The HP–67 Fun Rom

This project had its beginnings about a year and a half ago when it was noticed that in the wealth of programs available for the HP 41 calculator, there were some gaps. Some programs developed for the HP 67 calculator had no equivalent HP–41 series option. This rom set out to remedy that and preserve some great programs for that wonderful machine, converted to run on the HP–41.

Contributors: Obviously, the original authors deserve the most credit. Thanks to them. They are noted for the programs when the original author is known.

On the HP Museum forum, Dieter Lefeling did most of the heaving programming optimization and rework. In many ways, this is his rom and product more than anyone else. Robert Meyer spent a great deal of time on the Space War and Star Trek ports and was always willing to do a test run of anything. Many thanks. Angel Martin deserves more credit that we could possibly give for so many things related to HP calculators. Thanks for helping make this rom a reality.

The Rom. The HP67 FUN rom has 28 functions, listed below. The –HP67 FUN+ rom is XROM ID 23, chosen because it conflicts with very few existing roms. As of November 2016, the roms the HP67 FUN rom does conflict with are:

- HP Extended I/O
- Trans Neptunian Planets

28 FUNCTIONS	XROM Number	Description	Page in this manual
FCT: -HP67 FUN+	XROM 23,00	Rom Header	1
FCT: "1130"	XROM 23,01	Game of 11-30	2
FCT: "2636"	XROM 23,02	Game of 26 and 36	4
FCT: "ART"	XROM 23,03	Artillery game	7
FCT: "BSP"	XROM 23,04	Battleship	10
FCT: "CH"	XROM 23,05	Chess	12
FCT: "CHK"	XROM 23,06	Chuck-a-luck	15
FCT: "GOLF"	XROM 23,07	Golf	17
FCT: "JT"	XROM 23,08	Jive Turkey	19
FCT: "ML"	XROM 23,09	Moon Lander	21
FCT: "OAB"	XROM 23,10	One Arm Bandit	23
FCT: "SPW"	XROM 23,11	SpaceWar	25
FCT: "ST"	XROM 23,12	Star Trek	31
FCT: "TTT"	XROM 21,13	Tic-Tac-Toe	36
FCT: "AOS"	XROM 23,14	Algebraic Operating System	38
FCT: "+"	XROM 23,15	-- function in AOS Program	
FCT: "-"	XROM 23,16	-- function in AOS Program	
FCT: "*"	XROM 23,17	-- function in AOS Program	
FCT: "/"	XROM 23,18	-- function in AOS Program	
FCT: "<"	XROM 23,19	-- function in AOS Program	
FCT: ">"	XROM 23,20	-- function in AOS Program	
FCT: "="	XROM 23,21	-- function in AOS Program	
FCT: "YX"	XROM 23,22	-- function in AOS Program	
FCT: "NEG"	XROM 23,23	-- function in AOS Program	
FCT: "SD"	XROM 23,24	Sum of Digits game	43
FCT: "S"	XROM 23,25	SEED? prompt and store seed in R00 subroutine	45
FCT: "R"	XROM 23,26	Random number generator subroutine	45
FCT: "RF"	XROM 23,27	Reset Flags routine from PPC ROM	45

Where to find it? The rom in a .rom and .mod format should be available at www.hp41.org and also in the rom reference section at Monte Dalrymple's HP-41CL site: http://www.systemyde.com/zip/H67G.ZIP

Bugs. There are no known bugs in the code at this time (2017-8-23).

Note: In the program listings shown in this document, the instruction "`X!=Y?`" indicates X NOT EQUAL Y. In general, the symbol "`!=`" simply indicates NOT EQUAL.

Manual revision history:

Version 1.03 – 1) Fixed typo in the RF routine description that indicated incorrect flags affected by the text line.
2) Removed some ending periods in the "Specifics" section of routine documentation.
3) Fixed misspelling of "FUNCTIONS" on this main page!

Game of 1130 (LBL "1130")

History: This is based on the HP–65 game of Eleven–Thirty written by John Rausch. It appeared in the V2N3P28 issue of PPC Journal (March 1975).

Object: The calculator generates two random numbers between 11 and 30 and displays them as XX–YY BET? (for example, "15–22 BET?" could be displayed). The user then places a bet that the next number is equal to or between the two numbers displayed, endpoints inclusive. You win if it is and pay if it is not. Obviously, if "21–22 BET?" is displayed, then you should probably bet $0 since the most likely next number is probably not 21 or 22, so why lose the next bet? If the number 21 was shown next anyway, you would "win" but since your bet was $0, you won $0 – you certainly did not lose any money.

Instructions:

1) XEQ ALPHA 1130 ALPHA
2) At the SEED? prompt, enter a decimal seed between 0 and 1 and press R/S.
3) The display will show the size of the original POT, which is $0.
4) Press A to begin.
5) Calculator displays numbers in the form of XX–YY and asks you to enter a bet.
6) Enter a value for your bet (only INTEGERS allowed) that the next number is between XX and YY and press R/S.
7) Calculator displays the next number, WIN or PAY and shows the amount won or paid and then amount of winnings in the POT.
8) Press R/S to try again and go to step 5.
9) To check your winnings, press E in USER mode. Display shows POT=$ZZ. Press A to play again.

Example game:

	See	Press	
1)		XEQ ALPHA 1130 ALPHA	
2)	SEED?	0.445566 R/S	
3)	POT: $0	A	(Pot starts at $0. Pressing A begins the game.)
4)	13–26 BET?	10 R/S	(Bet $10 that next number is within 13–26)
5)	15: WIN $10		(Won $10, number was 15)
6)	POT: $10	R/S	
7)	21–22 BET?	0 R/S	(Bet $0 – don't expect a 21 or 22)
8)	27: PAY $0		(Good thing, since the next number was 27)
9)	POT: $10	R/S	
10)	23–24 BET?	0 R/S	(Bet $0 that next number is within 23–24)
11)	29: PAY $0		(Good thing, since the next number was 29)
12)	POT: $10	R/S	
13)	17–29 BET?	50 R/S	(Feeling lucky!)
14)	11: PAY $50		(Oops. Lost!)
15)	POT: $–40	R/S	(Press R/S to keep playing if desired)

Specifics:

1) Program is 121 bytes long.
2) Program is XROM 23,01.
3) Calls XROM "S" – XROM 23,25 (SEED? prompt) and XROM "R" – XROM 23,26 (Random Number Generator).
4) Uses registers 00 – 03. SIZE 004 required.
    - 00 – Random number seed
    - 01 – First two–digit number (XX)
    - 02 – Second two–digit number (YY)
    - 03 – Amount of winnings in pot
5) Labels used:
    - 01 – Random number routine. Calls XROM "R" (23,26).
    - A – Starts a new game
    - E – Displays winnings in pot
6) Flags used:
    - 27 – Flag 27 is set to allow use of user–defined keys
    - 29 – Flag 29 is used for formatted output
7) Display mode: FIX 00 is set. The existing display mode is not retained.

Program Listing:

```
01 LBL "1130"
02 XROM “S”
03 CLX
04 STO 03
05 SF 27
06 LBL E
07 FIX 00
08 CF 29
09 "POT: $"
10 ARCL 03
11 PROMPT
12 LBL A
13 XEQ 01
14 XEQ 01
15 X>Y?
16 X<>Y
17 STO 01
18 X<>Y
19 STO 02
20 CHS
21 CLA
22 ARCL Y
23 ARCL X
24 >"  BET?"
25 CLST
26 PROMPT
27 INT
28 ABS
29 XEQ 01
30 CLA
31 ARCL X
32 >":  "
33 RCL 02
34 X<>Y
35 -
36 LASTX
37 RCL 01
38 -
39 *
40 SIGN
41 X<0?
42 >"PAY "
43 X>0?
44 >"WIN "
45 *
46 ST+ 03
47 >"$"
48 ARCL Y
49 AVIEW
50 PSE
51 PSE
52 GTO E
53 LBL 01
54 XROM “R”
55 20
56 *
57 11
58 +
59 INT
60 END
```

Game of 26 and 36 (LBL "2636")

History: This is based on the game of Twenty–six or Thirty–six from the HP 67/97 Users' Library Solutions Book Games of Chance, page 6. Originally written by Matthew Bishop.

Object: Game of 26: You choose a number from 1 to 6 as your number and pay $0.25 to play. The calculator then rolls 10 dice 13 times for a total of 130 numbers of 1 to 6. If your number appears 11 or fewer times, then you win a dollar. If it appears exactly 13 times, then you win $0.50. If it appears 26 or more times, you win $1. If it appears 33 or more times, you win $2. (Note: For example, if your number appears 28 times, you win $1, if it appears 35 times, you win $2 total – $1 for it appearing more than 26 times and another $1 for it appearing more than 33 times.)

Instructions for the Game of 26:

1) XEQ ALHA 2636 ALPHA
2) At the SEED? prompt, enter a decimal seed between 0 and 1 and press R/S.
3) To play 26, key in die you choose (1–6) and press A. Each play of the game of 26 costs $0.25.
4) Display pauses and displays a sequence of 10–digit random numbers (13 times).
5) The number of times your number occurs is displayed, followed by the amount you won or lost, followed by your total monetary position. To play again, go back to step 3.

Example Game of 26:

	See	Press	
1)		XEQ ALPHA 2636 ALPHA	
2)	SEED?	0.123456 R/S	
3)	BANK=$0.00	6 A	(Note: Choose the number 6 and play the game of 26)
4)	5561161245		
	1435316463		
	…		(10 more sets of 10–digit numbers will be shown here)
	5552332416		
	20 TIMES		(6 occurred 20 times, so you win nothing)
	WIN $0.00		
	BANK=$–0.25		(To play again, pick your number and press A)

Object: Game of 36: Place a bet (deducted from your account). Player continues to roll dice until deciding to stop or the sum or all numbers rolled exceeds 36 automatically losing. When the first player is done, if his or its total is 36 or less, the second player rolls, following the same procedure. If the second player stops before his or its total exceeds 36, the totals are compared. Whoever comes closest to 36, wins. On a tie, you get your bet back. The calculator will match your bet (winner gets total bet by both players); the calculator uses a simple strategy to decide when to stop rolling dice.

Instructions for the Game of 36:

1) XEQ ALHA 2636 ALPHA
2) At the SEED? prompt, enter a decimal seed between 0 and 1 and press R/S.
3) To play 36, decide if you or the calculator should go first. There is no cost to play a game of 36.
4) If you wish to go first, enter your bet and press B and the calculator will roll your dice. Go to step 6.
5) If you wish to let the calculator go first, enter your bet and press C. Go to step 12.

6) The calculator will display your total points and show HP's points, which start at 0.
7) Continue to let the program run until you wish to stop accumulating your points. If you do not stop before your total goes over 36, then you automatically lose.
8) When you wish to stop, press the decimal point or the zero key while the score is displayed and the calculator will begin accumulating points. If you are not fast enough typing during the pause, you will probably lose!
9) If the calculator goes over 36 before it stops, you automatically win.
10) If the calculator stops with its total under 36, the winner is whoever is closer to 36.
11) Go to step 4 to play again.

12) The calculator will begin to accumulate points, and will stop at some point and then it will be your turn to accumulate points.
13) Continue to let the program run until you wish to stop accumulating your points. If you do not stop before your total goes over 36, then you automatically lose.
14) When you wish to stop, press the decimal point or the zero key while the score is displayed and the calculator will begin accumulating points. If you are not fast enough typing during the pause, you will probably lose!
15) If you stop before you have a higher score than the calculator or if you go over 36, then you will lose.
16) If you stop and have more points than the calculator, then you win!
17) Go to step 4 to play again.

Example Game of 36:

	See	Press	
1)		XEQ ALPHA 2636 ALPHA	
2)	SEED?	0.123456 R/S	
3)	BANK=$0.00	5 B	(Note: Bet $5 and go first in a game of 36)
4)	YOU–5 HP–0		
5)	YOU–10 HP–0		
6)	YOU–16 HP–0		
7)	YOU–17 HP–0		
8)	YOU–18 HP–0		
9)	YOU–23 HP–0		
10)	YOU–24 HP–0		
11)	YOU–26 HP–0		
12)	YOU–30 HP–0		
13)	YOU–35 HP–0	Press . (the decimal point) or the number 0 quickly to stop at 35!	
14)	YOU–35 HP–1		
15)	YOU–35 HP–5		
16)	YOU–35 HP–8		
17)	YOU–35 HP–13		
18)	YOU–35 HP–16		
19)	YOU–35 HP–17		
20)	YOU–35 HP–23		
21)	YOU–35 HP–27		
22)	YOU–35 HP–33		
23)	YOU–35 HP–36		
26)	PAY $5		(We lost!)
27)	BANK=$–5.00		

Specifics:

1) Program is 297 bytes long.
2) Program is XROM 23,02.
3) Calls XROM "S" – XROM 23,25 (SEED? prompt) and XROM "R" – XROM 23,26 (Random Number Generator).
4) Game of 26: Uses registers 00 – 05. SIZE 006 required.
    - 00 – Random number seed
    - 01 – Bank $
    - 02 – Times your die occurred in game of 26
    - 03 – Loop counter (13 runs)
    - 04 – Loop counter (10 numbers per loop)
    - 05 – Point for game of 26, amount won/lost
5) Game of 36: Uses registers 00 – 05. SIZE 006 required.
    - 00 – Random number seed
    - 01 – Bank $
    - 02 – Control number for indirect call (8=human, 9=HP)
    - 03 – Your score in 36
    - 04 – HP's score in 36
    - 05 – Amount of bet
6) Labels used:
    - 00, 06, 07 – used for various purposes.
    - 01 – Random number routine. Calls XROM "R" (23,26).
    - 02 – You lost
    - 03 – Common label to display results
    - 08 – User rolls a die
    - 09 – HP rolls a die
    - 77 – Roll a die, display scores, check if > 36
    - 88 – A win is detected, adjust/display amount won/lost, show bank
    - 99 – Roll a die (random number between 1 and 6)
    - A – Starts a new game of 26
    - B – Enter bet and starts a new game of 36 – human first
    - C – Enter bet and starts a new game of 36 – HP first
    - D – Displays BANK amount
7) Flags used:
    - 00 – Set if score > 36
    - 01 – Set = user first, clear = HP first for game of 36
    - 27 – Flag 27 is set to allow use of user–defined keys
8) Display mode: FIX 00 and FIX 02 are set for output. The existing display mode is not retained.

Program Listing:

```
 01 LBL "2636"
 02 XROM “S”
 03 CLX
 04 STO 01
 05 GTO D
 06 LBL 01
 07 10
 08 STO 04
 09 CLX
 10 LBL 02
 11 10
 12 *
 13 XEQ 99
 14 RCL 05
 15 X=Y?
 16 ISG 02
 17 LBL 00
 18 RDN
 19 +
 20 DSE 04
 21 GTO 02
 22 FIX 00
 23 CF 29
 24 VIEW X
 25 DSE 03
 26 GTO 01
 27 PSE
 28 PSE
 29 " "
 30 ARCL 02
 31 >" TIMES"
 32 AVIEW
 33 PSE
 34 1
 35 STO 05
 36 15
 37 RCL 02
 38 11
 39 -
 40 X<Y?
 41 X<=0?
 42 GTO 03
 43 2
 44 -
 45 ABS
 46 .5
 47 X<Y?
 48 INT
 49 STO 05
 50 GTO 88
 51 LBL 03
 52 22
 53 X<=Y?
 54 ISG 05
 55 LBL 88
 56 RCL 05
 57 ST+ 01
 58 FIX 02
 59 "WIN $"
 60 X<0?
 61 "PAY $"
 62 ABS
 63 ARCL X
 64 AVIEW
 65 PSE
 66 LBL D
 67 CF 01
 68 SF 27
 69 FIX 02
 70 RCL 01
 71 "BANK=$"
 72 ARCL 01
 73 AVIEW
 74 RTN
 75 LBL A
 76 STO 05
 77 .25
 78 ST- 01
 79 13
 80 STO 03
 81 CLX
 82 STO 02
 83 GTO 01
 84 LBL B
 85 SF 01
 86 CHS
 87 GTO 07
 88 LBL C
 89 CF 01
 90 LBL 07
 91 STO 05
 92 CLX
 93 STO 04
 94 STO 03
 95 9
 96 STO 02
 97 FS? 01
 98 DSE 02
 99 CF 00
100 FIX 00
101 CF 29
102 XEQ IND 02
103 FS?C 00
104 GTO 88
105 17
106 ST- 02
107 XEQ IND 02
108 RCL 05
109 CHS
110 STO 05
111 FS?C 00
112 GTO 88
113 RCL 03
114 RCL 04
115 -
116 X!=0?
117 SIGN
118 RCL 05
119 ABS
120 *
121 STO 05
122 GTO 88
123 LBL 08
124 3
125 XEQ 77
126 X>0?
127 FS? 00
128 RTN
129 GTO 08
130 LBL 09
131 4
132 XEQ 77
133 FS? 00
134 RTN
135 RCL 03
136 RCL 04
137 FS? 01
138 X=Y?
139 GTO 06
140 X<Y?
141 GTO 09
142 RTN
143 LBL 06
144 33
145 RCL 04
146 X<=Y?
147 GTO 09
148 RTN
149 LBL 77
150 XEQ 99
151 ST+ IND Y
152 36
153 RCL IND Z
154 X>Y?
155 SF 00
156 "YOU-"
157 ARCL 03
158 >" HP-"
159 ARCL 04
160 AVIEW
161 PSE
162 RTN
163 LBL 99
164 XROM “R”
165 6
166 *
167 INT
168 1
169 +
170 END
```

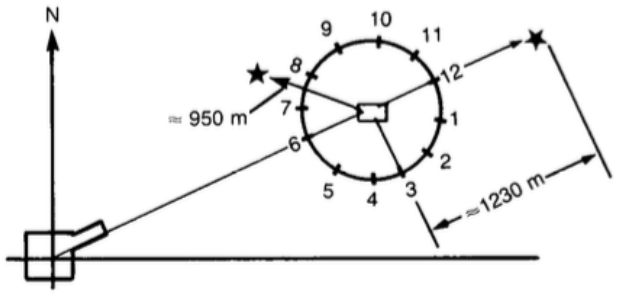
Artillery game (LBL "ART")

History: This program is modified from the Artillery game found in the HP 67 Games Pac, program 05.

Object: This program simulates the firing of an artillery round at a moving target whose initial position has been randomly selected. Feedback to the gunner is provided by a spotter plane weaving in and out of the clouds over the battle area.

Instructions:

1) XEQ ALPHA ART ALPHA
2) At the SEED? prompt, enter a decimal seed between 0 and 1 and press R/S.
3) Adjust some of the settings for the game if desired before you begin:
    a. Pressing shift b changes the speed of the target. The default is 500 per move. A lower value will be easier to hit and kill. A high value makes it much more difficult.
    b. Pressing shift c changes the effectiveness of the spotter. Values range from 1 (very poor) to 4 (perfect). The default is 3.
    c. Pressing shift d changes the kill range – how close to the target the shot needs to land in order to kill it. The default range is 100. A value of 1000 makes it fairly easy while a value of 10 makes it very difficult.
4) Press A to start the battle. Display shows approximate bearing.
5) Key your desired shot bearing, press ENTER and then key the elevation for the shot in degrees. Press E to shoot.
6) Display will show the B∠XX.X (bearing) and E∠YY.Y (elevation) of your shot.
7) It will then display a result of hh.DDDD, where:
    a) hh (0 to 12) = the shell hit as an hour position on a relative clock face with the target at the center and 6 o'clock in line with the gun, and
    b) DDDD is the estimated range from the target to the shell hit.
8) For example, in the diagram below, a display of 8.0950 would mean the shell hit a bit left and was short of the target by approximately 950 meters. A reading of 12.1230 would mean the shell went over the target and fell 1230 meters beyond it.



9) Note: After each shot, the target moves toward your position, so destroy it quickly.
10) Evaluate the information presented and return to step 5 until you destroy the target.
11) If the target gets closer than 500 meters to your position and you have not yet destroyed it, it will "blast your gun to pieces" (to quote the HP–67 Games Pac I manual). The calculator will display "CLOSE…" and then "YOU'RE HIT" to indicate you have failed.

Example game:

	See	Press	
1)		XEQ ALPHA ART ALPHA	
2)	SEED?	0.4711 R/S	
3)	0.00	shift b	(Target speed set to zero for an easier example)
4)	0.00	A	(Start the game)
5)	45.	45 ENTER 30 E	(Note: 45 is the approximate bearing to target)
6)	B∠45.0 E∠30.0		(Shot at bearing of 45 degrees and elevation of 30 degrees)
7)	8.2570		(Note: A bit to the left and 2570 meters from target)
8)	8.2570	60 ENTER 30 E	(Try a bearing a bit to the left, same elevation
9)	B∠60.0 E∠30.0		(Shot at bearing of 60 degrees and elevation of 30 degrees)
10)	6.0860		(Much better. We are 860 meters short)
11)	6.0860	60 ENTER 35 E	(Raise the shot a little)
12)	B∠60.0 E∠35.0		(Shot at bearing of 60 degrees and elevation of 35 degrees)
13)	9.0120		(Left of target, but very close! Elevation seems fine)
14)	9.0120	61 ENTER 35 E	(Adjust bearing slightly)
15)	B∠61.0 E∠35.0		(Shot at bearing of 61 degrees and elevation of 35 degrees)
16)	TARGET HIT		(Got it!)

17) DIST: 41.1 (Shell hit only 41 meters from target)

Notes: 1) 60.8 as bearing and 35.2 elevation would have been a nearly direct hit.
2) Pressing C displays the rounded bearing (61) and distance (9422 meters).

Specifics:

1) Program is 285 bytes long.
2) Program is XROM 23,03.
3) Calls XROM "S" – XROM 23,25 (SEED? prompt) and XROM "R" – XROM 23,26 (Random Number Generator).
4) Uses registers 00 – 09. SIZE 010 required.
   - 00 – Random number seed
   - 01 – Target N–S
   - 02 – Target E–W
   - 03 – Shell N–S
   - 04 – Shell E–W
   - 05 – Shell's rounded distance from target
   - 06 – Shell's rounded direction from target
   - 07 – Speed of target
   - 08 – Spotter rating
   - 09 – Kill range
5) Labels used:
   - 01 – Adjust target position by random amount. Calls XROM "R" (23,26).
   - 03 – Generate hh.DDD output
   - 07 – Check if target has approached to less than 500 meters
   - 08 – Target closer than 500 meters. Display "CLOSE" and "YOU'RE HIT" indicating game over.
   - 09 – Change exact bearing / distance slightly by a random amount
   - A – Starts a new game, resets target
   - C – Display Bearing and Distance to target. Output is bbb.DDDD.
   - E – Fire shot with new Bearing ENTER Elevation
   - a – Initialize game for first time
   - b – Set target speed if different from default 500
   - c – Change spotter abilities (1 – poor to 4 – perfect). Default is 3.
   - d – Set kill range. 1000 is easy, 10 is tough. Default is 100.
6) Flags used:
   - 27 – Flag 27 is set to allow use of user–defined keys
   - 42,43 – Flags 42 and 43 are cleared by the DEG command on line 05
7) Display mode: FIX 00, 01, 02, and 04 are set for output. Fix 03 is set in order to round the distance to the nearest 10 meters. The existing display mode is not retained.
8) Program listing note: Lines 64 and 66 show a code in the listings of "\0D" which represents the angle symbol, entered into the ALPHA register by pressing shift CHS (or SHIFT O for the letter designation).

Program listing:

```
 01 LBL "ART"
 02 XROM “S”
 03 SF 27
 04 LBL a
 05 DEG
 06 FIX 02
 07 500
 08 STO 07
 09 3
 10 STO 08
 11  E2
 12 STO 09
 13 CLST
 14 RTN
 15 LBL A
 16 5 E3
 17 ENTER
 18 XROM “R”
 19 *
 20 INT
 21 +
 22 360
 23 XROM “R”
 24 *
 25 STO 06
 26 R^
 27 P-R
 28 STO 01
 29 X<>Y
 30 STO 02
 31 45
 32 ST/ 06
 33 RCL 06
 34 FIX 00
 35 RND
 36 *
 37 RTN
 38 LBL C
 39 RCL 02
 40 RCL 01
 41 R-P
 42  E4
 43 /
 44 X<>Y
 45 360
 46 MOD
 47 FIX 00
 48 RND
 49 +
 50 FIX 04
 51 RTN
 52 LBL b
 53 STO 07
 54 RTN
 55 LBL c
 56 STO 08
 57 RTN
 58 LBL d
 59 STO 09
 60 RTN
 61 LBL E
 62 ADV
 63 FIX 01
 64 "B\0D"
 65 ARCL Y
 66 >" E\0D"
 67 ARCL X
 68 AVIEW
 69 ENTER
 70 +
 71 SIN
 72  E4
 73 *
 74 P-R
 75 STO 03
 76 X<>Y
 77 STO 04
 78 RCL 01
 79 XEQ 01
 80 STO 01
 81 RCL 02
 82 XEQ 01
 83 STO 02
 84 CHS
 85 RCL 04
 86 +
 87 RCL 03
 88 RCL 01
 89 -
 90 R-P
 91 STO 05
 92 RCL 09
 93 X<>Y
 94 X>Y?
 95 GTO 07
 96 "TARGET HIT"
 97 AVIEW
 98 PSE
 99 PSE
100 FIX 01
101 "DIST: "
102 ARCL X
103 AVIEW
104 RTN
105 LBL 01
106 RCL 07
107 XROM “R”
108 *
109 R^
110 SIGN
111 *
112 -
113 RTN
114 LBL 07
115 RDN
116 RDN
117 RCL 02
118 RCL 01
119 R-P
120 500
121 X>Y?
122 GTO 08
123 RDN
124 RDN
125 -
126 30
127 /
128 4
129 RCL 08
130 -
131 XEQ 09
132 12
133 MOD
134 FIX 00
135 RND
136 X=0?
137 12
138 STO 06
139 RCL 05
140 .2
141 ENTER
142 4
143 RCL 08
144 -
145 *
146 *
147 X=0?
148 GTO 03
149 XEQ 09
150 RCL 09
151 LBL 03
152 X<=Y?
153 X<>Y
154  E4
155 /
156 FIX 03
157 RND
158 RCL 06
159 +
160 FIX 04
161 VIEW X
162 RTN
163 LBL 08
164 X<>Y
165 "CLOSE..."
166 AVIEW
167 PSE
168 PSE
169 "YOU'RE HIT"
170 AVIEW
171 RTN
172 LBL 09
173 -
174 LASTX
175 ENTER
176 +
177 XROM “R”
178 *
179 +
180 END
```

Game of Battleship (LBL "BSP")

History: This program is a modification of Battleship from the HP–67 Games Users Library Solutions book page 60 and was written by Richard Toptani.

Object: An enemy battleship is somewhere ahead of you. For each torpedo you shoot targeting X and Y coordinates, your instruments will tell you how far away the shot was from the enemy and how many torpedoes you have left. Can you sink the enemy battleship before you run out of torpedoes?

Instructions:

1) XEQ ALPHA BSP ALPHA
2) At the SEED? prompt, enter a decimal seed between 0 and 1 and press R/S.
3) The calculator will display the number of torpedoes left and you will be prompted to SHOOT.
4) Key in the X coordinate for the shot, press ENTER and then key in the Y coordinate and press R/S.
5) The calculator will display the distance the shot landed from the target.
6) If the distance is more than 1 unit away but closer than 5 units away, HITS=# will be displayed, where # represents the cumulative number of hits the enemy ship has received.
7) If the distance is within 1 unit for a shot, *DESTROYED* will be displayed and you have won. For a new game, press A.
8) If you have no more torpedoes (because you have used all 12), you lose! Otherwise, go back to step 3.

Example game:

	See	Press	
1)		XEQ ALPHA BSP ALPHA	
2)	SEED?	0.123456 R/S	
3)	TORP=12		
4)	SHOOT	50 ENTER 50 R/S	
5)	DIST=28.60		
6)	TORP = 11		
7)	SHOOT	25 ENTER 25 R/S	
8)	DIST=42.05		
9)	TORP = 10		
10)	SHOOT	75 ENTER 25 R/S	
11)	DIST=8.25		
12)	TORP=9		
13)	SHOOT	70 ENTER 25 R/S	
14)	DIST=3.61		
15)	HITS=1		(A hit!)
16)	TORP=8		
17)	SHOOT	72 ENTER 27 R/S	
18)	DIST=13.89		(The ship has started moving away!)
19)	TORP=7		
20)	SHOOT	68 ENTER 20 R/S	
21)	DIST=8.00		
22)	TORP=6		
23)	SHOOT	72 ENTER 20 R/S	
24)	DIST=12.00		
25)	TORP=5		
26)	SHOOT	62 ENTER 20 R/S	
27)	DIST=2.00		
28)	HITS=2		(Another hit!)
29)	TORP=4		
30)	SHOOT		(Can you find the ship now?)

Specifics:

1) Program is 159 bytes long.
2) Program is XROM 23,04.
3) Calls XROM "S" – XROM 23,25 (SEED? prompt) and XROM "R" – XROM 23,26 (Random Number Generator).
4) Uses registers 00 – 05. SIZE 006 required.
    - 00 – Random number seed
    - 01 – Battleship's position (X)
    - 02 – Battleship's position (Y)
    - 03 – Number of hits
    - 04 – Battleship's maneuverability in the X/Y direction
    - 05 – Number of torpedoes left

5) Labels used:
   - 01 – Main loop (show torpedoes, prompt for coordinates)
   - 02 – Torpedoes > 0, skips "you lose" message
   - 03 – Enemy destroyed (direct hit or 5 minor hits)
   - 04 – Display final "win/lose" message. Resets display.
   - A – Starts a new game
6) Flags used:
   - 27 – Flag 27 is set to allow use of user–defined keys
   - 29 – Flag 29 is used for formatted output. The program ends with flag 29 set.
7) Display mode: FIX 00, 02 are set for output. The existing display mode is not retained and the program ends with FIX 04 set.

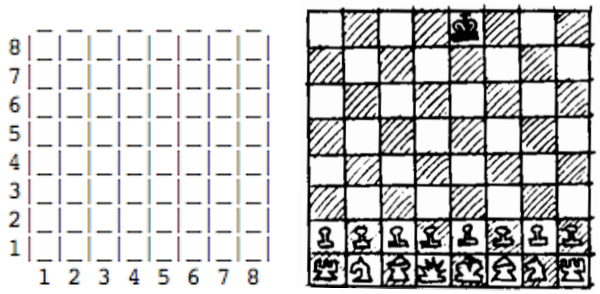
Program listing:

```
 01 LBL BSP"                 30 "TORP="                  59 "HITS="
 02 XROM "S"                 31 ARCL 05                  60 ARCL 03
 03 SF 27                    32 AVIEW                    61 AVIEW
 04 LBL A                    33 PSE                      62 PSE
 05 CF 29                    34 CLST                     63 4
 06 XROM "R"                 35 "SHOOT"                  64 RCL 03
 07  E2                      36 PROMPT                   65 X>Y?
 08 *                        37 FIX 02                   66 GTO 03
 09 INT                      38 RCL 02                   67 ST- 04
 10 STO 01                   39 -                        68 RCL 01
 11 LASTX                    40 X<>Y                     69 RCL 04
 12 FRC                      41 RCL 01                   70 X>Y?
 13  E2                      42 -                        71 CHS
 14 *                        43 R-P                      72 -
 15 INT                      44 "DIST="                  73 STO 01
 16 STO 02                   45 ARCL X                   74 RCL 02
 17 CLX                      46 AVIEW                    75 RCL 04
 18 STO 03                   47 PSE                      76 X>Y?
 19 8                        48 PSE                      77 CHS
 20 STO 04                   49 5                        78 -
 21 13                       50 X<Y?                     79 STO 02
 22 STO 05                   51 GTO 01                   80 GTO 01
 23 LBL 01                   52 SIGN                     81 LBL 03
 24 DSE 05                   53 X<>Y                     82 "*DESTROYED*"
 25 GTO 02                   54 X<=Y?                    83 LBL 04
 26 "YOU LOSE"               55 GTO 03                   84 FIX 04
 27 GTO 04                   56 ISG 03                   85 SF 29
 28 LBL 02                   57 CLA                      86 AVIEW
 29 FIX 00                   58 FIX 00                   87 END
```

Game of Chess (LBL "CH")

History: This program is a modified version of the Chess program that appeared in V7N6P18 of the PPC Journal, written by Valentin Albillo.

Object: This program allows you to play chess against the calculator. The calculator controls the 16 white pieces and you control only the black king. The calculator will try to checkmate you in 6 moves or less. Your goal is to survive longer than 6 moves. The game is played on the following board and the coordinates are how the user enters the human move. The black king (the human here) begins in square 58. Note that the program sometimes does not detect or indicate check properly (for example, it does not notice check in move 3 in the sample game). The program does detect checkmate properly, however. Also, the program does not check for illegal moves, so play honestly!



Instructions:

1) XEQ ALPHA CH ALPHA
2) The calculator will display its first move. The black king starts in square 58.
3) Enter your move as a coordinate on the 8x8 grid above and press R/S.
4) The calculator will display its next move. Go back to step 3 until you survive past six moves or the HP checkmates you!
5) Play honestly as the program does not detect illegal moves.

Example game:

	See	Press	
1)		XEQ ALPHA CH ALPHA	
2)	1. P–K4	48 R/S	
3)	2. Q–N4	37 R/S	
4)	3. Q–N7	46 R/S	(Note: check, but not indicated)
5)	4. P–Q4	36 R/S	
6)	5. B–KB4	26 R/S	
7)	6. Q–B7 ++		(Checkmate. Rats!)

Specifics:

1) Program is 455 bytes long.
2) Program is XROM 23,05.
3) Uses register 01. SIZE 002 required. Program also recalls / stores status register d (flags).
   01 – Move number
4) Labels used:
   00, 10 – 15 – Used
   01 – Translates a code of 1 into a letter for Pawn – append "P"
   02 – Translates a code of 2 into a letter for Knight – append "N"
   03 – Translates a code of 3 into a letter for Bishop – append "B"
   04 – Translates a code of 4 into a letter for Rook – append "R"
   05 – Translates a code of 5 into a letter for Queen – append "Q"
   06 – Translates a code of 6 into a letter for King – append "K"
   07 – Display HP's move (calls label 98), get user's move and divide by 10
   08 – Second to last move
   09 – Display final move, "checkmate"
   98 – Decode move, generate Alpha, including "check" (if applicable)
   99 – Display HP's move (calls label 98) and get user's move
5) Flags used: Flag 29 is cleared for formatted output, but since the original contents of the status register d that controls flag settings is RCL to the stack at step 173 and then STO back into register d at step 210, the flags are not changed unless the program is interrupted between steps 173–210. If that happens, XEQ the RF program included in this ROM to reset the flags to a default state.
6) Display mode: FIX 00 is set for output. The existing display mode is restored.

Program listing:

```
 01 LBL "CH"
 02 CLST
 03 STO 01
 04 164
 05 XEQ 99
 06 524
 07 XEQ 07
 08 INT
 09 6
 10 X=Y?
 11 GTO 13
 12 527
 13 XEQ 07
 14 FRC
 15 .8
 16 X=Y?
 17 GTO 14
 18 154
 19 XEQ 99
 20 26
 21 X!=Y?
 22 GTO 00
 23 557
 24 LBL 08
 25 XEQ 99
 26 525
 27 LBL 09
 28 CHS
 29 XEQ 98
 30 >"+"
 31 AVIEW
 32 RTN
 33 LBL 00
 34 10
 35 +
 36 X!=Y?
 37 GTO 00
 38 3634
 39 XEQ 99
 40 GTO 15
 41 LBL 00
 42 10
 43 +
 44 X!=Y?
 45 GTO 00
 46 -3634
 47 XEQ 99
 48 56
 49 X!=Y?
 50 GTO 15
 51 155
 52 GTO 09
 53 LBL 00
 54 10
 55 +
 56 X!=Y?
 57 GTO 00
 58 3525
 59 XEQ 99
 60 LBL 11
 61 557
 62 GTO 09
 63 LBL 00
 64 RDN
 65 15
 66 X!=Y?
 67 GTO 00
 68 5527
 69 GTO 08
 70 LBL 00
 71 565
 72 GTO 08
 73 LBL 14
 74 325
 75 XEQ 99
 76 18
 77 X!=Y?
 78 GTO 00
 79 -336
 80 GTO 10
 81 LBL 00
 82 10
 83 +
 84 X!=Y?
 85 GTO 00
 86 346
 87 GTO 10
 88 LBL 00
 89 10
 90 +
 91 X!=Y?
 92 GTO 11
 93 336
 94 XEQ 99
 95 28
 96 X!=Y?
 97 GTO 11
 98 527
 99 GTO 09
100 LBL 13
101 557
102 XEQ 07
103 FRC
104 .8
105 X=Y?
106 GTO 12
107 154
108 XEQ 99
109 66
110 X!=Y?
111 GTO 00
112 3534
113 XEQ 99
114 LBL 15
115 537
116 GTO 09
117 LBL 00
118 10
119 +
120 X!=Y?
121 GTO 00
122 567
123 GTO 08
124 LBL 00
125 RDN
126 X>0?
127 GTO 00
128 353
129 GTO 09
130 LBL 00
131 5627
132 GTO 08
133 LBL 12
134 1523
135 XEQ 99
136 68
137 X!=Y?
138 GTO 00
139 322
140 GTO 10
141 LBL 00
142 10
143 +
144 X!=Y?
145 GTO 00
146 -334
147 XEQ 99
148 68
149 X=Y?
150 GTO 15
151 322
152 GTO 09
153 LBL 00
154 -322
155 LBL 10
156 XEQ 99
157 527
158 GTO 09
159 LBL 07
160 XEQ 99
161 10
162 /
163 RTN
164 LBL 99
165 XEQ 98
166 AVIEW
167 STOP
168 RTN
169 LBL 98
170 ISG 01
171 CLA
172 " "
173 RCL d
174 FIX 00
175 CF 29
176 ARCL 01
177 >". "
178 X<>Y
179 ENTER
180 ABS
181 ENTER
182 LOG
183 INT
184 10^X
185 /
186 XEQ IND X
187 >"-"
188 FRC
189 10
190 *
191 XEQ IND X
192 FRC
193 10
194 *
195 FRC
196 X=0?
197 GTO 00
198 X<> L
199 XEQ IND X
200 FRC
201 10
202 *
203 ABS
204 LBL 00
```

```
205 ARCL L
206 RDN
207 X<0?
208 >" +"
209 X<>Y
210 STO d
211 RDN
212 RTN
213 LBL 01
214 >"P"
215 RTN
216 LBL 02
217 >"N"
218 RTN
219 LBL 03
220 >"B"
221 RTN
222 LBL 04
223 >"R"
224 RTN
225 LBL 05
226 >"Q"
227 RTN
228 LBL 06
229 >"K"
230 END
```

Game of Chuck–a–luck (LBL "CHK")

History: This is a modification of a program written by John Rausch that appeared in V2N3P33 of 65 Notes.

Object: Pick a lucky number and see if it shows up in a roll of three dice.

Instructions:

1) XEQ ALPHA CHK ALPHA
2) At the SEED? prompt, enter a decimal seed between 0 and 1 and press R/S.
3) Place a bet by entering an amount and pressing B.
4) The player then selects a number from 1 to 6 and presses A.
5) HP then rolls three dice.
6) The player is paid off 1–to–1 if the number selected appears on one of the dice, 2–to–1 if it appears on two of the dice and 3–to–1 if it appears on all three dice.
7) The player may continue by selecting another number and pressing A.
8) To see the total winnings at any time, press C.

Example game:

	See	Press	
1)		XEQ ALPHA CHK ALPHA	
2)	SEED?	0.123456 R/S	
3)	BANK=$0.00	5 B	
4)	BET=$5.00	4 A	
5)	5		
6)	55		
7)	556		(The number 4 did not occur)
8)	PAY $5.00		(So we owe $5)
9)	BANK=$–5.00	1 A	
10)	1		
11)	11		
12)	115		(The number 2 occurred twice!)
13)	WIN $10.00		(So we win $10 this time)
14)	BANK=$5.00		(Continue to play if you like!)

Specifics:

1) Program is 124 bytes long.
2) Program is XROM 23,06.
3) Calls XROM "S" – XROM 23,25 (SEED? prompt) and XROM "R" – XROM 23,26 (Random Number Generator).
4) Uses registers 00 – 04. SIZE 005 required.
    - 00 – Random number seed
    - 01 – Amount of bet
    - 02 – Count of times your number appeared
    - 03 – Number you chose
    - 04 – Amount in bank
5) Labels used:
    - 01 – Generates random numbers. Calls XROM "R" (23,26).
    - A – Play a round
    - B – Stores amount of bet. This is reused unless specifically changed.
    - C – Displays amount in bank
    - D – Resets bank to $0
6) Flags used:
    - Flag 27 is set to allow use of user–defined keys.
    - Flag 29 is cleared for formatted output
7) Display mode: FIX 00 and FIX 02 are set for output. The existing display mode is not retained.

Program listing:

```
 01 LBL "CHK"
 02 XROM “S”
 03 SF 27
 04 CF 29
 05 LBL D
 06 CLX
 07 STO 04
 08 LBL C
 09 FIX 02
 10 "BANK=$"
 11 ARCL 04
 12 PROMPT
 13 LBL B
 14 ABS
 15 STO 01
 16 FIX 02
 17 "BET=$"
 18 ARCL 01
 19 PROMPT
 20 LBL A
 21 INT
 22 STO 03
 23 LN
 24 6
 25 LASTX
 26 -
 27 SQRT
 28 CLST
 29 STO 02
 30 XEQ 01
 31 XEQ 01
 32 XEQ 01
 33 PSE
 34 RCL 02
 35 X=0?
 36 -1
 37 RCL 01
 38 *
 39 ST+ 04
 40 FIX 02
 41 "WIN $"
 42 X<0?
 43 "PAY $"
 44 ABS
 45 ARCL X
 46 AVIEW
 47 PSE
 48 PSE
 49 GTO C
 50 LBL 01
 51 10
 52 *
 53 XROM “R”
 54 6
 55 *
 56 1
 57 +
 58 INT
 59 RCL 03
 60 X=Y?
 61 ISG 02
 62 CLA
 63 RDN
 64 +
 65 FIX 00
 66 VIEW X
 67 END
```

Game of Golf (LBL "GOLF")

History: This is a modification of a program written by Jim Butterfield for the HP 67 that appeared in V4N6P44 of the PPC Journal.

Object: Choose your club and get the ball in the hole in as few shots as possible.

Instructions:

1) XEQ ALPHA GOLF ALPHA
2) At the SEED? prompt, enter a decimal seed between 0 and 1 and press R/S.
3) To tee up on each new hole, press A. See Yards to the green displayed.
   a) For wood shots, enter the wood club number (1–4) and press B.
   b) For iron shots, enter the iron number (1–9) and press C.
   c) For wedge shots, enter 1 for a regular wedge, 2 for a chip shot, and 3 for a pitch and run and press D.
   d) If the display shows the distance to the hole in FT (Feet), then try putting in the next step.
   e) For putting, enter the putt strength (1–15) and press E. The display will show the ball rolling to the hole. If the display ends up showing a 0, you sank the putt. If the value changes from negative to positive, you putted too hard and the ball went past the hole. Repeat this step until you sink the ball.
4) When you sink the putt, the display shows the number of shots.
5) For a new hole, go back to step 3.

Example game:

	See	Press	
1)		XEQ ALPHA GOLF ALPHA	
2)	SEED?	0.123456 R/S	
3)	397 YARDS	1 B	(Hit a 1 wood club – a driver)
4)	162 YARDS	4 C	(Hit a 4 iron)
5)	12 FT	6 E	(Putt the ball with medium strength)
6)	–8		(Watch the ball roll toward the cup)
7)	–6		
8)	–4		
9)	–2		
10)	–1		
11)	–1		
12)	–1		
13)	1 FT	1 E	(Just tap the ball this time)
14)	0		
15)	0		
16)	0 FT		
17)	4 SHOTS		(Pretty good!)
18)	4 SHOTS	A	
19)	418 YARDS	1 B	(Hit a 1 wood club)
20)	166 YARDS	4 B	(Hit a 4 wood club)
21)	5 FT	3 E	(Putt)
22)	–3		
23)	–1		
24)	–1		
25)	0		
26)	0 FT		
27)	4 SHOTS		(Pretty good!)

Specifics:

1) Program is 174 bytes long.
2) Program is XROM 23,07.
3) Calls XROM "S" – XROM 23,25 (SEED? prompt) and XROM "R" – XROM 23,26 (Random Number Generator).
4) Uses registers 00 – 04. SIZE 005 required.
   - 00 – Random number seed
   - 01 – Distance to hole
   - 02 – Modified user input at E, loop counter
   - 03 – User input at B, C or D
   - 04 – Shot counter (initialized to –1)
5) Labels used:
   - 01 – Used
   - 02 – Putter loop
   - 03 – Displays distance to hole in yards or feet

04 – Reads and checks user input at B, C, or D
A – Starts a new game
B – Hit a wood club. Valid inputs are 1–4.
C – Hit an iron. Valid inputs are 1–9.
D – Hit a wedge. Input 1 for a regular wedge, 2 for a chip shot, and 3 for a pitch and run.
E – Use the putter. Inputs are 1–15. A value of 1 is a light tap, 15 is a very hard putt.

6) Flags used:
   Flag 27 is set to allow use of user–defined keys
   Flag 29 is used for formatted output
7) Display mode: FIX 00 and FIX 04 are set for output. The existing display mode is not retained.

Program listing:

```
 01 LBL"GOLF"
 02 XROM “S”
 03 SF 27
 04 LBL A
 05 XROM “R”
 06 465
 07 *
 08 85
 09 +
 10 STO 01
 11 -1
 12 STO 04
 13 GTO 03
 14 LBL B
 15 45
 16 ENTER
 17 3.2
 18 ENTER
 19 4
 20 GTO 04
 21 LBL C
 22 41.8
 23 ENTER
 24 2.5
 25 ENTER
 26 9
 27 GTO 04
 28 LBL D
 29 18.9
 30 ENTER
 31 1.9
 32 ENTER
 33 3
 34 LBL 04
 35 R^
 36 X>Y?
 37 GTO 03
 38 X<>Y
 39 RDN
 40 STO 03
 41 RDN
 42 XROM “R”
 43 RCL 03
 44 +
 45 *
 46 -
 47 RCL 01
 48 LN
 49 *
 50 ST- 01
 51 LBL 03
 52 ISG 04
 53 CLA
 54 40
 55 RCL 01
 56 ABS
 57 INT
 58 X<=Y?
 59 CHS
 60 STO 01
 61 FIX 00
 62 CF 29
 63 ABS
 64 " "
 65 ARCL X
 66 LASTX
 67 X<=0?
 68 >" FT"
 69 X>0?
 70 >" YARDS"
 71 X!=0?
 72 PROMPT
 73 AVIEW
 74 PSE
 75 " "
 76 ARCL 04
 77 >" SHOTS"
 78 FIX 04
 79 SF 29
 80 PROMPT
 81 LBL E
 82 XROM “R”
 83 +
 84 STO 02
 85 GTO 01
 86 LBL 02
 87 2
 88 /
 89 RCL 01
 90 +
 91 STO 01
 92 INT
 93 PSE
 94 1
 95 ST- 02
 96 RCL 02
 97 LBL 01
 98 X>0?
 99 GTO 02
100 GTO 03
101 END
```

Game of Jive Turkey (LBL "JT")

History: This is a modification of an HP 67 program that appeared in V5N1P10 of the PPC Journal. The idea behind the game of Jive Turkey has a long history. Many early games (and even one in the TI 58/59 master library) were of the "High/Low" variety where a number was to be guessed. Either luck or the application of a binary search algorithm made these guessing games less than entertaining. Someone came up with the idea of having the calculator LIE to the user as to whether the entered guess was high or low. Users never knew when the response was real or a lie. Hence, Jive Turkey.

Object: Guess the secret number between 0 and 99 in response to the calculator's honest (?) answers. This certainly gives a new twist to the old High/Low game. It can be very difficult to win.

Instructions:

1) XEQ ALPHA JT ALPHA
2) At the SEED? prompt, enter a decimal seed between 0 and 1 and press R/S.
3) Calculator will ask for the probability of an honest answer (1–100%). Enter 90% as 90 R/S.
4) Start the game by pressing B. The calculator will generate a random number between 0 and 99.
5) Enter your guess for the secret number and press C.
6) Depending on the calculator's response, and whether you trust it, adjust guess up or down and press C again. Note: When the calculator displays HIGH you should lower your guess if the HP is telling the truth. If LOW is displayed you should raise your guess if you believe what the HP is indicating.
7) Continue steps 5–6 until you correctly guess the number.
8) Press D at any time to see how many guesses you have made so far. Return to step 5 to continue.

Example game:

	See	Press	
1)		XEQ ALPHA JT ALPHA	
2)	SEED?	0.123456 R/S	
3)	TRUTH%	90 R/S	Note: Have honesty set at 90%
4)	0.00	B	Start a new game.
5)	0 TRIES	50 C	Guess 50 to begin.
6)	LOW	75 C	Guess was low. Try 75.
7)	LOW	85 C	Guess was low. Try 85.
8)	HIGH	80 C	Guess was high. Try 80.
9)	HIGH	77 C	Guess was high. Try 77.
10)	HIGH	76 C	Guess was high. Try 76. If 75 was LOW and 77 is HIGH, then 76 should be the number!
11)	HIGH	70 C	Still high? HP must have lied somewhere. Try 70.
12)	HIGH	65 C	Guess was high. Try 65.
13)	LOW	67 C	Guess was low. Try 67.
14)	9 TRIES		So 67 was correct!

Specifics:

1) Program is 93 bytes long.
2) Program is XROM 23,08.
3) Calls XROM "S" – XROM 23,25 (SEED? prompt) and XROM "R" – XROM 23,26 (Random Number Generator).
4) Uses registers 00 – 03. SIZE 004 required.
    - 00 – Random number seed
    - 01 – Secret number
    - 02 – Number of guesses taken
    - 03 – Truth percentage
5) Labels used:
    - 01 – Random number routine. Calls XROM "R" (23,26).
    - A – Set honesty probability. Reset guess counter.
    - B – Start a new game
    - C – Check a guess for high or low
    - D – Displays number of guesses taken so far
6) Flags used:
    - 27 – Flag 27 is set to allow use of user–defined keys
    - 29 – Flag 29 is cleared for formatted output
7) Display mode: FIX 00 is set for output. The existing display mode is not retained.

Program listing:

```
01 LBL "JT"
02 XROM 23,25
03 SF 27
04 LBL A
05 "TRUTH%?"
06 PROMPT
07 STO 03
08 CLX
09 STO 02
10 RTN
11 LBL B
12 XEQ 01
13 STO 01
14 LBL D
15 FIX 00
16 CF 29
17 " "
18 ARCL 02
19 >" TRIES"
20 CLX
21 PROMPT
22 LBL C
23 ISG 02
24 CLA
25 RCL 01
26 X=Y?
27 GTO D
28 -
29 SIGN
30 XEQ 01
31 -
32 SIGN
33 *
34 " LOW"
35 X>0?
36 " HIGH"
37 PROMPT
38 LBL 01
39 RCL 03
40 XROM 23,26
41  E2
42 *
43 INT
44 END
```

Moon Lander game (LBL "ML")

History: Moon lander games go way back in the world of calculators. This game is modeled after the Moon Rocket Lander game from the HP 67 Standard Pac, program 14.

Object: Can you land on the moon without crashing before your fuel is exhausted?

Instructions:

1) XEQ ALPHA ML ALPHA
2) The display will pause to show your rate of descent and altitude in the form of V=–XXX A=YYYY, where XXX is your rate of descent and YYYY is your altitude.
3) The display will then show how much fuel you have left, displaying FUEL=ZZ, where ZZ are the units of fuel left.
4) Enter fuel to burn when the display pauses showing 0 during the countdown. You have 60 units to use.
5) If you try to burn more fuel than is available, you will "free fall to your doom" and your final velocity as you crash on the moon's surface will be displayed.
6) If final velocity is >= –3 down, the display will show CRASHED. If less than –3 down, LANDED will be displayed.
7) The calculator will then display Ve=–MM, where MM is your final velocity.

Example game:

	See	Press	
1)		XEQ ALPHA ML ALPHA	
2)	V=–50 A=500		Down at 50 per second, height is 500.
3)	FUEL=60		Note: 60 units of fuel left
4)	3 … 2 … 1 …		Note: these 3, 2, 1 are displayed one at a time during a pause
5)	0		Note: no entry. Let's coast a bit.
6)	V=–55 A=448		Down at 55 per second, height is 448. No fuel burn, +5 velocity.
7)	FUEL=60		Note: 60 units of fuel left
8)	3 … 2 … 1 …		
9)	0	5	Note: burn 5 units
10)	V=–50 A=395		Down at 50 per second, height is 395
11)	FUEL=55		Note: 55 units of fuel left
12)	3 … 2 … 1 …		
13)	0	7	Note: burn 7 units
14)	V=–41 A=350		Down at 41 per second, height is 350
15)	FUEL=48		Note: 48 units of fuel left
16)	3 … 2 … 1 …		
17)	0	2	Note: burn 2 units
18)	V=–42 A=308		Down at 42 per second, height is 308
19)	FUEL=46		Note: 46 units of fuel left
20)	3 … 2 … 1 …		
21)	0	5	Note: burn 5 units
22)	V=–37 A=269		Down at 37 per second, height is 269
23)	FUEL=41		Note: 41 units of fuel left
24)	3 … 2 … 1 …		
25)	0		Note: no entry. Let's coast a bit.
26)	V=–42 A=229		Down at 42 per second, height is 229
27)	FUEL=41		
28)	3 … 2 … 1 …		
29)	0	5	Note: burn 5 units
30)	V=–37 A=190		Down at 37 per second, height is 190
31)	FUEL=36		Note: 41 units of fuel left
32)	3 … 2 … 1 …		
33)	0	?	Can you continue from here and land safely?

Specifics:

1) Program is 150 bytes long.
2) Program is XROM 23,09.
3) Uses registers 01 – 03. SIZE 004 required.
   - 01 – Altitude
   - 02 – Rate of descent
   - 03 – Amount of fuel
4) Labels used:
   - 01 – Out of fuel, compute crash velocity
   - 02 – Calculates the crash velocity after an attempt to burn more fuel than is available
   - 03 – Displays the final velocity Ve as the craft lands or crashes

5) Flags used: Flag 29 is cleared at the start of the program to format output and set again at the end.
6) Display mode: FIX 00 is sued throughout the program. FIX 04 is set at the end of the program.

Program listing:

```
01 LBL "ML"
02 FIX 00
03 CF 29
04 500
05 STO 01
06 -50
07 STO 02
08 60
09 STO 03
10 LBL 01
11 "V="
12 RCL 02
13 X>0?
14 >"+"
15 ARCL 02
16 >" A="
17 ARCL 01
18 AVIEW
19 PSE
20 PSE
21 "FUEL="
22 ARCL 03
23 AVIEW
24 PSE
25 PSE
26 CLD
27 3
28 PSE
29 2
30 PSE
31 1
32 PSE
33 0
34 PSE
35 RCL 03
36 X<>Y
37 X>Y?
38 GTO 02
39 ST- 03
40 RCL 02
41 X<>Y
42 2
43 *
44 5
45 -
46 ST+ 02
47 2
48 /
49 +
50 ST+ 01
51 RCL 01
52 INT
53 X>0?
54 GTO 01
55 RCL 02
56 GTO 03
57 LBL 02
58 RCL 03
59 2.5
60 -
61 ST+ 01
62 2
63 *
64 ST+ 02
65 RCL 01
66 10
67 *
68 RCL 02
69 X^2
70 +
71 SQRT
72 CHS
73 LBL 03
74 "LANDED"
75 -3
76 X<>Y
77 X<=Y?
78 "CRASHED"
79 AVIEW
80 PSE
81 PSE
82 "Ve="
83 RND
84 ARCL X
85 LASTX
86 FIX 04
87 SF 29
88 AVIEW
89 END
```

One Arm Bandit (LBL "OAB")

History: This program is a modified version of an original program written by Roelf Backus that appeared in the V4N9P13 issue of the PPC Journal. It has been substantially modified here.

Object: Pull the levers to get a three–digit number that gives payouts of free games. The winning combinations and payouts are shown below. The symbol x22 means any number in the first digit followed by 22.

Combination	Payout	Combination	Payout	Combination	Payout
123	3	x11	1	111	2
234	6	x22	2	222	4
345	9	x33	3	333	6
456	12	x44	4	444	8
567	15	x55	5	555	10
678	18	x66	6	666	12
789	21	x77	7	777	14
		x88	8	888	16
		x99	9	999	18

Instructions:

1) XEQ ALPHA OAB ALPHA
2) At the SEED? prompt, enter a decimal seed between 0 and 1 and press R/S.
3) Enter your initial bank deposit and press D. Four games will be added for each $1 deposited.
4) Pull the slot machine lever by pressing E.
5) The three digits generated are displayed.
6) Hold any digits you want by pressing A to hold the first digit, B to hold the second digit, C to hold the third digit, or shift c to hold all of the digits.
7) If you hold a digit and change your mind, you can clear all holds by pressing shift b.
8) Once you have held the digits you want, pull the slot lever again by pressing E.
9) This second pull is considered the end of a game when the numbers are displayed.
10) Note: If you get a big winner on the first pull, press shift c to hold all digits and then E to collect!

Example game:

	See	Press	
1)		XEQ ALPHA OAB ALPHA	
2)	SEED?	0.123456 R/S	
3)	0 GAMES	10 D	
4)	40 GAMES		(Note: You have 40 games remaining)
5)	40 GAMES	E	(Pull the lever)
6)	798	C	(Hold third digit. See flag 3 set.)
7)	798	shift b	(Oops. Meant to hold first digit. Clear the holds!)
8)	798	A	(Hold first digit. See flag 1 set.)
9)	798	E	(Pull the lever)
10)	722		(This is a winner! Pays 2X the pull.)
11)	WIN 2		(Roll won two games, so a net of +1)
12)	41 GAMES		(Still 41 games on credit)
13)	41 GAMES	E	(Let's play again. Pull the lever.)
14)	273	C	(Hold third digit. See flag 3 set.)
15)	273	E	(Pull the lever)
16)	263		(Not a winner this time)
17)	WIN 0		
18)	40 GAMES		(There are 40 games remaining – continue to play!)

Specifics:

1) Program is 196 bytes long.
2) Program is XROM 23,10.
3) Calls XROM "S" – XROM 23,25 (SEED? prompt) and XROM "R" – XROM 23,26 (Random Number Generator).
4) Uses registers 00 – 05. SIZE 006 required.
    - 00 – Random number seed
    - 01 – Number of remaining games
    - 02 – Complete three–digit number
    - 03 – First digit
    - 04 – Second digit
    - 05 – Third digit

5) Labels used:
   - 00 – Check if number has three consecutive digits, 123, 234, 345, etc.
   - 01 – Update number of games
   - 02 – Replaces X with a random number from 1 to 9. Calls XROM “R” (23,26).
   - 03 – Show remaining number of games
   - A – Hold first digit
   - B – Hold second digit
   - C – Hold third digit
   - D – Put your money in the machine (make your bet, insert a coin or coins)
   - E – Pull the slot machine lever
   - b – Clears all holds on the digits
   - c – Hold all digits
   - e – Starts a new game and clear the bank / pot
6) Flags used:
   - 01 – Hold first digit
   - 02 – Hold second digit
   - 03 – Hold third digit
   - 05 – Hold at least one digit
   - 27 – Flag 27 is set to allow use of user–defined keys
   - 29 – Flag 29 is cleared for formatted output
7) Display mode: FIX 00 is set for output. The existing display mode is not retained.

Program listing:

```
01 LBL "OAB"
02 XROM “S”
03 SF 27
04 LBL e
05 CLX
06 STO 01
07 XEQ b
08 GTO 03
09 LBL b
10 CF 01
11 CF 02
12 CF 03
13 CF 05
14 RTN
15 LBL E
16 RCL 01
17 X<=0?
18 GTO 03
19 1
20 FC? 05
21 ST- 01
22 RCL 03
23 FC?C 01
24 XEQ 02
25 STO 03
26  E2
27 *
28 RCL 04
29 FC?C 02
30 XEQ 02
31 STO 04
32 10
33 *
34 +
35 RCL 05
36 FC?C 03
37 XEQ 02
38 STO 05
39 +
40 STO 02
41 FIX 00
42 CF 29
43 VIEW 02
44 FC?C 05
45 RTN
46 PSE
47 RCL 05
48 RCL 04
49 X!=Y?
50 GTO 00
51 RCL 03
52 X<>Y
53 X=Y?
54 +
55 GTO 01
56 LBL 00
57 RCL 02
58 123
59 -
60 111
61 /
62 FRC
63 X!=0?
64 GTO 01
65 LASTX
66 1
67 +
68 3
69 *
70 LBL 01
71 INT
72 ST+ 01
73 FIX 00
74 CF 29
75 "WIN "
76 ARCL X
77 AVIEW
78 PSE
79 GTO 03
80 LBL D
81 4
82 *
83 INT
84 ST+ 01
85 LBL 03
86 FIX 00
87 CF 29
88 " "
89 ARCL 01
90 >" GAMES"
91 PROMPT
92 LBL A
93 SF 01
94 SF 05
95 RTN
96 LBL B
97 SF 02
98 SF 05
99 RTN
100 LBL c
101 SF 01
102 SF 02
103 LBL C
104 SF 03
105 SF 05
106 RTN
107 LBL 02
108 RDN
109 XROM “R”
110 9
111 *
112 INT
113 1
114 +
115 END
```

Space War (LBL "SPW")

History: This is a heavily modified version of the Space War game from the HP 67 Games Pac, program 6. Robert Meyer modified this from a two–card original program to the unified basis.

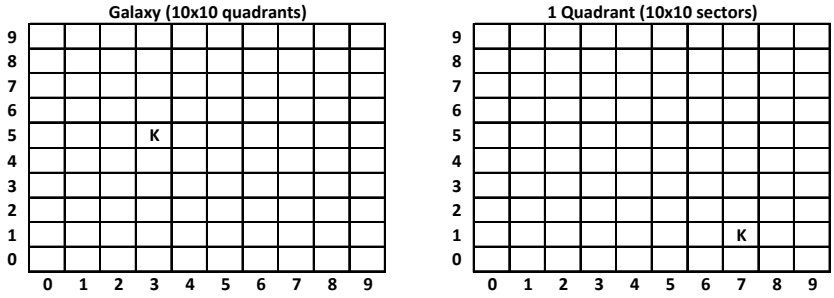
Object: You command the Nuclear–Powered Reconnoiterer Kittyhawk (KH) in a galaxy of 10 by 10 quadrants, each of which contains 10 by 10 sectors. Somewhere in the galaxy are 3 Alglogs (the bad guys) and 1 Base (the good guy). Your mission is to find and kill the Alglogs before your time (18 stardays) and energy (1000 units) are gone.

Instructions:

1) XEQ ALPHA SPW ALPHA
2) At the SEED? prompt, enter a decimal seed between 0 and 1 and press R/S.
3) The remaining information below will help explain how to play.

Positions

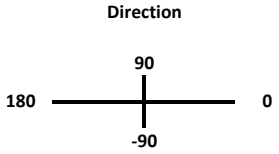
Positions are displayed as $Q_yQ_x.S_yS_x$ where Q_yQ_x specifies the quadrant and S_yS_x specifies the sector.



In the example above, the KH position is 53.17

Movement

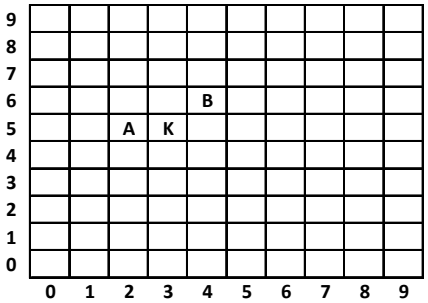
Movement is specified by an angle θ and distance r to be covered. Orientation of the angles is shown below and the angles are specified in degrees. Distance is specified in terms of quadrants. To move exactly one quadrant, specify an r of 1. To move from Q53 to Q52, enter θ = 180° and r = 1. To move from Q53 to Q64 would require θ = 45° and $r = \sqrt{2}$.



Each movement of the Kittyhawk consumes 1 starday. If a move is attempted when no stardays remain, "OUT OF DAYS" will be displayed, indicating the mission has failed.

Long Range Scan

A long range scan displays the Alglogs and Bases in the nine quadrants surrounding the Kittyhawk. These are displayed in the form QQ.ab4ab4ab where QQ is the middle quadrant displayed, a is 1 if an Alglog is present or zero otherwise, b is a 1 if a Base is present or zero otherwise, and the 4's are used as separators in the display.



In the example above the Kittyhawk is in Q53, and Alglog is in Q52 and a Base is in Q64. The output of the scan would be:

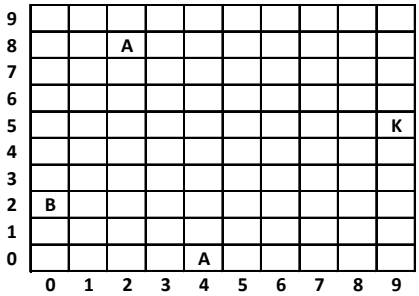
```
63.00400401
53.10400400
43.00400400
```

The first line shows the contents of Q62, Q63, Q64, second line is Q52, Q53, Q54, and the third line is Q42, Q43, and Q44.

Note: the contents of 9 quadrants are always shown. If the Kittyhawk is near or at the edge of the galaxy, some of this information may be meaningless.

Short Range Scan

A short–range scan displays the contents of one quadrant (10x10 sectors). The output is 10 lines of information, each line representing one row of the quadrant. Each line consists of 10 digits that represent the 10 sectors in the row. A "0" means that sector is unoccupied. A "3" marks the location of the Kittyhawk. A "4" represents and Alglog, and a "7" represents a Base. For the following quadrant:



The scan output would be:

```
0.000000000          (Note: Row 9)
0.040000000
0.000000000
0.000000000
0.000000003
0.000000000
0.000000000
7.000000000
0.000000000
0.000400000          (Note: Row 0)
```

Torpedoes

The Kittyhawk begins with 3 torpedoes, which can be fired within the same quadrant. If the torpedo passes within 1° of an Alglog, the Alglog is destroyed. To fire a torpedo, simply specify the angle in degrees (no need to specify a distance). If there are no torpedoes remaining, you will see "OUT OF TORP" displayed.

Phasers

Phasers fire a burst of energy equally in all directions and can destroy as many Alglogs as are within range (within the same quadrant as the Kittyhawk). The closer the Alglog is, the less energy is required to destroy it. A minimum of 105 units and a maximum of 275 units may be required to destroy an Alglog. To fire phasers, simply specify the amount of energy to be used (no need to specify a direction). If there are 3 Alglogs in the same quadrant, and you specify 275 units, then the Kittyhawk must have at least 575 units of energy to fire and live (275 to fire, and 3 hits of 100 units from each of the 3 Alglogs in the quadrant). If you run out of energy, you will see "OUT OF ENRGY" displayed, and this means your mission has failed.

Base/Docking

The Kittyhawk may dock at the Base by moving into a sector adjacent to that of the Base and then executing a DOCK maneuver. If the docking is successful, the Kittyhawk's supply of torpedoes and energy are replenished to their initial level: 3 torpedoes, 1000 units of energy. After docking, the display will show the number of stardays remaining, as well as the amount of energy and the number of torpedoes: e.g. "D:14 E:1000 T:3".

Win

"ALGLOGS: 0"

This will be displayed after firing a torpedo or phasers which destroys the last Alglog.

Loss

"OUT OF ENRGY"
This will be displayed after firing phasers or moving if the Kittyhawk energy drops to 0.

"OUT OF DAYS"
This will be displayed when you attempt to move and there are 0 stardays remaining.

START	XEQ ALPHA SPW ALPHA	Begin the game and display the coordinates of the Kittyhawk
SRSCAN	A	Short Range Scan shows the 10x10 sectors of the KH quadrant
LRSCAN	B	Long Range Scan shows the 3x3 quadrants around the KH
PHASER	n C	Fire a phaser using n strength (reduces energy by n)
TORPEDO	θ D	Fire a torpedo at angle θ (reduces torpedoes by 1)
MOVE	θ ENTER r E	Move the KH r distance (in quadrants) in the direction θ
STATUS	a	Display the remaining stardays, energy, and torpedoes
DOCK	d	Dock at the Base – must be adjacent to it

Example game:

	See	Press	
1)		XEQ ALPHA SPW ALPHA	
2)	SEED?	0.123456 R/S	
3)	15.30	B	We are in quadrant 15, sector 30. LRSCAN
4)	25.00400400		Note: nothing nearby. Quadrants shown are 24, 25, 26
	15.00400400		on top row, 14, 15, 16 on second row, and 4, 5, and 6
	5.00400401		on bottom row. Notice that Base in Q06 for later.
		76 ENTER 4 E	MOVE: see if we can find an Alglog
5)	56.20	B	LRSCAN: see what's around us
6)	66.00400410		Note: there is an Alglog (1) in the quadrant
	56.00400400		above and to our right
	46.00400400	45 ENTER 1.41 E	MOVE to Alglog quadrant to attack
7)	67.20	A	SRSCAN: find the Alglog sector
8)	0.000000000		
	0.000000000		
	0.000000000		
	0.000000000		
	0.000000000		
	0.000000000		
	0.000000000		
	3.000000400		KH (3) in sector S20, Alglog (4) in S27
	0.000000000		
	0.000000000		
		0 D	TORPEDO: Fire a torpedo at him! (0 degrees)
9)	ALGLOGS:2		Got him! (Started with 3 Alglogs, now 2)
		99 CHS ENTER 6 E	MOVE: go back to our Base in Q06
10)	6.31	A	SRSCAN: Find the sector of the Base
11)	0.000000000		
	0.000000000		
	0.000000000		
	0.000000000		
	0.000000000		
	0.000000000		
	0.300000000		Note: KH (3) is at sector S31
	0.000000000		and the Base (7) is at sector S08
	0.000000000		
	0.000000070	23 CHS ENTER .75 E	MOVE: get adjacent to the Base
12)	6.18	shift d	DOCK (we are 6.18, base is 6.08)
13)	D:14 E:1000 T:3		Shows stardays, energy and torpedoes
		116 ENTER 9 E	MOVE: let's try to find another Alglog
14)	82.29	B	LRSCAN: see what's around us
15)	92.00400400		
	82.00410400		Note: Alglog (1) in our quadrant!
	72.00400400	200 C	PHASER: Let's get him with phasers!
16)	ALGLOGS: 1		Got him! (1 Alglog left)

The rest of the mission will be left as an exercise for the cadet.

Specifics:

1) Program is 701 bytes long.
2) Program is XROM 23,11.
3) Calls XROM "S" – XROM 23,25 (SEED? prompt) and XROM "R" – XROM 23,26 (Random Number Generator).
4) Uses registers 00 – 25. SIZE 026 required.
   - 00 – Random number seed
   - 01 thru 05 – Temporary, multiple use
   - 06 – Energy remaining
   - 07 – Torpedoes remaining
   - 08 – Stardays remaining
   - 09 – Alglogs remaining
   - 10 thru 19 – Short range scan results
   - 20 – Alglog 1 position (QQ.SS quadrant.sector)
   - 21 – Alglog 2 position
   - 22 – Alglog 3 position
   - 23 – Base position
   - 24 – Kittyhawk position
   - 25 – Used for indirect addressing
5) Labels used:
   - 00 – Forward label used from LBL 03 and LBL 07
   - 01 – Test if Alglog hit by torpedo
   - 02 – Test if Alglog hit by phaser
   - 03 – Scan one line (3 quadrants)
   - 07 – Local loop in LBL E (MOVE) to test if QQ.SS is occupied
   - 09 – Compute ship positions
   - 10 – Local loop in LBL 50 for clearing registers
   - 11 – Local loop in LBL A (SRSCAN) to print SRSCAN registers 10–19
   - 12 – Test whether an object is in Kittyhawk's quadrant
   - 13 – Find angle and distance from Kittyhawk
   - 20 – Displays "OUT OF xxxx" message (xxxx in ALPHA)
   - 30 – Test if position (X) is occupied
   - 40 – Show Alglogs remaining
   - 50 – Clear registers by X (bbb.eee)
   - A – SRSCAN – Short range scan
   - B – LRSCAN – Long range scan
   - C – PHASER – Fire phaser
   - D – Torpedo – Fire torpedo
   - E – MOVE – Move the Kittyhawk
   - d – DOCK – Dock to the base
   - e – START – Initialize a new game
6) Flags used:
   - 05 – Position occupied?
   - 06 – Alglog hit?
   - 27 – Flag 27 is set to allow use of user–defined keys
   - 29 – Flag 29 is used for formatted output
   - 42,43 – Flags 42 and 43 are cleared by the DEG command on lines 130 and 308
7) Display mode: FIX 00, 01, 02, 08 and 09 are set for output. The existing display mode is not retained.

Program listing:

```
01 LBL "SPW"
02 XROM "S"
03 SF 27
04 LBL e
05 1.024
06 XEQ 50
07 20.024
08 STO 25
09 LBL 09
10 XROM "R"
11  E4
12 *
13 INT
14  E2
15 /
16 XEQ 30
17 FS?C 05
18 GTO 09
19 STO IND 25
20 ISG 25
21 GTO 09
22  E3
23 STO 06
24 3
25 STO 07
26 STO 09
27 18
28 STO 08
29 FIX 02
30 RCL 24
31 RTN
32 LBL 30
33 SF 05
34 RCL 20
35 X=Y?
36 RTN
37 RDN
38 RCL 21
39 X=Y?
40 RTN
41 RDN
42 RCL 22
43 X=Y?
44 RTN
45 RDN
46 RCL 23
47 X=Y?
48 RTN
49 RDN
50 CF 05
51 RTN
```

```
 52 LBL B
 53 FIX 08
 54 4
 55 STO 02
 56 999
 57 ST/ 02
 58  E2
 59 RCL 24
 60 INT
 61 STO 04
 62 10
 63 +
 64 X<Y?
 65 XEQ 03
 66 RCL 04
 67 XEQ 03
 68 RCL 04
 69 10
 70 -
 71 X<0?
 72 RTN
 73 LBL 03
 74 RCL 02
 75 STO 01
 76 RDN
 77 STO 03
 78 ST+ 01
 79 1
 80 -
 81 XEQ 00
 82  E3
 83 /
 84 ST+ 01
 85 RCL 03
 86 XEQ 00
 87  E5
 88 /
 89 ST+ 01
 90 RCL 03
 91 1
 92 +
 93 XEQ 00
 94  E8
 95 /
 96 ST+ 01
 97 VIEW 01
 98 PSE
 99 RTN
100 LBL 00
101 0
102 STO 05
103 RDN
104 RCL 20
105 INT
106 X=Y?
107 ISG 05
108 CLX
109 RCL 21
110 INT
111 X=Y?
112 ISG 05
113 CLX
114 RCL 22
115 INT
116 X=Y?
117 ISG 05
118 CLX
119 10
120 ST* 05
121 RDN
122 RCL 23
123 INT
124 X=Y?
125 ISG 05
126 CLX
127 RCL 05
128 RTN
129 LBL E
130 DEG
131 P-R
132 FIX 01
133 RND
134 STO 04
135 X<>Y
136 RND
137 FIX 02
138 RCL 24
139 10
140 /
141 STO 03
142 INT
143 RCL 24
144 10
145 *
146 STO 02
147 INT
148 10
149 /
150 FRC
151 +
152 +
153 FRC
154 LASTX
155 INT
156 10
157 *
158 +
159 RCL 02
160 FRC
161 RCL 03
162 FRC
163 10
164 *
165 INT
166 +
167 RCL 04
168 +
169 INT
170 LASTX
171 FRC
172 10
173 /
174 +
175 LBL 07
176 +
177 XEQ 30
178 FS?C 05
179 GTO 00
180 STO 24
181 "DAYS"
182 1
183 ST- 08
184 RCL 08
185 X<0?
186 GTO 20
187 RCL 24
188 RTN
189 LBL 00
190 .1
191 GTO 07
192 LBL A
193 FIX 09
194 10.019
195 XEQ 50
196 3
197 RCL 24
198 XEQ 12
199 4
200 RCL 20
201 XEQ 12
202 4
203 RCL 21
204 XEQ 12
205 4
206 RCL 22
207 XEQ 12
208 7
209 RCL 23
210 XEQ 12
211 19.009
212 STO 25
213 LBL 11
214 CLD
215 VIEW IND 25
216 PSE
217 PSE
218 DSE 25
219 GTO 11
220 RTN
221 LBL 12
222 ENTER
223 INT
224 RCL 24
225 INT
226 X!=Y?
227 RTN
228 RDN
229 RDN
230 FRC
231 10
232 *
233 INT
234 STO 25
235 RDN
236 LASTX
237 FRC
238 10
239 ST+ 25
240 *
241 10^X
242 /
243 ST+ IND 25
244 RTN
245 LBL D
246 CF 06
247 STO 04
248 "TORP"
249 RCL 07
250 1
251 X>Y?
252 GTO 20
253 -
254 STO 07
255 19
256 STO 25
257 XEQ 01
258 XEQ 01
```

```
259 XEQ 01
260 CF 06
261 GTO 40
262 LBL 01
263 ISG 25
264 CLX
265 FS? 06
266 RTN
267 RCL IND 25
268 INT
269 RCL 24
270 INT
271 X!=Y?
272 RTN
273 XEQ 13
274 RDN
275 RCL 04
276 -
277 ABS
278 1
279 X<=Y?
280 RTN
281 SF 06
282 CHS
283 STO IND 25
284 ST+ 09
285 RTN
286 LBL 13
287 RCL IND 25
288 FRC
289 10
290 *
291 STO 01
292 INT
293 RCL 24
294 FRC
295 10
296 *
297 INT
298 -
299 RCL 01
300 FRC
301 RCL 24
302 10
303 *
304 FRC
305 -
306 10
307 *
308 DEG
309 R-P
310 RTN
311 LBL C
312 STO 04
313 ST- 06
314 19
315 STO 25
316 "ENRGY"
317 XEQ 02
318 X<0?
319 GTO 20
320 XEQ 02
321 X<0?
322 GTO 20
323 XEQ 02
324 X<0?
325 GTO 20
326 LBL 40
327 RCL 09
328 FIX 00
329 CF 29
330 "ALGLOGS: "
331 ARCL X
332 PROMPT
333 LBL 02
334 RCL 06
335 X<0?
336 RTN
337 ISG 25
338 ENTER
339 RCL IND 25
340 INT
341 RCL 24
342 INT
343 X!=Y?
344 RTN
345  E2
346 ST- 06
347 XEQ 13
348 X^2
349  E2
350 +
351 RCL 04
352 X<=Y?
353 RTN
354 -1
355 STO IND 25
356 ST+ 09
357 CLX
358 RTN
359 LBL 20
360 ASTO X
361 "OUT OF "
362 ARCL X
363 CLST
364 PROMPT
365 LBL d
366 RCL 23
367 INT
368 RCL 24
369 INT
370 X!=Y?
371 GTO a
372 RCL 23
373 FRC
374 10
375 *
376 STO 04
377 INT
378 RCL 24
379 FRC
380 10
381 *
382 STO 01
383 INT
384 -
385 ABS
386 2
387 X<=Y?
388 GTO a
389 RCL 04
390 FRC
391 10
392 *
393 RCL 01
394 FRC
395 10
396 *
397 -
398 ABS
399 2
400 X<=Y?
401 GTO a
402  E3
403 STO 06
404 3
405 STO 07
406 LBL a
407 FIX 00
408 CF 29
409 "D:"
410 ARCL 08
411 >" E:"
412 ARCL 06
413 >" T:"
414 ARCL 07
415 AVIEW
416 FIX 02
417 SF 29
418 RCL 24
419 RTN
420 LBL 50
421 0
422 LBL 10
423 STO IND Y
424 ISG Y
425 GTO 10
426 END
```

2D Star Trek (LBL "ST")

History: Early computer games almost always included a version of Star Trek, even on a Sigma 7 mainframe! This program is a modified version of an HP 67 Users' Library Program 03398D written by John Nelson.

Object: You are the commander of the Enterprise existing in a galaxy represented by a two–dimensional grid (100x100). Your goal is to seek out and destroy 3 Klingon enemy ships before you run out of energy (20,000 units).

Instructions:

1) XEQ ALPHA ST ALPHA
2) At the SEED? prompt, enter a decimal seed between 0 and 1 and press R/S.
3) The remaining information below will help explain how to play.

Positions

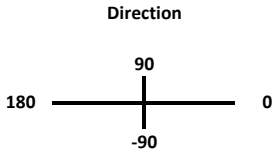
Positions are displayed as XX.YY.

	Galaxy (100x100 quadrants)									
99										
98										
97										
...										
5										
4							E			
3										
2										
1										
0										
	0	1	2	...	66	67	68	...	98	99

In the example above, the Enterprise position is 68.04.

Movement

Movement is specified by an angle θ and Warp w value. Orientation of the angles is shown below and the angles are specified in degrees. Distance is computed in terms of quadrants from the warp value using the formula quadrants = warp * 10. For example, to move 40 quadrants to the west from 68.04 to 28.04, enter $\theta = 180°$ and $w = 4$. To move from 68.04 to 66.02 would require $\theta = -135°$ and $w = (1/10 * \sqrt{8})$, or 0.2828.



Maximum warp is 8 * ((100–Damage)/100). So, if the Enterprise has 0 damage, max warp is 8. If the Enterprise is 50% damaged, max warp is 4.

Energy used is (warp * 10)^2. So, warp 4 uses 1600 unites of the Enterprise total energy.

Note: There is no coordinate checking. You can "warp" off the grid, which will have unexpected results.

Scan

A sensor scan is used to detect enemy Klingon ships. The range of the scan is 25 quadrants (in all directions). If one or more Klingon ships are located by the scan, each will fire on the Enterprise, and at the end of the scan, the total damage to the Enterprise will be displayed.

Each enemy ship within sensor range is displayed as "N: dXX ∠YY" where N is 1, 2 or 3, and XX is the distance in quadrant units to the ship, and YY is the angle (180 to 0 to –180).
Note: it is possible to be in the same quadrant as a Klingon: the display will show "N: d0 ∠0".

Shields

As commander, you divert some of your total energy to the shields. An enemy attack can at most be 100, so it is a good idea to keep your shields above 100, particularly during an attack. To add energy to shields, just enter the amount you want to deduct from total energy that will be added to shields.

Attack

To attack a Klingon, the distance between you two must be less than half the sensor range, so 12.5 quadrant units. Thus, it is possible to see a Klingon in the scan, but not be able to attack it. To attack a ship within range, you first specify a ship # (you can only attack one Klingon at a time, even if more than one is within range, and only the ship you attack will attack you back). The Klingon will fire on you, and your remaining shields will be displayed, followed by a prompt for POWER? Enter the phaser power (1–9) you want to fire, and press R/S. The energy used is phaser power * distance^2. The power of the enemy attack is (100 – enemy damage) * random#. This is the amount of hit on your shields. The damage to the Enterprise is ¼ the power of the enemy attack (if shields hold), or 4 * (power of enemy attack – shields) if shields do not hold.

Ships

Ships (registers R01, R02, R03, and R04) are represented as XX.YYDDDddd where XX is the x–coordinate, YY is the y–coordinate, and DDD.ddd is the cumulative damage of the ship (a damage of 100 or greater means the ship has been destroyed, which then also makes the entire value become negative (–XX.YYDDDddd).

Win

A win is when you have destroyed all 3 Klingons and have not run out of energy.

Loss

"4: DESTROYED"
This will be destroyed when total Enterprise damage reaches 100. This can happen during Scan or Attack.

"ENERGY=0"
This will be displayed when the Enterprise runs out of energy. This can happen during Scan, Warp, Shields, Attack, or Repair. *Note: your position will then be displayed as a negative number.*

Commands

Begin	XEQ ALPHA ST ALPHA	Begin the game and display the coordinates of the Enterprise
START	A	Start a new game and display the coordinates of the Enterprise
SCAN	B	Sensor Scan shows distance/angle to Klingon ships in range
WARP	*θ* ENTER *w* C	Move the Enterprise w * 10 quadrants in the direction *θ*
SHIELDS	*n* D	Divert n units of energy to the shields
ATTACK	*n* E	Attack ship n
DAMAGE	ship# SHIFT b	Display the total damage of requested ship
REPAIR	*energy* SHIFT c	Reduces total energy, and reduces Enterprise damage (repairs)
Shields	RCL 06	Displays the remaining shield amount
Energy	RCL 08	Displays the remaining total energy of the Enterprise

Example game:

	See	Press	
1)		XEQ ALPHA ST ALPHA	
2)	SEED?	0.123456 R/S	
3)	POS=67.27	B	SCAN
4)	POS=67.27		No Klingons found, shows Enterprise position
		75 ENTER 5 C	Let's move northeast (75°) about 50 units
5)	POS=79.75	B	And scan for enemy ships
6)	3: d3 ∡–18		Klingon Ship #3 is 3 units away at –18°
	4: DMG=2.18		Enterprise (ship 4) damage is 2.18
	POS=79.75	150 D	Divert energy to the shields
7)	SHLDS=150.00	3 E	Let's attack ship #3
8)	SHLDS=78.20		Enemy hit our shields big leaving 78.20 (ouch!)
	POWER?	9 R/S	Let's give them all we got
9)	3: DMG=3.52		Small hit on Klingon #3 of 3.52 units
	4: DMG=17.95		Moderate hit on Enterprise of 17.95 units
	POS=79.75	3 E	Let's attack again
10)	SHLDS=52.21		Hit our shields leaving 52.21 remaining
	POWER?	9 R/S	Full power (Note: simply pressing R/S also enters 9)
11)	3: DMG=29.09		Good hit on Klingon #3 of 29.09 units (yay!)
	4: DMG=6.50		Small hit on Enterprise of 6.50 units
	POS=79.75	70 D	Let's bring our shields back above 100
12)	SHLDS=122.21	3 E	And then attack again
13)	SHLDS=68.21		Enemy hit our shields big leaving 68.21(ouch!)
	POWER?	9 R/S	Full power

14)	3: DMG=2.92		Small hit on Klingon #3 of 2.92
	4: DMG=13.50		Hit on Enterprise of 13.50
	POS=79.75	4 SHIFT b	Check our damage
15)	4: DMG=40.12	3 SHIFT b	Check Klingon #3 damage
16)	3: DMG=35.52	40 SHIFT c	Since we are 40% destroyed, lets repair
17)	4: DMG=−40.00		
	POS=79.75	4 SHIFT b	Recheck our damage
18)	4: DMG=0.12	50 D	Bring our shields back above 100
19)	SHLDS=118.21	3 E	Now attack!
20)	SHLDS=75.91		Hit our shields leaving 75.91
	POWER?	9 R/S	Full power
21)	3: DMG=21.18		Hit on Klingon
	4: DMG=10.58		Hit on Enterprise
	POS=79.75	3 E	Attack again.
22)	SHLDS=46.80		Hit on our shields, 46.80 remaining
	POWER?	9 R/S	Full power
23)	3: DMG=23.50		Good hit on Klingon #3
	4: DMG=7.28		Small hit on Enterprise
	POS=79.75	3 E	Attack…
24)	SHLDS=46.32		Small hit on our shields
	POWER?	9 R/S	Full power
25)	3: DMG=53.88		Big hit on Klingon #3
	3: DESTROYED		Got him!
	4: DMG=0.12		Small hit on Enterprise
	POS=79.75	RCL 06	How are our shields?
26)	46.32	RCL 08	How much total energy do we have left?
27)	16,690.00	4 SHIFT b	How much total damage do we have?
28)	4: DMG=18.10		Find the other 2 Klingon ships and destroy them!

Specifics:

1) Program is 484 bytes long.
2) Program is XROM 23,12.
3) Calls XROM "S" – XROM 23,25 (SEED? prompt) and XROM "R" – XROM 23,26 (Random Number Generator).
4) Uses registers 00 – 11. SIZE 012 required.
   - 00 – Random number seed
   - 01 – Enemy ship 1
   - 02 – Enemy ship 2
   - 03 – Enemy ship 3
   - 04 – Enterprise ship
   - 05 – Hit to shield during attack or scan
   - 06 – Enterprise shield energy
   - 07 – Sensor range constant
   - 08 – Enterprise total energy
   - 09 – Indirect (ship) pointer
   - 10 – Temp: warp amount or phaser strength
   - 11 – Temp: distance to attacking ship or warp bearing
5) Labels used:
   - 01 – Random number routine. Calls XROM "R" (23,26).
   - 03, 05, 06, 09 – local loops
   - 10 – Display damage
   - 20 – Get ship's (R09) bearing (Y) and distance (X) from Enterprise
   - 33 – Display ship DESTROYED
   - 40 – Isolate cumulative damage ZZZ.QQQ value
   - 60 – ALL DONE (Enterprise out of energy)
   - 70 – REPAIR/DAMAGE (x=amount, negative for repair), ship# in R09
   - 80 – Take a hit on shields and show shields remaining
   - 88 – Hit on Enterprise and accumulate damage
   - 99 – End and display Enterprise position
   - A – START
   - B – SCAN
   - C – WARP
   - D – SHIELDS
   - E – ATTACK
   - b – SHOW DAMAGE (x=ship)
   - c – REPAIR Enterprise (x=amount)

6) Flags used:
   - 05 – Enterprise shields failed?
   - 27 – Flag 27 is set to allow use of user–defined keys
   - 29 – Flag 29 is used for formatted output
7) Display mode: FIX 00 and FIX 02 are set for output. The existing display mode is not retained.

Program listing:

```
 01 LBL "ST"
 02 XROM “S”
 03 SF 27
 04 LBL A
 05 CLST
 06 STO 01
 07 STO 02
 08 STO 03
 09 STO 04
 10 STO 05
 11 STO 06
 12 25
 13 STO 07
 14 2 E4
 15 STO 08
 16 4
 17 STO 09
 18 LBL 09
 19 XROM “R”
 20  E4
 21 *
 22 INT
 23  E2
 24 /
 25 RCL 01
 26 X=Y?
 27 GTO 09
 28 RDN
 29 RCL 02
 30 X=Y?
 31 GTO 09
 32 RDN
 33 RCL 03
 34 X=Y?
 35 GTO 09
 36 RDN
 37 STO IND 09
 38 DSE 09
 39 GTO 09
 40 CLST
 41 LBL 99
 42 DEG
 43 FIX 02
 44 SF 29
 45 "POS="
 46 ARCL 04
 47 PROMPT
 48 LBL B
 49 1.003
 50 STO 09
 51 LBL 06
 52 XEQ 20
 53 X<0?
 54 GTO 05
 55 RCL 07
 56 X<=Y?
 57 GTO 05
 58 RDN
 59 CLA
 60 FIX 00
 61 CF 29
 62 ARCL 09
 63 >": d"
 64 ARCL X
 65 >" \0D"
 66 ARCL Y
 67 AVIEW
 68 PSE
 69 PSE
 70 FIX 02
 71 SF 29
 72 XROM “R”
 73 ATAN
 74 ST+ 05
 75 LBL 05
 76 ISG 09
 77 GTO 06
 78 XEQ 88
 79 GTO 99
 80 LBL C
 81 STO 10
 82 X<>Y
 83 STO 11
 84 8
 85 ENTER
 86  E2
 87 RCL 04
 88 XEQ 40
 89 -
 90 %
 91 RCL 10
 92 X>Y?
 93 X<>Y
 94 10
 95 *
 96 STO 10
 97 X^2
 98 RCL 08
 99 X<>Y
100 X>Y?
101 GTO 60
102 ST- 08
103 RCL 11
104 RCL 10
105 P-R
106 INT
107 ST+ 04
108 X<>Y
109 INT
110  E2
111 /
112 ST+ 04
113 GTO 99
114 LBL D
115 RCL 08
116 X<>Y
117 X>Y?
118 GTO 60
119 ST- 08
120 LBL 80
121 FIX 02
122 ST+ 06
123 "SHLDS="
124 ARCL 06
125 AVIEW
126 RTN
127 LBL 60
128 "ENERGY=0"
129 PROMPT
130 LBL E
131 STO 09
132 CF 05
133 XEQ 20
134 X<0?
135 GTO 33
136 STO 11
137 RCL 06
138  E2
139 RCL IND 09
140 XEQ 40
141 -
142 XROM “R”
143 *
144 X<=Y?
145 GTO 03
146 RCL 08
147 X<>Y
148 X>Y?
149 GTO 60
150 ST- 08
151 R^
152 -
153 SF 05
154 LBL 03
155 STO 05
156 CHS
157 XEQ 80
158 PSE
159 PSE
160 9
161 "POWER?"
162 PROMPT
163 STO 10
164 RCL 07
165 2
166 /
167 RCL 11
168 X>Y?
169 GTO 99
170 X^2
171 RCL 10
172 *
173 RCL 08
174 X<>Y
175 X>Y?
176 GTO 60
177 ST- 08
178 XROM “R”
179  E2
180 *
181 RCL 11
182 SQRT
183 /
184 XEQ 70
185 XEQ 88
186 GTO 99
```

```
187 LBL 20
188 RCL IND 09
189 X<0?
190 RTN
191 FRC
192 FIX 02
193 RND
194 RCL 04
195 FRC
196 RND
197 -
198  E2
199 *
200 RCL IND 09
201 INT
202 RCL 04
203 INT
204 -
205 R-P
206 RTN
207 LBL b
208 STO 09
209 RCL IND 09
210 XEQ 40
211 GTO 10
212 LBL 40
213  E2
214 *
215 FRC
216  E3
217 *
218 RTN
219 LBL 88
220 RCL 05
221 X=0?
222 RTN
223 ST- 05
224 4
225 STO 09
226 FS?C 05
227 1/X
228 /
229 GTO 70
230 LBL c
231 CHS
232 4
233 STO 09
234 RDN
235 XEQ 70
236 GTO 99
237 LBL 70
238 XEQ 10
239 PSE
240 PSE
241 RCL IND 09
242 XEQ 40
243 +
244  E2
245 X<=Y?
246 GTO 33
247 RDN
248  E5
249 /
250 RCL IND 09
251 FIX 02
252 RND
253 +
254 STO IND 09
255 RTN
256 LBL 10
257 CLA
258 FIX 00
259 CF 29
260 ARCL 09
261 >": DMG="
262 SF 29
263 FIX 02
264 ARCL X
265 AVIEW
266 RTN
267 LBL 33
268 RCL IND 09
269 ABS
270 CHS
271 STO IND 09
272 CLA
273 FIX 00
274 CF 29
275 ARCL 09
276 >": DESTROYED"
277 FIX 02
278 SF 29
279 AVIEW
280 PSE
281 END
```

Game of Tic–Tac–Toe (LBL "TTT")

History: This is based on the HP–67 Games Pac Tic–Tac–Toe program (program 11 in that Pac), which is based in turn on the HP 65 Users' Library program written by Delmer D. Hinrichs.

Object: Connect three dots before the calculator does. The calculator will not lose, so your best hope is to play for a draw. The calculator keys represent the spaces on the Tic–Tac–Toe board, i.e., the 5 key is the center of the board, etc.

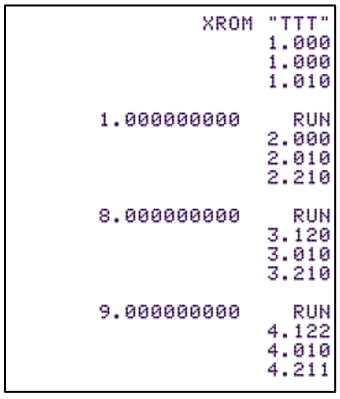
Instructions:

1) XEQ ALPHA TTT ALPHA
2) Calculator makes the first move. A number 1–9 is displayed indicating the number of the move to be displayed (1, 2, 3, etc.). The calculator always plays in space 2 of the board first.
3) Each row of the board is displayed 3 digits at a time after a move before the final display is made of the entire board, displayed in this form: 2.010000000, where the 2 represents the position on the board just occupied, and the next 9 digits after the decimal point represent the board. The first three decimals indicate positions 1, 2, and 3. The second three decimals 4,5,6 and the last three 7,8,9.
4) A value of 0 on the board indicates an empty space. A value of 1 indicates the calculator occupies the board in that position and a value of 2 indicates you occupy that position.
5) Enter your move and press R/S or B. Note: the calculator does not check for valid moves.
6) The calculator makes its next move.
7) Steps 5 and 6 are repeated until the calculator wins or the game is a tie. The calculator does not determine when a game is over.

Example game of TTT:

	See	Press	
1)		XEQ ALPHA TTT ALPHA	
2)	1.000		(Value is displayed during a pause. Decimal portion is top row of the playing board. The 1 indicates this is the first move.)
3)	1.000		(Decimals of the second number displayed are second row of board)
4)	1.010		(Decimals of the third number displayed are third row of board)
5)	2.010000000		(Move displayed. The calculator moved into position 2, middle of the bottom row. Rest of decimals indicate remainder of board is empty.)
6)	2.010000000	1 R/S	(R/S or B can be pressed here)
7)	2.000		(Move two has occurred. Top row still empty)
8)	2.010		(Middle row now has a 1 in center. This is HP's coming move.)
9)	2.210		(Bottom row is displayed.)
10)	5.210010000		(The calculator has moved to position 5, middle of the second row. Note how decimals of items 7–9 of this example indicate the board. This is printed if a printer is attached.)
11)	5.210010000	8 R/S	(User moves into middle of the top row)
12)	3.120		(This is the top row and HP has moved into position 7)
13)	3.010		(This is the middle row)
14)	3.210		(This is the bottom row)
15)	7.210010120		(Full board is displayed)
16)	7.210010120	9 R/S	(Afraid I have made a bad move here)
17)	4.122		(Top row displayed)
18)	4.010		(Middle row of board)
19)	4.211		(Bottom row. Note the 3–in–a–row by HP)
20)	3.211010122		(This is a win by the calculator. 7, 5, 3 form three in a row. I lose.)

This is how it looks on an printer from an HP–41 emulator:



Specifics:

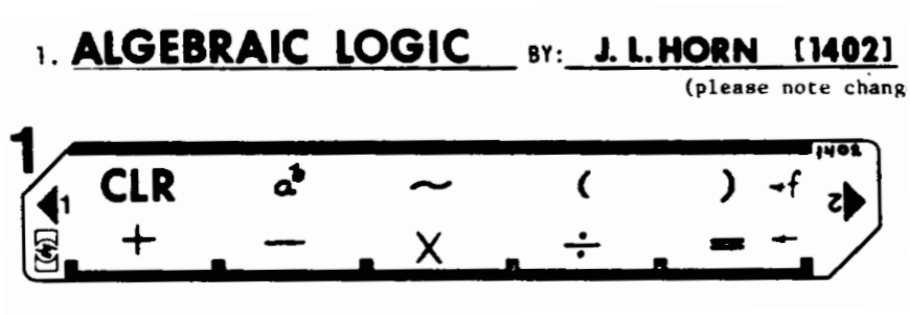
1) Program is 182 bytes long.
2) Program is XROM 23,13.
3) Uses registers 00 – 03. SIZE 004 required.
   - 00 – Move counter
   - 01 – Complete board stored
   - 02 – HP or players move
   - 03 – Calculator's move (digits represent fields to choose after 1st, 2nd, etc. move)
4) Labels used:
   - 01, 03 – 09: Return possible moves for board positions 1, 3–9.
     Since HP's first move is always position 2, there is no Label 02.
   - 10 – If X=1: record HP's move, or if X=2: record players move. Both stored in R02.
   - 11 – Display one row of the board
   - 12 – Determine HP's move after first move of player
   - 13 – Record HP's move on the board
   - 14 – Split off next digit from R03
   - A – Starts new game
   - B – Enter move
   - C – Display the board again
5) Flags used:
   - 05 – Flag 05 is the first move flag
   - 27 – Flag 27 is set to allow use of user–defined keys
6) Display mode: FIX 03 and FIX 09 are set for output. The existing display mode is not retained.

Program Listing:

```
01 LBL "TTT"
02 LBL A
03 CLX
04 STO 00
05 STO 01
06 SF 05
07 SF 27
08 2
09 LBL 13
10 STO 02
11  E
12 ST+ 00
13 XEQ 10
14 LBL C
15  E6
16 XEQ 11
17  E3
18 XEQ 11
19  E
20 XEQ 11
21 ADV
22 RCL 02
23 RCL 01
24 +
25 FIX 09
26 CLD
27 STOP
28 LBL B
29 STO 02
30 2
31 XEQ 10
32 FS?C 05
33 GTO 12
34 RCL 02
35 XEQ 14
36 X=Y?
37 XEQ 14
38 GTO 13
39 LBL 12
40 XEQ IND 02
41 STO 03
42 XEQ 14
43 GTO 13
44 LBL 14
45 RCL 03
46 FRC
47 10
48 *
49 STO 03
50 INT
51 RTN
52 LBL 10
53 RCL 02
54 10^X
55 /
56 ST+ 01
57 RTN
58 LBL 11
59 FIX 03
60 RCL 01
61 *
62 FRC
63 RCL 00
64 +
65 CLD
66 VIEW X
67 PSE
68 RTN
69 LBL 01
70 .5873649
71 RTN
72 LBL 03
73 .5891467
74 RTN
75 LBL 04
76 .13598
77 RTN
78 LBL 05
79 .1374698
80 RTN
81 LBL 06
82 .31578
83 RTN
84 LBL 07
85 .13589
86 RTN
87 LBL 08
88 .3175964
89 RTN
90 LBL 09
91 .31587
92 END
```

AOS Program (LBL "AOS")

History: This is based on the original AOS (Algebraic Operating System) program written by Jim Horn appearing in 65 Notes V4N10P25 for the write–up and on page 35 for the program itself. It incorporates an error fix Jim noted in PPC Notes V5N3P5. One of the adjustments made was to add the alpha labels allowing each function so these labels can be assigned to the associated keys in USER mode. This allows the calculator to function as if it were in AOS mode while in USER mode. The picture below is from V4N10P25 of 65 Notes.



Object: Allow the calculator to mimic functioning as an AOS–style machine rather than RPN. The author, Jim Horn, has graciously given these comments about the program:

> "The one or two digit codes given to each key in the program listing consist of a units digit that gives the operator hierarchy and the rest that gives a unique ID to each operator that can later be executed via an indirect XEQ. For instance, the A key is shown as the addition key, so LBL A and LBL "+" handle that by putting 61 in the X register and jumping ahead to the main handler (LBL 00, step 40). The 61 indicates that addition will happen by doing an XEQ 06 with a priority level of 1, putting it below multiplication and division (2), negation (3) or parenthesis and powers (always the highest, in this case, 4). As another example, the code entered for Y^X is 14, so execution will transfer to LBL 01 with a priority of 4.
>
> Flag 1 has the important role of noting when an implied multiply is needed and sees that it gets provided. Flag 2 indicates a unary (single operator) function. If not set, a function is treated as binary (two operator).
>
> There are two stacks set up in memory. R0 through R12 are the operator stack; R13 through R21 are the operand stack. The HP–67's remaining registers were used for the Last Operator, the Operator Stack Pointer, the Operand Stack Pointer, and the index register for all of the above.
>
> The algorithm was from a flow chart in a programming text from 1971. After I wrote the program and it was published in 65 Notes, I discovered that the book's flowchart and algorithm were incomplete and flawed. Thus the NOP published several issues later."

Running the program:

1) XEQ ALPHA AOS ALPHA
2) Optional: If you are using an HP 41CX or have the Extended Functions module for the HP 41 series, key in and use the auxiliary AOSKY program included after the AOS program listing to assign the labels in this program file to the "natural" keys to use the calculator more "normally." Or assign them manually.
    a. This program assigns the global labels in the AOS program file as follows:
        LBL "+" to the add key, LBL "–" to the subtract key, LBL "*" to the multiply key, and LBL "/" to the divide key.
        LBL "=" to the ENTER key
        LBL "YX", the algebraic Y^X function, to the shifted location of Y^X
        LBL "<", the label for open parenthesis, to the X<>Y key
        LBL ">", the label for the close parenthesis, to the RDN (Roll Down) key, and
        LBL "NEG" (Negate – not quite CHS) to the shifted location of the CHS key.
        LBL "AOS" to the shift of the backarrow key to use as a "CLR" style function similar to LBL a.
    b. Once you are done using the AOS program, execute CLKEYS to return the keyboard to normal.
3) The AOS program can handle up to 13 pending operations, counting all open parentheses as an operation, and up to 8 pending operands. If you exceed these, you will generate an error.
4) Note: While the program is not running, any built–in HP calculator math function can be executed on the displayed data. If, however, you need to execute a function on a key that has been reassigned or that is mapped to a local label in USER mode, you will need to press the USER mode rocker switch at the top of the calculator, execute the desired function, and then press the USER mode switch again before continuing with a R/S.
5) In fact, you can do any manual calculations in RPN mode you like (probably out of USER mode to keep things simple) and then when you are ready to return to the AOS program, make sure USER mode is set and press R/S. This is possible because LBL 99 in the code shown below is the common exit point for all operations. If you look

at the code, pressing R/S when the code stops after LBL 99 will set flag 22 so that the result of a manually executed math function can be used for subsequent calculations in the program as you resume it.

6) Running the program is perhaps best illustrated by the examples below.

Example 1: Evaluate 1 + 2 x 3 ^{4}

	See	Press (Assumes key assignments)	Press (without key assignments)
1)		XEQ ALPHA AOS ALPHA	XEQ ALPHA AOS ALPHA
2)	0.00	1 +	1 A
3)	1.00	2 x	2 C
4)	2.00	3 shift Y^X	3 shift b
5)	3.00	4 ENTER	4 E
6)	163.00		

Example 2: Evaluate (1 + 2) x (3 – (–4 / (5 – 6))) =

	See	Press (Assumes key assignments)	Press (without key assignments)
1)		XEQ ALPHA AOS ALPHA	XEQ ALPHA AOS ALPHA
2)	0.00	X<>Y	shift d
3)	0.00	1 +	1 A
4)	1.00	2 RDN (Note: Flag 1 is set)	2 shift e
5)	3.00	x (Note: Flag 1 is cleared)	C
6)	3.00	X<>Y	shift d
7)	3.00	3 –	3 B
8)	3.00	X<>Y	shift d
9)	3.00	4 CHS /	4 CHS D
10)	4.00	X<>Y	shift d
11)	4.00	5 –	5 B
12)	5.00	6 ENTER	6 E
13)	21.00		

Example 2 notes: 1) The last closing parentheses are automatically supplied by pressing E (or ENTER).
2) To enter a negative number, simply press the CHS key. The working of the NEG function is demonstrated by example 3 below.

Example 3: Evaluate 5 – (9 x 3) using the NEG function without using parentheses.

	See	Press (Assumes key assignments)	Press (without key assignments)
1)		XEQ ALPHA AOS ALPHA	XEQ ALPHA AOS ALPHA
2)	0.00	9 x	9 C
3)	9.00	3 ENTER	3 E
4)	27.00	shift CHS	shift c (This is the NEG function)
5)	27.00	+	A
6)	–27.00	5 ENTER	5 E (The –27.00 reflects the NEG)
7)	–22.00		

Example 3 note: NEG takes the previous result and makes it negative but does not display it until the next operation is performed.

Example 4: Evaluate 5 – (9 x 3) using the built–in CHS function.

	See	Press (Assumes key assignments)	Press (without key assignments)
1)		XEQ ALPHA AOS ALPHA	XEQ ALPHA AOS ALPHA
2)	0.00	9 x	9 C
3)	9.00	3 ENTER	3 E
4)	27.00	CHS R/S	CHS R/S
5)	–27.00	+	A
6)	–27.00	5 ENTER	5 E
7)	–22.00		

Example 4 note: After pressing CHS in step 4, the R/S key must be pressed so that flag 22, the numeric input flag, is set. This indicates to the program that an entry has been made allowing it to process the input properly.

Example 5: Evaluate $M = \sqrt{5\left[\left(\left\{\left\{\left[\left(1+0.2\left[\frac{350}{661.5}\right]^2\right)^{3.5}-1\right]\left[1-(6.875\times10^{-6})25500\right]^{-5.2656}\right\}+1\right)^{0.286}-1\right]}$

	See	Press (With key assignments)	Press (No key assignments)	Comments
1)		shift FIX 4	shift FIX 4	Show 4 decimals
2)		XEQ ALPHA AOS ALPHA	XEQ ALPHA AOS ALPHA	
3)	0.0000	5 x	5 C	5 x
4)	5.0000	X<>Y	shift d	(
5)	5.0000	X<>Y	shift d	(
6)	5.0000	X<>Y	shift d	(
7)	5.0000	X<>Y	shift d	(
8)	5.0000	X<>Y	shift d	(
9)	5.0000	1 +	1 A	1 +
10)	1.0000	0.2 x	0.2 C	0.2 x
11)	0.2000	X<>Y	shift d	(
12)	0.2000	350 /	350 D	350 /
13)	350.0000	661.5 RDN	661.5 shift e	661.5)
14)	0.5291	shift Y^X	shift b	Y^X
15)	0.5291	2 RDN	2 shift e	2)
16)	1.0560	shift Y^X	shift b	Y^X
17)	1.0560	3.5 –	3.5 B	3.5 –
18)	1.2101	1 RDN	1 shift e	1)
19)	0.2101	x	C	x
20)	0.2101	X<>Y	shift d	(
21)	0.2101	1 –	1 B	1 –
22)	1.0000	6.875 EEX 6 CHS x	6.875 EEX 6 CHS C	.000006875 x
23)	6.8750 –06	25500 RDN	25500 shift e	25500)
24)	0.8247	shift Y^X	shift b	Y^X
25)	0.8247	5.2656 CHS RDN	5.2656 CHS shift e	–5.2656)
26)	0.5796	+	A	+
27)	0.5796	1 RDN	1 shift e	1)
28)	1.5796	shift Y^X	shift b	Y^X
29)	1.5796	0.286 –	0.286 B	0.286 –
30)	1.1397	1 RDN	1 shift e	1)
31)	0.1397	=	E	=
32)	0.6984	USER SQRT USER	USER SQRT USER	SQRT
33)	0.8357			(Mach 0.8357)

Example 5 note: This is the famous "Mach Number" formula from many past RPN vs. AOS illustrations. With this AOS program, you can now choose either way to approach this problem. In AOS, you will need to do the square root last so begin with the 5 x portion of the formula. To execute the square root, notice how USER mode must be turned off, the square root key pressed, and then USER turned back on. For the power of 2 in the formula, you can turn USER off and execute the X^2 function and turn USER back on, or you can simply use shift Y^X and then 2.

Example 6: Evaluate (10 / 2)^2 + 1 using the built–in X^2 function.

	See	Press (Assumes key assignments)	Press (without key assignments)
1)		XEQ ALPHA AOS ALPHA	XEQ ALPHA AOS ALPHA
2)	0.00	X<>Y 10 /	shift d 10 D
3)	10.00	2 RDN	2 shift e
4)	5.00	USER X^2 USER	USER X^2 USER
5)	25.00	+	A
6)	5.00		

Note: This is incorrect at this point. Because the access to the built–in X^2 function did not occur with flag 22 set, the AOS program does not detect it properly. Flag 22 was set upon the entry of the "2" in line 3 above, but pressing RDN or executing shift e (to close the parenthesis) does not preserve flag 22. Flag 22 is how the AOS program determines if a number displayed has changed. The way to ensure it functions correctly is shown in example 7 below.

Example 7: Evaluate (10 / 2)^2 + 1 using the built–in X^2 function.

	See	Press (Assumes key assignments)	Press (without key assignments)
1)		XEQ ALPHA AOS ALPHA	XEQ ALPHA AOS ALPHA
2)	0.00	X<>Y 10 /	shift d 10 D
3)	10.00	2 RDN	2 shift e
4)	5.00	USER X^2 USER	USER X^2 USER
5)	25.00	R/S	R/S
6)	25.00	+	A
7)	25.00	1 ENTER	1 E
8)	6.00		

Note: This is correct. Pressing R/S enables the AOS program to detect the used computed square of 5 and use it for further computations.

Therefore, it is probably always safer to press R/S after making any calculations outside of the program itself.

Specifics:

1) Program is 350 bytes long.
2) Program is XROM 23,14 for the AOS label. XROMs 23,14 through 23,23 are used in this program file.
3) Uses registers 00 – 24. SIZE 025 required.
   - 00 – 12: Operator value
   - 13 – 21: Pending operator stack
   - 22 – Last operator
   - 23 – Stack pointer for values
   - 24 – Stack pointer for operations
4) Labels used:
   - 00 – 09: used
   - 99 – Common exit point for all operations
   - A and "+" – Addition
   - B and "–" – Subtraction
   - C and "*" – Multiplication
   - D and "/" – Division
   - E and "=" – Equals key
   - a and "AOS" – Clear AOS calculator
   - b and "YX" – Y^X function
   - c and "NEG" – Negate function
   - d and "<" – Open parenthesis
   - e and ">" – Close parenthesis. Note: At the end of a calculation, press E or = key assignment instead.
5) Flags used:
   - Flag 1 has the important role of noting when an implied multiply is needed and sees that it gets provided
   - Flag 2 indicates a unary (single operator) function. If not set, a function is treated as binary (two operator).
   - Flag 22 is used to detect user numeric input
   - Flag 27 is set
6) Display mode: Display mode is not changed. The existing display mode is retained.

Program Listing:

```
01 LBL "AOS"
02 SF 27
03 LBL a
04 CF 01
05 CF 02
06 CF 22
07 12
08 STO 23
09 -1
10 STO 24
11 CLX
12 RTN
13 LBL "+"
14 LBL A
15 61
16 GTO 00
17 LBL "-"
18 LBL B
19 51
20 GTO 00
21 LBL "*"
22 LBL C
23 42
24 GTO 00
25 LBL "/"
26 LBL D
27 32
28 GTO 00
29 LBL "YX"
30 LBL b
31 14
32 GTO 00
33 LBL "NEG"
34 LBL c
35 23
36 GTO 00
37 LBL "<"
38 LBL d
39 5
40 LBL 00
41 10
42 /
43 STO 22
44 INT
45 X!=0?
46 GTO 00
47 FS? 01
48 XEQ 03
49 LBL 00
50 RDN
51 FS?C 22
52 XEQ 02
53 RCL 22
54 INT
55 X=0?
56 GTO 00
57 LBL 07
```

```
 58 RCL 24
 59 X<0?
 60 GTO 00
 61 RCL IND 24
 62 FRC
 63 RCL 22
 64 FRC
 65 X>Y?
 66 GTO 00
 67 RCL IND 24
 68 INT
 69 X=0?
 70 GTO 00
 71 XEQ 01
 72 GTO 07
 73 LBL 00
 74 ISG 24
 75 ENTER
 76 RCL 24
 77 13
 78 X<=Y?
 79 ASIN
 80 RCL 22
 81 STO IND 24
 82 RCL IND 23
 83 CF 01
 84 GTO 99
 85 LBL ">"
 86 LBL e
 87 1
 88 STO 22
 89 X<>Y
 90 FS?C 22
 91 XEQ 02
 92 RCL 24
 93 X<0?
 94 SQRT
 95 RCL IND 24
 96 INT
 97 X=0?
 98 GTO 08
 99 XEQ 01
100 GTO e
101 LBL 08
102 DSE 24
103 ENTER
104 RCL IND 23
105 SF 01
106 GTO 99
107 LBL E
108 LBL "="
109 1
110 STO 22
111 X<>Y
112 FS?C 22
113 XEQ 02
114 RCL 24
115 X<0?
116 GTO 00
117 RCL IND 24
118 XEQ 01
119 GTO E
120 LBL 00
121 RCL IND 23
122 XEQ a
123 RDN
124 RDN
125 SF 22
126 GTO 99
127 LBL 02
128 ISG 23
129 ENTER
130 21
131 RCL 23
132 -
133 X<0?
134 SQRT
135 RDN
136 STO IND 23
137 RCL 22
138 INT
139 X!=0?
140 RTN
141 LBL 03
142 ISG 24
143 ENTER
144 4.2
145 STO IND 24
146 RDN
147 RTN
148 LBL 01
149 RCL IND 23
150 DSE 23
151 RCL IND 23
152 X<>Y
153 XEQ IND Z
154 FS?C 02
155 ISG 23
156 ENTER
157 RCL 23
158 13
159 -
160 X<0?
161 SQRT
162 X<>Y
163 STO IND 23
164 DSE 24
165 RTN
166 RTN
167 LBL 01
168 Y^X
169 RTN
170 LBL 02
171 CHS
172 LBL 00
173 SF 02
174 RTN
175 LBL 03
176 /
177 RTN
178 LBL 04
179 *
180 RTN
181 LBL 05
182 CHS
183 LBL 06
184 +
185 LBL 99
186 RTN
187 SF 22
188 END
```

Auxiliary Program Listing:

```
 01 LBL "AOSKY"
 02 "AOS"
 03 -44
 04 PASN
 05 "-"
 06 51
 07 PASN
 08 "+"
 09 61
 10 PASN
 11 "*"
 12 71
 13 PASN
 14 "/"
 15 81
 16 PASN
 17 "="
 18 41
 19 PASN
 20 "YX"
 21 -12
 22 PASN
 23 "<"
 24 21
 25 PASN
 26 ">"
 27 22
 28 PASN
 29 "NEG"
 30 -42
 31 PASN
 32 CLST
 33 END
```

Sum of the Digits game (LBL "SD")

History: A sum of the digits game like this appeared on page 25 of the HP Digest, Volume 5, 1979.

Object: The calculator generates a secret two–digit random number and computes the sum of the two digits. Add a number to that secret number so that the number gets to 99 while only seeing the sum of the digits.

Instructions:

1) XEQ ALPHA SD ALPHA
2) At the SEED? prompt, enter a decimal seed between 0 and 1 and press R/S.
3) The calculator will display SUM=XY, which is the sum of the two digits of the secret number.
4) Key your value to be added to the secret number to get the number to 99 and press R/S.
5) If you added the correct value to the number, the calculator "YES IT'S " and the original secret number. The calculator will then display how many attempts you required.
6) If you did not enter a value that got the secret number to 99, the calculator add the value you entered to the original secret number and compute the sum of the digits of that new number and display SUM=YZ.
7) If you add a number that makes the new total greater than 99, the calculator displays "HIGH" and you should consider adding a smaller number.
8) Repeat steps 4 – 7 until you win.

Example games:

	See	Press	
1)		XEQ ALPHA SD ALPHA	
2)	SEED?	0.123456 R/S	
3)	SUM=13	14 R/S	(Maybe the number is 85?)
4)	SUM=9	18 R/S	(Maybe it was 67 and is now 81?)
5)	YES IT'S 67		(Not too bad!)
6)	2 TRIES		

	See	Press	
1)		XEQ ALPHA SD ALPHA	
2)	SEED?	0.111111 R/S	
3)	SUM=7	92 R/S	(Maybe the number is 7?)
4)	HIGH		(Number plus 92 is > 99. Add a lower number)
5)	SUM=7	65 R/S	(Maybe it is 34?)
6)	HIGH		(Number plus 65 is > 99. Add a lower number)
7)	SUM=7	38 R/S	(Maybe it is 61?)
8)	SUM=9	9 R/S	(Maybe it was 52 and is now 90?)
9)	SUM=9	9 R/S	(No, must have been 43 and is really now 90?)
10)	YES IT'S 43		(Not too bad!)
11)	5 TRIES		

Specifics:

1) Program is 113 bytes long.
2) Program is XROM 23,24.
3) Calls XROM "S" – XROM 23,25 (SEED? prompt) and XROM "R" – XROM 23,26 (Random Number Generator).
4) Uses registers 00 – 03. SIZE 004 required.
    - 00 – Random number seed
    - 01 – Original number
    - 02 – Current number (previous number plus value to be added)
    - 03 – Number of guesses
5) Labels used:
    - 01 – Number added did not go over 99 so separate digits and compute new sum
    - 02 – Common label to display results
    - A – Starts a new game
6) Flags used:
    - 27 – Flag 27 is set to allow use of user–defined keys
    - 29 – Flag 29 is used for formatted output
7) Display mode: FIX 00 and FIX 04 are set for output. The existing display mode is not retained.

Program Listing:

```
01 LBL "SD"
02 XROM “S”
03 SF 27
04 LBL A
05 CLST
06 STO 03
07 XROM “R”
08  E2
09 *
10 INT
11 STO 01
12 STO 02
13 LBL 01
14 RCL 02
15 10
16 MOD
17 RCL 02
18 LASTX
19 /
20 INT
21 +
22 FIX 00
23 CF 29
24 "SUM="
25 ARCL X
26 PROMPT
27 INT
28 ST+ 02
29 1
30 ST+ 03
31 99
32 RCL 02
33 X<Y?
34 GTO 01
35 X=Y?
36 GTO 02
37 R^
38 ST- 02
39 "HIGH"
40 AVIEW
41 PSE
42 GTO 01
43 LBL 02
44 FIX 00
45 CF 29
46 "YES, IT'S "
47 ARCL 01
48 AVIEW
49 PSE
50 PSE
51 " "
52 ARCL 03
53 >" TRIES"
54 RCL 01
55 FIX 04
56 SF 29
57 AVIEW
58 END
```

Seed and Random Number (LBL "S" and LBL "R")

History: LBL "R" is the common random number generator for games in this rom. It is the same generator used in the PPC ROM routine RN. That generator was first used by Don Malm (1362) in the HP–65 Users Library Program #4867. For a six digit decimal seed, it generates 1 million numbers before cycling. LBL "S" is the common routine that prompts the user to enter a random number seed. Both routines use Register 00.

Object: LBL "S": Prompt for and store a random number seed into memory 00.
LBL "R": Generate a random number from the seed in memory 00 and update the seed for the next call.

Specifics:

1) Program is 49 bytes long for both labels
2) Program is XROM 23,25 for LBL "S" and XROM 23,26 for LBL "R"
3) Uses register 00 for the random number seed
4) Labels used: LBL "S" and LBL "R"
5) Note: LBL "S" truncates any entered seed to six digits

Program Listing:

```
01 LBL "S"
02 RCL 00
03 "SEED?"
04 PROMPT
05 ABS
06 FRC
07  E6
08 *
09 INT
10  E6
11 /
12 STO 00
13 RTN
14 LBL "R"
15 RCL 00
16 ABS
17 9821
18 *
19 .211327
20 +
21 FRC
22 STO 00
23 END
```

Reset Flags program (LBL "RF")

History: This is the PPC ROM routine RF found on page 376 of the PPC ROM users manual. The RF routine was devised by Carter Buck (4783).

Object: RF sets all flags to their default status, i.e., the state they would be in after MEMORY LOST. The one exception is that RF sets the FIX 2 display mode rather than FIX 4. RF is a short, stand–alone program that can be executed any time as a cleanup measure from the keyboard or it can be executed during a running program.

Specifics:

1) Program is 19 bytes long.
2) Program is XROM 23,27.
3) Uses no registers. SIZE 000 is fine.
4) Labels used: RF – executes the program.
5) Line 02 in the program below is HEX F4 2C 02 80 00.
6) The bits representing this string will be copied into status register d, setting/clearing the corresponding flags.
7) F4 is the text byte indicating four text characters in the following string. This text line is stored into status register d to set the flags. The text line is (Hex) F4 2C 02 80 00. The F4 indicates a 4 byte string.
    a. Byte 2C sets flags 26, 28, 29.
    b. Byte 02 sets flag 38.
    c. Byte 80 sets flag 40.
8) Flag 03 is set by the bits represented in the ALPHA text string. It is more byte efficient to have the CF 03 instruction than to make the ALPHA text in line 02 avoid setting this one flag.
9) The numeric stack is unchanged by this program. The ALPHA register (registers M, N, O and P) is cleared.

Program Listing:

```
01 LBL "RF"
02 ",\02\80\00"
03 ASTO d
04 CF 03
05 CLA
06 END
```

Final words?

That's it for now. Stay tuned for an HP67FUN2 rom perhaps. Enjoy.