UNIVERSITY OF CALIFORNIA, DAVIS

BERKELEY • DAVIS • IRVINE • LOS ANGELES • RIVERSIDE • SAN DIEGO • SAN FRANCISCO — SANTA BARBARA • SANTA CRUZ

COLLEGE OF ENGINEERING
DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

DAVIS, CALIFORNIA 95616

October 1983

TO: Recipients of MicroMUMPS Version 4.0

This package contains the deliverables associated with 8080, Z80 and 8086 versions of MicroMUMPS, version 4.0. Included in the package are:

- A distribution disk (8" or 5¼", depending on your order). The disk(s) contain object code for one version of MUMPS, some CP/M or CP/M86 utilities used in setting MUMPS files, and some additional program and/or data files. The disk(s) also contain MUMPS utilities and applications.

- Documentation in printed form.

- The October 1983 newsletter.

- Order forms for MUG reference texts and MUG membership.

- An additional order form for MicroMUMPS related materials.

Special Note to New Recipients

If you are a new subscriber to MicroMUMPS, you may be unfamiliar with the MUMPS language. Included with your shipment is an order form for documents published by the MUMPS Users' Group, and for membership in that organization. There are three documents that are valuable aids to MUMPS programming available from MUG:

Computer Programming in ANS MUMPS, by Krieg: This text is an excellent tutorial level text for the new user who is essentially unfamiliar with programming and computers. It conforms in most respects to the implementation included with this shipment. Written by a physician and intended for professionals in other non-computing fields, it has received very favorable reviews.

MUMPS Primer, Revised: This book is more suited to new MUMPS users who have had experience in other languages. Less tutorial, it can serve as a rapid reference for many programming examples, but does not start at quite the same entry point level as the Krieg text.

Programmer's Reference Manual: This is the "bible" of experienced MUMPS programmers, containing the detailed answers to complex questions in the use of the language. Revised in early 1982, it reflects many, though not quite all, of the changes in the 1982 Standard.

We recommend that any MUMPS programmer have one or more of these documents for their use with MicroMUMPS.

Disk Duplication and Verification

The volume of MicroMUMPS distribution has grown to the point that it is economically feasible to use a commercial duplicating service for disk duplication. If you experience difficulty in reading your disk, try first to read it on another machine in your vicinity. If all fails, return the disk to me with a note indicating the nature of the problem. I will replace the disk, and report the failure to the duplicating service. In view of the guarantees given by the duplicating service, I do not foresee problems with these disks.

Note to Recipients of Former Versions

MUMPS global files created on previous releases must be saved under that version, then restored under the new version to preserve their information. Routines have not been altered in the current version, which means that they are compatible with version 3.5 (but not with earlier versions).

Questions and Problems

The growth of interest in MicroMUMPS has been gratifying, but not without some difficulties. Chief among these problems has been the number of telephone calls requesting assistance. In order to enable us to continue to make progress on new versions, we are anxious to keep to a minimum telephone calls which often interrupt a process requiring concentration.

To assist us in helping you, we ask that, whenever possible, you send us a note, documenting questions that have arisen with your use of MicroMUMPS. We will try to respond as quickly as possible. In general, telephone calls will be handled by Tracy Jones. We are at present working on new MUMPS versions, and we hope you will help us by minimizing the telephone interruptions.

Please address all inquiries and trouble reports to: Richard F. Walters, Ph.D., Department of Electrical and Computer Engineering, University of California, Davis, CA 95616. Thank you for your help.

Sincerely,

[illegible]

Richard F. Walters, Ph.D.
Professor

RFW/tlj

MICROMUMPS

INSTALLATION MANUAL

Electrical and Computer Engineering
University of California
Davis, CA 95616

Stephen L. Johnson
Jeffrey L. Taylor
Richard F. Walters

October, 1983

MICROMUMPS Installation Manual

1. Materials Included

This version of Standard MUMPS has been implemented in 8080 or Z80 Assembly code under the CP/M operating system, and is available for many manufacturer's hardware. Delivery of the system includes an 8" or 5" floppy disk which contains the following materials:

MUMPS.COM - The Object Code for MUMPS running under CP/M. The version number will appear on the disk label as well as in the opening message when the system is entered.

SETGLOB.COM - An Assembly language programs used to initialize globals.

SETMUMPS.COM - A Utility used to reconfigure your MUMPS system.

GLOBALS.DAT - An initial globals file created by SETGLOB.

ERRORS.DAT - the error message file used to provide decoding of specific errors encountered in running MUMPS. Note that the MUMPS system will run without this file, giving only error numbers rather than complete messages. MUMPS assumes the file is on the current default drive, regardless of the location of routines and globals.

*.MMP - The disk will also contain Utility Programs and some Application programs that are available at distribution time. One-Page abstracts for these utilities and applications are distributed with the system. The specific programs distributed in this portion of the package will vary.

UT.GBL - A global backup file used in conjunction with the MUMPS utilities furnished by Dr. Lewkowicz.

The User's Manual, a description of MICROMUMPS features and extensions, is included with the system deliverables.

The Utility/Application Manual which descibes the CP/M and MUMPS utilities provided with the system.

A Test/Validation Dialogue is included to allow the users to verify correct performance of the system.

II. System Requirements for MICROMUMPS

* 8080, 8085 or Z80 CPU

* User area (TPA) of not less than 49K bytes.

This gives a partition size of 4K bytes (routine plus local variables) and two global buffers. This configuration is the absolute minimum that can be used for microMUMPS. Since MUMPS uses buffer space to speed up access of globals and routines, execution will be slower when less memory is available.

TPA bytes	Partition	Buffer space
49k	4k	2k
56k	4k	9k

* One floppy or winchester disk drive.

Two drives are recommended for performance and backup. MicroMumps can use one drive for routines and the other for global files. A recommended configuration is to place the MUMPS interpreter and associated CP/M utilities plus MUMPS routines on drive A, and place only the data base (GLOBALS.DAT) on drive B. This drive assignment is performed using the ZO command, described in the User's Manual, or can be permanently assigned using the SETMUMPS utility.

Standard distribution is provided on the following media:

8 inch single sided, single density floppy disk
(CP/M distribution format)

5 inch single or double sided, double density floppy for Apple with Z80 softcard

5 inch single density floppy for Osborne I

CP/M 2.2

Other compatible operating systems may run MicroMumps, please contact us with any problems or information on this.

* Standard I/O terminals, either printing or CRT, may be used with MICROMUMPS.

Global files are not stored as CP/M sequential ASCII format. There are two implications to this fact. First, it is not possible to read globals through CP/M. Second, when using the PIP command to copy files, it is necessary to use the [O] option to guarantee correct copying. The MUMPS globals must be copied using syntax as follows:

```
PIP B:=A:GLOBALS.DAT[O]
```

This precaution is necessary because these files may have embedded characters that are interpreted by CP/M as end of file.

III. Instruction for Installation

1. Before using the system, it is wise to make a backup disk of the programs and text included on the distribution disk. The distribution disk may be copied using any of the CP/M utilities such as PIP or DISKCOPY. Be sure, however, to use the [O] option for the globals file, GLOBALS.DAT.

2. In order to be certain that you have a clean version of the global file, mount a disk in drive A that has SETGLOB.COM, but no GLOBALS.DAT. Run SETGLOB by typing its name, and you will create a blank file GLOBALS.DAT, initiated for the acceptance of global files.

3. Using a backup disk (created by Step 1 above), Type MUMPS, thereby executing the MUMPS interpreter. The system will prompt for date, then issue a sign-on message including the version and type of system. When a prompt ">" appears, you may begin to enter commands.

4. Following the sequence illustrated in the accompanying Validation Dialog listing, type in the commands underlined, and check that the responses you receive are similar to those in the printout.

5. You may wish to test a few programs sent with the system, such as LEXICON and the FORMS Demo system (^FORMD), to gain additional familiarity with this version and with the MUMPS language.

6. If you wish to reconfigure the system, it is recommended that you run SETMUMPS on a separate copy of the object code, rather than the distribution disk. Follow the instructions provided with SETMUMPS as described in the Utilities portion of this documentation.

To date, this system has been implemented on many different CP/M systems, using large and small disks, Winchester drives and other peripherals are distributed by many manufacturers. If you encounter difficulties in getting this system to run (other than those in copying to different sizes or formats), please report these problems to Richard F. Walters, Dept Electrical and Computer Engineering, University of California, Davis CA 95616. Although we do not provide complete support services, we will do what we can to offer helpful advice or, if our shipment is in error, to replace the disk.

IV. Conversion of Old MUMPS Globals

The new global file structure is not compatible with earlier versions. In order to convert earlier globals to the correct format for the current version, you should do the following:

1. Using your previous version of MUMPS, do ^%MGS to save the globals.

2. Using the new version (4.0) of MUMPS, do ^%MGR to restore the globals in your new GLOBALS.DAT file. Note that you will need to use different disks for the two GLOBALS.DAT files.

MICROMUMPS

USER'S MANUAL

Department of Electrical
& Computer Engineering
University of California
Davis, CA 95616

Stephen L. Johnson
Jeffrey Taylor
Richard F. Walters

October, 1983

MICROMUMPS User's Manual

Table of Contents

MICROMUMPS User's Manual

Introduction

A single user version of Standard MUMPS has been implemented on 8080/Z80 based microcomputers operating under the CP/M Operating system. This research has been supported in part by the Veterans Administration, the MUMPS Systems Laboratory, Nagoya, Japan, Shared Medical Systems, the Lister Hill National Center for Biomedical Communications of the National Library of Medicine, and the University of California. Improvements to the system have been contributed by many individuals, notably John Althouse and J. Wesley Cleveland in London. It is distributed by the University of California, Davis and by the MUMPS Users Groups of Europe, Japan and Brazil.

In its current version, MICROMUMPS includes all commands of the ANSI Standard applicable to single user versions. $HOROLOG is implemented on a few systems, where a clock or timer is available.

MICROMUMPS is continuing to undergo modifications and improvements. Many changes have been introduced since the first version, released in 1979, and the performance has undergone dramatic improvements. The program is distributed with no guarantees of accuracy, although it has been extensively tested and continues to be evaluated and refined on a regular basis. Recipients are urged to report suspected errors to the distributer, enclosing a printout of the suspected error and any comments that might help pinpoint the cause.

This manual describes extensions to the MUMPS language that are included in the 8080/Z80 versions. Some extensions have been added to conform to changes in the language proposed since acceptance of the 1977 Standard (e.g., string subscripts). Some special capabilities available at the micro- computer level are also described in this manual.

Initiating MICROMUMPS

To enter the MUMPS interpreter, begin by turning on the machine and initiating CP/M in the method usualy employed for that system (modes of loading and initializing CP/M systems may vary). Then type "MUMPS", and the interpreter will be loaded. When loaded, the interpreter will prompt for date, give an initial message identifying itself, and then give a prompt ">", which is repeated each time the interpreter is awaiting further input from the user. If MUMPS has been configured for auto execute mode using the SETMUMPS utility, then the command line entered during setup will be executed instead of the initial prompt. See SETMUMPS utility documentation for more infomration on auto execute mode.

To exit MUMPS, type HALT or H, and the monitor will once again assume control of the system. Always exit using the HALT command and allow the system to rewrite all modified global files before returning control to CP/M.

Special Features of MICROMUMPS

MICROMUMPS now includes all the features added to Standard MUMPS for inclusion in the ANS 1983 proposed standard. These features are:

String Subscripts: See $ORDER function, p. 7.
SET $PIECE: the syntax SET $PIECE(X,delim,n)="NEW VALUE" establishes NEW VALUE as the nth piece of variable X.
$ASCII(expr1,intexpr2) returns the $ASCII function on the intexpr2th character of expr1
$EXTRACT(expr) returns the first character of the value of expr.
$JUSTIFY now includes a leading zero in front of decimal fractions.
$LENGTH(expr1,expr2) gives the number of non-overlapping times expr2 occurs in expr1
$TEXT(+0) returns the name of the routine
Pattern match: alternating forms permitted
READ A#n reads up to n characters, terminating at n.

In addition to permitting string subscripts, decimal subscripts are also allowed in MICROMUMPS. Note that the collating order specified by the MUMPS Development Committee treats "canonical numbers" as special cases, separating them from other numbers and alphanumeric strings. See the section on Language Extensions for further details on this topic.

Adherence to Portability Requirements of the MUMPS Standard

This version of Standard MUMPS adheres to or exceeds all provisions specified in part III of the ANSI Standard X11.1 String variables may be up to 255 characters in length, numeric precision is maintained to nine significant digits, numbers in excess of 10 raised to the 25th power can be represented, subscripts may be arbitrarily long, etc.

Implementation Specific Features of MICROMUMPS

The ANSI Standard X11.1 reserves for the implementor a number of language elements. This section describes briefly the manner in which some of those features have been treated in MICROMUMPS.

1. Direct entry of routine lines.

MUMPS can be executed by direct commands, processed as soon as they are typed, or by executing stored routines. provision must be made for the entry of command lines that can be stored as routines. In MICROMUMPS, the TAB character (control - I on some keyboards) is used to signal the start of a command line that is to be stored in the workspace. The Tab character should be positioned <u>after</u> the label, if any, and before the appearance of the first executable command. The Tab characters echoes as a space (a change from earlier versions).

When inserting new command lines into text, they are placed immediately in front of the current text pointer which is then updated to point after the inserted line. Language extensions that permit manipulation of the current text pointer are described in the section on language extensions.

When a routine is entered from an editor, the linestart character may be any combination of one or more tabs or spaces.

2. Control Character Functions

A number of special characters are interpreted by MICROMUMPS to provide actions similar to those encountered in the CP/M operating system. These characters are:

Control- A	information typed to next carriage return is not echoed.
Control- C	interrupt execution. When executed while in dirrect mode, execution is halted and control remains in direct mode. When a routine is being executed, the user returns to direct mode, but can continue execution of the routine by typing ZGO. A second Control-C will abort the routine and return to direct mode.
Control- G	talk-through mode. Everything typed until the next Ctl-G is transmitted directly to the output port, and information received from the port is echoed to screen.
Control- H	backspace and delete character
Control- P	echo to printer
Control- S	pause output. Resume by typing any character
Control- X	remove line, erasing it from screen
rubout	backspace pointer, echo deleted character.

3. BREAK and ZGO Commands

The ANSI standard does not specify completely the BREAK command. In this version, BREAK may be entered with an argument list, which is typed out when the BREAK is encountered. This message may be used to identify the location of the BREAK command invoked, a feature that is helpful in debugging command lines.

Return to execution following interruption by a BREAK command is accomplished by the ZGO command, which picks up routine execution at the point at which it was suspended.

4. READ * and WRITE *; READ A#n

The READ * syntax of Standard MUMPS permits user definition of the interpretation of this special command. In MICROMUMPS, the suggestion contained in the MUMPS Programmer's Reference Manual is followed, and READ * is used to implement a single character READ for use in those programs where single character input is an important requirement of certain applications. When MUMPS encounters the READ command followed by an asterisk, it returns a single character which is the ASCII code equivalent of the character entered from the keyboard. Note that the READ * command does not echo the character typed to the screen.

READ A#n, where n is a number o characters to be read, will cause the READ to be terminated either when n characters have been typed or when a carriage return is typed. This form of the READ command is a new feature of 1983 ANS MUMPS.

5. $STORAGE

The special variables $STORAGE returns a number equivalent to the number of unused bytes of storage remaining in the workspace of the system.

6. VIEW Command

VIEW is implemented with a single argument that may have the values 0 through 4. The meaning of these values is defined as follows:

0	returns a list of local variable names and values
1	returns local variable names, subscripts and values of local subscripted variables
2	returns a list Global Variable Names.
3	returns a list of MUMPS routines on disk
4	returns a directory of all CP/M files on drive specified as routine drive.

7. HANG and Timeout Functions

The HANG command permits a pause of specified number of seconds. Likewise, various commands such as READ allow for a timeout in the same units. Since most systems do not have an external clock, it is necessary to implement these features using a count-down algorithm which is dependent on the internal cycle time of the system. The Z80 implementation supplied with this documentation uses a 4MHz clock and the timing features are based on the internal cycle times of that system. Since other systems may run at different speeds, the precise time of the HANG command may vary.

8. OPEN #: (parameters)

OPEN allows assignment of different I/O and disk devices and files. The device assignments currently available in OPEN, CLOSE and USE are:

0	the primary input device (CP/M mode)
1	printer (CP/M listing device)
2	CP/M disk file
3	CP/M disk file
4	CP/M disk file
5	Serial Port (CP/M Reader/Punch)

If OPEN is invoked with either 1 or 5 as the device specification, parameters are ignored.

An OPEN command specifying one of the disk files requires the syntax (DRIVE:FILENAME). Failure to specify parameters will generate an error. These two parameters may be string variables or quoted constants. For example,

```
          OPEN 3:("A":"TEST.ASM")
```

will assign to device number 3 the CP/M file TEST.ASM. Note the necessity for both quotation marks and parentheses in this specification. If variables have been previously set, they may also be used. For example, if X="A" and Y="TEST.ASM", then the command

```
          OPEN 3:(X:Y)
```

is a valid syntax for this command.

MicroMUMPS now permits addition of a record at the end of an existing CP/M file. The syntax for this variation of the OPEN command is:

```
          OPEN 3:("A":"FILE.EXT":"END")
```

Thereafter, any records written to that file are added at the end. See description of the USE command for random read/write of CP/M files.

In order to use the remote device it it necessary to provide for a status check on the RDR device (each device has a status port and a data port). The routine should return the Z-flag set when no data are present, and the NZ set when data are available to be read. Following are a few examples of ways in which the status port can be set:

Radio Shack (TRS80 Model II)

```
     Port A              Port B
     IN   F6             IN   F7
     ANI  1              ANI  1
     RET                 RET
```

To set the routine on a new system, use the following command sequence:

```
 A>DDT MUMPS.COM
 DDT VERS 2.2
 NEXT PC
 A000 0100              (Remember these numbers: A0-01)
 -D105  106              the address of the status routine: 105-106
 0105 76    8D          the address is 8D76
 -A8D76                 assemble at 8D76
 8D76 IN   F7           this is the status port address for port B
 8D78 ANI  1            on the TRS80 mod II
 8D7A RET
 8D7B                   Just type return to terminate new code
 -^C                    back to CP/M
 A>SAVE 159 MUMPSX.COM     ;number of CP/M pages to save starting
                        at  A0-01.  note different name is  used  for
                        this version. It can be renamed later.
```

There are 20 bytes available for the status routine. In order to use it, you must find out which is the status port of your system and insert that value where F7 appears above.

9. CLOSE #: options

CLOSE refers to the same files and devices specified in OPEN and USE, described above. The two options available are 0 (close and lock the file, and 1 (purge the file and close it). For example,

CLOSE 3:0 (closes and saves the file specified as device 3)

CLOSE 2:1 (erases the file referred to as device 2)
The default value (if no number appears after the colon) is to leave the file in the status it was in when MUMPS use of the file started (a new file is deleted, an old file retained).

10. USE #[:arg]

USE instructs the interface to direct I/O to the device/file specified. USE 1 directs output to the printer, USE 2 references read and write commands to the CP/M file specified, etc.

The USE command has several variations that can be used with devices 2, 3 and 4 to permit appending data at the end of a sequential file and accessing directly records in a random CP/M file. For a sequential file,

```
USE 2:"END"
```

will move the file pointer to the end of the current file and add data at that point. This option should only be used when a pre-existing file has been opened.

Random CP/M files must first be declared outside MUMPS using standard CP/M procedures. Individuals who are not familiar with these procedures should not attempt to use this feature. When a random file exists, the syntax

```
USE 2:n
```

where n is the sector number to be read, will move the pointer to that location so that the next READ or WRITE is of the desired sector. An alternate syntax,

```
USE 2:n:m
```

addresses the m-th byte of the n-th sector. READ and WRITE actions are permitted in this mode, but extreme care not to overflow the existing file structure is required. In general, this option is intended only for the advanced programmer who is already familiar with CP/M. Potential hazards of its misuse include unknown errors when the user attempts to write past the end of a random CP/M file and disruption of the addressing when a sector is overflowed.

11. SET $X and SET $Y

Frequently, it is desirable, in screen formatting dependent routines, to redefine the $X and $Y special variables to conform to new locations of the cursor after terminal-specific write sequences have been executed. MicroMUMPS uses the same convention that has been adopted by several commercial versions of Standard MUMPS - $X and $Y may be set by the user, who must write special routines to accomplish this function. Version 4.0 now supports this feature.

12. WRITE (without arguments)

Several commercial MUMPS systems (DSM, ISM etc.) use WRITE without arguments as a way to view local variables. In order to make use of MicroMUMPS as convenient as possible for those individuals who use these versions, we have implemented WRITE with no arguments to perform exactly the same function as VIEW 0. The list of variables is displayed, but subscripted values are not displayed.

Features Specific to CP/M Operating System

A great deal of care has been taken to minimize the dependence on the CP/M operating system. However, certain function must be performed by an operating system: these functions, which include primarily those involving terminal and disk I/O are described below.

1 Terminal I/O

Calls for I/O have been converted in the current version to BIOS calls rather than BDOS references. This change permits interruption of execution by trapping special control characters as described earlier in this manual.

Formatting output control characters include the Form Feed (OCH) for new page or clear screen, and the conventional CR/LF new line start characters. Other special characters specific to different devices may be output using the WRITE *nn where nn is the decimal value of an appropriate ASCII code.

For example to position the cursor on some CRTs you would enter

```
WRITE  *27,"Y",*31+X,*31+Y ;  output ECS sequence,  then X Y
coordinates with displacement.
```

2 Serial Port I/O

Input and Ouput of data on a serial line other than the system console is supported in this version of MICROMUMPS. The facility is provided using the BDOS calls for the CP/M Reader/Punch devices as described in the CP/M documentation. It may be necessary to use STAT or other CP/M utilities to make appropriate IO Byte assigments for these devices. It is also necessary to set the speed and character length outside MUMPS. Note that at higher baud rates it is possible to lose characters, since interrupt handling and buffering is usually not provided in the BIOS for this device.

3 CP/M Sequential File I/O

As described above, MICROMUMPS provides three devices that allow reading and writing of CP/M sequential ASCII files. The USE, READ and WRITE commands operate like the SDP files provided by DSM-11, except that the file can only be accessed from the beginning. A single CNTRL/Z is returned to the MUMPS routine when a CP/M End of File is detected on reads. This facility is useful for writing MUMPS routines or globals for backup and for interface to other languages or word processors. Examples of use of these devices can be found in the routines %ZBU and %ZRS as well as the routine editor %EDIT.

Z-Command Extensions

The MUMPS Standard provides a means whereby the language can be extended in order to take care of I/O and other features not fully specified in the standard. The general provision is to allow new commands beginning with Z to be included in any implementation. MICROMUMPS Z extensions are similar to or identical to DSM and ISM comparable commands. In one or two instances, minor differences exist. Users familiar with DSM, ISM, or with earlier versions of MICROMUMPS should note these language features carefully to avoid potential confusion or misuse.

$ZCOUNT returns the number of free blocks available in GLOBALS.DAT. Can be used in program mode to avert any problems of overflowing allocated global area.

ZD(ELETE) <fname> will delete a routine file from disk. This command is different from the DSM and ISM method of deleting disk copies of routines.

ZE(RROR) <argument> When a BREAK is encountered, and ZERROR has been set, the error message giving location of the error is displayed (pointer to command, label plus offset). If ZERROR has arguments, control will pass to the entryref^routine in the argument.

$ZG(LOBAL) returns the drive name on which globals are currently located. It is useful in program mode to manage files when multiple disks are used.

ZI(NSERT) <string>:<label (+n)> inserts the string before the specified line, and moves the line pointer just past the newly inserted line. The format of the <string> may be one of the following:

"label (space) command line"
"(space) command line
VAR, where var is a previously defined string conforming to one of the two options listed above.

Note that the quotation marks are required if the string is entered directly, and the line start character must be a space.

ZL(OAD) <name> - retrieves the named routine from disk, placing it in the workspace. The line pointer is set at the end of the routine loaded.

ZM(OVE) (without arguments) : moves pointer to top of routine

ZM <label>(+n)	moves pointer to label and advances +n lines
ZM :	moves to end of routine

$ZN(AME) returns name of current routine

ZO(PTION) displays current routine and global drive names

```
ZO drvl          changes routine drive to drvl
ZO (drvl:drv2)   changes routine to drvl, global drive to drv2
ZO (:drv2)       changes global drive to drv2
    Example: ZO "B" places routines on drive B
             ZO ("C":"A") places routines on C, globals on A
        Note that the quotation marks are required
```

$ZO[RDER](args) returns the next full global subscript reference after the argument specified.

Example: assume that ^A(2), ^A(2,1), ^A(2,3,1,1), and ^A(3) are defined. The MUMPS command line:

```
SET X="^A("""")" FOR I=1:1 SET X=$ZO(@X) Q:X="" W X,!
```

would write

```
^A(2)
^A(2,1)
^A(2,3,1,1)
^A(3)
```

ZP(RINT) (without arguments) prints entire routine (P may be used without ZP, as in DSM)

ZP *	prints current line
ZP <label (+n)>	prints the line specified by <label>, optionally offset by (n) lines
ZP <label1 (+n1)>: label2 (+n2)>	prints all lines between and including the lines referenced by label1 and label2 as modified by n1 and n2
ZP <label (+n)>:	lists all lines from the referenced label to the end of the work space

Note that ZPRINT does not affect the position of the line pointer.

ZR(EMOVE)	(without arguments) deletes entire routine from workspace
ZR *	deletes current line
ZR <label (+n)>	deletes line specified by label and offset.
ZR <label1 (+n)>:label2 (+n)>	deletes all lines between and including label1 and label2 (plus offsets)
ZR <label (+n)>:	deletes all lines from specified label and offset to end of routine

$ZR(OUTINE) returns the drive name for routines. See $ZG.

ZS(AVE) (without arguments) saves the routine with the current name assigned ($ZNAME)

ZS <name> saves routine with name assigned.

Global Files

In order to increase execution speed by reducing the number of disk seeks (a major delay on floppy disk systems), in-memory buffers are used. A disk directory bit map is retained in memory, and updated each time globals are added or KILLED. This bit map is an essential key to the global files. If a system failure occurs before the bit map directory has been written to disk, serious degradation of the global files may occur, usually resulting in the loss of one or more variables previously set. For this reason, it is important to minimize the potential effect of this loss by following one or more of the suggestions given below.

One way to update files online is to use the ZOPTION command in either direct or indirect mode. Invoking ZOPTION will force update of the disk, and execution may continue normally.

Backup files of transactions, or other software mechanisms to protect against power failures may be used as in all systems. Occasional use of the global dumper utilities (see section on utility programs) will also provide checkpoint backup for sensitive file.

To maximize data base performance, the GLOBALS.DAT file should be created at the beginning of an empty disk, and ideally should be the only file on that disk. This avoids file fragmentation and positions the file near the CP/M directory, which is accessed when random disk I/O forces a directory lookup to change extents.

IMPORTANT: At the end of each MUMPS session, you must exit MUMPS with a HALT (normal exit) so that the global buffers in memory are written to disk. Failure to do so will corrupt the global file.

ERROR MESSAGES

0	MISSING LINESTART CHARACTERS
1	STACK OVERFLOW
2	ARITHMETIC OVERFLOW
3	UNDEFINED LOCAL VARIABLE
4	ILLEGAL FUNCTION NAME
5	ILLEGAL COMMAND NAME
6	UNDEFINED PROGRAM NAME
7	UNDEFINED GLOBAL NAME
8	GLOBAL+VALUE TOO LONG
9	DIRECTORY FULL
10	PROGRAM LARGER THAN PARTITION
11	STRING TOO LONG
12	UNMATCHED PARENTHESES
13	ILLEGAL USE OF NOT OPERATOR
14	ILLEGAL COMMAND TERMINATOR
15	TOO MANY NAKED REFERENCES
16	UNAUTHORIZED GLOBAL ACCESS
17	DIVISION BY ZERO
18	ILLEGAL CHARACTER
19	SYNTAX STACK OVERFLOW
20	ILLEGAL EXPRESSION
21	ILLEGAL PATTERN
22	MISSING COMMA
23	ILLEGAL VARIABLE NAME
24	ILLEGAL USE OF INDIRECTION
25	UNDEFINED ROUTINE LINE NUMBER
26	ILLEGAL NUMERIC LITERAL
27	MISSING EQUAL SIGN
28	ILLEGAL ROUTINE OR LABEL NAME
29	INVALID NAME SYNTAX
30	UNIMPLEMENTED OPERATION
31	SYMBOL TABLE OVERFLOW
32	TOO MANY LEVELS OF NESTING
33	DUPLICATE LABEL
34	INVALID LINE REFERENCE
35	NO TRUE VALUE IN $SELECT
36	NAKED GLOBAL REFERENCE ILLEGAL
37	GLOBAL FILE NOT ON DISK
38	ROUTINE NOT ON DISK
39	DISK I/O ERROR
40	ROUTINE IS UNNAMED
41	ROUTINE ALREADY IN LIBRARY
42	REMOVE OR SAVE ROUTINE
43	COMMAND ONLY USED IN INDIRECT MODE
44	ILLEGAL COMMAND DURING BREAK
45	COMMAND ONLY USED DURING BREAK
46	SUBSCRIPT MISSING
47	INVALID SUBSCRIPT
48	INVALID OPEN PARAMETER
49	DEVICE NOT OPEN
50	INVALID DRIVE
51	TRYING TO READ FROM WRITE ONLY DEVICE
52	INVALID DEVICE NUMBER
53	INVALID REPETITION COUNT
54	INVALID READ COUNT

MICROMUMPS

UTILITIES

Electrical and Computer Engineering
University of California
Davis, CA 95616

October, 1983

SETMUMPS.COM - Interpreter Configuration

SETMUMPS.COM is an assembly language utility that allows you to reconfigure a MUMPS System. The parameters that can be modified are the partition size (which indirectly determines the space available for routine and global buffers, default CP/M drive assignment for routines, globals, and error message file, revision of error messages (e.g., for foreign language messages), clear screen and the facility for an auto execute mode.

To use, type SETMUMPS and follow instructions as shown in the dialog below. You will be given options to change each parameter listed, with the current default also provided.

When auto execute mode is used the MUMPS interpreter automatically starts executing predetermined routine sequence. Auto execute mode is set up by typing yes to the auto execute question. Then enter the command(s) to execute when MUMPS starts. Any time control would return to direct mode an exit is made to CP/M.

The following examples illustrate different responses to SETMUMPS prompts:

```
A>setmumps

Enter drive of the MUMPS.COM to change? y
Enter partition size in bytes (6000)? 4096
That leaves 9382 bytes for routine and global buffers

Is that enough room (Y/N) y
Enter default global drive (B)?
Enter default routine drive (A)? b
Enter default error message drive (A)?
Do you want to change the error message file name? (Y/N) n
Currently the clear screen string is (decimal) -- 126 27
                                 (hexadecimal) -- 7E 1B
Do you want to change it? (Y/N) y
How many characters in the clear screen string? 1
Enter one character (hexadecimal or decimal) -- 12

Currently the auto-execute command is undefined
Do you want to change it? (Y/N) y
Enter the MUMPS commands to auto-execute
D ^DI

The new configuration of MUMPS is saved

A>
```

```
setmumps

Enter drive of the MUMPS.COM to change: a
Enter partition size in bytes (4096): 5050
That leaves 8428 bytes for routine and global buffers

Is that enough room (Y/N) y
Enter default global drive (B): a
Enter default routine drive (B):
Enter default error message drive (A): b
Do you want to change the error message file name? (Y/N) y
Enter name for new error message file: NEWERRS.DAT

Currently the clear screen string is (decimal) -- 12
                                (hexadecimal) -- 0C
Do you want to change it? (Y/N) y
How many characters in the clear screen string? 2
Enter one character (hexadecimal or decimal) -- 126
Enter one character (hexadecimal or decimal) -- 27

Currently the auto-execute command is D ^DI
Do you want to change it? (Y/N) y
Enter the MUMPS commands to auto-execute

Do you want to delete it? (Y/N) y

The new configuration of MUMPS is saved

A>
```

```
setmumps

Enter drive of the MUMPS.COM to change: a
Enter partition size in bytes (5050): 4096
That leaves 9382 bytes for routine and global buffers

Is that enough room (Y/N) y
Enter default global drive (A):
Enter default routine drive (B): a
Enter default error message drive (B): a
Do you want to change the error message file name? (Y/N) y
Enter name for new error message file: errors.dat

Currently the clear screen string is (decimal) -- 126 27
                                (hexadecimal) -- 7E 1B
Do you want to change it? (Y/N) n
Currently the auto-execute command is undefined
Do you want to change it? (Y/N) y
Enter the MUMPS commands to auto-execute
D ^TEST S A=100 D ^RETEST

The new configuration of MUMPS is saved

A>
A>
```

SETGLOB.COM - Global File Initialization

SETGLOB.COM is an assembly language utitlity that creates an empty global file (GLOBALS.DAT). To use type SETGLOB. When asked for a drive, enter the letter only of the drive to put the global file on. If the file already exists an error message is printed. Note that this utility clears all globals from the data base. The next question is about the maximum size of the global file. This value must be between 1 and 8000. disk. A file of the size specified is allocated at this time. but bitmaps are created to allow MUMPS to allocate and deallocate space within the GLOBALS.DAT file. Note: CP/M can't handle files larger than 8 megabytes,so for now 8000 is the practical upper limit for this size.

The distribution disk has a very small global file created. This file is sufficient to demonstrate MUMPS functions, but it should not be used when creating permanent data.

Errors will be generated if the user attempts to write beyond the upper limit of the global file. A warning is issued when only 7 or so blocks are available (slightly over 5 K bytes). It is possible to continue writing to globals at this time, but the user should be very cautious about doing so. It is better to purge or save the existing global so that new data can be accommodated.

SAMPLE RUN:

```
A>SETGLOB

Which drive do you want the globals on?  A
Enter the maximum size for your globals file (K-Bytes): 200
The global file is initialized.

A>
```

SHOW

SHOW is a routine to dump out random portions of a CP/M ASCII file as well as an example of some of the new features included in microMUMPS. SHOW asks for a file name and drive and then checks to see if the file is present. If so, a record number is requested. By entering a number between 0 and the file size (in 128 byte records), the file starting at that point is dumped out. Invalid record numbers are checked for. Instead of a record number, a RETURN can be typed, and the dump will start at the sector after the previous dump. Enter Q or q to exit SHOW.

The features found in SHOW start at the second line, SHOW+1, where an error trap is enabled to label ERR. At ENTER+1, $ZR is used to save the current routine drive in variable DR. At ENTER+2, a post conditional on ZO may change the routine drive. The next line uses $ZE to test the existence of the file to dump. The next new feature is found in the line at DSP+3 where device 2 is to be USEd at record NS to do the random read. At END, the routine drive may be changed back to what it was on entry to SHOW. The error trapping at ERR at the end uses the new form of the error trap message to determine if an error 48 (invalid parameter) occured. If so, the record number entered was bad. In which case a message is written, the error trap is reenabled, and control is given back to the program. Any other error causes exit of the program.

TK

The TK routines form a general purpose file transfer utility between MUMPS systems. Currently it will send and receive (with full handshaking, error detection, and retransmission) MUMPS routines, MUMPS globals, and CP/M ASCII files. A planned extension will also handle other (nonASCII) CP/M files. Without any changes, the utility will run between two microMUMPS systems that have the RDR: and PUN: devices assigned to an external serial line and the input status patch installed for this external line. With only minor changes, routine and global transfers can be made between any MUMPS systems.

Example uses:

Following is a sample dialogue for transfers between systems with an operator at each system.

To send:

```
>D ^TK

Are you connected to the remote MUMPS system as a terminal (Y/N) ? N
Do you wish to send or receive (R/S) ? S
Sending routines, globals, or CP/M files (R/G/F) ? R
Do you want outgoing data displayed (Y/N) ? N

Option -- ADD
Enter the routine name: %M
Enter the routine name: %MO
Enter the routine name:

Option -- LIST

%M         %MO

Option --

Routine %M being sent -- Sent
Routine %MO being sent -- Sent

Successful completion

>
```

To receive:

```
>D ^TK

Are you connected to the remote MUMPS system as a terminal (Y/N) ? N
Do you wish to send or receive (R/S) ? R
```

```
Receiving routines, globals, or CP/M files (R/G/F) ? R
Do you want incoming data displayed (Y/N) ? N

Routine %M being received -- Received
Routine %MO being received -- Received

Successful completion

>
```

Following is a sample dialogue for transfers between systems with an operator only at one of the machines.

To send:

```
>D ^TK

Are you connected to the remote MUMPS system as a terminal (Y/N) ? Y
Do you wish to send or receive (R/S) ? S
Sending routines, globals, or CP/M files (R/G/F) ? G
Do you want outgoing data displayed (Y/N) ? N

Option -- ADD
Enter the global name: ^%MU
Enter the global name: ^%MTC
Enter the global name:

Option -- LIST

%MU        %MTC

Option --

Global %MTC being sent -- Sent
Global %MU being sent -- Sent

Successful completion

>
```

To receive:

```
>D ^TK

Are you connected to the remote MUMPS system as a terminal (Y/N) ? Y
Do you wish to send or receive (R/S) ? R
Receiving routines, globals, or CP/M files (R/G/F) ? R
Do you want incoming data displayed (Y/N) ? N
Does remote site already have global list in global ^U (Y/N) ? N

Enter routine name: VV
Enter routine name: VVAC
Enter routine name:


Routine VV being received -- Received
Routine VVAC being received -- Received

Successful completion

>
```

Comments:

The first question would be answered yes (Y) only if there is one operator controlling the entire process, i.e. the MUMPS system is effectively acting as a terminal to another MUMPS machine. Answering yes (Y) to "Do you want incoming data displayed?" will cause all the data being transferred to also be displayed on the screen.

When sending data a list of routines, globals, or files must be made. Unfortunately, this package was developed independently of the Lewkowicz utilities package so there is no connection between the selection process and the list that may already exist in ^%MU. The two will be made compatible in a future release.

When transferring to a unmanned remote system the following procedure is used. Start microMUMPS until the first prompt. Then type control-G to begin talk-through mode. Now your MUMPS is acting as a dumb terminal. Everything typed is displayed and sent out and everything received is displayed. Now log into the remote system. When this is accomplished, type control-G again to return to microMUMPS. Now start the procedure outlined above. When the transfer is complete, don't forget to log off the remote system by typing control-G, doing the log out, and another control-G to return to microMUMPS.

Routines included:		
	TK	The main driver
	TKI	Initializes variables
	TKR	Receive data
	TKS	Send data
	%DIR, %NAMES	Collect names to send
	GT	Simulates $ZO function for machines without $ZO

%DATE - Date Utility

Purpose: to return, from $HOROLOG internal storage, the correct date.

%DATE calculates the date from $HOROLOG (which is created at the signon message). The value for the date is put in the local variable %DT.

%EDIT - General Purpose Editor

The program is executed first by typing:

```
>DO ^%EDIT
```

This envokes the editor and creates a global file ^%, which is then used to edit a routine or global. When complete, the program types:

```
THE EDITOR IS LOADED.
EDITOR HELPED TEXT IS LOADED
```

To execute this global file, type

```
XECUTE ^%
```

The response from this command is a new prompt: >>, which appears as long as you are in the editor global. To exit the editor, type a return when this prompt appears.

All commands operate on the current ZLOADED routine. There are several commands that may be invoked through the editor. To enter a command type in the first letter only and answer additional prompts as indicated.

? Displays a summary of commands available.

B (Backup routine): save a routine on drive A as (name).SAV, to provide a backup version, or to use another editor.

R (Restore routine): Restores a routine previously backed up.

C (Change): the program then asks for the label+offset of the target line. After you type that line identification, the program lists the line, and asks for the string that you wish to replace, then types the line up to that string, asks for the replacement, and, when you type a return, completes the line beyond the text that you have replaced. It repeats this request (on the same line) until you type a return with no other characters in response to a REPLACE statement. It then returns you to the general editor for other commands.

This option can also be used directly from the >> prompt by typing in a label with optional offset followed by one space.

Globals may be editted using the same facilities by entering a full global reference at the >> prompt.

D (Delete): asks for label+offset. Deletes that line.

I (Insert): insert before a (label+offset) line. When inserting, use a blank (space) for linestart character. Terminate insert mode with a blank line.

L (List): lists from label+offset to label+offset.

Editor commands (continued)

S (Search For): lists all lines in which the target string is found, with a >below the string.

E (Every Occurence): replaces every occuraece of one string with another in the current routine. Note that STACK OVERFLOW messages may appear when editting very long lines of code with this option.

In addition, any direct mode command(s) may be executed from the >> prompt, for example, ZPRINT to print the entire routine. Note that ZLOADing a new routine to edit must be performed at the > promtp, not in the editor.

When you have completed the use of the editor, you should resave your routine using the ZSAVE command.

Source: J.J. Althouse, SMS Europe

Utility Cross Reference Listing Package

A package that accepts one or more MUMPS routines and generates cross reference information as well as syntax checking has been provided on the distribution disk. The package was originally a large number of routines that were a portion of the DOC package distributed with QUEST by the MUMPS Users' Group. Robert Lushene (Veterans Administration, Florida) modified the routines to fit into a very few individual routines and still provide the basic benefits of the larger package.

In order to run the cross reference programs, one should type DO ^%INDEX. This program then asks the name of the routine(s) to be analyzed and asks further for output options (a note of unreferenced labels, print index, list routines and print errors and warnings). During execution, the program outputs a period for each line that it has processed. At the conclusion of the analysis of the routines, the requested output is produced on the current output device.

This package is helpful, not only in identifying the use of variables and possible missing referenced labels, but it also will note syntax problems encountered in programs, thereby providing a rapid verification of the accuracy of programs before they are executed.

The programs use mostly local variables. In addtion, global variables ^XCR and ^UTILITY are used during execution and killed thereafter.

Source: Robert Lushene
        Veterans Administration
        Florida
        For futher information, please contact:

Routines used: %INDEX, %INDX1, %INDX2, %INDX3, %INDX4, %INDX5,
%INDX6,%INDX7,%INDX8,%RSET,%IS

%ZFN Mathematical Functions Utility

Although MUMPS is not ideally suited to mathematical calculations, a need often exists for simple functions to be incorporated in other routines. %ZFN was written to provide a number of mathematical functions commonly found in other languages. The functions available are:

LN	%LN is returned as the natural logarithm of %X
EXP	%E is returned as the exponent of %X
PWR	%P is returned as the value of %X to the %Y power
SIN	%Y is returned as the sine of %X(in radians)
COS	%Y is returned as the cosine of %X(in radians)
DTR	%X is converted from degrees to radians
RTD	%X is converted from radians to degrees
SQRT	%Y is returned as the square root of %X

These functions are not rapid in MicroMUMPS, but they work.

LEXICON

Purpose:

This is a very brief demonstration program to illustrate the power of the language through use of string subscripts. it can serve as an illustration for slide presentations, or as a demonstration live on a screen.

The program accepts terms and their definitions, storing them in a global variable. Upon conclusion of entry mode, the user is given the option of listing the terms, which are then reproduced in alphabetical order.

An alternate form of the program output, WPRNT, allows for longer than one line definitions of words and gives formatted output.

Routines used: LEXICON, WPRNT

MICROMUMPS APPLICATIONS

MAILING LIST

Chapter 18 of the MUMPS PRIMER gives an example of a MUMPS package intended for use in creating and maintaining a mailing list. The routines listed in the PRIMER had a few typographical corrections which were made by Jack Bowie during the past year. We have taken those routines as edited by him and entered them into our system, making one or two minor additional changes to adapt them to our user devices.

Details of the mailing list package, it rationale and its limitations, are to be found in the MUMPS PRIMER.

There are 5 routines included in this package.

MOPT is the driving program that gives the user options to create, edit, or retrieve data from the global file.

MEDIT is used to add new entries and to edit or delete old entries. This routine uses an auxiliary routine, MLOOK, to convert between internal and external format of data, etc.

MFLG is used to create, define, and edit flags used to select specific subgroups from the mailing list.

MPRT is used to output either lists of the individuals in a mailing list, or else to generate mailing labels.

The mailing list package accepts up to 3 lines for street address, and one line each for city, state, zipcode and telephone number. It is thus fairly flexible, especially when the flag options are utilized.

The program was not designed to be comprehensive, but rather to be an introductory example of the use of the MUMPS language. Recipients of this package are encouraged to look at ways in which it can be improved, and perhaps to resubmit those changes, with documentation, to the MicroMUMPS Users' Group.

SPELL

This program, written by a student in a MUMPS class at UC Davis, is intended to be a self-extending spelling checker program. There are three versions of the program: SPELL, SPELL52 and SPELL100. The latter two are identical to the first except that they have been modified to generate screen formatted displays for the VT52 and VT100 terminals.

The program operates on a text file, stored in the global ^TXT(n), where n is a positive integer. The program will review all words in the file, flag those not yet in its dictionary (stored in the global ^SPELL, and give the user an opportunity either to add the unknown word, to accept it as a temporary word for that file, or to delete and/or correct it. It is not necessary to initialize ^SPELL, since it will extend itself during use of the program.

If you have a different global file and wish to check from a specific node in that file for spelling errors, you can reset two local variables, REF and NEXTN as shown in the following example, which assumes you have a text file ^RPT(TXT4):

```
S REF="^RPT(TXT,N,0)",NEXTN="N=$O(^RPT(TXT,N))" D NODE^SPELL
```

This program is an interesting example of the use of the MUMPS language. It may require some modification to adapt to new environments, but it should form the basis of a good utility.

Routines included: SPELL,SPELL52,SPELL100

MICROMUMPS APPLICATIONS

TURNSTILE (A Bibliography Program)

TURNSTILE is a set of programs written by a veterinary medicine student at U.C. Davis in an effort to learn about the MUMPS language. It is a rather good example of modular programming, and it serves a useful purpose, so we have included it in our applications package. It is also one that can be modified easily to suit the particular needs of a user.

The LIBRARY created in this package consists of references with author, subject, title, bibliographic citation, summary (abstract), and location of the reference.

Because the package consists of many short programs, it has been stored on disk using the %M utility under the name TURNSTIL.ROU. To restore this package, you will need a different disk with the %M utilities and space for about 25 (small) routines.

TURNSTILE consists of three major parts:

1. LIB used to enter new data
2. CHOOSE used to reference stored data by keyword indexing
3. BURN used to eliminate references

LIB

The LIB program consists of 8 subprograms that organize data, select keywords, and tailor referencing of keywords to the user's specifications.

CATALOG separates major keywords (authors, subjects) that are always in the library from potential words that will either be 'shelved' for referencing of 'trashed'.

BIN shunts strings of words to subprograms for processing.

DUST alphabetizes the 'sentences' for quicker filing.

ELIMIN shunts keywords to their alphabetic shelf in the library or their trash can.

CHOP is used bu DUST and ELIMIN to cut sentences into words. for the user to sort.

TRASH files unwanted words in a 'silent reference file' so that these words will be eliminated in subsequent entries.

SHELFER adds a new reference number to a keyword that already exists.

FINDSHLF creates new space in the reference library for a new keyword.

CHOOSE

The Choose program contains twelve subprograms that search for references by subject or author.

AUTHSEAR searches for author by name (last of last and first.

SUBJ searches by keyword

AUTHFIND initiates a search for an author with only the last m given.

MICROMUMPS APPLICATIONS

SEARCH is used by both full name author and keyword searches.
COMPARE and LISTER "read" the reference library in alphabetical order to find the desired references.
SHOWLIST gives a short listing (author, title) of references pointed to by a keyword.
SHOWBOOK gives a complete listing of references (title, subject, summary, author, and location of article).
SHOWCHOP prompts viewing of SHOWBOOK.
COMLIST, TITSPLIT and CHOP are used by the other subprograms to present references in readable form.

BURN

The BURN program assumes that the package is being referenced by several different individuals, each wishing to tailor the data base to their specifications. For this reason, a method of BURNing some books is needed to make way for new entries. This part of the program is hazardous to the references and should be used with caution (probably restricted to few individuals).

OPENBOOK starts a new library, purging all old references.
SCANBOOK eliminates reference points in LIB without removing previous references; when this program is run, then editing of the "trash" and "shelf" files can be done in direct mode in MUMPS. After editing, however, the reference points must be reset to be compatible with other programs.
SHELF sets up several global variables.
TOMES allows direct access to shelves. Browsing is permitted either by PAGE or ARCHIVES.
PAGE shows only certain alphabetic sections (by letter(s)).
ARCHIVES gives a complete list of everything in the liobrary. If a word was either trashed or shelved incorrectly, it can be detected and corrected via SCANBOOK.

Summary

The author of these programs intends them to be modified for use by individual users. Corrections and suggestions welcomed. Please forward suggestions to UC Davis MicroMUMPS project office.

FORMS DEMONSTRATION SYSTEM

This is a set of routines that uses internal tables to enter, edit and ouput a simple data entry form on the screen. The package illustrates the use of string subscripts, indirection and uses cursor addressing to display and prompt for data. Note that certain lines may need to be modified to output the correct sequence of control characters to perform cursor addressing for your terminal type. You can use %MTC to set your terminal.

FORMD - The Forms Demo driver routine, which prompts for option and dispatches to the input, edit or output routines.

FORMI - The forms input routine, which assigns an identification number, then prompts for and displays various data fields as defined in a table at the end of the routine. This table includes X and Y coordinates for display, a prompt text, a display text and a local variable name to store the data.

FORME - The forms edit routine, which prompts for an identification number, then allows the user to edit fields already enetered with FORMI. Enter return to each prompt to retain data. The table controls execution as above.

FORMO - The forms output routine, which prompts for an identification number, then displays the data as specified in the table.

MICROMUMPS Programming Utility

John Lewkowicz
New York State College of Veterinary Medicine
Cornell University
Ithaca, N.Y. 14853

Table of Contents

Introduction

This utility package is a unified collection of MUMPS programs designed to aid programmers in developing and maintaining MUMPS programs and data bases. It provides for creating, editing, listing, saving, restoring and killing of both routines and globals in a single interactive package.

To invoke any of the utility functions, issue the command:

>D ^%M

while in direct mode. The first time the program is run on any system it will automatically initialize all data bases and ask for various default conditions.

This package was designed specifically to run on **MicroMUMPS** developed by Dr. Walters (University of California, Davis). It is written in standard MUMPS but makes use of many special "Z" commands (ZO etc.) and "$Z" functions ($ZO etc.). In addition, it is set up to run in a single-user environment since none of the scratch globals support multiple-users.

All programs and globals in the utility package begin with the characters **%M** as will any future extensions. **No** data bases are needed initially since any necessary ones (such as the terminal definition global ^%MTC) will be created for you the first time the package is executed. Try to avoid creating programs or routines starting with %M since they might conflict with future expansions of the utility package.

All terminal I/O (including Console, Printer, Sequential CP/M files etc.) is conditioned via a terminal definition program and data base to provide considerable device independence. This package will make use of many special terminal characteristics (like HIGH and LOW intensity) if they are available. You can use one of the initial options to define or modify new terminals and tie a given terminal type to a specific I/O port. The first time this package is run, neutral default definitions will be used but you can pick the Terminal option to Define/Tie a new definition for your Console and Printer. This will clean up interactions with the program considerably.

A special feature available when Saving or Restoring either Routines or Globals is that they can be **transmitted between computers** by directing Input/Output of the Save file to MUMPS device number 5. The user need only establish a hardware link via the reader-punch port and %M will handle all the rest. **NOTE** that use of the reader/punch device requires a patch to the MUMPS operating system for a character ready check of the reader/punch port.

The later versions of %M (Versions 1.2 and above) also support **multiple volume global saves** where you can save large data bases on multiple floppy disk volumes. Routine saves are still output to single volumes since it is assumed the programmer can break the output up so that any one save will fit on a single floppy.

Standard Conventions:

There are a number of keyboard and name specification standards that are available in this package. Some of the conventions are always in effect (like ? or ??) while others (like ?L) are available only when they are logically needed.

? and ?? can be used to answer ANY question in the package to get additional HELP on how to answer that particular question. A single question mark (?) will get you one line of abbreviated Help. A double question mark (??) will get you the same abbreviated Help line along with any extended help that is available.

^ always lets you back up to the previous question or set of questions.

":" and "::" - Whenever you see a ":" it indicates that the program is waiting for a keyboard entry. A double colon ("::") will be displayed whenever a default answer has been supplied for you. If a default answer has been supplied, you can accept it by pressing the <RETURN> key only.

"?L" can be entered (when appropriate) to get a directory listing (either routines or globals depending on earlier choices). This option is available on all routine or global name specification questions.

"?S" is similar to "?L" but lists only the SELECTED routine or global names (more on selection later).

Routine/Global Name entry - when selecting routine or global names (flag them for Listing, Saving, Restoring, etc.) you can either enter specific names (one at a time), or use the "wild card" characters "*" and "?". The ? character in a name represents ANY **single** character while the * character represents **any number** of characters. For example:

"A*B"	Selects all names starting with "A" and ending with the character "B", regardless of name length.
"A?B"	Select all **three-character** names starting with the letter "A" and ending with "B".

Combinations of the above are allowed but avoid the special cases of ?, ??, ?L, and ?S as they conflict with other conventions already discussed.

The "'" (not) character is used to **de-select** one or more names from the list already selected. For example:

"A*"	Selects all names starting with "A"
"'A1*"	Then de-selects all of that subset that start with the characters "A1"

Global Reference - when working on globals, often you are given the option to specify an individual global or part of a global. This is used instead of the name question to allow you to pick individual nodes or branches of a global (say for listing or editing) rather than groups of global names (used when saving or restoring globals). When entering a Global Specification, the following rules apply:

NAME ONLY (^XYZ) implies that you want to process the ENTIRE global.
FULL REFERENCE (^XYZ(1,"abc",2)) implies that you want to work on that node ONLY.
PARTIAL REFERENCE (^XYZ(1, or ^XYZ(1) implies that you want to work on that node and on all of its descendents.

Functional Descriptions - Routines

Directory

A directory of all routines is maintained in ^%MU. This directory is updated when editing, restoring, or killing routines. If you choose the option to refresh the directory a temporary scratch CP/M file ("SCRATCH.DAT") is created and killed in the process of collecting the new directory. You get to specify the **Drive** and **output device** for the directory.

Edit

An option used to create and edit routines. The routine to be edited is loaded into a scratch global ^%MRE and all edit operations are performed on that global. The routine on disk is NOT updated until you give the editor the File command. When specifying a name to be edited, you can enter an asterisk (*) to indicate that you wish to use the routine currently in ^%MRE (saves loading time).

The editor also allows you to enter **COMMENTS** associated with labels in a routine. These comments are held in ^%M and do NOT take up partition space when the program is executed. They can be printed with the program when using the List option. They can also be **Saved** and **Restored** along with the routines.
To Enter/Edit comments for a given Label in the routine being edited, simply enter a semicolon and the Label name when the editor is looking for a command (e.g. you could enter >>**_;LABEL_** to Enter/Edit comments for the label LABEL).

List

Allows you to list routines (optionally with their comments) on the device of your choice. NOTE that you will want to use the Terminal definition option to establish the parameters of your devices (printer etc). If output to the console (or any CRT device) the program will pause with a keyboard read at the end of each page.

Find

Allows you to search for one or more strings (or patterns) in a specified set of selected routines. You specify where the search results are to go (if a CRT, program pauses at the end of each page).

Change

Allows you to change one or more strings in a specified set of routines to new strings. Output trace of the changes can be to the device of your choice (but no pauses at the end of screen for CRT's). You can set the TO string to a null (<RETURN> only) to delete the target (FROM) string.

Save

Allows you to save a specified set of routines (with an option to include the comments) to a specified output device. Usually the device is a CP/M sequential file (devices 2,3, and 4) and you will be asked for a Drive and File Name for the output file.

NOTE
You can specify device number **5** to direct the save operation out the punch port. This can be used in conjunction with a **RESTORE** operation on another computer (also using device 5) to effect computer to computer transfers of routines and globals. **However**, use of inter-computer transfers REQUIRES a patch to the MUMPS operating system to allow a character ready check from the Reader/Punch I/O port (see MUMPS installation notes from Dr. Walters).

Restore
Allows you to restore routines (and, optionally, comments) that were saved with the Save option. You specify the name(s) that are to be restored and whether or not you want the comments associated with those routines also restored. The directory is updated for all routines restored.
NOTE
You can restore from **device number 5** to read information being transmitted from another computer (see Save option above).

KILL
Allows you to KILL a **specified** set of routines. Also kills the routines entry in the directory (^%MU) and any comments that are associated with this routine (^%M).

Xtended Directory
Acts in a manner similar to the DIRECTORY command but also lists the 1st line of each routine and the 2nd line if it is a comment line. It is a **very** useful convention to have the first line of each routine contain (as a comment) the author's name (and, optionally, data last modified), and the 2nd line of each routine be a comment line that BRIEFLY describes the function of the routine. As an example, the following could be the first two lines of the routine %M:

```
%M    ;John Lewkowicz - NYS College of Vet. Medicine
      ;Programmer Utilities - Main Entry Point & Dispatch
```

The X function would then list both of these lines when displaying the directory.

Functional Description - Globals

Directory

Gives you a directory of all globals. This directory is maintained in ^%MU similar to the routine directory. Since it is much easier to create and delete globals and harder (if not impossible) to track these changes, it is recommended that you refresh the directory before you list it. When refreshing, the temporary scratch CP/M file "SCRATCH.DAT" is created and deleted. You will also get to specify the **Drive** and **output device** for the directory.

Edit

Lets you edit a global reference (entire global, specific node, or specific branch). Loops through your reference displaying each node and getting your changes. After each node you can specify whether to do the node **A**gain, go on to the **N**ext node, or **S**top editing this reference.

List

Lets you list a global reference (all or part of the global) to a specified device. If that device is a CRT (as defined by the Terminal definition option), program will pause at the end of each screen.

Find

Allows you to search for one or more strings (or patterns) for a given global reference. Output of search results is sent to the device of your choice with CRT's pausing at the end of each screen.

Change

Allows you to change one or more strings (FROM) to new strings (TO) for a global reference (all or part of a global). A trace of the changes is output to the device of your choice. There are NO pauses at the end of a screen on CRT's. The TO string can be a null ("") to DELETE the FROM string.

Save

Save one or more globals to a specified output device. Usually output to one of the sequential CP/M file devices (2, 3, or 4) where you will be required to enter a drive and file name specification for the output file.

When doing a global save, you can specify the maximum number of bytes to output to the output file. If you make use of this feature, you can create multiple output files (volumes) to save large data bases on small floppy disks. For example, to save 1 MBYTE worth of data bases on 8" SSSD floppy disks (storage capacity of 241 KBYTES) you could indicate that the maximum bytes to output is to be 240000 (question asked just before the "Ready to Start" question). The utility would then start a global save and continue until 240,000 bytes were output. At that time, the output file is closed and you will be advised to remove the disk, insert a new one, and define where the continued output is to go. Naming each of the continuation files is your responsibility. Each of the resulting save files is complete & standalone (including a definition and Date/Time lines) and can be individually retored using the RESTORE option.

Restore

Allows you to restore one or more specified global names from a file created with the Save option.

KILL

Allows you to kill a specified set of global names. Deleted globals are removed from the directory in ^%MU.

Xtended Directory

This command will provide you with a global directory on a specified output device and will list along with each entry the contents of the top data node if it is defined. Thus you can comment the data bases by defining a string at the name level as in the following example:

```
^%MTC="Terminal Definition & Port Assignments"
```

Terminal Characteristics

This utility allows you to define the characteristics of specific terminals and to tie those terminals logically to CP/M MicroMUMPS port numbers. Using these definitions in programs (for example this utility package %M) allows the programmer a certain device independence difficult to achieve when device specific codes are used in programs. Most of the terminal specific values that might be of use to a programmer are maintained by this utility and can be set up once at the beginning of a package. These parameters include:

Right Margin - The number of characters that will fit on one line. Use of this data allows programmer to insure that data fits on a line, to center data etc. As an example, to center a string of data on a line, the following code could be used (assuming string to be centered is in variable X and variable RM contains the right margin):

```
W !,?RM\2-($L(X)\2),X,!
```

Screen Length - The number of lines that will fit on a screen or page. Permits programmer to pause at end of screen or start new page (say with a header line) etc.

Form Feed - The character sequence necessary to clear the screen or start a new page. The MUMPS "#" does NOT always perform this function depending on the specific terminal. If this value is initialized correctly the programmer can always clear the screen with a:

```
W @FF   or a   W @IOFF
```

High Intensity - If a terminal supports multiple intensities (or multiple ribbon colors) it is possible to switch to the High (or say RED print) by writing out the character sequence in this variable with a:

```
W @LH   or a   W @IOLH
```

Low Intensity - String of characters that will switch terminals that support dual intensity back to LOW intensity with a:

```
W @HL   or a   W @IOHL
```

In **ALL** cases where outputting these strings or eXecuting strings of code, they are set up so that terminals that don't support a given feature will not be affected adversely. For example, a terminal that dosen't support dual intensity would have a high intensity and lo intensity string of "*0" which would cause a W *0 to the device, which should not effect the terminal.

Cursor positioning - On terminals that support cursor positioning, you can eXecute an indirect MUMPS command after setting the desired X and Y positions as follows:

```
S DX=10,DY=20 X XY...
      -or-
S DX=10,DY=20 X IOXY
```

NOTE - Under the current version of MUMPS (3.5), $X and $Y cannot be set to reflect the change effected by eXecuting the cursor positioning sequence. Later version will probably support this feature.

Terminal Type - The variable SUB (or IOST) contains a string that defines the terminal type. The 1st character defines the general type (C = CRT, P = Printer, O = Other) and the rest of the string is free text defining the specific type (like "DEC LA120").

The following entry points are available to the programmer for general access to these terminal values:

CURRENT^%MTC will open the device specified in $I and return the following parameters:

```
 FF - Form Feed character sequence
 HL - High intensity to Low intensity characters
 LH - Low intensity to High intensity characters
 RM - Right margin
 SL - Screen length
 XY - Cursor positioning command
SUB - Terminal Subtype
```

In addition to opening the device if necessary, an optional startup string will be sent to the device (can be used to setup initial default terminal conditions like high intensity, printer pitch etc.).

^%MTC will let the User pick which of the MUMPS I/O ports they want to direct data To/From. On return, the device has been opened, an optional startup string has been sent to it, and the following variables are available for use by the program:

```
IO - Device number picked (numeric)
     0 - Console
     1 - Printer port
     2 - Sequential CP/M file
     3 - Sequential CP/M file
     4 - Sequential CP/M file
     5 - Reader/Punch (Computer-Computer)
```

IOFF - Character sequence to clear screen/start new page
IOHL - Char. sequence to go to low intensity
IOLH - Char. sequence to go to high intensity
IORM - Right Margin (characters/line-1)
IOSL - Screen/Page length (lines-1)
IOST - Terminal subtype (see SUB above)
IOXY - MUMPS command string executed to position cursor

If the device selected is a CP/M sequential file then the default setup code for those devices (2-4) will be executed and will ask the user for a file specification (Drive and File Name), and then open that file.

Defining a Terminal

This option is used to define a new terminal type or to modify the definition of an existing terminal (say things like start up strings, right margin etc). The data maintained by this utility resides in ^%MTC. The global does not need to exist when you first run the %M utility package. If it does NOT exist, it will be initialized (by routine %M8) to include a number of standard CRTs and Printers as well as the CP/M sequential files. If the terminals on your system are in the file after this initialization, you can use the "Defining MUMPS Ports" option to attach a given terminal type to a specific I/O port.

This utility option allows you to define terminal specific characteristics. Enter a "?" or "??" if you need help on a specific question. After entering terminal specification, there will be NO apparent change on the terminal since you haven't tied the terminal to a Port number yet.

Example (ADM3A CRT) - {comments}

```
     Terminal Name: ADM3A  {name for terminal}
     Description: C-Lier Siegler ADM3A
                  {CRT-description}
     Form Feed character(s): *26,#
                                {# needed for MUMPS to clear $X,$Y}
     HI->LO character(s): *27,*41  {use *0 if none}
     LO->HI character(s): *27,*40  {use *0 if none}
     Right Margin: 79  {chars/line-1}
     Screen Length: 23  {lines/page-1}
     Cursor Positioning: W *24,"=",*DY+32,*DX+32
                         {move to position DX,DY}
     Startup Code: O %IO U %IO W *27,*40
                   {Start at HIGH intensity}
```

Defining a MUMPS Port

This function lets you tie one of the defined terminal types (like an ADM3A) and its description to one of the numeric MUMPS ports (Ø-4). The ports are defined as:

Ø - Console (input/output)
1 - Printer (output only)
2 - CP/M Sequential file (input/output)
3 - CP/M Sequential file (input/output)
4 - CP/M Sequential file (input/output)
5 - Reader/Punch (Computer-Computer)

After you have tied a terminal to a port, the program will issue a **D CURRENT^%MTC** and setup the correct parameters for the console.

Utility Data Bases

^%M	Holds comments associated with specific labels for specific routines. Entered/Edited when editing Routines.
^%MD	Defines the default Routine and Global drives to be used when running the %M utility functions. Initialized 1st time through by routine %M8. %M8 can be invoked directly (**D ^%M8**) to change these defaults.
^%MRE	Temporary scratch file into which routine lines are written when editing a routine. All of the actual editing of lines is done on data in this global.
%MTC	Contains device descriptions and current assignments of the MUMPS ports.
%MU	Utility global holding directories (Routine and Global) as well as Routine and Global restore specifications.

^%M

Description

This global contains free text comments associated with a given routine and label. The data is entered/edited when editing routines. It is used when listing routines and can be saved and restored when saving or restoring routines.

Structure

```
                                  |------|
                                  | %M   |
                                  |------|
                                     /\
                                    /  \ (ROUTINE)
                                   /
                                 |---|
                                 |---|
                                  /\
                                 /  \ (LABEL)
                                /
 ______________________________/
 |  |  |  |  |  |  |
 | (0)              | (n)
|-------|          |------|
|last line|        |Data  |
|  used   |        |------|
|---------|
```

ROUTINE is a Routine name
LABEL is a Routine line label
last line used an integer pointing to the last data node (n)
Data free text comments.

^%MD

Description

This global contains the Default Routine and Global drives for use by the Utility package %M.

Structure

```
^%MD=RG
```

R is the default Routine drive (A-P)

G is the default Global drive (A-P)

^%MRE

Description

This global holds the routine currently being edited. The editor is line oriented and the line number is the subscript into this global.

Structure

```
                                   |---------|
                                   | ^%MRE   |
                                   |---------|
                                       /\
                                      /
                                     /
    ---------------------------------/
    !   !   !   !   !   !   !
    !(Ø)                    !(n)
    !                       !
|----------|            |------------|
|last line |            | routine    |
|  used    |            | line       |
|----------|            |------------|
```

last line used is a number reflecting (n) the last data line in the file.

routine line a line of text from the routine

^%MTC

Description

This global contains terminal definitions and the current MUMPS port assignments. It is initialized by routine %MB and updated by routine %MTC (@ DEFINE and DEFAULT).

Structure

```
                    |--------|
                    | ^%MTC  |
                    |--------|
                       /\
                      /  \
                     /    \
                    /(n)   \ (TERMINAL)
                   /        \
           |----------|   |----|
           |TERMINAL  |   |Desc|
           |----------|   |----|
```

n is an integer MUMPS port number (0-6)
TERMINAL is a terminal name like ADM3A
Desc is a description of the terminal in the following format:

Desc="FF~HL~LH~RM~SL~XY~TD~SC" WHERE:

```
FF - Form Feed sequence
HL - Low intensity sequence
LH - High intensity sequence
RM - Right Margin
SL - Screen Length
XY - Cursor positioning code
TD - Terminal description (like C-ADM3A...)
SC - Startup code to be eXecuted when Opening
```

^%MU

Description

A utility global for use by the utility package %M. Holds directories (both routine and Global) as well as Routine and Global restore specifications.

Structure

```
                                   |-------|
                                   | ^%MU  |
                                   |-------|
                                      /\
                                     /
                                    /
  ----------------------------------/-------------------------------------
     |                                          |
     |  ("ROU" or "GBL")                        | ("RESR" or "RESG")
     |                                          |
    |--|                                       |--|
    |--|                                       |--|
     /\                                         /\
    /  \  (NAME)                               /  \  (n)
   /    \                                     /    \
  |---------|                               |-------------|
  |SELECTED|                                |SPECIFICATION|
  |---------|                               |-------------|
```

NAME is a Routine or Global Name (depending whether the subscript is "ROU" or "GBL".

SELECTED is a True/False indication (1/Ø) as to whether the name has been selected or not.

SPECIFICATION is a restore (either Routine or Global depending on whether the previous subscript is "RESR" or "RESG") specification in the following format:

```
NAME;IF
```

Where NAME is the actual User entry (like **'A?B*)** and IF is the indirect pattern match that will be used to test a target name (like **1"A"1E1"B".E**).

Save-Restore File Structure

The Save/Restore routines and globals are written out as sequential files to the designated devices. The 1st line is always a free-text description of the file and the second line is a free-text date and time. Both lines could be nulls.

```
GLOBAL                           ROUTINE
"Description"                    "Description"
"Date & Time"                    "Date & Time"
Global reference                 Routine name
   data element                     routine line
Global reference                    routine line
   data element                     routine line
 .                                     .
 .                                     .
 .                                     .
(Blank global reference             (blank routine line is used
 indicates done file)                to indicate end of routine)
                                 Routine Name
                                 .
                                 .
                                 (blank routine name indicates end
                                  of file)
```

You can use %M to save and restore both routines and globals between computers without the creation of a temporary file (using device # 5 on both computers). If you want to transmit or recieve routines or globals from OTHER systems, you can use the following sections of MUMPS code to transmit (TX) or recieve (RX) a line (L) to/from the other machine.

```
TX R X G TX:X'="" W L,! R X G TX:X'="" W $L(L),! R X G TX:X'=""
   Q
RX R !,A:10 E  W "*-ERROR-*",! G RX
R1 R !,X:10 E  W "*-ERROR-*",! G RX
   S X=$S(X'?1N.N:0,$L(A)'=X:0,1:1) W $S(X:"",1:"*-ERROR-*"),!
   G RX:'X
   Q
```

The code above assumes that the device in-Use for the read/writes is connected to the microcomputer via RS-232 line and that echo is turned off and that the default line length is set at 255 characters (i.e. O DEV:(255:1) for Intersystems MUMPS).

Utility Package Routines

%M	Main entry point. Gets primary options and dispatches control to other routines in the package as appropriate. Also process name specifications and flagging (selecting) entries in the appropriate directories.
%M0	Process **Directory** requests (including "?L" and "?S"), and both Global and Routine **List** options.
%M1	Controls the **Editing** of both Routines and Globals. The actual editing of routines is handled by the %ME... routines but global editing is done here.
%M2	Process **Delete** and **Find** options for both Routines and for Globals.
%M3	Routine and Global **Change** options are handled by this routine.
%M4	Does **Save** and **Restore** operations for both Routines and Globals.
%M5	Gets user options for **Terminal** and **Port** specifications. Dispatches control to %MTC for actual updates.
%M6	Does actual Routine/Global Save or Restore (%M4 used to collect options and generally setup for Save/Restore operation.
%M7	Handles eXtended Directory for both globals and Routines.
%M8	**Initializes** both the default drive data base (^%MD) and the terminal characteristics data base (^%MTC).
%M9	Displays help text for the Utility package %M.
%M10	Continuation of help text for Utility package.
%MEDT	Routine and comment file editor.
%MEDT1	Continuation of %MEDT.
%MEDTE	Extended explanations for %MEDT.
%MEDTE1	Continuation of %MEDTE.
%MTC	Terminal definition routine. Entry points at: %MTC To let user select a port. CURRENT^%MTC to get current device's parameters.
%MTCD	Continuation of %MTC with entry points to define characteristics of a terminal (**DEFINE**) and to assign a terminal to a MUMPS I/O port (**DEFAULT**).

MICROMUMPS
Validation Dialog

The dialog reproduced below represents a series of interactive statements and responses with MICROMUMPS. The user's entries are underlined. These dialogs test most language features of Standard MUMPS as implemented on the 8080/Z80, using for the most part, suggestions taken from the Programmer's Reference Manual (MDC Document 3/5). Comments elaborating the dialog are written in lower case.

Users wishing to verify the code on their own systems should be able to reproduce these dialogs as shown.Important discrepancies should be reported to:

Richard F. Walters
Dept Elec. Comp. Eng.
University of California
Davis, California 95616

Validation Dialog

```
A>MUMPS
Enter today's date (MM/DD/YY):12/01/82
Z80 MUMPS  Version 4.00 (version number may vary)

; test SET command

>SET X=2

>SET (A,B,C)=X WRITE A," ",B," ",C
2 2 2

>SET I=3,(I,A(I))=4 WRITE I,",",A(3)
4,4

>SET:X=2 X=4.5 WRITE X ; true postconditional
4.5

>SET X="X=3*2.5"

>SET @X WRITE X ; indirection in SET command
7.5
>

;test READ command

READ "Y?",Y:10 ; 10-second timeout.  No answer given
Y?
> ; note prompt on new line, after ten second wait.
  ; exact time may vary depending on cycle time of cpu.

>READ "X= ",X:10 WRITE !,X ;answer this time
X= 12
12

>WRITE $TEST ; see if $TEST set to 1 (timeout condition met)
1

>READ "B ",B:5 WRITE $TEST ;no answer this time
B 0

>SET Q="""NAME"",XYZ" ; set up for indirect read

>READ @Q
NAMEABC

>WRITE XYZ
ABC
```

Validation Dialog

;test KILL, VIEW (see User's Manual for VIEW implementation)

```
>VIEW 1 ;(local variables)
  XYZ         "ABC"
   Q          ""NAME",XYZ"
   B          ""
   Y          ""
   X          "12"
   C          "2"
   I          "4"
  *A          "2"

        3 "4"          ; subscripted variable value

     ;note:  sequence  and values may vary slightly depending  on
     exact duplication of dialog above.

>KILL X,C

>VIEW 1   ;now look at local symbols again
  XYZ         "ABC"
   Q          ""NAME",XYZ"
   B          ""
   Y          ""
   I          "4"
  *A          "2"
        3 "4"

>KILL (Q,I)

>VIEW 1
      Q         ""NAME",XYZ"
      I         "4"
>

      ;test IF, ELSE

>SET X=1,Y=2

>IF X=1,Y=3 WRITE "TRUE" ;(not true)

>IF X=1,Y=2 WRITE "CORRECT" ;(ok)
CORRECT

>IF X=2 WRITE "TEST"

>ELSE  WRITE !,"EXAM"  ;note 2 spaces required after ELSE

EXAM

>IF  WRITE !,"ARGUMENTLESS"
>
```

Validation Dialog

; test GOTO ;note that TAB character (<TAB> on printout) is used to write a command line in indirect mode. Tab must follow any label, and it must precede all commands.

```
>START<TAB>WRITE "HI THERE"

><TAB>SET A=2*4 WRITE A

><TAB>QUIT

ZSAVE START   ; store routine on disk

>GOTO START
HI THERE8

>ZREMOVE  ;
delete routine from workspace

>DO ^START
HI THERE8

     ; test FOR command; see p. 51, Reference Manual

>FOR I=1:1 WRITE I QUIT:I>2  WRITE "*"
               ;note two spaces required after QUIT command.
1*2*3

>FOR I=0.1:0.1:1.0 WRITE I,","
.1,.2,.3,.4,.5,.6,.7,.8,.9,1,

>FOR I=1:2:10 SET I=I-1 WRITE I ;reset index within loop
0123456789
>

     ;test XECUTE command

>SET A="SET XX=3"

>XECUTE A WRITE XX
3

>SET X=4,Y="ABC"

>XECUTE "SET B=X Q:X<3  WRITE Y" WRITE " OUT ",B
              ;note two spaces required after QUIT
ABC OUT 4
>

     ;test HANG and HALT commands

>HANG 10 ;should wait about 10 seconds before issuing prompt

>HALT    ;return to system
A>
```

Validation Dialog

```
     ;Test some of the special variables

A>MUMPS
Enter today's date (MM/DD/YY):      ;null response ok

Z80 MUMPS    VERSION 4.00

>WRITE $STORAGE  ;see how much workspace available
4096             ;4096 bytes available

          ;check $TEST

>SET X=1

>IF X=1 WRITE $TEST
1

>ELSE  WRITE "NO" ;two spaces required after ELSE
>      ;see also $T used above


      ;test $X,SY
>WRITE !,"123" SET A=$X WRITE " ",A
123 3

>WRITE:$X>72 !
>      ;no carriage return written

WRITE:$X>1 !

>      ;this time a carriage return is written

>WRITE $Y
2        ;exact response may vary

WRITE # ; form feed
        ;depending on device, the form feed character may or may
        not activate a move to the top of screen or of a new page.
        See SETMUMPS utility documentation to reconfigure your
        version for your terminal

>WRITE $Y ;this time $Y has been reset
0
>
```

Validation Dialog

Test of Standard MUMPS Functions

```
     ;test $ASCII; see p. 102, Reference Manual

>SET X="ABCDE"

>WRITE $ASCII(X,1)
65

>WRITE $ASCII(X,3)
67

>WRITE $ASCII(X) ;default to first value
65
>


     ;test $CHAR

>SET X=65,Y=66,Z="GOB"

>WRITE $CHAR(X)
A

>WRITE $CHAR(X,Y)
AB

>WRITE $CHAR(X,Y,67)
ABC

>WRITE $CHAR($ASCII(Z,1),$ASCII(Z,2),$ASCII(Z,3))
GOB
>
```

Validation Dialog

```
     test $DATA

>KILL Y

>WRITE $DATA(Y)
0

>SET Y=100 WRITE $DATA(Y)
1

>KILL A

>SET Y="AB" WRITE $DATA(Y)
1

>SET A(1)="TEST" WRITE $DATA(A(1))
1

>WRITE $DATA(A)
10

>SET B(1,2)="SAMPLE" WRITE $DATA(B(1,2)),!,$DATA(B(1))
1
10

>SET B(1)="ANOTHER SAMPLE" WRITE $DATA(B(1))
11

>KILL B(1,2) WRITE $DATA(B(1,2))
0

     ;test $EXTRACT

>SET X="ABCDE"

>WRITE $EXTRACT(X)
A

>WRITE $EXTRACT(X,1)
A

>WRITE $EXTRACT(X,1,4)
ABCD

>WRITE $EXTRACT(X,0,100)
ABCDE

>WRITE $EXTRACT(X,1.9)  ;should truncate to integer
A

>WRITE $EXTRACT(X,99)   ;null result

>SET A(1)="2BC"  WRITE $EXTRACT(A(1),A(1),3)
BC
```

Validation Dialog

```
      ;TEST  $FIND

>SET X="ABCAX"

>WRITE $FIND(X,"B")
3

>WRITE $FIND(X,"Z")
0

>WRITE $FIND(X,"A",2)
5
>

    ;test $JUSTIFY

>SET X=12.35

>WRITE $J(X,7)
  12.35       ;note default of 2 decimal places

>WRITE $JUSTIFY(X,5)
12.35

WRITE $JUSTIFY(X,7,4)
12.3500

>WRITE $JUSTIFY(X,7,1)
    12.4

     ;test $PIECE

>SET X="ABC*DEF",Y="*"

>WRITE $PIECE(X,Y,1)
ABC

>WRITE $PIECE(X,Y,2)
DEF

>WRITE $PIECE(X,Y,3)      ;null response

>WRITE $PIECE(X,"B",1)
A

>WRITE $PIECE(X,"/",1)
ABC*DEF
```

Validation Dialog

```
     ;test $RANDOM; values should differ each time it is invoked

>ZR    ;remove current program from workspace
>START<TAB>FOR I=1:1:10 WRITE $RANDOM(100)," "

>GOTO START
53 88 10 93 33 82 49 58 39 40   ;exact numbers will vary

>GOTO START
37 77 22 32 65 60 23 5 30 76

>GOTO START
46 95 63 6 12 53 20 9 61 50


     ;test $SELECT

>SET X=1

>WRITE $SELECT(X=1:8,2=3:0)
8

>WRITE $SELECT(X=2:8,2=3:0,1:3)
3

>WRITE $SELECT(X=3:8,2=3:0) ;no true value
NO TRUE VALUE IN SELECT ;error message from system


     ;test $TEXT; use LEXICON routine supplied with system
>ZREMOVE

>ZLOAD LEXICON  ;use Z command to retrieve routine from disk
       ;note: you may want to list the editor by typing ZPRINT.

>WRITE $TEXT(LIST) ;output line labelled LIST
LIST READ !,"WOULD YOU LIKE TERMS LISTED (Y/N)?",YESNO

>WRITE $TEXT(LIST+1)
     QUIT:YESNO'?1"Y".E
```

Validation Dialog

Test Global Variables, $NEXT and $ORDER

```
>VIEW 2  ;if response is "GLOBAL FILE NOT ON DISK",
          then SETGLOB has not been run.  See Installation Manual.

>SET ^A=1,^B=2 ;set global variables without subscripts

>VIEW 2 ;examine global directory again
    A         B       ;indicates both globals were set

>WRITE ^A," ",^B,!
1  2

>SET ^X(1)=3,^(2)=4,^("A")=5,^(1,5)=6 ;subscripts, naked refs

>VIEW 2   ;check globals again.
    A         B         X

>WRITE $NEXT(^X(-1))," ",$ORDER(^("")),! ;find the first numeric
     ; ($NEXT) or string ($ORDER) subscript at top level of ^X
1  1 ;first subscript of either type is 1

>WRITE $DATA(^(1)),!  ;check for descendants of ^X(1)
11                    ;descendants present

>WRITE $NEXT(^(1,-1))," ",$ORDER(^X(1,"")),! ;get value of subscript
5  5 ;result shows descendant is ^X(1,5)

>WRITE $DATA(^(5)),!  ;is there another descendant?
1   ;no

>WRITE $NEXT(^X(1))," ",$ORDER(^(1)),! ;^X(2) next at same level
2  2 ;next value of 2 confirmed

>WRITE $NEXT(^(2))," ",$ORDER(^(2)),!
A  A

>WRITE ^(1)," ",^(2),!  ;retrieve actual value
3  4

;demonstrate use of $ZORDER function

>SET Z="^X("""")"

>FOR I=1:1 SET Z=$ZORDER(@Z) QUIT:Z=""  WRITE Z," "
^X(1) ^X(1,5) ^X(2),^X("A")
```

```
     ;now test specific global kill
>KILL ^A

>VIEW 2   ;check global directory
       B      X ;note ^A gone

>KILL ^X(1) ;this kills X(1) and its descendants, X(1,5)

>WRITE $ORDER(^("")),!  ;get first ^X subscript
2      ;both X(1) and X(1,5) are gone

>KILL ^X  ;delete entire ^X global

>VIEW 2   ;final check of global directory
       B     ;only ^B left in directory
>      ;end of Validation Dialog.
```