# Julian Day Numbers With Forth

*Leonard F. Zettel*

*3820 Brookshire Dr. Trenton, Michigan 48183*

*Abstract*

The Julian Day Number system of time reckoning is described. Forth words for converting between Julian Day number and either Julian or Gregorian calendar dates are given.

## *Introduction*

The Julian Day Number of a calendar date is the number of days that have elapsed between it and noon of January 1, 4713 B.C. of our civil calendar. Astronomers universally record dates as Julian day numbers.

Keeping dates as Julian day numbers in a computer has a number of advantages. A unique representation of any date likely to be encountered can be placed in a double number. The number of days between any two dates is the difference of their Julian day numbers. The day of the week of a date can be found by dividing the Julian day number by seven. A remainder of 0 indicates Tuesday, 1 Wednesday, etc.

Conversion routines like those given here always remember about leap year, including the century exceptions of the Gregorian calendar. If the conversions into and out of Julian day number are known for a given calendar system, the day numbers allow quick conversions between that calendar and any other that can also be converted. [HAR83] gives conversions for the Jewish, Muslim, Chinese, and ancient Egyptian calendars among others.

In constructing these routines I followed the strictures of [SIN84]. Thus we manipulate not only the integer date but the fraction of day as well, expressed as the number of 1/65,536ths of a day. This gives us a total representation accuracy of about 1.3 seconds, which should be enough for most purposes, especially since current chronological practice involves sticking one or more"leap seconds" into the year at irregular intervals, and the algorithms used here do not take account of that. We also keep to the strict astronomical convention that the integral Julian day number changes at noon.

The routines should work for any date with a positive Julian day number up to A.D. 32767, which is the highest single precision signed number Forth handles. Years B.C. can be used following the convention that 1 B.C. is the year 0, 2 B.C. -1 etc. We also have provisions for handling either the Julian calendar (the calendar in common use in Christendom before the sixteenth century) or the Gregorian calendar, the present civil calendar of most of the world. The words can also be used to convert between Julian and Gregorian calendar dates, which can be useful when studying historical periods like that of the Russian revolution, since the Czarists used the Julian calendar and the Soviets the Gregorian.

## *The Algorithms*

Screens 1 and 2 give some extensions to the basic Forth vocabulary that will be used in the screens that follow. Versions of Forth that already have words like `D*` and `D/` may not need them.

Screen 3 sets up the apparatus for discriminating between the Julian and Gregorian calendars. The first 4 bytes of the storage area `GREGORIAN-BREAK` hold a double number that is a compressed form of the calendar date at which to switch between the two calendars. The rightmost two digits are the day of the month, the next two are the month of the year, and the next four are the year. The second four bytes of `GREGORIAN-BREAK` hold the corresponding Julian day number. `CALENDAR=` is a defining word that creates a class of words that will load appropriate constants into `GREGORIAN-BREAK`.

`CATHOLIC-CALENDAR` sets `GREGORIAN-BREAK` to the initial date of adoption of the Gregorian calendar by France, Italy, Luxembourg, Portugal, and Spain. `GREGORIAN-CALENDAR` forces all conversions to the Gregorian calendar, and `JULIAN-CALENDAR` forces them to the Julian. `BRITISH-CALENDAR` marks the date the British empire (including the American colonies) switched over. (Being protestant, they tended to be suspicious of popish contrivances, even when they were a good idea). The author would appreciate receiving documented constants for the calendar changeover dates of other countries and regions of the world.

`?GREGORIAN` returns `TRUE` if the year, month and day on the stack are to be treated as of the Gregorian calendar, `FALSE` if they are Julian calendar.

`HMS->.DAY` converts time of day in hours, minutes, and seconds to 65,536ths of a day. To do this we first convert the hours, minutes, and seconds to a whole number of seconds and then scale by 512/675, which is a reduced form of 65,536/86,400, where 86,400 is the number of seconds in a day. `.DAY->HMS` goes the other way, from 65,536ths of a day to hours minutes and seconds. Screens 5 and 6 have various decimal - binary conversions. Their major use is as a convenience in providing mechanisms for keyboard/screen i/o for the actual calendar conversion routines, screens 7 and 8.

`YMD.B->JD.B` generally follows [MEE82], with adjustments to make it reasonable Forth. First we put the fractional part of the day on the return stack for safekeeping, and then find out whether we are dealing with the Gregorian calendar. Then we add the half day (32768.) to adjust for the Julian day's starting at noon. Line 5 is essentially an adjustment to have March the first month of the year and February the last, because it is the most irregular. Line 6 makes the adjustments (if necessary) for the fewer leap years (three per four centuries) of the Gregorian calendar. `306 10 */` gives an integer result that falls in the cracks just in the right way to take care of the sequence of thirty and thirty-one day months starting with March. Line 8 puts in a correction for negative years. In line 9 we finish the calculation begun on line 7 that recognizes that in the Julian calendar there are 1461/4 days per year. Finally, `1720994. D+` sets the day count to the proper origin.

`YMD->JD` follows the Forth tradition of `/` vs. `/MOD`; for a few more memory cells we have a conversion from calendar to Julian day number of integer days only. Strictly speaking, it gives the Julian day number that begins at noon of the corresponding calendar day.

Screen 9 gives the reverse conversion, from Julian day number to calendar date. On line 6 we again adjust for the half day and check which calendar we are converting to. Lines 7 and 8 take care of the Gregorian calendar adjustments and lines 9 through 14 follow [MEE82], page 26, with constants adjusted to reflect the fact that Forth uses integer arithmetic (actually better suited to this kind of calculation) rather than the floating point of the pocket calculator assumed by Meeus.

## *Instrumentation and Software*

The above routines have been compiled and run with MVP Forth™ version 1.00.03 for the Amiga™ from Commodore. I have endeavored to adhere to standard Forth-79 throughout. Words not defined in this article and not in the 79-Standard can be found in [HAY85]. Their meanings are as follows:

| | |
|---|---|
| `2*` | ( n1 --- n2) Leave 2*(n1). Forth-79 Reference Word Set. |
| `ABORT"` | ( f ---) If the flag is true, print the following text till " , then execute `ABORT`. Otherwise continue execution following the quoted string. Forth-79 Reference Word Set. |

| | |
|---|---|
| `D+-` | ( d1 n --- d2) Apply the sign of n to the double number d1, leaving it as d2. This is the fig-FORTH definition. |
| `DPL` | ( --- addr) A user variable containing the number of digits to the right of the decimal on double integer input. The default value on single number input is -1. This is the usage defined in [FOR83] Uncontrolled Reference Words. |
| `M*` | ( n1 n2 --- d) Leave the double number signed product of two signed numbers. This is the fig-FORTH definition. |
| `M*/` | ( d1 n1 n2 --- d2) Multiplies double number d1 by single n1 and divides the triple precision product by n2 leaving double quotient d2. |
| `M/` | ( d n1 --- n2 n3) Leave the signed remainder n2 and signed quotient n3 from a signed double number dividend and signed divisor n1. The remainder takes its sign from the dividend. This is the fig-FORTH definition. |
| `M/MOD` | ( ud1 u2 --- u3 ud4) Leave an unsigned double quotient ud4 and unsigned remainder u3 from an unsigned double dividend ud1 and unsigned single divisor u2. This is the fig-FORTH definition. |
| `S->D` | ( n --- d) Sign extend a single number to form a double number. This is the fig-FORTH definition. |
| `\` | ( ---) Treat the rest of the line as comments. See [BRO84] p. 282. |

For convenience in checking the routines, some representative calendar dates and their Julian day numbers, taken from [SIN84], are given in table 1. The numbers as given in the table are mathematically exact.

Table 1.

| *Julian calendar* | | | *Gregorian calendar* | | | *Julian date* |
|---|---|---|---|---|---|---|
| −4712 | Jan. | 1.5 | −4713 | Nov. | 24.5 | 0.0 |
| −2000 | Jan. | 1.0 | −2001 | Dec. | 15.0 | 99 0557.5 |
| −584 | May | 28.6 | −584 | May | 22.6 | 150 7900.1 |
| +200 | Mar. | 1.0 | +200 | Mar. | 1.0 | 179 4167.5 |
| +1984 | Feb. | 16.2 | +1984 | Feb. | 29.2 | 244 5759.7 |
| +1999 | Dec. | 19.5 | +2000 | Jan. | 1.5 | 245 1545.0 |
| +3000 | Feb. | 29.9 | +3000 | Mar. | 21.9 | 281 6867.4 |

## *Acknowledgements*

At the suggestion of a referee, `CALENDAR=` replaces clumsier methods of setting `GREGORIAN-BREAK`. Other referee comments led to a number of revisions and extensions to the paper. Of course, any remaining errors are the full responsibility of the author.

## *References*

[BRO84] Brodie, Leo, *Thinking Forth*, Englewood Cliffs NJ: Prentice-Hall, 1984.

[FOR83] Forth Standards Team, *Forth-83 Standard*, Mountain View, CA: Mountain View Press, 1983.

[HAR83] Harvey, O. L. *Calendar Conversions by Way of the Julian Day Number*, Phildelphia: American Philosophical Society, 1983.

[HAY85] Haydon, Glen B. *All about Forth*, 2nd ed., Mountain View, CA: Mountain View Press, 1985.

[MEE82] Meeus, Jean, *Astronomical Formulae for Calculators*, 2nd. ed., Richmond, VA: Willman-Bell Inc., 1982.

[SIN84] Sinnott, Roger W. "Taming Our Chaotic Calendar" *Sky & Telescope*, May 1984, 454-5.

*Mr. Zettel has a B.S. in Chemical Engineering. He received an M.B.A. from Canisius College in 1975. He has been actively involved with computers since 1962, and is currently a researcher at the Ford Motor company.*

Manuscript received March 1986.

```
SCR #1
\ UD* 3DUP UD/U D/N                                    LFZ26Jan86
: UD* ( ud1  ud2 --- q1)  \ Unsigned double precision multiply.
 OVER 5 PICK U* 3 PICK 7 ROLL U* ROT 0 D+ >R 4 ROLL 5 PICK U*
 ROT 0 D+ 4 ROLL 5 ROLL U* ROT 0 D+ R> 0 D+ ;

: 3DUP ( n1 n2 n3 --- n1 n2 n3 n1 n2 n3)
 >R 2DUP R@ ROT ROT R> ;

: UD/U ( ud1 u1 --- ud2)  \ ud2 is ud1 divided by u1.
 M/MOD ROT DROP ;

: D/N ( d1 n1 --- d2)    \ d2 = d1/n1
 OVER >R >R DABS R@ ABS UD/U R> R> XOR D+- ;
```

```
SCR #2
\ UDU* DN*                                          LFZ26Jan86
: UDU* ( ud1 u1 --- d2)  \ d2 = ud1*u1
 >R SWAP R@ U* ROT R> * + ;

: DN* ( d1 n1 --- d2)    \ d2 = d1*n1
 OVER >R >R DABS R@ ABS UDU* R> R> XOR D+- ;
```

```
SCR #3
\ GREGORIAN-BREAK CALENDAR= Calendar constants         LFZ06JUL86
0 VARIABLE GREGORIAN-BREAK 6 ALLOT
: CALENDAR= CREATE , , , ,
              DOES> DUP 2@ GREGORIAN-BREAK 4 + 2!
                     4 + 2@ GREGORIAN-BREAK     2! ;

  15821015.   2299161. CALENDAR= CATHOLIC-CALENDAR
 -48000000.         0. CALENDAR= GREGORIAN-CALENDAR
 327671231.  13689325. CALENDAR= JULIAN-CALENDAR
  17520903.   2361222. CALENDAR= BRITISH-CALENDAR
\                      CALENDAR= ALASKAN-CALENDAR
\                      CALENDAR= SOVIET-CALENDAR
\                      CALENDAR= TURKISH-CALENDAR
```

```
SCR #4
  0 \ ?GREGORIAN HMS->.DAY .DAY->HMS                          LFZ06JUL86
  1
  2 : ?GREGORIAN ( year month day --- f)   \ Returns TRUE if the date
  3                                        \ on the stack is in the
                                             Gregorian calendar.
  4  SWAP 100 * + 0 ROT 10000 M* D+ GREGORIAN-BREAK 2@ D< NOT ;
  5
  6 : HMS->.DAY ( hours minutes seconds --- binary-fraction-day)
  7  >R SWAP 60 * + 60 U* R> 0 D+ 512 675 M*/ DROP ;
  8
  9 : .DAY->HMS ( binary-fraction-day --- hours minutes seconds)
 10  0 675 512 M*/ 3600 M/ SWAP 60 /MOD SWAP ;
 11
 12
 13
 14
 15

SCR #5
  0 \ TEN.POWERS JD.D->JD.B  JD.B.                            LFZ06JUL86
  1 CREATE TEN.POWERS 1 , 10 , 100 , 1000 , 10000 ,
  2
  3 : JD.D->JD.B ( Julian decimal day --- Julian binary day)
  4 \ The decimal point is given by DPL; IF DPL = -1 the decimal
  5  \ day is single precision. Binary day is triple precision with
  6  \ 16 bits to the right of the binary point.
  7
  8 DPL @ DUP 4 > ABORT" Too many decimal places for JD.D->JD.B"
  9 DUP 0< IF DROP 0 1 ELSE 2* TEN.POWERS + @ THEN
 10 DUP >R M/MOD ROT 0 SWAP R> UD/U DROP ROT ROT ;
 11
 12 : JD.B. ( binary day --- )   \ Print Julian binary day in decimal.
 13  D. 8 EMIT 46 EMIT 10000 U* SWAP 0< IF 1+ THEN 0
 14  <# # # # # #> TYPE SPACE ;
 15

SCR #6
  0 \ D.D->D.B D.B.                                           LFZ06JUL86
  1
  2 : D.D->D.B ( decimal day --- binary day) \ The decimal point is
  3                \ given by DPL; if DPL = -1 the decimal day is
  4                \ single  precision. Binary day is double  precision
  5                \ with 16 bits to  the right of the binary point.
  6
  7  JD.D->JD.B DROP ;
  8
  9 : D.B. ( binary day ---)   \ Print binary day in decimal.
 10  0 JD.B. ;
 11
 12
 13
 14
 15
```

```
SCR #7
\ YMD.B->JD.B YMD->JD                                    LFZ06JUL86
: YMD.B->JD.B ( n1 n2 d1 --- n3 d2 ) \ Convert calendar date of
          \ year n1 month n2 day and binary fraction of day d1 to
          \ integer Julian day d2 and binary fraction n3.
 SWAP >R 3DUP ?GREGORIAN SWAP R> SWAP 32768. D+ SWAP >R >R >R
 DUP 3 < IF 12 + SWAP 1 - SWAP THEN R>
 IF OVER 100 / 2 OVER - SWAP 4 / + ELSE 0 THEN
 SWAP 1+ 306 10 */ + R> + 0 ROT 1461 M* DUP 0<
 IF 3. D- THEN
 4 D/N D+ 1720994. D+ R> ROT ROT ;

: YMD->JD ( n1 n2 n3 --- d1)   \ Convert year month day to
                               \ Julian day number
 32768 SWAP YMD.B->JD.B ROT DROP ;


SCR #8
\ JD.B->YMD.B                                            LFZ06JUL86
: JD.B->YMD.B ( n1 d1 --- n2 n3 d2) m Convert Julian day number
      \ of integer d1, binary fraction n1 to calendar date n2 year
      \ n3 month and d2 day and fraction of day.
      \ Adapted from Jean Meeus "Astronomical formulae for
      \ Calculators" p. 26.
 ROT 0 32768. D+ SWAP >R 0 D+ 2DUP GREGORIAN-BREAK 4 + 2@
 D< IF ELSE 2DUP 4 DN* 7468865. D- 16233 M/ SWAP DROP
             9 / DUP 4 / - 1+ S->D D+ THEN
 1524. D+ 2DUP 10 DN* 1221. D- 2 DN*
 7305 M/ SWAP DROP DUP 1461 U* 4 D/N >R >R ROT ROT R> R>
 D- DROP DUP >R 10 306 */ DUP >R DUP 14 <
 IF 1 ELSE 13 THEN -
 SWAP OVER 3 < IF 4715 ELSE 4716 THEN - SWAP R> R> SWAP
 306 10 */ - R> SWAP ;


SCR #9
\ JD->YMD YMD.D->JD.B JD.D->YMD.B                        LFZ06JUL86
: JD->YMD ( d1 --- y m d ) \ Convert Julian day number to
                           \ calendar date.
 0 ROT ROT JD.B->YMD.B SWAP DROP ;

: YMD.D->JD.B ( n1 n2 d1 --- n3 d2)  \ Convert calendar date of
  \ year n1 month n2 decimal fraction day d1 to integer Julian
  \ day d2 binary fraction of day n3.
 D.D->D.B YMD.B->JD.B ;

: JD.D->YMD.B ( d1 --- n1 n2 d2 )    \ Convert decimal fraction
  \ Julian day number to year n1 month n2 day and binary fraction
  \ day d2.
 JD.D->JD.B JD.B->YMD.B ;
```