MTA
Annual Meeting Proceed
June 1994

MIIS TO M CONVERSION AND LANGUAGE COMPARISON

Gary S. Shumway, CCP,MA,MPH,MBA
Shumway & Associates
21935 Van Buren St., Suite B-11
Grand Terrace, CA 92324

Phone: (909) 370-4336
Fax: (909) 370-4214

ABSTRACT

The conversion of dated mini-computer based MIIS systems to micro-computer based M systems is becoming less of an option and more of a necessity. This paper compares the syntax of the MIIS and M languages from the perspective of converting MIIS routines to M and provides suggestions for programming MIIS in a more M like fashion.

INTRODUCTION

MIIS, a dialect of M, is Medi-Tech's (Medical Information Technology, Inc.) proprietary operating system and applications language. MIIS was first licensed circa 1970 and is currently supported (though further development ceased circa 1982) on the DEC VAX, and DG Eclipse, MV, and AVION mini-computers. Over the years and on a number of platforms, MIIS has proven itself to be bullet proof, fast and efficient. Regardless, the days of MIIS are numbered due to the following: 1). powerful micro-computer hardware is rapidly decreasing in both the purchase price and maintenance cost, 2). the increase in demand by users for routines utilizing Graphical User Interfaces (GUI) and the X Windowing environment, 3). the increasing use of portable micros and their interconnectivity to networks, 4). the increased flexibility and expandabilty of M vs MIIS, 5). the plethora of 'off the shelf' software (both M and non-M) that are available for the micro-computer, 6). the increasing robustness and functionality of M and networking systems, 7). the cost of maintaining the MIIS OS vs a M system, 8). the availability of M programmers vs MIIS programmers, and 9). the lack of further developement of MIIS by Medi-Tech (in fact Medi-Tech is encouraging users to migrate to their 'MAGIC' product.)

Before converting MIIS routines to a M based system the following questions should be addressed: is the MIIS system simply to be converted to M or rewritten and updated, who will do the conversion, for how much and in what time frame. Those questions appear to be best answered on an individual site and application basis. The question, which is the focus of this paper, is what language differences are going to be encountered during the conversion of MIIS to M.

This paper is then a comparison of the two languages. As some sites may anticipate converting in the more distant future, some suggestions for making any new MIIS code as M like as reasonably possible has been included. It is not suggested that the more 'advanced' or unique capabilities of MIIS be avoided. Rather, it is suggested, that when generating new code or updating existing routines that the programmer be aware of some of the more salient differences and similarities between the two programming languages and write code that would minimize conversion difficulties. The following then is a summary of the differences and similarities in the two languages with suggestions for programming MIIS in a more M like fashion.

MIIS TO M COMPARISON

The format for the MIIS to M language comparison that follows is:
a) first the MIIS command/function is shown,
b) then a two dot separator,
c) the M command/function is then displayed,
d) the name of the command follows in parenthesis (if the command names are different then they are separated by two dots, and
e) finally the comparison and discussion.
For example, 'MIIS..M (name) discussion'. If 'No change.' is used in the discussion field it indicates that the code used in MIIS is directly convertible to M even though M may offer additional functionality for that command. If there is no

command for a particular command or function in one of the languages then a lower case 'n' (for none) will be inserted.

Note that this is paper is not meant to be an all inclusive comparison of all the nuances of the two languages nor a MIIS or M programming tutorial but rather a guide to encourage M like programming in MIIS, to address the more salient differences between the languages, and thus ease the conversion from MIIS to M.

The language comparisons are broken into the following categories: commands, functions, extrinsic variables, special variables and reserved words, arithmetic unary operators, arithmetic binary operators, arithmetic relational operators, string binary operators, string relational operators, and logical operators. Comparisons between the two languages are based on MIIS version S.MIIS.R-II-1.1 and the 1990 ANSII Standard M.

Commands

A..O/U (Assign .. Open/Use) The MIIS Assign command is functionally similar to the Use command in M. Note that ownership of the device must first be established with an Open command in M before Use can be executed. The Assign and Use commands may also have different device parameters depending on the implementation. The Assign command will have to be changed to Open and Use commands at conversion time. Note: the Open command in M establishes ownership of a device whereas the Open command in MIIS establishes ownership of a global (see also the MIIS Open command discussion).

B..B (Break) No change.

C..D (Call .. Do) The Call command in MIIS functions similarly to the Do command in M in that it transfers control from one part of a routine to that of another routine with control returning upon encountering a matching Quit. In MIIS the Call command 'calls' other routines and the Do 'does' other lines within the original routine. In M the Call and Do function are combined into the Do function, i.e., there is no Call function in M. Note that not only will the Call have to be converted to a Do but the syntax of the argument may also have to be edited. For example in MIIS a 'C pgm@tag' will have to be converted to 'D tag^pgm' in M. Local variable passing between routines compound the problem as MIIS converts the variables to %0 through %n whereas M requires a receiving list of variable names. These differences will have to be dealt with upon conversion.

C..L (Close .. Lock) The Close command in MIIS is used in conjunction with the Open command. Open is primarily used to establish ownership of a global thus avoiding conflicting updates. The Open command creates an entry in the system's open table indicating that a particular job has Opened a global node while the Close command deletes that entry. The corresponding command in M is Lock; where 'Lock +gbl' is comparable to the MIIS Open command and 'Lock -gbl' is comparable to the MIIS Close command. Also see the MIIS Open and Unassign commands discussion. Note: the use of the Close command in M Closes (releases) ownership of a device not a global (see the MIIS Unassign command). The difference in use of the Close between MIIS and M will have to be resolved at conversion.

D..D (Do) The functional use of Do in MIIS is a syntactical subset of the use of Do in M. There should be no difficulties converting the use of Do in MIIS to M but the reverse does not hold true.

E..E (Else) The Else command in M permits conditional execution dependent upon the value of the $Test variable. In MIIS the Else command is dependent upon the previous If statement's truth (the If-Switch). As in MIIS the If command in M is evaluated and the $Test variable is set accordingly. There should be no conversion difficulties with the Else command if the $Test variable is not reset by code prior to executing the Else command.

E..n (Erase) The Erase command in MIIS is typically used in program editing by the programmer to erase lines of or entire routines from the partition. Erase may also be used by more utility type routines but should pose minimal difficulties in the conversion of application routines. Note: M vendors typically supply a 'Z' command, e.g., ZR, which is functionally similar to the MIIS Erase command.

(File) The File and File pgm:X commands for storing and deleting programs in MIIS are seldomly used within application routines. The lack of support in M for this command should not be a big problem but any uses of File in MIIS routines will have to be dealt with at conversion time. Note: M vendors typically supply a 'Z' command, e.g., ZS, which is functionally similar to the MIIS File command.

(For) The argumentless F command is the File command in MIIS whereas the argumentless F command in M is an endless For loop. Other than the argumentless F command the For commands are the same in MIIS as in M and should pose no problems at conversion.

G..n (Go) The Go command in MIIS is used to resume instruction execution after encountering a Break. The Go command may only be issued in direct mode with no arguments thus should pose no problem upon conversion.

G..G (Goto) The use of Goto command in MIIS should not pose a problem in the conversion to M, but the reverse is not true.

H..H (Halt) No change.

H..H (Hang) No change.

I..I (If) No change.

J..J (Job) The routine to be Jobbed in MIIS must be filed as 'pgm#', i.e., have the final character of it's name being the pound sign. The pound sign ending is not necessary in M. A routine in M is Jobbed by the syntax 'J ^pgm' or 'J line^pgm' or 'J line^pgm(v1,v2)' and no local variables are passed by default. In MIIS routine names are not referenced with the '^' prefix, lines are referenced after the routine name, e.g., 'J pgm#@line', and there is not specific parameter passing in MIIS but rather the passing to the Jobbed routine the $Q and $R values, principal device, Namespace names, $G reserved word, and the entire local symbol table of the routine doing the Jobbing. These differences in the use of Job will have to be resolved at conversion.

K..K (Kill) The use of the 'K *^gbl' syntax in MIIS and the fact that 'K ^glb(node)' kills all descendant nodes in M but not in MIIS could cause problems upon conversion. Other than the above caveats the Kill command syntax is the same.

.n (Load) The Load command is seldomly used from within application routines. If used, the Load command will have to be edited at conversion time replacing 'L' with the M vendor supplied 'Z' command, e.g., 'ZL'.

M..n (Move) The Move command is usually used in MIIS during the creation of a program enabling the programmer to move the program line pointer to the desired line. The Move command is not supported in M as a full screen editor is typically employed in program creation rather than the line editor supplied by Medi-Tech in MIIS.

N..n (Namespace) The MIIS Namespace command has no M equivalent. The use of Namespace in routines that are to be converted to M should be avoided to minimize conversion difficulties. The Namespace command is useful and more efficient (code and compute time) but not necessary and it's use is not difficult to avoid.

O..L (Open/Close .. Lock) The file locking Open and Close commands in MIIS will need to be converted to the M Lock command at conversion time. Also see the MIIS Close command discussion.

O..G (Overlay .. Goto) The MIIS Overlay is similar in function to the M Goto command. The syntax of the argument may have to be edited during conversion.

V..N (Variablize .. New) The M command that comes closest in function to the MIIS Variable command is the New command. The New command temporarily removes variables which are restored upon encountering a Quit from out of a Do or Xecute (see a M language reference for a full description). The Variable command creates another local copy of the variable equal to a specified value (similar to a set). Variablized variables are stacked and are killed in order by each Kill command. Variable passing in application routines will likely be a source of difficulty upon conversion.

n..O (Open) The equivalent to the Open command is not required in MIIS when establishing ownership of a device but an Open will be needed for device use after conversion to M. Also see MIIS Assign command discussion.

P..n (Print) The Print command in MIIS is usually supported in M by a vendor supplied Z command, i.e., 'ZP'.

(Quit) The Quit syntax is the same in both languages except that M requires two spaces after a Quit (or Quit be the last command on a line) and Quit returns a value of the argument following the Quit in extrinsic functions. If the MIIS programmer makes a habit of placing two spaces after the Quit command there would be no code change needed during conversion from MIIS to M.

(Read) The variable read, timed read, character read, and string/variable read commands function the same in both languages. Note that M also supports the syntax 'R VAR#n' where n is a numeric expression which specifies the maximum number of characters to be input. The Read command itself should cause few problems during conversion.

(Set) The Set syntax is the same in both languages except that M supports 'S (I,J,K)=1' and MIIS does not. Additionally, MIIS supports the 'S *^gbl' command for copying data structures and for optimizing the physical storage of files where M does not. If the 'S *' command is used in MIIS routines then the incompatibility will have to be resolved during conversion.

T..n (Transfer) The MIIS Transfer command is used to copy blocks from the disk to the view buffer and visa versa. M vendors typically supply a Z command with similar functionality. The use of the Transfer command is rare in MIIS application routines.

U..C (Unassign .. Close) The use of the Unassign command in MIIS is to relinquish control of a device. The M Close command parallels that functionality.

n..V (View) View provides an access point within M for the examination and/or modification of implementation-specific information. Also see the MIIS $View function.

W..W (Write) The Write command is the same in either language except for the special formatting supported in MIIS with the :D, :L, :R, and :T suffixes. The use of the Write formatting commands will have to be recoded to utilize the M $Justify and $FNumber commands or to access formatting sub-routines or utilities.

X..X (Xecute) The Xecute command itself is supported in both languages and does not need altering but the arguments Xecuted may have to be modified upon conversion.

Functions

$H..$A ($Hash .. $Ascii) A change of the function name $H to $A will need to occur during the conversion process though the general syntax of the functions remains the same. Note that M returns a -1 for $A("") whereas MIIS returns a 0 and this may have to be resolved in code.

$C..$C ($Character) No change. Note that M supports multiple ASCII codes, e.g., $C(65,66) yields 'AB' whereas MIIS supports only a single ASCII code per function.

$D..$D ($Define..$Data) The $Define function in MIIS returns a 0 if the variable is not defined (FALSE) or a 1 if it is defined (TRUE). In M the $Data function returns a 0 if the variable is undefined, 1 if the variable is defined but has no descendants, 10 if the variable is not defined but has descendants, and 11 if the variable is both defined and has descendants. The return of 0 or 1 in either language is the same but M returning a 10 could be a source of difficulty in conversion. Additionally, an optional second argument is supported in MIIS, which if present, is any identifier except a packed-data name and is the name of the variable in which to store the value assigned to the first argument (variable). To ease conversion the second argument should not be used.

$E..$E ($Extract) No change.

$F..$F ($Find) In MIIS if the substring is not found in the string then a value of 1 is returned; in M a value of 0 is returned.

n..$FN ($FNumber) The $FNumber function is used by M for formatting numbers. $FN can be used in conjunction with $J to supplant the use of some Write argument:suffix commands in MIIS. Also see the MIIS Write command and the M $Justify commands.

$G..$Q ($Globalget .. $Query) Rather than treating a file as tree structured, e.g., $N, the $G function in MIIS treats the file in a sequential or serial manner. The use of $G in MIIS necessitates the conversion to $Q or $O (non-sequential) in M at conversion because the syntaxes are greatly different. Some vendors also supply a 'Z' command for serial global access. Note the difference in usage of $Globalget in MIIS and $Get (see below) in M.

n..$G ($Get) $G in M returns the value of a variable or a null string if the variable is not defined.

$I..n ($Inverse) Not supported in 1990 ANSII M and will have to be dealt with at conversion.

n..$J ($Justify) MIIS uses :D, :R, :L, & :T for similar (though a superset of M) functionality. If possible, it is suggested that '?' (tab) and $E be used in MIIS. It is likely, due to the inconvenience of using other formatting techniques in MIIS that the majority of the formatting changes will have to occur upon conversion. Also see the MIIS Write command and M $FNumber discussions.

$L..$L ($Length) No change. Note that M supports a second argument which MIIS does not.

$M..na ($Multiprecision) Used in MIIS for full division and the formatting of results. The $M function will have to be dealt with at conversion by additional M code. Also see the M $FNumber function.

$N..$N ($Next) The use of $N in M will not be supported in future versions of M. Thus the $N function in MIIS will need to be converted to M's $O function which returns a null string (as $N in MIIS does to indicate end). Note that $N in M returns a -1 not a null string. Also see the M $O function discussion.

$P..$P ($Piece) $Piece is very similar syntax in both languages except that the delimiter in M (if a literal) must be surrounded in quotes whereas the use of ';', ',', and '.' in MIIS do not require quotes (other delimiters do require the quotes). The insertion of quotes in MIIS regardless of the decimeter would ease the conversion of $P from MIIS to M.

n..$Q ($Query) $Query in M returns the next subscripted value (in collated sequence) that has a value. The syntax of $Q is different enough from the MIIS $Get function that it is mentioned separately. See also the MIIS $G function discussion.

$R..$R ($Random) No change.

n..$S ($Select) No MIIS equivalent function.

$T..$T ($Text) Syntactical equivalent in both languages (except possibly the argument). The MIIS programmer must be aware that $T(+0) returns the first line of a program but in M that line may only be the program's name whereas the line containing the program name, description, date created, programmer initials, etc. is $T(+1). It's safer to use $T(pgm) where pgm is the program name when accessing the program description, programmer, etc.

n..$TR ($TRanslate) No MIIS equivalent function.

$U..n ($Update) $Update in MIIS increments the var in the expression $U(var,exp) by the numeric value of exp. If exp is not present the default value is 1. $Update takes place in a time-sharing protected fashion so that it is not necessary to use Open statements when $Updating global nodes. This time-sharing protected updating is the only critical difference from using the argument var=var+exp in M. The 'L'ock function in M will have to be used with $U when updating globals to retain the same functionality.

n..$V ($View) No direct MIIS equivalent though there is a View Buffer. $View is an implementation-specific function.

Extrinsic Variables

($Password) MIIS supports the partitioning of the disk or disks into unique sections or passwords. Unless the routine or global is a system routine or global (name preceded by a '%') then routines or globals are only accessible from within the specific password. Thus two routines can have the same name yet be filed in different passwords and thus co-exist. Depending on system set-up M also supports multiple partitioning of a disk or disks but does not use $Password.

($Queue) $Q is a MIIS reserved word containing an integer value of 0 to 7 which is explicitly Set to the appropriate value. The eight different wait queues are serviced by the CPU in the order of their number. A similar variable or command may be supported by the M vendor but Standard M does not have a similar variable thus the use of $Q should be avoided in MIIS though the conversion would not be difficult.

($Resource) The $Resource MIIS reserved word takes on values of 0 to 3 and indicates how long the job will run when run. Standard M does not have a similar function though a M vendor may support similar functionality. Use of $Resource in MIIS code to be converted to M should be avoided.

$T..$H ($Time .. $Horolog) As used by MIIS, $Time is the number of seconds since January 1, 1976, i.e., 12:00AM on January 1, 1976 is 0. In M $Horolog consists of two comma pieces where the first is the number of days since December 31, 1840 and the second piece the number of seconds since midnight. The differences in $T and $H will have to be resolved upon conversion.

Special Variables and Reserved Words

($Io) In both languages $Io is the unique numeric identification of the current input/output device. Depending upon the application and system set-up the $Io for the devices would likely be different for the two systems. Thus any routines relying on specific $Io's may have to be converted.

$J..$J ($Job) Each process in MIIS and M has it's own unique numeric job number ($J). Except in the unlikely event that routines require specific job numbers, there should be no difficulty in conversion for this special variable.

$S..$S ($Storage) In both MIIS and M, $Storage is the number of free (unused) bytes in a job's partition. Except that M partitions are typically larger than MIIS's 2048 byte partitions there is no change.

n..$T ($Test) $Test in M is similar to the IF-SWITCH in MIIS though with more functionality.

$X..$X ($X) No change. The first column is 0 not 1 in both languages.

$Y..$Y ($Y) No change. The first row is 0 not 1 in both languages.

?..? (Tab) No change. Note that the first column is 0 not 1 which is the same in both languages.

!..! (Line Feed) No change.

#..# (Form Feed) No change.

@ (Indirection) The use of indirection is one of the more powerful elements common to both languages. The indirection syntax is nearly the same in either language except for the use of '_' (underline) in MIIS and '@' (ampersand) in M. When using undefined variables in a MIIS indirection argument it is recommended that the argument should be enclosed by parentheses to make the syntax more acceptable for use in M. For example in MIIS the statement 'S A="XY",XY="MN",_("B="_A)' should be used instead of 'S A="XY",XY="MN",_"B="_A' as M supports the former. Name indirection and pattern indirection is syntactically the same in either language.

Arithmetic Unary Operators

A '+' as the first character of an argument causes both MIIS and M to take the numeric interpretation of the argument. The use of '+' is a mixed bag in MIIS. That is, if 'A=7' and 'W +A' is executed then the number 7 is written. If 'A="34B"' and 'W +A' is executed MIIS will return a ? Syntax error where M will return the number 34. Regardless, the syntax of the '+' unary operator as used in MIIS makes it compatible with M but not necessarily visa versa.

A '-' as the first character of an argument causes both languages to take the numeric interpretation of the argument and negate it. The '-' unary operator functions the same in either language, e.g., if 'A="34B"' executing 'W -A' produces a -34 in either language; thus no change.

Arithmetic Binary Operators

(Addition) No change.

(Subtraction) No change.

(Multiply) No change.

/.\ (Integer Divide) The result of the division is truncated to an integer. Note the use of the back slash in M vs the forward slash usage in MIIS.

n../ (Full Division) The use of $Multiprecision in MIIS will produce a full division result with the use of '/'. Note that MIIS does not support the use of '\' and that $M is not supported in M. Thus, at conversion time the $M function must be eliminated and the argument within the $M() function rewritten.

(Modulo) No change.

Arithmetic Relational Operators

(Less Than) No change.

(Greater Than) No change.

String Binary Operators

(Concatenation) No change. Note that MIIS also supports the period as the concatenation operator while M does not support the period as the concatenation operator. It is advised that the MIIS programmer use the '_' operator rather than the '.'.

String Relational Operators

(Equals) No change.

n. (Contains) Not supported in MIIS thus no conversion problem.

n. (Follows) Not supported in MIIS thus no conversion problem.

(Pattern Match) The pattern match codes are a mixed bag; three codes are the same (A, N, & P), the code U in MIIS is the same as E in M, and three codes are supported in M but not MIIS (C, L, & U). The following are the codes used by either language:

A..A (Alphabetic characters - upper & lower case)
n..C (the 33 control characters)
U..E (for the entire set of 128 ASCII characters)
n..L (for the 26 lower-case alphabetic characters)
M..AN (for any alpha or numeric characters)
N..N (for the 10 numeric characters)
P..P (for the 33 punctuation characters)
Q..AP (for any alpha or punctuation characters)
T..NP (for any numeric or punctuation characters)
n..U (for the 26 upper-case characters)

Note that the programmer can use a combination of pattern match codes for the three pattern match codes (M, Q, and T) not directly supported in M. Note also there may be a difference in the pattern match syntax between the two languages. For example, '?1N' functions properly in both languages, whereas '?3.5N' (checks for 3, 4 or 5 numeric characters) and '?.AP' work in M but not MIIS. Fortunately, the MIIS pattern match syntax works in most cases in M but not visa versa, e.g., '?2A1N' functions properly in either language.

Logical Operators

&..& (And) The use of the logical And is the same in both languages. Note that the '&' is also used in MIIS as a minimum operator where '2&3' yields 2. The use of '&' as a minimum comparison operator will have to be changed during conversion.

(Or) The use of the logical Or is the same in both languages. Note that the '!' is also used in MIIS as a maximum operator where '2!3' yields 3. The use of the '!' as a maximum comparison operator will have to be changed during conversion.

(Not) No change.

SUMMARY

In the author's experience, some of the more frequent sources of frustration to the experienced MIIS/novice M programmer in writing M code are, a) the two spaces needed after a Quit in M (see Quit above), b) the use of Do instead of Goto, c) the pattern match syntax, d) character output formatting, e) device assignment and usage, and f) the need to previously define variables in M (unless they are used with $Data or in variable passing). Most of the other syntaxes and conventions are reasonably easy to remember and the conversion of a MIIS programmer to a M programmer is not a difficult one.

The above comparison suggests that the conversion of MIIS to M, though not painless (costless), can be accomplished with minimal retraining of the MIIS programming staff, enable the use of more off-the-shelf software and less expensive and more expandable hardware, and enable the use of the advanced M features on micro-computers, i.e., GUI and X Windows. Converting from MIIS to M, though not a painless operation, is a task whose time has come.

REFERENCES

MIIS Reference Manual, Version S.MIIS.R-II- , Medical Information Technology, Inc., 1980.

Standard MUMPS Pocket Guide, MUMPS Users' Group (MTA), 1990.