HP Forum Archive 15

S&SMC#10: Final Notice

*Message #1 Posted by **Valentin Albillo** on 6 June 2005, 10:14 a.m.*

Hi all:

Many thanks to those of you interested in my S&SMC#10: Counting Beans challenge, most specially to those who found the time to post a contribution, all submitted are very clever and interesting.

Tomorrow I'll post my original solution in two flavors, a 5-line HP-71B BASIC program (can be reduced to 4 lines) which finds the only solution in 15 sec. average (nearly inmensurable in Emu71), and its optimized RPN translation for the HP-15C, which is a fairly small, 52-step program (can be reduced to 51 steps) which finds the solution in 2 min or so. I'll include relevant comments as well.

So, for those of you still wanting to contribute your own approach before I publish mine(s), you have an additional day to do so. Mind you, you can post them afterwards and I'll be equally pleased as well, of course.

Best regards from V.

Re: S&SMC#10: Final Notice

*Message #2 Posted by **Giancarlo** on 6 June 2005, 4:25 p.m.,*
in response to message #1 by Valentin Albillo

Hello Valentin. Your S&SMCs are very interesting, even if I haven't got any time to solve anyone of them, but..... never thought to collect all of them in a document, with your "official" solutions and the best ones from other contributors? If you already did that, sorry for my useless suggestion - where could it be possible to find such document? Thank you. Cheers. Giancarlo

Re: S&SMC#10: Final Notice

*Message #3 Posted by **Valentin Albillo** on 7 June 2005, 4:44 a.m.,*
in response to message #2 by Giancarlo

Hi, Giancarlo:

It's a pretty good idea but regrettably no such 'compilation' does exist. It's all a matter of available free time and priorities. I'm already using lots of time to concoct these challenges and my solutions to them, plus all the postings, time which should really be allocated to other matters.

Making a compilation as you suggest would require even more time, if done properly and I simply can't afford it right now. Perhaps some other interested people are collecting them and can assemble and make available such a compilation.

Also, in the early stages, not all my challenges were succesful, I seem to remember that a couple of them gathered no useful responses at all, perhaps because people thought they were too complicated or deeply mathematical in nature.

Thanks for your interest and

Best regards from V.

S&SMC#10: My Original Solutions & Comments

*Message #4 Posted by **Valentin Albillo** on 7 June 2005, 5:15 a.m.,*
in response to message #1 by Valentin Albillo

Hi all,

Thanks a lot to all of you interested in this small challenge of mine, we've got a surprising number of clever programs for a variety of HP machines, written in assorted languages including RPL, several RPN flavors, and even BASIC.

Here's my original RPN solution for the HP-15C, a small, 52-step program which uses matrix operations to find the solution quickly:

```
   01 *LBL A      17  GTO 8          32  CLX           47  RCL MATRIX A
   02   1         18  RCL MATRIX A   33  STO MATRIX C  48   -
   03  MATRIX 0   19 *LBL 3          34  RCL I         49  MATRIX 7
   04   10        20  GSB 5          35  STO 1         50  RETURN
   06  STO I      21  X=0 ?          36 *LBL 1         51 *LBL 0
   07  DIM A      22  GTO 0          37   1            52  RCL MATRIX C
   08  DIM B      23  RCL MATRIX C   38  RCL+ B
   09  DIM C      24  GSB 5          39  X<> 1
   10 *LBL 2      25  X=0 ?          40  ISG C
   11  MATRIX 1   26  GTO 2          41 *LBL 1
   12 *LBL 8      27  RCL MATRIX C   42  X<> 1
   13  RAN#       28  STO MATRIX A   43  DSE 1
   14  RCL* I     29  GTO 3          44  GTO 1
```

```
    15  INT        30 *LBL 5          45  RESULT B
    16 uSTO A      31  STO MATRIX B   46  RCL MATRIX C


    Note: Step 16 is a "User" STO operation, and must be entered while temporarily
          in USER mode, a small "u" should appear next to the step number. All other
          STO and RCL operations must be entered *out* of USER mode.
```

It doesn't use any allocatable numbered registers but the permanent ones, namely R_0-R_1 (system indexes for matrix operations) and R_I (index register used here merely to hold the constant 10).

The program requires extremely very little user-provided heuristics. Not even the fact that the digits must add up to 10 or that there can't be two 9's, say, is made use of. Instead, the only thing the program knows is that it must compute a function **f(N)**, say, than when applied to a 10-digit N returns another 10-digit result where each digit represents a count of the corresponding digit in the original number, so we have:

```
      f(N) = M
```

Obviously, the value of N we're looking for has the unique property that *when **f** is applied to it, it returns unchanged*, that is:

```
      f(N) = N
```

The program simply goes on to solve this equation. There are a number of reasonable ways we could proceed, but the most successful is to simply select an starting value of N, say N_0, then compute:

$f(N_0) \to N_1,\ f(N_1) \to N_2,\ \ldots$

till for some N_i we do have

$N_{i+1} = f(N_i) = N_i$

As f(N) maps 10-digit numbers into 10-digit numbers and there are only so many of them, we must have *cycles*, where successive applications of f(N) eventually repeat some previous value. We are interested in the *1-cycle*, where f(N) = N, but other cycles can and do exist.

To cater for this, my program initially selects a 10-digit value of N at random, then keeps on applying f(N) to the resulting values while checking if some cycle has been detected. If it is a 1-cycle, this is the solution and the program stops with the solution in the display. Else, if the program has stumbled into a greater order cycle, it simply restarts again, selecting another random starting value. Eventually, it succeeds, provably after very few restarts indeed, and the solution is displayed.

To maximize speed, my program uses a matrix representation for numbers, where a 1x10 matrix represents a 10-digit number. This way we can make use of the *powerful* and *fast* HP-15C matrix operations. Checking for repetition, for instance, is as simple as testing if the Row Norm (MATRIX 7) of the difference of two matrices is zero, which can be done in a few steps and with no slow user-code loops involved.

To run the program, follow these steps:

- Allocate enough registers to dimension the three 1x10 matrices needed (assuming this is the only program in program memory, 26 or any lower number will do):

  ```
  26, DIM (i)
  ```

  MEM should display [26 30 9-1] (i.e., 26 numbered registers plus R_0 and R_I + 30 registers reserved for matrices)

- Optionally, specify a seed for the random number generator. This step is not necessary, as the program will always converge from any starting seed, but if you want to repeat my timing, I always use PI in such cases:

  ```
  PI, STO #RAN,
  ```

- Now simply run the program:

  ```
  GSB A -> (after 2 min 56 sec) -> C   1  10
  ```

  The program stops displaying the matrix which represents and holds the solution found. You just need to display its elements as usual, for instance:

  ```
  MATRIX 1, USER,  RCL C -> (C1,1)  -> 6
                   RCL C -> (C1,2)  -> 2
                   RCL C -> (C1,3)  -> 1
                   RCL C -> (C1,4)  -> 0
                   RCL C -> (C1,5)  -> 0
                   RCL C -> (C1,6)  -> 0
                   RCL C -> (C1,7)  -> 1
                   RCL C -> (C1,8)  -> 0
                   RCL C -> (C1,9)  -> 0
                   RCL C -> (C1,10) -> 0
  ```

  which represents the unique solution: **6210001000**

The program can be made a step shorter using a flag, but I like it better as it is now. The equivalent version for the HP-71B, perhaps easier to understand and translate to other languages, is:

```
10 DESTROY ALL @ OPTION BASE 0 @ DIM N(9),X(9),T(9) @ RANDOMIZE
20 FOR I=0 TO 9 @ N(I)=INT(RND*10) @ NEXT I
30 MAT X=N @ GOSUB 50 @ IF NOT CNORM(X) THEN MAT DISP T @ END
40 MAT X=T @ GOSUB 50 @ IF CNORM(X) THEN MAT N=T @ GOTO 30 ELSE 20
50 MAT T=ZER @ FOR I=0 TO 9 @ T(X(I))=T(X(I))+1 @ NEXT I @ MAT X=T-N @ RETURN
```

which is 5 lines, 195 bytes long. A shorter version is possible, namely this 4-liner:

```
  1 DESTROY ALL @ OPTION BASE 0 @ DIM N(9),X(9),T(9) @ RANDOMIZE
  2 FOR I=0 TO 9 @ N(I)=INT(RND*10) @ NEXT I @ MAT T=N
  3 GOSUB 4 @ IF I THEN MAT DISP T @ END ELSE GOSUB 4 @ MAT N=T @ IF I THEN 2 ELSE 3
  4 MAT X=T @ MAT T=ZER @ FOR I=0 TO 9 @ T(X(I))=T(X(I))+1 @ NEXT I @ MAT X=T-N @ I=CNORM(X)=0 @ RETURN
```

but I think this is really overkill, besides it's actually one byte longer and slightly slower.

Some (long) statistical tests reveal that this kind of program always finds a solution after generating an average of 13.5 ten-digit numbers, though values as low as 1 (directly stumbling upon the solution by chance) or as high as 100 (entering n-cycles repeatedly) are possible, though with very low probabilities. Of course, even in the worst case of 100+ numbers generated and tested, an actual HP-71B can do it in just a few seconds and Emu71 takes usually much less than a second.

See you in S&SMC#11, thanks for your interest and

Best regards from V.

Re: S&SMC#10: My Original Solutions & Comments

Message #5 Posted by ***Chris dean*** *on 7 June 2005, 6:35 a.m.,*
in response to message #4 by Valentin Albillo

Valentin

I am sure everyone who took part in the challenge would like thank you for the time and effort you spent on them. Both in supplying them and in responding.

awaiting #11

Regards

Chris Dean

Thank you very much, Chris ! :-) [N.T.]

Message #6 Posted by ***Valentin Albillo*** *on 7 June 2005, 6:59 a.m.,*
in response to message #5 by Chris dean

Best regards from V.

Re: S&SMC#10: My Original Solutions & Comments

Message #7 Posted by ***Don Shepherd*** *on 7 June 2005, 9:00 p.m.,* *in response to message #4 by Valentin Albillo*

Hi Valentin. Great problem, and I'm going to study your 15c solution in detail because I want to learn more about matrices.

Having just bought a 33s, I wanted to see how it's programming worked, so I attempted to code your challenge on it. My results are below.

I also started by introducing a random 10 digit number and calculating its digit distribution. Then I fed the results back into the algorithm two more times, giving me 3 numbers. If number 3 = number 2, I exit because that would be the magical number. If number 3 = number 1, then I have a pattern ABABAB, so I start again with a new random number. Otherwise I generate the next 3 numbers and test them. The final result is obtained after about 2 or 3 random numbers are used, in general (although it once took 9 random numbers). It is a fascinating problem and I'm going to study it some more.

Anyhow, thanks for the idea.

Don Shepherd

```
Register        Usage
A-J             count of num 0s, 1s, etc.
K               temp holds # being examined
L               constant 10
M               digit distr number
N               first iteration
O               second iteration
P               third iteration



LBL A           entry point
10
10^X            create 10 digit number
RAND            randomly
X
IP              take integer part of it
LBL B
XEQ C           create its digit distribution
29              store in stat register 29
STO i           because C routine clears vars
RDOWN
STO (i)
XEQ C           create dist based on first num
30              store in stat register 30
STO i
RDOWN
STO (i)
XEQ C           create dist based on second num
```

```
STO P           store directly in P
29              restore first two nums to N and O
STO i
RCL (i)
STO N
30
STO i
RCL (i)
STO O
RCL P
X=Y             if O = P, stop you are done
RTN
RCL N           if P = N, two are repeating
X=Y             so get new random number
GTO A
GTO B           get next three numbers
LBL C           given a #, it calcs digit dist
CLRVARS         A-M (stat regs not cleared)
STO K           number to create dist from
10
STO L           constant 10
RDOWN
LBL D           loop to get next digit
RCL L
RMDR
1
+               digit zero maps to Reg A
STO i
1
STO + (i)       increment digit count R(A-J)
RCL K
RCL L
INTGDIV         adjust number to get next digit
STO K
X<>0            loop until all digits mapped
GTO D
RCL L
STO i           start at register J (10) (ones position)
LBL E
RCL (i)         start at reg J and work down to A
RCL L
RCL i
IP
-
10^X            power of 10 multiplier
X
STO + M         add to reg M
```

```
DSE i           work way down to A
GTO E           loop
RCL M           the final digit dist
RTN
```

Re: S&SMC#10: My Original Solutions & Comments

Message #8 Posted by ***Valentin Albillo*** *on 8 June 2005, 6:59 a.m.,*
in response to message #7 by Don Shepherd

Hi, Don:

"Great problem, and I'm going to study your 15c solution in detail because I want to learn more about matrices."

Thanks a lot for your kind comments, and for the 33s program you posted. The algorithm you use is very similar to the one featured in my HP-15C solution. It's actually quite amazing that you managed to write such a program for your 33s right after purchasing it.

As for matrices, it's a real pity that the HP32S, SII, and 33S left out support for them. Matrix operations might seem very specialized at first, but once you understand how they work, they tend to be extremely useful almost for any application, frequently resulting in much more concise, faster programs.

Take this challenge, for example. Who could guest at first sight that using matrix operations would be advantageous ? You would think that a bunch of for-next loops would be needed, but matrices ? Yet representing the numbers as matrices brings the double advantage of being able to deal with their digist individually, as matrix elements, without wasting time extracting them from the original number, and also you can process the number as a whole, using matrix operations that affect the entire matrix.

The best of both worlds, once again. Thanks again for your interest and

Best regards from V.

Re: S&SMC#10: My Original Solutions & Comments

Message #9 Posted by ***Jeff O.*** *on 8 June 2005, 1:15 p.m.,*
in response to message #4 by Valentin Albillo

Valentin,
First, I'll second (or third, or fourth..) the thanks to you for the S&SM Challenges. While I don't always have the time or abilities (or both) to have a crack at them, they are always educational and entertaining.

I did manage to find the solution prior to reading your "Original Solutions & Comments" post above. No clever tricks using matrix capabilities, no knowledge that a recursive approach would lead to a solution. Just brute force; pick a number, test to see if it is the answer, if not, increment the previous guess and check that and so on. I relied on the speed of Free42 to insure a quick solution. My initial simple heuristic was only that the answer must be greater than 1,200,000,000 since anything less than that would start 1,1…, which is logically inconsistent since it has two ones. After determining that my first program based on the above would take about a week to get the answer, I thought about the problem some more and added additional constraints, such as the sum of the digits must equal 10, there can be no nines, eights or sevens anywhere in the answer (which of course means that the answer must end in 000, allowing me to increment my guesses by 1000 rather than by 1), I came up with a program that finds the solution in just under 2 minutes. Again, that's using Free42 on my PC; a real 42S would take ***_much_*** longer (about 3 months according to a quick check). In case anyone is interested, the program listing is below. (On a side note, I guess I did apply too many heuristics to the problem. As I was attempting to determine if sixes could also be eliminated from consideration, I stumbled upon the solution. Sheer luck, I am sure.)

Of course your 15C solution is very clever, capitalizing both on the capabilities of the 15C, and on mathematical knowledge about the problem. With time, I think I can probably figure out the workings of the 15C program. I may attempt to use matrix processing capabilities of the 42S to facilitate crunching the numbers to implement the process that I used to find the answer. However, I'm not sure that I ever would have or could have determined that a recursive procedure in which the initial guesses and new guesses are randomly generated would have ever led to a solution. This leads me to a question. Is the procedure that you implemented mathematically guaranteed to lead to a solution? I guess it seems to me that the procedure depends to some extent on the sequence of numbers generated by the (pseudo) random number generator on the calculator. If an initial guess that is not in the sequence that leads to the solution is never tried, then the solution will not be found, will it? I guess I worry that I might try an initial seed that would result in a lot of "tedious mucking about"[1] in number sequences that would not lead to the solution. Therefore, my tendency would be to start at 0000000001, and count up. This could take a long time, depending on the lowest number that happens to reside in a (or is it the?) sequence that leads to the solution.

Thanks again and best regards.

HP42S (and Free42) program listing:

```
00 { 165-Byte Prgm }
01>LBL "FIND2"
02 RCL "LB"     :recall the current number
03 1E3          :steps 3 and 4 strip off the last three digits since they are
04 ÷            : zero, i.e., no sevens, eights or nines
05 STO "NUM"    :store it in a working register
06 CLRG         :initialize registers that will store and count digits
07 3            :steps 7 and 8 initialize R11 (that will sum the number of
08 STO+ 11      : zeroes) to three to account for the last three digits.
09 7
10 STO "A"      :initialize counter to loop through the 7 remaining digits
11>LBL 01       :routine to break the number into digits and count them
12 RCL "NUM"
13 RCL "NUM"
```

```
14 10
15 MOD          :strip the number in the rightmost position
16 7
17 X<=Y?        :if the number is 7, 8 or 9, branch out of loop to get the next
18 GTO 03       : 10 digit number to check
19 Rv           :Roll down
20 X<>Y
21 10
22 ÷
23 IP
24 STO "NUM"    :divide the number by 10 and store in the working register
25 X<>Y
26 STO IND "A"  :store the digits in R7 through R1, for the 7th through 1st digits
27 STO+ 21      :sum the digits in R21
28 11           :steps 28-31 use the digit itself to point to the registers (11
29 +            : through 17)  that will count the number of times each digit
30 1            : occurs.   For example, if the digit is a 6, a one will be added
31 STO+ IND ST Y: to R17
32 10           :steps 32-35 recall the running sum of the digits and if
33 RCL 21       : greater than 10, branch out of the loop to get the next
34 X>Y?         : 10 digit number to check
35 GTO 03
36 DSE "A"      :decrement the counter and go get the next digit of the current
37 GTO 01       : number
38 10           :all digits checked, the sum must be less than or equal to 10,
39 X!=Y?        : if not equal to 10, go get the next 10 digit number.
40 GTO 03
41 7            :initialize counter to loop through the routine that checks the
42 STO "I"      : digits against the counts of the digits
43>LBL 02       :routine that checks the digits against the counts of the digits
44 RCL "I"
45 10
46 +            :add 10 to the index
47 RCL IND ST X :recall the count of that digit
48 RCL IND "I"  :recall the digit
49 X!=Y?
50 GTO 03       :if they’re not equal, go get the next 10 digit number
51 DSE "I"      :if they are equal, decrement the counter to check the next digit
52 GTO 02       : vs. the count.
53 PRON         :If all digits are equal, i.e. the checking routine is not branched
54 PRV "LB"     : out of, this must be the answer!  Print it out and beep to
55 PROFF        : announce completion
56 BEEP
57 STOP
58>LBL 03       :routine to increment the 10 digit number to get the next one to
59 1E3          : check.  Since there can be no sevens, eight, or nines, the last
60 STO+ "LB"    : three digits are zero and the increment is 1000.
```

```
61 RCL "LB"
62 RCL "UB"     :check the new 10 digit number against an upper bound to
63 X!=Y?        : insure that the program stops someday.
64 GTO "FIND2"  :go to the beginning if the upper bound has not been reached
65 PRON
66 PRX          :if the upper bound is reached, print it out and stop to indicate
67 PROFF        : that no solution was found.
68 END
```

1 – due credit thankfully given to the late, great Douglas Adams.

Re: S&SMC#10: My Original Solutions & Comments

Message #10 Posted by ***Arnaud Amiel*** *on 8 June 2005, 1:52 p.m.,*
in response to message #9 by Jeff O.

I was using a recursive algorithm and was also worried that it might not converge to a solution. It happened that 0000000000 did converge and all the few numbers I tried so I assumed it would always converge and did not check for non convergence. I was actually thinking of writing a little C program to go through all 10 digit numbers and see which did converge and which did not. But I might not do it before next week end.

Arnaud

Re: S&SMC#10: My Original Solutions & Comments

Message #11 Posted by ***Valentin Albillo*** *on 9 June 2005, 5:14 a.m.,*
in response to message #9 by Jeff O.

Hi, Jeff:

Jeff posted:

"First, I'll second (or third, or fourth..) the thanks to you for the S&SM Challenges."

Why, thank you, you're all so very kind. I certainly appreciate it when you people post your inputs to my S&SMCs, because that's the only way I have to know that my humble efforts are indeed reaching you and contributing to the share of enjoyment and knowledge this MoHP Forum actually is.

"No clever tricks using matrix capabilities, no knowledge that a recursive approach would lead to a solution. Just brute force"

Actually, the iterative method I used isn't a trick, but a genuine, useful method to solve a large variety of equations than can be expressed as **f(x) = x**, specially when said equations aren't amenable to other faster, well-known methods such as Newton's method, etc.

The only 'bright' idea needed was to recognize that solving this puzzle was tantamount to solving f(N)=N, where f(N) was the function counting the digits in a 10-digit N and returning them assembled as another 10-digit number.

As for brute force, this would be a straightforward brute force approach for the HP-71B, using a minimum of heuristics, namely that the digits of the number must add exactly to 10:

```
10 DESTROY ALL @ OPTION BASE 0 @ DIM N(9),M(9)
20 FOR A=9 TO 1 STEP -1 @ N(0)=A
30 FOR B=9 TO 0 STEP -1 @ N(1)=B @ S=A+B @ IF S>10 THEN 210
40 FOR C=5 TO 0 STEP -1 @ N(2)=C @ T=S+C @ IF T>10 THEN 200
50 FOR D=3 TO 0 STEP -1 @ N(3)=D @ U=T+D @ IF U>10 THEN 190
60 FOR E=2 TO 0 STEP -1 @ N(4)=E @ V=U+E @ IF V>10 THEN 180
70 FOR F=2 TO 0 STEP -1 @ N(5)=F @ W=V+F @ IF W>10 THEN 170
80 FOR G=1 TO 0 STEP -1 @ N(6)=G @ X=W+G @ IF X>10 THEN 160
90 FOR H=1 TO 0 STEP -1 @ N(7)=H @ Y=X+H @ IF Y>10 THEN 150
100 FOR I=1 TO 0 STEP -1 @ N(8)=I @ Z=Y+I @ IF Z>10 THEN 140
110 N(9)=10-Z @ MAT M=ZER @ FOR J=0 TO 9 @ M(N(J))=M(N(J))+1 @ NEXT J
120 MAT M=M-N @ IF RNORM(M) THEN 140
130 FOR J=0 TO 9 @ DISP N(J); @ NEXT J @ END
140 NEXT I
150 NEXT H
160 NEXT G
170 NEXT F
180 NEXT E
190 NEXT D
200 NEXT C
210 NEXT B @ NEXT A @ DISP "No solution"
```

Running it on a physical HP-71B returns **6210001000** in 8.03 seconds (0.91 seconds under Emu71). You can very easily adapt this to your HP42S, and it will run pretty fast there.

"With time, I think I can probably figure out the workings of the 15C program. I may attempt to use matrix processing capabilities of the 42S"

It's actually quite easy to understand, and translates most easily to HP42S RPN and matrix capabilities. Try it !

"Is the procedure that you implemented mathematically guaranteed to lead to a solution?"

Yes, it is, but the actual, rigurous proof of it is quite cumbersome and lengthy. An informal argument would be as follows: the statistical distribution of the randomly generated initial guesses entirely depends on the properties of the HP-15C's RNG, and this one passes the spectral test and has a cycle length large enought that eventually every 10-digit number would be generated with probability 1. This means that, in the very worst case that only stumbling directly upon the solution would result in a 1-cycle, you're guaranteed that this would happen. Fortunately, it seems likely (and a little testing quickly demonstrates) that there are indeed many other initial guesses which quickly lead to the solution, so the probabilities increase to the point where finding one of them by chance has a large probability.

I did some empirical tests, trying out up to 10,000 random initial guesses, and as stated in a previous post, you'd only generate 13.5 numbers (average) before finding the solution, very frequently after just one or two restarts (new random initial guess). Many times I got by with only 3 numbers generated, one or two times it would need 100. And, of course, checking at most 100 numbers (out of the 9E9 possible) is incredibly fast on any decent machine, let alone checking just 13 or 3.

Thanks again for your interest and

Best regards from V.

Re: S&SMC#10: My Original Solutions & Comments

Message #12 Posted by ***Jeff O.*** *on 13 June 2005, 7:36 a.m.,*
in response to message #11 by Valentin Albillo

> Quote:
>
> Running it on a physical HP-71B returns **6210001000** in 8.03 seconds...

Valentin,
I typed your program into my 71B. After about 10 seconds of running with no response, I figured I must have made an error somewhere. However, I let it continue running. Some time later, I looked at the display. It had stopped running and was displaying the answer. I ran it again and timed it, finding that it takes approximately 1 minute, 45 seconds to find the answer. Is there some reason your physical 71B would run the program 12 times faster than mine?

Re: S&SMC#10: My Original Solutions & Comments

Message #13 Posted by ***Valentin Albillo*** *on 14 June 2005, 9:21 a.m.,*
in response to message #12 by Jeff O.

Hi Jeff,

My mistake, your timing is correct. When copying & pasting the results for my posting, I mistakenly pasted the timing for a previous, slower version under Emu71, instead of the correct timing for a physical HP-71B.

Thanks por pointing this out, and I'm sorry for any resulting confusion.

Best regards from V.

Edited: 14 June 2005, 9:21 a.m.

[Return to Index | Top of Index]

GTO Go back to the main exhibit hall