HP Forum Archive 19

[Return to Index | Top of Index]

4915 digits of pi - MCODE 41 (19660 digits optional)

*Message #1 Posted by **PeterP** on 26 Feb 2009, 5:27 p.m.*

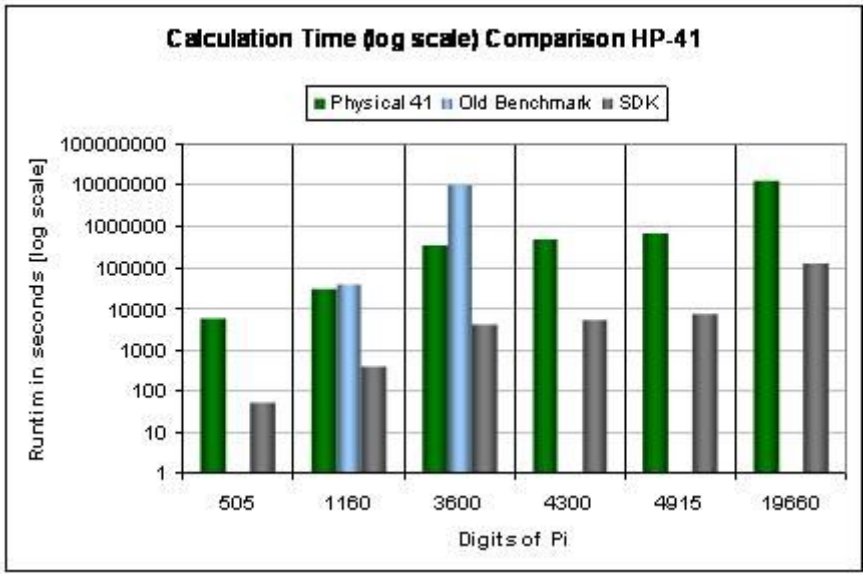
Hi,

A few months back Katie insidiously infected me with the Pi bug (see here), from which I had been save until then. She has created a series of deeply impressive programs for a variety of HP calculators:Katie's Pi article. Not knowing anything about the field I proceeded to read her programs as well as follow the kind explanations, links and book suggestions from her and Egan. Especially the Spigot algorithm was suggested as an interesting candidate.

Looking through the form I also found numerous threads for Pi calculations on the hp41. JFG had a program that calculated 560 digits in 22hours (see here). A program 'Deep Pi' is discussed here but seems to have extremely long run times. The first 6 digits take 38 minutes. And 625 digits take tens of days…
According to an old post from Gene Wright (see here) about a challenge between the TI-59 and HP-41 it would seem that with reasonable times, 1160 digits (15.5 hours) was the top mark. Allowing unreasonable run times, the incredible number of 3600 digits could be reached, albeit only in 4 month…

Please find below my own little contribution to the Many-Digits-of-Pi story on the HP-41 using MCODE. It is neither particularly elegant or ingenious (quite the opposite) but it does deliver 1160 digits in about 9 hours (instead of 15.5), 3600 digits in about 4 days (instead of 120 days), and 4915 digits in about 8 days. Using a RSU-2 unit with 128k (and a small extension of the code to switch banks, not implemented) one could calculate 19660 digits of pi in about 21.5 weeks. See a

logarithmic comparison chart:



The code implements a straight forward Spigot mechanism (see Pi Unleashed, thanks Egan) using MLDL ram to store the intermediate results. Each intermediate result (the a(i)'s) are stored in 2 MLDL words (left and right Nybble). The right Nybble is converted to hex as each word can store at most 399d but can store 3ffh(=4096d). As the numbers don't grow larger than 399,999, we can leave the left nybble in dec format. Also, as there are no 5 or 6 9's consecutively in Pi in the first 20,000 digits, we don't have to worry about that either.

The code asks for three inputs: The page where the MLDL ram starts to use, the number of digits and the base b to use (max = 5 for 5 digits at a time). One can set Flag 0 and the calc will stop at each group of digits and wait for a key to be pressed, otherwise it just keeps calculating, calculating, calculating… Setting Flag 1 will store the found digits in the same compressed format – each group of up to 5 digits is stored in 2 words, with the right nybble converted to hex. They are stored in reversed order though.

Runtimes are quite accurately n^2 (see table 1 below). Actual run-times on physical HP's where done up to 505 digits and above I used the SDK emulator. Times marked with (*) are extrapolated from existing times using a power fit through the first 9 data-points up to 4915 digits with an R2 of 0.9992 so they should be quite accurate.

Digits	Note	Physical 41	SDK	Old Benchmark	Configuration
505		1h 32m 49s	54s		MDOP in page 9, Rambox using page A
1160	old top mark	9h 01m 00s(*)	06m 32s	11h 30m 00s	MDOP in page 9, Rambox using page A, B
3600	old abs max dig	4d 04h 00m 00s(*)	1h 07m 03s	120d 00h 00m 00s	MDOP in page 9, RAMBOX using page A,B,C,D,E,F
4300	max without Clonix/Zeprom	6d 00h 06m 48s(*)	1h 25m 42s		MDOP in page 8, RAMBOX using page 9,A,B,C,D,E,F
4915	max without bankswitching	1w 0d 23h 08m 32s(*)	2h 07m 06s		MDOP in page 6, RAMBOX using page 8,9,A,B,C,D,E,F
19660	max with RSU-128k	21w 01d 11h 01m 49s(*)	1d 12h 30m 50s(*)		MDOP in page 6, RAMBOX using page 8,9,A,B,C,D,E,F x4 banks

Cheers, Peter

```
* PI.SRC
* Assembled by A41
* Thu Feb 26 16:39:13 2009
                              .TITLE  "PI"
                              .JDA
                              .EQU    [PCTOC] 00D7
;*******************************************************************************
;*  FAT for MANYDIGOFPI
;*******************************************************************************
 0000 005                     XROM    5       ;XROM number.
 0001 002                     FCNS    2       ;Header + functions
 0002 000013                  DEFR4K  [Header] 0013   ;1 - first executable of header
 0004 000018                  DEFR4K  [MDOP] 0018     ;2 - Many Digits of Pi
 0006 000                     NOP
 0007 000                     NOP
;-------------------------------------------------------------------------------
                              .NAME   "MANYDIGOFPI"
*0008 089                     #089            ; "I"
*0009 010                     #010            ; "P"
*000A 006                     #006            ; "F"
*000B 00F                     #00F            ; "O"
*000C 007                     #007            ; "G"
*000D 009                     #009            ; "I"
*000E 004                     #004            ; "D"
*000F 019                     #019            ; "Y"
*0010 00E                     #00E            ; "N"
*0011 001                     #001            ; "A"
*0012 00D                     #00D            ; "M"
 0013 3E0       [Header]      RTN
;-------------------------------------------------------------------------------
; Many Digits of PI
; Spigot algorithm from Pi-book
; uses base b <= 5 to show 5 digits at a time
;Flag0 - wait for key press after each group is shown
;Flag1 - store result digits in reverse order from end (iStart)
;Input:
; X : base b in powers of 10
; Y : n - number of digits wanted
; Z : p - page number of start of MLDL ram to use
;--------------
; All Stack and Alpha is used for temp storage
; 3(X):  i in dec, 1step              5(M): orig iStart in hex and 2 step
; 2(Y):  tmp                          6(N): last addr in hex and 2 step
; 1(Z):  iBits in dec, 1 step         7(O): iBits in hex, 2 step
; 4(L):  iStart in dec, 1 step        8(P): b|iStart in hex and 2 step
; 9(Q): q - remainder                 0(T): page number in hex in C:[0]
;-------------
```

```
; All numbers are integers without exponent starting at C[0]
; User-Flag 0 -> wait for key press after each numbers shown. Stored in M-Flag 9
;-------------------------------------------------------------------------------
                              .NAME   "MDOP"
*0014 090                     #090            ; "P"
*0015 00F                     #00F            ; "O"
*0016 004                     #004            ; "D"
*0017 00D                     #00D            ; "M"
 0018 1A0       [MDOP]        A=B=C=0         ;
; set Flag 0 if we want to wait for key
 0019 244                     CLRF    9
 001A 3B8                     READ    14(d)
 001B 046                     C=0     S&X
 001C 2FC                     RCR     13      ;first nybble into C[0}
 001D 3D8                     C<>ST
 001E 00C                     ?FSET   3       ;is userflag 0 set?
 001F 023                     JNC     (sf0l1) +4 0023
 0020 3D8                     C<>ST
 0021 248                     SETF    9       ;
 0022 013                     JNC     (sf0l2) +2 0024
 0023 3D8       (sf0l1)       C<>ST
;calc and store page number to start MLDL-RAM
 0024 078       (sf0l2)       READ    1(Z)    ;read in number from user
 0025 05E                     C=0     MS      ;kill any sign
 0026 056                     C=0     XS      ;kill any sign on exponent
 0027 31C                     R=      1
 0028 042                     C=0     @R      ;make sure it is a reasonable number <100
 0029 38D008                  ?NCXQ   [BCDBIN] 02E3   ;convert to hex in C[S&X]
 002B 0AE                     A<>C    ALL
 002C 04E                     C=0     ALL
 002D 260                     SETHEX
 002E 130010                  LDIS&X  010
 0030 306                     ?A<C    S&X     ;max page is F
 0031 0B50A2                  ?NCGO   [ERRDE] 282D
 0033 04E                     C=0     ALL
 0034 270                     RAMSLCT
 0035 0AE                     A<>C    ALL
 0036 2F0                     WRITDATA                ;store into T(0)
;---------- Calc iStart = floor(10*n/3 + 1)
 0037 2A0                     SETDEC
 0038 0B8                     READ    2(Y)    ;read in n
 0039 226                     C=C+1   S&X     ;*10
 003A 0AE                     A<>C    ALL
 003B 04E                     C=0     ALL
 003C 130003                  LDIS&X  003
 003E 23C                     RCR     2       ;C:=3
 003F 261060                  ?NCXQ   [DV2_10] 1898   ;C = A/C
```

```
0041 00E                    A=0     ALL
0042 35C                    R=      12
0043 162                    A=A+1   @R
0044 01D060                 ?NCXQ   [AD2_10] 1807   ;(10*n/3 + 1)
0046 088                    SETF    5       ;to get int function
0047 0ED064                 ?NCXQ   [INTFRC] 193B   ;C= int(10*n/3 +1) = iStart in dec and 1 step
0049 39C       [tmp1]       R=      0       ;get rid of exponent
004A 2A2                    C=-C-1  @R      ;get 10-exp
004B 262                    C=C-1   @R
004C 3DA       (pil1)       RSHFC   M
004D 262                    C=C-1   @R
004E 3F3                    JNC     (pil1) -2 004C
004F 046                    C=0     S&X     ;clear underflow
0050 03C                    RCR     3       ;flush right
0051 128                    WRIT    4(L)    ;iStart
0052 1EE                    C=C+C   ALL     ;2nyb per i
0053 0AE                    A<>C    ALL
0054 260                    SETHEX
0055 37903C187              ?NCXQREL        [4DEC2HEX] 0187
0058 0AE                    A<>C    ALL
0059 149024                 ?NCXQ   [ENCP00] 0952   ;just in case
005B 038                    READ    0(T)    ;page to use
005C 1BC                    RCR     11      ;=-3
005D 260                    SETHEX
005E 20E                    C=A+C   ALL     ; istart in Hex and 2 steps per i
005F 1BC                    RCR     11      ;to have it in right pos in C
0060 168                    WRIT    5(M)    ;store iStart for usage in later MLDL RAM Clean-up
0061 03C                    RCR     3       ;shift back for P reg storage format
0062 0AE                    A<>C    ALL
0063 0F8                    READ    3(X)    ;b
0064 2FC                    RCR     13      ;=-1
0065 0BE                    A<>C    MS
0066 0AE                    A<>C    ALL
0067 228                    WRIT    8(P)    ;b|iStart in hex
0068 138                    READ    4(L)    ;iStart in dec and 1 step
0069 0E8                    WRIT    3(x)    ;i in dec and 1 step
006A 238       [CalcInit]   READ    8(P)    ;b|iStart in hex
006B 2A0                    SETDEC          ;calc 2*(b-1)!!!
006C 0AE                    A<>C    ALL
006D 1BE                    A=A-1   MS      ;b-1
006E 04E                    C=0     ALL
006F 130002                 LDIS&X  002
0071 2FC       (pil2)       RCR     13
0072 1BE                    A=A-1   MS
0073 3F3                    JNC     (pil2) -2 0071
0074 33C       [InitW2]     RCR     1       ;undo overflow shift
0075 158                    M=C             ;save initvalue temporarily
```

```
0076 00E                    A=0     ALL     ;prep A & B
0077 02E                    B=0     ALL
0078 0A6                    A<>C    S&X     ;right Nybble into A
0079 260                    SETHEX
007A 37903C185              ?NCXQREL        [Do3rd_EP] 0185 ;convert A[S&X] into hex -> C[S&X]
007D 0A6                    A<>C    S&X     ;right Nybble into A[S&X]
007E 198                    C=M             ;bring back full init value in dec
007F 03C                    RCR     3
0080 0E6                    B<>C    S&X     ;left nybble in B[S&X], right Nybble in A[S&X]
0081 149024    [InitLoop]   ?NCXQ   [ENCP00] 0952   ;clear C and select chip 0
0083 038                    READ    0(T)    ;page number to use, hex and right justified
0084 13C                    RCR     8       ;=-6. C[3:6] = 8000
0085 23A                    C=C+1   M       ;8001 -> we don't use address 8000. i starts at 1 and not at 0
0086 0BA                    A<>C    M       ; A[M] = 8001
0087 238                    READ    8(P)    ;b|iStart in hex
0088 05E                    C=0     MS
0089 1BC                    RCR     11      ;=-3 to get iStart into C[M]
008A 260                    SETHEX
008B 0A6       [WR2L]       A<>C    S&X     ;right nybble
008C 040                    WROM
008D 000                    NOP
008E 0A6                    A<>C    S&X
008F 27A                    C=C-1   M       ;addre for left nybble
0090 0C6                    C=B     S&X     ;left nybble
0091 040                    WROM
0092 000                    NOP
0093 37A                    ?A#C    M       ;compare to 8000
0094 01B                    JNC     [DoneInit] +3 0097
0095 27A                    C=C-1   M
0096 3AB                    JNC     [WR2L] -11 008B
0097 2A0       [DoneInit]   SETDEC
0098 1A0                    A=B=C=0         ;init other vars
0099 149024                 ?NCXQ   [ENCP00] 0952
009B 268                    WRIT    9(Q)    ;q:=0
009C 238       [CalciBits]  READ    8(P)
009D 05A                    C=0     M
009E 046                    C=0     S&X
009F 33C                    RCR     1
00A0 130001                 LDIS&X  001     ;10*b
00A2 00E                    A=0     ALL
00A3 0AE                    A<>C    ALL
00A4 130003                 LDIS&X  003
00A6 23C                    RCR     2       ;C;= 3
00A7 261060                 ?NCXQ   [DV2_10] 1898   ;c=a/c = 10*b/3
00A9 00E                    A=0     ALL
00AA 35C                    R=      12
00AB 162                    A=A+1   @R
```

```
 00AC 01D060                    ?NCXQ   [AD2_10] 1807   ;C=A+C = 10*b/3 + 1
 00AE 088                       SETF    5
 00AF 0ED064                    ?NCXQ   [INTFRC] 193B
 00B1 39C                       R=      0
 00B2 2A2                       C=-C-1  @R
 00B3 262                       C=C-1   @R
 00B4 2E2       (pil3)          ?C#0    @R
 00B5 3DA                       RSHFC   M
 00B6 262                       C=C-1   @R
 00B7 3EB                       JNC     (pil3) -3 00B4
 00B8 046                       C=0     S&X     ;clear underflow
 00B9 03C                       RCR     3       ;C:= iBits in dec
 00BA 068       [tmp2]          WRIT    1(Z)    ;iBits in dec and 1 step in Z
 00BB 1EE                       C=C+C   ALL
 00BC 00E                       A=0     ALL     ;prep A&B
 00BD 02E                       B=0     ALL
 00BE 0A6                       A<>C    S&X     ;iBits in dec and 2 Steps into A[S&X] -> hex
 00BF 260                       SETHEX
 00C0 37903C185                 ?NCXQREL        [Do3rd_EP] 0185
 00C3 0A6                       A<>C    S&X     ;save result into A
 00C4 149024                    ?NCXQ   [ENCP00] 0952
 00C6 0A6                       A<>C    S&X     ;bring back iBits in hex and 2 step
 00C7 1E8                       WRIT    7(O)    ;iBits in hex and 2 step
;-----------------------------------------
; Start the loop
 00C8 238       [InitML]        READ    8(P)    ;
 00C9 05E                       C=0     MS
 00CA 1BC                       RCR     11      ;-3 ;iStart into C[M]
 00CB 260                       SETHEX
 00CC 23A                       C=C+1   M       ;GetA(i) expexts address of last nybble in A(i)
 00CD 0AE                       C<>A    ALL     ;so that A is only the hex address and all other stuff is clear
 00CE 260       [iLoop]         SETHEX          ;for later loops needed
 00CF 37903C1BA                 ?NCXQREL        [GetA(i)] 01BA
 00D2 0AE                       A<>C    ALL     ;A(i) into A[0:5]
 00D3 1A8                       WRIT    6(N)    ;last address into N in hex, 2 step
 00D4 260                       SETHEX
 00D5 37903C235                 ?NCXQREL        [Calc_bA+qi] 0235       ;res in C
 00D8 0AE                       A<>C    ALL     ;res into A
 00D9 0F8                       READ    3(x)    ;i in dec, 1 step
 00DA 2A0                       SETDEC
 00DB 1EE                       C=C+C   ALL
 00DC 26E                       C=C-1   ALL     ;2*i - 1
 00DD 0EE                       C<>B    ALL     ;into B
 00DE 260                       SETHEX
 00DF 37903C216                 ?NCXQREL        [CalcRnQ] 0216
 00E2 268                       WRIT    9(Q)    ;q in C, r in A
 00E3 06E                       A<>B    ALL     ;r into B. prep for SaveA(i)
```

```
 00E4 1B8                    READ    6(N)    ;last address in hex and 2 step
 00E5 260                    SETHEX
 00E6 23A                    C=C+1   M
 00E7 0AE                    C<>A    ALL
 00E8 37903C1E7              ?NCXQREL        [SaveA(i)] 01E7
 00EB 149024    [Nxti]       ?NCXQ   [ENCP00] 0952
 00ED 0F8                    READ    3(x)    ;i in dec and 1 step
 00EE 2A0                    SETDEC
 00EF 26E                    C=C-1   ALL
 00F0 0E8                    WRIT    3(X)    ;next i in dec and 1 step
 00F1 1B8                    READ    6(N)    ;last addr in hex and 2 step
 00F2 0AE                    C<>A    ALL     ;prep for [GetA(i)]
 00F3 0F8                    READ    3(X)    ;check if we are done
 00F4 26E                    C=C-1   ALL     ;if this is 0 we are done and need to process A(1)
 00F5 2EE                    ?C#0    ALL
 00F6 2C7                    JC      [iLoop] -40 00CE
 00F7 260       [DoA(1)]     SETHEX
 00F8 37903C1BA              ?NCXQREL        [GetA(i)] 01BA
 00FB 0AE                    A<>C    ALL     ;A(i) into A
 00FC 1A8                    WRIT    6(N)    ;last address into N in hex, 2 step
 00FD 260                    SETHEX
 00FE 37903C235              ?NCXQREL        [Calc_bA+qi] 0235       ;result in C
;right b digits = remainder = A(1)
 0101 158                    M=C             ;temp store result
 0102 149024                 ?NCXQ   [ENCP00] 0952
;---- Stepping stone ---
 0104 013                    JNC     (step1) +2 0106
 0105 21B       [StepML2]    JNC     [InitML] -61 00C8
;-----------------------
 0106 238       (step1)      READ    8(P)
 0107 0BE                    A<>C    MS      ;b into A[MS]
 0108 02E                    B=0     ALL
 0109 0E0                    SLCTQ
 010A 2DC                    R=      13
 010B 0A0                    SLCTP
 010C 2DC                    R=      13
 010D 2A0                    SETDEC
 010E 198                    C=M             ;bring back b*A(1) + q*i
 010F 3DC       (da1l1)      R=R+1
 0110 1BE                    A=A-1   MS
 0111 3F3                    JNC     (da1l1) -2 010F
 0112 0F2                    B<>C    P-Q     ;clean out digits and save result in B
 0113 0EE                    B<>C    ALL     ;remainder into B as prep for SaveA(i)
 0114 0A8                    WRIT    2(Y)    ;temp storage of result
 0115 1B8                    READ    6(N)    ;last address in hex and 2 step
 0116 260                    SETHEX
 0117 23A                    C=C+1   M
```

```
 0118 0AE                    C<>A    ALL
 0119 37903C1E7              ?NCXQREL        [SaveA(i)] 01E7
 011C 149024    [ShowDigs]   ?NCXQ   [ENCP00] 0952
 011E 238                    READ    8(P)    ;
 011F 158                    M=C             ;save iStart
 0120 2A0                    SETDEC
 0121 1FE                    C=C+C   MS      ;calc extra shifts needed
 0122 29E                    C=0-C   MS      ;
 0123 0BE                    A<>C    MS
 0124 006                    A=0     S&X     ;counter for leading 0s
 0125 166                    A=A+1   S&X
 0126 0B8                    READ    2(Y)    ; bring back result digits
 0127 35C                    R=      12      ; shift to left (attention! This kills leading 0s!)
 0128 1BC                    RCR     11      ;min shift left is 3 (as res is right justified)
 0129 35E       (sdl2)       ?A#0    MS
 012A 02B                    JNC     (sdl1ep) +5 012F
 012B 1BE                    A=A-1   MS
 012C 2FC                    RCR     13
 012D 3E3                    JNC     (sdl2) -4 0129
 012E 2FC       (sdl1)       RCR     13      ;shift left 1 step so that we always show d.dddd or the like
 012F 1A6       (sdl1ep)     A=A-1   S&X     ;for leading zeros
 0130 2E2                    ?C#0    @R
 0131 3EB                    JNC     (sdl1) -3 012E
 0132 0A8                    WRIT    2(Y)    ;store for potential later save
;---- Stepping stone ---
 0133 013                    JNC     [DoneShift] +2 0135
 0134 28B       [StepML1]    JNC     [StepML2] -47 0105
;-----------------------
 0135 0A6       [DoneShift]  A<>C    S&X     ;bring in neg exp for leading zeros
 0136 09902C                 ?NCXQ   [DSPCRG] 0B26   ;shows full C-reg. uses all flags0-7
 0138 198                    C=M             ;bring back iStart & B
 0139 228                    WRIT    8(P)
 013A 24C                    ?FSET   9       ;do we wait for key press?
 013B 03B                    JNC     [NoWaitC] +7 0142       ;no -> just continue
 013C 261000                 ?NCXQ   [RSTKB] 0098    ;debounce & reset keyboard
 013E 3CC       [KeyWL]      ?KEY            ;is a key pressed
 013F 3FB                    JNC     [KeyWL] -1 013E ;no --> keep waiting
 0140 3C10B0    [CNewiSt]    ?NCXQ   [CLLCDE] 2CF0
 0142 149024    [NoWaitC]    ?NCXQ   [ENCP00] 0952
 0144 3B8                    READ    14(d)   ;check if we want to write the numbers out to MLDL RAM
 0145 046                    C=0     S&X
 0146 2FC                    RCR     13      ;first nybble into C[0]
 0147 358                    ST=C            ;
 0148 20C                    ?FSET   2       ;is user flag 1 set
 0149 08B                    JNC     [NoSave] +17 015A
 014A 238                    READ    8(P)
 014B 0BE                    A<>C    MS      ;get base to A[MS]
```

```
014C 0B8                    READ    2(Y)    ;get digits
014D 2A0                    SETDEC
014E 2FC       (wl1)        RCR     13      ;shift one left
014F 1BE                    A=A-1   MS
0150 3F3                    JNC     (wl1) -2 014E
0151 0EE                    C<>B    ALL     ;prep for SaveA(i)
0152 178                    READ    5(M)
0153 10E                    A=C     ALL
0154 260                    SETHEX          ;now calc new last address
0155 27A                    C=C-1   M
0156 27A                    C=C-1   M
0157 168                    WRIT    5(M)
0158 39D004                 ?NCXQ   [SaveA(i)] 01E7
015A 149024    [NoSave]     ?NCXQ   [ENCP00] 0952
015C 238                    READ    8(P)
015D 05E                    C=0     MS
015E 0AE                    A<>C    ALL
015F 1F8                    READ    7(O)    ;2*iBits in hex
0160 260                    SETHEX
0161 1CE                    A=A-C   ALL     ;new iStart
0162 238                    READ    8(P)
0163 0BE                    A<>C    MS
0164 0AE                    A<>C    ALL
0165 228                    WRIT    8(P)
0166 138                    READ    4(L)    ;iStart in Dec
0167 0AE                    A<>C    ALL
0168 078                    READ    1(Z)    ;iBits in dec
0169 2A0                    SETDEC
016A 24E                    C=A-C   ALL
016B 057                    JC      [MDOP_Done] +10 0175    ;if underflow, we are definitely done :-)
016C 128                    WRIT    4(L)    ;new iStart in dec, 1 step
016D 0E8                    WRIT    3(X)    ; i in dec, 1 step
016E 0AE                    A<>C    ALL     ;check if we are done
016F 238                    READ    8(P)
0170 05A                    C=0     M
0171 046                    C=0     S&X
0172 2FC                    RCR     13      ;b into C[0]
0173 1CE                    A=A-C   ALL     ;iStart - b > 0
0174 203                    JNC     [StepML1] -64 0134      ;yes -> next loop
0175 149024    [MDOP_Done]  ?NCXQ   [ENCP00] 0952   ;clear MLDL RAM
0177 038                    READ    0(T)    ;read in page number
0178 13C                    RCR     8       ;=-6
0179 0AE                    A<>C    ALL
017A 178                    READ    5(M)    ;read in orig iStart (-stored pi digits if any)
017B 0AE                    A<>C    ALL
017C 040       (cl1)        WROM            ;loop to clear MLDL
017D 000                    NOP
```

```
 017E 23A                    C=C+1   M
 017F 31A                    ?A<C    M
 0180 3E3                    JNC     (cl1) -4 017C
 0181 345040                 ?NCXQ   [CLA] 10D1      ;clear our junk
 0183 3E5042                 ?NCGO   [CLST] 10F9     ;clear out junk and return to OS
;-------------------------------------------------------------------------------------------------------------------
;-------------------------------------------------------------------------------------------------------------------
;MCODE Function 4DEC2HEX
; convert 4-digit dec into hex
;
;---------------
;Some entry ponts
 0185 04E       [Do3rd_EP]   C=0     ALL     ;Clear crap from NCXQREL
 0186 08B                    JNC     [Do3rd+1] +17 0197
;-----------------------------------------------
 0187 04E       [4DEC2HEX]   C=0     ALL     ;4-dig dec number in A
 0188 02E                    B=0     ALL
 0189 130004                 LDIS&X  004
 018B 1BC                    RCR     11      ;=-3
 018C 130096                 LDIS&X  096     ;C[0:3] = 4096
 018E 0EE                    B<>C    ALL
 018F 01C                    R=      3
 0190 2A0       [4dl]        SETDEC
 0191 18E                    A=A-B   ALL
 0192 027                    JC      [Do3rd] +4 0196
 0193 260                    SETHEX
 0194 222                    C=C+1   @R
 0195 3DB                    JNC     [4dl] -5 0190
 0196 12E       [Do3rd]      A=A+B   ALL
 0197 0EE       [Do3rd+1]    C<>B    ALL
 0198 04E                    C=0     ALL
 0199 130256                 LDIS&X  256
 019B 0EE                    C<>B    ALL
 019C 21C                    R=      2
 019D 2A0       [3dl]        SETDEC
 019E 18E                    A=A-B   ALL
 019F 027                    JC      [Do2nd] +4 01A3
 01A0 260                    SETHEX
 01A1 222                    C=C+1   @R
 01A2 3DB                    JNC     [3dl] -5 019D
 01A3 12E       [Do2nd]      A=A+B   ALL
 01A4 0EE       [Do2nd+1]    C<>B    ALL
 01A5 130016                 LDIS&X  016
 01A7 0EE                    C<>B    ALL
 01A8 31C                    R=      1
 01A9 2A0       [2dl]        SETDEC
 01AA 18E                    A=A-B   ALL
```

```
 01AB 027                       JC      [Do1st] +4 01AF
 01AC 260                       SETHEX
 01AD 222                       C=C+1   @R
 01AE 3DB                       JNC     [2dl] -5 01A9
 01AF 12E       [Do1st]         A=A+B   ALL
 01B0 342       [Do1d]          ?A#0    @R      ;do we have 10-15 left?
 01B1 033                       JNC     [DoLast] +6 01B7
 01B2 260                       SETHEX
 01B3 22E                       C=C+1   ALL
 01B4 2A0                       SETDEC
 01B5 1AE                       A=A-1   ALL
 01B6 3D3                       JNC     [Do1d] -6 01B0
 01B7 260       [DoLast]        SETHEX
 01B8 20E                       C=C+A   ALL
 01B9 3E0       [4DHDone]       RTN
;------------------------------------------------------------------------------
;------------------------------------------------------------------------------
;MCODE Function GetA(i)
; Get A(i) stored in 2 words in MLDL-RAM
; right nybble is in hex, left nybble is in dec
; Input
; Last left nybble address in A[M]
; Ouput
; A(i) in dec in C[ALL], left nybble address in A[M]
;SETHEX expected
;------------------------------------------------------------------------------
 01BA 1BA       [GetA(i)]       A=A-1   M       ;next nybble
 01BB 0BA11A                    C=A     M
 01BD 330                       FETCHS&X                ;right nybble in C[S&X], still in hex
 01BE 006                       A=0     S&X
 01BF 0E6                       B<>C    S&X
 01C0 130256                    LDIS&X  256
 01C2 0E6                       B<>C    S&X
 01C3 21C                       R=      2       ;first digit
 01C4 260       [Do3_h2d]       SETHEX
 01C5 262                       C=C-1   @R
 01C6 027                       JC      [Prep2_h2d] +4 01CA
 01C7 2A0                       SETDEC
 01C8 126                       A=A+B   S&X
 01C9 3DB                       JNC     [Do3_h2d] -5 01C4
 01CA 222       [Prep2_h2d]     C=C+1   @R
 01CB 0E6                       B<>C    S&X
 01CC 130016                    LDIS&X  016
 01CE 0E6                       B<>C    S&X
 01CF 31C                       R=      1
 01D0 260       [Do2_h2d]       SETHEX
 01D1 262                       C=C-1   @R
```

```
 01D2 027                      JC      [Prep1_h2d] +4 01D6
 01D3 2A0                      SETDEC
 01D4 126                      A=A+B   S&X
 01D5 3DB                      JNC     [Do2_h2d] -5 01D0
 01D6 222       [Prep1_h2d]    C=C+1   @R
 01D7 39C                      R=      0
 01D8 260       [Do1_h2d]      SETHEX
 01D9 262                      C=C-1   @R
 01DA 027                      JC      [DoneH2D] +4 01DE
 01DB 2A0                      SETDEC
 01DC 166                      A=A+1   S&X
 01DD 3DB                      JNC     [Do1_h2d] -5 01D8
 01DE 1BA       [DoneH2D]      A=A-1   M       ;A[S&X] is right nybble in dec
 01DF 0BA11A                   C=A     M
 01E1 330                      FETCHS&X                ;left nybble
 01E2 05E                      C=0     MS
 01E3 05A                      C=0     M
 01E4 1BC                      RCR     11      ;=-3
 01E5 0A6                      A<>C    S&X     ;A(i) in C[0:5], last address in A[M]
 01E6 3E0                      RTN
;------------------------------------------------------------------------------
;------------------------------------------------------------------------------
;MCODE Function SaveA(i)
; Store A(i) in 2 words in MLDL-RAM
; right nybble is in hex, left nybble is in dec
; Input
; A[M]: i, right Nybble
; B[All] = A(i)
;
; Output
; i in C[M] for left Nybble
;------------------------------------------------------------------------------
 01E7 04E       [SaveA(i)]     C=0     ALL     ;clear crap from NCXQREL
 01E8 066                      A<>B    S&X     ;dec->hex right Nybble from A(i)
 01E9 130256                   LDIS&X  256
 01EB 0E6                      B<>C    S&X
 01EC 046                      C=0     S&X
 01ED 21C                      R=      2
 01EE 260       (sail1)        SETHEX
 01EF 222                      C=C+1   @R
 01F0 2A0                      SETDEC
 01F1 186                      A=A-B   S&X
 01F2 3E3                      JNC     (sail1) -4 01EE
 01F3 126                      A=A+B   S&X
 01F4 260                      SETHEX
 01F5 262                      C=C-1   @R
 01F6 0E6       (saipl2)       C<>B    S&X
```

```
 01F7 130016                  LDIS&X  016
 01F9 0E6                     C<>B    S&X
 01FA 31C                     R=      1
 01FB 260       (sail2)       SETHEX
 01FC 222                     C=C+1   @R
 01FD 2A0                     SETDEC
 01FE 186                     A=A-B   S&X
 01FF 3E3                     JNC     (sail2) -4 01FB
 0200 126                     A=A+B   S&X
 0201 260                     SETHEX
 0202 262                     C=C-1   @R
 0203 342       (saipl3)      ?A#0    @R      ;doe we have 10,11,12,13,14 or 15?
 0204 033                     JNC     (saipd) +6 020A ;no -> just finish up
 0205 260                     SETHEX
 0206 226                     C=C+1   S&X
 0207 2A0                     SETDEC
 0208 1A6                     A=A-1   S&X
 0209 3D3                     JNC     (saipl3) -6 0203
 020A 260       (saipd)       SETHEX
 020B 206                     C=A+C   S&X
 020C 0BA                     C<>A    M       ;bring in address for right nybble
 020D 040                     WROM
 020E 000                     NOP
 020F 27A                     C=C-1   M       ;now get left nybble
 0210 0EE                     C<>B    ALL
 0211 03C                     RCR     3
 0212 0FA                     C<>B    M
 0213 040                     WROM
 0214 000                     NOP
 0215 3E0                     RTN             ;i in C[M]
;------------------------------------------------------------------------------
;------------------------------------------------------------------------------
;MCODE Function CalcR&Q
; Calculates the quotient q and the remainder r of A(i) / (2i-1)
;
; Input
; A: = A(i)
; B: 2i - 1 in decimal, 1 step
;
; Ouput
; C: q
; A: r
;------------------------------------------------------------------------------
 0216 04E       [CalcRnQ]     C=0     ALL     ;clear crap from >NCXQREL
 0217 2A0                     SETDEC
 0218 32E                     ?A<B    ALL     ;is A(i) < (2i-1) -> res = 0
 0219 360                     ?CRTN           ;just return. C=q=0, A=r=A(i)
```

```
 021A 0A0                       SLCTP           ;find start of A(i)
 021B 39C                       R=      0
 021C 3D4       (crql1)         R=R-1
 021D 342                       ?A#0    @R
 021E 3F3                       JNC     (crql1) -2 021C ;find first dig of A
 021F 0E0                       SLCTQ           ;use Q as counter to shift B
 0220 2DC                       R=      13
 0221 0EE                       C<>B    ALL
 0222 33C                       RCR     1
 0223 2FC       (crql2)         RCR     13      ;=-1
 0224 0E0                       SLCTQ
 0225 3DC                       R=R+1
 0226 0A0                       SLCTP
 0227 2E2                       ?C#0    @R
 0228 3DB                       JNC     (crql2) -5 0223
 0229 0EE                       B<>C    ALL
 022A 0E0                       SLCTQ
 022B 222       (crql3)         C=C+1   @R
 022C 18E                       A=A-B   ALL
 022D 3F3                       JNC     (crql3) -2 022B
 022E 12E                       A=A+B   ALL
 022F 262                       C=C-1   @R
 0230 3AE                       RSHFB   ALL
 0231 3D4                       R=R-1
 0232 2D4                       ?R=     13
 0233 3C3                       JNC     (crql3) -8 022B
 0234 3E0                       RTN             ;q in C, r in A
;------------------------------------------------------------------------------
;------------------------------------------------------------------------------
;MCODE Function Calc_bA(i)+qi
; Calculates the b*A(i) + q*i
;
; Input
; A(i) in A[0:5]
;
;
; Ouput
; C:= b*A(i) + q*i
;
;------------------------------------------------------------------------------
 0235 238       [Calc_bA+qi]    READ    8(P)
 0236 0AE                       A<>C    ALL
 0237 2FC       (cbal1)         RCR     13      ;=-1
 0238 1BE                       A=A-1   MS
 0239 3F3                       JNC     (cbal1) -2 0237
 023A 33C                       RCR     1       ;c=b*A(i)
 023B 158                       M=C             ;store b*A(i) in M
```

```
 023C 278                       READ    9(Q)
 023D 0EE                       B<>C    ALL
 023E 0F8                       READ    3(x)    ;i in dec
 023F 00E                       A=0     ALL
 0240 39C                       R=      0
 0241 2A0                       SETDEC
 0242 05B                       JNC     [Qi_St] +11 024D
 0243 262       [QiL2]          C=C-1   @R      ;counting down
 0244 12E       [QiL1]          A=A+B   ALL     ;running sum
 0245 262                       C=C-1   @R
 0246 3F3                       JNC     [QiL1] -2 0244
 0247 3CE       [QiNxt]         RSHFC   ALL
 0248 0EE                       B<>C    ALL
 0249 2FC                       RCR     13      ;=LSHFC = c*10
 024A 0EE                       B<>C    ALL
 024B 2EE                       ?C#0    ALL     ;done with all digits?
 024C 023                       JNC     [QiDone] +4 0250
 024D 2E2       [Qi_St]         ?C#0    @R      ;is this a 0 digit?
 024E 3CB                       JNC     [QiNxt] -7 0247 ;yes, next digit
 024F 3A3                       JNC     [QiL2] -12 0243 ;no -> do loop
 0250 198       [QiDone]        C=M             ;get back b*A(i)
 0251 20E                       C=C+A   ALL
 0252 3E0                       RTN
;------------------------------------------------------------------------------
*
* GLOBAL SYMBOLS
* SYMBOL        VALUE   TYPE    REFERENCES
* [2dl]         01A9    REL     01AE
* [3dl]         019D    REL     01A2
* [4DEC2HEX]    0187    REL     0055
* [4DHDone]     01B9    REL
* [4dl]         0190    REL     0195
* [CNewiSt]     0140    REL
* [CalcInit]    006A    REL
* [CalcRnQ]     0216    REL     00DF
* [Calc_bA+qi]  0235    REL     00D5 00FE
* [CalciBits]   009C    REL
* [Do1_h2d]     01D8    REL     01DD
* [Do1d]        01B0    REL     01B6
* [Do1st]       01AF    REL     01AB
* [Do2_h2d]     01D0    REL     01D5
* [Do2nd+1]     01A4    REL
* [Do2nd]       01A3    REL     019F
* [Do3_h2d]     01C4    REL     01C9
* [Do3rd+1]     0197    REL     0186
* [Do3rd]       0196    REL     0192
* [Do3rd_EP]    0185    REL     007A 00C0
```

```
* [DoA(1)]      00F7    REL
* [DoLast]      01B7    REL     01B1
* [DoneH2D]     01DE    REL     01DA
* [DoneInit]    0097    REL     0094
* [DoneShift]   0135    REL     0133
* [GetA(i)]     01BA    REL     00CF 00F8
* [Header]      0013    REL     0002
* [InitLoop]    0081    REL
* [InitML]      00C8    REL     0105
* [InitW2]      0074    REL
* [KeyWL]       013E    REL     013F
* [MDOP]        0018    REL     0004
* [MDOP_Done]   0175    REL     016B
* [NoSave]      015A    REL     0149
* [NoWaitC]     0142    REL     013B
* [Nxti]        00EB    REL
* [PCTOC]       00D7    ABS
* [Prep1_h2d]   01D6    REL     01D2
* [Prep2_h2d]   01CA    REL     01C6
* [QiDone]      0250    REL     024C
* [QiL1]        0244    REL     0246
* [QiL2]        0243    REL     024F
* [QiNxt]       0247    REL     024E
* [Qi_St]       024D    REL     0242
* [SaveA(i)]    01E7    REL     00E8 0119 0158
* [ShowDigs]    011C    REL
* [StepML1]     0134    REL     0174
* [StepML2]     0105    REL     0134
* [WR2L]        008B    REL     0096
* [iLoop]       00CE    REL     00F6
* [tmp1]        0049    REL
* [tmp2]        00BA    REL
*
* LOCAL SYMBOLS
* SYMBOL        VALUE   TYPE    REFERENCES
* (cbal1)       0237    REL     0239
* (cl1)         017C    REL     0180
* (crql1)       021C    REL     021E
* (crql2)       0223    REL     0228
* (crql3)       022B    REL     022D 0233
* (da1l1)       010F    REL     0111
* (pil1)        004C    REL     004E
* (pil2)        0071    REL     0073
* (pil3)        00B4    REL     00B7
* (sail1)       01EE    REL     01F2
* (sail2)       01FB    REL     01FF
* (saipd)       020A    REL     0204
```

```
* (saipl2)      01F6    REL
* (saipl3)      0203    REL     0209
* (sdl1)        012E    REL     0131
* (sdl1ep)      012F    REL     012A
* (sdl2)        0129    REL     012D
* (sf0l1)       0023    REL     001F
* (sf0l2)       0024    REL     0022
* (step1)       0106    REL     0104
* (wl1)         014E    REL     0150
*
* EXTERNAL REFERENCES
* SYMBOL        REFERENCED AT
*
* MAINFRAME REFERENCES
* SYMBOL        VALUE   REFERENCES
* [AD2_10]      1807    0044 00AC
* [BCDBIN]      02E3    0029
* [CLA]         10D1    0181
* [CLLCDE]      2CF0    0140
* [CLST]        10F9    0183
* [DSPCRG]      0B26    0136
* [DV2_10]      1898    003F 00A7
* [ENCP00]      0952    0059 0081 0099 00C4 00EB 0102 011C 0142 015A 0175
* [ERRDE]       282D    0031
* [INTFRC]      193B    0047 00AF
* [RSTKB]       0098    013C
*
* A41:  0 WARNINGS(S)
* A41:  0 ERROR(S)
* END
```

Edited: 26 Feb 2009, 6:08 p.m.

Re: 4915 digits of pi - MCODE 41 (19660 digits optional)

*Message #2 Posted by **Xerxes** on 27 Feb 2009, 6:38 a.m.,*
in response to message #1 by PeterP

Impressive! Thanks for this further MCODE example.

Re: 4915 digits of pi - MCODE 41 (19660 digits optional)

*Message #3 Posted by **Antonio Maschio (Italy)** on 27 Feb 2009, 9:03 a.m.,*
in response to message #1 by PeterP

Make an article of it, or it will be buried under thousands of posts in a few months.

Anyway: Impressive!

-- Antonio

Re: 4915 digits of pi - MCODE 41 (19660 digits optional)

*Message #4 Posted by **Valentin Albillo** on 27 Feb 2009, 10:20 a.m.,*
in response to message #1 by PeterP

Hi, **PeterP**:

Impressive work. Just a quick note:

PeterP wrote:

"Also, as there are no 5 or 6 9's consecutively in Pi in the first 20,000 digits, we don't have to worry about that either."

That's not true. The string "999999" (6 9's) ***does*** occur at position **762** counting from the first digit after the decimal point (the 3. is not counted). Just as a visual check:

```
3.14159265358979323846264338327950288419716939937510
  58209749445923078164062862089986280348253421170679
  82148086513282306647093844609550582231725359408128
  48111745028410270193852110555964462294895493038196
  44288109756593344612847564823378678316527120190914
  56485669234603486104543266482133936072602491412737
  24587006606315588174881520920962829254091715364367
  89259036001133053054882046652138414695194151160943
  30572703657595919530921861173819326117931051185480
  74462379962749567351885752724891227938183011949129
  83367336244065664308602139494639522473719070217986
  09437027705392171762931767523846748184676694051320
  00568127145263560827785771342757789609173637178721
  46844090122495343014654958537105079227968925892354
  20199561121290219608640344181598136297747713099605
  187072113499999983729780499510597
                            ^^^^^^
                              |
```

Best regards from V.

Re: 4915 digits of pi - MCODE 41 (19660 digits optional)

*Message #5 Posted by **Katie Wasserman** on 28 Feb 2009, 12:28 p.m.,*
in response to message #1 by PeterP

This is very impressive piece of code, I don't know MCODE but it almost makes me want to learn it! I'm glad that my article helped inspire your obsession for pi calculations maybe you'll figure out a new algorithm for it.

One of the best places I've found to seriously indulge this obsession is Simon Plouffe's home page. Follow the links to the articles sections and dig it. Eventually I found this paper. Read the section on the Borweins (at the very end), there is an amazing, simple formula given relating pi, 10 and e with the very curious statement: "The following is not an identity but is correct to over 42 billion digits."

```
pi ~ ( 1/10^5 * SIGMA(n,exp(-(n^2/10^10)),-inf,+inf) ) ^ 2
```

Although I'm not a mathematician (although I started out that way) how could this not be an identity? Must there at least be a conjecture that this is an identity?

Edited: 28 Feb 2009, 12:31 p.m.

Re: 4915 digits of pi - MCODE 41 (19660 digits optional)

*Message #6 Posted by **Valentin Albillo** on 28 Feb 2009, 9:35 p.m.,*
in response to message #5 by Katie Wasserman

Hi, Katie:

Katie posted:

> *"there is an amazing, simple formula given relating pi, 10 and e with the very curious statement: "The following is not an identity but is correct to over 42 billion digits."*
> *pi ~ (1/10^5 * SIGMA(n,exp(-(n^2/10^10)),-inf,+inf)) ^ 2*
> *Although I'm not a mathematician (although I started out that way) how could this not be an identity? Must there at least be a conjecture that this is an identity? "*

It's **not** an identity and actually it's quite straightforward to demonstrate that it produces about 42.8631472996... { Pi^2*10/Ln(10) } billion correct digits of Pi, but no more.

The demonstration makes use a well-known transformation formula of a theta function. For similar **sums which give a large number of correct digits of some constant but nevertheless eventually differ**, have a look at sum #7 and sum #8 in my Short & Sweet Math Challenge #13: Adding up to infinity

Regards from V.

Edited: 28 Feb 2009, 10:12 p.m.

Re: 4915 digits of pi - MCODE 41 (19660 digits optional)

*Message #7 Posted by **Egan Ford** on 28 Feb 2009, 5:16 p.m.,*
in response to message #1 by PeterP

Awesome. Can you provide a MOD file? Your excellent SDK update doc ends short of documenting the process to create a MOD file. I'd like to give it a try on my 41.

Thanks.

[Return to Index | Top of Index]

Go back to the main exhibit hall