1. Identification

1.1 Title:

Data Record Functions

1.2 MDC Proposer and Sponsor:

Proposer:

David J. Marcus
Micronetics Design Corp
1375 Piccard Drive
Rockville,MD, 20850
Telephone: (301) 258-2605
FAX: (301) 840-8943
Email: djmarcus@micronetics.com

Sponsor:

David J. Whitten
Houston VAMC
2002 Holcombe
Houston,TX, 77030
Telephone: (713) 791-1414 ext 6116
Email: whitten@netcom.com

1.3 Motion:

The proposer recommends that this document be accepted for MDC Type A status superseding the documents: X11/SC13/TG3/96-2, X11/SC13/TG3/95-4, X11/95-67, X11/SC13/92-19, X11/91-28, X11/SC13/TG3/91-2, X11/SC1/TG15/91-2, X11/SC1/88-55,X11/SC13/TG3/97-2,X11/SC13/TG3/97-1,X11/SC13/TG3/1998-3

1.4 History:

Date	Document	Action
June 28,1998	X11/SC13/TG3/98-3	Proposal Passed SC A, 10:3:2. Amendments in History.
June 27,1998	X11/SC13/TG3/98-2	Proposal as modified by Task group under direction of MDC Document editor
Sept 1997	X11/SC13/TG3/97-1	Subcommittee vote for Replacement SC B on X11/SC13/TG/97-1 passed (9:6:4)
March 1997	X11/SC13/TG3/96-2	Subcommittee vote for Replacement SC B on X11/SC13/TG/96-2 passed (13:0:9)
Sep 1996	X11/SC13/TG3/96-2	Proposed as replacement Status B, clarify formalism and examples
1996	X11/96-60	External Proposal using angle bracket syntax of previous version of this proposal

Date	Document	Action
Oct 95	X11/SC13/TG3/95-4	re-write as two functions $RE, $RP replacement type B (11:10:5)
Apr 95	X11/95-67	re-write as single $RECORD function
Jan 95	X11/SC13/92-19	Subcommittee vote to demote to SC passed (15:0:1)
Mar 93	X11/SC13/92-19	Proposed as SC Type A (improved language)
June 92	X11/SC13/92-19	Proposed as SC Type A (improved language)
Feb 92	X11/91-28	Proposed as Type A (not addressed)
Oct 91	X11/SC13/TG3/91-2	Voted to type B status
Oct 91	X11/SC1/TG15/91-2	New Document to accomodate proposal format
Dec 88	X11/SC1/88-55	Approved as Type C document. Task Group 15 created to move to type B

1.5 Dependencies:

This proposal is dependent upon:

No other documents

Proposals depending on this proposal:

No other documents

2. Justification

2.1 Needs

M[UMPS] is well known for its strengths of data storage and retrieval. Avoiding the "data declaration" is often touted as another of the language's strengths. However once a "record" is retrieved from mass storage, there is a shortcoming. The M[UMPS] language sorely lacks built-in functions which operate on 'records' (data aggregates). Minimally, the M language needs the ability to assemble data records into a single string (data fan-in) and conversely extract data from a single string into selected variables (data fan-out). This proposal promotes a method that facilitates the multiple assignment process.

Specifically these benefits should be realized:

1. decrease in amount of code
2. increase organization and general readability of code
3. increase efficiency of M[UMPS] implementations by reducing the number of scanning operations (to a single one) to find delimiters for multiple pieces.

2.2 Existing Practice in the area of the Proposed Change

Specialized, home grown, M[UMPS] routines are developed to achieve desired functionality. These are generally slow and error prone (and inelegant for a language like M which proclaims to be the premier data and database manipulation language). Some vendors have implemented a limited form of

aggregation and decomposition, but in general there is no uniformly accepted solution. As the limited forms already existing have been well recieved, this document attempts to formalize and standardize this functionality with an easier to process form.

3. Description of the Proposed Change

3.1 General Description of the Proposed Change

This proposal introduces a method for manipulating multiple pieces and characters from a string by extending the syntax of the SET command. It introduces two new functions which define the aggregations (composition and decomposition) of data records. The new funtions, $DPIECE and $DEXTRACT can be used in expressions or on the left hand side of the equal sign in a SET command.

By using a function, the left hand side of a SET command is independent of the right hand side of the equal sign. It strongly parallels the $PIECE and $EXTRACT functions which can appear on either side of the equal sign in a SET command. Furthermore, the functional approach allows for future enhancements for user defined record templates by introduction of an appropriate function that generalizes $DPIECE and $DEXTRACT.

The $DPIECE and $DEXTRACT functions use a template to define how the "record" is assembled or taken apart. The template is defined in a way which is well suited to retrieval of the template from a data dictionary. This allows the actual record layout to be isolated in the dictionary rather than be distributed in every nook and cranny of the application. (a maintenance nightmare).

3.2 Annotated Examples of Use

M Code	Commentary

`SET DLM="^"`
`SET X=$DPIECE("",DLM,A+3,B,,D,,,)`
`; $DPIECE function with empty`
`; extractfields`
	Defines a record format specifier which relies on `$PIECE` to define the record fields. The piece delimiter is the value of `DLM` . The first argument specifies the string in which the substitution occurs (the empty string in this example). The value of `A+3` is used to replace the first `DLM`-piece, `B` replaces the second piece, the third piece is untouched (remains null due to the initial empty string), The value of `D` replaces the 4th piece, and pieces 5 through 7 inclusive are left equal to null (but the delimiter is inserted). The resultant string is assigned to the variable `X` .

`SET DLM="^"`
`SET X=$DP(X,DLM,A:5,,,$$ABC)`
`; example of $DPIECE function with`
`; with initial value and with`
`;fieldindex`
	The initial value `X` specifies the string in which substitution occurs. The `A:5` indicates the 5th piece. Each successive function argument applies to the next piece unless a fieldindex is specified that explicitly overrides what index is used. Thus pieces 6 and 7 are skipped (due to the commas) and remain at the values initially specified in `X`, and the value of `$$ABC` replaces the 8th piece.

M Code	Commentary
SET DLM="$" SET $DP(DLM,A,B:4,,D)=X ; $DPIECE as a setrecord value ; and using fieldindex	Uses the value of xto supply 'DLM' pieces. The first piece is assigned to variable A, the 4th piece (due to the :4 modifier on B:4) is assigned to the variable B, and the 5th piece is skipped with the 6th piece assigned to the variable D.
SET FMT="5,,,-6,,,9" SET X=$DEXTRACT("",FMT,A+5,B,,,,D) ; $DEXTRACT with extracttemplate	The first argument specifies the starting string. The second argument specifies the individual field widths. Unless specified otherwise, each successive field is of the same width and alignment (implied by the sign of the number) as its predecessor field. Thus the first field width is 5 (left-aligned) which due to commas, also applies to fields 2 and 3. Field 4 specifies a field width of 6 but is right aligned due to the minus). Note that a single format specifier of "5" would have made all fields 5 characters wide, and left aligned. A value of "-5" would have specified right alignment for all fields.
SET FMT="5,,-7,,4" SET $DEXTRACT(FMT,^DB,A,B:4,,D)=X ; $DEXTRACT as setrecord ;with extracttemplate	Uses the value of X to supply $EXTRACT substrings (fields) to the specified variables. Global variable ^DB is assigned the first 5 characters of X, local variable A is assigned the next 5 characters, B is assigned the 4th field which is 7 characters, but right aligned, the 6th field is assigned to the variable D.

3.3 Formalization

{ Add $DEXTRACT and $DPIECE as intrinsic functions }

In Section 7.1.5 Intrinsic functions, (X11.1 standard: X11/TG6/98-1) add DE[XTRACT] and DP[IECE] to the list of choices for functionname.

Add section
7.1.5.18 $DEXTRACT

$DE[XTRACT] ([initialrecordvalue] , extracttemplate , [,] [recordfieldvalue])

initialrecordvalue ::= expr

extracttemplate ::= expr V extractfields

extractfields ::= [,] | [-] fieldwidth [: fieldindex] |

fieldwidth ::= intlit

fieldindex ::= intlit

recordfieldvalue ::= expr [: fieldindex]

This function assembles the exprs of the recordfieldvalues into a single value. The value of initialrecordvalue is used as the starting value to which the

recordfieldvalues are applied. If initialrecordvalue is omitted, the empty string is used.

The extracttemplate specifies the $EXTRACT partitioning (fieldwidths and alignments) of initialrecordvalue into consecutive fields. Unsigned values specify width for left-aligned fields. Negative values specify width (absolute value of fieldwidth) for right-aligned fields. The fieldindex specifies the relative field number. If omitted, it defaults to the next successive value. For omitted recordfieldvalues the corresponding field is obtained from the initialrecordvalue. Although recordfieldvalue is optional, at least one recordfieldvalue (not necessarily the first) in the list must be nonempty.

Left-aligned fields are padded on the right with $C(32) or truncated on the right as needed. Right-aligned fields are padded on the left with $C(32) or truncated on the left as needed.

Assignment to fields proceeds in a left-to-right fashion. If a field is referenced multiple times, the rightmost value is the final value of the field.

Add section
7.1.5.19 $DPIECE

$DP[IECE] ([initialrecordvalue] , piecedelimiter , L [recordfieldvalue])

piecedelimiter ::= expr

This function assembles the exprs of recordfieldvalues into a single value. The value of initialrecordvalue is used as the starting value to which the recordfieldvalues are applied. If initialrecordvalue is omitted, the empty string is used. The piecedelimiter specifies the relative field number. If omitted, it defaults to the next successive value. For omitted recordfieldvalues the corresponding field is obtained from the initialrecordvalue. Although recordfieldvalue is optional, at least one recordfieldvalue (not necessarily the first) in the list must be nonempty.

In Section 8.2.22 SET,

Add setdextract and setdpiece to the list of choices for leftexpr.

setdextract ::= $DE[XTRACT] (extracttemplate , L [recordfieldglvn])

setdpiece ::= $DP[IECE] (piecedelimiter , L [recordfieldglvn])

recordfieldglvn ::= glvn : [fieldindex]

Add

6) For each setleft that is a setdextract, the expr is used as the starting value, which is partitioned into consecutive $EXTRACT fields using extracttemplate (see definition of $DEXTRACT section 7.1.5.18). Each glvn is assigned its corresponding field extracted from expr. The values corresponding to omitted glvns are ignored. The fieldindex specifies which field is to be assigned to the glvn. If omitted, the next successive field index is assigned. Although recordfieldglvn is optional, at least one recordfieldglvn (not necessarily the first) in the list must be nonempty.

7) For each setleft that is a setdpiece the expr is used as the starting value, which is partitioned into consecutive $PIECE fields using piecedelimiter (see definition of $DPIECE section 7.1.5.18). Each glvn is assigned its corresponding

field pieced from expr. The values corresponding to omitted glvns are ignored. The fieldindex specifies which field is to be assigned to the glvn. If omitted, the next successive field index is assigned. Although recordfieldglvn is optional, at least one recordfieldglvn (not necessarily the first) in the list must be nonempty.

4. Implementation Effects

4.1 Effect on Existing User Practices

These functions should increase reliability of code and decrease the maintenance burden. These operations are very commonly used, and when used in new programs should have a significant impact on applications. As this capability does not exist currently in a standard fashion, this change is backwards compatible.

4.2 Effect on Existing Vendor Practices

Several vendors have said this approach is preferable as it does not depend on the left and right sides of the SET command passing 'hidden' variables for delimiters, etc.

4.3 Techniques and Costs for Compliance Verification

None mentioned here.

4.4 Legal Considerations

None known.

5. Closely Related Standards Activities

5.1 Other X11 Proposals Under Consideration

None.

5.2 Other Related Standards Efforts

None.

5.3 Recommendations for Coordinating Liaison

None.

6. Associated Documents

None.

7. Issues, Pros and Cons, and Discussion

June 28, 1998: Boston, SC A Amended to change occurrences of RPIECE to DPIECE, and REXTRACT to DEXTRACT. Subcommitee Passed as amended to SC A 10:3:2

Document X11/SC13/TG3/97-2 Amended to include suggestions of taskgroup and of MDC document editor. new document published as X11/SC13/TG3/98-2.

PRO CON

1. Expresses subcommittee will
2. Has been implemented (functionality not syntax)
3. Useful

1. Too complex
2. Not Symmetric Between setleft and setargument
3. Introduces new concept to language(multiple parimeters)

Task Group 3 has addressed the CON list and offers the following rebuttals:

Con 1
Complexity is a matter of perception. These functions localize complex operations in a straightforward manner, and hence should reduce overall program complexity.

Con 2
The need to establish a default value is appropriate for the right hand side of the setargument, and hadn't been mentioned as desirable for the left hand side.

Con 3
It is unclear if the multiple parameters mentioned refers to the widths, or the multiple variables. $SELECT has parameters similar to widths, (but with separate semantics). As the purpose of this proposal is to SET multiple variables, and the SET command already sets multiple variables with the syntax of SET (L setleft)=expr , it is not clear to what this con is referring.

June 27, 1998: Boston, SC A Objection to Consideration raised due to history in published document. Vote of subcommittee supported objection.

Document X11/SC13/TG3/97-2 Amended to include suggestions of taskgroup and of MDC document editor. new document published as X11/SC13/TG3/98-3.

Sept 1997: Chicago, Replacement SC B Passed: (9:6:4)

Document X11/SC13/TG3/97-1 Amended to include definition of setrecord ruled as chair as substantive, thus became Replacement SC B vote. new document published as X11/SC13/TG3/97-2.

PRO

1. Needed Functionality
2. Expresses subcommittee Will
3. More efficient as intrinsic function

CON

1. Unnecessary functionality
2. Should be a library function
3. Technology trend is away from direct access to data

Task Group 3 has addressed the CON list and offers the following rebuttals:

Con 1
How necessary functionality is depends on the need of an organization. This proposal will increase readability by making code more concise and precise.

Con 2
Library functions are not as efficient as intrinsic functions. This proposal is motivated by timing and analysis of existing code, and replacement by a library function would not yield the desired efficiency.

Con 3
This is a matter of taste. Some organizations program with direct access to data to abstract appropriately. Generated code also increases performance by direct access.

March 1997: San Diego, X11/SC13/TG3/96-2 Replacement SC B Passed: (13:0:9)

PRO	CON
1. Useful Functionality 2. Has been partially implemented 3. improves programming clarity 4. could improved performance 5. currently in use <>	☐ None Offered

Sept 1996: Toronto, Replacement SC :Remanded to Task group

Document X11/SC13/TG3/92-2 brought forward to supercede X11/SC13/TG3/95-4.

PRO	CON
1. Desired Functionality	1. Metalanguage can be clearer 2. Reuses $RE 3. "as needed" is

Oct 1995: New Orleans, Replacement SC B Passed: (11:10:5)

Document X11/95-67 Superceded by X11/SC13/TG13/95-4 as replacement type B.

PRO	CON
1. Needed Functionality 2. Useful Functionality (but not needed)	1. Introduces missing parameter 2. Complex ideosyncratic syntax and semantics 3. Incomplete 4. Cost of implementation is unclear 5. Not very elegant

June 1995: Chicago, Replacement SC B

Document X11/95-67 brought forward and remanded to task group.

Jan 1995: Albuquerque, Set Positional X11/SC13/92-19: Demoted to Type C (15:0:1)

Document X11/SC13/92-19 demoted to type C because other method was more

desirable.

December 1988:

PRO	CON
1. Better program maintenance	1. Not a conventional assignment
2. More Efficient	2. Promotes Packed globals
3. Cleaner code	3. Meta generators are possible
4. Improved network performance	4. Capability exists
5. User Demand	5. Need to consider other "vector" ops
6. Discourages use of indirection	6. Would be seldom used
7. Reduces size of program	7. Context Dependent
	8. Should use braces

Task Group 3 has addressed the CON list and offers the following rebuttals:

Con 1
One strength of this proposal is the ability to perform more than one assignment, a kind of record management. The consensus is that this is a desired feature as noted when it was passed as a Type C.

Con 2
The consensus of the task group was that many members of the M[UMPS] community use packed globals and would benefit from this proposal.

Con 3
Meta generators could incorporate the ideas in this proposal and benefit from any gained efficiencies.

Con 4
The task group agreed that this proposal augments existing capabilities

Con 5
Other vector operations have been discussed by the task group. The task group decided to promote this proposal as the first one. The document editor notes that syntax such as SET A()=<B,C,D> and SET $P(A,"^",1,3)=<B,C,D>*2 are possible

Con 6
The consensus of the task group was that this proposal would often be used by programmers.

Con 7
Metalanguage and formalization address this con.

Con 8
The change from braces was made as a result of a vote in the full subcommittee.

8. Glossary

Packed globals
Using a single string to store multiple values by using a delimiter to separate the individual values.

9. Appendix

None.