PPC® JOURNAL

SPECIAL ISSUE E 1985

A reprint issue of material from the Journals of PPC.

PPC
PO Box 90579
Long Beach, CA 90809-0579
USA

INSIDE

To obtain your own copy of this special issue, send either a Self-Addressed, Stamped (with 3 ounces of postage) envelope (preferred), or a $1 bill, to:

See new address above.
Telephone (213) 498-2554

Bulk copies of this special issue are available to chapters, universities, and other interested groups. Contact PPC for details.

PPC, the Personal Programming Center, is a California non-profit benefit corporation dedicated to advancing the applications art of personal computing. PPC is the oldest personal computing users group and publishes the PPC Journal. Address all correspondence to PPC POB 9599 Fountain Valley, California 92728-9599 USA, Telephone (714) 754-6226.

PPC - A UNIQUE USERS GROUP

The Personal Programming Center, PPC, is a California non-profit public benefit corporation, dedicated to the advancement of truly personal computing. It is the world's oldest personal computing group, formed in June of 1974 with the advent of the HP-65. To PPC 'members', a personal computer is one that can be carried with him at all times, by choice. This does not include the so-called 'transportable' computers, but we are always open to reviews and articles on machines that PPC 'members' find of interest.

PPC 'members' have been, and continue to be, instrumental in the advancement of state-of-the-art machines. The majority of PPC articles are in support of Hewlett-Packard portable/personal computers and computational devices. We support the 10 series (slimline) programmable calculators, the 41 series (C/CV/CX), the 70 series (75C/D, 71B), the 80 series (85/86/87), and occasional inputs on the HP-110 (The Portable). PPC can help you to get every ounce of performance out of all of these machines.

What is PPC?

- PPC is an educational information gathering and disseminating organization that encourages active participation by its 'members'.

- PPC serves as a focal point for personal computer users to share their knowledge, expertise, and experiences. This is done in the spirit that the only fair compensation for a priceless idea is another priceless idea. We respect and encourage commercial interests.

- PPC has a unique editorial policy. Member's inputs are the main source of information, and as such, we encourage 'members' to submit their articles and programs in as near to a publication ready format as possible (see 'Submissions', below). When necessary, programs or articles may be retyped for reasons of proper reproduction, but the content is as submitted by the 'member'. We do reserve the right to correct spelling and grammatical errors, and generally remove statements such as 'enclosed is my check for renewal' in letters for the Feedback section. However, if requested, letters will be reproduced as submitted (depending on suitability and available space).

- PPC is not in business as a hardware or software vendor, but we occasionally make available special items that are not available through other sources. An example of this is the ERAMCO MLDL box. Due to customs problems and the need to commit to a quantity of boxes for purchase, PPC stepped in to allow this item (manufactured in the Netherlands) to be available to 'members' in the U.S. Another example is the famous PPC ROM project, which could not have been accomplished as a strictly commercial endeavor. The volunteer effort that was put in to making this ROM is what makes this module truly unique in history.

- PPC expends resources to maintain active bi-directional communication with manufacturers of the various products it supports.

- One of PPC's goals is to serve the average, non-programmer personal computer user. To this end, we publish tutorials and beginner's programs as appropriate.

- PPC contributes to state-of-the-art activities and cultivates activities that increase the quality and decrease the cost of hardware, firmware, and software, with the end goal of improvements to meet the user's needs.

- PPC is able to work directly with many manufacturers and dealers to provide you with discounts on items purchased for your portable systems. Announcements of these special buys are made in the Journal and on the PPC Phone Bulletin.

- PPC, being at the cutting edge of technology, is sponsoring the introduction of Amateur Radio classes and related information for the purpose of transmitting computer data over the airwaves (known as 'packet radio'). This is akin to modem transmission, without the attendant telephone charges.

- PPC encourages local chapters to provide an information network for rapid collection and dissemination of the latest discoveries, announcements, and 'member' and community accomplishments.

If you are the type of a person who desires to explore the capabilities of your personal computer, and find out new ways to pack more punch in your programs, then PPC is for you! People from all backgrounds, from all over the world, have found the many ways that PPC can assist them in their quest for knowledge. PPC strives to keep a balance, so that beginners can have their questions answered, intermediates can continue up the learning curve, and experts can share their knowledge at the cutting edge of technology.

<u>Correspondence</u> - When corresponding with PPC, or with other 'members', you should always use your 'member' number. Individual written responses are provided on a time available basis. To help facilitate a response, enclose a SASE or postcard with your inquiry. The PPC Workcenter phone number is (714) 754-6226 for direct conversation regarding current issues relating to PPC. Direct all mail to:

PPC
P.O. Box 9599
Fountain Valley, CA 92728-9599 USA

<u>Membership</u> - The rates for a 'membership' in PPC comprise two types. First, there is a one time 'new-member' processing fee. This fee is charged to cover initial entry costs, including items such as the Member Handbook. The second fee is the 'yearly' fee paid by the 'member' that entitles him or her to 12 issues of the PPC Journal. Since the PPC Journal is published based on material received by the 'members', there may be times when a month goes by without enough material for publication (or sometimes when there is enough to publish every two weeks!) Although we try to keep to a 'once per month' publication schedule, the 'yearly' rate is your fee to get 12 issues of the PPC Journal. All new 'memberships' and renewals received within two weeks of mailing of a Journal will be sent that current Journal, and 'membership' will start from that date. Those received after that time will start with the next Journal published. The current rates can be found in the attached application form.

<u>Activities</u> - Any activity that helps 'members' to better understand their personal computer is suitable for PPC. Local chapter meetings, community meetings for classes, and regional conferences are all common PPC activities.

<u>Submissions</u> - Submittals by 'members' of programs and/or articles for publication are actively encouraged. To assist us in making the best quality reproduction of your material, please follow these guidelines:

> All Journal submissions should be typed 5 1/2" wide, using a dark ribbon. For those of you using a dot-matrix printer, <u>do not</u> use compressed mode, and please <u>do</u> use bold mode. It <u>is also</u> extremely helpful if you can <u>submit</u> programs (and articles, if possible) on magnetic media (returned upon request). This will enable us to provide more legible listings, barcode, etc., as well as making it possible to download to the Remote Bulletin Board System.

<u>Product Reviews</u> - PPC would like the opportunity to review your hardware or software product. The Editor and/or reviewer reserve the right to present information on the product from the user's viewpoint. Contact the Editor at PPC for information on submitting products for review. All items returned upon request.

<u>PPC Workcenter</u> - The area used to house PPC quarters and where the Journals are produced, published, stuffed, etc., is called the PPC Workcenter. Meeting are generally held every Friday evening, starting about 6 P.M., and ending in the wee hours of the morning. Since meetings may occasionally not be held on holiday weekends, it is best to contact the PPC Workcenter first before visiting.

<u>PPC Hotline</u> - This telephone (714) 549-7674 provides the latest information relating to personal computing in a one to three minute message. Product announcements, bug reports, PPC news, and For Sale/Wanted ads are available for 'members' world wide. Members can leave a message at the end of the recording, if they so desire.

<u>PPC Chapters</u> - Local groups of users may form a PPC Chapter. The primary function of a Chapter is to form a communications network to aid in the information transfer process. A PPC Chapter list is enclosed with this issue.

<u>Member Handbook</u> - The PPC Member Handbook is a source of information on and for the 'members' of PPC. This reference will allow you to locate others in your area who share the same interests in Personal Computing.

<u>HP-67/97 Library</u> - PPC is the official custodian of the Hewlett-Packard User's Library. All inquiries should be addressed to: 67/97 Library, c/o PPC, P.O. Box 9599, Fountain Valley, CA, 92728-9599, USA.

PPC Publications

<u>PPC Journal</u> - The PPC Journal is the main publication of PPC. Previously, we published the PPC Computer Journal (abbreviated CO), and the PPC Calculator Journal (abbreviated CA). These two Journals were merged in 1985 because of the increasing overlap of the two machines caused by the advent of HP-IL and associated peripherals. The total number of pages per year was formerly 320 for CA, and 192 for CO. The total for the current PPC Journal is targeted at 576, or 64 pages more per year than the combined total of the previous Journals.

Sample Issue - This special sample issue representing a cross-section of the PPC Journal articles may be requested by any interested person by sending a Self-Addressed, Stamped Envelope (9" x 12") with 3 ounces of postage (U.S. First Class) attached, or by sending a $1 bill (please, no checks). The SASE is preferred, and no correspondence is necessary. For chapters, universities, and other large groups, contact PPC for quantity shipments.

NOMAS Listings - PPC has selected listings of the HP-41 system, associated ROMs (HP-IL, Timer, Extended Functions, etc.), and HP-75 system listings available on a NOt MAnufacturer Supported basis. These listings have been made available to the user community with the understanding that those receiving the listings not contact the manufacturer regarding them. Contact PPC for current prices and availability.

Remote Bulletin Board System - PPC is in the process of implementing a Remote Bulletin Board System to allow users to share software, send in articles and programs, and keep up to date on other PPC activities. This system is based on a heavily modified Xerox 820-II CP/M computer with a hard disk, and can normally be accessed during non-work hours (before 8 A.M., after 6 P.M., Pacific time).

Back Issues - Journals are continuously available in printed and/or microfiche form. A back issue table is included in material sent to new 'members', and is updated periodically.

PPC Journal 'Quotes' - Material may be quoted and described by authors, editors, or publications under the following conditions:

a) Reference source as PPC Journal V_N_P_.
b) Provide PPC and author a copy of material as reproduced.

Editor Exchange - Related computing publications may request to be placed on the Editors Exchange mailing list on a reciprocating basis.

WHY A SPECIAL ISSUE?

The intent of this issue is to introduce users of personal computers to the unique advantages that PPC can offer them. The articles in this issue have been culled from the previous years issues of the PPC Calculator Journal, the PPC Computer Journal, and the newly combined PPC Journal. You'll find that this issue contains:

Page 1 - The cover of the Special Issue. The cover of each PPC Journal is always page 1.

Pages 2-3 - An introduction to both PPC and to the Special Issue itself. What you're reading right now!

Page 4 - For all of you darkroom photography buffs, here's an example of using your HP-41 to make some of the tedious calculations easier. Olli Pera (7644) has a quick program that will allow you to adjust yourenlargement timing parameters for making blow-ups from your favorite negatives. From PPC Calculator Journal V11N5P5.

Page 4 - Gary Friedman (6522) has written a nice program for the HP-41 using the HP-IL Development module (or the Advantage ROM) to turn the HP-41 into an HP-16C (at least the most used functions). To aid in entering programs into your HP-41, most published programs are printed with barcode, as well. From PPC Calculator Journal V11N1P25.

Page 6 - Ever wonder what your phone number would be in alpha characters? Want to be able to have your friends reach you by calling "WILDGUY"? This program accepts your (7 digit) telephone number as input, and will print out or display all of the possible combinations of letters that your phone number will produce. Written by Carter Buck (4783), from PPC Calculator Journal V11N2P13.

Page 6 - SKWID is a pseudonym given to an otherwise anonymous 'member' to protect his (not so) innocence. Here, SKWID gives a brief introduction to assembly language programming with the HP-41 (also known as MCODE programming). The SKWID articles that appear periodically in the PPC Journal cover subjects ranging from MCODE programming, to HP-IL programming, to tutorials on the operation of the HP-41. This one is from PPC Calculator Journal V11N5P6.

Page 8 - This is a brief description of the simplest method of getting started in Synthetic Programming (without any other modules needed). Using this technique, you can immediately assign the "Byte Grabber" on your HP-41C/CV/CX, and get started in the world of Synthetic Programming. Also included are several examples of the use of the Byte Grabber for creating Synthetic instructions.

Page 9 - The HP-41 Hex Table shows the entire HP-41 instruction set in both decimal and hexadecimal formats. It also shows how each byte is represented both in the display and by the Thermal Printer/Plotter.

Page 10 - The Foreword of the PPC ROM manual is reproduced to give potential 'members' a feel for what is truly the greatest creation by a group of volunteer programmers in the history of personal computing. The PPC ROM is an 8K custom manufactured ROM containing an extremely wide variety of programs to enhance the use of your HP-41.

Page 11 - A complete list of all of the routines contained in the PPC ROM gives the user a feel for the immensity and power this ROM makes available.

Page 12 - The HP-41 Translator Pac for the HP-71 is introduced and discussed by its creator, Dr. William Wickes, of Hewlett-Packard's Portable Computer Division. This module allows the HP-71 to become a super-charged version of the HP-41, giving HP-71 users access to the huge established base of software already created for the HP-41. From PPC Journal V12N1P21.

Page 15 - Personal Applications Manager (PAM) has been written for the HP-75 computer by Brian Walsh (6951). This utility program is a 'shell' for the normal operating system that enables you to do your 'CATALL', 'PLIST', and mass storage manipulations easier than ever before. From PPC Computer Journal V3N4P31.

Page 15 - Another nice Biorhythm program for the HP-41 is demonstrated by Meindert Kuipers (7612) and Eric van der Wateren (8146). This program, which makes use of the functions in the Timer and Extended Functions modules (or an HP-41CX), is one of the shortest, quickest biorhythm programs ever. From PPC Calculator Journal V11N3P10.

Page 16 - Examples of the PPC Journal covers, showing some of the many programs and applications that are available to PPC 'members' in the PPC Journal.

Articles that have been in PPC Journals in recent months include:

- A complete description of the HP-75 I/O ROM by Raan Young (author of the ROM), including several undocumented commands included in the ROM. (CO V3N4P7)
- A complete telephone answering system, including Touch-tone decoding, speech synthesis, and phone logging, all built around an HP-41 with an HP-IL interface. Gary Friedman has done it again! (CA V11N6P23 & V11N7P6)
- 1001 (binary) methods of avoiding program decompilation, described by Roger Hill (4940). This article, originally presented at the Philadelphia PPC Conference, lets you keep programs that have been compiled from decompiling after being read in from mass storage devices, preventing longer waits while XEQ's and GTO's recompile. (CA V11N7P20)
- Butcher's Block, a (semi) regular column by David E. White (5353) (The Butcher). This feature delves into some of the hardware aspects of the HP systems, and shows how you can make modifications and improvements to your own devices. Also included are product reviews of new hardware. (various issues)
- Greatest divisor for both integers and non-integers, by Bob Hall (1859). Bob has a great interest in the power of the HP-15C, and is continually coming up with new programs for 10-series machines. (V12N1P15)

and, of course, much, much more, in every issue. Regular features include:

- HP Status - a list of product delivery times.
- Feedback - letters from the user community, with questions, tips, etc.
- Bits & Pieces - short notes to help you get the most out of your machines.
- Trading Post - a no-cost way for PPC 'members' to advertise their products to PPC Journal readers. Limited to twice-per-year insertion. The Editor reserves the right to present material from the users viewpoint.
- NOP - an errors column.
- Chapter Notes - information provided by PPC Chapters.

So there you have it! Take a look through this issue, and after you're through, you'll probably be another convert to 'the PPC way'. We look forward to having you join us, and hope that you get as much out of PPC as you can. We also look forward to your contributions to further the spread of information - so go to it!

ENLARGEMENT TIMING

The main purpose of this program is to calculate new exposure times for black & white prints after the magnification ratio is changed. The program prompts for the size, aperture, and exposure time of your old print, and the desired size and aperture for the new print, as well as how many F-stops darker or lighter you want the new print to be. Then it calculates the correct exposure time. The size of the print can be given in several ways:

1: If it is positive, it is a reading from the scale on the vertical rod on your enlarger.

2: If it is negative, it is the width of the image of the negative mask on your paper.

3: If it is zero, then the program assumes that the old print is your normal contact sheet and sets the values accordingly.

```
01*LBL "ENL"     21 /             41 FC? 22
02 FIX 1         22*LBL 01        42 0
03*LBL 00        23 "NEW H"       43 Y↑X
04 CLST          24 PROMPT        44 *
05 "OLD H"       25 X>0?          45 STOP
06 PROMPT        26 XEQ 11        46 GTO 00
07 X=0?          27 X<0?          47*LBL 10
08 GTO 12        28 XEQ 10        48 -2.3
09 X>0?          29 1             49 /
10 XEQ 11        30 +             50 RTN
11 X<0?          31 "NEW F"       51*LBL 11
12 XEQ 10        32 PROMPT        52 .1914
13 1             33 *             53 *
14 +             34 X↑2           54 1.362
15 "OLD F"       35 /             55 +
16 PROMPT        36 1/X           56 RTN
17 *             37 "CORR"        57*LBL 12
18 X↑2           38 2             58 268.5
19 "OLD T"       39 CF 22         59 GTO 01
20 PROMPT        40 PROMPT        60 END
```

Program lines from line 47 (LBL 10) ahead depend on the enlarger. The negative constant on line 48 is the width of your negative mask (I have got a 23*36 mm mask => -2.3 cm). If your input to the "H"-prompt is negative then it gets divided by this constant and the result is magnification ratio. The subroutine (LBL 11) calculates the magnification from the reading from the rod. I used the curve fit program to find out this equation. The result was straight line m = .1914*h+1.362 (R = 0.9998) in my enlarger (AXOMAT 4). The constant in line 58 can be found in X-register at "NEW H"-prompt when you run the program and give the normal values you use for contact sheet at the "OLD"-prompts (I use H=36 cm, F=5.6, T=10s, these give the 268.5 when "NEW H"-prompt is cleared).

The equation used is: $T2 = 2^{C}*T1*\left(\frac{(M2+1)*F2}{(M1+1)*F1}\right)^2$

T = time, M = magnification ratio, F = aperture, C = correction in F-stops.

It is based on the equation $t_c = t*(m+1)^2$ found in:

A. Hawkins, D. Avon: Photography, the guide to technique.

Examples: I want to make a 12 cm * 18 cm picture (so I adjust the enlarger head and read the value "21" from the rod). F = 8 and I want to make it a half F-stop darker than in the contact sheet:

Display:	I do:	
0.00	XEQ "ENL"	
OLD H	(R/S)	;contact sheet
NEW H	21(R/S)	;from rod scale
NEW F	8(R/S)	
CORR	2(1/X)(R/S)	;or .5(R/S)
13.7		;answer

I make the print and it is perfect (this is an example, not real life). Let's make a 30 cm * 40 cm picture from it (must be projected on to the floor):

Display:	I do:	
13.7	(R/S)	
OLD H	21(R/S)	
OLD F	8(R/S)	
OLD T	13.7(R/S)	
NEW H	-30(R/S)	;width of picture
NEW F	8(R/S)	
CORR	(R/S)	
66.3		;answer

REGISTERS: 19

ROW 1 (1-5)

ROW 2 (5-11)

ROW 3 (12-17)

ROW 4 (18-23)

ROW 5 (23-29)

ROW 6 (30-37)

ROW 7 (37-44)

ROW 8 (45-52)

ROW 9 (52-58)

ROW 10 (58-60)

Olli Pera (7644)
Hallituskatu 31 A 17
SF-90100 OULU 10
FINLAND

[R/S]

HP-16C EMULATOR

Those of us who program using assembly language had to invest a small amount in an HP-16C or (GaAsP!) a T.I. LCD Programmer just to help calculate absolute branching addresses, work with signed representation, or evaluate carry and overflow flag status.

This program, using the capabilities of the HP-IL Development ROM and XF/M, emulates the basic functions of the HP-16C. It offers variable word length of 0-32 bits, signed and unsigned notation, carry and overflow flags (indicated by flag annunciators 0 and 4), and quick conversion between hexadecimal, decimal, octal, and binary (up to 10 bits). This, combined with the Development ROM's ROT, AND, OR, XOR, and NOT functions let me use my 41 for just about everything I bought the 16C for.

For the uninitiated, the IL Development ROM offers the essential bit manipulation and Hex/Oct/Bin I/O capability that make this program possible. In addition, it offers low level IL commands, a scope mode so one can monitor the messages circling the loop, and a buffer that can transfer anything anywhere without fearof normalization. In short, this well designed ROM offers something for every PPC member to play with.

NOTE: Despite this article's title, it is not to imply that this program will replace the 16C. It merely emulates the 16C's behavior in simple addition and subtraction, no doubt the 16C's most frequently used functions.

INSTRUCTIONS

1) XEQ "16C". Display comes up in 8 bit, unsigned Hex display mode. 7 keys now have new global assignments: ENTER↑, +, and - are replaced by similar labels; RDN, SIN, COS, and TAN implement decimal, hex, octal, and binary respectively.

2) Key in problems as you normally would using RPN. For example, to add 5E and 20 (both hex), press 5E, ENTER↑, 20, +. The result, 7E, is displayed. To see the decimal equivalent hit "DEC" (RDN) and get 126. Octal is obtained by pressing "OCT" (COS), and binary can be seen (if the number is 10 digits or less) by pressing "BIN" (TAN).

3) To implement 2's complement notation, XEQ c (shift SQRT). Flag 2 annunciator is set indicating 2's complement. XEQ c again will toggle back to unsigned representation. Example:

DO	SEE	FLAGS	COMMENTS
XEQ "DEC"	D 126		
165 ENTER↑	D 165		
32 +	D 197		
XEQ c	D -91	2	Automatically checks display sign.
224 +	D -123	0,2	Flag0 set, indicating carry.

4) To change word size, XEQ d (shift LOG). Program will prompt for "WORD SIZE?" (friendly, huh?). Enter word size (up to 31 bits) in decimal and R/S. EXAMPLE: (continued from above)

DO	SEE	FLAGS	COMMENTS
	D -123	0,2	
XEQ d	WORD SIZE?		
16 R/S	D 133	2	MSB is now 8 bits to the left and is = 0, resulting in a positive interpretation.

5) To exit and restore "normal" format, XEQ e. The newly defined keys have now been cleared.

A few words about flag behavior are in order. The 16C treats both the carry and overflow flags a little differently for each function. The carry flag (flag 0) for addition indicates that the sum is 1 digit longer than the given word size. In subtraction it indicates that a borrow was needed to obtain the answer, and generally the status of the carry flag for subtraction is the inverse of the equivalent function for addition. FOR THIS REASON, 5 ENTER↑ 2 - WILL NOT PRODUCE THE SAME CARRY FLAG STATUS AS 5 ENTER↑ 2 CHS + !!! (See example #1 later on.)

The overflow or V flags (flag 4) function is a bit more involved (no pun intended). Generally, a carry indicates the result can't be represented in current word size, which is the definition of the overflow. In unsigned mode, therefore, the V flag is tied to the carry flag.

In 2's complement mode, the V flag will be set if [the two added numbers are of the same sign] and [the sign bit of the result is opposite that of the original numbers] and [the carry flag and the new sign bit are opposite]. The flag rules for multiplication and division are another story altogether.

In multiplication, the C flag isn't even used, and if overflow occurs the MSB in the displayed result is replaced with the sign bit of the complete answer. The multiplication and division functions, however, were not included in this program because I felt the extra coding needed to implement these seldom used functions would have made the program uncomfortably large. (I could be wrong. If I get enough requests I'll gladly expand the program to include these.)

EXAMPLES

1) $91_{10} - 32_{10} = ??$

```
     91           01011011
  +(-32)         +11100000
  ------         ---------
     59     C-->100111011
```

As mentioned earlier, there are two ways to do this problem. One way is to add 91 to the 2's compliment of 32, as illustrated on top. This is equivalent to keying in 91 ENTER↑ 32 CHS +, and results in a carry.

```
     91           01011011
  -(+32)         -00100000
  ------         ---------
     59           00111011
```

When subtracting, however, the carry flag means that no borrow from the 9th bit was necessary. As the second binary example shows, when you subtract these two numbers no 9th digit borrow was needed, (and no carry was produced either for that matter), and therefore no carry flag was set.

In both examples the numerical answers were the same. It is up to the user to know how to properly interpret the flags.

2) You're assembling 6502 machine code by hand (this is because your disk assembler has a fatal bug and the manufacturer refuses to update it with revision 1C). Here, if the index register (X) is not zero, we must branch backwards to a routine named RN(1), which is located at address 0208_{16}.

```
   .
   .
   .
0229 DEX              CA       Decrement index reg.
022A BNE    RN(1)     D0  ??   Branch if not equal
022C RTS              60       Return
```

What argument do we put at line 022B in order to branch backwards to line 208? Since this is a relative branch the argument must be added to the current program counter (which is 022C, since the PC is always 1 step ahead) and the result is the next address to be executed. The problem then is:

$$\begin{array}{r} 208_{16} \\ -22C_{16} \\ \hline DC \end{array}$$

DO	SEE	FLAGS
XEQ d	WORD SIZE?	
16 R/S	D 0	
XEQ "HEX"	0 H	
208 ENTER↑	208 H	
22C -	FFDC H	0

Since the answer's most significant bit is set, the 6502 treats this as a negative number and branches backwards, as we planned. Notice the correct answer is obtained whether we're in unsigned or 2's complement mode.

This program has been of tremendous help to me in my day to day DEC<>HEX<>DEC conversions and assembly work. Not only does it alleviate the need to carry 2 calculators, but it also provides an alternative to those of us who don't like ENTER↑ keys that mush down.

HP-16C EMULATOR by Gary Friedman (6522)

```
01*LBL "16C"
02 FIX 0
03 CF 29
04 7
05 STO 01
06 255
07 STO 02
08 "HEX"
09 23
10 PASN
11 "OCT"
12 24
13 PASN
14 "BIN"
15 25
16 PASN
17 "DEC"
18 22
19 PASN
20 "-"
21 51
22 PASN
23 "+"
24 61
25 PASN
26 "ENT↑"
27 41
28 PASN
29 CLX
30 STO 03
31*LBL "HEX"
32 1
33 STO 04
34 RDN
35*LBL 01
36 HEXVIEW
37 HEXIN
38*LBL "OCT"
39 2
40 STO 04
41 RDN
42*LBL 02
43 OCTVIEW
44 OCTIN
45*LBL "BIN"
46 3
47 STO 04
48 RDN
49*LBL 03
50 BINVIEW
51 BININ
52*LBL "DEC"
53 4
54 STO 04
55 RDN
56*LBL 04
57 " D      "
58 ARCL 03
59 PROMPT
60*LBL "ENT↑"
61 CF 00
62 CF 04
63 CF 05
64 CF 06
65 STO 03
66 X<0?
67 XEQ 05
68 RCL 02
69 AND
70 ENTER↑
71 GTO 00
72*LBL "+"
73 X<0?
74 XEQ 05
75 RCL 02
76 AND
77 RCL 01
78 BIT?
79 GTO 06
80 RDN
81 CF 05
82 FC?C 06
83 SF 05
84 GTO 07
85*LBL 06
86 RDN
87 CF 05
88 FS?C 06
89 SF 05
90*LBL 07
91 +
92 RCL 01
93 1
94 +
95 CF 00
96 BIT?
97 SF 00
98 RDN
99 XEQ 08
100 RDN
101 FS?C 07
102 XEQ 12
103 RCL 02
104 AND
105 STO 03
106*LBL 00
107 RCL 01
108 BIT?
109 GTO 13
110 RDN
111 GTO IND 04
112*LBL 13
113 RDN
114 SF 06
115 FC? 02
116 GTO IND 04
117 ENTER↑
118 NOT
119 1
120 +
121 RCL 02
122 AND
123 CHS
124 STO 03
125 RDN
126 GTO IND 04
127*LBL 08
128 CF 04
129 FC? 02
130 GTO 09
131 FC? 05
132 GTO 17
133 FC? 00
134 GTO 10
135 RCL 01
136 BIT?
137 RTN
138 GTO 11
139*LBL 10
140 RCL 01
141 BIT?
142 GTO 11
143 RTN
144*LBL 11
145 SF 04
146 RTN
147*LBL 09
148 FS? 00
149 SF 04
150 FC? 00
151 CF 04
152*LBL 17
153 0
154 RTN
155*LBL 12
156 FS? 00
157 GTO 14
158 SF 00
159 FC? 02
160 SF 04
161 RTN
162*LBL 14
163 CF 00
164 FC? 02
165 CF 04
166 RTN
167*LBL "-"
168 SF 07
169 CHS
170 GTO "+"
171*LBL 05
172 NOT
173 1
174 +
175 RTN
176*LBL c
177 FS? 02
178 GTO 15
179 SF 02
180 GTO 16
181*LBL 15
182 CF 02
183*LBL 16
184 0
185 GTO "+"
186*LBL d
187 "WORD SIZE?"
188 PROMPT
189 STO 01
190 2
191 X<>Y
192 Y↑X
193 1
194 -
195 STO 02
196 RDN
197 DSE 01
198 0
199 GTO "+"
200*LBL e
201 CLA
202 24
203 PASN
204 23
205 PASN
206 22
207 PASN
208 25
209 PASN
210 41
211 PASN
212 51
213 PASN
214 61
215 PASN
216 CLX
217 X<>F
218 FIX 4
219 SF 29
220 END

LBL "16C"
LBL "HEX"
LBL "OCT"
LBL "BIN"
LBL "DEC"
LBL "ENT↑"
LBL "+"
LBL "-"
END          428 BYTES
```

REGISTERS: 62

ROW 1 (1-5)

ROW 2 (6-11)

ROW 3 (11-15)

ROW 4 (16-21)

ROW 5 (21-26)

ROW 6 (26-31)

ROW 7 (31-38)
ROW 8 (38-45)
ROW 9 (45-51)
ROW 10 (52-57)
ROW 11 (57-60)
ROW 12 (60-66)
ROW 13 (67-72)
ROW 14 (72-79)
ROW 15 (80-87)
ROW 16 (88-96)
ROW 17 (97-103)
ROW 18 (104-112)
ROW 19 (113-121)
ROW 20 (122-130)
ROW 21 (130-136)
ROW 22 (137-145)
ROW 23 (146-153)
ROW 24 (154-161)
ROW 25 (162-167)
ROW 26 (168-176)
ROW 27 (176-181)
ROW 28 (182-187)
ROW 29 (187-192)
ROW 30 (193-201)
ROW 31 (202-208)
ROW 32 (208-214)
ROW 33 (215-220)

Happy bit manipulating!

Gary Friedman (6522)
5084 Gloria Ave.
Encino, CA 91436 USA

TELEPHONE NUMBER-LETTER COMBINATIONS

This program displays/prints all letter combinations of a seven digit phone number. For example, when we call "Time", we dial "POPCORN" instead of 767-2676. This program generates the entire list of combinations for your favorite numbers.

Input is any seven digit number in the form XXX.XXXX. For example, the number above would be entered as 767.2676 in X. XEQ "TEL" and the fun begins; the program halts after all combinations have been displayed.

Minimum SIZE is 014, and the program clears the 14 highest data registers. An Extended Functions module or HP-41CX is required. The ALPHA register is cleared, as are Flags 00 through 07, and Flag 25 should be clear on entry. The routine requires 37 program registers.

Carter Buck (4783)
P.O. Box 11203
Oakland, CA 94611-0203 USA

REGISTERS: 38
ROW 1 (1-5)
ROW 2 (5-13)
ROW 3 (13-22)
ROW 4 (23-30)
ROW 5 (31-38)
ROW 6 (38-44)
ROW 7 (44-50)
ROW 8 (50-55)
ROW 9 (56-60)
ROW 10 (61-68)
ROW 11 (68-75)
ROW 12 (76-83)
ROW 13 (84-92)
ROW 14 (92-100)
ROW 15 (101-109)
ROW 16 (110-117)
ROW 17 (118-126)
ROW 18 (127-134)
ROW 19 (135-142)
ROW 20 (142-149)

TELEPHONE NUMBER-LETTER COMBINATIONS by C. Buck (4783)

```
01*LBL "TEL"
02 SIZE?
03 14
04 X>Y?
05 PSIZE
06 .
07 X<>F
08 R↑
09 RCL X
10 SIZE?
11 8
12 -
13 E-3
14 ST* T
15 ISG X
16 *
17 R↑
18 FRC
19 SIGN
20 RDN
21 7
22 GTO 14
23*LBL 10
24 DSE Y
25*LBL 14
26 E1
27 ST* L
28 X<> L
29 INT
30 ST- L
31 GTO IND X
32*LBL 00
33 SF IND Y
34 "0"
35 GTO 14
36*LBL 01
37 SF IND Y
38 "1"
39 GTO 14
40*LBL 02
41 "ABC"
42 GTO 14
43*LBL 03
44 "DEF"
45 GTO 14
46*LBL 04
47 "GHI"
48 GTO 14
49*LBL 05
50 "JKL"
51 GTO 14
52*LBL 06
53 "MNO"
54 GTO 14
55*LBL 07
56 "PRS"
57 GTO 14
58*LBL 08
59 "TUV"
60 GTO 14
61*LBL 09
62 "WXY"
63*LBL 14
64 CLX
65 RDN
66 ASTO IND Y
67 DSE X
68 GTO 10
69 CLA
70 RDN
71 7.007
72 +
73 3
74*LBL 11
75 ISG T
76 AOFF
77 FC? IND T
78 STO IND Y
79 ISG Y
80 GTO 11
81 X<> L
82 -
83 R↑
84*LBL 12
85 DSE Y
86*LBL 13
87 ARCL IND Y
88 FS? IND X
89 GTO 14
90 2
91 CHS
92 AROT
93 RDN
94 ATOX
95 RDN
96 ATOX
97 RDN
98*LBL 14
99 DSE X
100 GTO 12
101 AVIEW
102 E
103*LBL 15
104 CHS
105 AROT
106 -
107 ATOX
108 SIGN
109 FS? IND Y
110 GTO 14
111 ASTO L
112 CLA
113 ARCL IND Z
114 AROT
115 RDN
116 ASTO IND Y
117 CLA
118 ARCL L
119 X<>Y
120 ABS
121 X<>Y
122 7
123 ST+ L
124 RDN
125 DSE IND L
126 GTO 13
127 3
128 STO IND L
129 SIGN
130*LBL 14
131 ISG Z
132 GTO 15
133 RDN
134 ST- Y
135 E3
136 /
137 +
138 .
139*LBL 16
140 STO IND Y
141 ISG Y
142 GTO 16
143 X<>F
144 R↑
145 SIGN
146 CLST
147 X<> L
148 CLD
149 END
```

R/S

MCODE FOR BEGINNERS

SKWID, THIS IS THE BOSS AGAIN

What do you want this time?

YOUR ARTICLE ON THE DEVELOPMENT MODULE WAS NOT TOO BAD. HOW WOULD YOU LIKE TO WRITE ANOTHER ARTICLE FOR US?

Great, we could become world famous and may even get published again.

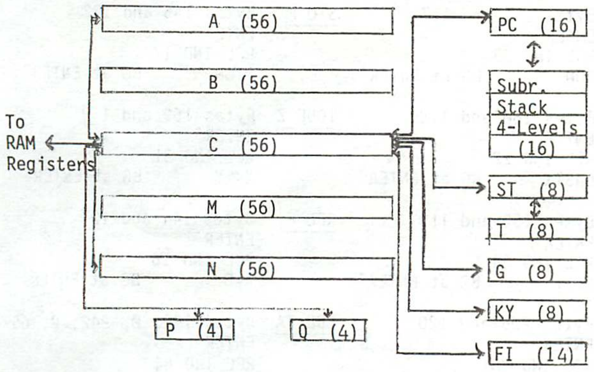
POP!

OH, SORRY ABOUT THAT, SKWID, BUT IT WAS GETTING A LITTLE TOO HARD TO SEE WITH YOUR HEAD SWELLED UP LIKE THAT.

NOW, YOUR MISSION, SHOULD YOU CHOOSE TO ACCEPT (does this sound familiar?) IS TO INSTILL UPON SOME MERE BEGINNERS THE RUDIMENTS OF MCODE PROGRAMMING. THIS MESSAGE WILL SELF DESTRUCT IMMEDIATELY.

Well, Skwid, the boss has done it again. He blew our 41's to bits, what do we do know? Well, guys, it looks like we're going to have to put in a little overtime on this one. Does anyone know how to construct a 41 from this mess? (who cares?)

In order to understand the structure of machine language programming (MCODE) on the 41 you must know the structure of the internal registers (this is much like a good synthetic programmer knows the RAM structure). A diagram of the basic structure is given below.



And now for a little program.

SKWID, THIS IS THE BOSS. HOW MANY TIMES MUST I TELL YOU TO DOCUMENT YOUR STUFF? ARE YOU TRYING TO LOSE EVERYONE? I, FOR ONE, AM ALREADY LOST.

Okay, here we go again.

Register	Usage
C	This is the main register. All communication with RAM is done through this register. It is the only register that can interact with all registers (except T).
A	This register may interact with the C and B registers. Arithmetic may be done between each of these registers.
B	Same as register A.
M and N	These registers are used for storage and may only interact with register C.
P and Q	These are the pointers. They point to digits in the A, B, and C registers. They may range from 0-13. Only one may be selected at any time.
Carry	This is only one bit. It may be set and tested. However, the next step after the carry is set will always clear the carry.
PC	This is the program counter. The subroutine stack is only 4 high (such is life). Returns may be pushed onto the stack and popped off of it.
G	This register interacts with the C register at the digit pointed to by the active pointer and the next highest digit. If the pointer = 13, then wraparound takes place.
ST	Flags 0-7. Interacts with digits 0 and 1 of register C. These flags may be cleared, tested, and set.
KEY	This is the keyboard flag. It becomes set whenever a key on the keyboard is set.
FI	Peripheral flag register.

Now it is time to show you the fields on a 56 bit register. These are used extensively to operate on only part of the C, B, or A registers.

```
Nybble: 13 12 11 10  9  8  7  6  5  4  3  2  1  0

Field:  <----------------------- ALL ----------------------->
Field:  <MS><----------------- M ----------------><XS>
Field:                       <---- ADR ----><-- S&X -->
Field:                                 <-KY-->
```

Okay, how about a program?

THERE YOU GO AGAIN, SKWID, JUMPING OVER ALL OF THE EXPLANATIONS TO GET TO THE PROGRAM.

Aw, come on, just this once.

DO YOU WANT TO BE PUBLISHED?

You win again.

I ALWAYS DO.

Field	Explanation
S&X	Exponent and exponent sign.
XS	Exponent sign only.
ALL	All 14 digits.
M	Mantissa.
MS	Mantissa sign.
ADR	This is where the address from the return stack is placed when it is popped from the return stack or where it is taken from when the address is pushed onto the stack.
KY	This is where the C and KY registers exchange contents.
P-Q	All digits P through Q (P<=Q). All digits P through 13 (P>Q).
@R	At digit pointed to by active pointer.
R<	All digits 0 through the digit pointed to by R.

At the end of this article (we hope) is a partial listing of the instruction set. The purpose of most of the instructions is self explanatory since they have RPN counterparts or the meaning can be obtained by just looking at the instruction. However there are a few oddities (wouldn't you know it, HP does it to us again). They will be explained below.

Word	Task
RSHFA	Shift register A right one nybble as specified by the postfix. The leftmost nybble is copied into the 2 leftmost digits.
RSHFB	Same as above but for register B.
RSHFC	Same as above but for register C.
LSHFA	Same as the right shift but does a left shift.
RCR	Rotate register C right by the number of digits specified by the postfix.
R=	Set the active pointer equal to the number specified by the postfix.
?R=	Set carry bit if pointer equals the number specified by the postfix.
LD@R	Load the number (0-F) at the digit the pointer is at. Decreases pointer by one.
?FSET	Set the carry flag if flag specified is set.

Well, boss, is it okay to write a short program now?

SURE SKWID, EVERYTHING LOOKS GOOD UP TO THIS POINT.

We know that.

We shall write a Y<>Z routine. Keith Jarett, one of our illustrious members (we can't remember why, it seems our memories have been synthetically cleared), suggested this routine. The way to fill out the function address table is given in PPC Calculator Journal V9N3P30 and the ERAMCO user's manual on page 23. Also, at the end of this article (again we hope) is a table containing the hex code for the names of rom functions.

Now here's the routine (it's even annotated. Surprise, surprise).

Hexcode	Mnemonic	Purpose
09A	"Z"	The functions name is in reverse order. The
03E	">"	last letter of the name has hex 080 added to
03C	"<"	it.
019	"Y"	
0B8	READ 2(Y)	Put Y register into C.
10E	A=C ALL	Save Y in A.
078	READ 1(Z)	Get Z register.
0A8	WRIT 2(Y)	Write Z out to Y.
0AE	A<>C ALL	Put Y back into C so it may be written to Z.
068	WRIT 1(Z)	Write Y to Z.
3E0	RTN	Return.

How's that, boss?

BEAUTIFUL! I COULDN'T HAVE DONE BETTER MYSELF.

We know that. Now for another routine. This one shall be a MCODE version of the go to end routine in the PPC ROM. This routine puts you at the first line in the program that has the permanent END as its end.

The object of the program is to place the location of the .END. into the last 2 bytes of the b register. As we all know (even if

Continued on page 12

KEYING HP-41 SYNTHETIC INSTRUCTIONS

Shortly after the introduction of the HP-41, avid PPC members discovered a method of breaking two-byte functions in half, taking the first byte of one function, giving it the second byte of another function, and creating a third function with different characteristics than the first two! This programming technique, originally dubbed "Synthetic Programming", was expanded to allow virtually full control of the HP-41 system registers. The use of synthetic instructions in the HP-41 has worked for all varieties of HP-41 that have been introduced thus far, but you should bear in mind that Hewlett-Packard does not support or endorse the use of Synthetic Programming on your calculator. If you should happen to call HP and ask about "Synthetic Programming", they'll either disavow any knowledge of it or refer you right back to PPC -- so <u>don't call Hewlett-Packard with questions about Synthetic Programming!</u>

Synthetic Programming is purely a software technique to access new functions in your HP-41, and as such it won't cause any damage to you HP-41 hardware. As with any new skill, though, there's a price that must be paid - and the price for learning Synthetic Programming is paid with plenty of "MEMORY LOST"'s.

Just to get you acquainted with some of the many synthetic instructions, a few of them are listed below:

Display	Printer	
Ø1 DSE M	01 DSE [	The 6 status registers M, N, O, P,
Ø2 ISG N	02 ISG \	Q, and F display differently than
Ø3 VIEW O	03 VIEW]	they appear on the thermal
Ø4 X<>P	04 X<> ↑	printer. Compare the display
Ø5 RCL Q	05 RCL _	characters with their printed
Ø6 STO ⊢	06 STO ᵀ	counterparts on the bytetable.
Ø7 TONE Z	07 TONE Z	One of 128 synthetic tones.
Ø8 RCL F	08 RCL F	Direct RCL of data register 1Ø7.
Ø9 LBL ᵀA	09♦LBL "A"	A global (<u>not</u> local) LBL "A".
1Ø ᵀ"PPC"	10 ""PPC""	Quotes in display text lines.
11 ᵀ⁊ⵅⵅ⵰	11 "ΓαΒ×"	Many special characters available.
12 E3	12 E3	Short form EEX saves one byte.
13 ᵀ	13 ""	PPC NOP, hex FØ, null text line.

Over the years, many techniques have been developed by PPC members world-wide to create these functions in their HP-41. Early techniques used "byte jumping", or using HP-67/97 cards, or modifying other HP-41 cards. With the development of the PPC ROM, Synthetic Programming became more readily available to thousands. And today, with such programmer's aids as the ZENROM and CCD-Module, you can key Synthetic instructions <u>directly</u> into your HP-41 from the keyboard. For those of you who would like to start out 'fresh' with Synthetic Programming, use the following technique (PPC bug 9) to enter the 'byte grabber' function into your HP-41.

1. Master Clear (back-arrow, ON) to obtain "MEMORY LOST".
2. ASN "+" to the LN key.
3. ASN "DEL" to the LOG key.
4. Switch to USER mode.
5. Switch to PRGM mode.
6. Enter LBL ᵀT into memory (any Alpha label will do).
7. Do CAT 1, and R/S <u>immediately</u> with LBL ᵀT in the display.
8. Delete 1 line (DEL ØØ1) by pressing LOG, then $\Sigma+$.
9. Wait a moment, then (for C or CV only), press BST.
10. Do GTO .ØØ5, and you'll see LBL Ø3 in the display.
11. Delete 3 lines (DEL ØØ3) by pressing LOG, then $\sqrt{x}$.
12. Go into Alpha mode, and enter "?AAAAAA"
13. C and CV owners will see "?A ______"
14. Switch out of PRGM mode, and do GTO.. (PACKs calculator).

You should now end up with that most indispensable tool of Synthetic Programming, the Byte Grabber (known as BG to his friends). To verify that you've done this correctly, press and hold down the LN key (until NULL appears), and confirm that it shows XROM 28, 63 (you're still in USER mode, right?). You can save this on a status card if you wish, or you may want to practice this little technique 'til you can do it at will. If you don't get the XROM 28, 63, try again from the beginning.

The table on the opposite page of this brief HP-41 byte instruction description is the HP-41 "HEX TABLE". Along the edges, you'll find the hexadecimal value of all of the various HP-41 instructions, both single- and multi-byte types. The decimal value for each instruction is in the lower left corner of the box. The top line of each box indicates the function (for single byte instructions) or the prefix (for multi-byte instructions). The next line shows the postfix instruction for two-byte functions on the left, with the display representation of that byte on the right. Finally, the symbol in the lower right corner of the box is the Thermal Printer/Plotter representation of each byte.

You can use the Byte Grabber (BG) function to 'snatch' the first byte off of a multi-byte function. Every time you press the BG key, the HP-41 opens another register of program memory (out of the unused registers left). Because the first byte of the BG is a Text 7 character, the HP-41 thinks that it needs 8 bytes (1 byte for the Text byte, plus 7 characters), and with only 7 bytes available in the new register, it 'grabs' the next byte of program memory. By first creating a two-byte function whose second byte is a useful 'prefix', and following it with a useful 'postfix', we can create virtually any combination of multi-byte functions.

To demonstrate, we'll show you how to make the example functions shown above. To make things easier, first ASN the PACK function to the LOG key, and the BST function to the TAN key.

WARNING: DO NOT BYTE-GRAB AT THE PROGRAM STEP IMMEDIATELY PRECEDING AN END!!! This will cause the CAT 1 chain to loose part of its linkage, and can result in a calculator 'lock-up' condition. For instructional purposes, we'll use the RCL byte as the first byte (of the three total bytes needed), which will be 'grabbed' by the Byte Grabber. To make sure that we have enough room in program memory for byte grabbing, and to prevent any accidents, we'll precede each sequence with ENTER^, then BST to the ENTER^. This will ensure that we are in position to do the byte grabbing.

<u>DSE M</u>	Bytes 151 and 117 ENTER^ RCL IND 23 RDN BG at ENTER^	<u>STO ⊢</u>	Bytes 145 and 122 ENTER^ RCL IND 17 SIGN BG at ENTER^
<u>ISG N</u>	Bytes 15Ø and 118 ENTER^ RCL IND 22 LASTX BG at ENTER^	<u>TONE Z</u>	Bytes 159 and 113 ENTER^ RCL IND 31 X<>Y BG at ENTER^
<u>VIEW O</u>	Bytes 152 and 119 ENTER^ RCL IND 24 CLX BG at ENTER^	<u>RCL F</u>	Bytes 144 and 1Ø7 ENTER^ RCL IND 16 R-D BG at ENTER^
<u>X<>P</u>	Bytes 2Ø6 and 12Ø ENTER^ RCL IND 78 X=Y? BG at ENTER^	<u>LBL ᵀA</u>	Bytes 192, Ø, 242, Ø, 65 ENTER^ RCL IND 64 + "ZA" BG at ENTER^ PACK to link CATalog 1
<u>RCL Q</u>	Bytes 144 and 121 ENTER^ RCL IND 16 X≠Y? BG at ENTER^		

<u>ᵀ"PPC"</u> Bytes 245, 34, 8Ø, 8Ø, 67, 34
Ø1 ENTER^
Ø2 ᵀXPPCX (X's used for ")

BG at ENTER^, and the ASCII characters in the text line will now be individual instructions (the byte grabber grabbed the text prefix byte). Lines Ø3 and Ø7 will be E^X-1. Delete these lines and replace them with RCL Ø2. BST to the ENTER^, then BG. Delete the resulting text line and the STO 15. You'll see the modified text line with the 'special' characters replacing the X's.

<u>ᵀ⁊ⵅⵅ⵰</u> Bytes 244, 6, 4, 5, 1
Ø1 ENTER^
Ø2 ᵀABCD (any four Alpha characters)

BG at ENTER^, and delete the four following instructions (which used to be ASCII characters!). Insert LBL Ø5, LBL Ø3, LBL Ø4, and LBL ØØ. BST to the ENTER^, then BG. Delete the resulting text line and the STO 15. You'll see the modified text line with 'hangman' characters replacing the old Alpha characters.

<u>E3</u> Bytes 27, 19
ENTER^
1 EEX 3 PACK, then BG at ENTER^

Note: the HP-41 uses a NULL byte (ØØ hex) to separate consecutive numeric entries. PACKing will remove the extra NULL bytes if the preceding instruction is not a number. The PACKing ensures that you byte-grab the '1' instruction, not the NULL.

<u>ᵀ</u> Byte 24Ø
ENTER^
RCL IND T BG at ENTER^

This has been just a brief description of a very few of the many possibilities that can open up to you with Synthetic Programming. For further information, you should refer to any of several books on Synthetic Programming, and back issues of the PPC Journal. Of particular interest are "SYNTHETIC PROGRAMMING ON THE HP-41C", by William Wickes, and "SYNTHETIC PROGAMMING MADE EASY" by Keith Jarett. Also, check pages 4 & 5 of this Special Issue for a description of the PPC ROM. This ROM, and its accompanying manual, will provide you with more information on Synthetic Programming than you can imagine. These (and many other HP-41 books) can be ordered through EduCalc Mail Store, 27953 Cabot Road, Laguna Niguel, CA 92677, (714) 831-2637.

R/S

HP-41C COMBINED HEX/DECIMAL BYTE TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	NULL 00 ‾ 0 ◆	LBL 00 01 ⊼ 1 ×	LBL 01 02 ▩ 2 x̄	LBL 02 03 ▩ 3 ←	LBL 03 04 ⊼ 4 α	LBL 04 05 ⊼ 5 β	LBL 05 06 Γ 6 Γ	LBL 06 07 ▩ 7 ↓	LBL 07 08 ▩ 8 Δ	LBL 08 09 ▩ 9 σ	LBL 09 10 ▩ 10 ◆	LBL 10 11 ▩ 11 λ	LBL 11 12 μ 12 μ	LBL 12 13 ∡ 13 ∡	LBL 13 14 ▩ 14 τ	LBL 14 15 ▩ 15 Φ	0
1	0 16 ▩ 16 θ	1 17 ▩ 17 Ω	2 18 ▩ 18 δ	3 19 ▩ 19 Ȧ	4 20 ▩ 20 ȧ	5 21 ▩ 21 Ä	6 22 ▩ 22 ä	7 23 ▩ 23 Ö	8 24 ▩ 24 ö	9 25 ▩ 25 Ü	. 26 ▩ 26 ü	EEX 27 ▩ 27 Æ	NEG 28 ▩ 28 œ	GTO ᵀ 29 ≠ 29 ≠	XEQ ᵀ 30 ▩ 30 £	W ᵀ 31 ▩ 31 ▒	1
2	RCL 00 32 32	RCL 01 33 ! 33 !	RCL 02 34 " 34 "	RCL 03 35 # 35 #	RCL 04 36 $ 36 $	RCL 05 37 % 37 %	RCL 06 38 & 38 &	RCL 07 39 ' 39 '	RCL 08 40 (40 (	RCL 09 41) 41)	RCL 10 42 * 42 *	RCL 11 43 + 43 +	RCL 12 44 , 44 ,	RCL 13 45 - 45 −	RCL 14 46 . 46 .	RCL 15 47 / 47 /	2
3	STO 00 48 0 48 0	STO 01 49 1 49 1	STO 02 50 2 50 2	STO 03 51 3 51 3	STO 04 52 4 52 4	STO 05 53 5 53 5	STO 06 54 6 54 6	STO 07 55 7 55 7	STO 08 56 8 56 8	STO 09 57 9 57 9	STO 10 58 : 58 :	STO 11 59 ; 59 ;	STO 12 60 < 60 <	STO 13 61 = 61 =	STO 14 62 > 62 >	STO 15 63 ? 63 ?	3
4	+ 64 @ 64 @	− 65 A 65 A	* 66 B 66 B	/ 67 C 67 C	X<Y? 68 D 68 D	X>Y? 69 E 69 E	X≤Y? 70 F 70 F	Σ+ 71 G 71 G	Σ− 72 H 72 H	HMS+ 73 I 73 I	HMS− 74 J 74 J	MOD 75 K 75 K	% 76 L 76 L	%CH 77 M 77 M	P→R 78 N 78 N	R→P 79 O 79 O	4
5	LN 80 P 80 P	X↑2 81 Q 81 Q	SQRT 82 R 82 R	Y↑X 83 S 83 S	CHS 84 T 84 T	E↑X 85 U 85 U	LOG 86 V 86 V	10↑X 87 W 87 W	E↑X-1 88 X 88 X	SIN 89 Y 89 Y	COS 90 Z 90 Z	TAN 91 [91 [	ASIN 92 \ 92 \	ACOS 93] 93]	ATAN 94 ↑ 94 ↑	→DEC 95 _ 95 _	5
6	1/X 96 ᵀ 96 ᵀ	ABS 97 a 97 a	FACT 98 b 98 b	X≠0? 99 c 99 c	X>0? 100 d 100 d	LN1+X 101 e 101 e	X<0? A ▩ 102 f	X=0? B ▩ 103 g	INT C ▩ 104 h	FRC D ▩ 105 i	D→R E ▩ 106 j	R→D F ▩ 107 k	→HMS G ▩ 108 l	→HR H ▩ 109 m	RND I ▩ 110 n	→OCT J ▩ 111 o	6
7	CLΣ T ▩ 112 p	X<>Y Z ▩ 113 q	PI Y ▩ 114 r	CLST X ▩ 115 s	R↑ L ▩ 116 t	RDN M[▩ 117 u	LASTX N\ ▩ 118 v	CLX O] ▩ 119 w	X=Y? P↑ ▩ 120 x	X≠Y? Q_ ▩ 121 y	SIGN ⊢ᵀ ▩ 122 z	X≤0? a ▩ 123 π	MEAN b ▩ 124 ∣	SDEV c ▩ 125 →	AVIEW d Σ 126 Σ	CLD e ⊢ 127 ⊢	7
	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111	

HP-41C COMBINED HEX/DECIMAL BYTE TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
8	DEG IND 00 128 ◆	RAD IND 01 129 ×	GRAD IND 02 130 x̄	ENTER↑ IND 03 131 ←	STOP IND 04 132 α	RTN IND 05 133 β	BEEP IND 06 134 Γ	CLA IND 07 135 ↓	ASHF IND 08 136 Δ	PSE IND 09 137 σ	CLRG IND 10 138 ◆	AOFF IND 11 139 λ	AON IND 12 140 μ	OFF IND 13 141 ∡	PROMPT IND 14 142 τ	ADV IND 15 143 Φ	8
9	RCL IND 16 144 θ	STO IND 17 145 Ω	ST+ IND 18 146 δ	ST− IND 19 147 Ȧ	ST* IND 20 148 ȧ	ST/ IND 21 149 Ä	ISG IND 22 150 ä	DSE IND 23 151 Ö	VIEW IND 24 152 ö	ΣREG IND 25 153 Ü	ASTO IND 26 154 ü	ARCL IND 27 155 Æ	FIX IND 28 156 œ	SCI IND 29 157 ≠	ENG IND 30 158 £	TONE IND 31 159 ▒	9
A	XR 0-3 IND 32 160	XR 4-7 IND 33 161 !	XR8-11 IND 34 162 "	X12-15 IND 35 163 #	X16-19 IND 36 164 $	X20-23 IND 37 165 %	X24-27 IND 38 166 &	X28-31 IND 39 167 '	SF IND 40 168 (	CF IND 41 169)	FS?C IND 42 170 *	FC?C IND 43 171 +	FS? IND 44 172 ,	FC? IND 45 173 −	GTO XEQ IND IND 46 174 .	SPARE IND 47 175 /	A
B	SPARE IND 48 176 0	GTO 00 IND 49 177 1	GTO 01 IND 50 178 2	GTO 02 IND 51 179 3	GTO 03 IND 52 180 4	GTO 04 IND 53 181 5	GTO 05 IND 54 182 6	GTO 06 IND 55 183 7	GTO 07 IND 56 184 8	GTO 08 IND 57 185 9	GTO 09 IND 58 186 :	GTO 10 IND 59 187 ;	GTO 11 IND 60 188 <	GTO 12 IND 61 189 =	GTO 13 IND 62 190 >	GTO 14 IND 63 191 ?	B
C	GLOBAL IND 64 192 @	GLOBAL IND 65 193 A	GLOBAL IND 66 194 B	GLOBAL IND 67 195 C	GLOBAL IND 68 196 D	GLOBAL IND 69 197 E	GLOBAL IND 70 198 F	GLOBAL IND 71 199 G	GLOBAL IND 72 200 H	GLOBAL IND 73 201 I	GLOBAL IND 74 202 J	GLOBAL IND 75 203 K	GLOBAL IND 76 204 L	GLOBAL IND 77 205 M	X<>-- IND 78 206 N	LBL -- IND 79 207 O	C
D	GTO -- IND 80 208 P	GTO -- IND 81 209 Q	GTO -- IND 82 210 R	GTO -- IND 83 211 S	GTO -- IND 84 212 T	GTO -- IND 85 213 U	GTO -- IND 86 214 V	GTO -- IND 87 215 W	GTO -- IND 88 216 X	GTO -- IND 89 217 Y	GTO -- IND 90 218 Z	GTO -- IND 91 219 [	GTO -- IND 92 220 \	GTO -- IND 93 221]	GTO -- IND 94 222 ↑	GTO -- IND 95 223 _	D
E	XEQ -- IND 96 224 ᵀ	XEQ -- IND 97 225 a	XEQ -- IND 98 226 b	XEQ -- IND 99 227 c	XEQ -- IND100 228 d	XEQ -- IND101 229 e	XEQ -- IND102 230 f	XEQ -- IND103 231 g	XEQ -- IND104 232 h	XEQ -- IND105 233 i	XEQ -- IND106 234 j	XEQ -- IND107 235 k	XEQ -- IND108 236 l	XEQ -- IND109 237 m	XEQ -- IND110 238 n	XEQ -- IND111 239 o	E
F	TEXT 0 IND T 240 p	TEXT 1 IND Z 241 q	TEXT 2 IND Y 242 r	TEXT 3 IND X 243 s	TEXT 4 IND L 244 t	TEXT 5 IND M[245 u	TEXT 6 IND N\ 246 v	TEXT 7 IND O] 247 w	TEXT 8 IND P↑ 248 x	TEXT 9 IND Q_ 249 y	TEXT10 IND ⊢ᵀ 250 z	TEXT11 IND a 251 π	TEXT12 IND b 252 ∣	TEXT13 IND c 253 →	TEXT14 IND d 254 Σ	TEXT15 IND e 255 ⊢	F
	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111	

FOREWORD

Because of the nature of the PPC ROM PROJECT, this manual is somewhat unusual. This manual is the effort of over one hundred users who worked directly on it, and many hundreds of others who indirectly contributed to its completion. Before diving into the routines, the PPC ROM user should first read the introductory material in Part 1, which includes the Preface, Organization and Use of Manual, Functional Grouping of Routines, Abstracts, and brief Introduction to Synthetic Programming. Once you have read Part 1 you may explore at random with a minimum of difficulty. Refer to the Glossary in the Appendices for definitions of unfamiliar terms.

This project is unique in the history of software projects. IBM and other large corporations have assigned multi-tens of programmers to a software project, but never before have over 100 programmers worked so long and so hard on a project--without compensation of any kind. The PPC ROM PROJECT is a community project in the true sense of the word. The project has always been completely public with month by month reports openly published for all to study and respond to.

It took two years and two months to complete. The first year was spent in mastering the HP-41 system, and while we were "first in line" for HP's announced Custom ROM Program, we waited until we could utilize the full power of the HP-41 to produce as complete a programmer's ROM as possible.

We believe in true personal computing and that a so-called higher level language is not always the path to greater computing power. We want to manage our always-too-small memory in ways we think are best. We prefer a flexible operating system that allows us to control our programming environment, and we want a well thought out operating system that can be altered if we wish. The routines in the PPC ROM express these interests and concerns. Much of the work that went into the ROM is original and makes a contribution to the Art. Here are a few examples.

- Programmed and documented by hundreds of users
- Outstanding ratio of features per byte
- Unusually complete technical details
- Personal contact for additional help
- A routines ROM - not an applications program ROM. This is a programmer's ROM.
- The full power of Synthetic Programming is made available to all HP-41 users.
- Operating system extension and enhancement programs
- Fastest known numerical sort routine
- Block and matrix operations defined and programmed
- Extended capability and improved accuracy in financial calculations
- Commendable integrator program
- Greatly expanded multiplot and high resolution graphics programs
- Matrix format printing of flags set in View Flags
- Skipping zero data in Block View
- Better access to all of HP's ROMs with **XE** Routine
- Expanded memory using **IP** and **PS** for QUAD "page" switching

One of the main objectives of the PPC ROM USER'S MANUAL is to provide an expression of the type of detail that programmers desire. This includes more than just a collection of general purpose routines with as many technical details as possible. The users are an essential part of the loop, and the PPC ROM project is designed to include user inputs. A portion of the ROM fund is being held in reserve for a follow-up addendum that will include:

a. Corrections for the errors found
b. Description of any BUGs that may be found
c. Additional examples
d. Additional Applications Programs
e. Suggestions for ROM or Manual improvement
f. Review of project
g. Conclusions and recommendations for future "user community" software development projects

A word about bugs. BUGs are of concern to all users. We define a BUG to be a failure of a routine or program to operate according to the complete instructions. Unless precise inputs and conditions are specified, you may have questions regarding the complete instructions. If you think you have found a BUG, we want to know about it. But first you should realize that after hundreds of hours of testing we haven't found any major BUGs. Therefore, a considerable effort on your part should be expended before you think BUG and call the PPC Clubhouse. Many "bugs" may be explained away by gaining a better understanding of the complete instructions. We do want to hear from you so your inputs may be included in the addendum. Happy BUG hunting.

There were many ideas for routines in the ROM that for various reasons never became a reality. It is possible that these creative ideas may appear in a future PPC ROM. We would like to have seen more alpha-string capabilities and diagnostic routines. In the math group we would like to have seen some routines in the statistics area. After reading this manual and mastering the PPC ROM, you will no doubt think of several routines that you will feel should also have been included.

We had planned special microcode routines that would have simultaneously simplified and expanded memory management, but the SDS system that would allow microcode in the ROM would have caused a three month delay, so these routines did not materialize. One reason alpha-string and diagnostic routines did not materialize was lack of space, and these kinds of routines tend to be memory intensive. There was very little discussion of statistics routines, and no specific statistics routines were actually submitted.

PPC ROM ROUTINES IN HEX TABLE ORDER

MISC.

- +K - Additional Key Assignments 1,1,43
- -B - Store half of LB (routine) 1,20,203
- 1K - First Key Assignment 1,1,40
- 2D - Decode 2 Bytes to Dec. 2,9,94

A

- A? - Assign Reg. Finder (# of) 1,5,163
- AD - Alpha Delete last character-1,6.98
- AL - Alphabetize X & Y 2,4,120
- AM - Alpha to Memory 5,9,37
- Ab - Alpha store b. 2,10,181

B

- BA - Barcode Analyzer 5,1,1,
- BC - Block Clear 5,5,208
- BD - Base B to base Decimal 4,1,1,
- BE - Block Exchange 5,3,32
- BI - Block Increment 2,7,61
- BL - BLDSPEC inputs for LB 2,7,1
- BM - Block Move 5,4,103
- BR - Block Rotate 5,4,126
- BV - Block View 3,5,99
- BX - Block Extremes 5,4,155
- BΣ - Block Statistics 5,5,195

C

- C? - Curtain Finder 1,6,46
- CA - Complex Arithmetic 4,4,1
- CB - Count Bytes 2,9,33
- CJ - Calendar Date to Julian Date 4,3,119
- CD - Character to Decimal 2,2,178
- CK - Clear Key Assignments 1,3,40
- CM - Combinations 4,2,98
- CP - Column Print formatting 4,6,76
- CU - Curtain Up 2,2,131
- CV - Curve Fit 3,5,1,
- CX - Curtain to absolute decimal location in X 2,2,128

D

- DC - Decimal to Character 1,5,175
- DF - Decimal to Fraction 3,10,50
- DP - Decimal to Program Pointer 2,9,67
- DR - Delete Record 5,4,97
- DS - Display Set 2,1,42
- DT - Display Test 1,6,77

E

- E? - End Finder 3,1,195
- EP - Erase Program Memory 2,1,79
- EX - Exponent of X 2,1,14

F

- F? - Free Register finder 1,3,196
- FD - First Derivative 3,9,125
- FI - Financial Calculations 3,1,1,
- FL - Flag inputs for LB 2,7,21
- FR - Fractions 3,9,1

G

- GE - Go to End 2,10, 158
- GN - Gaussian RN Generator 4,1,127

H

- HA - High Res. Hist. W/Axis 4,6,12
- HD - Hide Data Registers 1,7,143
- HN - Hex to NNN 2,6,61
- HP - High Resolution Plot 4,7,4,
- HS - High Resolution Histogram 4,6,48

I

- IF - Invert Flag 2,8,1
- IG - Integrate 3,7,1
- IP - Initialize Page 2,7,70
- IR - Insert Record 5,4,90

J

- JC - Julian Date to Cal. Date 4,3,158

L

- L- - Load half of LB(routine) 1,8,11
- LB - LOAD Bytes 1,7,4
- LF - Locate Free Reg. Block 1,3,0
- LG - PPC Logo 4,5,1
- LR - Lengthen Return Stack 3,4,26

M

- M1 - Matrix, Interchange any two Rows 5,3,28
- M2 - Matrix, Multiply a row by non-zero K 5,2,1
- M3 - Matrix, Add multiple of one row to other 5,3,9
- M4 - Matrix, absolute address to (i,j) 5,3,56
- M5 - Matrix, (i,j) to absolute address 5,3,66
- MA - Memory to Alpha 5,9,44
- MK - Make Multiple Key Assignment 1,1,1
- ML - Memory Lost resize to 017 1,5,1
- MP - Multiple Variable Plot (1-9) 4,7,1
- MS - Memory to Stack 2,8,155
- MT - Mantissa of X 2,1,29

N

- NC - Nth Character 2,5,172
- NH - NNN to Hex 2,5,1
- NP - Next Prime 4,1,99
- NR - NNN Recall 5,9,15
- NS - NNN Store 5,9,1

O

- OM - Open Memory 2,10,142

P

- PA - Program Pointer Advance 2,10,152
- PD - Program Pointer to Decimal 2,9,52
- PK - Pack Key Assignment Registers 1,4,77
- PM - Permutations 4,2,81
- PR - Pack Register 5,5,230
- PO - Paper Out 5, 9, 27
- PS - Page Switch 2,7,76

Q

- QR - Quotent Remainder 2,9,82

R

- RD - Recall Display Mode 3,4,69
- RF - Reset Flag 1,5,17
- RK - Reactivate Key Assignment 3,5,84
- RN - Random Number Generator 4,1,146
- RT - Return Address to Decimal 2,9,40
- RX - Recall from absolute address in X register 2,10,129
- Rb - Recall b 5,9,34

S

- S1 - Stock Sort 5,5,1
- S2 - Small Array Sort (≤32) 5,7,144
- S3 - Large Array Sort (>32) 5,6,34
- S? - SIZE Finder 1,6,32
- SD - Store Display Mode 3,4,45
- SK - Suspend Key Assignment 3,4,53
- SM - Stack to Memory 5,8,1
- SR - Shorten Return Stack 3,4,1
- SU - Substitute Character 2,5,175
- SV - Solve Routine 3,8,92
- SX - Store Y in Absolute Address X 2,10,122
- Sb - Store b (Rom Ø Entry) 3,4,24
- SW - Selection without replacement-5,8,18

T

- T1 - TONE-Beep Alternative 2,8,140
- TB - Base Ten to base B 4,2,37
- TN - Tones: Tone N(0-127) 2,2,118

U

- UD - Uncover Data Register 1,4,71
- UR - Unpack Register 5,5,216

V

- VA - View Alpha 1,4,62
- VF - View Flags 5,8,43
- VK - View Key Assignments 2,3,1
- VM - View Mantissa 1,10,1
- VS - Verify Size 2,1,59

X

- XD - Hex to Decimal 1,10,240
- XE - XROM Entry 1,7,119
- XL - XROM inputs for LB 5,8,32

- Σ? - Sigma Resiger Finder 1,5,23
- ΣC - Curtain Finder 1,7,154

TOTALS

	Labels	Routines
Housekeeping	31	31
Math	17	48
Peripheral	7	7
Synthetic	67	67
	122	153

8,130 Bytes

you don't) the address of the register that the .END. resides in is in the last three nybbles of the c register. The .END. resides in the last three bytes of this register. So we must place a 3 in the fourth nybble from the right and the address of the .END. in the last three nybbles of the b register.

We will introduce you to the use of flags. There are 14 flags. Flags 0-9 have no special meaning and may be set and cleared as desired. However flags 10-13 are given special meaning. They are listed below.

Flag	If Set
10	Program pointer is in ROM.
11	Stack lift is enabled.
12	Program pointer is in a private program.
13	A User coded (RPN) program is running.

Now here is the routine

Hexcode	Mnemonic	Description
085	"E"	Name
007	"G"	
378	READ 13(c)	Get pointer to .END. register.
106	A=C S&X	Save .END. pointer in A.
04E	C=0 ALL	Zero C so there will be no pending returns.
0A6	A<>C S&X	Put .END. pointer in C.
01C	R= 3	Set pointer = 3 for loading with constant.
0D0	LD@R 3	Load 3 at digit pointed to by R.
0C4	CLRF 10	Clear flag so that calculator thinks it is in RAM.
2C8	SETF 13	Set flag so calculator will execute .END.
328	WRIT 12(b)	Write .END. address to register b.
3E0	RTN	Return.

		POSTFIX							
	Instruction	ALL	S&X	M	R<	@R	MS	XS	P-Q
	A=0	00E	006	01A	00A	002	01E	016	012
	B=0	02E	026	03A	02A	022	03E	036	032
	C=0	04E	046	05A	04A	042	05E	056	052
	A<>B	06E	066	07A	06A	062	07E	076	072
	B=A	08E	086	09A	08A	082	09E	096	092
	A<>C	0AE	0A6	0BA	0AA	0A2	0BE	0B6	0B2
	C=B	0CE	0C6	0DA	0CA	0C2	0DE	0D6	0D2
	C<>B	0EE	0E6	0FA	0EA	0E2	0FE	0F6	0F2
	A=C	10E	106	11A	10A	102	11E	116	112
	A=A+B	12E	126	13A	12A	122	13E	136	132
	A=A+C	14E	146	15A	14A	142	15E	156	152
	A=A+1	16E	166	17A	16A	162	17E	176	172
	A=A-B	18E	186	19A	18A	182	19E	196	192
P	A=A-1	1AE	1A6	1BA	1AA	1A2	1BE	1B6	1B2
R	A=A-C	1CE	1C6	1DA	1CA	1C2	1DE	1D6	1D2
E	C=C+C	1EE	1E6	1FA	1EA	1E2	1FE	1F6	1F2
F	C=C+A	20E	206	21A	20A	202	21E	216	212
I	C=C+1	22E	226	23A	22A	222	23E	236	232
X	C=A-C	24E	246	25A	24A	242	25E	256	252
	C=C-1	26E	266	27A	26A	262	27E	276	272
	C=0-C	28E	286	29A	28A	282	29E	296	292
	C=-C-1	2AE	2A6	2BA	2AA	2A2	2BE	2B6	2B2
	?B≠0	2CE	2C6	2DA	2CA	2C2	2DE	2D6	2D2
	?C≠0	2EE	2E6	2FA	2EA	2E2	2FE	2F6	2F2
	?A<C	30E	306	31A	30A	302	31E	316	312
	?A<B	32E	326	33A	32A	322	33E	336	332
	?A≠0	34E	346	35A	34A	342	35E	356	352
	?A≠C	36E	366	37A	36A	362	37E	376	372
	RSHFA	38E	386	39A	38A	382	39E	396	392
	RSHFB	3AE	3A6	3BA	3AA	3A2	3BE	3B6	3B2
	RSHFC	3CE	3C6	3DA	3CA	3C2	3DE	3D6	3D2
	LSHFA	3EE	3E6	3FA	3EA	3E2	3FE	3F6	3F2

REG.		READ	WRIT	RCR	SETF	CLRF	?FSET	R=	?R=	LD@R-	?FI	SELP
0	T	XXX	028	XXX	388	384	38C	39C	394	010	3AC	024
1	Z	078	068	33C	308	304	30C	31C	314	050	32C	064
2	Y	0B8	0A8	23C	208	204	20C	21C	214	090	22C	0A4
3	X	0F8	0E8	03C	008	004	00C	01C	014	0D0	02C	0E4
4	L	138	128	07C	048	044	04C	05C	054	110	06C	124
5	M	178	168	0BC	088	084	08C	09C	094	150	0AC	164
6	N	1B8	1A8	17C	148	144	14C	15C	154	190	16C	1A4
7	O	1F8	1E8	2BC	288	284	28C	29C	294	1D0	2AC	1E4
8	P	238	228	13C	108	104	10C	11C	114	210	12C	224
9	Q	278	268	27C	248	244	24C	25C	254	250	26C	264
10	⊦	2B8	2A8	0FC	0C8	0C4	0CC	0DC	0D4	290	0EC	2A4
11	a	2F8	2E8	1BC	188	184	18C	19C	194	2D0	1AC	2E4
12	b	338	328	37C	348	344	34C	35C	354	310	36C	324
13	c	378	368	2FC	2C8	2C4	2CC	2DC	2D4	350	2EC	364
14	d	3B8	3A8	XXX	XXX	XXX	XXX	XXX	XXX	390	XXX	3A4
15	e	3F8	3E8	XXX	XXX	XXX	XXX	XXX	XXX	3D0	XXX	3E4

LCD Character table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
01	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	_
02	sp.	!	"	#	%	$	&	'	<	>	*	+	←	-	→	/
03	0	1	2	3	4	5	6	7	8	9	[illegible]	,	∠	=	>	?
04	-	a	b	c	d	e	‾	⊤	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	≠	Σ	∡

sp. = blank space

Well, that's all for now. Hope we have helped to increase your understanding of MCODE. We also hope the boss likes this enough to print it.

SKWID
P.O. Box 3103
Tustin, CA 92681 USA

R/S

THE HP-41 TRANSLATOR PAC FOR THE HP-71

Bridging the Gap

Can you imagine a new HP-41 with 5 times the memory and four times the speed of the HP-41? Able to display the entire stack after each operation? With up to 10,000 direct-access data registers? With the ability to add new functions in RAM? With a command stack? With 12-digit mantissa and 3-digit exponent numerical accuracy, and implementing IEEE floating-point math exception handling? With the capability of executing subroutines written in BASIC or FORTH? At half the cost of the HP-41?

Well, forget the last one, that's impossible for now. But all the rest just describe the HP-71B with the HP-41 Translator Pac installed. The Translator Pac is a 48K-byte plug-in ROM that adds to the HP-71 and HP-41 calculator mode, which emulates the calculator operation of the HP-41, and can run HP-41 programs. The Pac is scheduled to go on the HP price list February 1st (available mid-February). Here's a summary of the whole package:

PRODUCT DESCRIPTION

48K-byte HP-71 plug-in ROM module, containing:

*HP-41 emulator system, with 147 built-in standard HP-41 functions, plus 23 additional HP-41 functions and operations unique to the emulator.

*HP-71 Text Editor program, for editing HP-41 programs and other HP-71 text files.

*TRANS41 program, for translating HP-41 programs into the emulator system.

*READ41 program, for automatic program transfer from the HP-41 to the HP-71.

*KEYS41 key file, containing HP-71 key assignments for HP-41 keyboard functions.

*14 BASIC keywords, for BASIC language access to the FORTH and HP-41 systems, and to HP-71 text files.

*HP-71 FORTH language system, containing 335 built-in FORTH words, enhancing the FORTH '83 standard with floating-point, string, HP-IL, and file-handling words.

Owner's Manual

HP-71 keyboard overlay for HP-41 key assignments

A NEW RECORD

This module sets a new record for the number of functions added to an HP calculator by a plug-in module--there are 486 functions in the ROM:

147 standard HP-41 functions
23 new HP-41 functions
11 strange HP-41 functions used only by the system
291 additional FORTH words
14 BASIC keywords

This count does not include over 100 "headerless" words that could be used by a really ambitious and clever programming freak who has managed to decompile and decipher the ROM code.

The HP-41 function list includes the complete HP-41C/CV programmable function set, plus additional alpha register, flag, and conditional functions from the HP-41CX (the timer, alarm, and extended memory functions are not available in the module's built-in function set; the single function TIME is included). In addition, all of the character printing functions from the HP 82143A Thermal Printer are included for use with HP-IL printers. HP-41 functions not included in the built-in function set can be added to the system through the use of the underlying FORTH system. The translator program TRANS41 is open-ended; any function added to the HP-41 vocabulary can be handled by the translator.

The FORTH language system in the Translator Pac is nearly identical to that in the FORTH/Assembler ROM (the Assembler is not present in the Translator Pac). The ROM dictionary has 46 HP-41 words (mostly floating-point words, like E^X-1 or OCT) added to the FORTH/Assembler ROM word set, plus a separate vocabulary containing the remaining HP-41 words. The latter are less conventional FORTH words in that they depend on certain specific data structures (like the HP-41 data or alpha registers) or use non-RPN notation (like STO 5 or FIX IND 2), and hence cannot be casually included in standard FORTH programming.

WHAT, NO STACK LIFT DISABLE????

The HP-41 Translator makes a bold break with tradition (here's how you make a possible drawback into a feature) by not implementing any stack lift disable. Yes, that's right the ENTER^ key becomes a vestigal organ not worth including in the built-in keys file. The underlying reason that stack-lift disable is not implemented is to eliminate the associated system overhead. But a more virtuous sounding reason is that stack-lift disable was a mistake on the HP-35, and remains a mistake today. The HP-41 translator elects to use a more flexible input style, derived from the FORTH outer loop and the BASIC operating system, than the actual HP-41 provides. Given this, plus the difficulty of mapping the HP-41 keyboard onto that of the HP-71, eliminating stack-lift disable is not a big deal. You're going to have to learn a different input style anyway.

Consider how a traditional HP RPN calculator handles number entry. Every key on the calculator is an immediate-execute key; when you press a number key, you begin building a number string in the X-register and the display. Each subsequent number key adds one character to the string. To terminate the number entry, you just press any non-numeric function key. The problem arises when you want to enter two consecutive numbers, with no operation in between. Hence, the ENTER^ key. But here the HP-35 designers overshot the mark--instead of just having ENTER^ terminate digit entry, they (who knows why?) made it also carry out the unrelated task of duplicating the number into the Y-register, then disabling stack lift. This has the extremely unpleasant, AOS-like effect of leaving the calculator in generally indeterminate state--you don't know at any time whether stack lift is enabled or not, unless you know explicitly what its last operation was.

The HP-41 translator does what the HP-35 and its descendants should have done: it has a special key (in this case the [SPC] key) that has no role in life except to terminate digit entry (or to separate commands). If you key in two numbers, separated by a space, the first number is lifted into the Y-register by the second, which goes into the X-register. No stack-lift disable is needed, period. What about CLX, you ask? Well, if you want zero, press [0]. If you want to replace X, press [RDN]. Why complicate life by making CLX play both roles?

Note: stack lift disable is "implemented" in HP-41 programs, so that programs transferred from the HP-41 will run on the HP-71 without modification by the user. The translation program TRANS41 figures out whether the stack should be lifted or not, and writes the translated program accordingly. (This works for all functions except ANUM, which is indeterminate in its stack use.) Here's a puzzle for RPN experts: How many forms of ENTER^ are necessary to handle all possible situations in programs? For example, if ENTER^ precedes a RCL, it can be replaced by a NOP. What other cases are there?

COMMAND LINE VS. KEY-PER-FUNCTION

The HP-41 is strictly a key-per-function calculator. HP-71 FORTH and BASIC both use a command line approach instead, i.e., you type in one or more commands together using a simple line editor, then press [ENDLINE] to execute the commands. This is less keystroke efficient than the key-per-function method, but more flexible in that you can execute several instructions together, and you can edit the command line. The command stack is another bonus of this method.

The HP-41 translator allows you to use either method. The default mode is command lines. You can type in up to 96 characters of functions or numbers together, each entry separated by a space, then press [ENDLINE] to execute the whole sequence. Each command line is saved in the command stack. Thus to compute 5+SIN(25)*10, you can key in

5 25 SIN 10 * [ENDLINE]

Or, if you want to see intermediate results:

5 [ENDLINE]
25 SIN [ENDLINE]
10 * [ENDLINE]

But if you prefer key-per-function operation, you can make the built-in file KEYS41 the active keys file, so that in user mode, any HP-41 function can be immediate execute. For example, the [+] key is assigned to the string " +", so that pressing that key appends a + to the edit line and performs endline. The KEYS41 file provides immediate execute key assignments for most of the HP-41 keyboard functions. You can change or add any key assignment using normal HP-71 key assignment procedures.

The translator is actually more keystroke efficient than the HP-41 in two cases. First, you can execute any function by just spelling it, without needing to resort to the clumsy HP-41 XEQ [ALPHA] <function> [ALPHA]. Second, register functions can directly access any register, since they are not tied to prompted input. Thus STO 1234 or X<> 5432 or VIEW IND 653 are legitimate commands.

ALL THIS AND ALGEBRAIC TOO?

The principal (only) virtue of algebraic calculators is that you can evaluate an algebraic expression by typing it in exactly as it is written in normal algebraic form. RPN calculators shine when you are doing interactive calculating, where you don't know in advance the precise path the calculation will follow. The HP-41 Translator is actually the world's first RPN/Algebraic calculator. That is, you can type in any algebraic expression (actually any numeric expression understood by the HP-71 BASIC interpreter), and the calculator will evaluate it and return the result to the X-register. The only constraint is that the expression contain no spaces, which act as expression terminators. Thus, to evaluate the expression used as an example above, you do not have to parse the expression mentally into RPN; just type in

5+SIN(25)*10 [ENDLINE]

The result 9.23 goes into X, lifting the previous stack contents.

You can mix RPN or algebraic format as you please, for example,

1+2+3 4+5 2*6 9-8*2 + - * [ENDLINE]

returns the result 24.00--evaluating

(1+2+3)*((4+5)-((2*6)+(9-8*2))).

HOW DOES IT WORK?

One can imagine several ways of providing HP-41 capability on the HP-71. The most obvious, perhaps, is to translate the HP-41 operating system into the HP-71 CPU assembly language. While this could provide exact compatibility and maximum program execution speed, it would be a massive task, complicated by the differing structures of the CPU's, memories and keyboards of the HP-41 and the HP-71. The system would also have to be integrated somehow with the native HP-71 operating system. And all future functions to be added to the system would have to be written in assembly language.

Another approach is to write an HP-41 interpreter in a high-level language. BASIC is an obvious choice, since it is provided on the HP-71. However, this approach does not take advantage of the normal program flow in the language of choice--there are two levels of interpretation required, which likely would lead to slow HP-41 program execution. The evaluation of FORTH, which is intrinsically much faster than BASIC, as a language for writing an HP-41 interpreter, actually led to the introduction of the FORTH/Assembler ROM for the HP-71.

A third method is to translate HP-41 programs into a language already understood by the HP-71. This allows the translated programs to run under the control of the normal language interpreter, which eliminates the speed penalty of a second level of interpretation. It also allows you to take advantage of all of the features of the native language, in modifying or extending the HP-41 user language. It does require the preliminary step of translating the original program into the new language, but this needs only to be done once, and does not affect run-time execution speed.

The HP-41 Translator Pac, as you might guess from its name, takes the translation approach. HP-41 programs are translated from HP-41 user language into compiled HP-71 FORTH, and executed like any other FORTH program, under control of the FORTH "inner loop." FORTH was chosen over BASIC for this purpose for several reasons, execution speed, RPN logic, and the existence in HP-71 FORTH of most of the HP-41 floating-point arithmetic operations, including the 5-level HP-41-like floating-point stack. In addition, FORTH is a logical "next language" for HP-41 user language aficionados.

The choice of FORTH does have certain disadvantages. Because FORTH is compiled, it is generally not possible to edit a program in its final, executable form. Nor does normal FORTH provide an easy means of single-stepping. There are no program line numbers, so if you halt a program, there is no way to tell where it halted. FORTH's memory management system is relatively primitive, which precludes any straightforward implementation of CLP. And the HP-71 FORTH system requires a hard-addressed kernal--the upshot of which is that the Translator Pac and the FORTH/Assembler ROM can not both be present in the HP-71 at the same time, since they must both be hard addressed at the same addresses. All of this notwithstanding, it was the judgment of the Pac's developer that the advantages of FORTH outweigh the disadvantages.

The first step in converting an HP-41 program for HP-71 execution consists of transferring a text version of the program to an HP-71 text file. You can do this either by using the text editor (included in the Translator Pac) to type the program in by hand, or you can transfer the program via HP-IL from HP-41 memory to the HP-71, using the READ41 program in the Pac. READ41 (written in BASIC) copies HP-41 program lines sent on HP-IL by the HP-41 HP-IL module function PRP, which conveniently converts HP-41 program bytes into ASCII text. In either case, the program ends up in the HP-71 as a replica of an HP-41 program listing. In this form, you can use the editor further to modify the program, delete lines, search/replace, etc. You can also insert comments into the program. In this text form, you can save the program on magnetic media. The Translator Pac can not read HP-41 programs saved on mass storage by the HP-41 because they are in a tokenized form.

The next step is translation from HP-41 user language into a form suitable for use by the FORTH compiler. This formidable task is accomplished by the Pac program TRANS41 (also BASIC). The output of TRANS41 is another text file that looks very similar to the original program, but has certain important differences:

1) FORTH memory management instructions are inserted;

2) The program's stack-lift enable/disable logic is sorted out;

3) Program comments are removed;

4) HP-41 number entry lines are converted to HP-71 format;

5) Test functions are augmented with a FORTH branch word; and

6) Alpha program lines are tweaked into a suitable format.

Any HP-41 function not requiring special handling according to this list is left unchanged. This has the benefit that new functions can be added to the HP-41 translator by simply adding new words to the HP-41 portion of the FORTH dictionary. The translator will just pass the new words along as is and let the compiler worry about them.

When translation of a program is complete, TRANS41 will continue at your option with the last step of the process, compilation of the program into the FORTH dictionary. This step can also be performed from the HP-41 emulator environment. Once the program is compiled, you can use the HP-41 functions RUN, GTO, XEQ, RTN, and END just as you would on the HP-41.

The HP-41 emulator is activated by the keyword HP41, which can execute either from BASIC or from FORTH. The first time you run the emulator, you must specify an initial SIZE to reserve some memory for HP-41 data registers. After that, you can set the size using SIZE or PSIZE just as you would on the HP-41. To return to FORTH or BASIC, you just type FORTH or BASIC and hit [ENDLINE]. The HP-41 environment is preserved until you reenter it (some HP-41 flags are initialized, more or less like turning the HP-41 off then on).

PERFORMANCE

HP-41 programs executed by the Translator Pac will run significantly faster than their HP-41 counterparts. The exact amount of speed increase is program dependent; the range is from 3 to 8 times faster. Straight-line, math intensive programs will run at the higher end of the range. The gamma function program from the High Level Math Solutions book executes about 7.5 times faster on the HP-71 than on the HP-41. Programs with lots of branches will run more slowly. The multiple curve fit program from Curve Fitting for Programmable Calculators, by William M. Kolb, runs about 4.5 times faster on the HP-71.

The Translator Pac is more profligate in memory use than the HP-41. The final, compiled version of a program needs about 2.5 times as much memory in the HP-71 than in the HP-41. This difference corresponds to the difference between the one-byte program tokens used by the HP-41 and the 5 nibble FORTH execution addresses used by the HP-71. The translator also requires three different versions of a single program: the HP-41 user language text, the translated text, and the compiled version. A maximum of two of these needs to be present at any time. If you have a tape or disk drive, only one version needs to reside in HP-71 memory at any time.

RELATION TO THE FORTH/ASSEMBLER ROM

The Translator Pac FORTH system is very similar to that contained in the HP-82441A FORTH/Assembler ROM. At first approximation, the Translator Pac just is the FORTH/Assembler ROM, with the HP-41 vocabulary substituted for the Assembler (the KEYBOARD IS lex file is also not present in the Translator Pac). This has the drawback noted previously that both modules can not be plugged into the HP-71 simultaneously. But further, the two FORTH systems cannot share the same RAM files. This is due to the differing organizations of the system portions of the RAM files, and to the fact that the ROM-based FORTH dictionaries are different, so that the compilation addresses of the ROM words are not the same for the two systems. The FORTH/Assembler ROM's RAM file is named FORTHRAM; the Translator Pac's is named FTH41RAM. The different names should help programmers keep the two types of files sorted out.

Here is a brief summary of the primary differences between the two FORTH systems:

*The Translator Pac does not contain the Assembler, the associated words ASSEMBLE, PAGESIZE, LISTING, and VARID, and the Assembler user variables.

*With the exception of the Assembler words, the Translator Pac ROM dictionary is a superset of that of the FORTH/Assembler ROM. The Translator dictionary is organized into two vocabularies: FORTH and HP41V. The former is the parent vocabulary of the latter, so that FORTH words are available when the context vocabulary is HP41V, but not vice-versa.

*The Translator Pac FORTH vocabulary is augmented by numerous HP-41 floating-point words that are not included in the FORTH/Assembler ROM.

*Translator Pac floating-point words follow the HP-41 convention that errors leave the floating-point stack intact. The FORTH/Assembler words drop the stack, update LASTX, etc., before error-checking.

*The Translator Pac HP41V vocabulary contains HP-41 words that depend on HP-41 data structures or use post-fix notation.

*The user variable area in the FTH41RAM file contains the HP-41 flags, program pointer, return stack, alpha register, size and sigma register variables, and other HP-41 system variables.

*The FTH41RAM user dictionary begins with the FORTH word, but also contains the HP41V vocabulary word, and a null word used to link the various RAM and ROM dictionaries together.

*The outer interpreter loop in the Translator Pac checks an emulator-active flag following interpretation of the input buffer. If the flag is clear, the OK { n } message is displayed. If set, a vectored HP-41 display word is executed (typically, to display the X-register).

*HP-41 error messages (Alpha Data, Data Error, etc.) are added to the system error table in the Translator Pac.

*The [ATTN] key and poll check carried out during execution of semicolon and branching has been rewritten for the Translator Pac, resulting in somewhat faster FORTH execution.

SUMMARY

The Translator Pac extends the flexibility of the HP-71 by providing an extensive RPN calculator/programming capability closely modeled on that of the premier RPN calculator, the HP-41. It is a translator rather than an emulator. Its calculator properties are designed to work with the strengths of the HP-71 rather than to be a keystroke copy of the HP-41. HP-41 programs are translated into FORTH, a language of more general application than HP-41 user language.

The primary purpose of the Pac is to allow HP-71 owners to access the HP-41 software base, either their own programs, or published programs. The real-time calculator capabilities of the Pac are necessary to support this objective. Inclusion of an editor is a step beyond the initial purpose, in that programmers can write new HP-41 language programs, or modify existing ones, on the HP-71. Full access to a FORTH language system goes even further, since programs can be written that have no backwards compatibility path to the HP-41.

HP-41 users/programmers can use the HP-41 translator without any knowledge of FORTH. (FORTH programmers can use the FORTH system without any regard for the HP-41 emulator--but they would be better off with the FORTH/Assembler ROM, which contains an assembler.) You might view the FORTH language system underlying the HP-41 system as a bonus feature that can provide a growth path for HP-41 programmers to carry their RPN skills into a language similar in spirit to HP-41 language, but providing vastly improved performance. The price for this performance is a requirement for more careful programming practices--FORTH does not have the fool-proof system protection of HP-41 language or BASIC.

The documentation of the FORTH system in the Translator Pac manual is taken mostly whole from the FORTH/Assembler ROM manual. That is, it is only a brief description of the properties of the system, plus a list of definitions for each of the words in the built-in dictionary. The documentation is suitable only for programmers already familiar with FORTH--there is no tutorial material provided. There are many fine FORTH books available; to learn HP-71 FORTH from scratch, you will have to study one of these books, keeping in mind the differences between HP-71 FORTH and "standard" FORTH, which are described in the Pac owner's manual.

by William C. Wickes
Hewlett-Packard Portable Computer Division

PAM FOR THE HP-75

This program was written some time ago in an effort to make the 75 act more "friendly" in everyday use and eliminate much of the drudgery involved in typing CAT ALL, COPY, PLIST, EDIT, etc. many time. The program is an outgrowth of discussions with Jim Walters (7692), without whos original idea this program may never have been written. I use this program all of the time and find it speeds operation a great deal. It operates much the way PAM does on the 110/150 and hence is named such. It's not perfect, but is a real convenience.

The reader should note the system configuration for which this is written and make the modifications necessary for his system. The 75 is used with or without the MC 80-column video interface, 82161A Cassette Drive, and either the 82162A Thermal (strip) Printer or 2225B Thinkjet Printer. The AUTOLOOP program or I/O ROM is required. I have PAM assigned to autostart (CHR$(159)) and also [SHIFT] [RUN] (CHR$(173)), the latter because it's often necessary to start PAM running again (such as after editing a file).

Here is the command summary:

[FET]	Display a menu of most commands
P	PLIST the displayed file to specified device:
	T designates output to Thinkjet, Normal (N) or Compressed (C)
	S output to 82162A strip printer
	V output to 80-column video display
L	LIST the displayed file to DISPLAY IS device
W	Write the displayed file to to Mass Storage (M) or to Card (C)
R	Read specified file name in from Mass Storage (M) or from Card (C)
M	Display available memory (bytes and Kbytes)
C	Clear either DISPLAY IS device (D) or Loop (L)
Shf[DEL]	Purge displayed file
[EDIT]	Edit displayed file
[RUN]	Run displayed file (if BASIC)
[TIME]	Display the current time and date
up cursor	Go up one entry in the CAT (sort of like a continuous CAT ALL)
dn cursor	Go down one entry in the CAT
[CTL] L	LIST keys
[CTL] [EDIT]	EDIT keys

HP-IL commands could be used to allow more than one of the same type device on the loop and selection of which one (or all) of them would be active listeners/talkers.

Brian Walsh (6951)
2103 Huntingdon Chase
Dunwoody, GA 30338

```
10Ø DISP CHR$(27)&"E" @ F=Ø
2Ø ON ERROR WIDTH 32 @ DELAY .3
3Ø DISPLAY IS ":D1" @ WIDTH 8Ø @ PWIDTH INF @ DELAY Ø @ OFF ERROR
4Ø L$=CHR$(137)&"PLWRMC"&CHR$(17Ø)&CHR$(131)&CHR$(141)&CHR$(129)&
   CHR$(132)&CHR$(133)
5Ø L$=L$&CHR$(12)&CHR$(195)
6Ø DISP CAT$(F)
7Ø K$=UPRC$(KEY$) @ IF K$="" THEN 7Ø
8Ø L=POS(L$,K$) @ IF L=Ø THEN 7Ø
9Ø ON L GOTO 1ØØ,18Ø,17Ø,33Ø,33Ø,27Ø,29Ø,42Ø,15Ø,28Ø,16Ø,11Ø,13Ø,
   43Ø,41Ø
1ØØ DISP "P/Lst Wrt Rd Mm Clr Pu Ed Run Tm" @ WAIT 1.5 @ GOTO 6Ø
11Ø IF F>Ø THEN F=F-1
12Ø GOTO 6Ø
13Ø IF CAT$(F+1)#"" THEN F=F+1
14Ø GOTO 6Ø
15Ø GOSUB 44Ø @ EDIT CAT$(F) @ OFF ERROR @ END
16Ø DISP TIME$&"   "&DATE$ @ WAIT 1.2 @ GOTO 6Ø
17Ø GOSUB 44Ø @ LIST CAT$(F) @ OFF ERROR @ GOTO 6Ø
18Ø GOSUB 44Ø @ DISP "Thinkjet or Strip or Video ?"
19Ø K$=UPRC$(KEY$) @ IF K$#"T" AND K$#"S" AND K$#"V" THEN 19Ø
2ØØ IF K$="S" THEN PRINTER IS ":P1" @ GOTO 25Ø
21Ø IF K$="V" THEN PRINTER IS ":D1" @ DISP CHR$(27)&"[" @ GOTO 25Ø
22Ø PRINTER IS ":P2" @ PRINT CHR$(27)&"&sØC" @ DISP "Normal or
    Compressed ?"
23Ø K$=UPRC$(KEY$) @ IF K$#"N" AND K$#"C" THEN 23Ø
24Ø IF K$="C" THEN PRINT CHR$(27)&"&k2S" ELSE PRINT CHR$(27)&
    "&kØS"
25Ø PRINT CAT$(F) @ PRINT TIME$&"   "&DATE$ @ PRINT @ PLIST CAT$(F)
    @ PRINT CHR$(12)
26Ø OFF ERROR @ GOTO 6Ø
27Ø DISP MEM;RES/1Ø24 @ WAIT 1.2 @ GOTO 6Ø
28Ø GOSUB 44Ø @ CALL CAT$(F) @ OFF ERROR @ GOTO 6Ø
29Ø DISP "Display or Loop ?"
3ØØ K$=UPRC$(KEY$) @ IF K$#"D" AND K$#"L" THEN 3ØØ
31Ø IF K$="D" THEN DISP CHR$(27)&"E" @ GOTO 6Ø
32Ø GOSUB 44Ø @ CLEAR LOOP @ OFF ERROR @ GOTO 6Ø
33Ø DISP "Mass Sto. or Card ?"
34Ø C$=UPRC$(KEY$) @ IF C$#"M" AND C$#"C" THEN 34Ø
35Ø IF K$="R" THEN 38Ø
36Ø IF C$="C" THEN COPY CAT$(F) TO CARD @ GOTO 6Ø
37Ø GOSUB 44Ø @ COPY CAT$(F) TO ":M1" @ OFF ERROR @ GOTO 6Ø
38Ø INPUT "File name to read in ? ";K$
39Ø IF C$="M" THEN GOSUB 44Ø @ COPY K$&":M1" TO K$ @ OFF ERROR
    ELSE COPY CARD TO K$
4ØØ GOTO 6Ø
41Ø EDIT KEYS @ END
42Ø PURGE CAT$(F) @ GOTO 6Ø
43Ø LIST KEYS @ GOTO 6Ø
44Ø ON ERROR GOTO 6Ø
45Ø RETURN
```

END LINE

BIORHYTHMS

This program calculates Meinderts biorhythm if you input anything which is not a valid date (e.g. zero) for the current date, and any biorhythm if you enter a birthdate and any date for that day. Change line 06 according to your own birthdate (note the format, DMY or MDY, depending upon the status of flag 31!). Besides giving values for physical, sensitive, and cognitive cycles, the output shows whether the curve is ascending (+) or descending (-).

To use, key in the birthdate, press ENTER↑, key in the biorhythm date, XEQ "BIOR". R/S after the last output will show your biorhythm for the next day.

The program uses no registers, the Extended Functions and Time modules are required.

Synthetic text lines (in hex):

14: F5, 17, 50, 48, 59, 53
16: F5, 1C, 53, 45, 4E, 53
18: F5, 21, 43, 4F, 47, 4E

Meindert Kuipers (7612)
Laan 2/10
9712 AV Groningen
NETHERLANDS

Eric van der Wateren (8146)
Aquamarijnstraat 57
9743 PB Groningen
NETHERLANDS

```
Ø1*LBL "BIOR"
Ø2 SF 25
Ø3 DDAYS
Ø4 FS?C 25
Ø5 GTO ØØ
Ø6 24.Ø2196
Ø7 DATE
Ø8 DDAYS
Ø9*LBL ØØ
1Ø RCLFLAG
11 X<>Y
12 ABS
13 FIX 1
14 "■PHYS"
15 XEQ Ø1
16 "■SENS"
17 XEQ Ø1
18 "!COGN"
19 XEQ Ø1
2Ø RDN
21 STOFLAG
22 R↑
23  E
24 +
25 RTN
26 GTO ØØ
27*LBL Ø1
28 ENTER↑
29 "⊢: "
3Ø ATOX
31 MOD
32 LASTX
33 /
34 36Ø
35 *
36 SIN
37 X≠Ø?
38 X>Ø?
39 "⊢ "
4Ø ARCL X
41 X<> L
42 COS
43 X<Ø?
44 "⊢ -"
45 X>Ø?
46 "⊢ +"
47 AVIEW
48 RDN
49 END

112 BYTES
```

REGISTERS: 17
ROW 1 (1-4)
ROW 2 (4-7)
ROW 3 (8-14)
ROW 4 (14-17)
ROW 5 (17-21)
ROW 6 (22-30)
ROW 7 (30-39)
ROW 8 (39-46)
ROW 9 (46-49)

R/S

Copyright © 1984

SEPTEMBER
OCTOBER
1 9 8 4
V 3 N 5

COMPUTER JOURNAL OF PPC®

Copyright © PPC 1984

PPC® CALCULATOR JOURNAL

SEPTEMBER
OCTOBER
1 9 8 4
V 1 1 N 8

The Personal Programming Center is an international users group of People Programming Computers

A P P L I C A T I O N S

P R O G R A M S

R E G U L A R   C O L U M N S

Copyright © PPC 1984

PPC® CALCULATOR JOURNAL

M A Y
1 9 8 4
V 11 N 4

The Personal Programming Center is an international users group of People Programming Computers.

A P P L I C A T I O N S

P R O G R A M S

R E G U L A R   C O L U M N S

Copyright © 1985

PPC® JOURNAL

J A N U A R Y
1 9 8 5
V 1 2 N 1

The Personal Programming Center is an international users group of People Programming Computers

A P P L I C A T I O N S

P R O G R A M S

R E G U L A R   C O L U M N S