INTEGER RATIOS for FASTER CODE v1.1

by RetroDan@GMail.Com

CONTENTS:

1. INTRODUCTION: INTEGER VS. FLOATING-POINT

Floating-point operations use a lot of memory and run very slow compared to integer operations. Included in this document are integer ratios estimates for many everyday constants. Using these ratios will help produce faster code in a smaller space especially if you can avoid using floating-point math.

2. AVOIDING DIVISION FOR MORE SPEED

Of the four main integer operations (+,-,*,/) division is the slowest. To help you produce even quicker code in minimal space, I have also calculated ratios that are divisible by a power-of-two indicated with (*) since such division can be done without a division routine by either ignoring the lower byte(s) of an operation, or by shifting the result to the right producing a division-by-two, thus avoiding the need to do a long division. For those ratios that have a denominator or 256* or 65536* the division can be accomplished by treating the lower-byte in the case of 256* or the lower-two-bytes in the case of 65536* as the fractional part of your calculations.

Below are listed the powers-of-two that may appear in the denominators and the operations that can be used to avoid using division.

```
    2 – 1 right shift
    4 – 2 right shifts
   16 – 3 right shifts
   32 – 4 right shifts
   64 – 5 right shifts
  128 – 6 right shifts
  256 – drop lowest byte

  512 – drop lowest byte and 1 right shift
 1024 – drop lowest byte and 2 right shifts
 2048 – drop lowest byte and 3 right shifts
 4096 – drop lowest byte and 4 right shifts
 8192 – drop lowest byte and 5 right shifts
16384 – drop lowest byte and 6 right shifts
32768 – drop lowest byte and 7 right shifts
65536 – drop 2 lowest bytes
```

3. ACCURACY AND ERROR ESTIMATES

For increased accuracy the dropped bytes, or shifted bits can be used as fractional parts in further calculations. Error estimates were made with the following formula assuming no bits or bytes are dropped:

```
  100 x (TargetValue – NumeratorEst/DenominatorEst)/TargetValue
```

Naturally if you chop-off or discard part of the result, then the error will be different.

I have limited these calculations to ratios that can be accomplished using single-byte or sixteen-bit routines. If you come across a constant which you would like "fractured" into an integer ratio, please feel free to contact me at RetroDan@GMail.com.

4. EXAMPLE #1: MULTIPLYING 100 BY pi

The following example in assembler shows how we can use the ratios to multiply 100 by pi. From the table we can see the various ratios for pi:

```
      ESTIMATES FOR pi = 3.14159265358979323846543...
      201/64*    =   3.14062500 ERROR = 0.0308014%
      245/78     =   3.14102564 ERROR = 0.0180486%
    51472/16384* =   3.14160156 ERROR = 0.0002836%
    65298/20785  =   3.14159249 ERROR = 0.0000051%
```

In this case we will use the first ratio of 201/64* since 64 is a power-of-two. In the code below we first multiply the numerator of 201 by 100 and the results appear in the register pair R1:R0, then we use right-shifts on these registers to accomplish division by 64 without the use of a division subroutine. The program is about 30 bytes long and executes in about 15 clock cycles.

```
.DEF  ANSL = R0             ;Answer Low Byte
.DEF  ANSH = R1             ;Answer High Byte
.DEF     A = R16            ;General Purpose Registers
.DEF     B = R18            ;

        LDI A,100           ;Load Multiplier
        LDI B,201           ;Load Numerator for PI
        MUL A,B             ;Multiply
        LSR ANSH            ;Shift Right (Division by 2)
        ROR ANSL            ;
        LSR ANSH            ;Shift Right (Division by 4)
        ROR ANSL            ;
        LSR ANSH            ;Shift Right (Division by 8)
        ROR ANSL            ;
        LSR ANSH            ;Shift Right (Division by 16)
        ROR ANSL            ;
        LSR ANSH            ;Shift Right (Division by 32)
        ROR ANSL            ;
        LSR ANSH            ;Shift Right (Division by 64)
        ROR ANSL            ;
```

5. EXAMPLE #2: MULTIPLYING 100 BY 1/pi

The following example in assembler shows how we can use the ratios to multiply 100 by 1/pi. From the table we can see the various ratios for 1/pi:

```
      ESTIMATES FOR 1/pi = 0.318309886...
      81/256*    =   0.31640625 ERROR = 0.5980449%
      78/245     =   0.31836735 ERROR = 0.0180519%
    20861/65536* =   0.31831360 ERROR = 0.0011664%
    20785/65298  =   0.31830990 ERROR = 0.0000051%
```

In this case we will use the first ratio of 81/256* because not only is 256 a power-of-two, division can be accomplished by simply treating the lowest byte as a fraction. In other words if we are interested in the integer part of the answer, we just ignore the lowest byte and treat the second byte of the answer as our integer.

In the code below we first multiply the numerator of 81 by 100 and the results appear in the register pair R1:R0, then to accomplish the division by 256 we treat the low-byte in R0 as the fractional part and the high-byte in R1 as the integer part of the answer. The routine is about 6 bytes long and executes in about 3 clock cycles.

```
.DEF  ANSF = R0             ;Answer Fractional Part
```

```
.DEF   ANS = R1              ;Answer Integer Part
.DEF     A = R16             ;General Purpose Registers
.DEF     B = R18             ;

        LDI A,100            ;Load Multiplier
        LDI B,81             ;Load Numerator for PI
        MUL A,B              ;Multiply
```

6. EXAMPLE #3: MULTIPLYING 100 BY SQRT(2)

The following example shows how we can use the table to multiply 100 by the sqaure root of two. From the table we can see the various ratios for the square root of two:

```
   ESTIMATES FOR SQRT(2) = 1.4142135623730950488...
   181/128*    =     1.41406250 ERROR = 0.0106817%
   239/169     =     1.41420118 ERROR = 0.0008753%
 46341/32768*  =     1.41421509 ERROR = 0.0001079%
 47321/33461   =     1.41421356 ERROR < 0.0000001%
```

For this example we will use the second ratio of 239/169 assuming we want to achieve greater accuracy. In the routine below we multiply 100 by the the numerator of 239 then we divide by 169. The routine is about 50 bytes long and executes in about 235 clock cycles.

```
.DEF ANSL = R0            ;To hold low-byte of answer
.DEF ANSH = R1            ;To hold high-byte of answer
.DEF REML = R2            ;To hold low-byte of remainder
.DEF REMH = R3            ;To hold high-byte of remainder
.DEF   AL = R16           ;To hold low-byte of dividend
.DEF   AH = R17           ;To hold high-byte of dividend
.DEF   BL = R18           ;To hold low-byte of divisor
.DEF   BH = R19           ;To hold high-byte of divisor
.DEF    C = R20           ;Bit Counter

        LDI AL,100        ;Load 100
        LDI BL,239        ;Load Numerator for SQRT(2)
        MUL AL,BL         ;Multiply 100 by Numerator
        MOVW AH:AL,ANSH:ANSL ;Move result to prepare for division
        LDI BL,LOW(169)   ;Load low-byte of denominator of SQRT(2)
        LDI BH,HIGH(169)  ;Load high-byte of denominator
DIV1616:MOVW ANSH:ANSL,AH:AL ;Copy dividend into answer
        LDI C,17          ;Load bit counter
        SUB REML,REML     ;Clear Remainder and Carry
        CLR REMH          ;
LOOP:   ROL ANSL          ;Shift the answer to the left
        ROL ANSH          ;
        DEC C             ;Decrement Counter
         BREQ DONE        ;Exit if sixteen bits done
        ROL REML          ;Shift remainder to the left
        ROL REMH          ;
        SUB REML,BL       ;Try to subtract divisor from remainder
        SBC REMH,BH
         BRCC SKIP        ;If the result was negative then
        ADD REML,BL       ;reverse the subtraction to try again
        ADC REMH,BH       ;
        CLC               ;Clear Carry Flag so zero shifted into A
         RJMP LOOP        ;Loop Back
SKIP:   SEC               ;Set Carry Flag to be shifted into A
         RJMP LOOP
DONE:
```

7. ESTIMATES FOR pi = 3.14159265358979323846254...

```
  201/64*    =   3.14062500 ERROR = 0.0308014%
  245/78     =   3.14102564 ERROR = 0.0180486%
51472/16384* =   3.14160156 ERROR = 0.0002836%
65298/20785  =   3.14159249 ERROR = 0.0000051%
```

8. ESTIMATES FOR 2*pi = 6.283185307...

```
  201/32*    =   6.28125000 ERROR = 0.0308014%
  245/39     =   6.28205128 ERROR = 0.0180486%
51472/8192*  =   6.28320313 ERROR = 0.0002836%
64943/10336  =   6.28318498 ERROR = 0.0000051%
```

9. ESTIMATES FOR pi/2 = 1.570796327...

```
  201/128*   =   1.57031250 ERROR = 0.0308014%
  245/156    =   1.57051282 ERROR = 0.0180486%
51472/32768* =   1.57080078 ERROR = 0.0002836%
52174/33215  =   1.57079633 ERROR < 0.0000001%
```

10. ESTIMATES FOR pi/3 = 1.047197551...

```
   67/64*    =   1.04687500 ERROR = 0.0308014%
  244/233    =   1.04721030 ERROR = 0.0012175%
34315/32768* =   1.04721069 ERROR = 0.0012550%
34546/32989  =   1.04719755 ERROR < 0.0000001%
```

11. ESTIMATES FOR pi/4 = 0.785398163...

```
  201/256*   =   0.78515625 ERROR = 0.0308013%
  183/233    =   0.78540773 ERROR = 0.0012175%
51472/65536* =   0.78540039 ERROR = 0.0002836%
25732/32763  =   0.78539816 ERROR = 0.0000001%
```

12. ESTIMATES FOR 1/pi = 0.318309886...

```
   81/256*   =   0.31640625 ERROR = 0.5980449%
   78/245    =   0.31836735 ERROR = 0.0180519%
20861/65536* =   0.31831360 ERROR = 0.0011664%
20785/65298  =   0.31830990 ERROR = 0.0000051%
```

13. ESTIMATES FOR pi^2 = 9.869604401...

```
  158/16*    =   9.87500000 ERROR = 0.0546688%
  227/23     =   9.86956522 ERROR = 0.0003970%
20213/2048*  =   9.86962891 ERROR = 0.0002483%
54648/5537   =   9.86960448 ERROR = 0.0000008%
```

14. ESTIMATES FOR SQRT(pi) = 1.772453850905516027298167...

```
  227/128*   =   1.77343750 ERROR = 0.0554965%
  218/123    =   1.77235772 ERROR = 0.0054234%
 1815/1024*  =   1.77246094 ERROR = 0.0003998%
60111/33914  =   1.77245385 ERROR = 0.0000002%
```

15. ESTIMATES FOR pi^e = 22.459157718361045473427 15...

```
    180/8*     =  22.50000000 ERROR = 0.1818514%
    247/11     =  22.45454545 ERROR = 0.0205362%
  45996/2048*  =  22.45898438 ERROR = 0.0007718%
  44267/1971   =  22.45915779 ERROR = 0.0000003%
```

16. ESTIMATES FOR e = lim(1+1/n)^n = 2.71828182845904523536028 7...

```
   174/64*    =    2.71875000 ERROR = 0.0172231%
   193/71     =    2.71830986 ERROR = 0.0010312%
 44536/16384* =    2.71826172 ERROR = 0.0007398%
 49171/18089  =    2.71828183 ERROR < 0.0000001%
```

17. ESTIMATES FOR 1/e = 0.367879441...

```
    94/256*   =    0.36718750 ERROR = 0.1880891%
    71/193    =    0.36787565 ERROR = 0.0010311%
24109/65536*  =    0.36787415 ERROR = 0.0014395%
18089/49171   =    0.36787944 ERROR < 0.0000001%
```

18. ESTIMATES FOR SQRT(e) = 1.64872127070012814 68...

```
  211/128*    =    1.64843750 ERROR = 0.0172116%
   61/37      =    1.64864865 ERROR = 0.0044048%
54025/32768*  =    1.64871216 ERROR = 0.0005527%
34361/20841   =    1.64872127 ERROR < 0.0000001%
```

19. ESTIMATES FOR e^e = 15.154262241479264190...

```
  242/16*     =   15.12500000 ERROR = 0.1930958%
  197/13      =   15.15384615 ERROR = 0.0027457%
 7759/512*    =   15.15429688 ERROR = 0.0002285%
58844/3883    =   15.15426217 ERROR = 0.0000005%
```

20. ESTIMATES FOR e^pi = 23.140692632779269006...

```
  185/8*     =    23.12500000 ERROR = 0.0678140%
  162/7      =    23.14285714 ERROR = 0.0093537%
47392/2048*  =    23.14062500 ERROR = 0.0002923%
10691/462    =    23.14069264 ERROR < 0.0000001%
```

21. ESTIMATES FOR RADIANS FROM DEGREES = pi/180 = 0.01745329251994…

```
   4/256*    =     0.01562500 ERROR = 10.4753445% (Large Error!)
   3/172     =     0.01744186 ERROR = 0.0655008%
1144/65536*  =     0.01745605 ERROR = 0.0158261%
  71/4068    =     0.01745329 ERROR = 0.0000085%
```

22. ESTIMATES FOR RADIANS/s FROM Hz = 1/(2*pi) = 0.159154943...

```
   41/256*    =     0.16015625 ERROR = 0.6291397%
   39/245     =     0.15918367 ERROR = 0.0180519%
10430/65536*  =     0.15914917 ERROR = 0.0036273%
10336/64943   =     0.15915495 ERROR = 0.0000052%
```

23. ESTIMATES FOR RADIANS/s FROM RPMs = 60/(2*pi) = 9.549296586...

```
   153/16*     =     9.56250000 ERROR = 0.1382658%
   191/20      =     9.55000000 ERROR = 0.0073661%
 39114/4096*   =     9.54931641 ERROR = 0.0002076%
   678/71      =     9.54929577 ERROR = 0.0000085%
```

24. ESTIMATES FOR DEGREES FROM RADIANS = 180/pi = 57.295779513082...

```
   229/4*      =    57.25000000 ERROR = 0.0799003%
   172/3       =    57.33333333 ERROR = 0.0655438%
 58671/1024*   =    57.29589844 ERROR = 0.0002076%
  4068/71      =    57.29577465 ERROR = 0.0000085%
```

25. ESTIMATED FOR DEGREES/s FROM Hz = 1/360 = 0.0027777777...

```
     1/256*    =     0.00390625 ERROR = 40.625000% (Large Error!)
     1/360     =     0.00277777 ERROR = 0.0000000% (Exact)
   182/65536*  =     0.00277710 ERROR = 0.0244141%
```

26. ESTIMATES FOR DEGREES/s FROM RPMs = 1/6 = 0.1666666...

```
     1/6       =     0.16666666 ERROR = 0.0000000% (Exact)
    42/256*    =     0.16406250 ERROR = 1.5625000%
10923/65536    =     0.16667175 ERROR = 0.0030518%
```

27. ESTIMATES FOR RPMs FROM RADIANS/s = (2*pi)/60 = 0.104719755...

```
    27/256*    =     0.10546875 ERROR = 0.7152375%
    20/191     =     0.10471204 ERROR = 0.0073655%
  6863/65536*  =     0.10472107 ERROR = 0.0012551%
    71/678     =     0.10471976 ERROR = 0.0000086%
```

28. ESTIMATES FOR RPMs FROM Hz = 1/60 =0.016666666666...

```
     4/256*    =     0.01562500 ERROR = 6.2500000% (Large Error!)
     1/60      =     0.01666666 ERROR = 0.0000000% (Exact)
  1092/65536*  =     0.01666260 ERROR = 0.0244141%
```

29. ESTIMATES FOR Hz FROM RADIANS/s = 2*pi = 6.283185307...

```
   201/32*     =     6.28125000 ERROR = 0.0308014%
   245/39      =     6.28205128 ERROR = 0.0180486%
 51472/8192*   =     6.28320313 ERROR = 0.0002836%
 64943/10336   =     6.28318498 ERROR = 0.0000051%
```

30. ESTIMATES FOR SQRT(2) = 1.4142135623730950488...

```
   181/128*    =     1.41406250 ERROR = 0.0106817%
   239/169     =     1.41420118 ERROR = 0.0008753%
 46341/32768*  =     1.41421509 ERROR = 0.0001079%
 47321/33461   =     1.41421356 ERROR < 0.0000001%
```

31. ESTIMATES FOR SQRT(3) = 1.7320508075688772935...

```
  222/128*    =     1.73437500 ERROR = 0.1341873%
  168/97      =     1.73195876 ERROR = 0.0053142%
56756/32768*  =     1.73205566 ERROR = 0.0002804%
51409/29681   =     1.73205081 ERROR < 0.0000001%
```

32. ESTIMATES FOR SQRT(5) = 2.2360679774997896964...

```
  143/64*     =     2.23437500 ERROR = 0.0757123%
  161/72      =     2.23611111 ERROR = 0.0019290%
36636/16384*  =     2.23608398 ERROR = 0.0007158%
51841/23184   =     2.23606798 ERROR < 0.0000001%
```

33. ESTIMATES FOR SQRT(6) = 2.449489743...

```
  157/64*     =     2.45312500 ERROR = 0.1484087%
  218/89      =     2.44943820 ERROR = 0.0021041%
40132/16384*  =     2.44946289 ERROR = 0.0010962%
47525/19402   =     2.44948974 ERROR < 0.0000001%
```

34. ESTIMATES FOR SQRT(7) = 2.645751311...

```
  169/64*     =     2.64062500 ERROR = 0.1937563%
  127/48      =     2.64583333 ERROR = 0.0031002%
43348/16384*  =     2.64575195 ERROR = 0.0000243%
32257/12192   =     2.64575131 ERROR = 0.0000001%
```

35. ESTIMATES FOR SQRT(8) = 2.828427125...

```
  181/64*     =     2.82812500 ERROR = 0.0106817%
   99/35      =     2.82857143 ERROR = 0.0051019%
46341/16384*  =     2.82843018 ERROR = 0.0001079%
19601/6930    =     2.82842713 ERROR = 0.0000001%
```

36. ESTIMATES FOR SQRT(10) = 3.16227766...

```
  202/64*     =     3.15625000 ERROR = 0.1906113%
  117/37      =     3.16216216 ERROR = 0.0036524%
51811/16384*  =     3.16229248 ERROR = 0.0004687%
27379/8658    =     3.16227766 ERROR = 0.0000001%
```

37. ESTIMATES FOR CUBE_ROOT(2) = 1.25992105...

```
  161/128*    =     1.25781250 ERROR = 0.1673557%
  223/177     =     1.25988701 ERROR = 0.0027021%
41285/32768*  =     1.25991821 ERROR = 0.0002252%
60005/47626   =     1.25992105 ERROR = 0.0000001%
```

38. ESTIMATES FOR CUBE_ROOT(3) = 1.44224957...

```
  185/128*    =     1.44531250 ERROR = 0.2123717%
   75/52      =     1.44230769 ERROR = 0.0040300%
47260/32768*  =     1.44226074 ERROR = 0.0007746%
59650/41359   =     1.44224957 ERROR = 0.0000001%
```

39. ESTIMATES FOR CUBE_ROOT(4) = 1.587401052...

```
   203/128*    =     1.58593750 ERROR = 0.0921980%
   227/143     =     1.58741259 ERROR = 0.0007267%
  3251/2048*   =     1.58740234 ERROR = 0.0000814%
  4813/3032    =     1.58740106 ERROR = 0.0000002%
```

40. ESTIMATES FOR CUBE_ROOT(5) = 1.709975947...

```
   219/128*    =     1.71093750 ERROR = 0.0562320%
   171/100     =     1.71000000 ERROR = 0.0014066%
  1751/1024*   =     1.70996094 ERROR = 0.0008778%
 41944/24529   =     1.70997595 ERROR < 0.0000001%
```

41. ESTIMATES FOR CUBE_ROOT(6) = 1.817120593...

```
   233/128*    =     1.82031250 ERROR = 0.1756574%
   149/82      =     1.81707317 ERROR = 0.0026097%
 59543/32768*  =     1.81710815 ERROR = 0.0006845%
   467/257     =     1.81712062 ERROR = 0.0000016%
```

42. ESTIMATES FOR CUBE_ROOT(7) = 1.912931183...

```
   245/128*    =     1.91406250 ERROR = 0.0591405%
    44/23      =     1.91304348 ERROR = 0.0058703%
 62683/32768*  =     1.91293335 ERROR = 0.0001133%
 33329/17423   =     1.91293118 ERROR < 0.0000001%
```

43. ESTIMATES FOR CUBE_ROOT(9) = 2.080083823...

```
   133/64*     =     2.07812500 ERROR = 0.0941704%
    52/25      =     2.08000000 ERROR = 0.0040298%
  1065/512*    =     2.08007813 ERROR = 0.0002739%
 50623/24337   =     2.08008382 ERROR < 0.0000001%
```

44. ESTIMATES FOR CUBE_ROOT(10) = 2.15443469...

```
    69/32*     =     2.15625000 ERROR = 0.0842592%
   237/110     =     2.15454545 ERROR = 0.0051412%
 35298/16384*  =     2.15441895 ERROR = 0.0007308%
 59415/27578   =     2.15443469 ERROR = 0.0000002%
```

45. ESTIMATES FOR FIFTH_ROOT(2) = 1.148698355...

```
   147/128*    =     1.14843750 ERROR = 0.0227087%
   224/195     =     1.14871795 ERROR = 0.0017057%
 37641/32768*  =     1.14871216 ERROR = 0.0012016%
 40286/35071   =     1.14869835 ERROR < 0.0000001%
```

46. ESTIMATES FOR FIFTH_ROOT(3) = 1.245730940...

```
   159/128*    =     1.24218750 ERROR = 0.2844467%
   218/175     =     1.24571429 ERROR = 0.0013369%
 10205/8192*   =     1.24572754 ERROR = 0.0002730%
 64124/51475   =     1.24573094 ERROR = 0.0000002%
```

47. ESTIMATES FOR LOG10(2) = 0.301029995...

```
    77/256*   =      0.30078125 ERROR = 0.0826313%
    59/196    =      0.30102041 ERROR = 0.0031847%
 19728/65536* =      0.30102539 ERROR = 0.0015295%
  8651/28738  =      0.30103000 ERROR < 0.0000001%
```

48. ESTIMATES FOR LOG10(3) = 0.477121254...

```
   122/256*   =      0.47656250 ERROR = 0.1171094%
    73/153    =      0.47712418 ERROR = 0.0006139%
 31269/65536* =      0.47712708 ERROR = 0.0012201%
 24483/51314  =      0.47712125 ERROR = 0.0000001%
```

49. ESTIMATES FOR LOG10(4) = 0.602059991...

```
   254/256*   =      0.60156250 ERROR = 0.0826315%
    59/98     =      0.60204082 ERROR = 0.0031848%
 39457/65536* =      0.60206604 ERROR = 0.0010047%
 33961/56408  =      0.60205999 ERROR = 0.0000001%
```

50. ESTIMATES FOR LOG10(5) = 0.698970004...

```
   179/256*   =      0.69921875 ERROR = 0.0355875%
   137/196    =      0.69897959 ERROR = 0.0013717%
 45808/65536* =      0.69897461 ERROR = 0.0006589%
 29384/42039  =      0.69897000 ERROR < 0.0000001%
```

51. ESTIMATES FOR LOG10(6) = 0.77815125...

```
   199/256*   =      0.77734375 ERROR = 0.1037716%
   193/248    =      0.77822581 ERROR = 0.0095812%
 50997/65536* =      0.77815247 ERROR = 0.0001562%
   463/595    =      0.77815126 ERROR = 0.0000013%
```

52. ESTIMATES FOR LOG10(7) = 0.84509804...

```
   216/256*   =      0.84375000 ERROR = 0.1595129%
    60/71     =      0.84507042 ERROR = 0.0032680%
 55384/65536* =      0.84509277 ERROR = 0.0006232%
   431/510    =      0.84509804 ERROR = 0.0000001%
```

53. ESTIMATES FOR LOG10(8) = 0.903089987...

```
   231/256*   =      0.90234375 ERROR = 0.0826315%
   205/227    =      0.90308370 ERROR = 0.0006961%
 59185/65536* =      0.90309143 ERROR = 0.0001599%
 51263/56764  =      0.90308999 ERROR < 0.0000001%
```

54. ESTIMATES FOR LOG10(9) = 0.954242509...

```
   244/256*   =      0.95312500 ERROR = 0.1171095%
   146/153    =      0.95424837 ERROR = 0.0006138%
 62537/65536* =      0.95423889 ERROR = 0.0003791%
 17038/17855  =      0.95424251 ERROR < 0.0000001%
```

55. ESTIMATES FOR LOG10(e) = 0.434294481...

```
   111/256*   =      0.43359375 ERROR = 0.1613493%
    76/175    =      0.43428571 ERROR = 0.0020186%
28462/65536* =      0.43429565 ERROR = 0.0002702%
12456/28681  =      0.43429448 ERROR = 0.0000001%
```

56. ESTIMATES FOR ln(2) = 0.69314718...

```
   177/256*   =      0.69140625 ERROR = 0.2511631%
   131/189    =      0.69312169 ERROR = 0.0036770%
45426/65536* =      0.69314575 ERROR = 0.0002060%
43888/63317  =      0.69314718 ERROR < 0.0000001%
```

57. ESTIMATES FOR ln(3) = 1.098612289...

```
   141/128*   =      1.10156250 ERROR = 0.2685398%
    78/71     =      1.09859155 ERROR = 0.0018878%
35999/32768* =      1.09860229 ERROR = 0.0009097%
24621/22411  =      1.09861229 ERROR < 0.0000001%
```

58. ESTIMATE FOR ln(4) = 1.386294361...

```
   177/128*   =      1.38281250 ERROR = 0.2511632%
    61/44     =      1.38636364 ERROR = 0.0049972%
45426/32768* =      1.38629150 ERROR = 0.0002061%
25469/18372  =      1.38629436 ERROR < 0.0000001%
```

59. ESTIMATE FOR ln(5) = 1.609437912...

```
   103/64*    =      1.60937500 ERROR = 0.0039089%
52738/32768* =      1.60943604 ERROR = 0.0001166%
 9993/6209   =      1.60943791 ERROR < 0.0000001%
```

60. ESTIMATES FOR ln(6) = 1.791759469...

```
   229/128*   =      1.78906250 ERROR = 0.1505207%
    43/24     =      1.79166667 ERROR = 0.0051794%
58712/32768* =      1.79174805 ERROR = 0.0006375%
 1609/898    =      1.79175947 ERROR = 0.0000002%
```

61. ESTIMATES FOR ln(7) = 1.945910149...

```
   249/128*   =      1.94531259 ERROR = 0.0307131%
    72/37     =      1.94594595 ERROR = 0.0018396%
63764/32768* =      1.94592285 ERROR = 0.0006528%
54359/27935  =      1.94591015 ERROR < 0.0000001%
```

62. ESTIMATES FOR ln(8) = 2.079441542...

```
   133/64*    =      2.07812500 ERROR = 0.0633123%
   131/63     =      2.07936508 ERROR = 0.0036771%
34070/16384* =      2.07946777 ERROR = 0.0012615%
25469/12248  =      2.07944154 ERROR < 0.0000001%
```

63. ESTIMATES FOR ln(9) = 2.197224577...

```
   141/64*    =      2.20312500 ERROR = 0.2685398%
   156/71     =      2.19718310 ERROR = 0.0018878%
 35999/16384* =      2.19720459 ERROR = 0.0009097%
 49242/22411  =      2.19722458 ERROR < 0.0000001%
```

64. ESTIMATES FOR ln(10) = 2.302585093...

```
   147/64*    =      2.29687500 ERROR = 0.2479862%
   175/76     =      2.30263158 ERROR = 0.0020189%
 37726/16384* =      2.30261230 ERROR = 0.0011818%
 53443/23210  =      2.30258509 ERROR < 0.0000001%
```

65. ESTIMATES FOR CENTIMETERS FROM INCHES = 2.54

```
   163/64*    =      2.54687500 ERROR = 0.2706693%
   127/50     =      2.54000000 ERROR = 0.0000000% (Exact)
 41615/16384* =      2.53997803 ERROR = 0.0008651%
```

66. ESTIMATES FOR INCHES FROM CENTIMETERS = 0.3937007...

```
   101/256*   =      0.39453125 ERROR = 0.2109597%
    50/127    =      0.39370079 ERROR = 0.0000000% (Exact)
 25802/65536* =      0.39370728 ERROR = 0.0016701%
```

67. ESTIMATES FOR METERS FROM FEET = 3.280839895...

```
   105/32*    =      3.28125000 ERROR = 0.0125029%
   187/57     =      3.28070175 ERROR = 0.0042076%
 53753/16384* =      3.28082275 ERROR = 0.0005196%
  1250/381    =      3.28083989 ERROR = 0.0000000% (Exact)
```

68. ESTIMATES FOR FEET FROM METERS = 0.3048

```
    78/256*   =      0.30468750 ERROR = 0.0369094%
    57/187    =      0.30481283 ERROR = 0.0042107%
 19975/65536* =      0.30479431 ERROR = 0.0018663%
   381/1250   =      0.30480000 ERROR = 0.0000000% (Exact)
```

69. ESTIMATES FOR METERS/s FROM Km/Hr = 3.60

```
    18/5      =      3.60000000 ERROR = 0.0000000% (Exact)
   115/32*    =      3.59375000 ERROR = 0.1736111%
 29491/8192*  =      3.59997559 ERROR = 0.0006782%
```

70. ESTIMATES FOR METERS/s FROM FEET/s = 3.280840...

```
   105/32*    =      3.28125000 ERROR = 0.0124968%
   187/57     =      3.28070175 ERROR = 0.0042137%
 53753/16384* =      3.28082275 ERROR = 0.0005257%
 64521/19666  =      3.28084003 ERROR = 0.0000009%
```

71. ESTIMATES FOR METERS/s FROM MPH = 2.236936...

```
   143/64*    =      2.23437500 ERROR = 0.1144870%
    85/38     =      2.23684211 ERROR = 0.0041975%
 18325/8192*  =      2.23693848 ERROR = 0.0001107%
 60187/26906  =      2.23693600 ERROR < 0.0000001%
```

72. ESTIMATES FOR FEET/s FROM Km/Hr = 1.097280

```
    35/32*    =      1.09375000 ERROR = 0.3217046%
   203/185    =      1.09729730 ERROR = 0.0015764%
  8989/8192*  =      1.09729004 ERROR = 0.0009149%
  3429/3125   =      1.09728000 ERROR = 0.0000000% (Exact)
```

73. ESTIMATES FOR FEET/s FROM METERS/s = 0.30480

```
    78/256*   =      0.30468750 ERROR = 0.0369094%
    57/187    =      0.30481283 ERROR = 0.0042107%
 19975/65536* =      0.30479431 ERROR = 0.0018663%
   381/1250   =      0.30480000 ERROR = 0.0000000% (Exact)
```

74. ESTIMATES FOR FEET/s FROM MPH = 0.681818...

```
   175/256*   =      0.68359375 ERROR = 0.2604434%
    15/22     =      0.68181818 ERROR < 0.0000001%
 44684/65536  =      0.68182373 ERROR = 0.0008405%
```

75. ESTIMATES FOR MPH FROM Km/Hr = 1.6093440

```
   103/64*    =      1.60937500 ERROR = 0.0019263%
 52735/32768* =      1.60934448 ERROR = 0.0000300%
 25146/15625  =      1.60934400 ERROR < 0.0000001%
```

76. ESTIMATES FOR MPH FROM METERS/s = 0.447040

```
   114/256*   =      0.44531250 ERROR = 0.3864307%
    38/85     =      0.44705882 ERROR = 0.0042107%
 29297/65536  =      0.44703674 ERROR = 0.0007285%
  1397/3125   =      0.44704000 ERROR = 0.0000000% (Exact)
```

77. ESTIMATES FOR MPH FROM FEET/s = 1.466666...

```
    47/32*    =      1.46875000 ERROR = 0.1420455%
    22/15     =      1.46666666 ERROR = 0.0000000% (Exact)
 12015/8192*  =      1.46667480 ERROR = 0.0005549%
```

78. ESTIMATES FOR ACCELERATION DUE TO GRAVITY IN ft/s^2 ~= 32.11740...

```
    32        =     32.00000000 ERROR ~= 0.3655339%
   225/7      =     32.14285714 ERROR ~= 0.0792628%
  4111/128*   =     32.11718750 ERROR ~= 0.0006616%
 15320/477    =     32.11740042 ERROR ~= 0.0000013%
```

79. ESTIMATES FOR ACCELERATION DUE TO GRAVITY IN m/s^2 ~= 9.780327...

```
    39/4*     =      9.75000000 ERROR ~= 0.3100817%
   225/23     =      9.78260870 ERROR ~= 0.0233294%
```

```
 10015/1024*  =      9.78027344 ERROR ~= 0.0005477%
 37087/3792   =      9.78032700 ERROR  < 0.0000001%
```

80. ESTIMATES FOR THE GOLDEN RATIO Q = [1+ROOT(5)]/2 = 1.6180339887...

```
   207/128*   =      1.61718750 ERROR = 0.0523158%
   233/144    =      1.61805556 ERROR = 0.0013329%
 13255/8192*  =      1.61804199 ERROR = 0.0004947%
 46368/28657  =      1.61803399 ERROR < 0.0000001%
```

81. ESTIMATES FOR THE INVERSE GOLDEN RATIO 1/Q = 0.6180339887...

```
   158/256*   =      0.61718750 ERROR = 0.1369646%
   144/233    =      0.61802575 ERROR = 0.0013328%
 40503/65536* =      0.61802673 ERROR = 0.0011738%
 17711/28657  =      0.61803399 ERROR < 0.0000001%
```

82. ESTIMATES FOR THE GOLDEN ANGLE IN DEGREES = 137.5077641...

```
   255/2*     =    127.5000000  ERROR = 7.2779629% (Large Error!)
    137       =    137.0000000  ERROR = 0.3692621%
 35202/256*   =    137.5078125  ERROR = 0.0000352%
 62016/451    =    137.5077605  ERROR = 0.0000026%
```

83. ESTIMATES FOR THE GOLDEN ANGLE IN RADIANS = 2.39996323...

```
    77/32*    =      2.40625000 ERROR = 0.2619528%
    12/5      =      2.40000000 ERROR = 0.0015321%
 39321/16384* =      2.39996338 ERROR = 0.0000062%
 52216/21757  =      2.39996323 ERROR < 0.0000001%
```

84. ESTIMATES FOR EULER CONSTANT y=SUM[(1/k)-ln(n)]=0.57721566490153286060651...

```
   148/256*   =      0.57812500 ERROR = 0.1575382%
    71/123    =      0.57723577 ERROR = 0.0034835%
 37828/65536* =      0.57720947 ERROR = 0.0010728%
 33841/58628  =      0.57721566 ERROR < 0.0000001%
```

85. ESTIMATES FOR e^y = 1.7810724179901979852...

```
   228/128*   =      1.78125000 ERROR = 0.0099705%
   244/137    =      1.78102190 ERROR = 0.0028365%
 58362/32768* =      1.78106689 ERROR = 0.0003101%
 24447/13726  =      1.78107242 ERROR < 0.0000001%
```

86. ESTIMATE FOR LIOUVILLES CONSTANT SUM(10^n!)=0.110001000000000000000001...

```
    28/256*   =      0.10937500 ERROR = 0.5690857%
    11/100    =      0.11000000 ERROR = 0.0009091%
  7209/65536* =      0.11000061 ERROR = 0.0003542%
  6595/59954  =      0.11000100 ERROR = 0.0000007%
```

87. ESTIMATES FOR CATALAN'S CONSTANT SUM[(-1)^n/(2n+1)^2] = 0.915965594...

```
   234/256*   =      0.91406250 ERROR = 0.2077692%
   109/119    =      0.91596639 ERROR = 0.0000865%
 60029/65536* =      0.91596985 ERROR = 0.0004645%
 48559/53014  =      0.91596559 ERROR < 0.0000001%
```

88. ESTIMATES FOR gamma(1/2) = (1/2 -1)! = 1.77245385090551602729816 7...

```
     227/128*   =   1.77343750 ERROR = 0.0554965%
     218/123    =   1.77235772 ERROR = 0.0054234%
    1815/1024*  =   1.77246094 ERROR = 0.0003998%
   60111/33914  =   1.77245385 ERROR = 0.0000002%
```

89. ESTIMATES FOR gamma(1/3) = (1/3 -1)! = 2.678938534707748...

```
    171/64*     =   2.67187500 ERROR = 0.2636692%
    217/81      =   2.67901235 ERROR = 0.0027552%
  43892/16384*  =   2.67895508 ERROR = 0.0006175%
  25541/9534    =   2.67893854 ERROR < 0.0000001%
```

90. ESTIMATES FOR gamma(1/4) = (1/4 – 1)! = 3.625609908221908...

```
     29/8*      =   3.62500000 ERROR = 0.0168222%
  59402/16384*  =   3.62561035 ERROR = 0.0000122%
  57959/15986   =   3.62560991 ERROR < 0.0000001%
```

If you come across a constant which you would like "fractured" into an integer ratio, please feel free to contact me at RetroDan@GMail.com.