X11/94-23

# MUMPS Development Committee

Extension to the MDC Standard
Type A Release of the MUMPS Development Committee

# Library proposal

June 1994
Produced by the MDC Subcommittee #15
Programming Structures

Jamie Crumley, Chairman
MUMPS Development Committee

Kate Schell, Chairman
Subcommittee #15

The reader is hereby notified that the following MDC specification has been approved by the MUMPS Development Committee but that 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, the reader is further reminded that changes are likely to occur in the specification released, herein, prior to a complete republication of the MDC Standard.

# 1. Identification

## 1.1 Title Library proposal

## 1.2 MDC proposer and sponsor

Proposer/Sponsor: Jon Diamond, Hoskyns Group

## 1.3 History of MDC actions

| Date | Doc# | Action |
|---|---|---|
| June 94 | This document | Approved as MDC Type A |
| February 94 | X11/94-13 | Approved by SC15 Type A (23:4:4) |
| December 93 | X11/SC15/94-3 | Approved by MDCC-E (5:0:1)<br>1. All issues raised in Oct 93 meeting addressed.<br>2. Clarification added in complete library being mandatory/optional, not individual elements<br>3. Mandatory/optional flag added to library element parameter definitions |
| Aug/Oct 93 | X11/SC15/93-32 | Approved by MDCC-E and proposed as SC15 Type A (remanded to TG)<br>1. Access to library elements changed from $&LIB.SIN to $%SIN<br>2. The change in access implies that all library elements must have a result, with resultant changes to the proposal.<br>3. NAME added as a data type. |
| Jul 93 | X11/TG14/93-4 | Approved as SC15 Type B |
| June 93 | X11/TG14/93-2 | Approved as SC15 Type B with X11/TG14/93-4 corrections |
| June 93 | X11/93-26 | The concept of CODE and PROCEDURE library elements deleted. Individual library elements allowed to have character set profiles. |
| Feb 93 | X11/93-2 | Discussed in MDC/TG14 and MDCC-E |
| Oct 92 | X11/92-44 | Proposed as Type B - discussed in general terms in MDC/TG14 |
| Aug 92 | | Library extensions to ssvns |
| Apr 91 | X11/SC8/91-2 | Proposal guidelines and requirements for SC8 proposed as Type B |
| 1990 | ? | Other proposals for library function specifications |

# 2. Justification

## 2.1 Needs

There are a number of facilities which it is desirable to make available to developers in **MUMPS** which are specialised 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.

### 2.2 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 name of such libraries and their entry points do not overlap with other ones.

In other languages many functions are standardised 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.

## 3. Description Of The Proposed Change

### 3.1 General Description

This proposal attempts to meet all the requirements as expressed at the meeting of MDC/TG14 on 20th October 1992. These can be summarised as:

| | Mandatory | Desirable |
|---|---|---|
| Availability | Standardized, but may be replaced by user MDC defines mandatory library elements User defined libraries are not always mandated ssvn should be available to determine presence Must be guaranteeable on all MUMPS systems | Mandatory libraries should be Optionally installable |
| Implementation | Implementable | Mandatory libraries should be optionally installable May be supplied in MUMPS at vendors or users discretion May be supplied by other sources |
| Calling mechanism | Explicit namespace separation required (uniquely resolvable) | Call syntax must be simple C-style include should be available Should be able to escape to different library for a single call |
| Content and specification | Libraries may contain procedures, functions, data or code-segments MDC may support a registry of library elements | |

The 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.

Note: The issue as to whether this document should document how ^$JOB($j,"LIBRARY") has been discussed and the TG agreed that this was a more general issue relating to ssvns that should be tackled as a separate exercise.

### 3.2 Annotated Examples of Use

See X11/TG8/93-15 for a comprehensive example of a sample definition of a library element.

*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.

In words this means that SIN is called via the function syntax from the library MATH, has one real valued parameter (called X in the subsequent definition) and returns a real result.

Another library element might be

```
PI^MATH:REAL
```

to return the mathematical value 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:O)
```

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 were the only available ones then the entries might be ^$JOB($J,"LIBRARY",1)="MATH" and ^$JOB($J,"LIBRARY",2)="STRING", to search through MATH before STRING, or ^$JOB($J,"LIBRARY","A")="STRING" and ^$JOB($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.

### 3.3 Formalization

Amendments to RMDS X11/TG6/93-6

Add libraryref to the list of alternative elements in the definitions of exfunc and exvar (7.1.4.8 and 7.1.4.9)

Add a new section

8.1.6.4 Library reference

```
libraryref ::= % libraryelement [ ^ library ]
libraryelement ::= name
library ::= name
```

If no library is specified as part of a libraryref then the libraries specified in ^$JOB($J,"LIBRARY") are used. Note: This does not imply that the libraries specified in ^$JOB($J,"LIBRARY") can necessarily be dynamically changed during the lifetime of a process.

Unless explicitly specified in an individual libraryelement definition accessing a libraryref has no effect on local variables for a process, $REFERENCE and $TEST, except for a return value and changes to variables passed by reference.

If an argument to a libraryref has an invalid value (such as a value outside the domain of the function) the behaviour of the reference to the libraryref is undefined.

The restrictions specified in 8.1.7 Parameter passing also apply to the referencing of libraryrefs.

If a 1libraryelement or a library is not available for a library reference then an error condition occurs with ecode = "M13".

Add a new section to the **MUMPS** standard.

n. Library

n.l Library definitions

A library consists of a set of libraryelements - functions and data which are accessed from **MUMPS** and which have unique names within the library. The access method for each libraryelement is the external calling syntax,which normally has no side-effects.

A library is defined as being either mandatory or optional. library names starting with a Z are reserved for implementors. library names starting with a Y are reserved for users. All other unused library names are reserved for future use.

The **MUMPS** Standard Library is the set of library definitions in this standard.

The following librarys are defined:

n.1.1 Mandatory Libraries

. . .

n.1.2 Optional Libraries

. . .

n.2 Library Element Definitions

The definition of a libraryelement states which library the element belongs to, return value type and full specification.

libraryelement names starting with a Z are reserved for implementors. libraryelement names starting with a Y are reserved for users. All other unused libraryelement names are reserved for future use.

A libraryelement definition is of the form:

```
libraryelementdef ::= libraryelement ^ library
      libraryresult [ (L libraryparam ) ]
libraryparam ::= [.] name  [ : [ libdatatype ] [:
      libraryopt ] ]
libraryresult ::= [ : libdatatype ]
```

```
                  |     BOOLEAN          |
                  |     COMPLEX          |
                  |     INTEGER          |
libdatatype ::=   |     NAME             |
                  |     REAL             |
                  |     STRING           |
                  |     Z[unspecified]   |

libraryopt ::=    | M |
                  | O |
```

If a libraryparam starts with a period then this parameter is-passed by reference.

Z is the initial letter reserved for implementation specific libdatatypes. All other values for libdatatypes are reserved for future expansion of the standard.

Input and output values to libraryelements undergo the appropriate data interpretation below:

For BOOLEAN see 7.1.4.7 (Truth value ).

COMPLEX is a number represented in the format REAL_"%"_REAL, (that is two REAL numbers separated by the % character).

For INTEGER see 7.1.4.6.

For NAME see 7.1.5.15 (namevalue).

For REAL see 7.1.4.5 (Numbers not constrained to be INTEGER).

STRING is a string made up of any characters and not constrained in format.

If no libdatatype is specified for a libraryparam or libraryresult then the libdatatype defaults to STRING.

If no libraryopt is specified then the libraryparam is M (mandatory). A libraryopt of 0 specifies that the libraryparam is optional.

n.3 Availability of library elements

An implementation of **MUMPS** shall

a. provide the mandatory librarys defined in this standard

and

b. provide a means by which replacement definitions in routines of libraryelements can be installed so that a routine can access them as if they were

part of the implementation. An implementation may additionally provide a means by which non-**MUMPS** code can be installed to implement libraryelements.

An implementation may also provide a means by which specific librarys or libraryelements of the MUMPS standard library are only optionally installed.

n.4 Library elements

Note: The **MUMPS** code that approximates any function in the following definitions only serves as an example.

Editors Note: Library definitions will follow here in the standard.

Add a new ssvn

n.n ^$LIBRARY

^$LIBRARY provides information about the availability of libraries and library elements in a system.

When and only when a library 1 exists, ^$LIBRARY(1) has a value; all non-empty string values are reserved for future expansion of the standard. Library information is stored beneath the ^$LIBRARY(library) node:

```
^$LIBRARY(libraryexpr,expr V
     "ELEMENT",libraryelementexpr)

libraryexpr  ::= expr V library

libraryelementexpr  ::= expr V libraryelement
```

When and only when a library 1 and libraryelement e exist, ^$LIBRARY(1,"LIBRARY",e) has a value; all non-empty string values are reserved for future expansion of the standard.

Insert a new ssvn node:

```
^$JOB(processidexpr,expr1 V "LIBRARY",expr2) =
     libraryexpr
```

This node identifies a library currently available to the process. The order in which the librarys are searched to locate a specific. libraryelement is defined by the collating order of the values of $expr_2$ for the specified librarys.

Editors note: There is already a prohibition on updating ssvns unless specified otherwise. Since this is not done here "$JOB($J,"LIBRARY") could be fixed by the implementation for the lifetime of a process.

# 4. IMPLEMENTATION IMPACTS

### 4.1 Impact on Existing User Practices and Investments

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

### 4.2 Impact on Existing Vendor Practices and Investments

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.

# 5. Closely related standards activities

### 5.1 Other X11 Proposals (Type A or Type B) Under Consideration

X11/SC13/93-54 REPLACE library function, -53 PRODUCE library function, -56 Mathlib/General, -57 Mathlib/Complex, -58 Mathlib/Trigonometry, -59 Mathlib/Hyperbolic etc

### 5.2 Other Related Standards Efforts

ISO/IEC 9899 - C Programming language.
ISO/IEC DIS 11430 - Generic package of elementary functions for ADA.

### 5.3 Recommendations For Co-ordinating Liaison

None.

# 6. Associated documents

X11/TG8/93-15 Library Proposal Format

# 7. Issues, Pros and Cons and Discussion

### February 1994 meeting

Pros: Provides definition for libraries. Does not allow for optional elements within a library. Allows MDC to declare a library either mandatory or optional. SC decided library elements should not be optional.
Cons cited: Incomplete document. Does not allow for optional elements within a library. Unfortunate use of metalanguage term datatype.

### December 1993 MDCC-E meeting

Mandatory/optional suggested as part of parameter specification.

### October 1993 meeting

Con: Incomplete formalization (index). Doesn't provide implementation specific datatypes. Semantics of charset unclear. All these issues are addressed in the revised proposal.

### June 1993 meeting

Potentially dynamic nature of ^$JOB makes binding to external functions very difficult, since this cannot be determined until run-time. Parameters passed by reference should be handled as for external packages, however some library elements need this for their operation (eg $REPLACE). Pro: Needed in many areas of work of the MDC, Allows search order specification
Con: Library list should be subscripted
Straw polls:
Allow for NAME as datatype (7:1)
Library list to be subscripted (14:1)

## 8. GLOSSARY

### Library

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.

### Library element

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

### MUMPS Standard Library

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