The FileMan Date Format

by George Timson

Introduction

In the last several columns, we looked at Input Transforms. Every VA FileMan field has an Input Transform associated with it, in the form of M[UMPS] code. Virtually every *date-valued* field's Input Transform contains a call to the FileMan routine "%DT," which is supposed to turn input into a canonic "FileMan Date." In this column we will take an in-depth look at the FileMan date format and its shortcomings. Related topics mentioned will include SQL, Data-Dictionary listings, and Shakespeare's birthday (time to plan your celebration!).

Y2K—is there a problem?

Even characters in the funny papers know by now about the portentous "Y2K" problem. Computers were programmed in the 1960's and early 1970's so that dates were only stored with 2-digit years. As we come to a new century, some of this programming is, amazingly, still in use! But we in M have always had dates handed to us by $HOROLOG, which represents anything past 1840 unambiguously. Furthermore, since M was often used for medical records, which include birthdates, M developers were accustomed to considering dates that represented pre-1900 events.

When FileMan was designed twenty years ago, a seven-digit date format was fairly uncontroversial. EDP systems had increased byte capacity by then, compared to the 1960's. The "YYYMMDD.HHMMSS" format was easy to display in human-readable form without a lot of number-crunching. It could represent dates alone, as well as *times* within dates, in a way that would collate better than $HOROLOG, (i.e., as canonic M numerics). And, it was capable of representing imprecise dates (JAN 1999=2990100, 1999=2990000) distinctively, as needed in medical practice.

So . . . since "today's date" in FileMan will simply roll over from 2991231 to 3000101 on the stroke of that dreaded midnight two years hence, all should be well. But of course there is never any guarantee that programmers, even FileMan programmers, always use ^%DT and %DTC (which subtracts dates from dates) for all their date-related work. Ed de Moel's *Windmills* column from the last issue (*M Computing*, Volume 5, Number 5) has already given examples of the millennial problems that can arise when M routines take a well-formatted date and truncate it for their own evil purposes. Every programmer must search his own heart and his own code to see whether Ed's warnings apply to him.

Let's consider carefully, from the perspective of the FileMan %DT utility, a different issue: how to help the user who enters dates at the terminal. When %DT takes a date input like "JAN 1," it generally *assumes the current year*. Similarly, when it gets a date like "JAN 1 99" it assumes the current *century*. But this will many times be the wrong assumption during the next few years. We are already hearing from users who complain that "the computer should be smart enough" to recognize that "JAN 1 01" now means 1 January 2001, not 1901. And after 2000 arrives, we will similarly hear them complain that "JAN 1 99" should be understood to refer to 1999, not 2099. Well, of course the customer is always right, and the %DT utility offers us two different ways to help "make the computer smart enough."

First, we can make sure that dates are entered only within appropriate ranges. We don't even have to have "programmer access" to do this. Just go back through MODIFY FILE ATTRIBUTES and make sure "EARLIEST DATE" and "LATEST DATE" are filled

in for all your DATE/TIME fields. Note that the letters "DT" constitute a legal input to either of these questions; they mean "Date Today"—the date on which the data entry will be done. (This DT input is not mentioned in the Help-prompt to these questions in Version 21).

Second, with just a smidgen of M, we can tell the %DT utility to assume the past, or assume the future in the case of input ambiguity, like "JAN 1 01." We do this by inserting "P" or "F" into the input parameter %DT that must be set before DOing ^%DT. Thus, in a typical Input Transform for a FileMan DATE/TIME field, we will see

```
S %DT="EX" D ^%DT...
```

and we will use the FileMan Utility Option called INPUT TRANSFORM(SYNTAX CHECK) to edit that M code to look like either:

```
S %DT="EXP" D ^%DT...
```

if we expect the date to be *prior to the date of entry*, like 1999 after 2000 or

```
S %DT="EXF" D ^%DT...
```

if we expect it to be *after the date of entry*, like 2001 now.

By the way, %DT's help-message is context-sensitive to those P's and F's, and will tell the user what to expect, if he types "?" (e.g., "If the year is omitted, the computer assumes a date in the FUTURE"). Really, these days, such a message also implies: "If the century is omitted, the computer also assumes a date in the FUTURE."

A "CUSTOM" Data Dictionary Listing

But now how will we easily be able to review all the DATE-VALUED fields that need updating in the two ways just suggested? Pity the poor COBOL programmer trying to tackle the Y2K problem without knowing systematically where every outmoded "YYMMDD" date record might be stored! We who use a DBMS are working with data that is well-structured enough to be able to pull out all the descriptions of our Date fields for scrutiny, and *only* those. But how, exactly?

Not every FileMan user is familiar enough with the data-dictionary format that FileMan calls "CUSTOM-TAILORED." This format, among other uses, can produce a "DD listing" that includes *only fields of a certain type*. Below is the FileMan dialogue that would produce a report showing the DATE-type fields that occur in File 2, listed in field-number order, and displaying the field number, name, and the field's Input Transform.

Since Input Transforms can be long lines of M, we "wrap" each Input Transform value within a page-width. Try this one at home.

When is midnight?

Now let's consider some other date-format issues. Jule Meyn (omeyn@cpcug.org) has described another problem with VA FileMan's dates. The problem deals with how "midnight" is expressed in FileMan format. Take that infamous moment we've been anticipating: Since "3000101," as an integer, refers just to the date "1 January 2000," we need a different convention for representing "1 January 2000 at exactly midnight." How do we express that date/time as a canonic number? FileMan answers this issue by defining *one second after midnight* as the beginning of the day and *86400 seconds after midnight* as the end of the day, so all time values are non-zero. The time (fractional component) is stored as .000001 (1 second) through .24 (24 hours = 86400 seconds). Thus "1 January 2000 at exactly midnight" will be "2991231.24." FileMan will take one second longer than you may have thought, to roll over to 3000101!

```
Select OPTION: DATA DICTIONARY UTILITIES
Select DATA DICTIONARY OPTION: LIST FILE ATTRIBUTES
 START WITH WHAT FILE: PATIENT
  GO TO WHAT FILE: PATIENT
  Select: SUB-FILE:
Select LISTING FORMAT: STANDARD// CUSTOM-TAILORED
SORT BY: LABEL// @TYPE
START WITH TYPE: FIRST// D
GO TO TYPE: LAST// E
  WITHIN TYPE, SORT BY:
FIRST PRINT ATTRIBUTE: .001   NUMBER
THEN PRINT ATTRIBUTE: .01   LABEL
THEN PRINT ATTRIBUTE: INPUT TRANSFORM;W77
  . . . .
```

To be sure, %DT and %DTC are consistent in this convention, but there are a couple of nagging concerns here. For one, consider the (rarely used) subroutine, `H^%DTC`, which converts an internal FileMan date to (supposedly) a $HOROLOG format. An input of "2991231.24" representing midnight, will be converted by

```
DO H^%DTC
```

to 86400 seconds (in the variable "%T"), consistent with the above convention. But the time portion of $HOROLOG's value is defined in the MUMPS Standard as being "an integer value modulo 86,400," and thus can't ever be 86400. So, to be perfectly careful, every call to `H^%DTC` should be followed by:

```
IF %T=86400
SET %H=%H+1,%T=0
```

Conversely, the subroutine, `YMD^%DTC`, also rarely used, is supposed to convert a $H date/time to FileMan format. But when its input is "exactly midnight," it does *not* return the previous date +.24. Picky, picky, Jule.

More seriously, there is a problem when SQL looks at "midnight" dates. Standard SQL database systems, according to Jule, define the day to begin (logically enough) at 12:00:00 A.M. ". . . This leads to apparent inconsistencies in reports of the same data for the same range of dates produced by SQL Servers versus those produced by FileMan. . . . This tiny difference means that midnight of 9/19/97 in a FileMan report corresponds exactly to midnight of 9/20/97 in most other database management systems, including KB_SQL. . . . Any FileMan report constrained by a date or a range of dates that contains transactions time-stamped at midnight of the final date in the range will include those transactions. But an SQL report with the same constraints will include transactions time-stamped at midnight of the first date and exclude those of the last date."

Is there an SQL work-around? The fix, says Jule, is that "the constraint 'WHERE TRANSACTION_DATE=TODAY' must be replaced by the constraint 'WHERE TRANSACTION_DATE BETWEEN TODAY@12:00:01AM AND TODAY+1@12:00:00AM'." *But*, this will work only if TRANSACTION_DATE is defined always to include time! Ugly!

Taking the Long View

Well, the good news, certainly, is that FileMan-based systems are well-equipped to handle dates into the 21st century, so long as programmers have used the available %DT and %DTC utilities. Looking further ahead, and backwards, though, we may just mention some other, less-urgent, issues with the date format.

Back in the early 1980's, some MUMPS-savvy folks at the Smithsonian Institution in Washington were seriously looking into using VA FileMan as a DBMS with which to catalogue the Institution's holdings. The way I heard it, they rejected FileMan because it couldn't handle dates before 1800 properly!

Why has %DT never allowed input of dates before 1800—"1000000"? The first difficulty, of course, is with dates in the 18th century. We can easily see that Mozart's birthday (27 January 1756) should be represented by FileMan as 560127 but this means that all the existing M code, including FileMan code, that takes `$E(DATE,1,3)+1700` to find the displayable year of the variable "DATE," will fail to display this date correctly. Alternatively, if FileMan were to represent Mozart's birthday as 0560127 it would violate the rule that its dates are always canonic numbers. Such a number, indeed, would collate in M *after* all post-1800 dates!

Dates before 1700 would obviously present an even bigger difficulty, because for these, only a negative value can be appropriate. But what value? We can imagine representing Shakespeare's birthday (23 April 1564) as -1640423 . . . but perhaps that should be -1360423 since 1564 is 136 years before 1700?

And when we go back to archeological and paleontological time, things get ridiculous. Should the Cretaceous-Tertiary event be represented as occurring on the date -650000000000?

As far as the future is concerned, it's obvious that when we start handling dates past 2700 AD, we again have a problem with $EXTRACTing the characters of the formatted date. 1 January 2700 will have to be represented as 10000101—with eight digits in the integer part. As the centuries roll by, we will refer to this oncoming catastrophe as VA FileMan's "Y2.7K" problem. Personally, I'm not losing a lot of sleep.

But, seriously, not many know it, but a more imminent bug is buried in the %DT and %DTC routines—one that applies to dates a mere 102 years from now. As Marcus Werners (Marcus@DHZB.DE) has discovered, FileMan now considers the year 2100 to be a leap year, whereas of course, by Pope Gregory's algorithm, it is not! Your trusty FileMan not only allows the invalid date "4000229" to be input and stored, but also miscalculates by one the number of days from then on. The DVA plans to release a fix to %DT and %DTC soon, but it turns out that the (very dense) code embedded in these routines to calculate leap years actually was "borrowed" (by me) from M code written in the 1970's at Massachusetts General Hospital—code which did not consider what would happen in 2100! It just goes to show once again how long these software glitches can live on. **M**

George Timson (gtimson@pacbell.net) was the principal author of VA FileMan. He is now an independent consultant living in Berkeley, CA.
