, we will not announce the winner until the February issue. So, if you sent an entry from a faraway place and worried about it getting in the contest, stop worrying; I included all of them. But, please, NO MORE! The contest is officially closed.

If you use your calculator or computer or any software or peripherals to increase your productivity in your business or at your job, you really should look into the probability that you can deduct those expenses from your income tax. This can save you a lot of money ... and then you can go see your local authorized HP Dealer and buy yourself a nice Christmas present.

A lot of you faithful readers will soon be receiving KEY NOTES subscription renewal notices from the friendly computer, and I sincerely hope you decide to renew. We are going to get more information into KEY NOTES next year, and a lot of it will deal with all the peripherals and the Interface Loop. At only $5 a year for the U.S. and Canada, you have to admit that KEY NOTES is a real bargain. For example, with 64 pages a year, that works out to 7.8125 cents a page. And although I've never accurately counted all of them, there are about 1400 words per page, which works out to somewhere over 180 words for each U.S. penny you spend on KEY NOTES. So make your tired old editor happy by sending in your renewal on time; then he won't have to get the computer on your trail. By the way, your expiration date is at top right on your label.

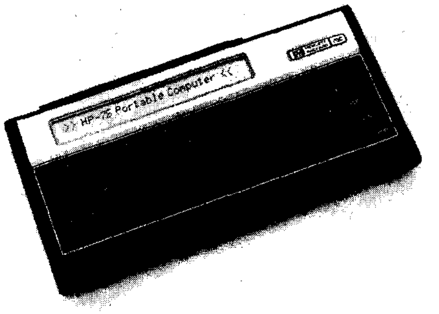
Letters to the editor should be sent to:

Henry Horn, Editor
HP KEY NOTES
Hewlett-Packard Co.
1000 N.E. Circle Blvd.
Corvallis, Oregon 97330 U.S.A.

We cannot guarantee a reply to every letter, but we do guarantee that every letter will be read by the editor or technical editor, and that as many as possible will be answered in KEY NOTES or in a personal response. Be sure to put your address *on the letter*; sometimes the envelope gets lost.

This is the ninth year that it is my pleasure to thank all of you for your support of KEY NOTES, for your steadfast participation, and for all your nice comments about this newsletter. So, on behalf of my assistant, **Ted Wadman**, and all the staff in the Users' Library, plus all of Hewlett-Packard, I wish every one of you, all over the world, a happy, safe, and joyous holiday season; a prosperous New Year; and many happy hours of productive programming ... and that you get a subscription to KEY NOTES for Christmas.

HP-75C Computer KEY NOTES



You will notice that there are 20 pages in this issue of KEY NOTES. That's because there are 4 extra pages devoted to the new HP-75C Portable Computing System. We are NOT going to usurp pages from the original KEY NOTES. You will find 16 pages, as in the past, devoted to the HP-67/97, and to the HP-41 system.

Because there are not many HP-75C's in the hands of users as yet, we have not had much feedback from them — except to tell us that they love the HP-75C. So we have written a couple of articles about the HP-75C to give you a head-start on using it, and this will also show our other KEY NOTES readers something about the world of computers and programming in BASIC. If you have some comments or ideas, we would like to hear from you.

Testing the HP-75C

Testing is an important part in the development of a new product.

Furthermore, thorough testing benefits both the customer and Hewlett-Packard, because the customer can purchase a new machine knowing that it was built to last, and HP saves the time and money that would have been spent repairing returned units.

The HP-75C had to go through a rigorous series of tests before it was released to production.These tests included the standard tests for HP calculators: line voltage variations, storage temperature range, operating temperature range, magnetic susceptibility, humidity, ESD (electrostatic discharge), vibration and shock, conducted and radiated RFI (radio frequency interference), 1-meter drop, condensation, supersoak (storage for 24 hours in a high-temperature, high-humidity environment), altitude, and a drop test of the final packaged product.

In addition, the HP-75C also was put through initial performance checks and 48 hours of preconditioning (temperature cycling) before the standard tests, and it also endured 6 weeks of temperature and power cycling with card reader checks after the tests. HP-75C's also were tested with X-ray levels hundreds of times more powerful than those in typical airport security systems to verify that users could take their portable computing machines with them without causing harm to the system.

Along the assembly line, each HP-75C receives a diagnostic test, a benchdrop test, and a card reader test. Following final assembly, 20% of the units go through temperature cycling, and a different 20% go through a shake test.

So, as you can see, the HP-75C has already proven itself before it even gets out of the factory, resulting in a better machine for you, and fewer repair bills for Hewlett-Packard.

Graphics with the HP-75C and HP 82905B Printer

The HP-75C can be used to do dot graphics on the HP 82905B (option 248) HP-IL Printer using simple PRINT statements. The PRINT statements control the dot patterns printed, and they also can be used to prevent the vertical spacing between lines of characters.

For example, let's look at creating an alphabet of letters whose size is three times larger than the standard printed characters. First, draw on graph paper what you will want the characters to look like. One pass of the printhead will print 8 dots, so the height in dots for each character will be 24 dots.

Here's an example for the letter 'A'.

```
........................x...............
.......................xxx................
......................xxxxx.............
.....................xxxxxxx..............
....................xxxxxxxxx...........
...................xxxxx.xxxxx............
..................xxxxx..xxxxx..........
.................xxxxx...xxxxx.............
................xxxxx.....xxxxx.......
...............xxxxx.......xxxxx.......
..............xxxxx.........xxxxx.....
.............xxxxx...........xxxxx.....
............xxxxxxxxxxxxxxxxxxxxxxxxx..
...........xxxxxxxxxxxxxxxxxxxxxxxxxxx..
..........xxxxxxxxxxxxxxxxxxxxxxxxxxxxx
.........xxxxx...................xxxxx...
........xxxxx.....................xxxxx
.......xxxxx.......................xxxxx
......xxxxx.........................xxxxx
.....xxxxx...........................xxxxx
....xxxxx.............................xxxxx
...xxxxx...............................xxxxx
..xxxxx.................................xxxxx
xxxxxxxx.................................xxxxxxxx
```

Now, let's look at just the first row of 8 dots. Wherever a dot is presented, we want a dot to be printed. Starting at the left, we will look at each column of dots as if it were a binary representation for a number, with the least significant bit on the bottom, and the most significant bit on the top.A dot means that that bit is set, and a space means that bit is zero.

Starting at the left in our example, for the first column we read (from the top down) 00000000 binary, or 0 decimal. The next column is also equal to 0, and so on until we get to column 18. In column 18, we read 00000001B, or 1 decimal. Column 19 equals 00000011B, or 3 decimal, and so on.

```
........................x...............
.......................xxx..............
......................xxxxx.............
.....................xxxxxxx............
....................xxxxxxxxx...........
...................xxxxx.xxxxx..........
..................xxxxx...xxxxx........
.................xxxxx.....xxxxx..........
```

After we have calculated the decimal values for all the columns in the first row, we will calculate the values for the columns in the second and third rows.

We are now ready to write a program to print our letter 'A' on the printer. For our example, we will assume that the printer is the only HP-IL device connected to the HP-75. The first thing to do is to assign the printer as the PRINTER IS device, and set the print width to infinite:

```
10 Assignio ':PR'
20 Printer is ':PR'
30 Pwidth inf
```

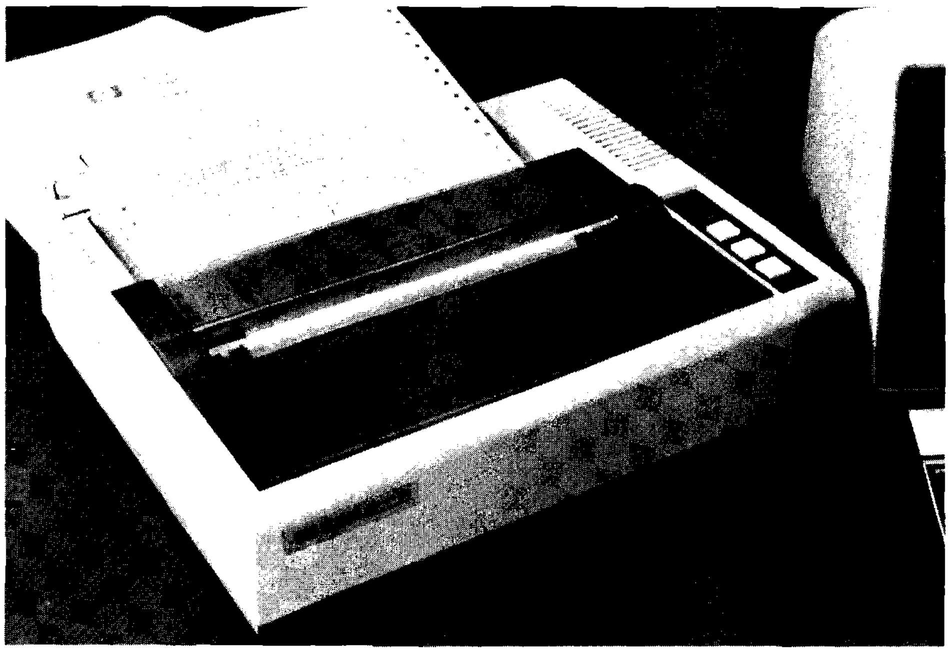
We now change the vertical spacing on the printer from 6 to 9 lines per inch, so we won't get a white space between lines of dots:

```
40 Print Chr$(27)&'&l9D';
```

(Notice that it is an 'l' and NOT A '1' in front of the 'D'.)

Since our letter has 49 dots across, we now tell the printer to expect 49 dots of dot graphics information:

```
50 Print CHR$(27)&'*b49G';
```



Now we send the decimal values of the dot columns for the first row:

```
60 For I=1 to 49
70 Read A @ Print Chr$(A);
80 Next I
90 Data 0,0,0,0,0,0,0,0,0,0
100 Data 0,0,0,0,0,0,0,1,3,7
110 Data 15,31,62,124,248,124,62,31,15,7
120 Data 3,1,0,0,0,0,0,0,0,0
130 Data 0,0,0,0,0,0,0,0,0
```

We now send a carriage return, and tell the printer to expect 49 more bytes of dot graphics information:

```
140 Print @ Print Chr$(27)&'*b49G';
```

The byte values for the second row are now sent, and the process is repeated for the last row of dots. We now finish our printing by printing a carriage return, setting the vertical spacing back to 6 lines per inch, and the width to 80 characters per line.

•

•

•

```
500 Print @ Print Chr$(27)&'&16D'
510 Pwidth 80
```

Some other uses for dot graphics include logos, signatures, personalization of memos and letters, and illustrating with graphs, maps, or charts.

The HP-75C Card Reader

The "hand-pulled" card reader is one of the most innovative features of the HP-75C. It is a small, light-weight, low-power design with no moving parts. Both the analog and digital circuits are contained on the same CMOS chip, including over 20 operational amplifiers. The card density is double that of the HP-67/97 cards.

All HP-75C cards have some prerecorded information on them, even the blank cards. If this information is erased, the card will be useless, so users must be careful about getting the cards near strong magnetic fields.

The information on the cards consists of a data track and timing track for each direction of recording. The data track contains four records:

(1) A card header block that holds information about the origin of the card (HP-Corvallis), and the maximum possible size of the card. This allows for possible future extensions in the card-handling capabilities of the HP-75C.

(2) A write-protect field that will prevent protected card tracks from being overwritten.

(3) A data header that identifies which file is stored on the card.

(4) A data field that holds up to 650 bytes of data from that file.

The timing track tells the hardware how fast the card is moving, enabling the hardware and software to accurately read the data from the card. The hardware can compensate for some variations in speed this way, allowing different users to pull cards at different rates, or even for a given pull to vary in speed.

Redefining Keys on the HP-75C

Another powerful feature of the HP-75C is the ability to redefine almost all the keys on the keyboard, and even some that cannot be accessed from the keyboard. The key redefinitions are kept in a text file, and can be listed or edited at any time. Key redefinitions can be added to the file by EDITing the file KEYS (notice that there are no quotes around the name for the keys file), or by executing a DEF KEY statement.

Editing keys, such as the left and right arrows or the insert/replace key, can be included in key redefinitions. For example, if you wanted to define [CTL][C] to print out: Copy card to '' and then backspace the cursor to the last quote and turn on the insert cursor, you could look up the decimal codes for [CTL][C], [←], and [I/R] in the reference tables in the HP-75C manual, and write the DEF KEY like this:

```
Def Key Chr$(3), "Copy card to" "''&Chr$
    (134)&Chr$(136);
```

Alternately, you could type in those keys directly by pressing [SHIFT] [I/R] (the 'literalize' sequence) before pressing the [I/R] and [←] keys. The literalized [I/R] key will show up in the display as a ≃, and the literalized [←] key will show up as a Γ.

```
Def key '←', "Copy card to 'Γ≃'";
```

The semicolon on the end of the DEF KEY means that the key will simply print the characters assigned to it and wait for the user to type. If there is no semicolon, the key would print out the string and do an immediate carriage return.

If you want to disable a particular key on the keyboard, you can assign a null string to it, followed by a semicolon.

The key definitions will stay in effect until a new definition is assigned or the KEYS file is purged. To get back the original definition for the key, press [SHIFT] [I/R] before pressing [CTL][C]. To permanently restore the original definition to the key, do a DEF KEY and assign the key back to itself. Remember to press [CTL][C] to literalize the key if you want the original definition.

Here's an idea for you to try. When the HP-75C times out to sleep after 5 minutes of inactivity, it does it by pretending key #160 [SHIFT][ATTN] was pressed. Try redefining that key to clear the TV screen before going to sleep, to run a program, or to display the time and stay awake.

Notes on HP-75C User Functions

User functions are normally used to compute a numeric or string value, based on parameters passed to the function. However, the multiple local parameters of the HP-75C's user functions allow another use that has some interesting possibilities. Functions may be used as a parameterized subroutine that does not destroy the parameters passed to it, and that returns a value simply because it has to.

For example, suppose we want a subroutine that prints on the display device all the integers between two numbers. For a normal GOSUB subroutine, this would require that the starting and ending values be assigned to two global variables, and that a GOSUB be done (i.e., X=5 @ Y=8 @ GOSUB 100). This requires that two variables must be tied-up by this call. Also, if the variables are changed by the subroutine, this change is reflected throughout the program. Using a user function for this subroutine (i.e. X=FNC(5,8)), means the values can be passed as parameters, and any changes in the subroutine are local and do not affect the rest of the program. However, the function must return a value, so in this case, we could return the count of numbers printed.

This technique can lead to more readable code, since functions can be written as self-contained modules with better defined inputs and outputs.

Multi-line string functions are limited to a returned string length of 32 characters. This can be gotten around in two ways: (1) single line definitions are not subject to this restriction, the value they return can be any length; (2) if you

(Continued)

cannot express your function as a single line, then the value can be returned in a global variable.

Global variables are normal BASIC variables, they can be accessed from anywhere in the program, and any change is reflected throughout the program.

Local variables are the variables declared in the parameter list of the function definition. They are only accessible to the statements located between the DEF FN and the END DEF, and any changes to them are only effective within this range; the value or variable passed-in is not affected. It is important to note that the parameters are only accessible to statements PHYSICALLY located in the function definition, NOT logically. This means that, if a program uses variable X and a function with parameter X, the program will access the variable outside the function, and the parameter inside the function. For example:

```
 10 X=5
 20 DEF FNA(X)
 30 GOSUB 100
 40 FNA=X
 50 X=10
 60 END DEF
 70 DISP FNA(6)
 80 STOP
100 X=X+2
110 RETURN
```

This program will display '6', and the final value of X will be 7. This is because the value of the parameter X was not incremented in the subroutine (the variable X was, so it now equals 7); parameter X still equals 6 (passed in) when it is assigned to the function value, the parameter X is then set to 10, but this has no effect on the variable X (which still equals 7).

How EXACT Time Is Calculated

When the first EXACT is done on the HP-75C after a system reset, the time of that EXACT is noted, the error and adjust accumulators are cleared, and the EXACT flag is set. All changes to the clock (by SET or ADJST) after this are broken into two parts: the time zone adjustments, and the errors. Time zones are to the nearest half hour, the error is what is left after the time zones are removed.

For example: +25 minutes is +1 time zone and −5 minutes error (5 minutes too fast); −25 minutes would be −1 time zone, and +5 minutes error (5 minutes slow). These adjustments and errors are accumulated until the next EXACT. The time of the new EXACT is adjusted to remove all the accumulated time-zone adjustments (this time is saved for the next EXACT sequence). The previous EXACT time is subtracted from this time to obtain the accurate elapsed time between the EXACTs. This timebase is used with the accumulated errors to determine the rate of clock adjustment needed to cancel the error. The sign of the accumulated errors determines whether the clock is slow or fast (+ means slow, − means fast). Clock adjustments are done in 0.25-second jumps, so an error of 1 second slow per day will result in four 0.25-second increments of the clock per day, or 1 every 8 hours. This is done automatically, regardless of what else the HP-75C is doing at the time.

Accessing Text Files From Programs

Text files can be accessed both from the keyboard or from a program by the use of the ASSIGN # command. This is handy for using or updating distribution lists, form letters, or address lists.

Under user or program control, you can create new text files, write lines of text to a text file, sort or rearrange lines in a text file, delete lines in a text file, update lines in a text file, and even search for a specific word in the text file. Text files created with an ASSIGN # command are treated the same as any other text file in the HP-75C. You can list them, edit them, rename them, merge them, and even transform them into BASIC. Of course, the lines of text would need to look like BASIC statements for the file to transform without errors.

The text file may be set up as a simple collection of lines of text, or it may be set up as a series of text records with a different item on each line.. The first way is the easiest way to store letters, memos, lists, and most things that involve sentences or phrases. These text files are usually accessed serially, meaning that you access the first line first, then move to the next line, and continue until you reach the end of the file.

Setting up a text file as a series of records is useful if you want to store the same several pieces of information for different items. Using an address list as an example, you may want to store last names, first names, titles, employers, addresses, and telephone numbers for several people. This file of address records might be set up as follows:

Line Number	Information
1	Last name
2	First name
3	Title
4	Employer
5	Address line 1 - information
6	Address line 2 - for person
7	Address line 3 - #1
8	Address line 4
9	Work phone
10	Home phone
11	Last name
12	First name
13	Title
14	Employer
15	Address line 1 - information
16	Address line 2 - for person
17	Address line 3 - #2
18	Address line 4
19	Work phone
20	Home phone
21	Last name
•	
•	
•	

The information may be put in the file originally either by doing an ASSIGN # TO 'FILE', TEXT and then PRINT#ing the lines of text to the file, or by simply EDITting the text file and inserting the lines that way. It is not necessary to create the file ahead of time if you use ASSIGN#, since the ASSIGN# will create the file for you if it doesn't already exist. One word of caution: only strings may be PRINT#ed to text files, and only one string is allowed per line.

The information may now be accessed by using PRINT#s, READ#s, and RESTORE#s. For example, let's say you want to see if you have an entry for John Jones in your address file. Assuming that the file 'ADDRFILE' is set up in records

as shown earlier, you could search for John Jones like this:

```
 10 Assign #1 to 'Addrfile'
 20 On error goto 120
 30 Input 'Last name?';a$
 40 For I=1 to 9999 step 10
 50 Read #1, I;b$
 60 If uprc$(a$)=uprc$(b$) then goto 80
 70 Next I
 80 For J=1 to 10
 90 Read #1,j;c$ @ Disp c$
100 Next J
110 Wait 1 @ goto 30
120 Disp 'Sorry, '&a$&' not found.'
130 Goto 30
```

To print out the address file, you could either write a program to print out the names and addresses in whatever format you wanted, or you could simply PLIST the address file. All of the line numbers in a text file are stripped off when the file is PLISTed.

HP-75C I/O (Input/Output)

The HP-75C is the first HP computer that has, built into it, the connectors and accompanying hardware for the Hewlett-Packard Interface Loop (HP-IL). So, the HP-75C is capable of input and output communication (I/O) with HP-IL devices without the purchase of an additional module or interface card. The implications of this step are many. The major implication is that understanding HP-75C I/O is a requirement for making full use of the machine. It is on this premise that this article is based.

After realizing that HP-IL I/O capabilities are built into the HP-75C, there are two more discoveries to be made. First, I/O may be turned on and off, and second, devices on the HP-IL loop must be assigned device codes with which they can be addressed by the HP-75C. These two features do allow more flexibility in directing information flow.

The standard HP-75C contains 13 functions for controlling HP-IL. These functions are listed below where we have adopted this notation: dc = device-code (any two-letter code that you choose); fn = file name; fs = file specifier, in the form "fn:dc" where fn is the name of the file to be found on the device specified by dc.

ASSIGN IO: allows you to assign names (two letter device-codes) to devices on the HP-IL loop.

DISPLAY IS ":dc": declares a display device by its previously assigned device-code.

PRINTER IS ":dc": declares the printer device by its device-code.

CLEAR LOOP: resets devices on the loop.

OFF IO: turns off loop communications.

RESTORE IO: restores loop communications.

INITIALIZE ":dc": initializes the mass storage medium.

CAT ":dc": displays the directory of files on the mass storage medium.

COPY "fn" TO "fs": copies a file from memory to mass storage.

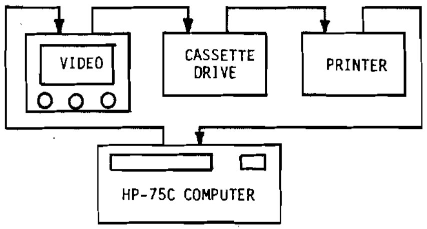
COPY "fs" TO "fn": copies a file from mass storage to memory.

RENAME "fs" TO "fs": renames a mass storage file.

PURGE "fs": purges the specified mass storage file.

PACK ":dc": packs a mass storage media.

Before we can execute any of these commands, it is necessary to ASSIGN IO. An example loop is diagrammed below, and the following command sequence would be used to assign "pr" as the device-code for the printer, "cd" as the device-code for the cassette drive, and "tv" as the device code for the video.



After plugging both cables into your HP-75C, the smaller of the two free ends indicates the direction of data flow and device location #1. After assigning I/O, it is necessary to tell the HP-75C which device on the loop is the display and which device is the printer. In the example, we would type-in: PRINTER IS ':PR' [RTN] DISPLAY IS ':TV' [RTN].

The terms 'tv' and 'cd' are two-letter codes that you make-up. Once a device in the loop is assigned a code, then all references to that device are made by using a colon, followed by the code within either single or double quotation marks.

Once a device is specified as a display, it will receive all information that appears in the HP-75C display. Similarly, a device specified as a printer will receive all information bound for a printer. An example would be to specify a video display device, such as the HP 82163A Video Interface, as the DISPLAY IS device. Now, the display will be duplicated on the screen. The edit keys will even move the cursor on the screen. Ad-

```
KEY IN              SEE DISPLAYED              ACTION TAKEN

assignio            3 DEVICE(s) ON LOOP
                    DEVICE #1 = '  '
tv                  DEVICE #1 = 'tv'             Assigns device 1 as the video.

                    DEVICE #2 = '  '
cd                  DEVICE #2 = 'cd'             Assigns device 2 as the mass
                                                 storage.
                    DEVICE #3 = '  '
pr                  DEVICE #3 = 'pr'             Assigns device 3 as the printer.

printer is':pr'     printer is':pr'              defines the selected printer.
display is':tv'     display is':tv'              defines the selected display.
```

Back Issue and Subscription Information

BACK ISSUES

Back issues of KEY NOTES are available back to V3N3, which introduced the HP-41. All **back issues are available ONLY from Corvallis, Oregon.** An index of these will be furnished on request. Available issues are:

V3N3 August 1979 (12 pages)
V3N4 November 1979 (12 pages)
V4N1 March 1980 (12 pages)
V4N2 Jun-Aug 1980 (12 pages)
V4N3 Sep-Dec 1980 (12 pages)
V5N1 Jan-Apr 1981 (16 pages)
V5N2 May-Aug 1981 (16 pages)
V5N3 Sep-Dec 1981 (16 pages)
V6N1 Jan-Feb 1982 (16 pages)
V6N2 Mar-May 1982 (16 pages)
V6N3 Jun-Aug 1982 (16 pages)

Prices for KEY NOTES back issues are as follows. All prices include first-class or air mail. Payment must accompany your order and must be a check or money order in U.S. dollars drawn on a U.S. bank. Or you may use your American Express, VISA, or MasterCard account; be sure to include your account number and card expiration data. Your order will be promptly mailed in an envelope.

NO. OF ISSUES	U.S., MEXICO CANADA	ALL OTHER COUNTRIES
1	$ 2.00	$ 3.50
2	$ 3.50	$ 5.00
3	$ 5.00	$ 6.50
4	$ 6.00	$ 8.00
5	$ 7.00	$ 9.00
6	$ 8.00	$10.00
7	$ 9.00	$11.00
8	$10.00	$12.00
9	$11.00	$13.00
10	$12.00	$14.00
11	$13.00	$15.00

SUBSCRIPTIONS

KEY NOTES is published quarterly in February, May, August, and November. A one-year subscription in the U.S. and Canada is $5* a year. It is free (worldwide) if you are a member of the Corvallis Users' Library ($20* U.S. and Canada; $35* elsewhere). Send your payment and complete name and address to the Corvallis address on back cover.

To get a subscription to KEY NOTES in Europe, contact the UPLE (Geneva address on back cover). To get KEY NOTES elsewhere, contact your nearest HP Sales Office or send your name, address, and calculator serial number to the Corvallis Users' Library, and we will forward your request to your nearest HP Sales Office.

**U.S. dollars. Payment must accompany order.*

ditionally, if we were to specify a printer as the DISPLAY IS device, it will mimic the display.

There are two methods you can use to avoid the manual assignment of devices on the loop. The first method assumes that the loop configuration is known. The second method can be used with various loop configurations.

The first method is to take advantage of the HP-75C key-definition feature. Simply assign, to a key, the command sequence required to ASSIGN IO. If we were to define the [*] key to accomplish the above task, we would use the command: def key "*", "assignio':tv,:cd,:pr'@ printeris':pr'@ displayis':tv" [RTN]. Then, we could make the loop assignments at the touch of the [*] key.

The second method requires the User's Library *I/O Utilities Solutions Book* and the magnetic cards that go along with it. Plus, it requires a short BASIC routine. The following routine is an example of how a simple loop could be automatically assigned without regard to the order or number of devices used. It handles the HP 82161A Digital Cassette Drive, the HP 82162A Thermal Printer, the HP 82163A/B Video Interface, and the HP 82905B Option 248 Printer.

We gave the name "AUTOIO" to the file containing this program and assigned it to our [CTL]/[ATTN] key. We used the command: defkey chr$(192), "run 'autoio'" [RTN] to define this key. Now, whenever we set up an HP/IL loop consisting of any of the above peripherals (one of each), in any order, it's a simple press of the [CTL]/[ATTN] key and we're in business.

```
10 ASSIGN IO ':zz'              !wake up the loop
20 SENDIO '','aau,aad1',''      !auto address the loop
30 Y$=''                        !null the Y$ string
40 FOR I=1 TO 30                !set up to address each device
50 IF I=1 THEN Z$='' ELSE Z$=','      !place a comma between each device code
60 B$=ENTIO$('','tad'&STR$(I)&',sai)!ask for accessory I.D.
70 IF B$='' THEN GOTO 140             !is there a device here
80 IF NUM(B$)=48 THEN X$=':tv'  !is this a displav device?
90 IF NUM(B$)=16 THEN X$=':cd'  !is this a mass memory?
100 IF NUM(B$)=32 THEN X$=':p1' !is this a strip printer?
110 IF NUM(B$)=33 THEN X$=':p2' !is this an 80 column printer?
120 Y$=Y$&Z$&X$                 !build the device code string in order
130 NEXT I                      !get the next device
140 ASSIGN IO Y$                !assign the loop according to Y$
150 A=POS(Y$,':p1') @ B=POS(Y$,'p2') @ C=POS(Y$,':tv') !find the printers
                                and video positions in the loop
160 IF A>0 THEN PRINTER IS ':p1' @ PWIDTH 24 !make stip pr system printer
170 IF B>0 THEN PRINTER IS ':p2' @ PWIDTH 80 !make 80 col pr system pr
180 IF C>0 THEN DISPLAY IS ':tv'             !make video system display
190 !if both printers are present then the 80 col printer is system printr
```

HEWLETT-PACKARD

HP-12C

REAL ESTATE

APPLICATIONS HANDBOOK

HP Targets the Real Estate Industry

Continuing its commitment to provide portable solutions for the real estate industry, Hewlett-Packard recently announced the new *HP-12C Real Estate Applications Handbook*. This handbook is the first industry-specific software written for the HP-12C Advanced Financial Programmable Calculator.

Written by two industry practitioners to exacting HP specifications, the *HP-12C Real Estate Applications Handbook* provides the solutions needed by not only real estate professionals but also home buyers. Brokers, appraisers, investors, analysts, and others who daily have to make wise real estate decisions will benefit from the wide range of real estate solutions.

Over 100 pages of easy-to-read and easy-to-use pre-written programs, keystrokes procedures, and detailed explanations provide quick and accurate solutions for real estate decision-making. The range of solutions available in this handbook is nothing short of amazing. From alternative financing to appraising, and from complex investing to cash-flow analyzing, the HP-12C and this handbook make it easy for the adept professional or the first-time calculator user to become an expert at solving complex real estate problems, especially those that are prevalent in today's economic situation.

But this new handbook only *complements* the powerful built-in features of the remarkable HP-12C. However, with *both* at your command, you will have a truly awesome set of tools to conquer any and all real estate industry challenges.

The handbook number is 00012-90015, and it is available now at your local authorized HP Dealer. It is not only a very useful and special Christmas present for an HP-12C owner, it is undoubtedly a tax-deductible asset for everyone in the real estate industry . . . maybe even for prospective home buyers.

HP KEY NOTES
Editor
Henry C. Horn
Technical Editor
Ted Wadman

Published quarterly (see page 19) for owners of Hewlett-Packard handheld computers. 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 U.S.A.

Hewlett-Packard SA
Users' Program Library Europe
7, Rue du Bois-du-Lan
P.O. Box, CH-1217 Meyrin 2
Geneva-Switzerland

Hewlett-Packard Company
Corvallis Division
1000 N.E. Circle Boulevard
Corvallis, Oregon 97330 U.S.A.

Address Correction Requested

FIRST-CLASS MAIL
U.S. POSTAGE
PAID
PERMIT NO. 814
PORTLAND, OR

FIRST CLASS

Printed in U.S.A.