MUMPS Development Committee

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

Portable length limit of names

October 26, 1995

Produced by the MDC Subcommittee #15
Programming Structures

Ed de Moel, Chairman
MUMPS Development Committee

Art Smith, 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 of proposed change

1.1 Title

Portable length limit of names.

1.2 MDC Proposer and Sponsor

SC15/TG11 – Portability Size Issues
Chair: Rod Dorman
Polylogics Consulting, Inc.
136 Essex Street
Hackensack NJ 07601

phone: (201) 489-4200
fax: (201) 489-4340
email: rodd@panix.com

1.3 Motion

This proposal has reached MDC type A status superseding X11/SC15/95-34.

1.4 History

March 1996	<this document>	Final pros & cons recorded.
October 1995	X11/SC15/95-34	SC15 editorial corrections incorporated, submitted to become a MDC type A. Passed 26-0-2
April 1995	X11/SC15/TG11/95-1	Submitted for task group discussion and elevation to SC15 type A. Passed 21-1-5
January 1995	X11/SC15/TG11/94-1	Submitted as a revised SC15 type B subsuming both documents X11/SC15/94-28 and X11/SC15/94-29. Passed 18-2-2
June 1994	X11/SC15/94-28 X11/SC15/94-29	Task group and SC15 editorial corrections incorporated, submitted to become a MDC type A. After much discussion was withdrawn.
October 1993	X11/SC15/TG11/93-5	Task group editorial corrections incorporated, submitted to become a SC15 type A. Vote passed 13-6-3
March 1993	X11/SC15/TG11/93-1	Results of task group decisions incorporated, submitted to become a SC15 type B. Vote passed 27-3-5.
February 1993	X11/SC15/93-10	Original proposal submitted to become a SC15 type C. Assigned to a new task group, TG11.

1.5 Dependencies

None

2. Justification of proposed change.

2.1 Needs

There currently is a conflict in the standard where the portability limit on names restrict them to eight characters and there exists an intrinsic special variable name (PRINCIPAL) and an intrinsic function (TRANSLATE) which exceeds the limit.

M is being promoted as a "modern day" programming language yet identifiers are restricted to only eight characters. Languages such as COBOL, C, PL/1, and even some dialects of Basic allow longer names. Some have done so for decades!

There has been a lot of interest in having M inter-operate with processes outside of the M environment. The API descriptions of these are often given in terms of C calls and variables in which the identifiers are significantly longer than eight characters. The ability to use equivalent identifiers in M would greatly ease the task.

2.2 Existing practice in Area of Proposed Change

Those wishing to adhere to the portability limits perform some mapping of what they'd like to name something to an eight or less character name. When "prefix" characters are required for a naming convention this mapping process becomes harder. Note that different people might choose different encodings and could misinterpret the meaning of an identifier. e.g. does Ptr mean Pointer or Printer, does Pat mean Pattern or Patient?

Those willing to trade portability for readability, or to take advantage of a vendors increased limit, do so with the knowledge that they could get different results on different implementations. A reference to an undefined variable would be immediately apparent. "Changing" a different variable (thus not changing the intended one) could postpone the realization and discovery of the bug. Hopefully the first eight characters of the "extended" name are unique and the spelling is consistent.

3. Description of Proposed Change

3.1 General description of proposed change

The portability limit for the length of a name is increased. **ALL** characters in a name will be significant! Extra characters beyond an implementers limit will be in error. It should be noted that the introduction to the Portability Section states that an implementation must meet or exceed the limits in the Portability Section. A vendor may exceed the limit and allow names of more than 31 characters but **ALL** the characters in the name will be significant, not just the first thirty-one.

3.2 Annotated examples of use

```
SET X=$PRINCIPAL ; sets X to the principal I/O device
SET X=$TRANSLATE("ABC","B","-") ; sets X to "A-C"
SET X=$QSUBSCRIPT(NAMEVALUE,3) ; sets X to the value of the third subscript
SET X=$$XDPSEXTENSIONPRESENT
SET STATUS=$$^XDPSIMAGEFILEINTODRAWABLE(DPSCONTEXT,PIXELSPERPOINT)
DO ^SWAPBITSRASTPORTCLIPRECT(RASTERPORTID,.ALTERNATEBITMAP)
```

3.3 Formalization (References are to Canvass Document for ANSI/MDC X11.1-1994)

Replace Section II Clause 2.1 with:

> The use of ident in names is restricted to upper case alphabetic characters. Portable name length is limited to thirty one (31) characters. All characters in a name are significant in determining uniqueness. Therefore the length restriction places an implicit limit on the number of unique names on an implementation. If a name's length exceeds an implementor's limit an error condition occurs with ecode = "M56".

4. Implementation Effects

4.1 Effect on Existing User Practices and Investments

There are several cases to consider:

- Routines written with name lengths limited to eight or less characters will not be affected.
- Routines written with name lengths limited to a specific vendors extended limit will not be affected.
- Routines written that exceed the name length limit with a consistent spelling will not be affected unless the name is more than 31 characters.

- Routines written that exceed the name length limit with inconsistent spellings that are intended to be the same will probably be affected. Note that a static analysis of the routines can highlight all instances of names (except those occurring in indirection or executes) that exceed the limit and can check and highlight spellings that differ.

4.2 Effect on Existing Vendor Practices and Investments

Some implementers already use characters beyond the eighth to determine name uniqueness. Implementers may have to modify their data structures and/or symbol table algorithms to accommodate longer names.

4.3 Techniques and Costs for Compliance Verification

Set two local variables whose names differ only in the portability limitth character to different values. WRITE them out and ensure the values are correct. Attempt to set a local variable whose name is vendors limit+1 characters long and ensure that an "M56" error occurs.

Set two global variables whose names differ only in the portability limitth character to different values. WRITE them out and ensure the values are correct. Attempt to set a global variable whose name is vendors limit+1 characters long and ensure that an "M56" error occurs.

Create two different routines whose names differ only in the portability limitth character. Ensure that both were created. Attempt to create a routine whose name is vendors limit+1 characters long and ensure that an "M56" error occurs.

Create a routine with two subroutines with labels whose names differ only in the portability limitth character. Have the code in the two subroutines write different values. DO the two labels and ensure the values are correct. Create a routine with a label which is vendors limit+1 characters long. Attempt to DO or GOTO it and ensure that an "M56" error occurs.

4.4 Legal Considerations

None known

5. Closely Related Standards Activities

5.1 Other X11 Proposals Under Consideration

Lower-case characters in names proposal.

5.2 Other Related Standards Efforts

None known.

5.3 Recommendations for Coordinating Liaison

None.

6. Associated Documents

None.

7. Issues, Pros and Cons, and Discussion

October 1995 MDC meeting

Pro	Con
1. Users want this	
2. Has been implemented	

June 1995 MDC meeting

Pro	Con
1. Allows more readable names	1. Implementation issues not explicit.
2. Not backwards incompatible with standard programs	2. Conflict in standard may lead to perception of backward incompatibility.
3. Corrects a conflict in the standard	3. Is backwards incompatible
4. Users want this	

Re con two, this proposal is *removing* the conflict in the standard. Re con three, as has been mentioned several times before, this is backward incompatible only to code **NOT** written to the standard which has names that exceed 31 characters and/or use inconsistent spelling beyond the eighth.

January 1995 MDC meeting

Pro	Con
1. Allows more readable names	1. Breaks and slows existing programs due to $S limitations.
2. Not backwards incompatible with standard programs	2. Breaks existing programs that use variable names > 8 characters
3. Corrects a conflict in the standard	3. Restricts variables to upper case

June 1994 MDC meeting

No formal vote was taken when this proposal was submitted because it was withdrawn. However the discussions/debates helped to formulate this current document.

October 1993 MDC meeting

Pro	Con
1. Allows more readable names	1. Allows maximum name length less than eight
2. Not backwards incompatible with standard programs	2. Now maximum name length, wasn't before
3. Corrects a conflict in the standard	3. Backwards incompatible

Con one is simply not true! Section 3.1 has been enhanced to point out the requirement that an implementation must meet or exceed the limits in the Portability Section.

Con two is complaining about one of the specific intentions of this proposal which is to allow us to further increase the limit in the future without concern about backwards incompatibility.

Con three is true only for non-standard code that exceeds the current portability limit with inconsistent spellings. It's quite possible that this will uncover some subtle bugs that have been lurking around for years in existing non-standard code.