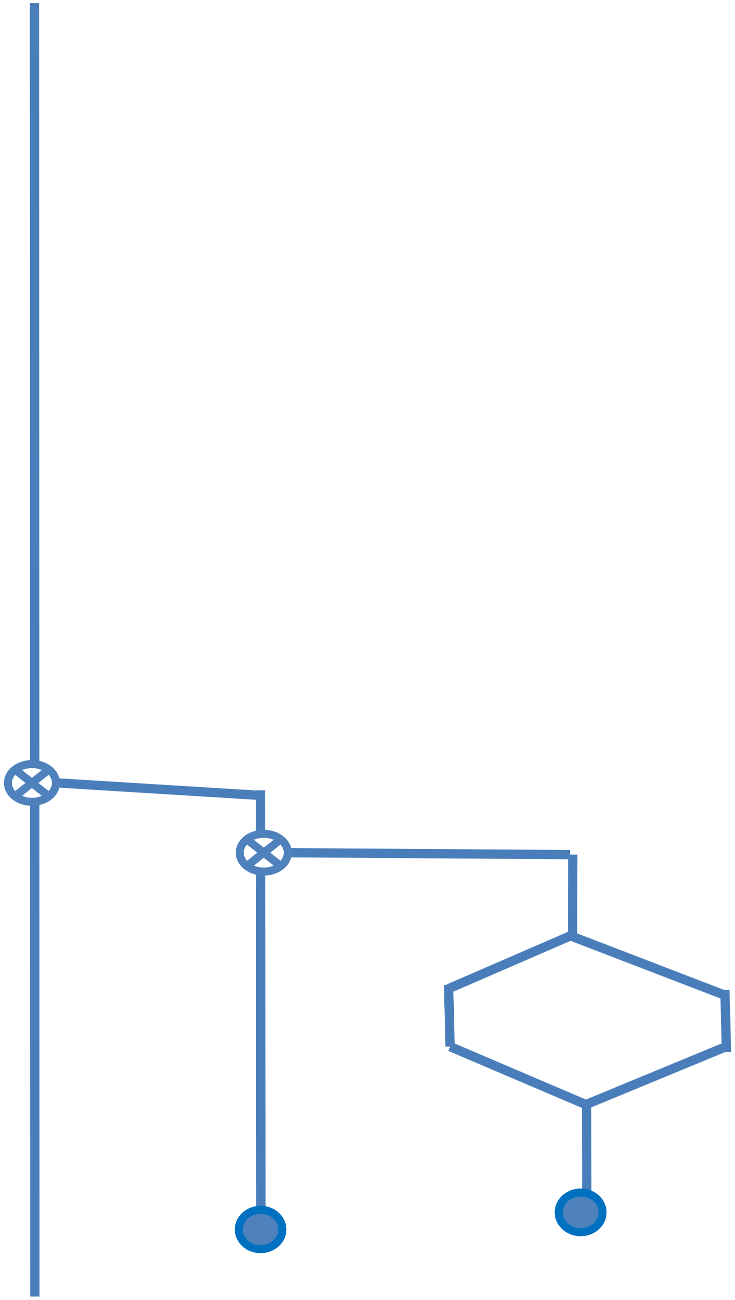


Calendar Tools Leading To Week Number Determination

SVFIG

Nov. 18, 2023

Bill Ragsdale

Disclaimer

We will cover a huge amount of material today.

Just follow the concepts, not the code.

Refer to the archive locations on the next slide.

Did You Know?

The SVFIG slides, handouts and videos for 24 YEARS are archived at:

forth.org

- SVFIG – Silicon Valley FIG
  - Past Meeting slides, video and notes.
  - Meeting videos (YouTube).

https://github.com/BillRagsdale/

THIS MATERIAL IS GOLDEN

Challenge

- The ISO 8601 Standard week is used for planning business finance and operations.

- The ISO weeks, beginning on Monday, are numbered 1 to 52 or 53 with Week One containing the first Thursday of the year.

- Challenge: Program the ISO week for any date. Consider using Zeller's rule. Check with the Excel ISOWEEKNUM.

Waypoints To The Challenge

We’ll review of tools for calendar support.

Leading to the calculation of ISO Week Numbers.

A key is Zeller’s Rule for finding the day of any date. It is ideal for Forth as it uses integer arithmetic.

It is tricky.

Input A Date

```
: Accept ( --- ) \ load day, month, year
   ." Input the day number "   get-day    to day
cr ." Input the month number " get-month  to month
cr ." Input the year in four digits "
                               get-year   to year ;
```

get-day, get-month and get-year do range checking.

Interactive Input

```
Accept
Input the day number 26
Input the month number  10
Input the year in four digits 2023     ok

Report
Day  26 Month 10 Year 2023
and see:  Thursday ok
```

Leap Years

To determine the day position in a year for a date, you must allow for the extra day in leap years.

If the year is evenly divisible by 400 is it a leap year. If the year is evenly divisible by 4 and not 100 it is a leap year.

2000 YES,   2001 NO,   1900 NO,

Leap Years

```
: ?LeapYear  ( year  -- flag )
 \ True for a leap year.
   dup                  \ year year
   400 mod 0=           \ year flag
   over 100 mod 0<>     \ year flag flag
   rot 4 mod 0=         \ flag flag flag
   and or  ;
2000 ?LeapYear .   and see:  -1 ok
2001 ?LeapYear .   and see:   0 ok
```

Leap Days

```
CREATE DaysPerMonth
\ byte array for normal and leap years.
31 c, 28 c, 31 c, 30 c, 31 c, 30 c,  \ normal
31 c, 31 c, 30 c, 31 c, 30 c, 31 c,  \ normal

31 c, 29 c, 31 c, 30 c, 31 c, 30 c,  \ leap year
31 c, 31 c, 30 c, 31 c, 30 c, 31 c,  \ leap year
\ 1    2     3     4     5     6
\ 7    8     9    10    11    12
```

Leap Days

Select the days per month array depending on leap year.

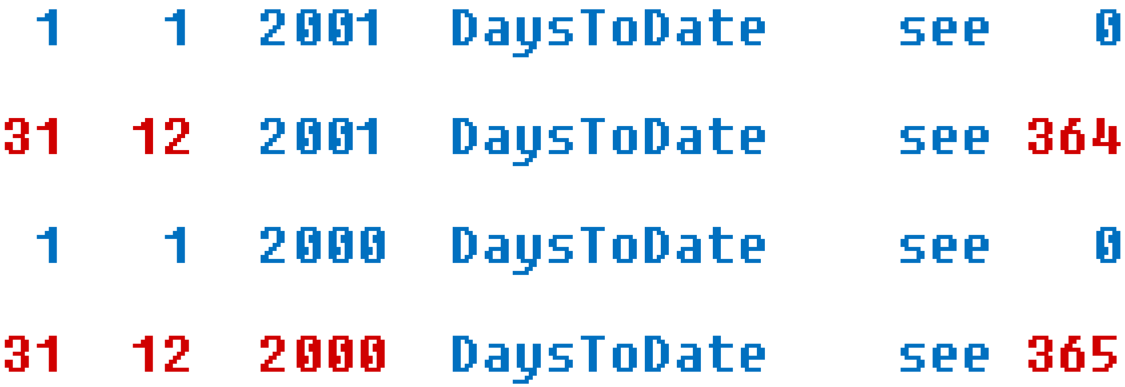
```
: DayArray  ( year - adjustedaddress )
   DaysPerMonth swap ?LeapYear if 12 + then ;


2001  DayArray  .   see: 4495656 ok
2000  DayArray   .  see: 4495668 ok
```

Days To A Date

```
( Return the days from Jan. 1 in a given year )
\ Jan 1 = 0,  Dec. 31 = 364
: DaysToDate  ( day month year -- days )
  0  -rot  DayArray
  swap 1-  over + swap
  ?do i c@ + loop
  + 1- ;
```

Days To A Date, cont.

Note: January 1 is day zero
Dec. 31 is day 364 in a non-leap year.



```
 1    1   2001   DaysToDate    see   0
31   12   2001   DaysToDate    see 364
 1    1   2000   DaysToDate    see   0
31   12   2000   DaysToDate    see 365
```

Introduction

The Zeller rule uses a calculation year beginning on the first day AFTER a leap day, Feb. 29.

It continues for four years ending on the next Feb. 29, in a leap year.

This can cause a bit of confusion calculating the month and year offsets.

At least it did for me.

The Zeller Year

The Zeller year begins on the first day AFTER a leap-day, Feb. 29 and runs for 1461 days, ending on a leap-day.

3/1/2000 --- 2/28/2001

3/1/2001 --- 2/28/2002

3/1/2002 --- 2/28/2003

3/1/2003 --- 2/29/2004

Adjustments

The adjusted month numbers runs from March as month 1 to December as 10 and then the following January as 11 and February as 12.

The year adjustment means for January and February you use the prior year, as these months conclude the prior Zeller year.

Adjustments, Year 2004

```
                 Jan. Feb. Mar. Apr. May  Jun.

Month Number      1    2    3    4    5    6

Adjusted Month   11   12    1    2    3    4

Adjusted Year   2003 2003 2004 2004 2004 2004



                Jul. Aug. Sep. Oct. Nov. Dec.

Month Number     7    8    9   10   11   12

Adjusted Month   5    6    7    8    9   10

Adjusted Year  2004 2004 2004 2004 2004 2004
```

Adjustments

```
: adjustedYear ( month year -- zY )
  \ allowing for month 1 and 2
   over 1 = rot 2 = or if 1- then ;

: adjustedMonth ( month -- zM )
 \ months run 11 12 1 2 . . . 10
   10 + dup 12 > if 12 - then ;
```

The Zeller Rule

Sum the following ignoring decimal fractions:

```
                        day  +
(26 * adjustedmonth - 2)/10  +
  mod(adjustedyear,100)      +
  mod(adjustedyear,100)/4    +
      adjustedyear/400       +
    2*adjustedyear/100       -
```

And take modulo(7) of the sum.

Individual Factors

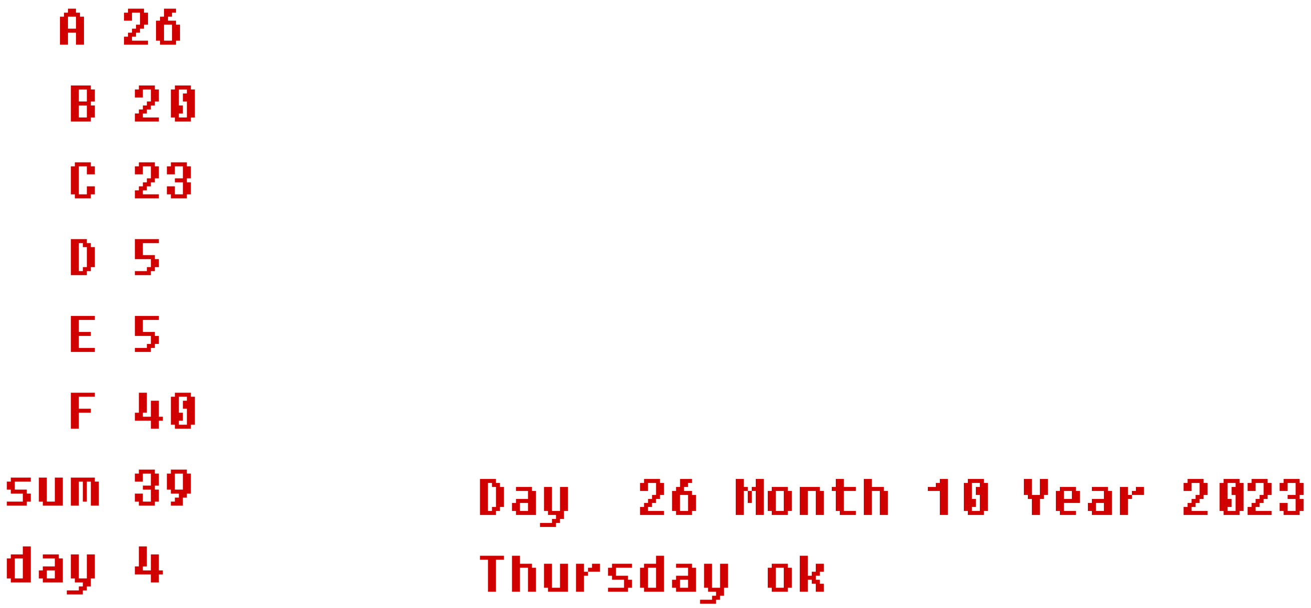
```
: factorA  ( -- A  )  day ;  \ d in formula

: factorB  ( -- B )          \ calculate from m
   adjustedmonth 26 *  2 -  10 / ;

: factorC ( -- C )      \ last two digits of the year
     adjustedyear 100 mod ;

: factorD ( -- D )           \ four year cycle y/4
    factorC 4 / ;

: factorE ( -- E )           \ the century / 4
   adjustedYear 100  /  4 / ;

: factorF ( -- F )           \ century c * 2
   adjustedYear 100 / 2 * ;
```

The Final Summation

```
: DayOfDate ( -- day )
      factorA
      factorB +
      factorC +
      factorD +
      factorE +
      factorF -
      7 mod  ;
```

```
0 = Sunday through 6 = Saturday
```

My Diagnostic Printout



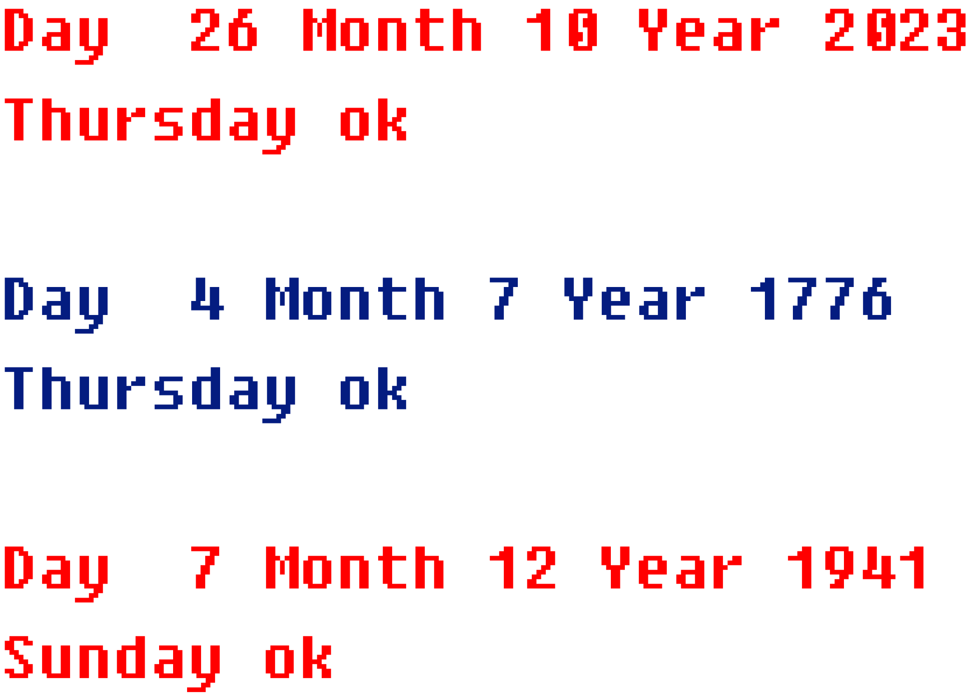
```
  A 26
   B 20
   C 23
   D 5
   E 5
   F 40
sum 39          Day  26 Month 10 Year 2023
day 4           Thursday ok
```

Modulo of negative numbers is tricky.
Most Forths get it right.

Simplified Day of Date

```
: DayOfDate ( -- day )
    over swap adjustYear >r
    adjustMonth 26 * 2 - 10 /
    r@ 100 mod dup
    4 /
    r> 100 /  dup 4 /
    swap  2 *
    - + + + +  7 mod ;

0 = Sunday through 6 = Saturday
```

Day Of Date Tests



```
Day  26 Month 10 Year 2023
Thursday ok

Day  4 Month 7 Year 1776
Thursday ok

Day  7 Month 12 Year 1941
Sunday ok
```

Week Number Overview

- Determine the number of days between the first Thursday of the year and the Thursday in the week of your target day. Mon…Sun
- Divide by 7 and add 1.
- Adjust for the first week and last week of some years. Tricky.

Pseudocode

```
WeekNumber   ( day month year -- n )

Find the day number 0..6 of target day
Find days in year to the target day
Calculate offset in week from target day to Thursday
Apply this offset to the day in year position
Find days in year to first Thursday
Subtract, divide by 7,  add one.
Have the ‘raw’ week number of year.
Apply the first and last week adjustment.
```

Target Thursday

```
<snip>
    3dup DayOfDate >r  \ get day value 0..6
    3dup DaysToDate    \ location of day in year
    r>
    6 + 7 mod 3 -      \ offset to Thursday
    negate  +          \ days to Thursday
<smip>
```

Day of First Thursday

Search for a Thursday over day 1 to day 7 of year.

```
: FirstThursday ( year -- Thursday’s# )
   8 1  \ over the first seven days of the year
   DO i over 1 swap
       DayOfDate           \  year day0.6
       4 = if drop i leave then
     LOOP  ;
```

Final Week Number

```
: WeekNumber  ( day month year -- n )
    dup>r
    3dup DayOfDate >r        \ day value 0..6
    3dup DaysToDate          \ location of day in year
    r>
    6 + 7 mod 3 -            \ adjust to Thursday
    negate  +
    r> FirstThursday  1-     \ locate 1st Thursday
    -  7 /  1+
    AdjustWeekNumber ;       \ 1st and last week.
```

How About First and Last Week?

52, 53 or 1

```
January 2010
  Week        M   T   W   Th  F    Sa   Su
   53                         1    2    3
    1         4   5   6   7   8    9   10
    2        11  12  13  14  15   16   17
```

```
December 2014
 Week         M   T   W   Th  F    Sa   Su
   51        15  16  17  18  19   20   21
   52        22  23  24  25  26   27   28
    1        29  30  31
```

52, 53 or 1

Last Week Adjustment

- The ‘raw’ last week of the year may compute to ‘53’ when it is a partial week of the next year.
- If the target day of week < 4 , i.e. before Thursday AND
- Raw week number is 53 THEN force week number to ‘1’.

First Week Adjustment

- The ‘raw’ first week will compute to ‘zero’ if it is a partial week of the prior year.
- It is a week 52 or week 53?
- In the PRIOR year: if the last day is Thursday OR last day is Friday AND a leap year, force 53 ELSE 52.

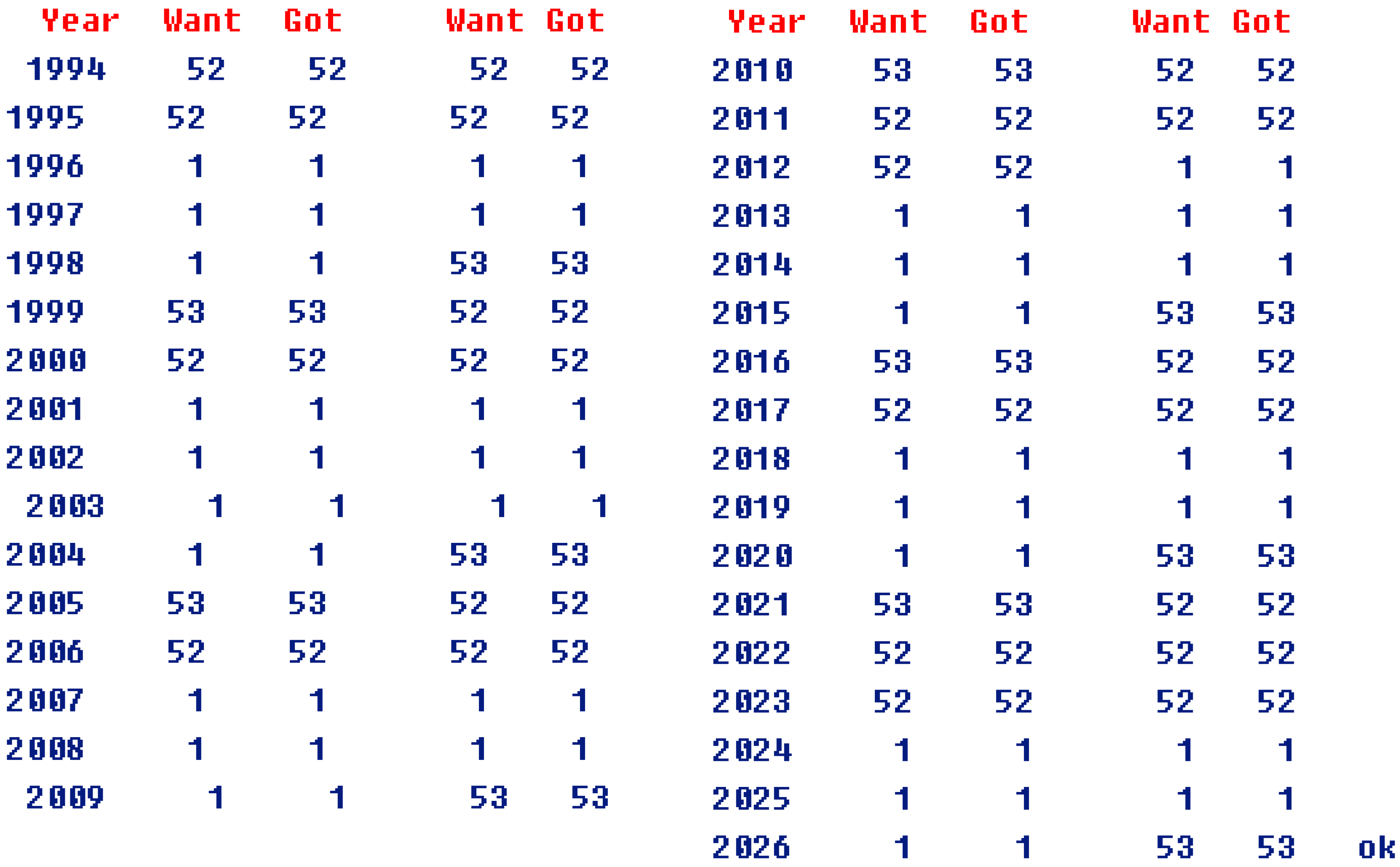
Adjusting Week Number

```
: AdjustWeekNumber (  d m y n1 -- n2 )
  dup>r

\ Test for a week ‘0’
  0= if  dup 1-  ThurFriTest
             if 3drop r>drop 53 exit
                 else 3drop r>drop 52 exit then
          then

\ Test for a week ’53’
   DayOfDate 4 < r@ 53 = and
             if r>drop 1 exit   then

\ no adjustment, recover original week number
   r>  ;
```

Final: Week Number of Year

```
: WeekNumber  ( day month year -- n )
    dup>r   3dup DayOfDate >r
    3dup DaysToDate  r>  6 + 7 mod 3 -
    negate  +  r> FirstThursday  1-  -
    7 /  1+
    AdjustWeekNumber ;
```

Final Tests



```
  Year   Want   Got     Want  Got     Year   Want   Got     Want  Got
  1994    52     52      52    52     2010    53     53      52    52
 1995    52     52      52    52      2011    52     52      52    52
 1996     1      1       1     1      2012    52     52       1     1
 1997     1      1       1     1      2013     1      1       1     1
 1998     1      1      53    53      2014     1      1       1     1
 1999    53     53      52    52      2015     1      1      53    53
 2000    52     52      52    52      2016    53     53      52    52
 2001     1      1       1     1      2017    52     52      52    52
 2002     1      1       1     1      2018     1      1       1     1
  2003     1      1       1     1     2019     1      1       1     1
 2004     1      1      53    53      2020     1      1      53    53
 2005    53     53      52    52      2021    53     53      52    52
 2006    52     52      52    52      2022    52     52      52    52
 2007     1      1       1     1      2023    52     52      52    52
 2008     1      1       1     1      2024     1      1       1     1
  2009     1      1      53    53     2025     1      1       1     1
                                      2026     1      1      53    53    ok
```

Conclusions

- Twenty years ago I attempted this in Excel and then Visual Basic not knowing the ‘tricks’.

- Most smart phone calendars can set Monday as first day of the week and show week numbers.

- On calendars with Sunday first, Sunday is actually part of the prior numbered week.

References

https://beginnersbook.com/2013/04/calculating-day-given-date/

https://forth.org

SVFIG – Silicon Valley FIG

Past Meeting slides, video and notes.

Meeting videos (YouTube).

https://github.com/BillRagsdale/