# Do Centuries Really Change?

Winfried Gerum

*by Winfried Gerum*

You rarely see a deadly car accident, but each day you see happy motorists populating the streets by the thousands. So most people act as if driving is perfectly safe. Statistics tell you a different story.

In a similar way, many think that centuries never change. The two digit shorthand to denote a year is by orders of magnitude more frequently used than the full (four digit) version. Quite naturally, virtually all people alive now were either born during the twentieth century or they were infants at the time of the New Century celebrations January 1, 1900. Virtually all business transactions stored in computers carry dates out of this century. Except for the handful of historians, astronomers, life insurers, and mortgage bankers, few are aware of any significant date that is not within the twentieth century. So, generations of programmers wrote applications, tools, and operating systems assuming a two-digit year, thereby doing the logical equivalent of laying landmines.

With each morning we come closer to $H=458074,0 (i.e., 2000/01/01,00:00:00). This is a deadline that no amount of persuasion or armtwisting can postpone. Hey, why so much fuss over two digits? Well, some argue that this deadline might be really deadly for quite a few software applications. You better think about the implications right now. It is much more than simply having two digits more or less. A lot of algorithms are affected, sometimes in subtle, non-obvious ways.

A VERY SPECIAL NEW YEAR'S DAY

December 1999

| CW | 48 | 49 | 50 | 51 | 52 |
|---|---|---|---|---|---|
| SU | | 5 | 12 | 19 | 26 |
| MO | | 6 | 13 | 20 | 27 |
| TU | | 7 | 14 | 21 | 28 |
| WE | 1 | 8 | 15 | 22 | 29 |
| TH | 2 | 9 | 16 | 23 | 30 |
| FR | 3 | 10 | 17 | 24 | 31 |
| SA | 4 | 11 | 18 | 25 | |

January 2000

| CW | 52 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| SU | | 2 | 9 | 16 | 23 | 30 |
| MO | | 3 | 10 | 17 | 24 | 31 |
| TU | | 4 | 11 | 18 | 25 | |
| WE | | 5 | 12 | 19 | 26 | |
| TH | | 6 | 13 | 20 | 27 | |
| FR | | 7 | 14 | 21 | 28 | |
| SA | 1 | 8 | 15 | 22 | 29 | |

February 2000

| CW | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|
| SU | | 6 | 13 | 20 | 27 |
| MO | | 7 | 14 | 21 | 28 |
| TU | 1 | 8 | 15 | 22 | 29 |
| WE | 2 | 9 | 16 | 23 | |
| TH | 3 | 10 | 17 | 24 | |
| FR | 4 | 11 | 18 | 25 | |
| SA | 5 | 12 | 19 | 26 | |

58074 $H
January 1, 2000 AD
Ramadan,24 1420 AH
Tevet, 23 5760 AM

As long as a date is only stored and retrieved, it does not really matter which format is used. When it comes to sorting, comparisons, or arithmetic, an incomplete year specification may give you unexpected results.

## Just a Few Examples

A frequently used format for dates is YYMMDD. This is very convenient, for now. With a popular trick it can be converted to the usual American, British, or German format:

```
WRITE $TR("Mm/Dd/Yy"
,"YyMmDd",YYMMDD) ;
American

WRITE $TR("Dd/Mm/Yy",
"YyMmDd",YYMMDD) ;
British

WRITE $TR("Dd.Mm.Yy"
,"YyMmDd",YYMMDD) ;
German
```

It always produced the proper collation (i.e., early dates sort before late ones). If you try this with values beyond 991231, surprisingly in M, you still get the expected collation. How is this? Numbers with leading zeroes are not canonic numbers. They sort alphabetically (i.e., according to the ASCII value of their characters). But don't count on M solving everything miraculously. First, this collation mechanism would postpone the conversion deadline by only another ten years. Second, the comparison operators will place 0YMMDD before 991231. Note that replacing YYMMDD by YYYYMMDD is not the perfect solution. That would merely replace the year 2000 problem with a year 10000 problem. I am not kidding! Sure, the year 10000 seems ridiculously far into the future. But remember, humans have populated this planet for at least 500,000 years now. So, we had better look for solutions that will not produce headaches for anyone.

This is not easy. Consider how to convert the four-digit year into a two-digit year:

```
WRITE YEAR#100
```

did the trick as long as you can remember. Try again with YEAR=2003. Surprise, surprise, this gives you just the last digit!

If you think about using:

```
WRITE $EXTRACT(YEAR,3,4)
```

to do the job, please remember that having four digits is not an intrinsic property of a number representing a year. The politically correct way looks something like:

```
WRITE $E(YEAR, $L(YEAR)1,
$L(YEAR))

WRITE $RE($E($RE(YEAR),1,2))
```

The M language features the system date as a special variable $HOROLOG . This is the count of days since 31-DEC-1840. Such a continuous day count is ideally suited as a basis to remedy all date format problems. For the benefit of those who missed the algorithms for how to convert the common date to this $HOROLOG format and vice versa, we print the algorithm below:

Your TO-DO-LIST for the year 2000 problem (Y2K):

- inventory all production software
- find all places where the year is represented with two digits
- change the year representation to four digits
- change all algorithms that use the year field

```
       ;converts ISO-style date to $H-Format date
DATE2H(DATE) QUIT $$YEAR2H(DATE)
-->   +$PIECE($$MTAB(DATE),",",$PIECE(DATE,"/",2)-1)
-->   +$PIECE(DATE,"/",3)-1

       ;converts $H-Format date to ISO-style date
H2DATE(H) NEW D,L,M,Y
      SET Y=$$H2YEAR(H),D=H-$$YEAR2H(Y),L=$$MTAB(Y)
      FOR M=1:1 QUIT:$PIECE(L,",",M)>D
      SET D=D-$PIECE(L,",",M-1)+1
      QUIT Y_"/"_$EXTRACT(0,M<10)_M_"/"_$EXTRACT(0,D<10)_D

       ;converts YEAR $H-Value of its first day
YEAR2H(Y) QUIT:Y>0 Y*365+(Y-1/4)-(Y-1/100)+(Y-1/400)-672410
       SET Y=Y-1 ;NO YEAR ZERO
       N E S E=Y/400-1
       QUIT E*146097+$$YEAR(-E*400+Y)

       ;returns YEAR of $H-Format date
H2YEAR(H) NEW D,Y
       SET D=H+672410
       SET Y=D<0+D\146097-(D<0)*400,D=D#146097
       SET Y=D\36524*100+Y,D=D#36524
       SET Y=D\1461*4+Y,D=D#1461
       SET Y=D\365+Y
       SET:Y<1 Y=Y-1 ;NO YEAR ZERO
       QUIT Y

       ;returns cumulative month lengths for a given YEAR
MTAB(YEAR) QUIT:$$ISLEAP(YEAR)
-->   "31,60,91,121,152,182,213,244,274,305,335,366"
      QUIT "31,59,90,120,151,181,212,243,273,304,334,365"

      ;returns 1 if Y is a leap year, 0 otherwise
ISLEAP(YEAR) N Z S Z=Y
       SET:Z<1 Z=Z-1 ;NO YEAR ZERO
       QUIT:Z#4 0 QUIT:Z#100 1 QUIT Z#400=0

       ;converts $H-Format date to CENTURY_DIGITS
H2CENT(H) QUIT H+672411\36524.25
```

- fix all production globals

- fix all archived globals

- inventory all commercial software packages (CSP)

- ensure that all CSP correctly interface with the revised data structures

- inventory all operating systems (OS)

- ensure that all OS are Y2K compliant

It was four years ago that I became aware of the problem. Our software had all the textbook problems: YY (two digits) for year, YYQ to denote a quarter year, DD-MM-YY to denote a date, $H-Format date only in very few places. The operating system did not even allow us to enter a date beyond 31-DEC-99.

It took me that four years to fix all related problems in our software. I had many a fight to get "these unnecessary" changes through. Again and again I was told to care about this year's bottom line, not about some fancy future stuff.

The actions were not to have any noticeable effect on the end user. The changes were done within the context of routine software maintenance to avoid the need for significant additional quality assurance capacity. Now all of our YEAR fields are in full spelling (four digits). All dates are internally represented in $H-Format. Quarters are denoted by the $H-value of their first day.

All date entries are handled by a single input routine. A parameter tells this routine whether a "past" or "future" date is to be expected. Incomplete date entries are permitted. Missing parts are automatically inserted to produce a valid and complete date in the specified domain. There are a few instances when a requested date can be either past or future. In this case any missing parts are taken from the current date. The return value of this date entry routine is either an action indicator (ABORT, BACK TO PREVIOUS ENTRY...), an empty string (where permitted) or (on entry of a date) the $H-format value of the date.

This architecture works wonders. For anything but display purposes, the $H-Format is superior to any other date format. Comparisons between two dates are trivial, chronological collation is automatic. The change made all affected programs more concise and arguably more readable. We did not make any performance comparisons. But no slowdown in dialogues or lengthy printouts has been reported in this context by our users. Listings of globals are not as straightforward or as easy to read as they used to be.

A side effect of the universal use of $H as an internal date format is that it is completely culture independent. Adaptations to any variations of the common calendar are as easy as adaptations to completely different calendars (Jewish, Arabic, . . .).

We are confident that the Y2K problem has been solved for the majority of our software. We are aware of the fact that the operating system currently in use with the bulk of our users does not even allow us to set the system date to a value beyond 31-DEC-99. The new release of the operating system has fixed that. But we expect that in three years many sites will still be using the old release. So, we implemented our own procedure to set the system date.

Last year (1995) medical insurance companies in Germany introduced magnetic cards as proof of coverage. Despite the imminent end of the century, "valid through" field features a date in the format MMYY. Technology would allow us to store a megabyte of data on these cards. But politics allows us just about 200 bytes. It is absolutely ridiculous that they did not supply the additional two digits. To make things worse, it is required that medical systems store the contents of these cards unmodified. Of course, it can be handled. IF YY > 94, the century digits are 19, otherwise use 20. This example illuminates the problem: Not only do legacy systems ignore the Y2K problem, but so do brand new ones. While it is easy to work around the problem in each case, there is not a universal solution. And there is a multitude of these sins. The ugly thing is, that they lay dormant for some time (an additional 1200 days). But they will surface all at once!

M has been helpful in making the change in several ways. First, the existence of $H made us aware of the problem very early and indicated a path to solve this problem. Second, the variable length date fields made the change of date formats easy. Furthermore, the ultra-efficient M globals made it possible to avoid any archiving in our software so far. So we are not confronted with the challenge of restructuring archived data.

If your organization and your software are not yet Y2K compliant, make sure the change is started soon. If you plan to get a new job in an other organization, make sure their computers speak M. Should you be retired by $H=58074 congratulations ... unless some crazy software stops pension payments to people with future birth dates like yours! M

---

Winfried Gerum is with Winner Software GmbH, in Roettenbach, Germany. Contact him by phone at 49-9195-940022 or by fax at 49-9195-940030.

---

MTA Conference '97

Week of May 18, 1997

Hynes Convention Center, Boston

In conjunction with Database and Client/Server World

See cover 2 and cover 3 for more information.

---

**Which database lets you scale up to thousands of users without changing a single line of code?**

FOR YOUR INFORMATION, IT'S OPEN M.™

OPEN M
InterSystems

The high-performance database for client server

For more information, call 617-621-0600, Department CW.
Or check our web site at **www.intersys.com/cw**

Open M is a trademark of InterSystems Corporation. Copyright © 1996 InterSystems Corporation.