THE UNIVERSITY OF MICHIGAN

Memorandum

MAD/I: PRELIMINARY DRAFT

B. Galler

CONCOMP: Research in Conversational Use of Computers
ORA Project 07449
F. H. Westervelt, Director

supported by:

DEPARTMENT OF DEFENSE
ADVANCED RESEARCH PROJECTS AGENCY
WASHINGTON, D.C.

CONTRACT NO. DA-49-083 OSA-3050
ARPA ORDER NO. 716

administered through:

OFFICE OF RESEARCH ADMINISTRATION ANN ARBOR

November 1966

MAD/I

PRELIMINARY DRAFT*

1.0 CHARACTER SET

At present, there are 61 characters in the MAD/I language: the 10 Arabic numerals 0, 1, 2, . . . 9; the 26 letters of the alphabet A, B, C, . . . Z (the letters and numerals together are called alphanumeric characters); and 25 special characters:

Name	Graphic
1. Blank (Space)	
2. Plus	+
3. Minus	−
4. Asterisk	*
5. Slash	/
6. Left Parenthesis	(
7. Right Parenthesis	)
8. Comma	,
9. Period (Decimal Point)	.
10. Quotation Mark	"
11. Semicolon	;
12. Colon	:
13. Equals	=
14. Greater Than	>
15. Less Than	<
16. Not Symbol	¬

*This is a tentative draft and is subject to change without notice.

Name		Graphic
17.	Vertical line ("or" symbol)	|
18.	Ampersand ("and" symbol)	&
19.	Percent	%
20.	Number Sign	#
21.	Commercial At-sign	@
22.	Dollar Sign	$
23.	Question Mark	?
24.	Break Character (Underline)	_
25.	Apostrophe (prime, single quote)	'

MAD/I will also accept lower-case letters (a, b, c, . . . z), but these are not necessarily available to all users. It treats these letters exactly like upper-case letters, but preserves them in lower-case form. We expect to add other characters to this list.

2.0 IDENTIFIERS AND CONSTANTS

2.1 IDENTIFIERS

An identifier is a character or string of characters from the set listed in Section 1.0.

2.1.1 ALPHANUMERIC IDENTIFIER

An alphanumeric identifier is formed from an alphabetic character which may be followed by a string of alphanumeric characters.
EXAMPLES: A MAD7090 VARIABLE B3 UPPERAND LOWERcase ARRAYELEMENT
NOTE: Readers familiar with MAD (i.e., MAD for the IBM 7090) will recognize these identifiers as the names of variables, but in MAD/I these identifiers may have other uses as well, as in the substitution

statement A=-A +B3.

Total length of an alphanumeric identifier may not exceed 256 characters.

2.1.2 PRIME IDENTIFIER

A prime identifier is formed by enclosing any character string (excluding semicolons, parentheses, and primes) within primes.

EXAMPLES: 'IF' 'GO TO' 'GOTO' '7B3' 'A*B'

'A + B' '@#7$*' 'FUNCTION' 'ARRAYELEMENT'

NOTE: MAD statements such as, for example, WHENEVER, TRANSFER TO, and FUNCTION RETURN will be written in MAD/I as 'IF', 'GO TO', and 'RETURN' respectively.

Blanks within prime identifiers will be automatically removed, i.e., 'GO TO' and 'GOTO' are equivalent. The total length, including primes but not blanks, may not exceed 256 characters.

2.1.3 PERIOD IDENTIFIERS

A period identifier is formed by enclosing between periods either a single alphabetic character or else two alphabetic characters followed by an arbitrary string of alphanumeric characters.

EXAMPLES: .E. .MOD5. .MAD7090OPERATOR. .plus. .EE.

NOTE: Because MAD allows only letters between periods, all MAD operators go directly over into MAD/I without changing their appearance.

Total length including periods must not exceed 256 characters.

2.1.4 SPECIAL IDENTIFIERS

The special identifiers are the 25 special characters and some combinations of them. Examples are:

equal to (relational operators)	=
not equal to	¬=
greater than or equal to	>= (or =>)
less than or equal to	<= (or =<)
ellipsis	...
substitution	==

2.1.5 ATTRIBUTE IDENTIFIERS

An attribute identifier is formed by placing a commercial "at" sign (@) before a string of alphanumeric characters.

EXAMPLES: @LENGTH @HEX @HOHO @EXISTENCE

NOTE: Total length of an attribute identifier including @ may not exceed 256 characters.

2.1.6 ATTRIBUTE NOTATION

The following notation is used to indicate the attributes of a constant or variable. All predefined attribute names are attribute identifiers and begin with @.

The attribute specifications, separated by commas, are enclosed in parentheses and follow the occurrence of a constant or variable. The attribute applies to all subsequent occurrences of the variable, but only to the initial occurrence of the constant. If an attribute has a single value, it is written with a substitution operator, ==, separating it from the attribute name, e.g., @LENGTH==6. If an attribute has more than one value, these values are enclosed in parentheses following the substitution operator, e.g., @ALIGN==(2,4). Although these may look like executable substitutions, they are effective only during translation (or "compile time").

To declare an attribute for several variables simultaneously, the @ sign is changed to a prime identifier, and this is followed by the list of variables. A value for the attribute, or a set of values enclosed in parentheses, may be inserted after the prime identifier. Thus, one can write:

```
'Boolean' P,Q,R
'LENGTH' 6,V,W,Y,X
```

An empty list of variables establishes the attribute as the default case for all variables.

The following are legal, predefined attributes:

Name	Abbreviations (if any)	Meaning
@PACKED	@P	packed decimal mode
@BOOLEAN	@B	Boolean mode
@INTEGER	@I	integer mode
@FLOATING	@F	floating-point mode
@CHARACTER	@C	character mode
@ENTRYNAMEVAR	@ENV	entry name variable
@HEX	@X	hexadecimal constant
@RIGHTJUSTIFIED	@RJ	causes character constant to be right justified
@NULLS	@N	causes nulls (zeros) to be inserted for the preceding or trailing fill characters of a character constant
@FILL==x	@FL==x	x is a one-byte character constant (e.g., "*")which will be inserted for the preceding or trailing fill characters of a character constant
@LENGTH==x	@L==x	specifies the length in bytes (i.e., x bytes long)

Name	Abbreviations (if any)	Meaning
@ALIGN==(x,y)	@A==(x,y)	produces alignment on the xth byte after a yth byte boundary, i.e., as though CNOP x,y preceded in assembly code)
@MODE==x	@MD==x	assigns mode x
@BYTE	@BT	same as @LENGTH==1
@HALFWORD	@H	same as @LENGTH==2, @ALIGN==(0,2)
@WORD	@W	same as @LENGTH==4, @ALIGN==(0,4)
@DOUBLEWORD	@D	same as @LENGTH==8, @ALIGN==(0,8)
@EXTERNAL	@EXT	variable is external
@HERE	@HR	for variables declared external, this attribute designates that they are defined in this program

2.2 CONSTANTS

There is no maximum length restriction on constants. Constants may be modified by attributes. (See Section 1.1.6 for description of attributes.)

2.2.1 INTEGER CONSTANTS

An integer constant is a string of decimal digits containing no decimal point, comma, or scale factor. It may be modified by @LENGTH or @ALIGN.

EXAMPLES: 12345 99 2000(@LENGTH==5)

NOTE: Integer constants are stored in binary form. A negative integer constant is stored in 2's complement form. The length of an integer constant will be assumed to be a full word, i.e., four bytes, unless otherwise modified. The value of the length attribute is always given as a number of bytes.

2.2.2 PACKED DECIMAL CONSTANTS

A packed decimal constant is a string of decimal digits containing no decimal point or scale factor and modified with @PACKED.

EXAMPLES: 25(@PACKED, @LENGTH==8) 1984 (@ALIGN==(3,8), @PACKED) 43(@PACKED) -267(@P)

NOTE: A packed decimal constant is stored in the packed decimal format. If no length is given, it will be assigned only as much storage as needed. It will be aligned on any available byte boundary unless otherwise specified.

2.2.3 FLOATING-POINT CONSTANTS

A floating-point constant is a string of decimal digits containing a decimal point or an exponent or both. The exponent is written as the letter E followed by a decimal integer, possibly signed, indicating the power-of-ten scale factor.

EXAMPLES: 3.141592 .217E-2 -5000E4 5000E+4

NOTE: Unless otherwise specified, the constant will be given the same length and alignment as the normal mode for floating-point constants and variables ("long" unless specified otherwise).

2.2.4 CHARACTER CONSTANTS

A character constant is represented by enclosing the character string in quotation marks. To represent a quotation mark within the constant, two adjacent quotation marks must be used. Character constants may be modified with @RIGHTJUSTIFIED, @ALIGN, @FILL, or @NULLS.

EXAMPLES: "ABCDEFG" (@LENGTH==14) "doN'T""ME"(@ALIGN==(1,2)).

NOTE: The default justification is to the left, with blanks (as fill characters) added to the right as necessary. If no length is specified,

the length is assumed to be the length of the characger constant. If a length is specified and is too short, characters will be truncated from the right-hand end of the character constant, regardless of justification.

2.2.5 HEXADECIMAL CONSTANTS

A hexadecimal constant is a string of hexadecimal digits (0,1,2,3,4,5,6,7,8,9, A,B,C,D,E,F) enclosed by quotation marks. Hexadecimal constants must be modified with @HEX and may be modified with @LENGTH and @ALIGN.

EXAMPLES: "ACE" (@HEX) "10C"(@HEX, @LENGTH==3) "abadabadaba"(@X)

NOTE: If no length is specified, the smallest amount of storage that will accommodate the constant will be chosen. Alignment is to an arbitrary byte boundary unless otherwise specified.

2.2.6 BOOLEAN CONSTANTS

A Boolean constant is a 1 (for true) or a 0 (for false) modified with @BOOLEAN.

EXAMPLES: 0(@BOOLEAN) 1(@BOOLEAN)

NOTE: @BOOLEAN can be abbreviated as @B. The above examples could then be shortened to 0(@B) and 1(@B), respectively. Optionally, the prime identifiers 'F' (for 0(@BOOLEAN)) and 'T' (for 1(@BOOLEAN)) may be used.

2.2.7 ENTRY NAMES

Entry names mark points of entry, i.e., beginnings of statements to which sequencing control may be transferred. Some entry names may be singled out as available for external reference, also, by being

listed in a function declaration (with optional parameter lists).

EXAMPLES: GCD OKAY here HERE

NOTE: Entry names are identifiers and must be followed by a colon when they appear as the entry name of a statement. A colon is implied after the entry name in the fixed-field card format (see below).

2.2.8 ENTRY NAME VARIABLES

A variable having the attribute @ENTRYNAMEVAR may assume values that indicate entry names. Thus the statements

```
NAME(@ENV)==L
'GO TO' NAME
    :
    ,
L:  X==Y
```

are compatible, and the 'GO TO' statement causes the statement named L to be executed next.

2.2.9 FUNCTION EVALUATION

An entry name or entry name variable followed by a period implies that the statement named is to be invoked at that time. If a list of parameters in parentheses follows, they are used for initialization at the time of transfer to the statement named. A transfer of control to an entry name may be invoked in three ways, each of which may involve parameter initialization.

(1) An entry name (with period and with an optional parameter list) may be imbedded in an expression.

EXAMPLE: X==Y+GCD.(M,N)

(2) An entry name may be "called," establishing a new return point and "save area."

EXAMPLE: 'CALL' SORT.(A,N)

Note: The period is required; the parameter list and the identifier 'CALL' are optional.

(3) An entry name may be "transferred to," without changing the current "return point" or "save area."

EXAMPLE: 'GO TO' L

'GO TO' L.(M,Q)

Note: The identifier 'GO TO' is required, and the period is required if the optional parameter list is present, otherwise the period is optional.

2.2.10 RETURN

Successive "calls" on function entry names, as in (1) and (2) in Section 2.2.9, establish a last-in-first-out chain of "return points." Execution of a 'RETURN' statement transfers control to the most recent return point and removes it from the chain. The corresponding "save area" is also reinstated. If S is a parameter with @ENV attribute, then the statement

'RETURN TO' S

transfers control to the entry name associated with S by the most recent parameter initialization. The "return" chain is restored to the state it was in at the time the entry name was defined, even if this implies removing several return points from the chain. The corresponding "save area" is also reinstated.

3.0 EXPRESSION AND STATEMENT STRUCTURE

3.1 ARITHMETIC EXPRESSIONS

Constants, variables, and function references are arithmetic expressions. Arithmetic expressions are combined by means of the special identifiers, */+-, and period identifiers, .P., .ABS., etc. Parentheses are used, as in ordinary mathematics, to indicate the order in which computations are made.

If E and F are any arithmetic expressions, and I and J are integer expressions, then the following are also arithmetic expressions: +E, -E, .ABS.E, E +F, E-F, E*F, E/F, (E), .N.I, I.A.J, I.V.J, I.EV.J, I.RS.J, and I.LS.J.

3.1.1 ARITHMETIC OPERATORS

Identifier	Function	Precedence
==	substitution (FACTOR)	A
+	addition	H
-	subtraction	H
*	multiplication	I
/	division	I
-	arithmetic negation (UNARY)	J
**(or .P.)	exponentiation (FACTOR)	K
.V.	bitwise inclusive "or"	L
.EV.	bitwise exclusive "or"	L
.A.	bitwise "and"	M
.N.	bitwise negation	N
.LS.	left shift	N
.RS.	right shift	N
.ABS.	absolute value (UNARY)	N
+	arithmetic plus (UNARY)	P

NOTE: Precedence determines the order in which operations are performed unless otherwise grouped by parentheses. The operations with higher relative precedence are performed before those with lower precedence. If operators have the same precedence, usually the one appearing left-most in the expression will be done before an operator with the same precedence to its right. The exceptions to this rule are the so-called "factor" operators which are carried on from right to left. Relative precedence of the predefined operators is A < B < C < D < E < F < G < H < I < J < K < L < M < N < P (see Section 3.2.1 for precedence letters not yet encountered).

3.2 BOOLEAN EXPRESSIONS

Boolean constants, Boolean variables, and Boolean-valued functions are Boolean expressions.

If H and F are arithmetic expressions, then H=F, H¬=F, H<F, H<=F (or H=<F), H>F, and H>=F (or H=>F) are Boolean expressions.

If M and P are Boolean expressions, then M.EQV.P, M.THEN.P, M|P, M.EXOR.P, M&P, ¬M, and (M) are Boolean expressions.

3.2.1 BOOLEAN OPERATORS

Identifier	Function	Precedence
.EQV.	equivalence	B
.THEN.	implication	C
|	inclusive "or")	D
.EXOR.	exclusive "or")	D
&	"and"	E

Identifier	Function	Precedence
¬	negation	F
=	compare equal	G
¬ =	compare unequal	G
>	compare greater than	G
>= (or =>)	compare greater than or equal	G
<	compare less than	G
<= (or =<)	compare less than or equal	G

4.0 STATEMENT FORMAT

4.1 FREE-FORM CONVENTIONS

NOTE: Free-form will be used on remote consoles. It may also be used on cards when the model 029 keypunch character set is punched.

Line numbers will be used (for line updating) on remote consoles.

4.1.1 COLON AND SEMICOLON CONVENTIONS

An entry name is followed by a colon. Semicolons are used as punctuation between statements.

NOTE: A statement continues until explicitly ended by a semicolon. Hence, a statement may be as long or as short as one would like and may continue for any number of lines. Conversely, more than one statement can be written per line. In fixed-field form, colons and semicolons may be omitted if desired.

4.2 FIXED-FIELD CONVENTIONS FOR PUNCHED CARDS

Columns 2 through 71 of the card are available for writing statements. If any character, other than a semicolon or blank, is

punched in Column 72, the statement is continued on the next card. Column 72 is not used, and Column 2 of the next card follows immediately after Column 71.

Entry names must start in Column 1 and are ended by the first blank.

NOTE: Punching a semicolon in Column 72 or leaving it blank has the same effect, i.e., a blank in Column 72 is an implied semicolon. The continuation punch in Column 72 has only the function of indicating continuation and is not otherwise used.

The first blank after an entry name is an implied colon. There must be at least one blank (or a colon) between an entry name and the following statement.

With a few substitutions, the fixed-field conventions may be used in punching programs on a model 026 keypunch. The substitutions are:

029	026
"	$
>	.G.
<	.L.
>=	.GE.
<=	.LE.
¬=	.NE.
|	.OR.
&	.AND.
@_____	'AT_____'

5.0 COMPARISON OF MAD AND MAD/I STATEMENTS

Declaration Statements

MAD	MAD/I
DIMENSION List	'DIMENSION' List
EQUIVALENCE List	'EQUIVALENCE' List
ERASABLE List PROGRAM COMMON List	'EXTERNAL' List
NORMAL MODE IS . . .	(See NOTE below)
INTEGER List	'INTEGER' List
FLOATING POINT List	'FLOATING LONG' List 'FLOATING SHORT' List
BOOLEAN List	'BOOLEAN' List
STATEMENT LABEL List FUNCTION NAME List	'ENTRYNAMEVAR' List
FORMAT VARIABLE List	(no equivalent)

NOTE: Remark (R in Column 11). List items are separated by commas.

If the first non-blank character in a statement is an asterisk (*), the following text, up to but not including the next semicolon, is treated as a remark.

Computational Statements

MAD	MAD/I
V=E	V==E
TRANSFER TO S	'GO TO' S ;
WHENEVER B, STATEMENT	'IF' B, STATEMENT;

MAD	MAD/I
Computational Statements	
WHENEVER B1	'IF' B1;
.	.
.	.
.	.
OR WHENEVER B2	'OR IF' B2;
.	.
.	.
.	.
OTHERWISE	'OR ELSE';
.	.
.	.
.	.
END OF CONDITIONAL	'ENDIF';
THROUGH S, FOR V = E1,E2,B	'FOR' V==E1,E2,B,STATEMENT
S STATEMENT	
THROUGH S, FOR V = E1,E2,B	'FOR' V==E1,E2,B;
.	.
.	.
.	.
S	'ENDFOR';
THROUGH S, FOR VALUES OF V=E1,E2,.....	'FOR' V==(E1,E2,...), STATEMENT
S STATEMENT	
THROUGH S, FOR VALUES OF V=E1,E2,...	'FOR' V==(E1,E2,...);
.	.
.	.
.	.
S	'ENDFOR';
CONTINUE	(Empty statement);
PAUSE NO.	'SUSPEND' (operator);
END OF PROGRAM	'END'

NOTE: Overlapping of 'FOR's and 'IF's is not permitted. More generally, any construction that can be included in the scope of another construction must be completely included.

Function-related Statements

EXTERNAL)) FUNCTION INTERNAL)	'FUNCTION'
ENTRY TO F	F: (as entry name)
END OF FUNCTION	'END' (see NOTE below) or 'END' entry name
FUNCTION RETURN	'RETURN' or 'RETURN TO' entry
ERROR RETURN	'ERROR RETURN'

NOTE: The end of a function may be noted

'END' Entry name

where Entry name is an entry name of the function. If the function is ended in this way, even when normally more than one compound statement ends at this point, only the one 'END' Entry name is required (see Section 6.0 for an example).

Entry names (and their parameter lists) intended to be referenced externally are listed in the 'FUNCTION' statement, as in the following example.

```
'FUNCTION' (F,G(X,Y)), H(W,X),K
```

where F and G are entry names with parameters X and Y, H is an entry name with parameters W and X, and K is an entry name with no parameters.

If no entry names (and no parameters) occur, the program is a "main" program, to be executed first, starting with the first executable statement. If a list of entry names is present, this effect of being a "main" program can also be achieved by including the prime identifier 'MAIN' in the list, without a parameter list.

A "one-line definition" is written using only the 'FUNCTION' statement:

```
'FUNCTION' REM(A,B) == A - (A/B)*B;
```

General I/O

Unformatted:

READ DATA	'READ'
READ AND PRINT DATA	'READ AND PRINT'
READ COMMENT $\_\_\_\_\_$	'PRINT CHARACTERS' "____"
PRINT BCD RESULTS List	'PRINT CHARACTERS' List
PRINT RESULTS List	'PRINT' List
PRINT OCTAL RESULTS List	'PRINT HEX' List

Formatted:

PRINT FORMAT Format, List	'PRINT FORMAT' Format, List
PRINT ON LINE FORMAT Format, List	(no equivalent)
PUNCH FORMAT Format, List	'PUNCH FORMAT' Format, List
READ FORMAT Format, List	'READ FORMAT' Format, List
LOOK AT FORMAT Format, List	'LOOK AT' Format, List

Note: The occurrence of 'c' α in a PRINT CHARACTERS list causes α to be interpreted as carriage control at that point. Double spacing will be used by default.

Tape Statements

WRITE BINARY TAPE)
)
WRITE BCD TAPE)
)
READ BINARY TAPE) (These will be implemented as
)
READ BCD TAPE) subroutines.)
)
END OF FILE TAPE)
)
BACKSPACE FILE)
)
BACKSPACE RECORD)

Miscellaneous Statements

Recursion:

SET LIST TO V	'SET LIST' V
SAVE DATA List	'SAVE' List
RESTORE DATA List	'RESTORE' List
SAVE RETURN))	These are accomplished by putting 'EXIT' as one item on a 'SAVE'
RESTORE RETURN)	or 'RESTORE' list.

Presetting:

VECTOR VALUES V = List	'PRESET' V==List
VECTOR VALUES V(I),...V(J) = Constant	'PRESET' V(I),...V(J)= Constant
PARAMETER List	'PARAMETER' List
SYMBOL TABLE VECTOR V	'SYMBOL TABLE' V
FULL SYMBOL TABLE VECTOR V	'SYMBOL TABLE' V

Output Control:

LISTING ON	(New forms will be developed.)
LISTING OFF	
REFERENCES ON	
REFERENCES OFF	

Operator Definition

The DEFINE OPERATOR and MODE STRUCTURE statements will have appropriate counterparts. The definition scheme has not been settled yet, however. In addition, there will be a facility for defining certain types of new statements.

5.1 NESTED STATEMENTS

Several statement types will have values assigned to them. For example, the value of an assignment statement V == E is the value of E.

Any such statement can be enclosed in parentheses and used wherever an expression of the same mode as its value is legitimately used.

6.0 EXAMPLE MAD/I PROGRAM July 19, 1966

```
*This program is written as it might appear in fixed-field    1
card format.  Labels start in Column 1 and are ended by the   2
first blank.  Columns 2 through 71 are available for state-   3
ments.  A blank in Column 72 is an implied semicolon.

'FUNCTION' GCD
'INTEGER'
GCD 'RESTORE' Z,Y
'IF' Y>Z
     'SAVE' 'EXIT', Z,Y
       X==GCD.(0)
     'RESTORE'  'EXIT'
     'RETURN' X
'OR IF' REM.(Z,Y)==0
     'RETURN' Y
'ENDC'
'SAVE'  'EXIT', REM.(Z,Y),Y
X==GCD.(0)
'RESTORE'  'EXIT'
'RETURN' X
'FUNCTION' REM(A,B)==A-(A/B)*B
'END' GCD
```

```
MO5       21:46      July 19, 1966

100       * Same Program [On Remote Console] ;
110   'FUNCTION' GCD;
120   'INTEGER'; * NO LIST AFTER 'INTEGER' IMPLIES THE NORMAL MODE
130   IS INTEGER
140   GCD: 'RESTORE' Z,Y:
150   'IF' Y>Z;
160         'SAVE' 'EXIT', Z,Y;
170         X==GCD.(0);
180         'RESTORE' 'EXIT';
190         'RETURN' X;
200   'OR IF'REM.(Z,Y)==0;
210         'RETURN' Y;
220   'ENDC';
230   'SAVE'  'EXIT',REM.(Z,Y),Y;
240   X==GCD.(0);
250   'RESTORE'  'EXIT';
260   'RETURN' X;
270   'FUNCTION' REM(A,B);
280   REM: 'RETURN' A-(A/B)*B;
290   *THIS ILLUSTRATES THE USE OF 'END' GCD TO END MORE THAN ONE FUNCTION
      DEFINITION;
300   'END' GCD;
```