

The FreeM Manual

THE OFFICIAL MANUAL OF FREEM
Version 0.1.6

John P. Willis

This manual is for FreeM, (version 0.1.6), which is a free and open-source implementation of the MUMPS programming language and database system.

Copyright © 2020 Coherent Logic Development LLC

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

Table of Contents

Introduction	1
Production Readiness	1
Contributors	1
1 FreeM Invocation	3
1.1 Synopsis	3
1.2 Command-Line Options	3
2 The FreeM REPL	4
3 Intrinsic Special Variables	5
3.1 \$DEVICE	5
3.2 \$ECODE	5
3.3 \$ESTACK	5
3.4 \$ETRAP	5
3.5 \$HOROLOG	5
3.6 \$IO	5
3.7 \$JOB	5
3.8 \$KEY	5
3.9 \$PRINCIPAL	6
3.10 \$QUIT	6
3.11 \$STACK	6
3.12 \$STORAGE	6
3.13 \$SYSTEM	6
3.14 \$TEST	6
3.15 \$TLEVEL	6
3.16 \$TRESTART	6
3.17 \$X	6
3.18 \$Y	6
3.19 \$ZA	7
3.20 \$ZB	7
3.21 \$ZCONTROL	7
3.22 \$ZDATE	7
3.23 \$ZERROR	7
3.24 \$ZF	7
3.25 \$ZHOROLOG	7
3.26 \$ZINRPT	7
3.27 \$ZJOB	7
3.28 \$ZLOCAL	7
3.29 \$ZMATCHCONTROL	7
3.30 \$ZMATCHNUMERIC	7
3.31 \$ZMATCHPUNCTUATION	7

3.32	\$ZMATCHALPHABETIC	7
3.33	\$ZMATCHLOWERCASE	7
3.34	\$ZMATCHUPPERCASE	7
3.35	\$ZMATCHEVERYTHING	7
3.36	\$ZNAME	7
3.37	\$ZORDER	7
3.38	\$ZPRECISION	7
3.39	\$ZREFERENCE	8
3.40	\$ZSYSTEM	8
3.41	\$ZTIME	8
3.42	\$ZTRAP	8
3.43	\$ZVERSION	8
4	Intrinsic Functions	9
4.1	\$ASCII	9
4.2	\$CHAR	9
4.3	\$DATA	9
4.4	\$EXTRACT	9
4.5	\$FIND	9
4.6	\$FNUMBER	9
4.7	\$GET	9
4.8	\$JUSTIFY	9
4.9	\$LENGTH	9
4.10	\$NAME	9
4.11	\$NEXT	9
4.12	\$ORDER	9
4.13	\$PIECE	10
4.14	\$QLength	10
4.15	\$QSUBSCRIPT	10
4.16	\$QUERY	10
4.17	\$RANDOM	10
4.18	\$REVERSE	10
4.19	\$SELECT	10
4.20	\$STACK	10
4.21	\$TEXT	10
4.22	\$TRANSLATE	10
4.23	\$VIEW	10
4.24	\$ZBOOLEAN	10
4.25	\$ZCALL	10
4.26	\$ZCR	10
4.27	\$ZCRC	10
4.28	\$ZDATA	10
4.29	\$ZDATE	10
4.30	\$ZEDIT	10
4.31	\$ZHOROLOG	10
4.32	\$ZHT	10
4.33	\$ZKEY	11
4.34	\$ZLENGTH	11

4.35	\$ZLSD	11
4.36	\$ZM	11
4.37	\$ZNAME	11
4.38	\$ZNEXT	11
4.39	\$ZORDER	11
4.40	\$ZPIECE	11
4.41	\$ZPREVIOUS	11
4.42	\$ZREPLACE	11
4.43	\$ZSYNTAX	11
4.44	\$ZSORT	11
4.45	\$ZTIME	11
4.46	\$ZZIP	11
5	Commands	12
5.1	BREAK	12
5.2	CLOSE	12
5.3	DO	12
5.4	ELSE	12
5.5	FOR	12
5.6	GOTO	12
5.7	HALT	12
5.8	HANG	12
5.9	IF	12
5.10	JOB	12
5.11	KILL	12
5.12	KSUBSCRIPTS	12
5.13	KVALUE	12
5.14	LOCK	12
5.15	MERGE	12
5.16	NEW	12
5.17	OPEN	12
5.18	QUIT	12
5.19	READ	12
5.20	SET	13
5.21	TCOMMIT	13
5.22	TRESTART	13
5.23	TROLLBACK	13
5.24	USE	13
5.25	VIEW	13
5.26	WRITE	13
5.27	XECUTE	13
5.28	ZALLOCATE	13
5.29	ZBREAK	13
5.30	ZDEALLOCATE	13
5.31	ZGO	13
5.32	ZHALT	13
5.33	ZINSERT	13
5.34	ZJOB	13

5.35	ZLOAD	13
5.36	ZNEW	13
5.37	ZPRINT	13
5.38	ZQUIT	13
5.39	ZREMOVE	13
5.40	ZSAVE	14
5.41	ZTRAP	14
5.42	ZWRITE	14
6	Structured System Variables	15
6.1	^\$CHARACTER	15
6.2	^\$DEVICE	15
6.3	^\$DISPLAY	15
6.4	^\$EVENT	15
6.5	^\$GLOBAL	15
6.6	^\$JOB	15
6.7	^\$LOCK	15
6.8	^\$PDISPLAY	15
6.9	^\$ROUTINE	15
6.10	^\$SYSTEM	15
6.11	^\$WINDOW	15
7	Operators	16
8	User-Defined Z Functions	17
9	User-Defined SSVNs	18
10	System Library Routines	19
10.1	^%ED	19
10.2	^%FLIST	19
10.3	^%FUTIL	19
10.4	^%GD	19
10.5	^%GL	19
10.6	^%KEY	19
10.7	^%KEYINT	19
10.8	^%KEYTEST	19
10.9	^%KILLJOB	19
10.10	^%LOGIN	19
10.11	^%MEN	19
10.12	^%MESS	19
10.13	^%MUTIL	19
10.14	^%N	19
10.15	^%RD	19
10.16	^%RDEL	19
10.17	^%SS	19

10.18	~%SYS	19
10.19	~%SYSDEV	19
10.20	~%SYSGEN	20
10.21	~%SYSINFO	20
10.22	~%SYSNSP	20
10.23	~%SYSNSP0	20
10.24	~%T	20
10.25	~%u	20
10.26	~%uflis	20
10.27	~%ufxxx	20
10.28	~%ug	20
10.29	~%ugdir	20
10.30	~%ugexp	20
10.31	~%ugimp	20
10.32	~%uglis	20
10.33	~%ugxxx	20
10.34	~%ulmath	20
10.35	~%ulstring	20
10.36	~%ur	20
10.37	~%urdel	20
10.38	~%urdir	20
10.39	~%uredi	20
10.40	~%urexp	21
10.41	~%urimp	21
10.42	~%ursea	21
10.43	~%urxxx	21
10.44	~%uxdat	21
10.45	~%uxxxx	21
10.46	~%ZCOLUMNS	21
10.47	~%ZOS	21
10.48	~%ZROWS	21
10.49	~%UTILITY	21
11	Namespace Management	22
11.1	Namespace Management Overview	22
11.2	Listing Namespaces	22
11.3	Switching Namespaces	23
11.4	Adding New Namespaces	23
11.5	Returning to the Default Namespace	24
12	Configuration and SYSGEN	25

13	Accessing FreeM from C Programs	26
13.1	mllib.h and mref_t Data Structure.....	26
13.2	Initializing mlib API	26
13.3	m_init_mref()	26
13.4	m_add_subscript()	26
13.5	m_version()	26
13.6	m_set()	26
13.7	m_get()	26
13.8	m_kill()	26
13.9	m_kvalue()	26
13.10	m_ksubscripts()	26
13.11	m_data()	26
13.12	m_order()	26
13.13	m_query()	26
13.14	m_lock()	26
13.15	m_unlock()	26
13.16	m_str_function()	26
13.17	m_num_function()	26
13.18	m_procedure()	26
Appendix A	FreeM Utilities	27
A.1	Global Compactor (gcompact)	27
A.2	Block Examiner (gfix)	27
A.3	Global Lister (gl)	27
A.4	Lock Examiner (glocks)	27
A.5	Global Repair Tool (grestore)	27
A.6	Global Validator (gverify)	27
Appendix B	FreeM VIEW	
	Commands and Functions	28
Appendix C	Implementation Limits	29
Appendix D	US-ASCII Character Set	30
Appendix E	FreeM Project Coding Standards ..	31
E.1	Module Headers	31
E.2	Variable Naming	31
E.3	Indentation and General Layout	31
E.4	Brace Placement (Functions)	32
E.5	Brace Placement (if-for-while-do)	32
E.6	Labels and goto	33
E.7	Preprocessor Conditionals	33
E.8	coding standards, preprocessor conditionals	33
E.9	Overall Program Spacing	33

E.10	The switch() Statement	33
E.11	Comments	34
Appendix F Conformance Clause		35
Index		36

Introduction

FreeM started its life as *FreeMUMPS*, written for MS-DOS and ported to Linux and SCO UNIX by a mysterious individual going by the name of "Shalom ha-Ashkenaz". It was released to MUG (MUMPS User Group) Deutschland in 1998, and maintenance was taken over by the GUMP (Generic Universal M Project) thereafter, which changed its name first to PSM (Public Standard MUMPS) and then by popular request to FreeM.

When GT.M was open-sourced in late 1999, FreeM and GUMP were essentially abandoned. L.D. Landis, the owner of the original GUMP SourceForge project, and one of FreeM's significant contributors, passed maintenance of FreeM and ownership of its SourceForge project to John Willis in 2014. At this point, FreeM would not compile or run on modern Linux systems, so steps were taken to remedy the most pressing issues in the codebase. Limitations on the terminal size (previously hard-coded to 80x25) were lifted, and new `$VIEW` functions were added to retrieve the terminal size information. `$X` and `$Y` intrinsic special variables were updated to support arbitrary terminal sizes, and FreeM was once again able to build and run.

In February of 2020, work began in earnest to build a development and support infrastructure for FreeM and begin the careful process of refining it into a more stable and robust product.

Production Readiness

FreeM is not yet production-ready. There are several show-stopping bugs that preclude a general release for public use:

- The use of `NEW` and argumentless `DO`, only supported when `-DNEWSTACK` is passed to the C compiler, causes occasional segmentation faults that have not been isolated, as they don't appear when running under the control of symbolic debuggers such as `gdb`, or leak testers such as `valgrind`.
- The common idiom of a `$ORDER` loop is broken, as `QUIT:SUB=""` (or similar) is not properly honored, resulting in infinite loops and incorrect output.
- `MERGE` is not implemented. Code exists in `symtab.c` (responsible for handling local variables) to support the `MERGEing` of local variables, but it causes segmentation faults as well.
- SSVNs (Structured System Variables) are not fully implemented, though skeletal support exists in the FreeM source code (`ssvn*.c`).
- `VIEW` commands and `$VIEW` functions are used extensively to configure and inspect the run-time behavior of FreeM, rather than the "canonical" SSVN-based approach.

Per the conditions under which Shalom ha-Ashkenaz originally released the *FreeMUMPS* implementation to the community, a full and supported public release of the software is not permitted until the above issues are resolved, and the MUG (MUMPS User Group) organization (or its successors) certify its quality and suitability for purpose.

Contributors

Current contributors denoted with a + following their name and role.

- Shalom ha-Ashkenaz (Original Implementer)

- Jon Diamond (Library, Utilities, Conformance)
- Winfried Gerum (Code, Advice, MTA coordination)
- Greg Kreis (Hardhats coordination, Dependencies)
- Larry Landis (Coordination, Code, Documentation)
- Frederick D.S. Marshall (MDC Standards Conformance) +
- Lloyd Milligan (Code, Testing, Documentation)
- Steve Morris (Code, Microsoft)
- John Murray (Code, Conformance)
- Wilhelm Pastoors (Testing, Documentation)
- Kate Schell (Coordination, Conformance, MTA, MDC, Advice)
- Lyle Schofield (Advice, Prioritization, Tracking, Project Management)
- Axel Trocha (Code, Utilities)
- Dick Walters (Project Lead, Chief Coordinator, MTA)
- David Whitten (QA Test Suite, MDC, Advice) +
- David Wicksell (Debugging, Code, Testing) +
- John Willis (Current Maintainer and Project Lead) +
- Steve Zeck (Code)

1 FreeM Invocation

1.1 Synopsis

```
$ ./mumps [OPTIONS...] [[-r <entryref>] | [--routine=<entryref>]]
```

When FreeM loads, it searches the system library in `../mlib/` for the `^%SYS` routine. When `^%SYS` runs, it searches for a default namespace, activates it, configures the programmer-mode command prompt, prints a welcome message, and presents the programmer-mode prompt.

If no default namespace is configured, FreeM will prompt you to create a new namespace.

When `-r` or `--routine` are passed on the command line, FreeM will load and run the specified routine instead of `^%SYS`, which means that routines launched in this manner will need to do their own namespace configuration using `VIEW` commands prior to accessing any routines or globals existing outside of the system library namespace.

1.2 Command-Line Options

`-h, --hardcopy`

Enables hardcopy mode, echoing all output to a disk file. By default, this disk file is `/usr/tmp/hardcopy`, but can be changed with the following command:

```
USER> VIEW 13:"</path/to/hardcopy/file>"
```

`-f, --filter`

Allows your MUMPS routines to be used as UNIX filters.

`-n, --noclear`

Disables automatic screen clearing when FreeM loads.

`-s, --standard`

Restricts the use of non-standard language features, including `$Z...` intrinsic special variables, `$Z...` intrinsic functions, `Z...` commands, as well as `VIEW` and `$VIEW`.

`-i, --import`

Causes your UNIX environment variables to be imported into FreeM's local symbol table.

`-r <entryref>, --routine=<entryref>`

Causes `<entryref>` to be executed at load, instead of `^%SYS`.

2 The FreeM REPL

3 Intrinsic Special Variables

3.1 \$DEVICE

Returns the status of the device currently in use, and is writable.

If \$DEVICE returns *1*, an error condition exists on the current device.

3.2 \$ECODE

Returns a comma-delimited list of error conditions currently present, and is writable. An empty \$ECODE indicates no errors.

FreeM currently sets \$ECODE only for errors established by the MDC (MUMPS Development Committee).

Setting \$ECODE has no effect, as FreeM does not currently implement standard error-handling with \$ETRAP.

3.3 \$ESTACK

Returns an empty string, as FreeM does not currently implement standard error-handling with \$ETRAP.

3.4 \$ETRAP

Returns an empty string and is writable, but unused as FreeM does not currently implement standard error-handling with \$ETRAP.

3.5 \$HOROLOGY

Returns a string containing the current date and time as *<days>*,*<seconds>*, where *<days>* represents the number of days since the MUMPS epoch (midnight on 31 December 1840), and *<seconds>* represents the number of seconds since the most recent midnight.

Non-Standard Behavior

In FreeM, \$HOROLOGY is writable, and will set the current system time as long as the user has sufficient permissions to do so.

3.6 \$IO

Represents the current input/output device. Read-only.

3.7 \$JOB

Represents the process ID of the FreeM instance currently in use.

3.8 \$KEY

Represents the sequence of control characters that terminated the last READ command on \$IO.

3.9 \$PRINCIPAL

Represents the primary input/output device. Usually a terminal or virtual terminal.

3.10 \$QUIT

If the current execution context was invoked as an extrinsic function, `$QUIT` returns `1`. Otherwise, returns `0`.

When `$QUIT` returns `1`, a subsequent `QUIT` command must have an argument.

3.11 \$STACK

Represents the current stack level.

3.12 \$STORAGE

Represents the number of bytes of free space available in FreeM's heap.

3.13 \$SYSTEM

Returns the MDC system ID of FreeM.

3.14 \$TEST

`$TEST` is a writable ISV that is `1` if the most recent conditional expression returned *true*. Otherwise, returns *false*.

`$TEST` is implicitly `NEWed` when entering a new stack frame.

3.15 \$TLEVEL

Returns an empty string, as FreeM does not currently implement transaction processing.

3.16 \$TRESTART

Returns an empty string, as FreeM does not currently implement transaction processing.

3.17 \$X

Represents the current column position of the FreeM cursor. In FreeM, setting

Non-Standard Behavior

In FreeM, setting `$X` will move the FreeM cursor.

3.18 \$Y

Represents the current row position of the FreeM cursor.

Non-Standard Behavior

In FreeM, setting `$Y` will move the FreeM cursor.

3.19 \$ZA

3.20 \$ZB

3.21 \$ZCONTROL

3.22 \$ZDATE

3.23 \$ZERROR

3.24 \$ZF

3.25 \$ZHOROLOG

3.26 \$ZINRPT

3.27 \$ZJOB

3.28 \$ZLOCAL

3.29 \$ZMATCHCONTROL

3.30 \$ZMATCHNUMERIC

3.31 \$ZMATCHPUNCTUATION

3.32 \$ZMATCHALPHABETIC

3.33 \$ZMATCHLOWERCASE

3.34 \$ZMATCHUPPERCASE

3.35 \$ZMATCHEVERYTHING

3.36 \$ZNAME

3.37 \$ZORDER

3.38 \$ZPRECISION

3.39 \$ZREFERENCE

3.40 \$ZSYSTEM

3.41 \$ZTIME

3.42 \$ZTRAP

3.43 \$ZVERSION

4 Intrinsic Functions

4.1 \$ASCII

Returns the ASCII code (in decimal) for one character in a string.

```
SET RESULT=$ASCII(<string>[,<index>])
```

If <index> is not supplied, \$ASCII will return the ASCII code of the first character. Otherwise, returns the ASCII code of the character at position <index>.

4.2 \$CHAR

Returns a string of characters corresponding to a list of ASCII codes.

```
SET RESULT=$CHAR(<ascii-code>[,<ascii-code>,...])
```

4.3 \$DATA

Returns a numeric value 0, 1, 10, or 11, depending on whether a referenced node is defined, has data, or has children:

```
SET RESULT=$DATA(<node>)
```

The return values are as follows:

```
0: <node> is undefined
1: <node> has data but no children
10: <node> has children but no data
11: <node> has children and data
```

4.4 \$EXTRACT

4.5 \$FIND

4.6 \$FNUMBER

4.7 \$GET

4.8 \$JUSTIFY

4.9 \$LENGTH

4.10 \$NAME

4.11 \$NEXT

4.12 \$ORDER

4.13 \$PIECE

4.14 \$QLENGTH

4.15 \$QSUBSCRIPT

4.16 \$QUERY

4.17 \$RANDOM

4.18 \$REVERSE

4.19 \$SELECT

4.20 \$STACK

4.21 \$TEXT

4.22 \$TRANSLATE

4.23 \$VIEW

4.24 \$ZBOOLEAN

4.25 \$ZCALL

4.26 \$ZCR

4.27 \$ZCRC

4.28 \$ZDATA

4.29 \$ZDATE

4.30 \$ZEDIT

4.31 \$ZHOROLOG

4.32 \$ZHT

4.33 \$ZKEY

4.34 \$ZLENGTH

4.35 \$ZLSD

4.36 \$ZM

4.37 \$ZNAME

4.38 \$ZNEXT

4.39 \$ZORDER

4.40 \$ZPIECE

4.41 \$ZPREVIOUS

4.42 \$ZREPLACE

4.43 \$ZSYNTAX

4.44 \$ZSORT

4.45 \$ZTIME

4.46 \$ZZIP

5 Commands

5.1 BREAK

5.2 CLOSE

5.3 DO

5.4 ELSE

5.5 FOR

5.6 GOTO

5.7 HALT

5.8 HANG

5.9 IF

5.10 JOB

5.11 KILL

5.12 KSUBSCRIPTS

5.13 KVALUE

5.14 LOCK

5.15 MERGE

5.16 NEW

5.17 OPEN

5.18 QUIT

5.19 READ

5.20 SET

5.21 TCOMMIT

5.22 TRESTART

5.23 TROLLBACK

5.24 USE

5.25 VIEW

5.26 WRITE

5.27 XECUTE

5.28 ZALLOCATE

5.29 ZBREAK

5.30 ZDEALLOCATE

5.31 ZGO

5.32 ZHALT

5.33 ZINSERT

5.34 ZJOB

5.35 ZLOAD

5.36 ZNEW

5.37 ZPRINT

5.38 ZQUIT

5.39 ZREMOVE

5.40 ZSAVE

5.41 ZTRAP

5.42 ZWRITE

6 Structured System Variables

6.1 ^\$CHARACTER

6.2 ^\$DEVICE

6.3 ^\$DISPLAY

6.4 ^\$EVENT

6.5 ^\$GLOBAL

6.6 ^\$JOB

6.7 ^\$LOCK

6.8 ^\$PDISPLAY

6.9 ^\$ROUTINE

6.10 ^\$SYSTEM

6.11 ^\$WINDOW

7 Operators

8 User-Defined Z Functions

9 User-Defined SSVNs

10 System Library Routines

10.1 ^%ED

10.2 ^%FLIST

10.3 ^%FUTIL

10.4 ^%GD

10.5 ^%GL

10.6 ^%KEY

10.7 ^%KEYINT

10.8 ^%KEYTEST

10.9 ^%KILLJOB

10.10 ^%LOGIN

10.11 ^%MEN

10.12 ^%MESS

10.13 ^%MUTIL

10.14 ^%N

10.15 ^%RD

10.16 ^%RDEL

10.17 ^%SS

10.18 ^%SYS

10.19 ^%SYSDEV

10.20 `^%SYSGEN`

10.21 `^%SYSINFO`

10.22 `^%SYSNSP`

10.23 `^%SYSNSP0`

10.24 `^%T`

10.25 `^%u`

10.26 `^%uflis`

10.27 `^%ufxxx`

10.28 `^%ug`

10.29 `^%ugdir`

10.30 `^%ugexp`

10.31 `^%ugimp`

10.32 `^%uglis`

10.33 `^%ugxxx`

10.34 `^%ulmath`

10.35 `^%ulstring`

10.36 `^%ur`

10.37 `^%urdel`

10.38 `^%urdir`

10.39 `^%uredi`

10.40 `^%urexp`

10.41 `^%urimp`

10.42 `^%ursea`

10.43 `^%urxxx`

10.44 `^%uxdat`

10.45 `^%uxxxx`

10.46 `^%ZCOLUMNS`

10.47 `^%ZOS`

10.48 `^%ZROWS`

10.49 `^%UTILITY`

11 Namespace Management

11.1 Namespace Management Overview

The namespace utilities (provided by %SYSNSP.m, %SYSNSPO.m, and %SYSGEN.m in \$DIST_ROOT/mlib) are used for the management of namespaces. A *namespace* is a particular collection of FreeM globals and routines. If you are accustomed to other MUMPS implementations, you might have seen this concept referred to as a *UCI* or *global directory*.

FreeM has something of a *system library* namespace, located in \$DIST_ROOT/mlib which contains the "percent" globals and routines (globals and routines whose names begin with the percent sign %). Routines and globals beginning with the percent sign character must be contained in \$DIST_ROOT/mlib in the current version of FreeM. However, these routines and globals are always accessible, regardless of the currently-active namespace.

Any user-defined namespace may only access globals and routines residing either in the current namespace, or in the system library namespace. For instance, if you have two user-defined namespaces, USER1 and USER2, routines in the USER1 namespace may not access globals in the USER2 namespace, but may access globals in the system library namespace.

At least one user-defined namespace is required for proper FreeM operation, and FreeM will automatically ask you to create one if one does not already exist. The first user-defined namespace you create is automatically chosen as the startup namespace when FreeM first runs. However, the FreeM programmer may easily switch to other namespaces using the utility entry points contained in the ^%SYSNSP routine.

All namespaces aside from the system library namespace are located in \$DIST_ROOT/namespace, and are laid out as follows:

```
+---namespace      namespace directory
|   |
|   +---<name>      name of the namespace (can have more than one)
|       |
|       +---routine  namespace routines
|       |
|       +---global   namespace globals
```

11.2 Listing Namespaces

From the FreeM REPL, enter the following commands:

```
USER> d show^%SYSNSP
```

```
index      namespace
-----
1    *USER
2    MVTs
```

In the above display, the USER namespace is the default, denoted by the asterisk character preceding its name. The index column is of special importance, as it is the index value (and not the name) that is used when switching namespaces.

11.3 Switching Namespaces

To switch namespaces, use the `show^%SYSNSP` entry point documented above to determine the index of the desired namespace, and then call `switch^%SYSNSP`, passing the index as the sole argument.

For example, to switch from the `MVTS` namespace (index 2) to the `USER` namespace, with an index of 1:

```
MVTS> S RESULT=$$switch^%SYSNSP(1)
```

```
USER>
```

Note that the programmer mode prompt of the FreeM REPL always¹ indicates the name of the currently-selected namespace.

11.4 Adding New Namespaces

To add a new namespace, you will use the `^%SYSGEN` routine, which is user-friendly and menu-driven.

In this example, we will add a new namespace, called `TESTING`:

```
USER> D ^%SYSGEN
```

```
FreeM SYSTEM SETUP UTILITY
2020/02/22 20:28:51
```

```
Main Menu
```

```
-----
```

- 1 - Namespace Settings
- 2 - Routine-Editor Settings
- 3 - Configuration Settings
- 4 - Startup Options

```
Your choice: 1
```

```
Namespace Settings
```

```
-----
```

- 1 - List Namespaces

¹ This is not entirely true. `VIEW 200:<mumps-expression>` can be used to adjust the programmer-mode prompt to match the preferences of specific users and/or sites.


```
2 - add Namespace
3 - delete Namespace
4 - rename Namespace
5 - repair Namespace
```

```
Your choice: 2
```

```
Name .....: TESTING
Ok to create? <Y> y
Done.
```

11.5 Returning to the Default Namespace

To switch to the default namespace, enter the following command in the FreeM REPL:

```
TESTING> S RESULT=$$switch~%SYSNSP($$dflt~%SYSNSP))
```

```
USER>
```

Note that the programmer-mode prompt has changed to `USER>` to reflect the namespace change.

12 Configuration and SYSGEN

13 Accessing FreeM from C Programs

13.1 mlib.h and mref_t Data Structure

13.2 Initializing mlib API

13.3 m_init_mref()

13.4 m_add_subscript()

13.5 m_version()

13.6 m_set()

13.7 m_get()

13.8 m_kill()

13.9 m_kvalue()

13.10 m_ksubscripts()

13.11 m_data()

13.12 m_order()

13.13 m_query()

13.14 m_lock()

13.15 m_unlock()

13.16 m_str_function()

13.17 m_num_function()

13.18 m_procedure()

Appendix A FreeM Utilities

A.1 Global Compactor (gcompact)

A.2 Block Examiner (gfix)

A.3 Global Lister (gl)

A.4 Lock Examiner (glocks)

A.5 Global Repair Tool (grestore)

A.6 Global Validator (gverify)

Appendix B FreeM VIEW Commands and Functions

Appendix C Implementation Limits

Appendix D US-ASCII Character Set

Code	Character
000	<NUL>
001	<SOH>
002	<STX>
003	<ETX>
004	<EOT>
005	<ENQ>
006	<ACK>
007	<BEL>
008	<BS>
009	<HT>
010	<LF>
011	<VT>
012	<FF>
013	<CR>
014	<SO>
015	<SI>
016	<DLE>
017	<DC1>
018	<DC2>
019	<DC3>
020	<DC4>
021	<NAK>
022	<SYN>
023	<ETB>
024	<CAN>
025	
026	<SUB>
027	<ESC>
028	<FS>
029	<GS>
030	<RS>
031	<US>
032	<space>
033	!
034	"
035	#

Appendix E FreeM Project Coding Standards

E.1 Module Headers

Module headers should adhere to the following format:

```
/*
 *
 *          *
 *        * *
 *      *   *
 *
 *      *****
 *      * *       * *
 *      *  MUMPS  *
 *      * *       * *
 *      *****
 *
 *          *   *
 *          * *
 *          *
 *
 *  mlib.h
 *  Function prototypes, structs, and macros for FreeM
 *  binding library
 *
 *
 *  Author: John P. Willis <jpw@coherent-logic.com>
 *  Copyright (C) 1998 MUG Deutschland
 *  Copyright (C) 2020 Coherent Logic Development LLC
 *
 *  Last modified: 29 February 2020
 *
 */
```

The Star of David in module headers is a convention started by Shalom ha-Ashkenaz, the unidentified original author of FreeMUMPS/FreeM. We will continue to employ it in honor of his most valuable contribution to the MUMPS community.

E.2 Variable Naming

Variables should be named in all lowercase letters, and words within them delimited by underscores, such as `my_useful_variable`. `PascalCase` and `camelCase` are not to be used in this codebase under any circumstances.

Constants defined via the C preprocessor should be in all uppercase letters, with words within them likewise delimited by underscores, such as:

```
#define MY_USEFUL_CONSTANT 1
```

E.3 Indentation and General Layout

This project uses four spaces for indentation. Tabs are not to be used under any circumstances, and all source files must use a linefeed character to delineate lines. If you are

working on a Windows machine, you must take care to follow this, as Windows will use a carriage return followed by a linefeed by default.

This project follows a modified version of what is known as the Stroustrup indentation style.

E.4 Brace Placement (Functions)

We use modern, ANSI-style function prototypes, with the type specifier on the same line as the function name. You may encounter other styles in the code, but we are transitioning to the new style as time permits.

Below is a correct example:

```
int main(int argc, char **argv, char **envp)
{

}
```

E.5 Brace Placement (if-for-while-do)

The `if` keyword should be followed by one space, then the opening paren and conditional expression. We also use Stroustrup-style `else` blocks, rather than the K&R 'cuddled' `else:`

```
if (x) {
    ...
}
else {
    ...
}

while (1) {
    ...
}

for (i = 1; i < 10; i++) {
    ...
}

do {
    ...
} while (x);
```

Single-statement if blocks should be isolated to a single line:

```
if (x) stmt();

not:
if(x)
    stmt();
```

Notice that there is a space between `if` and `(x)`, but not between `stmt` and `()`. This should be followed throughout the code.

If an `if` block has an `else if` or `else`, all parts of the construct must be bracketed, even if one or more of them contain only one statement:

```
if (x) {
    foo();
}
else if (y) {
    bar();
}
else {
    bas();
}
```

E.6 Labels and goto

Labels must begin in column 1, and have two lines of vertical space above and one beneath.

E.7 Preprocessor Conditionals

E.8 coding standards, preprocessor conditionals

I have struggled with this, but have settled upon the standard practice of keeping them in column 1.

E.9 Overall Program Spacing

- Variable declarations fall immediately beneath the opening curly brace, and should initialize the variable right there whenever initialization is used.
- One line between the last variable declaration and the first line of real code.
- The `return` statement of a function (when used as the last line of a function) should have one blank line above it and none below it.
- Really long functions (those whose entire body is longer than 24 lines) should have a comment immediately following the closing curly brace of the function, telling you what function the closing brace terminates.

E.10 The switch() Statement

We indent `case` one level beneath `switch()`, and the code within each `case` beneath the `case`. Each `case` should have one line of vertical whitespace above it:

```
switch(foo) {

    case some_const:
        foo();

        break;

    case some_other_const:
        bar();
```

```
        break;

    default:
        exit(1);

        break;
}
```

E.11 Comments

We use C-style comments (`/* comment */`) exclusively, even on single-line comments. C++ comments (`// comment`) are not permitted.

Appendix F Conformance Clause

Index

\$

\$ASCII	9
\$CHAR	9
\$DATA	9
\$DEVICE	5
\$ECODE	5
\$ESTACK	5
\$ETRAP	5
\$EXTRACT	9
\$FIND	9
\$FNUMBER	9
\$GET	9
\$HOROLOG	5
\$IO	5
\$JOB	5
\$JUSTIFY	9
\$KEY	5
\$LENGTH	9
\$NAME	9
\$NEXT	9
\$ORDER	9
\$PIECE	10
\$PRINCIPAL	6
\$QLength	10
\$QSUBSCRIPT	10
\$QUERY	10
\$QUIT	6
\$RANDOM	10
\$REVERSE	10
\$SELECT	10
\$STACK	6, 10
\$STORAGE	6
\$SYSTEM	6
\$TEST	6
\$TEXT	10
\$TLEVEL	6
\$TRANSLATE	10
\$TRESTART	6
\$VIEW	10
\$X	6
\$Y	6
\$ZA	7
\$ZB	7
\$ZBOOLEAN	10
\$ZCALL	10
\$ZCONTROL	7
\$ZCR	10
\$ZCRC	10
\$ZDATE	7, 10
\$ZEDIT	10
\$ZERROR	7
\$ZF	7
\$ZHOROLOG	7, 10
\$ZHT	10

\$ZINRPT	7
\$ZJOB	7
\$ZKEY	11
\$ZLENGTH	11
\$ZLOCAL	7
\$ZLSD	11
\$ZM	11
\$ZMATCHALPHABETIC	7
\$ZMATCHCONTROL	7
\$ZMATCHEVERYTHING	7
\$ZMATCHLOWERCASE	7
\$ZMATCHNUMERIC	7
\$ZMATCHPUNCTUATION	7
\$ZMATCHUPPERCASE	7
\$ZNAME	7, 11
\$ZNEXT	11
\$ZORDER	7, 11
\$ZPIECE	11
\$ZPRECISION	7
\$ZPREVIOUS	11
\$ZREFERENCE	8
\$ZREPLACE	11
\$ZSORT	11
\$ZSYNTAX	11
\$ZSYSTEM	8
\$ZTIME	8, 11
\$ZTRAP	8
\$ZVERSION	8
\$ZZIP	11

^

^\$CHARACTER	15
^\$DEVICE	15
^\$DISPLAY	15
^\$EVENT	15
^\$GLOBAL	15
^\$JOB	15
^\$LOCK	15
^\$PDISPLAY	15
^\$ROUTINE	15
^\$SYSTEM	15
^\$WINDOW	15
^%ED	19
^%FLIST	19
^%FUTIL	19
^%GD	19
^%GL	19
^%KEY	19
^%KEYINT	19
^%KEYTEST	19
^%KILLJOB	19
^%LOGIN	19
^%MEN	19

~%MESS	19
~%MUTIL	19
~%N	19
~%RD	19
~%RDEL	19
~%SS	19
~%SYS	19
~%SYSDEV	19
~%SYSGEN	20
~%SYSINFO	20
~%SYSNSP	20
~%SYSNSP0	20
~%T	20
~%u	20
~%uflis	20
~%ufxxx	20
~%ug	20
~%ugdir	20
~%ugexp	20
~%ugimp	20
~%uglis	20
~%ugxxx	20
~%ulmath	20
~%ulstring	20
~%ur	20
~%urdel	20
~%urdir	20
~%uredi	20
~%urexp	21
~%urimp	21
~%ursea	21
~%urxxx	21
~%UTILITY	21
~%uxdat	21
~%uxxxx	21
~%ZCOLUMNS	21
~%ZOS	21
~%ZROWS	21

B

BREAK	12
-------------	----

C

CLOSE	12
coding standards, brace placement, functions ...	32
coding standards, brace placement, if-for-while-do	32
coding standards, comments	34
coding standards, goto	33
coding standards, indentation	31
coding standards, labels	33
coding standards, layout	31
coding standards, module headers	31
coding standards, spacing of programs	33
coding standards, switch()	33
coding standards, variable naming	31

commands	12
commands, BREAK	12
commands, CLOSE	12
commands, DO	12
commands, ELSE	12
commands, FOR	12
commands, GOTO	12
commands, HALT	12
commands, HANG	12
commands, IF	12
commands, implementation-specific	13, 14
commands, JOB	12
commands, KILL	12
commands, KSUBSCRIPTS	12
commands, KVALUE	12
commands, LOCK	12
commands, MERGE	12
commands, NEW	12
commands, OPEN	12
commands, QUIT	12
commands, READ	12
commands, SET	13
commands, TCOMMIT	13
commands, TRESTART	13
commands, TROLLBACK	13
commands, unimplemented	12, 13
commands, USE	13
commands, VIEW	13
commands, WRITE	13
commands, XECUTE	13
commands, ZALLOCATE	13
commands, ZBREAK	13
commands, ZDEALLOCATE	13
commands, ZGO	13
commands, ZHALT	13
commands, ZINSERT	13
commands, ZJOB	13
commands, ZLOAD	13
commands, ZNEW	13
commands, ZPRINT	13
commands, ZQUIT	13
commands, ZREMOVE	13
commands, ZSAVE	14
commands, ZTRAP	14
commands, ZWRITE	14
configuration, using SYSGEN	26

D

Diamond, Jon	1
DO	12

E

ELSE	12
execution, interactive	4

F

FOR..... 12

G

Gerum, Winfried..... 1
GOTO..... 12

H

ha-Ashkenaz, Shalom..... 1, 31
HALT..... 12
HANG..... 12

I

IF..... 12
intrinsic functions, \$ASCII..... 9
intrinsic functions, \$CHAR..... 9
intrinsic functions, \$DATA..... 9
intrinsic functions, \$EXTRACT..... 9
intrinsic functions, \$FIND..... 9
intrinsic functions, \$FNUMBER..... 9
intrinsic functions, \$GET..... 9
intrinsic functions, \$JUSTIFY..... 9
intrinsic functions, \$LENGTH..... 9
intrinsic functions, \$NAME..... 9
intrinsic functions, \$NEXT..... 9
intrinsic functions, \$ORDER..... 9
intrinsic functions, \$PIECE..... 10
intrinsic functions, \$QLENGTH..... 10
intrinsic functions, \$QSUBSCRIPT..... 10
intrinsic functions, \$QUERY..... 10
intrinsic functions, \$RANDOM..... 10
intrinsic functions, \$REVERSE..... 10
intrinsic functions, \$SELECT..... 10
intrinsic functions, \$STACK..... 10
intrinsic functions, \$TEXT..... 10
intrinsic functions, \$TRANSLATE..... 10
intrinsic functions, \$VIEW..... 10
intrinsic functions, \$ZBOOLEAN..... 10
intrinsic functions, \$ZCALL..... 10
intrinsic functions, \$ZCR..... 10
intrinsic functions, \$ZDATE..... 10
intrinsic functions, \$ZEDIT..... 10
intrinsic functions, \$ZHOROLOG..... 10
intrinsic functions, \$ZHT..... 10
intrinsic functions, \$ZKEY..... 11
intrinsic functions, \$ZLENGTH..... 11
intrinsic functions, \$ZLSD..... 11
intrinsic functions, \$ZM..... 11
intrinsic functions, \$ZNAME..... 11
intrinsic functions, \$ZNEXT..... 11
intrinsic functions, \$ZORDER..... 11
intrinsic functions, \$ZPIECE..... 11
intrinsic functions, \$ZPREVIOUS..... 11
intrinsic functions, \$ZREPLACE..... 11

intrinsic functions, \$ZSORT..... 11
intrinsic functions, \$ZSYNTAX..... 11
intrinsic functions, \$ZTIME..... 11
intrinsic functions, \$ZZIP..... 11
intrinsic functions,
 implementation-specific..... 10, 11
intrinsic special variables, \$DEVICE..... 5
intrinsic special variables, \$ECODE..... 5
intrinsic special variables, \$ESTACK..... 5
intrinsic special variables, \$ETRAP..... 5
intrinsic special variables, \$HOROLOG..... 5
intrinsic special variables, \$IO..... 5
intrinsic special variables, \$JOB..... 5
intrinsic special variables, \$KEY..... 5
intrinsic special variables, \$PRINCIPAL..... 6
intrinsic special variables, \$QUIT..... 6
intrinsic special variables, \$STACK..... 6
intrinsic special variables, \$STORAGE..... 6
intrinsic special variables, \$SYSTEM..... 6
intrinsic special variables, \$TEST..... 6
intrinsic special variables, \$TLEVEL..... 6
intrinsic special variables, \$TRESTART..... 6
intrinsic special variables, \$X..... 6
intrinsic special variables, \$Y..... 6
intrinsic special variables, \$ZA..... 7
intrinsic special variables, \$ZB..... 7
intrinsic special variables, \$ZCONTROL..... 7
intrinsic special variables, \$ZDATE..... 7
intrinsic special variables, \$ZERROR..... 7
intrinsic special variables, \$ZF..... 7
intrinsic special variables, \$ZHOROLOG..... 7
intrinsic special variables, \$ZINRPT..... 7
intrinsic special variables, \$ZJOB..... 7
intrinsic special variables, \$ZLOCAL..... 7
intrinsic special variables,
 \$ZMATCHALPHABETIC..... 7
intrinsic special variables,
 \$ZMATCHCONTROL..... 7
intrinsic special variables,
 \$ZMATCHEVERYTHING..... 7
intrinsic special variables,
 \$ZMATCHLOWERCASE..... 7
intrinsic special variables,
 \$ZMATCHNUMERIC..... 7
intrinsic special variables,
 \$ZMATCHPUNCTUATION..... 7
intrinsic special variables,
 \$ZMATCHUPPERCASE..... 7
intrinsic special variables, \$ZNAME..... 7
intrinsic special variables, \$ZORDER..... 7
intrinsic special variables, \$ZPRECISION..... 7
intrinsic special variables, \$ZREFERENCE..... 8
intrinsic special variables, \$ZSYSTEM..... 8
intrinsic special variables, \$ZTIME..... 8
intrinsic special variables, \$ZTRAP..... 8
intrinsic special variables, \$ZVERSION..... 8
intrinsic special variables,
 implementation-specific..... 7, 8

intrinsic special variables, unimplemented 6
 invocation, command-line 3

J

JOB 12

K

KILL 12
 Kreis, Greg 1
 KSUBSCRIPTS 12
 KVALUE 12

L

Landis, Larry 1
 limitations, memory 29
 LOCK 12

M

Marshall, Frederick D.S. 1
 maximum size, global 29
 maximum size, routine 29
 maximum size, string 29
 MERGE 12
 Milligan, Lloyd 1
 mlib, initializing 26
 mlib, m_add_subscript() 26
 mlib, m_data() 26
 mlib, m_get() 26
 mlib, m_init_mref() 26
 mlib, m_kill() 26
 mlib, m_ksubscripts() 26
 mlib, m_kvalue() 26
 mlib, m_lock() 26
 mlib, m_num_function() 26
 mlib, m_order() 26
 mlib, m_procedure() 26
 mlib, m_query() 26
 mlib, m_set() 26
 mlib, m_str_function() 26
 mlib, m_unlock() 26
 mlib, m_version() 26
 mlib, mref_t 26
 modes, programmer 4
 Morris, Steve 1
 Murray, John 1

N

namespaces 22
 namespaces, adding 23
 namespaces, listing 22
 namespaces, managing 22
 namespaces, returning to default 24
 namespaces, switching 23
 NEW 12

O

OPEN 12
 options, command-line 3

P

Pastors, Wilhelm 1

Q

QUIT 12

R

READ 12
 REPL, FreeM 4

S

Schell, Kate 1
 Schofield, Lyle 1
 SET 13
 SSVNs 15
 standards, ANSI 35
 standards, implementation
 conformance clause 35
 structured system variables 15, 18
 structured system variables, ^\$CHARACTER .. 15
 structured system variables, ^\$DEVICE 15
 structured system variables, ^\$DISPLAY 15
 structured system variables, ^\$EVENT 15
 structured system variables, ^\$GLOBAL 15
 structured system variables, ^\$JOB 15
 structured system variables, ^\$LOCK 15
 structured system variables, ^\$PDISPLAY 15
 structured system variables, ^\$ROUTINE 15
 structured system variables, ^\$SYSTEM 15
 structured system variables, ^\$WINDOW 15
 structured system variables, user-defined 18
 SYSGEN 26
 SYSNSP 22
 system library routines 19
 system library routines, ^%ED 19
 system library routines, ^%FLIST 19
 system library routines, ^%FUTIL 19
 system library routines, ^%GD 19
 system library routines, ^%GL 19

system library routines, ^%KEY 19
 system library routines, ^%KEYINT 19
 system library routines, ^%KEYTEST 19
 system library routines, ^%KILLJOB 19
 system library routines, ^%LOGIN 19
 system library routines, ^%MEN 19
 system library routines, ^%MESS 19
 system library routines, ^%MUTIL 19
 system library routines, ^%N 19
 system library routines, ^%RD 19
 system library routines, ^%RDEL 19
 system library routines, ^%SS 19
 system library routines, ^%SYS 19
 system library routines, ^%SYSDEV 19
 system library routines, ^%SYSGEN 20
 system library routines, ^%SYSINFO 20
 system library routines, ^%SYSNSP 20
 system library routines, ^%SYSNSP0 20
 system library routines, ^%T 20
 system library routines, ^%u 20
 system library routines, ^%uflis 20
 system library routines, ^%ufxxx 20
 system library routines, ^%ug 20
 system library routines, ^%ugdir 20
 system library routines, ^%ugexp 20
 system library routines, ^%ugimp 20
 system library routines, ^%uglis 20
 system library routines, ^%ugxxx 20
 system library routines, ^%ulmath 20
 system library routines, ^%ulstring 20
 system library routines, ^%ur 20
 system library routines, ^%urdel 20
 system library routines, ^%urdir 20
 system library routines, ^%uredi 20
 system library routines, ^%urexp 21
 system library routines, ^%urimp 21
 system library routines, ^%ursea 21
 system library routines, ^%urxxx 21
 system library routines, ^%UTILITY 21
 system library routines, ^%uxdat 21
 system library routines, ^%uxxxx 21
 system library routines, ^%ZCOLUMNS 21
 system library routines, ^%ZOS 21
 system library routines, ^%ZROWS 21

T

TCOMMIT 13
 TRESTART 13
 Trocha, Axel 1
 TROLLBACK 13

U

USE 13
 utilities 27
 utilities, gcompact 27
 utilities, gfix 27
 utilities, gl 27
 utilities, glocks 27
 utilities, grestore 27
 utilities, gverify 27

V

variables, intrinsic special 5
 variables, structured system 15
 VIEW 13

W

Walters, Dick 1
 Whitten, David 1
 Wicksell, David 1
 Willis, John P. 1
 WRITE 13

X

XECUTE 13

Z

z functions, user-defined 17
 ZALLOCATE 13
 ZBREAK 13
 ZDEALLOCATE 13
 Zeck, Steve 1
 ZGO 13
 ZHALT 13
 ZINSERT 13
 ZJOB 13
 ZLOAD 13
 ZNEW 13
 ZPRINT 13
 ZQUIT 13
 ZREMOVE 13
 ZSAVE 14
 ZTRAP 14
 ZWRITE 14