X11/98-19

MUMPS Development Committee

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

User-defined ssvns

June 28, 1998

Produced by the MDC Subcommittee #13
Data Management and Manipulation

Art Smith, Chairman
MUMPS Development Committee

Dan Bormann, Chairman
Subcommittee #13

1. IDENTIFICATION

1.1 Title

User-defined ssvns

1.2 MDC proposer and sponsor

Proposer:	SC12
Author:	Jon Diamond, Diamond Consulting Ltd

1.3 Motion

~~Proposed as MDC Type A superceding X11/SC13/98-2~~ NONE

1.4 History of MDC actions

Date	*Doc#*	*Action*
Jul 98	This document	None - final approved version.
Jun 98	X11/~~98-?~~ SC13/98-7	Approved as MDC Type A (9:3:7)
Mar 98	X11/SC13/98-2	Approved as SC13 Type A (5:2:10)
Nov 97	X11/SC13/97-8	Proposed as MDC Type A – failed (6:11:9)
Mar 97	X11/SC13/TG16/96-5	Approved as SC13 Type A (15:0:6)
Oct 96	X11/SC13/TG16/96-3	Proposed as SC13 Type A - failed (8:12:6)
Mar 96	X11/SC13/TG16/96-1	Proposed to demote to SC13 Type C (3:12:5)
Dec 95	X11/SC12/TG16/95-5	Discussed in MDCC-E
Oct 95	X11/SC13/TG16/95-4	Discussed in SC13/TG6
Jun 95	X11/SC13/TG16/95-2 also numbered as X11/SC15/95-8	Approved as SC13 Type B (19:0:4)
Jan 95	X11/SC15/95-1	Proposed as SC15 Type C - remanded to Task group
Dec 94	Discussion on Newsnet	

1.5 Dependencies

Two argument form of $QUERY, unless already elevated to MDC Type A.

2. JUSTIFICATION

2.1 Needs

See Annex for suggestion.

2.2 Existing practice in the area of the proposed change

Micronetics have a similar implementation used for other ssvns.

An OO type solution might be considered to be equivalent functionality. However it seems unlikely that any OO solution would preserve the M type datastructures which are involved, so for example copying a global sub-tree would not seem to be an easy possibility as it is here:

```
S ^A("A",5)=^$YA(5)
```

3. DESCRIPTION

3.1 General description of the proposed change

The proposal allocates a range of ssvns to users, in a similar fashion to that from some other aspects to the language.

In addition to that it proposes a mechanism whereby references to these ssvns are then mapped into subroutine/function calls, in a similar fashion to that of portable controlmnemonics.

3.2 Annotated examples of use

When ^%XYZ has been made the routine associated with ^$YJOB then

Example	*Equivalent to*
`S ^$YJOB(12,123)=12`	`D %SET^%XYZ("^$YJOB(12,123)",12)`
`W $ORDER(^$YJOB(""),-1)`	`W $$%ORDER^%XYZ("^$YJOB("""")",-1)`
`S a=B+^$YJOB(1)`	`S a=B+$$%VALUE^%XYZ("^$YJOB(1)")`

3.3 Formalization

Add to the list of ssvns in 7.1.3

| ^$Y [<u>unspecified</u>] |

Add a new clause

7.1.3.x ^$Y [<u>unspecified</u>] structured system variables

These <u>ssvns</u> are reserved for users and are called user-defined structured system variables. The syntax is

^ $Y <u>name</u> [(<u>L</u> <u>expr</u>)]

An implementation is required to provide a means of associating a <u>routine</u> with one or more specific user-defined structured system variables.

This <u>routine</u> is called whenever a reference is made to a user-defined structured system variable by calling one of the following <u>labels</u> in the <u>routine</u> with first parameter being $NAME of the reference and the second parameter, if any, as below:

Reference Type	Label	Second parameter	Result?
Evaluation	%VALUE		Yes
$DATA argument	%DATA		Yes
$GET argument	%GET	Second argument of $GET	Yes
$ORDER argument	%ORDER	Second argument of $ORDER	Yes
$QUERY argument	%QUERY	Second argument of $QUERY	Yes
KILL command	%KILL		

KSUBSCRIPTS command	%KSUBSCRIPTS	
KVALUE command	%KVALUE	
MERGE command target	%MERGE	Source glvn
MERGE command source	%MERGES	Target glvn
Value assignment	%SET	Value to be assigned.

The usage of $ORDER and $QUERY with unsubscripted user-defined structured system variables has the same effect as if they were not user-defined.

If user-defined structured system variables are both the source and target of a MERGE command then only the MERGE label for the target ssvn is called.

If no such routine exists when a reference is made to a user-defined structured system variable then an error condition occurs with ecode = M97. If no such label exists when a reference is made to a user-defined structured system variable then an error condition occurs with ecode = M13.

Note: names which differ only in the use of corresponding upper and lower case letters are NOT equivalent.

Note: Users providing such routines are responsible for ensuring that any other side-effects (such as a change to $TEST or $DATA values which are not 0, 1, 10 or 11), which would not have taken place had the reference been to a global variable do not occur as a result of calling the routine.

4. IMPLEMENTATION IMPACTS

4.1 Impact on users existing practice and investments

None on existing code. However this additional functionality could be used as a means of easy encapsulation of such code, as well as allowing better techniques for data and interface manipulation.

4.2 Impact on existing vendor practices and investments

This is potentially significant, however many implementations already provide similar facilities for their own implementation of certain ssvns. Thus it is anticipated that this would not be a major burden on implementors. At least one vendor has implemented a very similar scheme.

4.3 Techniques and costs for compliance verification

Compliance verification for the most part would be as per normal for any additional language functionality. However, associating the routine with the ssvn would be an implementation specific issue.

4.4 Legal considerations

None known.

5. CLOSELY RELATED STANDARDS ACTIVITIES

Portable controlmnemonics and User-definable I/O handling.

6. ASSOCIATED DOCUMENTS

None

7. DISCUSSION

June 98
Pro:
1) Desired Functionality. 2) User requested functionality
Con:
1) Not needed (already accomplished by current standard).

March 98
Pro:
1) Desired Functionality. 2) User requested functionality
Con:
1) Not needed (already accomplished by current standard).

November 97
The |env| concept has been removed, since issues of where the routine should be executed need further discussion. (This can always be added later.)
Pros
1) User requested functionality 2) Environment not specified 3) Functional approach fits the current standard
Cons
1) Environment not specified - **HAS BEEN REMOVED IN THIS REVISION.**
2) Unclear formalisation - **HAS BEEN CLARIFIED IN RESPECT OF ENVIRONMENT.**
3) Should specify association method: ssvn to routine -**THIS METHOD IS ALSO NOT SPECIFIED FOR SUCH THINGS AS LIBRARY ROUTINES ETC. It should probably be better left to the implementors to devise optimal schemes for their systems.**
4) Limited usefulness

March 97
Pros
1) User requested functionality 2) Similar functionality has been implemented. 3) Ensures consistency
Cons

Oct 96
Pros
1) Clean way to facilitate OO. 2) Facilitates things not OO. 3) Similar functionality has been implemented. 4) Requested functionality.
Cons
1) True encapsulation can always be broken in MUMPS. 2) Lacks environment syntax. 3) Unnecessary. 4) Case ambiguity.

Mar 96

Discussion that objects would be a better method of achieving this capability. No changes made to proposal as a result of discussion. **Note that this solution preserves the existing MUMPS data structures which it is extremely unlikely any OO proposal would do.**

PROS (FOR DEMOTION):
Good example if OO, More powerful alternative being considered
CONS:
Simple and cleanest functionality, Here now, Not a pig in a poke, Best way to implement OO, User interest, X11.1 enhancement

Oct 95
Minor typographical and format errors identified, which prevented TG recommending elevation to SC type A. No technical issues raised.

Jun 95
Separation of KILL types of commands changed to reference different labels. Use of existing ecode recommended. Use of source or target calling with MERGE and *both* the source and target being user-defined ssvns was discussed. There was a slight preference for the current formulation, based on what was believed to be existing Micronetics practice for its implementation of other ssvns, although it was appreciated that there were difficulties with either choice.
PROs: Enables Object Orinted Programming, Enables programming in an independent elegant M like manner, Similar to existing implementation, Uses standard approach used elsewhere in X11.
CONs: None

Jan 95
$ORDER unsubscripted effect not clear - now acts as normal ssvns
Needs to take care of MERGE in/out separately
Very similar to Micronetics implementation of ssvn

ANNEX

```
Date: Sat, 10 Dec 1994 11:17:37 GMT

The SSVN Ms' new interface.
----------------------------
The introduction of the MWAPI (M windows application programming interface) to the M language
standard brought with it the SSVN (structured system variable name). The SSVN provides the
mechanism for virtualiseing access to system resources and routines. To the M programmer the SSVN
resembles an M array and is handled using the same commands and functions as used for array
manipulation and interrogation.

The physical implementation of SSVNs is vendor specific, DSM (Digital Standard Mumps) and DTM
(Datatree Mumps) are two with which I have experimented, the following description is based on the
DSM implementation and is not compatible with DTM.

M arrays support the following functions: $Order, $Get, $Data, $Query
M arrays support the following commands: Kill, Set, Merge

When operating on SSVNs each of these actions is implemented as a call-back to a DSM routine. i.e.
the source code:-
 S ^$JOB(12,123)=12
is equivalent to
 D SET^%iJOB("^$JOB(12,123)",12)
 W $ORDER(^$JOB(""),-1)
is equivalent to
 W $$ORDER^%iJOB("^$JOB("""")",-1)

The call-back routine (%iJOB) has entry points for each action and is written in ANSI standard M.
```

So that's the background - much of which I am sure you already knew, however armed with this knowledge...

Native M is currently under attack from several fronts, one of which is SQL and ODBC access to M databases. Much of this is possible only because of Ms' flexible and open global data structures. To partially counter this and for fun I have been working on SSVN functions in the following form :-

All access to ^$JOB($J,filename) is mapped on to the named RMS file which may be in DBase II or DBase III format:-

$G(^$JOB($J,filename)) ; returns a delimited list of the field names and types as defined in the file header (the first field is always 'Deleted Flag',"C",1)

$G(^$JOB($J,filename, n)) ; returns the n'th record from the file

$O(^$JOB($J,filename,n)) ; will return subsequent record numbers

M ^XW($J)=^$JOB($J,filename) ; will copy the complete data file on the named global

M ^$JOB($J,filename)=global_ref ; will create a DBase file from the named global.

User written call-backs are not (yet?) available in DTM, however being PC based the above technique would be more usefully implemented in DTM. Also, I have used the $JOB SSVN for this experiment as although I have found the call-back routines for the vendor supplied SSVNs I dont know how to define new SSVNs (or indeed if it is currently possible) under DSM.

This technique is obviously not limited to accessing DBase files, an SSVN node could just as easily be mapped on to an SQL query or even to an ODBC 'engine' - giving your M applications and M programmers access to almost any data format with all the benefits this PC data integration would entail.

Finally, user implementable SSVNs could also be used for 'virtual' access to native globals as in 'S ^$PERSON(ident)="Mr/Tommy/Atkins/123 The Drive/The Town/UK'. calling...

```
PERSON
SET(REF,VAL) N id
 S id=$QS(REF,1)
 S ^ABC(1,id)=VAL ; Main Record
 S ^ABC(2,$P(VAL,"/",3),id) ; Index by Surname
 S ^ABC(3,$P(VAL,"/",6),id) ; Index by Country of Residence
 Q
```

Giving an additional layer of robustness to updates.

So any thoughts on User Definable SSVNs - how about the next standard :-)

RGDS to those who read this far !

Paul Perrin /)/+)