HP Forum Archive 17

[Return to Index | Top of Index]

HP-15C Mini-Challenge: Bits o'Pi

Message #1 Posted by ***Valentin Albillo*** *on 7 Mar 2007, 9:12 a.m.*

Hi, all:

A new month's just begun, the Spring is nearing, and before posting my traditional ***"S&SMC 2007 Spring's Special"*** next April, 1st, I'll submit for your consideration this petite & nifty

HP-15C Mini-Challenge: Bits o' Pi

(Well, not actually of Pi proper but it's *reciprocal*, **1/Pi** = .31830988618379...)

You're asked to write a program for the **HP-15C**, which must be *as short as possible*, take no input, and upon running *it must produce in order the bits of 1/Pi*, i.e.: the binary digits of 1/Pi when expressed in base 2.

Notes:

- As 1/Pi = .31830988618379... = .010100010... in base 2, your program should work like this:

  ```
  GSB A -> 0 -> 1 -> 0 -> 1 -> 0 -> 0 -> 0 -> 1 -> 0 -> ...
  ```

- The HP-15C being a 10-decimal digit calculator, your program is expected to produce *at least the first **30** correct bits of 1/Pi* (but the more the merrier of course, subject to minimum program size).

- Your program can assume any necessary display or angular modes are in effect (such as FIX 0 or radians mode, etc.) if required, no need to set them up in code. It must not assume any other requirements at all, such as specific registers' or stack's contents.

Extra-Goals !

As Bill has quickly succeeded in achieving the original goal with his 11-step, 31-bit solution (see his post below; congratulations, Bill !), I've added the three following additional goals, namely:

- **Goal 2:**

  To improve upon Bill's solution to get ***more than 31 bits*** in 11-steps or less.

- **Goal 3:**

  To get a solution in 11 steps or less and which can produce at least 30 correct bits, ***without using the [Pi] function***.

- **Goal 4:**

  To get a solution in 11 steps or less and which can produce at least 30 correct bits, ***using neither the [Pi] function nor the [1/x] function***.

Well, give it a try. It may seem simple, even trivial, but let me assure you that there's more than meets the eye, and you'll be hard pressed to come up with the *shortest* possible HP-15C program to deliver what's asked.

If you don't have a physical HP-15C at hand, you can download Nonpareil for free, which provides a perfect HP-15 virtual replica, and you're of course welcome to use any other HP (or SHARP, say) model although in such case the shortest code will certainly vary a lot.

As always, within a few days I'll post my original solution (plus comments) which is an ***11-step*** *HP-15C-specific program* (including LBL A at the beginning but no RTN instruction at the end, as always) which meets the above requirements and correctly produces *the first **34** bits of 1/Pi*

Note:

For the sake of it, I'll also post a *27-character command-line expression* for the HP-71B (i.e., slightly over 1/4th of a full-length line) which outputs even more correct bits; no ROMs required, just a bare-bones HP-71B. See if you can duplicate that as well ! :-).

Best regards from V.

Edited: 7 Mar 2007, 1:17 p.m. after one or more responses were posted

Re: HP-15C Mini-Challenge: Bits o'Pi

*Message #2 Posted by **Bill (Smithville, NJ)** on 7 Mar 2007, 12:28 p.m.,*
in response to message #1 by Valentin Albillo

Hi Valentin,

Since I used the snow today as an excuse to skip work, I get a chance to try one of your wonderful HP-15C mini-challenges.

The following is an 11 line program for the HP-15C which is good to 31 digits.

```
LBL A
PI
1/X
LBL B
2
*
ENTER
INT
R/S
-
GTO B
```

Thanks again for an enjoyable challenge.

Bill

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #3 Posted by ***Valentin Albillo*** *on 7 Mar 2007, 1:01 p.m.,*
in response to message #2 by Bill (Smithville, NJ)

Hi, Bill:

> Congratulations on your quick solution, which perfectly meets the mini-challenge requirements ! You're a winner ! :-)
>
> Just to give you and the rest of interested people extra fun, I've added three new additional goals directly edited in my original post above which you might want to try :-)
>
> Thanks again, I hope you'll get interested in the additional goals and

Best regards from V.

Edited: 7 Mar 2007, 1:14 p.m.

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #4 Posted by ***hugh steers*** *on 8 Mar 2007, 4:07 a.m.,*
in response to message #3 by Valentin Albillo

Nice one Bill!

i like bill's answer. in fact, if he's allowed to also assume FIX 0. can't his solution be presented in 10 steps as,

```
LBL A
2
PI
/
LBL B
R/S
FRAC
2
*
GOTO B
```

furthermore, by taking the 2/pi out, this form should be more accurate than his original (don't have my 15 here to test unfortunately). ??

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #5 Posted by ***Bill (Smithville, NJ)*** *on 8 Mar 2007, 7:26 a.m.,*
in response to message #4 by hugh steers

Hi Hugh,

> Quote:
>
> Nice one Bill!

Thanks. I normally don't have the time to do Valentin's challenges justice.

> Quote:
>
> in fact, if he's allowed to also assume FIX 0

Unfortunately, the display is rounded up, so 1.55 displays as 2 and not binary digit 1. Likewise, 0.55 would display as 1 instead of 0.

Bill

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #6 Posted by ***Bill (Smithville, NJ)*** *on 8 Mar 2007, 10:47 a.m.,*
in response to message #1 by Valentin Albillo

I've tried a couple of PC programs to calculate the binary sequence for 1/PI and have come up with the following for the first 50 bits:

```
Bits 1-10     0101000101
Bits 11-20    1111001100
Bits 21 - 30  0001101101
Bits 31 - 40  1100100111
Bits 41 - 50  0010001000
```

Could someone please verify that I have it correct?

Thanks,

Bill

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #7 Posted by ***GE*** *on 8 Mar 2007, 12:23 p.m.,*
in response to message #6 by Bill (Smithville, NJ)

I see an interesting formula on http://mathworld.wolfram.com/PiFormulas.html :
1/pi = Sum(n=0,+inf,Comb(2n,n)^3*(42n+5)/2^(12n+4))
The point is that the denominator is a power of 2, thus a sinple shift.
No idea on how to apply it on the 15C, however, as the conversion of the numerator [Comb(2n,n)^3*(42n+5)] into binary would be quite a feat IMHO.

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #8 Posted by ***GE*** *on 20 Mar 2007, 5:14 a.m.,*
in response to message #7 by GE

No takers ? I typed that formula and it doesn't work well.
Actually the term in that sum actually diverges !!
Thinking that this is "An infinite sum due to Ramanujan" (formula number 74), I wonder if Ramanujan was wrong, which is possible, or if Wolfram typed it

wrong.
An uneasy feeling in any case, first time I find an error on the Mathematica Web site.

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #9 Posted by ***Rodger Rosenbaum*** *on 20 Mar 2007, 6:07 a.m.,*
in response to message #8 by GE

I don't get the result that you do.

I find that the first 7 terms give 12 accurate digits of 1/PI.

They are: {.3125,.0057373,.000071615,9.5315E-7,1.31759E-8,1.86518E-10,2.6833E-12}

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #10 Posted by ***Valentin Albillo*** *on 20 Mar 2007, 7:15 a.m.,*
in response to message #8 by GE

Hi, GE:

GE posted:

> *"I typed that formula and it doesn't work well."*

You don't work well ! :-) Try this straightforward HP-71B implementation:

```
10 DEF FNC(M,N)=FACT(M)/FACT(N)/FACT(M-N)


20 DESTROY ALL @ S=0 @ FOR N=0 TO 10 @ S=S+FNC(2*N,N)^3*(42*N+5)/2^(12*N+4)
30 NEXT N @ DISP S,1/S


 >RUN


   .318309886185       3.14159265358
```

You should really double-check your work before publicly labeling other people's as wrong :-)

Best regards from V.

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #11 Posted by ***GE*** *on 21 Mar 2007, 9:27 a.m.,*
in response to message #10 by Valentin Albillo

Sorry for some unwarranted criticism, I looked again at the convergence of terms and searched for the limit of the term for n->infinity. After more research I found it to be equivalent to 2^(-6n) which DOESN'T diverge...
So you're right this series has a limit.
I'll try to program this in BASIC.

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #12 Posted by ***Valentin Albillo*** *on 8 Mar 2007, 8:37 p.m.,*
in response to message #6 by Bill (Smithville, NJ)

Hi, Bill:

Your 50 bits are correct and there you are, 50 more:

```
   1-10   0101000101
  11-20   1111001100
  21-30   0001101101
  31-40   1100100111
  41-50   0010001000


  51-60   0010101001
  61-70   0100111111
  71-80   1000010011
  81-90   1010101111
 91-100   1010001111
```

These were produced by my program running in a multiprecision environment. The 11-step HP-15C version can produce the first 34 bits *(in bold face above)* correctly.

Best regards from V.

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #13 Posted by ***Karl Schneider*** *on 9 Mar 2007, 12:30 a.m.,*
in response to message #12 by Valentin Albillo

One way to get the 11th and 12th significant digits of pi from an HP-15C (or HP-41, HP-10C, HP-11C, or Spice-series scientific) is to take the sine of pi radians. The second digit is rounded to 9.

Now, how to incorporate that without adding steps is not clear to me, so there might be another way.

-- KS

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #14 Posted by ***Bill (Smithville, NJ)*** *on 8 Mar 2007, 11:45 a.m.,*
in response to message #1 by Valentin Albillo

Hi Valentin,

For Goal 2 & 3, I have a 12 step solution that gives 33 bits without use of PI.

```
RAD MODE


LBL A
1
ASIN
1/X
LBL B
ENTER
INT
R/S
-
2
*
GTO B
```

Now how do I get rid of that 12th step :)

And how do I do it without 1/X.

Something to mull over during lunch hour.

Bill

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #15 Posted by ***Gerson W. Barbosa*** *on 9 Mar 2007, 7:42 a.m.,*
in response to message #14 by Bill (Smithville, NJ)

Hi Bill,

Congratulations for you nice routines!

> Quote:
>
> ---
>
> Now how do I get rid of that 12th step :)
>
> ---

I wish I knew! Anyway, another 12-step solution, in case it might be helpful:

```
RAD MODE


LBL A
CLEAR SIGMA
ACOS
1/x
LBL 0
STO+ 2
RCL 2
INT
R/S
STO- 2
RCL 2
GTO 0
```

Regards,

Gerson.

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #16 Posted by ***Les Wright*** *on 9 Mar 2007, 8:09 a.m.,*
in response to message #14 by Bill (Smithville, NJ)

> Quote:
>
> ---
>
> Now how do I get rid of that 12th step :)

Isn't that the most important one of all, the spiritual awakening and passing on the message to others?

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #17 Posted by ***Valentin Albillo*** *on 9 Mar 2007, 8:40 a.m.,*
in response to message #16 by Les Wright

Les wrote:

> *"Isn't that the most important one of all, the spiritual awakening and passing on the message to others?"*

Now, Les, you should go see a psychiatrist (preferably one who has taken time away from full-time practice to study music) :-)

Best regards from V.

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #18 Posted by ***Karl Schneider*** *on 10 Mar 2007, 7:31 p.m.,*
in response to message #1 by Valentin Albillo

There are quite a number of ways to calculate pi or its inverse without using the "pi" or "1/x" functions. Some of these can even utilize the complex-number mode, such as

```
(flag 8 clear)


2
ENTER
Re<->Im
COS-1
CF 8
*
```

gives pi in any angular mode.

Since not to use those functions would add steps, obtaining solutions to the additional goals would lengthen the programs. If there is a super-simple way to calculate the specific digits of a number, I don't know what it is.

Instead, I will utilize some of the HP-15C's built-in programming that can minimize the total amount of code for three of the objectives. The driver program in each is less than 11 steps, not counting the modified "Bill" program that is run as a subroutine labeled in his honor as "B". :-)

```
(in radians mode)


LBL B
2
*
ENTER
INT
R/S
-
x=0?
RTN
GTO B


Goal 1    Goal 2    Goal 3    Goal 4


LBL A     LBL C     LBL D     LBL E
PI        ????      1         1
1/x       RTN       CHS       CHS
GSB B               COS-1      ENTER
RTN                 1/x       COS-1
                    GSB B     x<>y
                    RTN       yx
                              GSB B
                              RTN
```

Granted, these may not be admissible solutions, but do show some thought...

Edited: 10 Mar 2007, 7:55 p.m.

Re: HP-15C Mini-Challenge: Bits o'Pi

*Message #19 Posted by **Gerson W. Barbosa** on 10 Mar 2007, 8:49 p.m.,*
in response to message #18 by Karl Schneider

Hello, Karl!

Quote:

> There are quite a number of ways to calculate pi or its inverse without using the "pi" or "1/x" functions.

```
  .        .        1
  5        5        8
  x!      CHS       0
 x^2       x!     ->RAD
  4       x^2
  *
```

Also give pi, regardless of the angular mode and flag settings. With so many extra steps, none of these is likely to be part of Valentin's solution though.

Best regards,

Gerson.

Edited: 11 Mar 2007, 9:37 p.m.

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #20 Posted by ***Bill (Smithville, NJ)*** *on 10 Mar 2007, 11:28 p.m.,*
in response to message #18 by Karl Schneider

Hi Karl,

> Quote:
>
> not counting the modified "Bill" program that is run as a subroutine labeled in his honor as "B". :-)

Thank you. I am honored :)

I've come up with a 13 step program that meets Goal 4 and is good to 33 bits:

```
RAD MODE


LBL A
CLEAR SIGMA
ACOS
SIGMA+
RCL / 3
```

```
LBL B
ENTER
INT
R/S
-
2
*
GTO B
```

Still too many steps. I just can't see how to reduce them.

Bill

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #21 Posted by ***Arnaud Amiel*** *on 11 Mar 2007, 4:27 a.m.,*
in response to message #1 by Valentin Albillo

I am a bit late on this one and would not like to look stupid but I have absolutely no idea as to how to get the bits of a non integer. Any pointer as to how these are defined would be welcome.

Arnaud

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #22 Posted by ***Egan Ford*** *on 11 Mar 2007, 10:19 a.m.,*
in response to message #21 by Arnaud Amiel

> Quote:
>
> I am a bit late on this one and would not like to look stupid but I have absolutely no idea as to how to get the bits of a non integer. Any pointer as to how these are defined would be welcome.

http://mathforum.org/library/drmath/view/55744.html

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #23 Posted by ***Gerson W. Barbosa*** *on 11 Mar 2007, 12:02 p.m.,*
in response to message #22 by Egan Ford

Quote:

http://mathforum.org/library/drmath/view/55744.html

Very good link, thanks!

A straightforward implementation of the algorithm therein yields yet another 11-step solution for the first 31 bits:

```
LBL A
 pi
 1/x
LBL 0
ENTER
  +
 INT
 R/S
 LSTx
 FRAC
GTO 0
```

Regards,

Gerson.

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #24 Posted by ***Bill (Smithville, NJ)*** *on 11 Mar 2007, 12:58 p.m.,*
in response to message #22 by Egan Ford

Here's another good link:

Converting Decimal Fractions to Binary

Bill

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #25 Posted by ***Gerson W. Barbosa*** *on 11 Mar 2007, 6:27 p.m.,*
in response to message #1 by Valentin Albillo

Hello Valentin,

Quote:

> I'll also post a *27-character command-line expression* for the HP-71B

This is not so short as yours, but works:

```
X=1/PI@KEY"A",'X=2*X@DISPMOD(IP(X),2)':
```

After pressing ENDLINE each successive press of the "A" key on the user keyboard (f USER), will display the following correct bits, one at a time:

```
010100010111110011000001101101110010011
```

39 bits, the same lenght of the command-line expression! Woudn't it be better just hard-coding the bit string? :-)

Best regards,

Gerson.

Edited: 11 Mar 2007, 6:28 p.m.

Re: HP-15C Mini-Challenge: Bits o'Pi

*Message #26 Posted by **Valentin Albillo** on 11 Mar 2007, 7:00 p.m.,*
in response to message #25 by Gerson W. Barbosa

Hi, Gerson:

Gerson posted:

> *"39 bits, the same lenght of the command-line expression! Woudn't it be better just hard-coding the bit string? :-)"*

> Possibly, but the 27-character solution is shorter than the equivalent string formed by the concatenation of the "0"s and "1"s it outputs.
>
> BTW, the 27-char HP-71B version, being just my HP-15C's 11-step RPN program translated to HP-71B's BASIC native language, it also lacks both division ("1/X") and PI. It includes a conditional, though :-)
>
> Thanks for your interesting idea, see if the above gives you any usable hints and

Best regards from V.

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #27 Posted by ***Gerson W. Barbosa*** *on 11 Mar 2007, 8:43 p.m.,*
in response to message #26 by Valentin Albillo

Hello again Valentin,

> Quote:
>
> the 27-character solution is shorter than the equivalent string formed by the concatenation of the "0"s and "1"s it outputs.

When I talked about hard-coding the binary string I was referring to my *lengthy* solution.

> Quote:
>
> Thanks for your interesting idea

Actually, rather than an idea that was the result of an experimentation. I took 1/pi and observed what happened to the integer part each time it was doubled. It was easy to notice that when the integer part was odd then the corresponding binary digit was "1"; when it was even the binary digit was "0":

```
0.318309886 * 2 ->  0.636619772  -> 0
                ->  1.273239544  -> 1
                ->  2.546479088  -> 0
                ->  5.092958176  -> 1
                -> 10.18591635   -> 0
                -> 20.37183270   -> 0
                -> 40.74366540   -> 0
                -> 81.48733080   -> 1


                    and so on...
```

Hence,

```
0.318309886 (base 10) = 0.01010001 (base 2).
```

So, the first *m* binary digits of a fractional base 10 number *n* can be obtained with this simple algorithm:

```
k := 0;
repeat
  k := k + 1;
  n := n * 2;
  d_k := mod(int(n),2)
until k = m
```

Unfortunately, it may not be so efficient on the HP-15C, due to the lack of MOD function. I don't think an 11-step solution is possible even on the HP-33S, which has the RMDR function. Anyway, it appears to be simpler than other algorithms. Of course, this has yet to be proven, but I'll leave that as an exercise for the interested reader :-)

Best regards,

Gerson.

P.S.:

I've just tested the algorithm on Turbo Pascal 3 (does anyone still remember it? :-)

Here is the result:

```
program dec2bin;
var k,m: integer;
      n: real;
begin
  k:=0;
  m:=16;
  n:=0.3183098861;
  ClrScr;
  WriteLn(n:11:10);
  WriteLn;
  repeat
    k:=k+1;
    n:=n*2;
    Write(trunc(n) mod 2:1)
  until k=m
end.


----------------


0.3183098861


0101000101111100
```

Edited: 11 Mar 2007, 9:46 p.m.

Re: HP-15C Mini-Challenge: Bits o'Pi

*Message #28 Posted by **Bill (Smithville, NJ)** on 12 Mar 2007, 7:14 a.m.,*
in response to message #27 by Gerson W. Barbosa

Hi Gerson,

> Quote:
>
> I've just tested the algorithm on Turbo Pascal 3 (does anyone still remember it? :-)

Sure do - I still use it on a regular basis and is one of the programs I keep on a USB flash drive along with several other useful utilities. I used Turbo Pascal 6 with Extended Real to calculate the first 50 bits of this challenge.

But Turbo Pascal 3 is my favorite for developing quick and dirty one-off type programs. Also use it a lot to create quick data filtering programs. Glad to see someone else still uses it.

Bill

Re: HP-15C Mini-Challenge: Bits o'Pi

*Message #29 Posted by **Valentin Albillo** on 12 Mar 2007, 8:27 a.m.,*
in response to message #28 by Bill (Smithville, NJ)

Hi, Bill & Gerson:

> I also do, and exactly for the very same reasons: it's about the smallest, simplest way of quickly developing and running a "console" executable anywhere, with no installation and minimum size, in any Windows system. It's ideal for small utilities to do the kind of tasks you would do with awk in Unix systems, for instance.
>
> The one and only problem is it doesn't deal with long filenames, which is kind of a nuisance. But apart from that, it's really useful, capable and fast.

> I usually begin all such created routines by first checking paramcount and, if no parameters are supplied, I immediately print its description and the correct syntax to call the routine. I also allow for input and output piping to enhance the usefulness if the routine is to be used more than once, thus allowing the routine to work like a filter. For instance:
>
> ```
> C:>trimfile
>
>        SYNTAX: trimfile <ascii file to trim> <# of columns>
>
> C:\trimfile  somedata.log  40 > trimdata.log
>
>        Ok, done.
> ```

Best regards from V.

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #30 Posted by ***Gerson W. Barbosa*** *on 12 Mar 2007, 8:48 a.m.,*
in response to message #28 by Bill (Smithville, NJ)

7718.07734 (*)

> Quote:
>
> Glad to see someone else still uses it.

Pascal was practically the first programming language I was introduced to in 1984, in the DEC-10 environment. Unfortunately, I don't use it so often as I should (I had to check the MOD syntax in the book). Unlike many here, my job is not related to information technology. But it's nice remembering things and learning from you and many other people here.

Best regards,

Gerson.

(*) You won't have trouble decyphering this one too :-)

Edited: 12 Mar 2007, 9:04 a.m.

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #31 Posted by ***Bill (Smithville, NJ)*** *on 13 Mar 2007, 7:41 a.m.,*
in response to message #30 by Gerson W. Barbosa

Hi Gerson,

> Quote:
>
> 7718.07734 (*)
>
> (*) You won't have trouble decyphering this one too :-)

Took me a second or two and then it hit me :)

Also broght back a lot of memories. Years ago, several of my co-workers and I wasted a whole afternoon. Someone had brought in a small book - may have been called "Fun with your Calculator" or "Calculator Fun" - something like that. It had a whole bunch of joke type problems where you'd do the calculations in the joke and then turn the calculator over to read the punch line in the display.

Great fun.

Bill

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #32 Posted by ***Egan Ford*** *on 12 Mar 2007, 10:05 p.m.,*
in response to message #1 by Valentin Albillo

Here is my 12 step solution for all goals:

```
1 LSTx
2 FRAC
3 ENTER
4 +
5 INT
6 RTN
7 LBL A
8 1
```

```
  9 ENTER
 10 ASIN
 11 /
 12 INT
```

Just need to lose a step.

Re: HP-15C Mini-Challenge: Bits o'Pi

Message #33 Posted by ***Valentin Albillo*** *on 13 Mar 2007, 8:02 a.m.,*
in response to message #32 by Egan Ford

Hi, Egan:

Congratulations for your extremely ingenious near-solution ! The idea of placing the entry point (LBL A) mid-program is really neat, and works perfectly, up to 33 correct bits, while meeting all four goals save for the fact that it's 12 steps long, not 11 as requested.

That's where the real challenge is, of course ! :-) Some comments, which might perhaps give some usable hints (or not):

- *None* of the solutions or near-solutions already posted have stumbled upon my original challenge-solving algorithm.
- Your solution in particular isn't HP-15C-specific (it could perfectly run on other models), while the challenge-solving solution actually *is*. That's no fault of your solution, I just want to point out the fact that making use of HP-15C-specific functionalities might help decrease the number of steps.
- It may come as a surprise but not only does my challenge-solving solution make no use of either [Pi] or [1/X] instructions, as stated in goals 3 and 4, but also uses none of the *ersatz* methods to get Pi already posted, such as ASIN, GAMMA, or even DEG-RAD conversions ! Come to that, no division instructions of any kind are present at all either.

Thanks again for your excellent idea and

Best regards from V.

Ersatz pi? [HP-15C Mini-Challenge: Bits o'Pi]

Message #34 Posted by ***Karl Schneider*** *on 14 Mar 2007, 1:24 a.m.,*
in response to message #33 by Valentin Albillo

Quote:

...but also uses none of the *ersatz* methods to get Pi already posted, such as ASIN, GAMMA, or even DEG-RAD conversions...

"Ersatz"? Hey, what about ACOS? :-)

They all give answers identical to that of the pi function, although the tenth decimal digit is incorrect in all cases. Of course, it's 1/pi that matters.

Here's one that requires only the input value 1/2 and the square root function:

1/pi = 0.5 * sqrt(0.5) * sqrt(0.5 + 0.5*sqrt(0.5)) * sqrt[0.5 + 0.5*sqrt(0.5 + 0.5*sqrt(0.5))] * ...

But, I doubt that this is on the right track...

-- KS

The last hint (was Re: Ersatz pi? [...])

Message #35 Posted by ***Valentin Albillo*** *on 14 Mar 2007, 10:34 a.m.,*
in response to message #34 by Karl Schneider

Hi, Karl:

Karl posted:

> *"Ersatz"? Hey, what about ACOS? :-)"*
>
>> ASIN, ACOS, ATAN, ..., none of them is relevant in this case.
>>
>> The heart of the matter is (and this is the last 'hint' I'll give before finally posting my original solution), that most people trying this mini-challenge out *automatically assume* that they must *first* generate the decimal value of 1/Pi in order to *then, and only then*, proceed to convert it to its binary form and output its bits, and of course, in the case of Goals 3 and 4, they do their best to try and generate 1/Pi using as few steps as possible and do likewise with the conversion loop, only to discover thay they can't lower the step count to the 11-step requisite.
>>
>> The key is: *there's no need to generate a decimal version of 1/Pi first*, a way must be sought to actually generate its binary form *directly*, without ever having a decimal Pi or 1/Pi to work with in the first place.
>>
>> Once you realize this you discover you can save all steps to get Pi or 1/Pi and this is what ultimately allows you to achive the 11-step goals, with the nice side effect of getting 34 correct bits in the bargain.
>
> *Here's one that requires only the input value 1/2 and the square root function: [...] But, I doubt that this is on the right track..."*

> Pretty, indeed; it's the well-known, old Vieta formula. And if *"on the right track"* means re this mini-challenge, certainly it's got nothing to do whatsoever. :-)

I hope the above 'last hint' sheds some really new light on the subject, and I'll await a couple of days to see if interested people like yourself take it for good and produce the ultimate solution. :-)

Thanks a lot for your comments and always interesting postings in each and every thread you post in.

Best regards from V.

Re: The last hint (was Re: Ersatz pi? [...])

Message #36 Posted by ***Egan Ford*** *on 14 Mar 2007, 12:33 p.m.,*
in response to message #35 by Valentin Albillo

> Quote:
>
> The key is: *there's no need to generate a decimal version of 1/Pi first*, a way must be sought to actually generate its binary form *directly*, without ever having a decimal Pi or 1/Pi to work with in the first place.

This had occurred to me and I am able to generate the same sequence of bits, but it is not close to 11 lines.

Post it, nevertheless ...

Message #37 Posted by ***Valentin Albillo*** *on 14 Mar 2007, 1:07 p.m.,*
in response to message #36 by Egan Ford

It will be interesting and perhaps someone can improve upon it after being aware of your basic idea. That's synergy !

Best regards from V.

Re: Post it, nevertheless ...

Message #38 Posted by ***Egan Ford*** *on 14 Mar 2007, 1:19 p.m.,*
in response to message #37 by Valentin Albillo

IANS convert this number:

3984801418

to base 2 starting with least significant. The challenge is getting 3984801418. That number alone is 10 steps.

I dumped the code shortly after writing it.

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #39 Posted by ***Valentin Albillo*** *on 16 Mar 2007, 7:15 a.m.,*
in response to message #1 by Valentin Albillo

Hi, all

Thanks for your interest in this HP-15C Mini-Challenge to both posters & lurkers, we've seen a very good early solution to the initial goal by ***Bill (Smithville, NJ)*** and some truly ingenious attempts to try and achieve the remaining three goals, which came pretty close to solve them all but for a single step, as well as worthy food-for-thought comments by ***Gerson*** and ***Karl***.

As I told in a number of 'hints', the key lies in *avoiding* the seemingly obvious strategy of first computing the *decimal* value of **1/Pi** in order to then convert it to binary. This leads to very good solutions but not in 11 steps, there's simply not margin enough for that.

What must be done then is to use some algorithm that can compute the binary bits of 1/Pi *directly*, without ever needing its decimal form in the first place. Such a convenient algorithm does exist and, most awesomely, Pi never explicitly appears at any stage but, implicitly, at the final output.

My original 11-step solution for all 4 goals is thus the following HP-15C-specific routine:

```
01  LBL A
02  MATRIX 1
03  LBL 0
04  RCL 0
05  STO+ 0
06  TAN
07   0
08  TEST 7  (X>Y?)
09   1
10  PSE     (or R/S)
11  GTO 0
```

which, upon running, produces the *first 34 correct bits* of 1/Pi:

```
RAD, FIX 0  { set radians and FIX 0 display mode }
```

```
GSB A -> 0  1  0  1  0  0  0  1  0  1
         1  1  1  1  0  0  1  1  0  0
         0  0  0  1  1  0  1  1  0  1
         1  1  0  0
```

Notice that it doesn't explicitly use Pi and inverse trigonometric functions are also notably absent. The maximum precision obtainable depends on both the HP-15C being a 10-decimal digit calculator (13-digit internally) and the maximum accuracy of the TAN function for large arguments, but it suffices to correctly compute up to 34 base-2 digits which is roughly equivalent to 10 base-10 digits, as expected. Also, the only thing that makes it HP-15C-specific is the **MATRIX 1** instruction, so by replacing it by **1, STO 0** you'll have a 12-step solution for many RPN models.

This RPN routine can be trivially converted to a 27-character command line for the HP-71B, namely (in RADIANS and STD modes):

```
FORI=0TO40@TAN(2^I)<0@NEXTI
```

which upon running produces the first 41 correct bits of 1/Pi.

Can the RPN routine be improved further ? Yes ! By using the *mid-program entry point* technique posted by ***Egan Ford*** with my algorithm, *two additional steps can be saved* while keeping the algorithm and maximum achievable number of correct bits intact, thus resulting in an ultimate *9-step, 34-correct-bit RPN solution* for the HP-15C (10-step for many other RPN models) ! How's that for maximum results with a minimum of steps ? :-)

```
01  RCL 0
02  STO+ 0
03  TAN
04   0
05  TEST 7
06   1
07  RTN
08  LBL A
09  MATRIX 1
```

The output sequence is now (after RAD, FIX 0, as always):

```
GSB A, R/S -> 0
       R/S -> 1
       R/S -> 0
       ...
```

the only difference being the additional R/S necessary after GSB A to output the first bit.

Next April, 1st I'll post my ***S&SMC Spring 2007 Special***, I hope to 'see' you there (if you think this Mini-Challenge is pretty unusual math, wait till you see the S&SMC ! :-)).

Thanks for your continued interest and

Best regards from V.

Edited for minor grammatical corrections

Edited: 16 Mar 2007, 10:01 a.m.

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #40 Posted by ***Gerson W. Barbosa*** *on 16 Mar 2007, 3:33 p.m.,*
in response to message #39 by Valentin Albillo

Hi Valentin,

Very nice original solution, as always. Thanks for taking the time to think of interesting problems and presenting them here. It's nice to see what team work, or synergy, can do: the resulting 9-step program is just amazing!

Now, a small variation in your challenge: imagine someone has asked for a binary expansion of, say, Plouffe's b-constant, and it has to be achieved in only 9 steps on an HP-15C. Impossible? Not at all! Just replace TAN with COS in Albillo-Ford's wonderful routine. At least the first 36 bits appears to be correct!

Now, what is Plouffe's b-constant anyway? I don't know, but thanks to your original solution, I can compute it on the HP-71B :-)

```
10 S=0
20 FOR I = 0 TO 40
30 IF COS(2^I)<0 THEN S=S+1/2^(I+1)
40 NEXT I
50 DISP S
```

This gives

```
.475626076737
```

The first 11 decimal digits are correct. Of course, there's a slight chance this is just a coincidence... Could someone compute 50 or more bits? On QBASIC I can compute only 32 bits (a little more on the HP-71B) :

```
'Binary expansion of Plouffe's b-constant
CLS
DEFDBL A-Z
S = 0
FOR I = 0 TO 31
  PRINT ABS((COS(2 ^ I)) < 0);
  IF COS(2 ^ I) < 0 THEN S = S + 1 / 2 ^ (I + 1)
  NEXT I
PRINT
PRINT S


 0  1  1  1  1  0  0  1  1  1  0  0  0  0  1  0  1  0  1  0  0  0  0  1  0  1  1  0  1  1  0  0
```

Best regards,

Gerson.

P. S.: Replacing TAN with SIN gives 1/(2 * pi), also interesting.

Edited: 16 Mar 2007, 3:43 p.m.

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #41 Posted by ***Valentin Albillo*** *on 16 Mar 2007, 6:48 p.m.,*
in response to message #40 by Gerson W. Barbosa

Hi, Gerson:

Thanks for your interest & kind comments, much appreciated.

Gerson posted:

"Now, what is Plouffe's b-constant anyway? I don't know"

Well, it is the XOR (Exclusive-OR) of the binary expansions of 1/Pi and 1/(2*Pi), i.e:

```
Plouffe's b-constant = .475626... = 1/Pi (+) 1/(2*Pi)
```

where the (+) denotes a XOR operation applied to their binary expansions. As you correctly pointed out, it can be computed with the 11- or 9-step solutions by merely changing the TAN to COS.

By the way, these direct binary expansions aren't uncommon, every function which satisfies some addition formula has an inverse amenable to this treatment.

Best regards from V.

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #42 Posted by ***Rodger Rosenbaum*** *on 17 Mar 2007, 1:21 a.m.,*
in response to message #40 by Gerson W. Barbosa

> Quote:
>
> ---
>
> P. S.: Replacing TAN with SIN gives 1/(2 * pi), also interesting.
>
> ---

Notice that the algorithm extracts the period of the function used (its reciprocal, actually).

A short program to do this on the HP50 is shown below. I have created a list of powers of 2 and saved it in a variable so I can play around from the keyboard. Create it like this:

2 {0, 1, 2, 3, ... , 37, 38} ^ 'pow2' sto

To carry out the algorithm, I then created the following little program:

{1, 2, 4, 8, ... , 274877906944 } TAN 0 <

or,

pow2 TAN 0 <

This will return a list of the binary digits of 1/pi

If you set the calculator to degrees mode and run the program again, you will get the binary digits of 1/180.

Now, show how to use this technique to compute the base 3 representation of 1/pi.

On the HP50 I get 26 accurate trits.

For a hint, notice that this program:

{1, 2, 4, 8, ... , 274877906944 } TAN ATAN 0 <

gives the same result as:

{1, 2, 4, 8, ... , 274877906944 } TAN 0 <

Edited: 17 Mar 2007, 4:28 a.m.

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #43 Posted by ***Gerson W. Barbosa*** *on 17 Mar 2007, 12:24 p.m.,*
in response to message #42 by Rodger Rosenbaum

> Quote:
>
> Now, show how to use this technique to compute the base 3 representation of 1/pi.

Assuming 26 is on the stack, the following returns

```
{ 0. 2. 2. 1. 2. 1. 0. 0. 1. 0. 2. 1. 2. 2. 0. 2. 2. 1. 2. 0. 2. 1. 1. 1. 0. 1. }
```

```
« -> n
   « 3. 1. n
     FOR n n 1. -
     NEXT n ->LIST ^
TAN ATAN DUP ABS 1. >
SWAP DUP 0. < SWAP
-1. > AND 2. * 2.
    « +
    » DOLIST
  »
»
```

Is there an easier way I missed?

Thanks for the interesting post.

Regards,

Gerson.

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #44 Posted by ***Rodger Rosenbaum*** *on 17 Mar 2007, 6:46 p.m.,*
in response to message #43 by Gerson W. Barbosa

Your program does indeed return the correct ternary representation of 1/pi, but I think it may be because of fortuitous circumstances.

The program I gave for finding the binary result:

{1, 2, 4, 8, ... , 274877906944 } TAN 0 <

also gives the correct result if the calculator is in degrees mode (1/180), or grads mode (1/200).

Do you know why your program doesn't give the correct result for the other two modes?

Can you modify it so it works properly for all modes?

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

*Message #45 Posted by **Gerson W. Barbosa** on 17 Mar 2007, 10:33 p.m.,*
in response to message #44 by Rodger Rosenbaum

> Quote:
>
> Can you modify it so it works properly for all modes?

Certainly, but don't expect for an elegant solution:

```
« 100. SIN ASIN 100.
+ DUP 100. <
  « DUP
  » pi IFTE / -> n k      ; pi = 3.14159...
  « 3. 1. n
    FOR n n 1. -
    NEXT n ->LIST ^
TAN ATAN k / DUP ABS
1. > SWAP DUP 0. <
SWAP -1. > AND 2. *
2.
    « +
    » DOLIST
  »
»
```

Using pi explicitly in the fourth line spoils a bit the fun though.

Very *fortuitously* the program computes correctly the first 32 digits of 1/180 when in degrees mode:-)

```
{ 0. 0. 0. 0. 1. 1. 0. 0. 1. 1. 0. 0. 1. 1. 0. 0. 1. 1. 0. 0. 1. 1. 0. 0. 1. 1. 0. 0. 1. 1. 0. 1. }
```

And the first 28 digits of 1/200 when in grads mode:

```
{ 0. 0. 0. 0. 1. 0. 1. 2. 2. 1. 0. 2. 0. 1. 2. 1. 1. 2. 1. 1. 0. 0. 0. 0. 1. 0. 1. 2. }
```

And, of course, the first 26 digits of 1/pi when in radians mode:

```
{ 0. 2. 2. 1. 2. 1. 0. 0. 1. 0. 2. 1. 2. 2. 0. 2. 2. 1. 2. 0. 2. 1. 1. 1. 0. 1. }
```

The program will give always the base-three digits of 1/pí, regardless of the angle mode, if the eighth line is replaced with this one:

```
k * TAN ATAN k / DUP ABS
```

Regards,

Gerson.

P.S.: cygwin and bc have been very useful when checking the results. Thanks again, Egan! :-)

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #46 Posted by ***Gerson W. Barbosa*** *on 19 Mar 2007, 1:32 p.m.,*
in response to message #45 by Gerson W. Barbosa

> Quote:
>
> the program computes correctly the first 32 digits of 1/180 when in degrees mode

Actually, the first 31 digits.

Fixed version:

```
%%HP: T(3)A(D)F(.);
\<< 100 SIN ASIN 100 + DUP 100 <
  \<< DUP
  \>> \pi IFTE / \pi 3 / \-> n k k1
  \<< 3 1 n
    FOR n n 1 -
    NEXT n \->LIST ^ TAN ATAN k / DUP ABS k1 > SWAP DUP 0 < SWAP k1 NEG > AND 2 * 2
    \<< +
    \>> DOLIST
  \>>
\>>
```

This fixed version shouldn't fail for n=115, 138, etc. as pointed by Rodger Rosenbaum.

Edited: 19 Mar 2007, 7:14 p.m.

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #47 Posted by ***Gerson W. Barbosa*** *on 18 Mar 2007, 8:53 a.m.,*
in response to message #44 by Rodger Rosenbaum

> Quote:
>
> Do you know why your program doesn't give the correct result for the other two modes?

No, I don't. Actually, I don't know why the program works in radians mode either... :-)

It took me about twenty minutes to come up with the first version of the program. I just computed atan(tan(x)) in radians mode for the first seven integer powers of three, as you had suggested. My first criterion was placing the results in three regions, according to the table below, which gave four correct digits. By dividing the table into six regions instead of three I obtained six correct digits (the sixth digit was wrong). This would lead to nowhere though. Then, a closer look to the numbers revealed a working criterion. I expanded the table to ten digits. Since the additional digits met the new criterion, I assumed the sixteen remaining possible correct digits on the HP-50G would meet the criterion as well.

When I lack the math background to solve an unusual problem, I try some unorthodox methods. Sometimes they work, but most of times they don't... By the way, I had tried sin(x) and cos(x) for x=1, 2, 4... on Valentin's orginal problem. I didn't try tan(x) because of the large results I would get. Anyway, I wouldn't think of the MATRIX 1 trick, even though it shows on the back of the calculator :-)

```
        x:  1   3   9   27  81 243 729
 pi/2   --+---+---+---+---+---+---+---+
        0 | * |   |   |   |   | * |   |
 pi/6   --+---+---+---+---+---+---+---+
        2 |   | * | * |   |   |   | * |
-pi/6   --+---+---+---+---+---+---+---+
        1 |   |   |   | * | * |   |   |
-pi/2   --+---+---+---+---+---+---+---+
 base-3 digits:  0   2   2   1  *1  *0  *2     (*) wrong digits


y=atan(tan(x)):  1  -0  -0  -1  -0   1   0
                 .   .   .   .   .   .   .
                 0   1   4   2   6   1   1
                 0   4   2   7   8   0   5


abs(y)  > 1 =>               1       1
-1 < y  < 0 =>       2   2       2
 0 > y =< 1 =>   0                       0
               ----------------------------
 base-3 digits:  0   2   2   1   2   1   0   all of them correct!
```

I'm looking forward to your orthodox solution.

Best regards,

Gerson

P. S.: Here is the second version of the program, in a more usable form:

```
%%HP: T(3)A(D)F(.);
\<< 100 SIN ASIN 100 + DUP 100 <
  \<< DUP
  \>> \pi IFTE / \-> n k
  \<< 3 1 n
    FOR n n 1 -
    NEXT n \->LIST ^ TAN ATAN k / DUP ABS 1 > SWAP DUP 0 < SWAP -1 > AND 2 * 2
    \<< +
    \>> DOLIST
  \>>
\>>
```

Edited: 18 Mar 2007, 4:52 p.m.

What about base 4?

Message #48 Posted by ***Gerson W. Barbosa*** *on 18 Mar 2007, 8:51 p.m.,*
in response to message #44 by Rodger Rosenbaum

The program below returns the digits of 1/pi, 1/180 and 1/200 in base 4, depending on the angle mode (20, 22 and 21 correct digits, respectively):

```
%%HP: T(3)A(G)F(.);
\<< 100 SIN ASIN 100 + DUP 100 <
  \<< DUP
  \>> \pi IFTE / \pi 4 / \-> n k k1
  \<< 4 1 n
    FOR n n 1 -
    NEXT n \->LIST ^ TAN ATAN k / DUP k1 > SWAP DUP 0 < SWAP k1 NEG < SWAP OVER XOR 3 * SWAP 2 * 3
    \<< + +
    \>> DOLIST
  \>>
\>>
```

```
1/pi:  { 1. 1. 0. 1. 1. 3. 3. 0. 3. 0. 0. 1. 2. 3. 1. 3. 0. 2. 1. 3. }
```

Regards,

Gerson.

```
-----------



  ]pi/2..pi/4[ -> 1
     ]pi/4..0[ -> 0
    ]-pi/4..0[ -> 3
]-pi/2..-pi/4[ -> 2
```

Edited: 19 Mar 2007, 7:41 p.m.

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #49 Posted by ***Rodger Rosenbaum*** *on 19 Mar 2007, 6:44 a.m.,*
in response to message #43 by Gerson W. Barbosa

Quote:

Assuming 26 is on the stack, the following returns

```
{ 0. 2. 2. 1. 2. 1. 0. 0. 1. 0. 2. 1. 2. 2. 0. 2. 2. 1. 2. 0. 2. 1. 1. 1. 0. 1. }
```

```
« -> n
   « 3. 1. n
     FOR n n 1. -
     NEXT n ->LIST ^
TAN ATAN DUP ABS 1. >
SWAP DUP 0. < SWAP
-1. > AND 2. * 2.
    « +
    » DOLIST
  »
»
```

Is there an easier way I missed?

Thanks for the interesting post.

Regards,

Gerson.

What I mean when I say that this program returns the correct ternary digits because of fortuitous circumstances is this:

Your program correctly gives the first 26 digits, but it makes errors for larger numbers of digits. For example, if the procedure of your program is carried with an arbitrary precision arithmetic package using 500 digits of precision, you will find that it makes an error at the 115th digit, and at the 138th digit, etc.

For example, the procedure of your program gives this result using high precision:

```
{0, 2, 2, 1, 2, 1, 0, 0, 1, 0, 2, 1, 2, 2, 0, 2, 2, 1, 2, 0, 2, 1, 1, 1, 0, 1,
2, 1, 2, 1, 1, 1, 2, 1, 0, 2, 0, 1, 1, 1, 0, 1, 2, 1, 2, 1, 1, 2, 2, 1, 1, 0,
1, 1, 2, 1, 1, 2, 2, 1, 0, 1, 0, 1, 1, 1, 1, 2, 0, 1, 1, 2, 0, 1, 1, 1, 1, 2,
0, 0, 2, 2, 0, 1, 2, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 2, 1, 0, 1, 2, 0, 1,
1, 1, 1, 2, 0, 2, 1, 0, 0, 0, 1, 2, 2, 2, 1, 2, 0, 1, 1, 1, 1, 2, 0, 1, 0, 1,
0, 2, 2, 0, 0, 1, 1, 1, 2, 2, 2, 0, 0, 1, 1, 1, 0, 2, 1, 1, 2, 0, 1, 1, 1, 1,
1, 2, 1, 2, 2, 2, 0, 1, 2, 1, 2, 2, 0, 0, 2, 0, 1, 0, 0, 1, 1, 2, 0, 1, 2, 2,
2, 2, 0, 1, 1, 1, 1, 1, 0, 1, 2, 2, 1, 2, 1, 2, 2, 2, 1, 0, 1, 1, 1, 0, 2, 1,
1, 2, 2, 2, 0, 0, 1, 1, 0, 2, 2, 0, 2, 2, 0, 2, 1, 1, 1, 0, 1, 0, 2, 0, 1, 2,
2, 1, 2, 2, 2, 2, 1, 2, 1, 1, 0, 2, 1, 2, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0,
0, 0, 1, 2, 0, 1, 1, 2, 2, 2, 2, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 2, 0, 2, 2,
1, 0, 0, 2, 0, 0, 2, 0, 2, 0, 2, 0, 2, 1, 0}
```

However, the correct result is:

```
{0, 2, 2, 1, 2, 1, 0, 0, 1, 0, 2, 1, 2, 2, 0, 2, 2, 1, 2, 0, 2, 1, 1, 1, 0, 1,
2, 1, 2, 1, 1, 1, 2, 1, 0, 2, 0, 1, 1, 1, 0, 1, 2, 1, 2, 1, 1, 2, 2, 1, 1, 0,
1, 1, 2, 1, 1, 2, 2, 1, 0, 1, 0, 1, 1, 1, 1, 2, 0, 1, 1, 2, 0, 1, 1, 1, 1, 2,
0, 0, 2, 2, 0, 1, 2, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 2, 1, 0, 1, 2, 0, 1,
1, 1, 1, 2, 0, 2, 1, 0, 0, 0, 0, 2, 2, 2, 1, 2, 0, 1, 1, 1, 1, 2, 0, 1, 0, 1,
0, 2, 2, 0, 0, 1, 1, 0, 2, 2, 2, 0, 0, 1, 1, 1, 0, 2, 1, 1, 2, 0, 1, 1, 1, 1,
1, 2, 0, 2, 2, 2, 0, 1, 2, 1, 2, 2, 0, 0, 2, 0, 1, 0, 0, 1, 1, 2, 0, 1, 2, 2,
2, 2, 0, 1, 1, 1, 1, 1, 0, 1, 2, 2, 1, 2, 1, 2, 2, 2, 1, 0, 1, 1, 1, 0, 2, 1,
1, 2, 2, 2, 0, 0, 1, 1, 0, 2, 2, 0, 2, 2, 0, 2, 1, 1, 1, 0, 1, 0, 2, 0, 1, 2,
2, 0, 2, 2, 2, 2, 1, 2, 1, 1, 0, 2, 1, 2, 0, 2, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0,
0, 0, 1, 2, 0, 1, 1, 2, 2, 2, 2, 1, 0, 0, 1, 1, 2, 0, 0, 0, 1, 0, 2, 0, 2, 2,
1, 0, 0, 2, 0, 0, 2, 0, 2, 0, 2, 0, 2, 1, 1}
```

The first list minus the second is:

```
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1}
```

Can you figure out why?

This shows how simply using numerical results can lead one astray. Just because a formula gives the correct result for the first n digits doesn't mean that it's correct for all n.

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #50 Posted by ***Gerson W. Barbosa*** *on 19 Mar 2007, 10:15 a.m.,*
in response to message #49 by Rodger Rosenbaum

Quote:

> This shows how simply using numerical results can lead one astray. Just because a formula gives the correct result for the first n digits doesn't mean that it's correct for all n.

I am aware of that. Anyway, isn't it amazing a rule deduced from sheer observation of ten samples remains valid for the first 114 digits? Are those seven errors way out of the rules so the formula should be completely discarded? Have you verified the base four case? Is it totally wrong as well?

Could you present the working formula for base 3? Other bases would be nice, especially base 10. As a consequence of the latter, one would be able to compute the *n*th *decimal* digit of *1/pi* without having to compute all previous ones (at least for a few digits). Hasn't this been done already?

I am sorry I cannot take a closer look to this interesting matter now. Among other things, I have to finish studying a thick Portuguese grammar for an examination due next April 1st (still deciding whether I will attend it or not). Did you know verb *ser* (to be) appears in at least 53 forms (if I haven't miscounted them), some completely different from each other? And that's the easy part, as I don't have problem with verbs (Portuguese verbs, I mean). :-)

Regards,

Gerson.

Update:

This appears to work, although I don't know why:

```
%%HP: T(3)A(D)F(.);
\<< 100 SIN ASIN 100 + DUP 100 <
  \<< DUP
  \>> \pi IFTE / \-> n k
  \<< 3 1 n
    FOR n n 1 -
    NEXT n \->LIST ^ TAN ATAN k / FLOOR 3 MOD
  \>>
\>>
```

Forget about this one. It's essentially the same. Looks like I've found it. Take a look at the next reply.

Edited: 19 Mar 2007, 7:08 p.m.

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #51 Posted by ***Gerson W. Barbosa*** *on 19 Mar 2007, 7:07 p.m.,*
in response to message #49 by Rodger Rosenbaum

> Quote:
>
> Can you figure out why?

I think I can. The correct limits should be *pi/3* and *-pi/3* instead of *1* and *-1*. The odds of computing wrong digits because of this mistake are even greater than what you have observed. Perhaps your sample is not large enough.

> Quote:
>
> This shows how simply using numerical results can lead one astray. Just because a formula gives the correct result for the first n digits doesn't mean that it's correct for all n.

You're right! An example is this "algorithm" for finding the square root of four-digit numbers:

Separate the number into two halfs and add them together. Let's try it with three random examples, say, 2025, 3025 and 9801:

```
sqrt(2025) = 20 + 25 = 45;   45 * 45 = 2525  ok!
sqrt(3025) = 30 + 25 = 55;   55 * 55 = 3025  ok!
sqrt(9801) = 98 + 01 = 99;   99 * 99 = 9801  ok!
```

The program below may pass your quality-control test, although I think you have a better algorithm:

```
%%HP: T(3)A(D)F(.);
\<< 100 SIN ASIN 100 + DUP 100 <
  \<< DUP
  \>> \pi IFTE / \pi 3 / \-> n k k1
  \<< 3 1 n
    FOR n n 1 -
    NEXT n \->LIST ^ TAN ATAN k / DUP ABS k1 > SWAP DUP 0 < SWAP k1 NEG > AND 2 * 2
    \<< +
    \>> DOLIST
  \>>
\>>
```

Regards,

Gerson.

P.S.:

From Cygwin & bc:

```
scale=70


a(s(3^114)/c(3^114))
1.03068168


a(s(3^137)/c(3^137))
1.01053215
```

I should have examined the remaining five errors, but I think these are enough. Thanks for the lists!

Edited: 19 Mar 2007, 8:23 p.m.

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #52 Posted by ***Rodger Rosenbaum*** *on 20 Mar 2007, 1:50 a.m.,*
in response to message #51 by Gerson W. Barbosa

> Quote:
>
> I think I can. The correct limits should be pi/3 and -pi/3 instead of 1 and -1.

This is exactly the correct reason. The TAN ATAN sequence is in effect the same as a PI MOD sequence applied to the input argument, the only difference being that the ATAN may return a negative result. In that case, adding PI will give the same result as PI MOD.

In the HP50, the range reduction process uses a PI of effectively 31 decimal digits. Try 1E20 PI MOD and compare to the correct result obtained from an arbitrary precision math package. The HP50 gets a wrong result. But now try 1E20 TAN ATAN PI + (in radians mode) and you will see the correct result (+- possibly a couple of LSD for some input arguments).

This might lead one to think that you could get 31 * LN(10)/LN(2) = 102.979, or about 102 correct binary digits for 1/PI with the algorithm we've been using. Unfortunately, the highest power of 2 that can be accurately represented with 12 decimal digits is 39, and that limits us to 39 correct binary bits.

Thus, it doesn't gain us anything on the HP50 to use the trig functions.

> Quote:
>
> The odds of computing wrong digits because of this mistake are even greater than what you have observed. Perhaps your sample is not large enough.

How do you know what the odds of computing wrong digits are?

I looked at a sample of 5000 digits, and there were 142 digit errors in that sample. In looking at the difference list, I didn't see any appreciable increase in the density of errors toward the end of the list, but of course I would have to look all the way to infinity to make a definitive statement about the density of errors.

Edited addition: Further edited to correct typo:

Your original program makes an error when the arctangent of the tangent of some power of 3 lies between 1 and 1.047+ (PI/3), and between -1.047+ and -1. These two tiny zones are .047+ wide. So out of a total range of PI, if the arctangent of the tangent falls in one of these two zones, an error will occur. The ratio of these zones to the total range is (2*(PI/3-1)/PI = .03004689+, so in my sample of 5000 digits, I would expect to find .03 * 5000 = 150 errors. I actually found 142. This supports the notion that approximately 3% of the digits will be in error.

Using the trig functions on the HP50 can't get us any more significant digits because of the limitation I mentioned above in representing powers of 2 (or 3, or 4, etc.), so we might as well use the MOD function directly. Here is a little program to do the job. Just put the number whose reciprocal is to be converted to another base, the the number of digits desired, and the base, on the stack and execute. It doesn't test for errors, and, of course, if you ask it to give you too many digits, it will return garbage after a certain number of correct digits. Also, it can't output alphabetic characters to represent digits in bases greater than 10.

The idea is to apply NUM MOD to the list of powers of BASE, and then realize that the result is a list of numbers, each of which can range from 0 to NUM. We need to transform them so that they range from 0 to BASE. We do this by dividing the list by NUM and then multiplying by BASE. Then a simple IP will give us the digits we want.

```
« -> NUM DIGITS BASE
  « BASE 0 DIGITS 1 -
    FOR i i
    NEXT
    DIGITS ->LIST ^ NUM MOD
    NUM / BASE * IP
  »
»
```

The program only gives the digits of the fractional part of the reciprocal of the input number, so putting 1/PI on the stack and executing it only gives the fractional part of PI in the other BASE.

If you want to see the digits of PI (or some number whose reciprocal is less than 1) in another BASE, shift PI right by dividing by the BASE enough times so that the reciprocal of that number is greater than 1.

For example, to see the digits of PI in base 9, do this: PI 9 / 1/x 12 9 then execute the program. You will see: {3,1,2,4,1,8,8,1,2,4,0,7}

To see the digits of PI in base 2, do this: PI 2 / 2 / 1/x 39 2 then execute the program.

The last two digits are in error, in this case.

Edited: 25 Mar 2007, 8:18 a.m. after one or more responses were posted

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #53 Posted by ***Gerson W. Barbosa*** *on 20 Mar 2007, 10:48 a.m.,*
in response to message #52 by Rodger Rosenbaum

> Quote:
>
> The ratio of these zones to the total range is (2*(PI/3-1)/PI = .003004689+, so in my sample of 5000 digits, I would expect to find .003 * 5000 = 150 errors. I actually found 142.

Your calculation is right, despite the typo (.003 instead of .03). I went wrong by a factor of 2.75: I did (PI/3-1)/(PI/2-1) = 0.0826872. That's the ratio between the sum of the "0" and "2" digits wrongly converted to "1" and the total of "1" digits in the first list. This would have made me expect 10 wrong "1" digits out of the total 122 "1" digits (0.0827 * 122 = 10), which compared to the actual seven errors wouldn't have called my attention as my wrong assumption did (0.0827 * 301 = 25).

> Quote:
>
> ```
> « -> NUM DIGITS BASE
>   « BASE 0 DIGITS 1 -
>     FOR i i
> ```

```
        NEXT
        DIGITS ->LIST ^ NUM MOD
        NUM / BASE * IP
      »
    »
```

This is a very nice tiny program! I will keep it.

Just out of curiosity, how far can you go beyond the 5000 digits you have already gone? I remember Carl Sagan in *Contact*, about computing *pi* to zillions of places: *"they don't do it because they need to, but because they like to"*. I am not sure this is his exact phrase, as my book is a translation and I cannot find it to check this.

Best regards,

Gerson.

Edited: 20 Mar 2007, 12:48 p.m.

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #54 Posted by ***Rodger Rosenbaum*** *on 20 Mar 2007, 1:28 p.m.,*
in response to message #53 by Gerson W. Barbosa

> Quote:
>
> Your calculation is right, despite the typo (.003 instead of .03).

I was doing the calculations on the HP50. The ratio was in scientific format on the HP50 (3.00468943012E-2). I made the typo when I first typed it in my message, and propagated it subsequently. But, I did the multiplication by 5000 on the HP50 with the correct 3.00468943012E-2 and got the correct 150 in fixed format, which I then copied to the message.

I've noticed that on this forum, dealing with a lot of numbers, it's easy to make typos. I always proofread a couple of times, but they slip by anyway. I was up too late! Mea culpa.

> Quote:
>
> Just out of curiosity, how far can you go beyond the 5000 digits you have already gone?

I only let it go to 5000 which took a couple of minutes. If I remember correctly, it takes about 2400 decimal digits to represent 3^5000, so I used 3000 digits in all the calculations, which leads to slowness. I suppose I could let it run all night!

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #55 Posted by ***Rodger Rosenbaum*** *on 20 Mar 2007, 6:06 p.m.,*
in response to message #52 by Rodger Rosenbaum

> Quote:
>
> In the HP50, the range reduction process uses a PI of effectively 31 decimal digits. Try 1E20 PI MOD and compare to the correct result obtained from an arbitrary precision math package. The HP50 gets a wrong result. But now try 1E20 TAN ATAN PI + (in radians mode) and you will see the correct result (+- possibly a couple of LSD for some input arguments).
>
> This might lead one to think that you could get 31 * LN(10)/LN(2) = 102.979, or about 102 correct binary digits for 1/PI with the algorithm we've been using. Unfortunately, the highest power of 2 that can be accurately represented with 12 decimal digits is 39, and that limits us to 39 correct binary bits.
>
> Thus, it doesn't gain us anything on the HP50 to use the trig functions.

It occured to me that this isn't strictly true. In the one case where the BASE we want for our output is 10, the list of powers of the BASE can contain numbers that *exactly* represent those powers, because the mantissa parts of powers of 10 only contain the single digit 1. Thus, the HP50 can exactly represent powers of 10 up to the exponent limit of 499.

So, change the little program like this to do the TAN ATAN sequence, and then add PI to each element of the list to compensate for the fact that ATAN sometimes returns a negative result:

```
« -> NUM DIGITS BASE
   « BASE 0 DIGITS 1 -
     FOR i i
     NEXT
     DIGITS ->LIST ^ TAN ATAN
     NUM ADD NUM MOD
     NUM / BASE * IP
   »
»
```

We are now effectively doing a MOD with a 31 digit PI.

Now type PI 31 10 and execute the program.

See 31 correct decimal digits of 1/PI.

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #56 Posted by ***Gerson W. Barbosa*** *on 21 Mar 2007, 8:50 a.m.,*
in response to message #55 by Rodger Rosenbaum

> Quote:
>
> In the one case where the BASE we want for our output is 10, the list of powers of the BASE can contain numbers that *exactly* represent those powers, because the mantissa parts of powers of 10 only contain the single digit 1. Thus, the HP50 can exactly represent powers of 10 up to the exponent limit of 499.

Keen observation! And producing the first 31 decimals of *1/pi* on the HP-50G without explicitly using it is really nice! By what I remember, it's possible to display *pi* to 24 places on the HP-50G this way:

```
RAD mode


« pi 1E-11 - DUP SIN»
```

A made a plot, not on the computer but by hand, of arctan(tan(x)). I turned the saw-tooth graph into a straight line *(y=x)* by adding successive multiples of *pi* to each discontinuous section. This way, it was easy to see why the algorithm I presented on March 11th works (http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/forum.cgi?read=110018#110018). The algorithm can be generalized to any base, as we can see in the following 91-byte program:

```
« -> n d b
  « "" 1 d
    FOR i b i ^ n * IP b MOD ->STR HEAD +
    NEXT
  »
»
```

It returns the first *d* digits in base *b* of the fractional part of *n*. For instance, running the program after entering *pi 1/x 17 3* produces a string with the first 17 ternary digits of *1/pi*:

```
"02212100102122022"
```

If I had thought a bit rather than "blindly going where no one had gone before" I would have saved some time - and I might be now at page 500 of my Grammar book instead of page 300 :-)

Anyway, learning something from the mistakes is not bad. I have learned that analyzing a graph rather than a few numbers produces a better result. (Not to mention *ADD* is better than *2 «+» DOLIST* :-)

Thanks!

Gerson.

P.S.: Although I said the algorithm is valid to any base, the program will handle bases up to 10, unless some changes are made to allow it to work for greater bases up to the limitations of the calculator.

Edited: 21 Mar 2007, 9:12 a.m.

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #57 Posted by ***Rodger Rosenbaum*** *on 21 Mar 2007, 4:51 p.m.,*
in response to message #56 by Gerson W. Barbosa

> Quote:
>
> ```
> « -> n d b
>   « "" 1 d
>     FOR i b i ^ n * IP b MOD ->STR HEAD +
>     NEXT
>   »
> »
> ```
>
> It returns the first d digits in base b of the fractional part of n. For instance, running the program after entering pi 1/x 17 3 produces a string with the first 17 ternary digits of 1/pi:
>
> "02212100102122022"
>
> P.S.: Although I said the algorithm is valid to any base, the program will handle bases up to 10, unless some changes are made to allow it to work for greater bases up to the limitations of the calculator.

That last characteristic is why my program created a list rather than a string. Even though the alphabetic characters aren't created for hexadecimal based output, for instance, you can still see the digits with value over 9 in the list.

e.g., modifying your program like this will do it:

```
« -> n d b
  « {} 1 d
    FOR i b i ^ n * IP b MOD +
    NEXT
  »
»
```

> Quote:
>
> If I had thought a bit rather than "blindly going where no one had gone before" I would have saved some time - and I might be now at page 500 of my Grammar book instead of page 300 :-)

To be at page 500 of a grammar book! Is that not a fate worse than death? Or, almost worse, anyway!

What better way to take a break from the tedium of a grammar book, than to play with your calculator?

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #58 Posted by ***Egan Ford*** *on 17 Mar 2007, 2:24 a.m.,*
in response to message #40 by Gerson W. Barbosa

> Quote:
>
> Could someone compute 50 or more bits?

```
0111100111
0000101010
0001011011
0010110100
1011001100


0011111101
1110100000
0100011010
0111111000
0111001000
```

If you have a Linux box or Cygwin on Windows run bc -l then enter this to generate the above bits.

```
scale=30
for(i=0;i<100;i++) {
  if(c(2^i) < 0) print 1 else print 0;
  scale=0
  if((i+1) % 10 == 0) print "\n";
  if((i+1) % 50 == 0) print "\n";
  scale=30
}
```

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #59 Posted by ***Gerson W. Barbosa*** *on 17 Mar 2007, 1:18 p.m.,*
in response to message #58 by Egan Ford

Thanks! Perhaps it's time for me to finally starting to read those two books on C and C++ that are lying forever on my shelf :-)

Since we're still on the topic, does anyone know how to extract the binary digis of expressions like this one for ln(2):

```
SUM(k=1, inf) 1/k * 2^-k
```

This should be easy because of the *2^-k* factor, but I couldn't figure it out.

Regards,

Gerson.

Edited: 17 Mar 2007, 1:21 p.m.

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #60 Posted by ***Egan Ford*** *on 17 Mar 2007, 1:28 p.m.,*
in response to message #59 by Gerson W. Barbosa

bc is an arbitrary precision calculator language, no need to learn C/C++, just type "man bc". Cygwin is a free Linux environment for Windows, just install that and you will have a nice tool set--the bash shell, bc, perl, vi editor, etc...

Cygwin includes all the nice Perl math packages too, e.g. Math::Complex, Math::BigFloat, etc...

Perl is interpreted and very portable and easier to learn than C/C++. I only use C if I need speed. I try to approach any problem with "try smarter" instead of "try harder", and Perl always does the job.

If you want an easy way to get the LN(2) bits you are looking for just run bc -l and type:

```
ibase=10
obase=2
scale=30
l(2)
```

Output:

```
.1011000101110010000101111111011111010001110011110111100110101011110010011110001110110011100110000000
```

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #61 Posted by ***Gerson W. Barbosa*** *on 17 Mar 2007, 4:12 p.m.,*
in response to message #60 by Egan Ford

Thanks for the explanation. I've just used bc to check the digits of 1/pi in base 3:

```
ibase=10
obase=3
scale=30
1/(4*a(1))
```

Output:

```
.0221210010212202212021110121211121020111012121122110112112211010
```

Great tool!

Gerson.

Re: HP-15C MC Bits o'Pi - My Original Solution & Comments

Message #62 Posted by ***Bill (Smithville, NJ)*** *on 16 Mar 2007, 8:55 p.m.,*
in response to message #39 by Valentin Albillo

Hi Valentin,

Very Very Nice!

Quote:

> the only thing that makes it HP-15C-specific is the MATRIX 1 instruction

When I read you're earlier hint about it being HP-15C specific, I did spend some time with the HP-15C manual but just didn't see how Matrix, Complex, etc could be used. Thanks for pointing out another method of initialing a register to 1. Very neat.

> Quote:
>
> Next April, 1st I'll post my S&SMC Spring 2007 Special, I hope to 'see' you there (if you think this Mini-Challenge is pretty unusual math, wait till you see the S&SMC ! :-)).

Uh-Oh I just realized what day that is :) Looking forward to it.

Bill

Re: HP-15C MC Bits o'Pi (proof of theorem?)

Message #63 Posted by ***Karl Schneider*** *on 18 Mar 2007, 2:50 p.m.,*
in response to message #39 by Valentin Albillo

Hi, Valentin --

Well, I "never woulda thunk it" in a thousand years -- "tangent of 2^{k-1} radians gives the k-th binary digit of 1/pi" as follows:

tan (2^{k-1}) > 0 => digit is 0
tan (2^{k-1}) < 0 => digit is 1
tan (2^{k-1}) = 0 => (can't happen)"

This just can't be a coincidence -- it must be a theorem. However, my cursory Google search yielded nothing about this. Could you point to a proof?

This exqmple ties in with a recent archived thread, "Trigonometrics for really big input". The calculations in your example are possible because the trig functions can be calculated accurately for inputs up to the limit of 10 or 12 signficant digits. These, in turn, are possible because internal extended precision permits accurate calculation of, e.g., MOD(8589934592, 2*pi) for reduction of the input argument.

P.S.: Now, aren't you glad that "MATRIX 1" sets R0 and R1 instead of creating an identity matrix? *(Just kidding!)*

-- KS

(Edited to correct an erroneous-word mental lapse that was identified below...)

Edited: 19 Mar 2007, 1:22 a.m. after one or more responses were posted

Re: HP-15C MC Bits o'Pi (proof of theorem?)

Message #64 Posted by ***Egan Ford*** *on 18 Mar 2007, 4:09 p.m.,*
in response to message #63 by Karl Schneider

If you have not already start here: http://mathworld.wolfram.com/PlouffesConstants.html

HP-35 is mentioned. A happy coincident?

Re: HP-15C MC Bits o'Pi (proof of theorem?)

Message #65 Posted by ***Valentin Albillo*** *on 18 Mar 2007, 7:45 p.m.,*
in response to message #63 by Karl Schneider

Hi, Karl:

Karl posted:

> *"Well, I "never woulda thunk it" in a thousand years -- "tangent of 2k-1 radians gives the k-th decimal digit of 1/pi"*
>
> > That's because, actually, it's unthinkable indeed: the Tan algorithm merely gives *binary* digits, not *decimal* ... ;-)
>
> *"This just can't be a coincidence -- it must be a theorem. However, my cursory Google search yielded nothing about this. Could you point to a proof ?"*
>
> > Yes. I don't know about Google, I actually took the idea from my extensive math-related printed materials, in this case an article which can be located using this reference:
> >
> > ```
> >           "Addition Theorems and Binary Expansions"
> >            Canadian Journal of Mathematics, 47:262-273, 1995.
> > ```
>
> *"P.S.: Now, aren't you glad that "MATRIX 1" sets R0 and R1 instead of creating an identity matrix? (Just kidding!)"*

Touché, *very* good memory ! :-) :-) However, I rest my case and still wish that MATRIX 1 were an identity matrix creation statement instead of merely setting R0 and R1 to 1.

Thanks for your interest and

Best regards from V.

Re: HP-15C MC Bits o'Pi (proof of theorem?)

Message #66 Posted by ***Karl Schneider*** *on 19 Mar 2007, 1:44 a.m.,*
in response to message #65 by Valentin Albillo

Hi again, Valentin --

Thank you for the reference. I just might take time to look it up online or see if I can access the article in a collegiate library.

I corrected the silly mental lapse in my earlier post. Of course I meant, "*binary* digit"; I'm just not accustomed to writing it.

Best regards,

-- KS

Re: HP-15C MC Bits o'Pi (proof of theorem?)

Message #67 Posted by ***Egan Ford*** *on 22 Mar 2007, 8:12 p.m.,*
in response to message #66 by Karl Schneider

http://www.lacim.uqam.ca/~plouffe/articles/additiontheorems.pdf

Re: HP-15C MC Bits o'Pi (proof of theorem?)

Message #68 Posted by ***Karl Schneider*** *on 24 Mar 2007, 3:57 p.m.,*
in response to message #67 by Egan Ford

Hi, Egan --

Thanks; the answer seems to be in the "Introduction" section of the paper.

-- KS