®PPC™ Personal Programming Center

V14N7

JULY 1987

JOURNAL

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

Contents

PPC, the Personal Programming Center, is a California non-profit public 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, P.O. BOX 90579, Long Beach, California 90809 USA, Telephone (213) 498-2554

RELATIONS BETWEEN A LINE AND A CONIC

By Ted Levine #12360
The Villages 89D
Freehold, NJ 07728

If a line and a conic are given, it is evident that
a) the line is a secant of the conic
b) the line is tangent to the conic,or
c) the line does not meet the conic.
The line is a first degree equation and the conic a second degree equation. Eliminate (y) from the line equation by substituting into the second degree equation using program 'LICO' resulting in the quadratic equation $Ax^2+Dx+F=0$. Solving the quadratic by the use of 'QE' roots 1 and 2 will be obtained.
Let Delta = D^2-4AF
1. If Delta is positive the line is a secant
2. If Delta = 0 the line is a tangent
3. If Delta is negative the line will not meet the conic,the roots will be imaginary thus no points of intersection.
In the quadratic equation
a) If A=0 one root is at infinity
b) If A=0 and D=0 then both roots are at infinity,the line is said to be' tangent at infinity'
c) If A=0,D=0,and F=0 the equation will be satisfied by all values of x,thus an infinite number of roots.
NOTE: IN XEQ'LICO' due to internal operation of calculator the result may show SEC= with a value close to ZERO. The program will stop at Line 10 of 'DELTA'.If the value is close to ZERO enter ZERO and R/S, the result will show TAN=0.
I. $Ax^2+Bxy+Cy^2+Dx+Ey+F=0$
II. $y=mx+b$
Enter I as in the CONIC programs and enter m STO 01 and enter b STO 02.

```
XEQ'CONIC'
XEQ'LICO'
RCL 18↑RCL 21↑RCL 23
XEQ'QE' See ROOT 1,ROOT 2
II y=mx+b
ROOT 1= x  y=
ROOT 2=x  y=
```

These are the points of intersection or tangency.

EXAMPLE I

I $X^2+2XY+Y^2+4X-4Y=0$

II X+Y=1 (Y=mX+b) Y=-X+1 m=-1,b=1

DO	SEE
1 STO 18	
2 STO 19	
1 STO 20	
4 STO 21	
4 CHS STO 22	
XEQ'CONIC'	PARABOLA
XEQ'RMVXY'	-5.65685
RCL 30	45°
RCL 34	2
RCL 35	0
RCL 36	0
RCL 37	0
RCL 38	-5.65685
RCL 23	0

$2(X')^2-5.65685Y'=0$<2
Divide through by 2
$(X')^2=2.82843Y$
This is in standard form for parabola
2p=2.82843 p=1.41422 p/2=.70711

(m) -1 STO 01
(b) 1 STO 02

XEQ'LICO'	64
R/S	SEC=64
RCL 18	0
RCL 21	8
RCL 23	-3

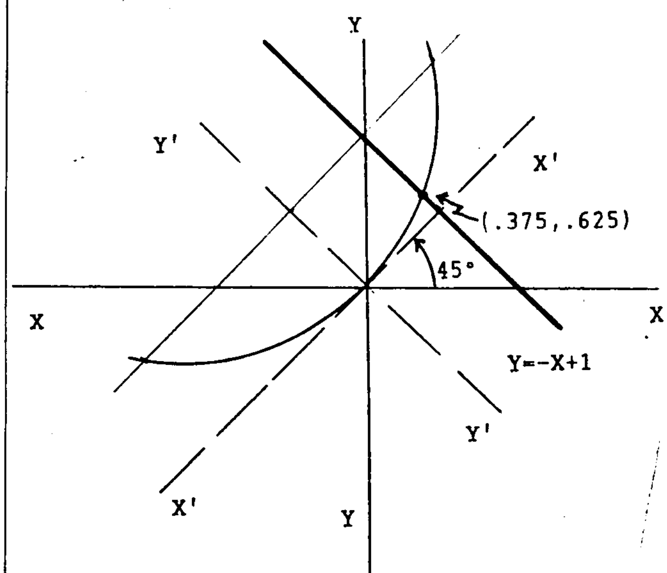
$0X^2+8X-3=0$
8X=3 X=.375
Y=-X+1 Y=-.375+1 Y=.625

Line intersects parabola at (.375,.625)

EXAMPLE I

I X²+2XY+Y²+4X-8Y=0

II Y=-X+1



EXAMPLE II

I $X^2+Y^2-16=0$

II X-2Y+20=0 (Y=mX+b) Y=.5X+10 m=.5 b=10

I is a circle with radius of 4

While the Personal Programming Center (PPC), publishers of the PPC Journal, is operated on a volunteer basis, the opinions, insights and predictions of the author's contributing articles to be published in the PPC Journal do not necessarily represent the views and opinions of PPC. If you have any questions or comments, send your correspondence to PPC, P.O. Box 90579, Long Beach, California 90809-0579.

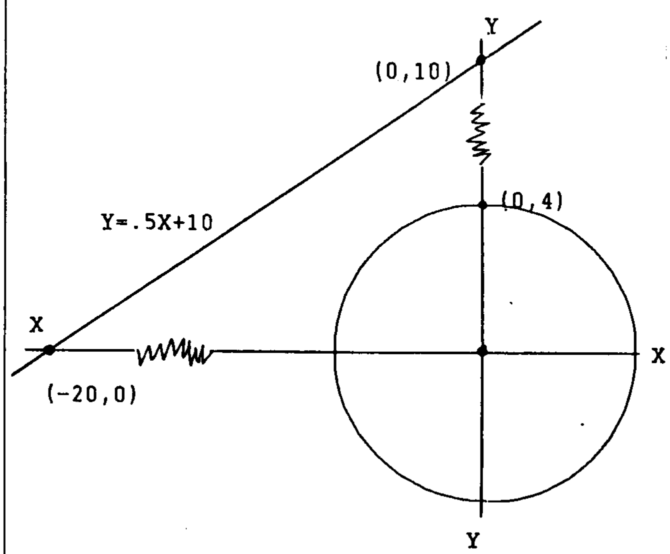
DO	SEE
1 STO 18	
1 STO 20	
16 CHS STO 23	
(m) .5 STO 01	
(b) 10 STO 02	
XEQ"LICO"	-320
R/S	NTMT=-320

The line does not meet the circle

EXAMPLE II

I $X^2+Y^2-16=0$

II Y=.5X+10



EXAMPLE III

I 2XY-3X-Y=0

II Y+3X-6=0 (Y=mX+b) Y=-3X+6 m=-3 b=6

DO	SEE
2 STO 19	
3 CHS STO 21	
1 CHS STO 22	
(m) 3 CHS STO 01	
(b) 6 STO 02	
XEQ"CONIC"	HYPERBOLA
XEQ"LICO"	0
R/S	TAN=0
RCL 18↑	-6
RCL 21↑	12
RCL 23	-6

$(-6X^2+12X-6=0)$

DO	SEE
XEQ"QE"	1
R/S	ROOT 1=1
R/S	ROOT 2=1

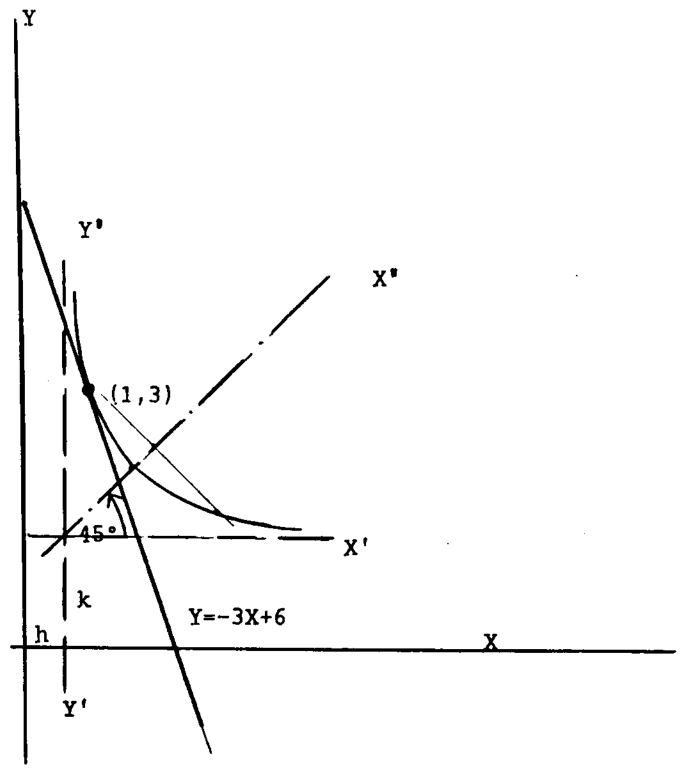
Y=-3X+6 X=1 Y=3

Point of tangency (1,3)

EXAMPLE III

I 2XY-3X-Y=0

II Y=-3X+6



EXAMPLE IV

I $8X^2-6Y^2+16X-32=0$

II 2X-3Y=0 (Y=mX+b) Y=2/3X m=2/3 b=0

DO	SEE
8 STO 18	
6 CHS STO 20	
16 STO 21	
32 CHS STO 23	
(m) 2↑3 / STO 01	
XEQ"CONIC"	HYPERBOLA
XEQ"LICO"	938.6
R/S	SEC=938.6
RCL 18 ↑	5.33333
RCL 21 ↑	16
RCL 23	-32

$(5.33333X^2+16X-32=0)$

DO	SEE
XEQ"QE"	1.37228
R/S	ROOT 1=1.37228
R/S	ROOT 2=-4.37228

Y=2/3 X

X=1.37228 Y=.91485

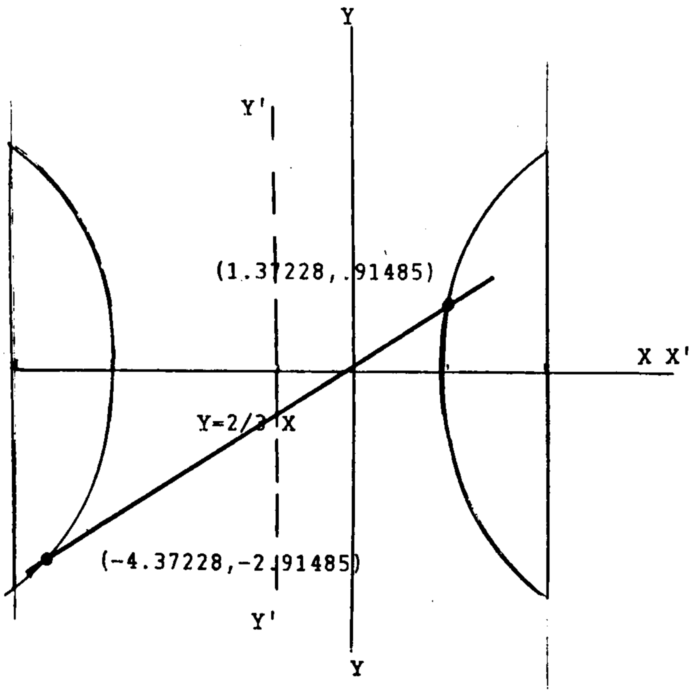
X=-4.37228 Y=-2.91485

Points of intersection are
(1.37228,.91485) (-4.37228,-2.91485)

EXAMPLE IV

I $8X^2-6Y^2+16X-32=0$

II Y=2/3 X



EXAMPLE V

I $X^2+Y^2-6X-8Y=0$

II X-2Y-6=0 (Y=mX+b) Y=.5X-3 m=.5 b=-3

DO	SEE
1 STO 18	
1 STO 20	
6 CHS STO 21	
8 CHS STO 22	
(m) .5 STO 01	
(b) 3 CHS STO 02	
XEQ"CONIC"	ELLIPSE
XEQ"LICO"	4
R/S	SEC=4
RCL 18 ↑	1.25
RCL 21 ↑	-13
RCL 23	33

$(1.25X^2-13X+33=0)$

XEQ"QE"	6
R/S	ROOT 1=6
R/S	ROOT 2=4.4

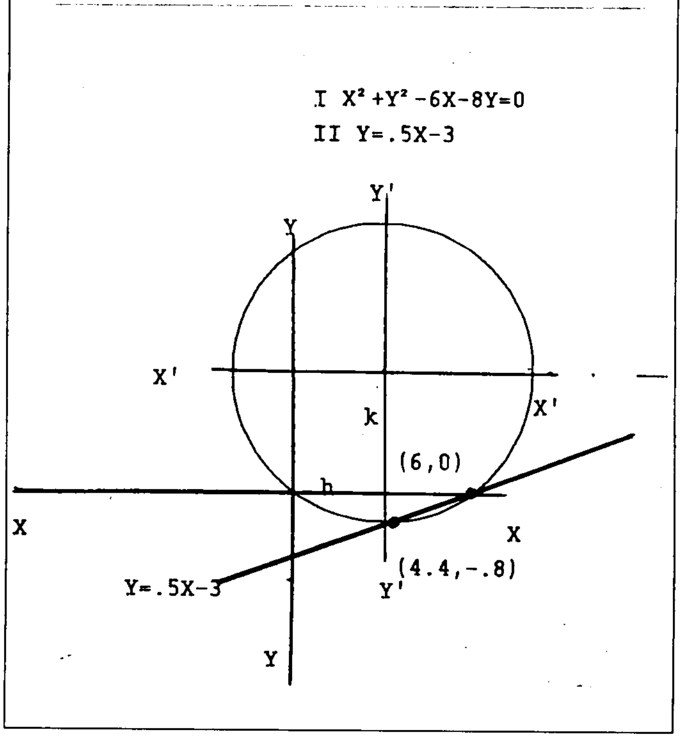
Y=.5X-3

X=6 Y=0

X=4.4 Y=-.8

Points of intersection are
(6,0) (4.4,-.8)

EXAMPLE V



EXAMPLE VI

I $X^2-Y^2+3X-3Y=0$

II X-Y=0 (Y=mX+b) Y=X m=1 b=0

DO	SEE
1 STO 18	
1 CHS STO 20	
3 STO 21	
3 CHS STO 22	
(m) 1 STO 01	
XEQ"CONIC"	X LINES
XEQ"LICO"	0
R/S	TAN=0
RCL 18 ↑	0
RCL 21 ↑	0
RCL 23	0

Line is tangent at all points or coincidental to one of the intersecting lines that is at 45° to the X axis.

EXAMPLE VII

I $X^2-6XY+9Y^2+X-3Y-2=0$

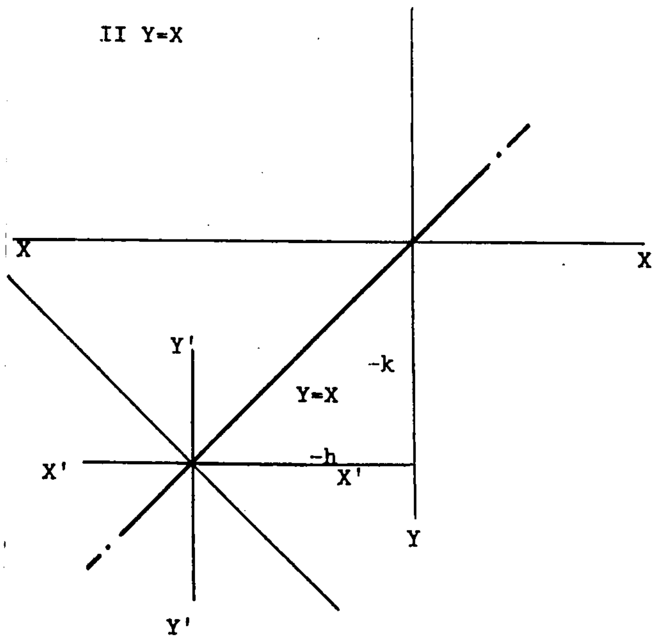
II X-3Y=1 (Y=mX+b) Y=1/3 X-1/3,m=1/3,b=-1/3

DO	SEE
1 STO 18	

EXAMPLE VI

I $X^2-Y^2+3X-3Y=0$

II Y=X



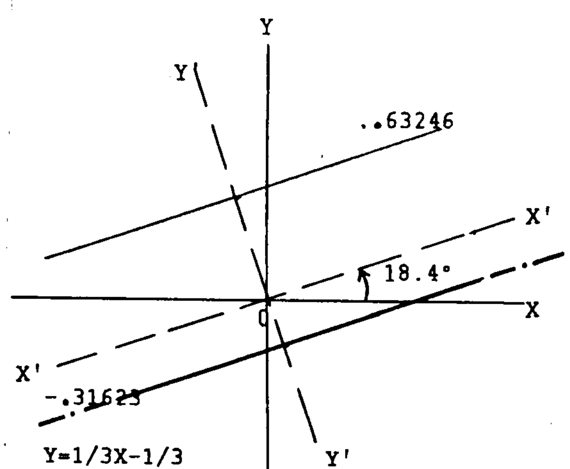

```
6 CHS STO 19
9 STO 20
1 STO 21
3 CHS STO 22
2 CHS STO 23
(m) 1↑3 / STO 01
(b) 1↑3 / CHS STO 02
```

XEQ"CONIC" // LINES

EXAMPLE VII

I $X^2-6XY+9Y^2+X-3Y-2=0$

II Y=1/3X-1/3




```
XEQ"LICO"          -3E-20
```

This value is close to ZERO.
Enter Zero and
R/S TAN=0

```
RCL 18 ↑           0
RCL 21 ↑           0
RCL 23             0
```

Line is tangent at all points or coincidental to one of the parallel lines. It will be coincidental to the parallel line that is below the X axis since b=-1/3.

```
01 LBL"LICO"
02 RCL 01
03 RCL 19
04 *
05 ST+ 18
06 RCL 19
07 RCL 02
08 *
09 ST+ 21
10 RCL 20
11 RCL 01
12 ST* X
13 *
14 ST+ 18
15 RCL 20
16 RCL 01
17 RCL 02
18 *
19 *
20 ST+ X
21 ST+ 21
22 RCL 20
23 RCL 02
24 ST* X
25 *
26 ST+ 23
27 RCL 22
28 RCL 01
29 *
30 ST+ 21
31 RCL 22
32 RCL 02
33 *
34 ST+ 23
35 XEQ"DELTA"
36 END
```

```
01 LBL"DELTA"
02 RCL 21
03 ST* X
04 4
05 RCL 18
06 RCL 23
07 *
08 *
09 -
10 STOP
11 X>0?
12 GTO "A"
13 X=0?
14 GTO"B"
15 X<0?
16 GTO"C"
17 LBL"A"
18 STO 01
19 CLA
20 "├ SEC="
21 ARCL 01
22 AVIEW
23 STOP
24 LBL"B"
25 STO 01
26 CLA
27 "├ TAN="
28 ARCL 01
29 AVIEW
30 STOP
31 LBL"C"
32 STO 01
33 CLA
34 "├ NTMT="
35 ARCL 01
36 AVIEW
37 STOP
38 RTN
39 END
```

```
01 LBL"QE"
02 CF 04
03 X<>Z
04 ST/Z
05 ST+X
06 /
07 CHS
08 ENTER↑
09 ENTER↑
10 X↑2
11 R↑
12—
13 X<0?
14 SF 04
15 ABS
16 SQRT
17 ST-Z
18 X<>Y
19 FC? 04
20 +
21 RTN
22 SF 21
23 FS?C 04
24 GTO 14
25 "ROOT 1="
26 ARCL X
27 AVIEW
28 "ROOT 2="
29 ARCL Y
30 AVIEW
31 RTN
32 LBL 14
33 CLA
34 ARCL X
35 "├ +-"
36 ARCL Y
37 "├ I"
38 AVIEW
39 END
```

(Robert Groom 5127)

R/S

To Whom It May Concern,

I have back issues of HP65 Notes, Vol. 1 #1 thru Vol. 4 #10. Also PPC Journal Newsletter Vol. 5 #1 thru December, Vol. 9 #8. Be happy to send to you at location you advise, you pay postage.

Elmer E. Hinson #291
519 Cookston Avenue
Springfield, Ohio 45503

Perhaps some member in the Ohio area would like to contact PPC'er Elmer Hinson.

000033D **Program Description I**

Program Title MICROSTRIP CALCULATIONS

Contributor's Name

Address

City **State** **Zip Code**

Program Description

This program accepts conductor width w, dielectric thickness h, and relative permittivity ϵ_r, and computes relative phase velocity v_r and characteristic impedance Z_c for lossless line. The following formulas are used.

$$\epsilon_{eff} = \frac{\epsilon_r + 1}{2} + \frac{\epsilon_r - 1}{2}\left(1 + 10\frac{h}{w}\right)^{-1/2}$$

$$v_r = \frac{1}{\sqrt{\epsilon_{eff}}}$$

$$Z_0 = \begin{cases} 60 \ln\left(8\frac{h}{w} + \frac{w}{4h}\right), & \frac{w}{h} \leq 1 \\ \dfrac{120\pi}{\frac{w}{h} + 2.42 - 0.44\frac{h}{w} + \left(1 - \frac{h}{w}\right)^6}, & \frac{w}{h} > 1 \end{cases}$$

$$Z_c = v_r Z_0$$

where

ϵ_r = relative permittivity of dielectric

ϵ_{eff} = effective permittivity of dielectric

h = dielectric thickness

w = width of microstrip

v_r = relative phase velocity of lossless line

Z_0 = characteristic impedance of corresponding air line, Ω

Z_c = characteristic impedance of lossless microstrip, Ω

It then accepts the conductor thickness and computes a normalized conductor loss A.

Operating Limits

000033D **Program Description I**

Program Title

Contributor's Name

Address

City State Zip Code

Program Descript

$$A = \begin{cases} \dfrac{20}{\ln 10}\dfrac{h}{w Z_0}\dfrac{dB}{\Omega}, \text{ uniform current distribution} \\[2ex] \dfrac{10}{\pi \ln 10} \dfrac{\left(8\dfrac{h}{w} - \dfrac{w}{4h}\right)\left(1 + \dfrac{h}{w} + \dfrac{h}{w}\dfrac{\partial w}{\partial t}\right)}{Z_0 e^{Z_0/60}} \dfrac{dB}{\Omega}, & \dfrac{w}{h} \leq 1 \\[2ex] \dfrac{Z_0}{720\pi^2 \ln 10}\left[1 + 0.44\dfrac{h^2}{w^2} + \dfrac{6h^2}{w^2}\left(1 - \dfrac{h}{w}\right)^5\right] \\ \quad \times \left[1 + \dfrac{w}{h} + \dfrac{\partial w}{\partial t}\right]\dfrac{dB}{\Omega}, & \dfrac{w}{h} > 1 \end{cases}$$

where

$$\frac{\partial w}{\partial t} = \begin{cases} \dfrac{1}{\pi}\ln\dfrac{4\pi w}{t}, & \dfrac{w}{h} \leq \dfrac{1}{2\pi} \\[2ex] \dfrac{1}{\pi}\ln\dfrac{2h}{t}, & \dfrac{w}{h} > \dfrac{1}{2\pi} \end{cases}$$

Finally, the program accepts conductor resistivity ρ and frequency f and computes copper loss α_c, resistance per unit length R, and unloaded quality factor Q_0 using the following equations.

$$\alpha_0 = \frac{R_S A}{h}$$

$$\mu_0 = 4\pi \times 10^{-9} \text{ H/cm}$$

$$R_S = \sqrt{\pi f \mu_0 \rho}$$

Operating Limits :

$$R = 2R_S/w$$

$$\alpha_c = \frac{\alpha_0}{v_r}$$

$$Q_0 = \frac{20\pi}{\ln 10}\frac{f}{c\, v_r\, \alpha_c}$$

$$c = 3 \times 10^{10} \text{ cm/s}$$

Reference:

M.V. Schneider, "Microstrip Lines for Microwave Integrated Circuits," *Bell System Technical Journal, 48,* No. 5 (May–June 1969).

Program Description II

Example 1:

What are the characteristics of 50-mil microstrip on a 50-mil alumina ($\epsilon_r = 9.5$) substrate at 2GHz? Assume a line thickness of 1 mil and a conductor resistivity of 3 E −6.

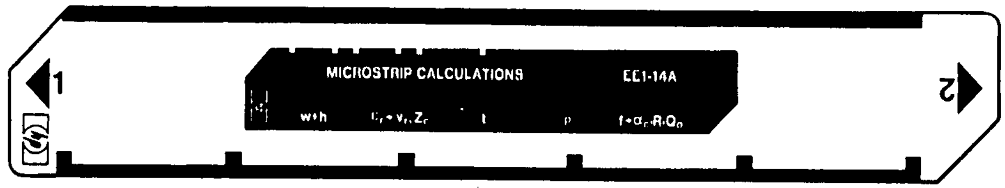
Keystrokes:	Outputs:
.05 [ENTER↑] 2.54 [×] [ENTER↑] [A]	
9.5 [B] ⟶	391.3−03 *** v_r
	49.54+00 *** Z_c
.001 [ENTER↑] 2.54 [×] [C] 3 [EEX]	
[CHS] 6 [D] 2 [EEX] 9 [E] ⟶	11.01−03 *** α
	242.4−03 *** R
	422.3+00 *** Q_0

Example 2:

Repeat the above example, but assume a uniform current distribution.

Keystrokes:	Outputs:
.05 [ENTER↑] 2.54 [×] [ENTER↑] [A]	
9.5 [B] ⟶	391.3−03 *** v_r
	49.54+00 *** Z_c
3 [EEX] [CHS] 6 [D] 2 [EEX] 9 [E] ⟶	21.25−03 *** α
	242.4−03 *** R
	218.8+00 *** Q_0

User Instructions



STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS	OUTPUT DATA/UNITS
1	Load program.			
2	Input width of microstrip	w, cm	[ENTER↑]	
	thickness of dielectric	h, cm	[A]	w/h
3	Input relative permit-			
	tivity and print relative			
	phase velocity and imped-			
	ance of lossless line.	ϵ_r	[B]	v_r
				Z_c
4	If a uniform current			
	distribution is desired,			
	skip to step 6.			
5	Input conductor thickness.	t, cm	[C]	A
6	Input conductor resistivity.	ρ	[D]	
7	Input frequency and			
	print copper loss,			
	resistance per unit			
	length and unloaded Q.	f, Hz	[E]	α_c
				R
				Q_0

Program Listing

Microstrip Transmission Line Calculations

```
*LBL A
SF 1
STO 7            Store h
X<>Y
STO 6            Store w
SF 0
X>Y?             IF w > h
CF 0             THEN clear flag 0
+
STO 9            Store h/w
1/X
STO 8            Store w/h
1
0
LN
STO 0            Store ln 10
RTN
-----------------------------------------
*LBL B
1
X<>Y
+
LSTX             Compute ε_eff
1
-
1
0
RCL 8
÷
1
+
√X
÷
+
2
÷
√X
1/X
STO 2            Store v_r
FS? 0            IF w < h
GTO 01           THEN GO TO 1.
1
RCL 9
-
6
Y^X              Compute Z_e for w > h
.
4
4
RCL 8
+
-
2
.
4
2
+
RCL 8
+
2
PI
x
X<>Y
÷
GTO 02
-----------------------------------------
*LBL 01          Compute Z_e for w < h.
8
RCL 8
÷
LSTX
4
÷
+
LN
-----------------------------------------
*LBL 02
6
0
x
STO A            Store Z_e
x
STO 1            Store Z_c
X<>Y
PRTX             Print v_r
X<>Y
PRTX             Print Z_c
RTN
-----------------------------------------
*LBL D
√X               IF uniform assumption
FS? 1            THEN GO TO 1.
GTO 01
*LBL 4
RCL 4
x
RCL 7
÷
2
x
PI
÷
STO 3            Store partial result.
RTN
-----------------------------------------
*LBL 01
2
0
RCL 8
÷
RCL 9
x
RCL A
+
STO 4
R↓
GTO 04
-----------------------------------------
*LBL E
EEX
9
÷
STO 5
√X
RCL 3
x                Print α_c
RCL 2
÷
STO B
PRTX
LSTX
x
2
x                Print R
RCL 4
÷
RCL 9
x
PRTX
2
PI
x
RCL 0
÷
RCL 5
x
3
÷                Print Q_o
RCL 2
÷
RCL 0
÷
PRTX
RTN
*LBL C
CF 1
RCL 7
2
x                Calculate ∂w/∂t
X<>Y
÷
RCL 8
PI
2
x
x
x
X>Y?
X<>Y
LN
PI
÷
RCL 8
+
1
+
FS? 0            IF w < h
GTO 01           THEN GO TO 1.
-----------------------------------------
1
RCL 9
-                Compute A for w > h
5
Y^X
5
x
.
4
4
+
RCL 9
X^2
x
1
+
x
RCL A
x
-
2
0
÷
PI
STO 0
-----------------------------------------
*LBL 01          Compute A for w < h.
3
1
RCL 9
X^2
x
1
-
x
RCL A
÷
LSTX
6
0
+
e^X
÷
.
4
-----------------------------------------
*LBL 0
+
RCL 0
+
PI
÷
STO 4            Store A
RTN
```

REGISTERS

0	1	2	3	4	5	6	7	8	9
ln 10	Z_c	v_r	α_c	A	$f/10^9$	w	h	w/h	h/w
S0	S1	S2	S3	S4	S5	S6	S7	S8	S9

A	B	C	D	E	I

LABELS

A	B	C	D	E
w → h	$\epsilon_r \to V_r, Z_C$	t → A	ρ	$f \to \alpha_c, R, Q_0$
a	b	c	d	e
0 Used	1 Used	2 Used	3	4 Used
5	6	7	8	9

FLAGS

0	w > h
1	uniform
2	
3	

SET STATUS

FLAGS	TRIG	DISP
ON OFF: 0 ☒, 1 ☒, 2 ☒, 3 ☒ (OFF)	DEG ☒ GRAD ☐ RAD ☐	FIX ☐ SCI ☐ ENG ☒ n 3

Program Listing

```
*LBLA     21 11
SF1    16 21 01
STO7      35 07
X⇄Y         -41
STO6      35 06
SF0    16 21 00
X>Y?      16-34
CF0    16 22 00
÷           -24
STO9      35 09
1/X          52
STO8      35 08
1            01
0            00
LN           32
STO0      35 00
RTN          24
*LBLB     21 12
1            01
X⇄Y         -41
+           -55
LSTX      16-63
1            01
-           -45
1            01
0            00
RCL8      36 08
÷           -24
1            01
+           -55
√X           54
÷           -24
+           -55
2            02
÷           -24
√X           54
1/X          52
STO2      35 02
F0?    16 23 00
GTO1      22 01
1            01
RCL9      36 09
-           -45
6            06
Yˣ           31
.           -62
4            04
4            04
RCL8      36 08
÷           -24
-           -45
2            02
.           -62
4            04
2            02
+           -55
RCL8      36 08
+           -55
2            02
Pi        16-24
x           -35
X⇄Y         -41
÷           -24
GTO2      22 02
*LBL1     21 01
8            08
RCL8      36 08
÷           -24
LSTX      16-63
4            04
÷           -24
+           -55
LN           32
*LBL2     21 02
6            06
0            00
x           -35
STOA      35 11
x           -35
STO1      35 01
X⇄Y         -41
PRTX        -14
X⇄Y         -41
PRTX        -14
RTN          24
*LBLD     21 14
√X           54
F1?    16 23 01
GTO1      22 01
*LBL4     21 04
RCL4      36 04
x           -35
RCL7      36 07
÷           -24
2            02
x           -35
Pi        16-24
x           -35
STO3      35 03
RTN          24
*LBL1     21 01
2            02
0            00
RCL0      36 00
÷           -24
RCL9      36 09
x           -35
RCLA      36 11
÷           -24
STO4      35 04
R↓          -31
GTO4      22 04
*LBLE     21 15
EEX         -23
9            09
÷           -24
STO5      35 05
√X           54
RCL3      36 03
x           -35
RCL2      36 02
÷           -24
STOB      35 12
PRTX        -14
LSTX      16-63
x           -35
2            02
x           -35
RCL4      36 04
÷           -24
RCL9      36 09
x           -35
PRTX        -14
2            02
Pi        16-24
x           -35
RCL0      36 00
÷           -24
RCL5      36 05
x           -35
3            03
÷           -24
RCL2      36 02
÷           -24
RCLB      36 12
÷           -24
PRTX        -14
RTN          24
*LBLC     21 13
CF1    16 22 01
RCL7      36 07
2            02
x           -35
X⇄Y         -41
÷           -24
RCL8      36 08
Pi        16-24
2            02
x           -35
x           -35
x           -35
X>Y?      16-34
X⇄Y         -41
LN           32
Pi        16-24
÷           -24
RCL8      36 08
+           -55
1            01
+           -55
F0?    16 23 00
GTO1      22 01
1            01
RCL9      36 09
-           -45
5            05
Yˣ           31
6            06
x           -35
.           -62
4            04
4            04
+           -55
RCL9      36 09
X²           53
x           -35
1            01
+           -55
x           -35
RCLA      36 11
x           -35
7            07
2            02
0            00
÷           -24
Pi        16-24
GTO0      22 00
*LBL1     21 01
3            03
2            02
RCL9      36 09
X²           53
x           -35
1            01
-           -45
x           -35
RCLA      36 11
÷           -24
LSTX      16-63
6            06
0            00
÷           -24
eˣ           33
÷           -24
.           -62
4            04
*LBL0     21 00
÷           -24
RCL0      36 00
÷           -24
Pi        16-24
÷           -24
STO4      35 04
RTN          24
```

MICROSTRIP CALCULATIONS
(HP-41 Translation)

By Ross Cooling #12433
R.R. #1
Kimberley, Ontario
Canada N0C 1G0

This is a translation of a HP-67 Electrical Engineering program called Microstrip Calculations (number 00033 in the HP-67 library).

I won't go into the mathematics, I'll just tell you how to use the program. The values are input by use of the A-E keys, so you should remove any key assignments for those keys. The results will be displayed at the end if flag 01 is clear, or printed if flag 01 is set. The minimum SIZE is 012, and the printer is optional. You are advised to set the mode to ENG 3.

There are two methods of using this program. The first is when you know the width of the microstrip (in cm.), the thickness of the dielectric (in cm.), and the relative permitivity and you want to find the relative phase velocity and the impedance of a lossless line. The second is when you know the thickness and resistivity of a conductor, and a frequency and you want to find the copper loss, resistance per unit length and the unloaded Q. For uniform current distribution the thickness isn't necessary.

These two examples should help to clarify things:
1) What are the characteristics of a 50-mil microstrip on a 50-mil alumina (ϵ_r=9.5) substrate at 2GHz? assume a line thickness of 1 mil and a conductor resistivity of 3 E-6.
2) Repeat 1), but assume a uniform current distribution.

The keystrokes are as follows:

```
1) 0.05, ENTER, 2.54, *, ENTER,  press A
   9.5,  press B to see  391.3 -03
                    and  49.54  00
   0.001, ENTER, 2.54, *,  press C
   3 -06,  press D
   2  09,  press E  to see  11.01 -03
                       and  242.4 -03
                       and  422.4  00

2) 0.05, ENTER, 2.54, *, ENTER,  press A
   9.5,  press B to see  391.3 -03
                    and  49.54  00
   3 -06,  press D
   2  09,  press E  to see  21.25 -03
                       and  242.4 -03
                       and  218.8  00
```

```
PRP "
01♦LBL "MSTRIP"
02♦LBL A
03 SF 01
04 STO 07
05 X<>Y
06 STO 06
07 SF 00
08 X>Y?
09 CF 00
10 /
11 STO 09
12 1/X
13 STO 08
14 10
15 LN
16 STO 00
17 RTN
18♦LBL B
19  E
20 X<>Y
21 *
22 LASTX
23  E
24 -
25 10
26 RCL 06
27 /
28  E
29 +
30 SQRT
31 /
32 +
33 2
34 /
35 SQRT
36 1/X
37 STO 02
38 FS? 00
39 GTO 01
40  E
41 RCL 09
42 -
43 6
44 Y↑X
45 .44
46 RCL 08
47 /
48 -
49 2.42
50 +
51 RCL 08
52 +
53 PI
54 ST+ X
55 X<>Y
56 /
57 GTO 02

58♦LBL 01
59 8
60 RCL 08
61 /
62 LASTX
63 4
64 /
65 +
66 LN
67♦LBL 02
68 60
69 *
70 STO 10
71 *
72 STO 01
73 X<>Y
74 FS? 21
75 PRX
76 FC? 21
77 PSE
78 X<>Y
79 FS? 21
80 PRX
81 RTN
82♦LBL D
83 SQRT
84 FC? 01
85 GTO 04
86 20
87 RCL 00
88 /
89 RCL 09
90 *
91 RCL 10
92 /
93 STO 04
94 RDN

95♦LBL 04
96 RCL 04
97 *
98 RCL 07
99 /
100 ST+ X
101 PI
102 *
103 STO 03
104 RTN
105♦LBL E
106  E9
107 /
108 STO 05
109 SQRT
110 RCL 03
111 *
112 RCL 02
113 /
114 STO 11
115 FS? 21
116 PRX
117 FC? 21
118 PSE
119 LASTX
120 *
121 ST+ X
122 RCL 04
123 /
124 RCL 09
125 *
126 FS? 21
127 PRX
128 FC? 21
129 PSE
130 PI
131 ST+ X
132 RCL 00
133 /
134 RCL 05
135 *
136 3
137 /
138 RCL 02
139 /
140 RCL 11
141 /
142 FS? 21
143 PRX
144 RTN
145♦LBL C
146 CF 01
147 RCL 07
148 ST+ X
149 X<>Y
150 /
151 RCL 08
152 PI
153 2
154 *
155 *
156 *
157 X>Y?
158 X<>Y
159 LN
160 PI
161 /
162 RCL 08
163 +
164  E
165 +
166 FS? 00
167 GTO 01
168  E
169 RCL 09
170 -
171 5
172 Y↑X
173 6
174 *
175 .44
176 +
177 RCL 09
178 X↑2
179 *
180  E
181 +
182 *
183 RCL 10
184 *
185 720
186 /
187 PI
188 GTO 00

189♦LBL 01
190 32
191 RCL 09
192 X↑2
193 *
194  E
195 -
196 *
197 RCL 10
198 /
199 LASTX
200 60
201 /
202 E↑X
203 /
204 .4

205♦LBL 00
206 /
207 RCL 00
208 /
209 PI
210 /
211 STO 04
212 END
```

MSTRIP
REGISTERS: 40

ROW 1 (1-3)

ROW 2 (3-13)

ROW 3 (14-24)

ROW 4 (25-36)

ROW 5 (37-45)

ROW 6 (46-54)

ROW 7 (55-66)

ROW 8 (67-76)

ROW 9 (76-84)

ROW 10 (85-95)

ROW 11 (96-106)

ROW 12 (106-116)

ROW 13 (117-126)

ROW 14 (127-136)

ROW 15 (137-146)

ROW 16 (146-157)

ROW 17 (158-168)

ROW 18 (169-179)

ROW 19 (180-189)

ROW 20 (190-200)

ROW 21 (201-212)

ROW 22 (212-212)

R/S

CORRESPONDENCE

Dear PPC,

The April Journal just arrived here - congratulations, I found it much more interesting and original than some recent issues. I hope it will stay this good or even get better - and in that hope I enclose my 1988 dues to beat the increase.

The idea of a Questions & Answers column is very good and Ross Cooling is doing a great job with it. Could I reply to some questions as well? Rick Wenger asked about other publications for programmable calculators and Ross said he does not know of any. This is a little surprising, since the PPC Journal has mentioned several European clubs and their journals recently - for example the February issue had an article in which I mentioned the British club and its journal DATAFILE. Look in some back issues Rick and you will find plenty about other clubs and journals. The April journal itself carries an invitation to the international conference being organised by PPC-Denmark and they publish an excellent journal. Do not be too worried about these journals being in foreign languages - at a meeting of European club leaders in 1985 we agreed that non-English journals would have an English summary of their contents, and in any case most programs are commented in English. There is an Australian journal too, and a new journal will soon be starting up in the U.S. Maybe Ross meant that he knows of no journals for users of non-HP handhelds - well the EduCALC catalog regularly gives the address of TI PPC who publish "TI PPC Notes".

Simon Hughes asks about 28C features and adding memory to it. DATAFILE has been publishing a regular HP-28C column since the beginning of the year and the June column describes how to open up the 28C, what is inside, how to add extra memory, and which components to change if you want to try speeding it up. We send a copy of every DATAFILE issue to PPC, so anyone visiting PPC does not even need to subscribe to our club to read DATAFILE.

Based on an original idea by Dijkstra,

Best Wishes,

Włodek Mier-Jędrzejowicz,8498
42,Heathfield Rd.,
London W3 8EJ,
England

BITS & PIECES

I am including my dues for another year.

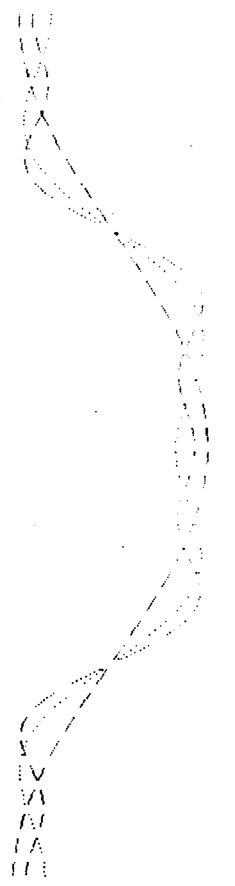
I am also including a plot and listing prompted by Joan Hampton's question on page 34 of V14N4 (Questions and Answers by Ross Cooling #12433). The problem was not so much logic, but a misunderstanding of how points are plotted by PPC ROM program, "HP." In this case the program calls label ONE 7 times, then label THREE four times, and then label FIVE 7 times to get the 18 dots printed by one sweep of the printer. Joan attempts to speed up the program by saving values from ONE and THREE which are used in

THREE and FIVE, but these values are incorrect except for the bottom part of each printed row.

Another problem is the use of Register 41 to store intermediate results. According to the write-up for "HP" in the PPC ROM User's Manual. Registers above #39 may be used if more than 16 dots are needed in each printer sweep. I believe, however, that R41 in this particular program is used to store the result of calling FIVE for the 7th time (the last time), so does not affect the results of the plot.

Tom Sears #11627
P.O. Box 421
Columbia, MO 65205-0421

```
 ONE
 THREE
 FIVE
Y: -0.200 TO 1.300
X: -3.142 TO 3.142
ΔX=0.016
```



```
12:47AM 05/31
 01♦LBL "ONE"        38 COS
 02 R-D              39 3
 03 COS              40 /
 04♦LBL 01           41 -
 05 ST+ X            42 X<>Y
 06 PI               43 COS
 07 /                44 +
 08 .5               45 GTO 01
 09 +                46♦LBL "MTPL"
 10 RTN              47 XROM "[illegible]F
 11♦LBL "THREE"      48 SF 21
 12 R-D              49 [illegible]
 13 ENTER↑           50 XROM "IF"
 14 ENTER↑           51 -.2
 15 3                52 STO 00
 16 *                53 1.3
 17 COS              54 STO 01
 18 3                55 168
 19 /                56 STO 02
 20 CHS              57 PI
 21 X<>Y             58 STO 09
 22 COS              59 CHS
 23 +                60 STO 08
 24 GTO 01           61 .016
 25♦LBL "FIVE"       62 STO 10
 26 R-D              63 .121
 27 ENTER↑           64 STO 12
 28 ENTER↑           65 "ONE"
 29 ENTER↑           66 ASTO 15
 30 5                67 "THREE"
 31 *                68 ASTO 16
 32 COS              69 "FIVE"
 33 5                70 ASTO 17
 34 /                71 SF 04
 35 X<>Y             72 3
 36 3                73 XROM "HP"
 37 *                74 END
```

I am enclosing my membership renewal and a couple of programs that might be found useful. Both are simple CCD-ROM utilities.

RAMED

May be a substitute for the ZENROM function of the same name (I have not had access to the ZENROM). With any legal address in X (the bytes above and below must also exist), XEQ **RAMED.** The display will show an address and, in hex, the byte requested and the ones immediately above and below. The 4 "toggle keys" function to move the "window" through memory either a register or a byte at a time. To edit the addressed byte, press the gold shift key, respond Y (or N), and at the prompt, key the replacement byte in hex. To exit, allow the program to time out, or press any other keys. To resume, merely R/S. Required are XF/M and CCD-ROM. The program runs in size

```
11:44PM 07/01
 01♦LBL "RAMED"      25 AVIEW            49♦LBL 02
 02 FIX 1            26 LASTX            50 2
 03 .                27 GETKEY           51 /
 04 WSIZE            28 X=0?             52 -
 05 UNS              29 GTO 00           53 X<0?
 06 CF 05            30 5                54 SF 05
 07♦LBL 01           31 X>Y?             55 ABS
 08 RDN              32 GTO 02           56 INT
 09♦LBL 03           33 CLX              57 7
 10 CLA              34 31               58 *
 11 ARCL X           35 -                59 X=0?
 12 "├  "            36 X≠0?             60 SIGN
 13 A+               37 GTO 00           61 FC?C 05
 14 PEEKB            38 "POKE:"          62 CHS
 15 ARCLH            39 ARCL Y           63 A+B
 16 LASTX            40 "├ YN"           64 GTO 03
 17 "├,"             41 CLX              65♦LBL 00
 18 PEEKB            42 PMTK             66 RDN
 19 ARCLH            43 DSE X            67 END
 20 LASTX            44 GTO 01                    PLNG "RAMED"
 21 "├,"             45 CLX                    122 BYTES
 22 A-               46 PMTH              Terrence Jay Cohen, M.D.
 23 PEEKB            47 POKEB                  PPC 10956
 24 ARCLH            48 GTO 01
```

0. The only synthetic is line 3 which can be replaced with 0 if preferred. Line 12 is append 2 spaces. Of course, the program does affect the status registers, unlike an M-Code ROM function.

DUMP

Provides a hex dump on a thermal printer beginning at the word including the address specified in X and continuing until stopped by R/S. For programs, use **PHD** to obtain this address in an appropriate form. The listing is 1 word per line in hex with the address in decimal and in decending address order to facilitate interpretation of program dumps. A printer and the CCD-ROM are required. Line 7 is an FO (or any other) NOP. Size is 0, but the status registers are, of course, changed by the program making status register dumps by this method of little utility.

```
11:50PM 07/01
 01♦LBL "DUMP"       09 CF 29            17 LASTX
 02 ABS              10♦LBL 01           18 DSE X
 03 ENTER↑           11 CLA              19 GTO 01
 04 INT              12 PEEKR            20 END
 05 X≠Y?             13 DCD                       PLNG "DUMP"
 06 ISG X            14 CLX                    37 BYTES
 07 ""               15 X<>Y              Terrence Jay Cohen, M.D.
 08 FIX 0            16 PRAXY                  PPC 10956
```

I have several more substantial programs that might be of interest including games and medical programs but lack time to adequately document them for submission. Perhaps, sometime in the future I will have an opportunity. If there are any PPC'ers in Palm Beach County, Florida, I would be interested in hearing from them.

Terrence Jay Cohen, M.D. #10956
3175 South Congress Avenue, Suite 105
Palm Springs, FL 33461

BENCHMARK SURPRISES

In the March '87 issue of "Sky & Telescope" magazine, the section "Astronomical Computing" includes the following benchmark for computers:

```
A=1 @ FOR I=1 TO 2499 @
A=TAN(ATN(EXP(LOG(SQR(A*A)))))+1 @ NEXT I
```

Running it in the HP-75, in one line as is shown here, and in RADIANS mode, it spends 5min 39sec, and gives a result of A= 2499.99942403, that is to say, only 0.000023% less than the exact value of A= 2500. It is better than the same test run in a computer with a Z80 processor, 4 MHz clock speed, CP/M 2.2 operating system and loaded with Turbo Pascal, that gives A= 2500.0046341 in 6min 41sec: an error of 0.00019% and a computation time 18% longer.

And our "first love", the HP-41? I have run in my CX the same benchmark: in RAD mode, the result of 2499,970322 (error -0.00119%) is reached in 49min 17sec, and in DEG mode the value is 2499,987092 (error -0.00052%) and the computation time 57min 39sec. Not quite excessive, because the computer tested in the cited article spends 38min 22sec to run the benchmark when loaded with the compiler version (5.2) of Microsoft Basic used in double precision (but giving a percentual error of about 10^-7).

It is unfortunate that the obsolescense of the HP-75 have left the computer market without machines of this type; the 71 is supported... yet, but it is not a direct equivalent. I assume the 71 would run the benchmark program more or less in the same time than the 75, but I cannot get one to do the experience; it is indeed a challenge for a four-bit machine!

Cristian Rusquellas (12044)
Miranda 4832
1407 Buenos Aires
Argentina

NOP

In V11N2p29d (Comments on the QRG) and V13N8p29b (Feedback) I stated that the IL Module functions WRTRX, WRTR, READRX and READR still had to be checked for normalization of data. However, recently I discovered that I had not done my homework very well. William C. Wickes (3735) did already check these functions back in 1982, soon after the Cassette Drive was released. And in V9N4p37a he reported that none of them normalize NNN's. I hope this clears up any confusion that I may have caused.

Frans de Vries (10993)
Abelenstraat 82
7556 DV Hengelo (o)
The Netherlands

Here is the copy of the program list for "AAZ", Equatorial to Horizon Coordinate Conversion, that I neglected to enclose with the program descrption. Sorry about that.

Just a note on the program. I mentioned that Jean Meeus' examples might give a slightly different answer than Peter Duffett-Smith's. There are two things about the coordinate measurement in his book to take note of.

Article "Equatorial to Horizon Coordinate Conversion" published in the May (V14N5P30) issue.

Meeus measures azimuth west from south, not east from north as my program does, so the answers will have a difference of 180 degrees. The answers will be correct, it is the reference point that is different. He also measures longitude as positive if west and negative if east. My program is set up with west as negative and east as positive. If you are using Meeus' book, you will have to convert the input as required. His azimuth answer will be west from south either positive or negative.

```
01*LBL "AAZ"
02*LBL 01
03 CF 21
04 CF 22
05 .
06 X<>F
07 "DATE D↑M↑Y"
08 PROMPT
09 FC?C 22
10 GTO 00
11 STO 03
12 RDN
13 STO 02
14 RDN
15 STO 01
16*LBL 00
17 "TIME (24 HR.)"
18 AVIEW
19 "T="
20 5.04
21 XEQ 06
22 PSE
23 "DST ?"
24 PROMPT
25 FS?C 22
26 SF 07
27 "R.A.="
28 11.07
29 XEQ 06
30 "DEC.="
31 12.08
32 XEQ 06
33 "LAT. (N+/S-)"
34 AVIEW
35 "LAT="
36 13.09
37 XEQ 06
38 "LONG. (E+/W-)"
39 AVIEW
40 "LNG="
41 14.1
42 XEQ 06
43 RCL 01
44 RCL 02
45 RCL 03
46 XEQ "JD"
47 STO 18
48 XEQ "GMT"
49 XEQ "GST"
50 XEQ "LST"
51 XEQ "H∠"
52 RCL 08
53 COS
54 RCL 09
55 COS
56 RCL 15
57 COS
58 *
59 *
60 RCL 08
61 SIN
62 RCL 09
63 SIN
64 *
65 +
66 ASIN
67 STO 00
68 HMS
69 RCL 08
70 SIN
71 RCL 09
72 SIN
73 RCL 00
74 SIN
75 *
76 -
77 RCL 09
78 COS
79 RCL 00
80 COS
81 *
82 /
83 ACOS
84 FS? 05
85 .
86 FC?C 05
87 -360
88 +
89 ABS
90 HMS
91 SF 21
92 "ALT="
93 ARCL Y
94 AVIEW
95 "AZ="
96 ARCL X
97 AVIEW
98 GTO 01
99*LBL "JD"
100 "x'♦♦♦←xrσP♦Γ"
101 "├←e%♦♦♦x̄x̄♦♦♦♦♦♦"
102 ST- ↑
103 RCL Z
104 ST- ↑
105 RCL Z
106 ST- ↑
107 RCL Z
108 X<> ↑
109 X>0?
110 SF 00
111 X<> ↑
112 CLX
113 3
114 X<=Y?
115 GTO 00
116 CLX
117 12
118 ST+ Y
119 DSE T
120*LBL 00
121 X<> T
122 ST* \
123  E2
124 /
125 INT
126 ST- [
127 4
128 /
129 INT
130 ST+ [
131 X<> [
132 FS?C 00
133 ST- X
134 30.6001
135 ST* Z
136 +
137 +
138 INT
139 ST+ ]
140 X<> \
141 INT
142 RCL ]
143 +
144 +
145 RTN
146*LBL "GMT"
147 RCL 04
148 FC? 07
149 GTO 00
150 1
151 -
152*LBL 00
153 RCL 10
154 15
155 /
156 INT
157 -
158 XEQ A
159 STO 20
160 RTN
161*LBL "GST"
162 CLST
163 1
164 RCL 03
165 XEQ "JD"
166 STO 00
167 "x̄0♦♦♦♦♦xx̄0♦Q$ ←"
168 2415020
169 -
170 36525
171 /
172 ST* [
173 X↑2
174 2.581 E-5
175 *
176 ST+ [
177 X<> [
178 6.6460656
179 +
180 RCL \
181 RCL 03
182 1900
183 -
184 *
185 -
186 ST- \
```

```
187 X<> \
188 STO 19
189 RCL 01
190 INT
191 RCL 02
192 RCL 03
193 XEQ "JD"
194 RCL 00
195 -
196 6.57098 E-2
197 *
198 RCL 19
199 -
200 RCL 20
201 1.002738
202 *
203 +
204 XEQ A
205 STO 17
206 RTN
207♦LBL "LST"
208 RCL 17
209 RCL 10
210 15
211 /
212 +
213 XEQ A
214 STO 16
215 RTN
216♦LBL "HA"
217 RCL 16
218 RCL 07
219 -
220 XEQ A
221 15
222 *
223 STO 15
224 SIN
225 X<0?
226 SF 05
227 RTN
228♦LBL 06
229 ASTO 00
230 INT
231 ST- L
232 LASTX
233  E2
234 *
235 ARCL IND Y
236 PROMPT
237 FC?C 22
238 RTN
239 CLA
240 ARCL 00
241 ARCL X
242 AVIEW
243 STO IND Z
244 HR
245 STO IND Y
246 RTN
247♦LBL A
248 24
249 X<>Y
250 X>Y?
251 -
252 X<0?
253 +
254 ABS
255 RTN
256 .END.
```

Duane Smith (10552)
900 E. Karen #C-203
Las Vegas, Nevada 89109

There is a mistake in the listing of my program "Plot with the ThinkJet" (V14 N5 P29), owing to that certain characters are printed only in Display Functions mode; the expression of the string R$ at line 210 must be:

R$=FILL$(CHR$(0),'','',148)

The program longitude is increased in two bytes.

The promised execution times, not included, are: for 'FUNC', 9min 58sec, and for 'EXPON', 3min 55sec.

The legend at the foot of the fig. at page 30 corresponds to an example not sent; it is a mistery for me its presence there.

Cristian Rusquellas (12044)
Miranda 4832
1407 Buenos Aires
Argentina

In the fifth paragraph of his article Labeless Program Generation via Barcode in V14N4p8c, Philip Frohne (9660) says that labeless programming results in a reduction of two to three bytes per GTO or XEQ. This is not correct, as labels are the instructions that are deleted, not the two or three bytes GTO's or three bytes XEQ's. The reduction is of course one to two bytes per LBL. Don't forget that you can't delete labels that could be called by GTO IND or XEQ IND instructions, because these instructions cannot be compiled and always perform a label search.

Philip also states (in the second but last paragraph) that programs with compiled GTO's/XEQ's and no labels can not be recorded on cards. This again is not true. You can record these programs on cards like any other program, and if you use my programs given in V12N5p17b, then you can load them and append a compiled END without any problem.

Instead of using Philip's method you can also use the method published by Dejan Ristanović (9799) in his article Building Labelless Programs in V11N1p32a (for which you need my program "NRX" in V12N1p28b), or better yet use the function CMPDL (CoMPile and Delete Labels) from the Operating System EPROM that ERAMCO Systems supplies with their MLDL's.

\- - -

In the U2/L8 EPROM Programmer Loading Utility in V14N4p18a the instructions on addresses P096 through P0A9 are missing. Could the author or the Editor supply these as yet?

Frans de Vries (10993)
Abelenstraat 82
7556 DV Hengelo (o)
The Netherlands

HP-28C BYTE-COUNT ROUTINE

By Charles W. Jess #12782
2645 Hartford Street
San Diego, CA 92110

HP-28C BYTE TABLE
(Rev. July, 1987)

Byte Cost of this when used in this → CONTEXT (a)

OBJECT	Variable (b)	« » Prgm Del.	{ } List	' ' Alg. Del.	Arrays	Loc. Var. (c)
« » Prgm Delimiter	10	12.5	10			5
{ } List	5	5	5			
' ' Algebraic Del.	5	5	5			5
[ ] Vector	12.5	12.5	12.5			
[[ ]] Matrix	15	15	15			
" " String, Empty	5	2.5	2.5			
" " String, w/char(a)	5	5	5			
Real Numbers				(d)		
Single Digit	10.5	2.5	2.5	2.5	8	
2 or more Digits	10.5	10.5	10.5	10.5	8	
Complex Numbers (e)	18.5	18.5	18.5	18.5	16	
Binary Integers (e)	13	13	13			
Names (a)(f) ' '		5				
Single Capital (g)	4.5	2.5	2.5	2.5		
Single L.case ?Σμ°	4.5	4.5	4.5	4.5		
2 Characters	5.5	5.5	5.5	5.5		
Add'l. Chars., ea.	1	1	1	1		
Catalog						
ELSE		7.5				
WHILE...REPEAT		10				
IF or IFERR...THEN		15				
Symbolic Constant	7.5	2.5	2.5	2.5		
Other		2.5	2.5	2.5		

Footnotes

a Two delimiters do not have columns. In strings, characters are 1 byte each. Since name delimiters, stored only in programs, do not affect contents, use name rows and program column to get byte values.

b Storage in a 1-character variable is 5.5 bytes. Add 1 for each additional character.

c For values between delimiters in → ...'...' or → ...«...» use applicable delimiter column.

d In algebraics, add 2.5 bytes for negative mantissas.

e Entities include numbers.

f Page 44 of 28C Reference Manual says first character of a name must be a letter; however, at least seven other characters seem to work in this position. ?Σμ° can be used exactly as Alpha. →π % must be in names of at least two characters. Also, except when being entered into a program, a name/algebraic delimiter must precede %.

g Single capitals require 4.5 bytes when one of the following:
1. Stored as a value in a variable. (Subsequent entries not affected.)
2. Used in local variable structure. (Entries outside structure, even in same program, not affected.)
3. Entered into an object at same time capital is on USER menu. (All entries are affected. Subsequent change of menu does not affect created value.)

Byte values of most objects are determined by the immediate context in which they are used: closest pair of delimiters, local variable structure, program branching structure, etc. The following program illustrates how the table may be used.

```
'→Σ'
« « → A 'A+1' » { "XYZ" (123,456) } # 789 A 'ΣK' »
```

Object	Context	Bytes	Remarks
« »	Variable	10	
« »	« »	12.5	
→	« »	2.5	Catalog
A	Loc Variable	9	First 2 A's, 4.5 ea., Footnotes c, g2
' ' Alg	Loc Variable	5	
+	Algebraic	2.5	Catalog
1	Algebraic	2.5	
{ }	« »	5	
"XYZ"	{ }	8	String plus 3 chars, Footnote a
(123,456)	{ }	18.5	Complex Number entity
# 789	« »	13	Binary Number entity
A	« »	2.5	Unquoted, single-capital name
'ΣK'	« »	10.5	Quoted, 2-character name (5+5.5)
		101.5	Total
		6.5	Storage in 2-character variable, Footnote b

This simple, but potent, routine takes most of the work out of counting bytes required by an object stored in a variable or used in another object. It can verify and add to the contents of the byte table. 'BC' displays the number of bytes, in half-byte units of whatever is placed in level 1, ranging from zero to programs containing hundreds of bytes, like the Base-to-Base Conversion routine 'BB'* in last month's article. Remarkably, USER memory and the modes of operation remain intact and the stack need not be cleared.

```
'BC'  Bytes:  66.5 (60 + 6.5 storage)
« -5 -6 + MEM + OVER DUP J STO K STO MEM - 2 /
{ J K } PURGE »
USER Memory:  J and K not on menu.
```

Basically, the object in level 1 is stored in two 1-character variables, MEM readings are taken before and after, and the difference between them halved. To insure that the result yields only the byte cost of the objects in level 1, the MEM readings are taken with identical byte loads on the stack and the cost of storing the object in the variables is deducted before halving the difference. The first MEM is taken with the object and a number, -11, on the stack. The 11-byte cost for storing is deducted and the object stored in the variables. Next, the object and a different number, the first MEM reading, are on the stack for the second reading. 'J' and 'K' are purged to permit immediate re-use of 'BC', to minimize the between-use byte drain, and to maintain an uncluttered USER menu. The final display contains the object in level 2 and the result in level 1.

Using 'BC'.

1. Insure J and K are not on USER menu.

2. Put object, in context, in level 1. Scan for accuracy.

3. Press BC.

4. Deduct byte value for context, if other than a variable.

An object stored in a variable. Put an object from a row of the byte table into level 1 and press BC. Algebraic delimiters and arrays will not be accepted empty; however, they may be checked easily by adding contents of known byte values and deducting those values from the result. For example, '1+1' could be used for the former, deducting 7.5 for the contents. In like manner, 1 may be used in a vector, deducting 8 and 1's may be used in each row of a matrix, deducting 8 for each one used.

An object used in another object. With an object from a row of the table in a context other than a variable, enter the whole into level 1. After pressing BC, deduct the byte cost for the context (the cost for storing it in a variable) from the result. For example, to get the byte cost of an empty string to be used in a list within a program, put { "" } into level 1, press BC, deduct 5. Alternatively, to get the cost of both list and string simultaneously, put « { "" } » in level 1 and deduct 10 from the result. To get the byte cost of 'BC', just recall it to level 1 and press BC.

If you think Footnote g3 of the byte table smacks a little of voodoo (and who's to blame you?), test it quickly using a simple algebraic. Key in 'A+B', or any two capitals not on the menu, and press BC. Leave the display on the stack. Store anything in variables 'A' and 'B' to get them on the menu. Key in a fresh 'A+B' and press BC. To verify the last sentence of the footnote, drop the first 'A+B' into level 1, press BC, and note the values have not changed, despite the menu addition.

Checking local variable structures and some catalog values is a bit more challenging. Some require rigid word sequences, but all should yield if the unknowns are minimized and isolated to be tried in different formats. You may very well come up with different values because of different approaches. After all, it would indeed be surprising if the byte table is complete and nothing is set in concrete.

*'BB' can be bobbed by five bytes: before SUB, substitute MEM for OVER SIZE; before END, substitute NOT for Ø SAME.

R/S

Program Description I

Program Title TRANSMITTER POWER, FIELD STRENGTH, AND DISTANCE INTERCHANGEABLE SOLUTIONS

Contributor's Name DONALD BEATY

Address 2130 QUEEN'S LANE

City SAN MATEO **State** CALIFORNIA **Zip Code** 94402

Program Description, Equations, Variables THE FIELD STRENGTH DEVELOPED BY A TRANSMITTER'S ANTENNA MAY BE CONSIDERED TO VARY INVERSELY AS THE ANTENNA'S DISTANCE TO A SAMPLING POSITION, AND IS PROPORTIONAL TO THE SQUARE ROOT OF THE POWER RADIATED BY THE ANTENNA. THIS PROGRAM USES THE INTERCHANGEABLE-SOLUTION FORMAT TO PERMIT CALCULATION OF ANY VARIATION OF THIS RELATIONSHIP. RELATED VARIABLE PAIRS MUST BE INPUT IN CONSISTENT UNITS e.g., NEW AND OLD FIELD STRENGTH IN $\mu V/m$ OR mV/m, OLD AND NEW POWER IN WATTS OR KILOWATTS, AND OLD AND NEW DISTANCE IN MILES OR METRES.

TO USE THIS PROGRAM, KEY IN THE FIVE KNOWN VALUES IN ANY ORDER, FOLLOWING EACH INPUT BY SELECTION OF ITS CORRESPONDING USER-DEFINEABLE KEY. THE SIXTH VALUE IS THEN DISPLAYED FOLLOWING DEPRESSION OF THE REMAINING USER-DEFINEABLE KEY WITHOUT INTERVENING DATA ENTRY.

EQUATIONS USED ARE:

$$f_N = f_o \frac{D_o}{D_N}\sqrt{\frac{P_N}{P_o}} \qquad P_o = P_N\left(\frac{D_o}{D_N}\frac{f_o}{f_N}\right)^2$$

$$f_o = f_N \frac{D_N}{D_o}\sqrt{\frac{P_o}{P_N}} \qquad P_N = P_o\left(\frac{D_N}{D_o}\frac{f_N}{f_o}\right)^2$$

$$D_o = D_N \frac{f_N}{f_o}\sqrt{\frac{P_o}{P_N}} \qquad D_N = D_o \frac{f_o}{f_N}\sqrt{\frac{P_N}{P_o}}$$

f_o = OLD FIELD STRENGTH
f_N = NEW FIELD STRENGTH
P_o = OLD POWER
P_N = NEW POWER
D_o = OLD DISTANCE
D_N = NEW DISTANCE

Operating Limits and Warnings

Program Description II

Sample Problem(s)

1) A 5-kW TRANSMITTER DEVELOPS A FIELD STRENGTH OF 150 mV/m AT A DISTANCE OF 3.7 MILES FROM THE ANTENNA. IF THE TRANSMITTER POWER IS CHANGED TO 1-kW, WHAT WILL BE THE NEW FIELD STRENGTH AT A DISTANCE OF 12 MILES?

2) SUPPOSE THE OLD POWER OF THE TRANSMITTER OF PROBLEM #1 WAS 10KW AND THE NEW POWER IS 50 kW. ALL OTHER PARAMETERS HAVE THE ORIGINAL VALUES AFTER RUNNING PROBLEM #1, RE-RUN THE PROBLEM WITH THE NEW VALUES, CALCULATING THE NEW FIELD STRENGTH.

3) A 25-kW TRANSMITTER DEVELOPS A FIELD STRENGTH OF 1250 μV/m AT A DISTANCE OF 14.7 KILOMETRES. IF THE TRANSMITTER CHANGES POWER TO 100-KW, WHAT WILL BE THE DISTANCE TO THE 2500 μV/m CONTOUR?

4) AFTER RUNNING PROBLEM #3, YOU WISH TO DETERMINE THE POWER NEEDED TO DEVELOP 6700 μV/m AT 14.7 KILOMETRES. CALCULATE THE NEW POWER.

Solution(s)

1) 5 [C] 150 [E] 3.7 [A] 1 [B] 12 [f A] [D] → 20.68 mV/m
2) 10 [C] 50 [B] [D] → 103.42 mV/m
3) 25 [C] 1250 [E] 14.7 [A] 100 [B] 2500 [D] [f A] → 14.70 KM
4) 6700 [D] [B] → 718.24 kW

User Instructions

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS		OUTPUT DATA/UNITS
1	LOAD SIDE 1				
2	INPUT FIVE OF THE FOLLOWING:				
	NEW DISTANCE	D_N	f	A	
	OLD DISTANCE	D_o	A		
	NEW POWER	P_N	B		
	OLD POWER	P_o	C		
	NEW FIELD STRENGTH	NEW f.s.	D		
	OLD FIELD STRENGTH	OLD f.s.	E		
3	CALCULATE REMAINING VALUE BY SELECTING				ANSWER
	THE SIXTH USER-DEFINEABLE KEY				
4	OPTIONAL:				
	TO CHANGE ANY OR ALL OF THE ABOVE				
	INPUTS, KEY IN THE NEW VALUES FOLLOWED				
	BY THE CORRESPONDING USER-DEFINEABLE KEYS				
5	FOR A NEW CASE, RETURN TO STEP 2				

02050D

Program Listing I

STEP	KEY ENTRY	KEY CODE	COMMENTS
001	g LBL f A	32 25 11	TESTS FOR KEY BOARD ENTRY OF NEW DISTANCE
	STO Ø	33 00	
	h F?3	35 71 03	
	h RTN	35 22	
	RCL 1	34 01	CALCULATES NEW DISTANCE
	RCL 5	34 05	
	x	71	
	RCL 4	34 04	
	÷	81	
010	RCL 2	34 02	
	RCL 3	34 03	
	÷	81	
	f √x	31 54	
	x	71	
	STO Ø	33 00	
	h RTN	35 22	
	f LBL A	31 25 11	TESTS FOR KEYBOARD ENTRY OF OLD DISTANCE
	STO 1	33 01	
	h F?3	35 71 03	
020	h RTN	35 22	
	RCL Ø	34 00	CALCULATES OLD DISTANCE
	RCL 4	34 04	
	x	71	
	RCL 5	34 05	
	÷	81	
	RCL 3	34 03	
	RCL 2	34 02	
	÷	81	
	f √x	31 54	
030	x	71	
	STO 1	33 01	
	h RTN	35 22	
	f LBL B	31 25 12	TESTS FOR KEYBOARD ENTRY OF NEW POWER
	STO 2	33 02	
	h F?3	35 71 03	
	35 22	35 22	
	RCL Ø	34 00	CALCULATES NEW POWER
	RCL 1	34 01	
	÷	81	
040	RCL 4	34 04	
	RCL 5	34 05	
	÷	81	
	x	71	
	g x²	32 54	
	RCL 3	34 03	
	x	71	
	STO 2	33 02	
	h RTN	35 22	
	f LBL C	31 25 13	TESTS FOR KEY BOARD ENTRY OF OLD POWER
050	STO 3	33 03	
	h F?3	35 71 03	
	h RTN	35 22	
	RCL 2	34 02	CALCULATES OLD POWER
	RCL 1	34 01	
	RCL 5	34 05	
	x	71	

STEP	KEY ENTRY	KEY CODE	COMMENTS
	RCL Ø	34 00	
	RCL 4	34 04	
	x	71	
060	÷	81	
	g x²	32 54	
	x	71	
	STO 3	33 03	
	h RTN	35 22	
	f LBL D	31 25 14	TESTS FOR KEYBOARD ENTRY OF NEW FIELD STRENGTH
	STO 4	33 04	
	h F?3	35 71 03	
	h RTN	35 22	
	RCL 5	34 05	CALCULATES NEW FIELD STRENGTH
070	RCL 2	34 02	
	RCL 3	34 03	
	÷	81	
	f √x	31 54	
	RCL 1	34 01	
	x	71	
	RCL 0	34 00	
	÷	81	
	x	71	
	RCL 4	33 04	
080	h RTN	35 22	
	f LBL E	31 25 15	TESTS FOR KEYBOARD ENTRY OF OLD FIELD STRENGTH
	STO 5	33 05	
	h F?3	35 71 03	
	h RTN	35 22	
	RCL 4	34 04	CALCULATES OLD FIELD STRENGTH
	RCL 3	34 03	
	RCL 2	34 02	
	÷	81	
	f √x	31 54	
090	x	71	
	RCL Ø	34 00	
	RCL 1	34 01	
	÷	81	
	x	71	
	STO 5	33 05	
	h RTN	35 22	
100			
110			

REGISTERS

0	1	2	3	4	5	6	7	8	9
NEW DISTANCE	OLD DISTANCE	NEW POWER	OLD POWER	NEW FIELD STRENGTH	OLD FIELD STRENGTH				
S0	S1	S2	S3	S4	S5	S6	S7	S8	S9

Program Listing II

LABELS					FLAGS	SET STATUS
A OLD DISTANCE	B NEW POWER	C OLD POWER	D NEW FIELD STRENGTH	E OLD FIELD STRENGTH	0	FLAGS / TRIG / DISP
a NEW DISTANCE	b	c	d	e	1	FLAGS ON OFF: 0 ☐ ☒; 1 ☐ ☒; 2 ☐ ☒; 3 ☒ ☒ — TRIG: DEG ☒ GRAD ☐ RAD ☐ — DISP: FIX ☒ SCI ☐ ENG ☐ n 2
0	1	2	3	4	2	
5	6	7	8	9	3	

TRANSMITTER POWER, FIELD STRENGTH, AND DISTANCE INTERCHANGEBALE SOLUTIONS HP-41 TRANSLATION

By Ross Cooling #12433
R.R. #1
Kimberely, Ontario
Canada NOC 1GO

This is a translation of a HP-67 program called Transmitter Power, Field Strength, and Distance Interchangeable Solutions (number 02050 in the HP-67 library).

The field strength developed by a transmitter's antenna may be considered to vary inversely as the antenna's distance to a sampling position, and is proportional to the square root of the power radiated by the antenna. This program uses the interchangeable-solution format to permit calculation of any variation of this relationship. Related variable pairs must be input in consistent units ie. new and old field strength in µV/m or mV/m, old and new power in watts or kilowatts, and old or new distance in miles or metres.

To use this program, key in the five known values in any order, following each input by selection of its corresponding user-definable key. The sixth value is then displayed following depression of the remaining user-definable key, without intervening data entry.

The equations used are:

The keys are set up as follows:
To enter D_N (new distance) press a (shift A)
To enter D_o (old distance) press A
To enter P_N (new power) press B
To enter P_o (old power) press C
To enter f_N (new field strength) press D
To enter f_o (old field strength) press E

Here are some sample problems:
1) A 5 kW transmitter develops a field strength of 150 mV/m at a distance of 3.7 miles from the antenna. If the transmitter power is changed to 1 kW what will be the new field strength at a distance of 12 miles?
2) Suppose the old power of the transmitter of problem 1) was 10 kW and the new power is 50 kW. All other parameters have the original values. What would the new field strength be?
3) A 25 kW transmitter develops a field strength of 1250 µV/m at a distance of 14.7 kilometres. If the transmitter changes power to 100 kW what will be the distance to the 2500 µV/m contour?
4) After running problem 3) determine the power needed to develop 6700 µV/m at 14.7 kilometres.

The solutions are:
1) 5, press C,
150, press E,
3.7, press A,
1, press B,
12, press a
press D for the answer 20.68 mV/m

$$f_N = f_o \frac{D_o}{D_N}\sqrt{\frac{P_N}{P_o}} \qquad P_o = P_N\left(\frac{D_o}{D_N}\frac{f_o}{f_N}\right)^2$$

$$f_o = f_N \frac{D_N}{D_o}\sqrt{\frac{P_o}{P_N}} \qquad P_N = P_o\left(\frac{D_N}{D_o}\frac{f_N}{f_o}\right)^2$$

$$D_o = D_N \frac{f_N}{f_o}\sqrt{\frac{P_o}{P_N}} \qquad D_N = D_o \frac{f_o}{f_N}\sqrt{\frac{P_N}{P_o}}$$

f_o = OLD FIELD STRENGTH
f_N = NEW FIELD STRENGTH
P_o = OLD POWER
P_N = NEW POWER
D_o = OLD DISTANCE
D_N = NEW DISTANCE

The minimum SIZE is 006. No printer is used. Keys A-E and a are used, so any key assignments must be removed from these keys.

2) 10, press C,
50, press B
press D for the answer 103.42 mV/m

3) 25, press C,
   1250, press E,
   14.7, press A,
   100, press B,
   2500, press D
   press a for the answer 14.70 kM
4) 6700, press D
   press B for the answer 718.24 kW

Simple, right! Just enter the values you want, the other values remain as they were, and press the key for the new result without entering a value.

```
PRP ""

01♦LBL "TRANSP"
02♦LBL a
03 STO 00
04 FS?C 22
05 RTN
06 RCL 01
07 RCL 05
08 *
09 RCL 04
10 /
11 RCL 02
12 RCL 03
13 /
14 SQRT
15 *
16 STO 00
17 RTN

18♦LBL A
19 STO 01
20 FS?C 22
21 RTN
22 RCL 00
23 RCL 04
24 *
25 RCL 05
26 /
27 RCL 03
28 RCL 02
29 /
39 SQRT
31 *
32 STO 01
33 RTN

34♦LBL B
35 STO 02
36 FS?C 67
37 *
38 STO 05
39 -
40 LN1+X
41 9
42 RCL 01
43 /
44 *
45 X↑2
46 RCL 03
47 *
48 STO 02
49 RTN

59♦LBL C
51 STO 03
52 FS?C 22
53 RTN
54 RCL 02
55 RCL 01
56 RCL 05
57 *
58 RCL 00
59 RCL 04
60 *
61 /
62 X↑2
63 *
64 STO 03
65 RTN

66♦LBL D
67 STO 04
68 FS?C 22
69 RTN
70 RCL 05
71 RCL 02
72 RCL 03
73 /
74 SQRT
75 RCL 01
76 *
77 RCL 00
78 /
79 *
80 STO 04
81 RTN

82♦LBL E
83 STO 05
84 FS?C 22
85 RTN
86 RCL 04
87 RCL 03
88 RCL 02
89 /
90 SQRT
91 *
92 RCL 00
93 RCL 01
94 /
95 *
96 STO 05
97 END
```

TRANSP
REGISTERS: 18
ROW 1 (1-3)

ROW 2 (4-15)

ROW 3 (16-26)

ROW 4 (27-37)

ROW 5 (38-50)

ROW 6 (50-61)

ROW 7 (62-72)

ROW 8 (73-84)

ROW 9 (84-96)

ROW 10 (97-97)

R/S

HP-28C COLUMN

By Charles W. Jess #12762
2645 Hartford Street
San Diego, CA 92110

HP-28C BYTE COUNTING

Once we've figured out how to open the HP-28C, as Simon Hughes (2508, V14N4P48) implies, our first concern, as would-be programmers, is how to make the most of its limited number of bytes. This is an attempt at getting a handle on the problem. It may not be the answer and it's certainly not the complete answer; however, it does provide something to start with, pick apart, and build upon.

In securing an accurate byte count, many obstacles had to be overcome. The four major ones were: bytes could slip into unmeasurable reservoirs (COMMAND, LAST, and UNDO); the byte value often changes with the context in which it's used (for example, in 'A+B' and 'CAA' the A can have four different values); the byte value of the same object used in the same manner often fluctuates each time it is used (for example, it might alternate between two and three); and the only mechanism for showing available bytes doesn't seem to give the complete story. The first obstacle was overcome by going to MEMORY LOST, setting Mode to -CMD, -LAST, and -UND, and keeping the stack clear. (MEM should now show 1693, the maximum reading possible as far as I can determine.) The others were gradually bypassed by limiting the measurements, at least initially, to those of programs assigned to a variable. This is the count likely to be of most interest to programmers. By "weighing" an empty program, then interchanging internal objects, a pattern very slowly began to emerge. Finally, the apparent fluctuation of values forced the reluctant recognition that one has to work with half-byte values even though MEM provides readings in whole bytes only. The following list is predicated on that assumption. Following it are some examples, drawing mainly upon the program assigned to 'BB' in this issue.

The following short program stored in variable '?' and the program stored in 'BB', also in this issue, are used below to illustrate application of the byte-table values ('?' is explained

BYTE TABLE
FOR HP-28C PROGRAMS

Catalog	
ELSE	7.5
WHILE...REPEAT, combined	10
IF or IFERR...THEN, combined	15
Others	2.5
Numbers	
In « » or LIST, One digit, + or -	2.5
More than one digit	10.5
Stored directly in a variable	10.5
(See Vectors, Complex and Binary Numbers)	
Characters	
Names, First character	2.5
Second	3
Additional, each	1
In String, each	1
In Algebraics, each	2.5
As Local Variables, each usage, 1 char.	4.5
Add'l., ea.	1
Stored directly in a variable, 1 char.	4.5
Add'l., ea.	1
Program Delimiters, « »	
Outermost or in LIST	10
In Program Delimiter, but not in LIST	12.5
After →	5
Name Delimiters, ' '	
In Program Delimiter	5
In LIST or stored directly in a variable	Ø
Algebraic Delimiters, ' '	5
Alpha Strings, " "	
In « » or LIST, empty	2.5
with characters	5
Stored directly in a variable	5
Vectors, [ ]	12.5
Numbers, each	8
Complex Numbers, (123,456), numbers included	18.5
Binary Numbers, # 12345678, numbers included	13
Storage in a Variable	
Variable of one character	5.5
Additional characters, each	1

further in the last paragraph.):

'?' « 5 → a « 1 SWAP FOR b b a + + DUP NEXT DROP» »

Since most objects require 2.5 bytes, it seems easiest to list the others first, then include the 2.5-byte objects in an "all other" category.

'?'		'BB'	
« », outer	10	« », outer	10
« », after →	5	58 x 2.5	145
Local Variables		Storage in 'BB'	
a, 2 x 4.5	9	5.5 + 1	6.5
b, 2 x 4.5	9		161.5
10 x 2.5	25		
Storage in '?'	5.5		
	63.5		

Counting bytes for program segments can be illustrated using the HALT and N_{10} inserts for 'BB':

HALT	2 x 2.5	5	Or combined	
N_{10}	3 x 2.5	7.5	5 x 2.5	12.5

Note that since 'BB' has 161.5 bytes, adding the N_{10} segment or the combination, both of which contain fractional bytes, will result in a MEM display showing an 8- or 13-byte increase. The inability of MEM to cope with the 2-byte - 3-byte fluctuation mentioned in paragraph two is easily demonstrated. The program stored in 'BB', sans segments, leaves the available bytes at 1532 (see the 'BB' writeup). If one HALT is added, MEM will declare a decrease of 3 available bytes. When the second one is added, however, the reading drops by only 2 bytes to 1527.

Since byte counts involve half-byte units, there is a simple way to bypass the whole-byte limitation of MEM. Starting -CMD, -LAST, -UND, and a clear stack, of course, get a MEM reading. After poking in the program or segment, store it in two different variables of equivalent length (say, 'X' and 'Y' or 'BB' and 'CC'), subtract the new MEM reading, and divide the result by two.

The program assigned to variable '?' was designed not only to illustrate byte counting, but also to offer an example of the use of local variables with → and FOR, parts of the manual which seemed to me to be especially confusing. What the program does is trivial and it could be accomplished more simply and economically. Starting with a number N in level 2 and a number X in level 1, it puts N, N1, N2,...NX through the FOR...NEXT loop X times, yielding a display of

```
N1  (or N + 1 + 5)
N2  (or N1 + 2 + 5)
N3  (or N2 + 3 + 5)
...
NX  (or N(X-1) + X + 5)
```

Local variables a and b are equal in byte value, but they differ in content. Five is stored in a as a constant. One, the starting value for the loop, is stored in b, but it is augmented by one with each repetition of the loop. The first appearance of each local variable is merely for storage purposes; subsequent appearances put the variable's content on the stack. a is valid only within the inner program delimiters and b only between FOR and NEXT. FOR uses 1 and the user-supplied X as its start-finish control. DUP insures that a copy of the result of each passage through the loop remains on the stack. DROP (which could have been placed alternatively between the closing program delimiters) removes the last, and hence redundant, DUP.

HP-28C BASE-TO-BASE CONVERSION

This version of BB (V13N5P20) provides a comparison between the HP-41 and the HP-28C and helps to illustrate HP-28C byte counting also covered in this issue.

'BB' is designed to convert an Alpha string representing a number in one base (N_a) to its equivalent in another (N_b). (Note: the 28C already has rapid built-in conversion among HEX, Decimal, Octal, and Binary bases.) Sample beginning and ending displays for converting a string from base 18 to base 25 are:

Level	Input	End Display	Comment
4		"1A8"	N_a
3	"1A8"	18	Base a
2	18	25	Base b
1	25	"KC"	N_b

'BB' Bytes: 161.5

```
« Ø 4 PICK 1 OVER SIZE START DUP NUM 7 SQ - DUP 8 >
6 -1 IFTE - ROT 5 PICK * + SWAP 2 OVER SIZE SUB
NEXT SWAP DO 3 PICK / LAST MOD DUP 9 > 6 -1 IFTE +
7 SQ + CHR ROT + SWAP IP UNTIL DUP Ø SAME END
DROP »     Mode: +LAST required.
```

The decimal equivalent of N is an intermediate product. It can be preserved in the final display by inserting DUP 6 ROLLD before DO, adding 7.5 bytes. To view level 5, use DROP, VIEW↑, or 5 PICK.

Before START. The input is maintained intact on the stack throughout, a bonus of the 28C's "unlimited" stack. Zero starts the accumulation of N_{10}. N_a is duplicated and start-finish control established by 1 OVER SIZE.

START...NEXT loop. Converts N_a to N_{10} and empties another N_a duplicate. NUM converts each character to its decimal equivalent but, unlike ATOX, the character must be removed later by 2 OVER SIZE SUB. 7 through IFTE - reduce the decimal to its base value. 5 through + compute its positional value and accumulate N_{10}.

DO...UNTIL...END loop. Converts N_{10} to N_b while reconstituting the string emptied by the previous loop. The 3 PICK / LAST MOD...IP operations function like PPC ROM QR. 9 through SQ + compute the decimal equivalent of the new character and CHR ROT + appends the character to the string. When N_{10} has been reduced to zero, it is dropped, moving the final display into place.

HP recommends breaking a lengthy program into several smaller ones for ease of handling; however, it seems that inserting several HALT's at strategic points when poking in the program and using them to test the stack at those points works just as well at considerable byte savings. In 'BB', HALT's can be inserted advantageously before START and after NEXT, temporarily adding five bytes. When using the sample input from paragraph two, the display at the first HALT (in sequence level 1, level 2, etc.) should be 3 1 "1A8" 25 18 "1A8" and at the second "" 512 25 18 "1A8". Later, the HALT's can be deleted easily using VISIT.

To get the most from both this and the byte-counting writeup, the two can be worked together. Start with MEMORY LOST, Mode set at -CMD, -LAST, and -UND, and with the stack clear. MEM will then yield the following available byte readings before and after entering 'BB':

1693	Before entry.
1532	After the program has been stored in 'BB'.
1527	If the HALT's have been added.
1524	If the N_{10} segment alone has been added.
1519	If both segments have been added.

(Remember to use -LAST for byte testing, but +LAST for 'BB' execution.)

This program assigned to variable 'BB' consumes more than twice as many bytes as the referenced BB's 70. Considerable effort has gone into converting to HP-28C use; however, no program is ever complete and the machine's flexibility may provide much more economical alternatives. For example, little time has been devoted to using lists instead of strings because of the appearance of the strung-out displays. Also, attempts to use an algebraic approach to the arithmetical portion thus far haven't produced any substantial advantage, probably because the lion's share of the routine must be involved in stack and string manipulation. For the same reason, the use of local variables was abandoned early on.

HP-28C COMMENTS AND EEX/CHS BUG?

By Stephan J. Thomas #6907
4224 Jackson Street
Hollywood, FL 33021

Anybody who has had the opportunity to use the new HP-28C would certainly agree that it is a fascinating calculator. Among its many strengths are true RPN and an "unlimited" stack, not to mention the wealth of built-in applications.

As usual though, the 28C is woefully lacking in RAM -- a total of 2K, of which only about 1.6K are available to the user and for dynamic operations. One soon finds that this small amount of RAM is very quickly used up by surprisingly few stored variables.

The Reference Manual is agonizingly thorough in listing the workings of the many functions, but is often difficult to follow. If the 28C is somebody's first HP calculator, they will soon be pulling out their hair in frustration. (I believe that the authors of the HP-41CX manual volume 2 must have left HP and taken all the RAM chips with them!)

There is a surprising number of high-level programming commands available (especially for branching and looping), but debugging a program is more difficult than on the 41. When a program errors out, it terminates without letting you know which command caused the error (unless you put in lots of testing for error flags). There are no numbered program lines.

HP finally gave us a larger multi-line display, only to equip the calculator with just a couple of terribly primitive commands to control it.

Running a couple of simple off-the-cuff benchmark tests, the 28C seems to run about 6 to 10 times faster than the 41. Execution time can be substantially increased when the arguments are symbolic rather than numeric.

THE BUG (or is it?)

After having entered a real number followed by a command (separated by a SPACE) into the command line one fine spring day, I realized that I had left the exponent off the number. I used the cursor LEFT key (◀, not ⬅) to move the cursor to the space following the real number, entered INSert mode and pressed EEX. Instead of inserting an 'E' into the command line, the 28C inserted '1E'!

Yes, that extra '1' which has plagued 41 users is back again in the 28C.

To see this 'bug', enter:

5
SPACE
◀ (cursor LEFT, not ⬅)
INS (this step not required)
EEX

You will see '51E' instead of the expected '5E'.

Apparently, if there are <u>any</u> characters to the right of a real number on the same displayed line of the command line, the 28C does not consider a cursor immediately to the right of the number to be positioned <u>at</u> the number -- even in INSert mode!

Page 33 of the HP-28C Reference Manual states: "If the cursor is not positoned at a valid number . . . pressing EEX adds the characters '1E' to the command line."

CHS (change sign) behaves similarly to EEX under the same conditions.

Be careful anytime you are editing numbers in the command line, such as:

- -- VISITing a STOred object or stack level greater than 1
- -- EDITing stack level 1
- -- EDITing a command line which has not yet been ENTERed or EVALuated, or which has been recalled by COMMAND.

This anomaly is present when editing "stand-alone" real numbers or real numbers which are part of complex number, vectors, arrays, algebraics or programs. It does not matter whether the number is an integer or non-integer.

My HP-28C is version 1BB, serial number 2712A.

<u>WORKAROUNDS</u>

There are at least two workarounds:

1. Position the cursor <u>over</u> any digit of the real number, then press EEX. A lone 'E' will be entered into the command line at the end of the real number. This technique also works for CHS.

2. With the cursor positioned immediately to the right of the number, use the alphabetic character 'E' to begin exponent entry.

<u>HP-28C REFERENCE MANUAL CORRECTIONS</u>

A. Page 69, second from last paragraph:
'7+X+T' should be '7+(X+T)'.
'10+T' should be '7+(3+T)'.
Note: to get '10+T' you must use the COLCT function.

B. Page 161, middle of page:
'page 000' should be 'page 126'.

<u>MISCELLANEOUS</u>

Note that the % function -- which returns x percent of y -- does NOT leave y in level 2 (the "y-register"), unlike previous HP calculators.

Executing CHS on a negative number does not immediately remove the minus sign, but changes it to a plus sign. This plus sign is then dropped when the command line is ENTERed.

When creating a user-defined unit for unit conversions, be sure to surround the unit string with double quotes (") rather than single quotes (') -- even if the unit string is a system-defined unit.

Can this machine solve the indefinite integral $\int(1/x)dx$?

A "lower-case" annunciator would have been really helpful.

If anybody has discovered uses of SYSEVAL other than determining the operating system version, I would greatly appreciate any info. Better yet, write it up for the Journal.