Building the Millennium Standard

or . . . "Library" Controversy Resolved

The MUMPS Development Committee meets several times in a year to discuss proposals to enhance the M Technology ANSI standards, including the M Language standard. The MDC last met in September and plans to meet again in March 1998. Several M Computing *readers who were not able to attend the MDC meetings are, nevertheless, interested in some of the specifics of these proposals. We'll try to, very briefly, describe a few of them in this column in each issue of* M Computing.

The MUMPS Development Committee plans to begin the canvass process in 1998 to revise several of the current M Technology ANSI standards. If you would like to participate in the canvass process to seek ANSI approval, contact the MDC Secretariat at the MTA office to be put on the preliminary canvass list.

An MDC Type A proposal is one that will be included in the next revised version of the MDC standard document. The MDC standard is the draft standard that will be canvassed for ANSI approval as a revision of the current ANSI standard document.

This is a brief summary of one of the recent proposals that the MDC has passed to MDC Type A status. Not all the sections of the proposal or the formalization of the actual changes proposed for the current ANSI standard have been included.

"Library proposal", document X11/94-23.

A library is a collection of library elements, with unique names, which are referenced using a single library name. A library is defined as being either mandatory or optional.

A library element is an individual function which is separately defined and accessible from a MUMPS process using the library reference syntax.

The MUMPS Standard Library consists of all libraries and library elements defined within the MUMPS Standard, whether mandatory or optional.

Justification of the proposed change:
It is desirable to make available to developers in MUMPS a number of facilities which are specialized in functionality or which do not justify incorporation into implementations as intrinsic functions. In order to make these truly useful in portable programs they need to be available on all MUMPS systems. In addition, it would be desirable if facilities could be implemented in all MUMPS systems without having to rely on the MUMPS vendor to provide them. This would allow the construction and sharing of useful functionality and allow for reduced time to market for new products and developments.

Existing practice:
No existing portable MUMPS libraries exist, although individual application developers have produced libraries which are used throughout their own and other applications. Each developer has to ensure that the names of such libraries and their entry points do not overlap with other ones.

In other languages many functions are standardized in libraries and can thus be used by application developers without worrying about being portable. Some of these libraries are provided as part of an implementation and some as add-ons.

General description of the proposed change:
This proposal sets out a format for specifications of library elements, how they are called, how they are made available (or just mandating that an implementor provide a mechanism for users to include their own code) and means to test what is or is not available.

This proposal would be valuable to users by providing a higher degree of portability of code, without collision of names between different packages and routines installed on the same system.

The impact on vendors is relatively minor. An interface to some calling table would need to be provided, which would allow either MUMPS or non-MUMPS code to be called from a function call.

Example of the use of the proposed change:

"Library definitions"

A library function `SIN` might have a header definition of

```
SIN^MATH:REAL(X:REAL)
```

followed by the definition of the meaning of the `SIN` function. This means that `SIN` is called, via the function syntax, from the MATH library. It has one real valued parameter (called X in the subsequent definition) and it returns a real result.

Another library element might be `PI^MATH:REAL` to return the mathematical value of pi. This definition allows for no parameters, but returns a real result.

Another example of a library definition might be:

```
REPLACE^STRING:STRING(str:STRING,.SPEC:STRING)
```

which takes a string and transforms it according to the array SPEC. (Although this is only an input array, it needs to be passed by reference since it is an array.)

Finally,

```
MOVE^STRING(.in,.out,transform,max:INTEGER:0)
```

might be the definition of a function which moves data from one string or array to another according to some transformation algorithm, which is executed a maximum number of times (optional parameter). It has a result which is a string, but this might be a success code or even always a null string.

"Library accessing"

The SIN function defined above would be accessed by:

```
SET X=$%SIN^MATH(Y)
```

If this reference were coded as `SET X=$%SIN(Y)`, then this wouldn't necessarily use the definition of `SIN^MATH` above. Instead, the library(s) defined in `^$JOB($J,"LIBRARY")` would be used to access a `SIN` function.

The PI library function definition above is accessible by

```
S X=$%PI^MATH
```

since it has no parameters.

Finally, the MOVE example would be accessed as

```
S A=$%MOVE(.A,.B,C)
```

```
"^$LIBRARY"
```

The above definitions would result in the following nodes in `^$LIBRARY:`

```
^LIBRARY("MATH","ELEMENT","PI")
^LIBRARY("MATH","ELEMENT","SIN")
^LIBRARY("STRING","ELEMENT","REPLACE")
^LIBRARY("STRING","ELEMENT","MOVE")
```

At run-time `^$JOB($J,"LIBRARY")` would contain the accessible libraries. If the above two (MATH and STRING) were the only available ones, then the entries might be

```
^$JOB($J,"LIBRARY",1)="MATH"
^$JOB($J,"LIBRARY",2)="STRING"
```

to search through MATH before STRING, or

```
^$JOB($J,"LIBRARY","A")="STRING"
^$JOB($J,"LIBRARY","X")="MATH"
```

to search STRING before MATH. Note that the final subscripts 1,2, "A", and "X" are used as examples in order to get the right sequencing only.

"The above MDC specification has been approved by the MUMPS Development Committee but it may be a partial specification that relies on information appearing in many parts of the MDC standard. This specification is dynamic in nature, and the changes reflected by this approved change may not correspond to the latest specification available. Because of the evolutionary nature of MDC specifications, changes are likely to occur in the specification released prior to a complete republication of the MDC standard."

To keep current on all details of the proposals that the MDC is considering you can subscribe to the MDC document mailings (a one-year subscription costs $120 plus shipping and handling) through the MTA. **M**
