M Technology Threads

Contributors: Ben Bishop, Joe Charnock, Roger Cope, Don Gall, Lee Hirz, Scott Jones, Bobby McBee, Michael Poxon, Kate Schell, Sean Kelly, Jim Self, and David Whitten (with some editing and rearranging assistance from Valerie Harvey).

Online discourse is an important realm for exploring technical topics. Again I have selected topics from recent M-List traffic and used a format that will treat collections of excerpts as a contribution to *M Computing*. This approach provides print recognition to those who take the time to share their technical insights and engage in such discussion. (My ulterior motive is still: I would like to encourage eventual technical article contributions to be published in *M Computing* from some of these folks!)

For this issue the thread topics are: *The Year 2000* and *The Bitwise AND in M*. The opinions expressed here are not necessarily the opinions of the MTA or even of the other contributors!

Threads are conversations—they have elements of "parallel distributed processing"—parts of the conversation go on independently. It is difficult to find an appropriate "linear" arrangement for this mode of presentation and this format cannot precisely represent the dynamics of online interaction.

It is important that more M users have contact with those who are exchanging technical ideas and that the authors get credit and acknowledgement in print. The representation of the "thread" intentionally has a different appearance from the full intertwining and embedding of the internet exchanges—each contributor is acknowledged as separately as is feasible for ideas and expression. The result is at the same time far less-formal (given the conversational tone and cross-references between posts) than a technical article.

The "Year 2000" Thread

Don Gall: If you use any of the varied vendor-supplied date routines, you need to watch out for some discontinuities on January 1, 2000. For example, the DTM routine ^%datecheck returns a string. One piece of this is the date in a YearMonthDay order. For 12-31-99, it returns 991231. The following date is returned as 20000101. That is 1900 years and 1 day later [for arithmetic using these strings].

Kate Schell: [Embedded in a post about M technology.] It still runs fast. The databases are lean and comparatively inexpensive, and it uses the "Relative Integer Date" approach that was so thoroughly discussed in comp.software.year-2000 several months ago.

[The] RID (known as $Horolog) is a system variable, not a data type.

Lots of MUMPSters are rather complacent about their code and databases, because of the RID factor. They ought to be looking at it a lot more closely. Many of the routines and functions supplied by their vendors to manipulate the RID are not Y2KOK. Arithmetic performed using the resulting two-digit years can have life-threatening consequences.

Like the rest of the world, we need to get the lead out and get moving on code corrections.

Joe Charnock: This also sounds very unlikely to me [that Y2K work in M looks to be potentially far more difficult than COBOL work]. As well as the M internal date format, making writing Y2K-compliant code easier and therefore more likely in the first place. The nature of M as an interpreted language (at least as far as the programmer is concerned) makes the task of making lots of very small changes to a large number of routines a whole lot quicker. You don't have to compile, link, sort out your copy members, allocate databases etc. etc. as I'm currently having to do in COBOL. This takes a **huge** amount of time.

Jim Self says that to change a database used by a COBOL system to cope with the wider fields needed for dates including the century would be very difficult. I've no doubt this is true and that to convert an M data

base in a similar way would be much easier—however this doesn't matter if the Y2K problem is being dealt with by leaving the databases as they are and changing the logic of the code to treat certain dates (for example those with years less than 50) as being 21st century and the rest as being 20th century. This works, though it clearly won't work forever but of course these old "legacy systems" aren't going to be around forever.

Roger Cope: I'm an ex-NHS M employee currently doing Y2K remedial work on non-health service M systems. In the first place, M is inherently Y2K compatible. The only three reasons why M would have any problems with the Y2K are:

•programmers relying on non-standard date-handling routines instead of using ones built in standard M

• programmers deliberately converting the default (Y2K compatible) date to a non-compatible format and storing it in the database

• interfacing to external components

In the real world of course, all these things have happened over time but my own experience is that M is very easy to convert for Y2K issues. The language was designed to be a RAD environment many years before such concepts were talked about and in direct response to the old fashioned and much slower methods used by COBOL programmers.

Although it's true that there are many fewer M programmers in the U.K. than COBOL programmers, it's also true that far fewer *are required* to complete the work involved.

This isn't an attempt to trivialize the job. In most workplaces, systems will have to undergo the same analysis, documentation and testing phases whatever they're written in and these phases will take significant amounts of manpower and budget. But, in my experience, the coding required is a significantly smaller part of the whole than in systems written in other languages.

To take a specific example (my information is only taken from press reports), the Family Health Services computer systems in the U.K., written entirely in M and representing a distributed database containing the non-medical records of the entire population registered with doctors, together with various systems using this data (including doctor payment systems) has a Y2K budget of £3million. I would compare this system against the consumer systems of a typical national bank, most of which have reported Y2K costs of several 10s of millions of pounds.

So to conclude an overlong posting, M may be under-resourced in the U.K. but the environment is particularly suitable to low cost, low manpower re-engineering of the kind required by the Y2K problem and is, as far as the health service is concerned, the least of its worries. I'd be much more concerned about the hardware they use and the non-M systems that are equally underfunded.

Sean Kelly: Leaving the databases as they are should not be the preferred option. Changing code to treat certain dates as either 21st century (e.g., year<50) or 20th century (e.g., year>49) is a recipe for further trouble. For a start, it won't work with dates of birth. So you would then need to further refine your algorithm by distinguishing between different types of date (e.g., actual date of birth can only be today or before).

Also, how will you distinguish between someone born in 1898 and someone born in 1998? As people live longer, this situation will become more common. Or will scheduling pediatric appointments for everyone over 100 years old be a price worth paying for such laziness?

Fix the dates, man !!

The "Bitwise AND in M" Thread

Bobby McBee:
The bit-wise compare is the "&" symbol.
Example:
S A=1,B=1
I A&B ....(TRUE)
S A=1,B=2
I A&B ....(FALSE)

Scott Jones: That actually is NOT a bit-wise AND . . . it is a logical AND. There is no bitwise AND in **standard** M (and there isn't a conditional AND in M either . . . even though many M programmers mistakenly think that there is . . .) Your second example is incorrect. 1&2 evaluates to 1 in M, not 0.

In some dialects of M (such as MSM, ISM, and Cache'), there is a somewhat crufty function called $ZBOOLEAN, which takes two expressions and an

integer expression which specifies the type of boolean operation requested.

The cruftiness is because this function . . . breaks one of the normal conventions of the M language . . . that M is always a typeless language . . . (i.e., "1" is always equivalent to 1). It depends on the **INTERNAL** type of an expression and on implementation-dependent behavior. It really should have been split up into two separate functions, one that treated strings as arrays of bytes, and another that operated on integer values. (The same cruftiness is present in the $ZHEX(expr) function, which, if the INTERNAL type of the expression is string, does a hex to decimal conversion, but if the INTERNAL type is numeric, it does a decimal to hex conversion. Once again, this should have been two functions) conditional AND (i.e., && in C, C++, Java, and some other languages) and conditional OR (i.e., || in many languages), where the second expression is not even evaluated, is not available in M, but can be simulated using tricks with $SELECT() (i.e., a && b -> $SELECT('a:0,'b:0,1:1) and a || b -> $SELECT(a:1,b:1,1:0)).

Many M programmers are under the mistaken impression that *IF a,b,c statements* is an example of a conditional AND. However, it really is just a consequence of the normal M rule that command ARG, ARG, ARG is the same as command ARG command ARG command ARG.

It really is nothing more than doing:
if (expr1) if (expr2) if (expr3) {statements}
in other languages . . . .
That still doesn't mean that M has a conditional AND operator. I've seen many mistakes, where people put something into a variable that they **think** is an expression, but is really a list of expressions . . . which does work in the context of an IF, such as IF @var statements, but then breaks when people try to do something like: SET:@var x=y. I've also seen a lot of impossible to understand code, where it was really trying to get around the lack of a conditional OR operator in the language (and people didn't understand that $SELECT can be used to provide the functionality). . .
(i.e., IF 'a,'b ELSE statements
instead of: IF $S(a:1,1:b) statements.

Jim Self: The M fragment *IF a,b,c statements* is equivalent to *IF a IF b IF c statements* in that "b" will be evaluated only if "a" is true and "c" will be evaluated only if both "a" and "b" are true and "statements" will be evaluated only if all three are true.

That certainly fits my idea of a logical construction which could well be referred to as a "conditional AND." However, "a,b,c" is a list of arguments and not an EXPRESSION. It therefore cannot be used as part of a larger expression in quite the same way as the conditional OPERATORS, such as "&&" and "||" of some other languages.

David Whitten: On the issue of bit-wise operations as presented to the MDC, the two approaches I have seen proposals on are:

a) the Internal Bit-Stream approach: Where a string is examined from beginning of string to end of string, and the internal representation of the characters is viewed as a series of bits. As Jim Self has mentioned, this is prone to problems with Endian issues. (i.e., if you internally store "AB" as 66 65 because 16-bit entities are stored in this fashion, do you have to reverse all the pairs of bytes to determine which bits are "on" and "off" ?) There is also a question about Internationalization, since there is no guarantee in M that a character is represented internally by eight bits. (in fact, in ISO-10646, it is stored (conceptually) as 32 bits).

b) the BitString approach: This method says that a series of bits is represented externally as if it were a series of "1" and "0" characters. All issues of internal storage are hidden by the M Implementation. The Bit Manipulation functions are then defined as operating on strings of characters, with the understanding that the vendor is free to represent the result and input values in whatever way they deem most efficient.

Perhaps someone else remembers another approach that has been presented. As I recall, this issue has been raised many times over the years, but there has not been enough discussion outside the MDC for it to be included in the Standard.

Jim Self: Where "assert" represents some boolean assertion about records in a database, and "bitstring" would be of a fixed length, say 1K (1024), so that it would represent the assertion over 8192 (8*1024) records and bit "b" of "bitstring" would represent the assertion for record 8192*n+b. Then considering, for example, a large database of clients where such an index contains assertions regarding age and sex categories such that "M" represents the assertion that a client is male and "over 40" represents the assertion

that a client is over 40, you could count the number of male clients in a given set of 8192 clients with the function $ZBITCOUNT(^IX("M",n)) or construct a string representing which of a set of 8192 clients are both male and over 40 with $ZBITAND(^IX ("M",n), ^IX("over40",n)).

For various reasons I haven't used these functions much beyond limited experimentation but I think that they could be extremely fast and efficient for working with certain kinds of large data sets. I wonder if anyone here has used these functions very much or in this way. Are they, or something like them used in the implementation of SQL interfaces by InterSystems or Micronetics or others?

[In response to: The MDC is driven, in great part at least, by user demand; there is no great user demand for these "features."]: I think that the lack of perception of demand has often been confused with an absence of demand. People who truly "demand" such features could never wait for the MDC. They would simply go elsewhere.

We have long supported extensive content search capabilities for diagnostic reports and patient records in our Hospital Information System through automatic indexing and retrieval of words in context. I believe this kind of capability is common to many M systems. We expect to begin work in the near future to add a semantic layer of indexing and retrieval but that is still in a formative stage.

Michael Poxon: Thanks! Where were you when I was complaining about having to do back-to-back conditional QUITs when using $Q? Example:

```
F  S G=$Q(@G) Q:G=""  Q:$QS(G,1)'=3 . . .
```

I guess I should've realized I could rewrite that as

```
F  S G=$Q(@G) Q:$S(G="":1,$QS(G,1)'=3:1,1:0)
. . .
```

I find that to be a little more elegant (or clever, at least). But . . . now that I've seen the alternative, I might stick with the original just so somebody coming behind me doesn't have to sit and wonder what that code is trying to do.

Lee Hirz: You have to make use of the ability of M to know the ASCII value of the character being evaluated as follows. If X contains one character, then $A(X)= "an ASCII value." Using this fact the ASCII value can be manipulated as desired. Bits can be AND'd or OR'd together. So to do a bit by bit AND of bytes stored in the variables X & Y as single characters, you would first calculate their ASCII values. Then you need to find out if each bit is set. When both bits are set, you need to add the value of that bit to a sum.

In the following code, AX and AY are ANSII values for X and Y, the answer is a character that contains the result of bit by bit ANDing of X and Y.

```
Set AX=$A(X),AY=$A(Y),Answer=0
For J=2:2:512 Do
. Set BX=AX#J\(J/2)
. Set BY=AY#J\(J/2)
. S Answer=$Select(BX&BY:1,1:0)*J+Answer
. Q
```

And if you want to change to OR'ing the bits, simply change the $Select to "$Select(BX!BY:1,1:0)."

Ben Bishop: The trouble starts when you get to strings instead of single characters. How do you "AND" "ABCD" with "XYZ"? Which side do you pad out? (and I presume one would pad out with $C(0).)

What happens when one character is a 16-bit character (like Kanji) and the other is straight ASCII (7 bit)?

Unicode makes the possible character problems even "worse" with 4-byte/ 32-bit characters.

Ninety percent of the work always seems to be those niggling little details that you don't encounter often, but you have to deal with.

(I'd also have done the 1st $SELECT as $S('BX:0,'BY:0,1:1) and the OR version as $S(BX:1,BY:1,1:0)—I'd expect them to be faster just from not "always" having to lookup the "BY" value.)

Scott Jones: Unicode is not 4 byte 32-bit characters. That is ISO 10646. Unicode is 2 byte 16-bit characters (which is mapped into the first 64K of ISO 10646, and is also known as the BMP in 10646 . . . the Basic Multilingual Plane).

[Regarding: I'd also have done the 1st $SELECT as $S('BX:0,'BY:0,1:1) and the OR version as $S(BX:1,BY:1,1:0)—I'd expect them to be faster just

from not "always" having to lookup the "BY" value)]: Actually, you'd do $S('BX:0,1:BY) for the conditional AND, and $S(BX:1,1:BY) for the conditional OR for best performance . . . M

—Valerie J. Harvey, Ph.D.

Ben Bishop	*aci@SHORE.NET*
Joe Charnock	*joec@lineone.net*
Roger Cope	*roger.cope@virgin.net*
Don Gall	*dongall@GOODNET.COM*
Lee Hirz	*lhirz@hanfs01.ha.osd.mil*
Scott Jones	*scott@intersys.com*
Sean Kelly	*sean@qmi.co.uk*
Bobby McBee	*bemcbee@flash.net*
Michael Poxon	*Michael.Poxon@kp.ORG*
Kate Schell	*cschell@jaquardsystems.com*
Jim Self	*jaself@ucdavis.edu*
David Whitten	*whitten@NETCOM.COM*

VIRGINIA MASON | Medical Center

Virginia Mason Medical Center, one of the Northwest's most respected healthcare providers, is located in beautiful Seattle, Washington, where the air is clean, the crime rate low and the climate is mild. Boating, skiing, biking, golf, swimming, roller blading, windsurfing, fishing, hiking, you name it - it's a recreational paradise. And the arts and music scenes are phenomenal. Conde Nast Traveler calls the Puget Sound Region one of the world's top vacation destinations. At Virginia Mason, we call it home. Virginia Mason is enjoying tremendous growth. To manage this growth while maintaining our high standards, we have the following opportunities available:

SYSTEMS ANALYST
PROGRAMMERS - MUMPS

Mid to senior level opportunities for programmers with 3-5 years experience in M Language & DCL programming. IDX experience preferred.

As one of the Northwest's most respected healthcare providers, we offer competitive salaries and a comprehensive benefits package. Relocation assistance is available. Please apply with resume to: Virginia Mason Medical Center, PO Box 900, Attn: C. Hizon, M6-PFS, Seattle, WA 98111, email: hrocdh@vmmc.org Fax (206) 223-6924. EOE. ***To learn more about Virginia Mason, please see our website at www.vmmc.org***

MTA Job Referral Service

Your key to a successful job search!
Your key to finding a qualified employee!

EmployEEs:
- Free listing in **JRS** to **MTA** Members
- Receive names of **ALL** employers in **JRS**
- Your resume will be sent out for 6 months

EmployERs:
- Employers may use the **JRS** for a fee
- Receive an initial packet of at least 40 to 50 resume briefs
- Your job opening(s) sent to **ALL** job candidates

Let MTA help you unlock doors.
To receive your key, contact Marlo Brown:
M Technology Association - North America
phone: (301) 431-4070 fax: (301) 431-0017
email: **MTA@mtechnology.org**

Advertiser Index

We appreciate these sponsors of the March issue and all the companies who support the M community through their commitment to excellence.

This index appears as a service to our readers. The publisher does not assume any liability for errors or omissions.