Matrix Statements Reference Manual

WANG LABORATORIES, INC.

ONE INDUSTRIAL AVENUE, LOWELL, MASSACHUSETTS 01851, TEL. (617) 459-5000, TWX 710 343-6769, TELEX 94-7421

Disclaimer of Warranties and Limitation of Liabilities

The staff of Wang Laboratories, Inc., has taken due care in preparing this manual; however, nothing contained herein modifies or alters in any way the standard terms and conditions of the Wang purchase, lease, or license agreement by which this software package was acquired, nor increases in any way Wang's liability to the customer. In no event shall Wang Laboratories, Inc., or its subsidiaries be liable for incidental or consequential damages in connection with or arising from the use of the software package, the accompanying manual, or any related materials.

WANG LABORATORIES, INC.

ONE INDUSTRIAL AVENUE, LOWELL, MASSACHUSETTS 01851, TEL. (617) 459-5000, TWX 710 343-6769, TELEX 94-7421

HOW TO USE THIS DOCUMENT

This document is provided to give quick answers to questions concerning the operations of the Matrix Statements. It assumes the reader is familiar with the general operation of his system hardware, and the hardware is sufficient to support the Matrix statement set. Additionally, the reader is expected to have a basic knowledge of matrix operations and definitions (linear matrix algebra), and BASIC programming.

TABLE OF CONTENTS

Section I

General Information

INTRODUCTION

The Matrix statements contained in this manual are standard instructions in the BASIC language set available on all 2200T CPU's (including PCS, WCS; and Work Station type processors). These statements can also be installed as an option on most 2200 series CPU's not already containing them. The matrix statement set for the 2200VP, 2200MVP and 2200VS are described in separate manuals.

TABLE I. MATRIX OPERATIONS

OPERATION	DESCRIPTION	EXAMPLE
MAT addition†	array = array + array	MAT A = B+C
MAT CON*	each element of array = 1	MAT A = CON
MAT equality†	array = array	MAT A = B
MAT IDN*	matrix = identity matrix	MAT A = IDN
MAT INPUT*,**	receive array elements from keyboard	MAT INPUT A,B$
MAT INV,d†	matrix = inverse of matrix, d = determinant of matrix	MAT A = INV(B),D
MAT multiplication†	array = array x array	MAT A = B*C
MAT PRINT**	print elements of array	MAT PRINT A,B$
MAT READ*,**	array = DATA values	MAT READ A,B$
MAT REDIM*,**	redimension array	MAT REDIM A(X,Y)
MAT scalar multiplication†	array = constant x array	MAT A = (3)*B
MAT subtraction†	array = array – array	MAT A = B–C
MAT TRN†	array = transpose of array	MAT A = TRN(B)
MAT ZER*	each element of array = 0	MAT A = ZER

*Resultant array redimensioned explicitly. (i.e. by specifying new dimensions in the statement)
†Resultant array redimensioned implicitly. (i.e. depends upon dimension of arguments)
**Can be performed on alphanumeric arrays.

Operations are performed on numeric arrays according to the rules of linear algebra and can be used for the solution of systems of non-singular homogenous linear equations. Inversion of matrices can be done in significantly shorter time than is possible with BASIC programs. MAT operations on alphanumeric arrays can be used for simple and rapid I/O (input/output) and printing of alphanumeric material. Error messages for Matrix Statements are described in Appendix A.

INSTALLATION

The Matrix statements are either factory installed, or retrofit by a Wang Service Representative in the user's System.

ARRAY DIMENSIONING

Both numeric and alphanumeric arrays can be manipulated with the Matrix statements. The rules of the BASIC Language require that an array be dimensioned with a DIM or COM statement in order to reserve space for the array variables, prior to the variable's use in a program statement.

The COM statement defines arrays which can be used in common by several programs or program segments. Common variables are stored in an area of memory which is not cleared as subsequent programs are run. All non-common variables, however, are cleared from memory. A COM statement must not change the dimensions of a previously defined common variable.

Space may be reserved for more than one array with a single dimension statement by separating the entries for array names with commas. The space to be reserved must be explicitly indicated – expressions are not allowed. The maximum allowable dimension of arrays are $1 \leqslant (d_1, d_2) \leqslant 255$. Note that the space required for the array must not exceed the machines capacity (see Appendix B). Additionally, with alphanumeric arrays, the length of each element in the array can be specified between 1 and 64 bytes, inclusive.

If an array's dimensions are not specified in DIM or COM statement, it will be automatically dimensioned as a 10 X 10 matrix. For an alphanumeric array, the maximum length of each element is defined equal to 16. The total number of elements in any array must be ⩽ 4,096.

ARRAY REDIMENSIONING

The dimensions of an array can be changed explicitly during the execution of MAT statements by giving the new dimensions, enclosed in parentheses, following the array name in any of the following MAT statements:

```
MAT  CON
MAT  IDN
MAT  INPUT
MAT  READ
MAT  REDIM
MAT  ZER
```

Example:

```
MAT  CON(5,5)
```

Arrays can also be redimensioned implicitly.

Example:

```
10 DIM A(10,10),B(2,2),C(2,2)
20 . . .
30 . . .
40 MAT A=B+C
```

The array A is redimensioned at statement 40 from a 10 x 10 array to a 2 x 2 array.

For alphanumeric arrays, the maximum length of each element can be changed by specifying the new length after the dimension specification.

Example:

```
REDIM A$(2,3) 10
```

This statement redimensions the array A$ to be two rows by three columns with the maximum length of each element in the array equal to 10.

> **NOTE:**
> *With either explicit or implicit redimensioning, the newly dimensioned array must not take up more space than was available in the array as it was originally dimensioned. For numeric arrays this implies there are the same number or less total elements. For alphanumeric arrays, there must be the same number or less total characters.*

MATRIX STATEMENT RULES

Certain rules must be followed in using Matrix Statements.

1. Each matrix statement must begin with the word MAT.
2. Each variable used in a MAT statement must be an array variable.
3. Multiple matrix operations are not permitted in a single MAT statement. (e.g. MAT A = B+C – D is illegal but the same result can be achieved by using the two MAT statements: MAT A = B+C, MAT A = A–D).
4. Arrays which contain the result of certain MAT statements are automatically redimensioned; other arrays can be redimensioned explicitly in the MAT statement (see Table I). A redimensioned numeric array cannot contain more elements than given in its previous definition; a redimensioned alphanumeric array cannot contain more characters than given in its previous definition.
5. A vector (a singly subscripted array) cannot be redimensioned as a matrix (a doubly subscripted array); nor can a matrix be redimensioned as a vector.
6. When overlaying programs (i.e., loading and executing program segments under program control) or running programs from a particular line number, common variables must be currently dimensioned as originally defined in the COM statement. If not, the defining COM statement must be deleted from the program when the program overlay is executed. (A common variable need only be listed in the COM statement of the original program; it does not have to be specified in COM statements of subsequently chained or overlayed programs.) Caution must be exercised when using DIM or COM statements in programs where redimensioning occurs.
7. The same array variable cannot appear on both sides of the equation in matrix multiplication and matrix transposition.

   MAT C = A*B and MAT A = TRN (C) are legal MAT statements;
   MAT C = C*B and MAT B = TRN (B) are not.

8. Matrix operations are valid only when the dimensions and/or types of the matrix operands are compatible.

In the Matrix statement general forms given in Section II: items in brackets ([]) are optional.
items in uppercase must occur as shown.
items in lowercase must be defined by the user.
items in braces ({}) are alternatives, one of which must be used.
symbols (+,=,*,(),etc.) must occur as shown.

Section II

Matrix Operations

MAT + (MAT addition)

General Form:	MAT c = a + b where c, a, and b are numeric array names.

Purpose:

Adds two matrices or vectors of the same dimension. The sum is stored in array c. Array c can appear on both sides of the equation. Array c is redimensioned to have the same dimensions as arrays a and b.

An error message is printed and execution terminated if the dimensions of a and b are not the same.

Example 1:

This example illustrates syntax use of the MAT addition statement only and does not constitute a working program.

```
10 DIM A(5,5),D(5,5),E(7),F(5),G(5)
20 MAT A = A + D
30 MAT E = F + G
40 MAT A = A + A
```

Example 2:

The program provided adds the corresponding elements of the 3 by 3 arrays D and E to give the new array F. Array F is automatically redimensioned as a 3 by 3 array.

```
10 DIM D(3,3),E(3,3),F(5,2)
20 PRINT "ENTER ELEMENTS OF ARRAY D"
30 MAT INPUT D
40 PRINT "ENTER ELEMENTS OF ARRAY E"
50 MAT INPUT E
60 MAT F = D + E
70 PRINT "ELEMENTS OF ARRAY F" :PRINT
80 MAT PRINT F;
```

$$\text{Let } D = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 2 & 2 & 2 \end{bmatrix}, \quad E = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \end{bmatrix}$$

When the program is executed, array F is displayed:

```
ELEMENTS OF ARRAY F
  4    4    4
  4    4    4
  5    5    5
```

MAT CON (MAT CONstant)

General Form:	MAT c = CON [(d_1 [,d_2])] where c is a numeric array name and d_1, d_2 are expressions specifying new dimensions. ($1 \leqslant d_1, d_2 < 256$; default $d_1 = d_2 = 10$.)

Purpose: This statement sets all elements of the specified array to one (1). Using (d_1,d_2) causes the matrix to be redimensioned. If (d_1,d_2) are not used, the matrix dimensions are as specified in a previous COM, DIM or MAT statement, or are the default values.

Examples of MAT CON syntax:

```
10 MAT A = CON(10)
15 MAT C = CON(5,7)
20 MAT B = CON(5*Q,S)
30 MAT A = CON
```

Examples showing usage in a program:

```
10 MAT A = CON(2,2)
20 MAT PRINT A;
```

when this program is executed the CRT displays the result in packed format:

```
1    1
1    1
```

MAT = (MAT equality)

General Form:	MAT a = b where a and b are numeric array names.

Purpose: This statement replaces each element of array a with the corresponding element of array b. Array a is redimensioned to conform to the dimensions of array b.

Examples showing statement syntax:

```
10 DIM A(3,5),B(3,5)
20 MAT A=B
30 DIM C(4,6),D(2,4)
40 MAT C=D
50 DIM E(6),F(7)
60 MAT F=E
```

Example showing use in a program:

$$\text{Let } A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \end{bmatrix}$$

MAT = (Continued)

Program:

```
10 DIM A(3,3),B(2,3)
20 MAT A = CON
30 MAT PRINT A
40 MAT INPUT B
50 MAT A = B
60 MAT PRINT A
```

when this program is executed the constant 3 by 3 array A is displayed as:

```
1      1      1
1      1      1
1      1      1
```

in zoned format; the array B is input via the keyboard; and the new array A is displayed as:

```
9      8      7
6      5      4
```

in zoned format.

MAT IDN (MAT identity)

General Form:

MAT c = IDN [(d_1 ,d_2)]
where c is a numeric array name and d_1 ,d_2 are expressions specifying new dimensions. ($1 \leqslant d_1 ,d_2 \leqslant 255$;)

Purpose:

This statement causes the specified matrix to assume the form of the identity matrix. If the specified matrix is not a square matrix, an error message is displayed and execution is terminated.

Using (d_1 ,d_2) causes the matrix to be redimensioned. If (d_1 , d_2) are not used, the matrix has the dimensions specified in a previous COM, DIM or MAT statement.

Example showing statement syntax:

```
10 MAT A = IDN (4,4)
20 MAT B = IDN
30 MAT C = IDN(X,Y)
```

Example in which the identity matrix is displayed:

```
10 DIM A(4,4)
20 MAT A = IDN
30 MAT PRINT A
```

When this program is executed, the matrix A is displayed in zoned format as:

```
1      0      0      0
0      1      0      0
0      0      1      0
0      0      0      1
```

MAT INPUT

General Form:

MAT INPUT {numeric array name [(d_1 [,d_2])] / alpha array name [(d_1 [,d_2]) [length]]} [, ...]

where: d = expression specifying a new dimension ($1 \leq d_1, d_2 \leq 255$; default : $d_1 = d_2 = 10$)

length = expression specifying maximum length of each alpha array element ($1 \leq$ length < 65) (default = 16)

Purpose:

The MAT INPUT statement allows the user to supply values from the keyboard for an array during the running of a program. When the system encounters a MAT INPUT statement, it displays a question mark (?) and waits for the user to supply values for the arrays specified in the MAT INPUT statement. The dimensions of the array(s) are as last specified in the program (by a COM, DIM or MAT statement) unless the user redimensions the array(s) by specifying the new dimension(s) after the array name(s). The maximum length for alphanumeric array elements can be specified by including the length after the dimensions specification; if no length is specified, a default value of 16 is used.

The values which are input are assigned to an array row by row until the array is filled. If more than one value is entered on a line, the values must be separated by commas. Alphanumeric data with leading spaces or commas in it can be entered by entering a quotation character (") before and after the data value. Several lines can be used to enter the required data. Excess data are ignored. If there is a system detected error in the entered data, the data must be re-entered beginning with the erroneous value. The data which preceded the error are accepted. Input data must be compatible with the array (i.e., numeric data for numeric arrays, alphanumeric literal strings for alphanumeric arrays). Entering no data on an input line (i.e., only touching the RETURN/EXECUTE key to enter a carriage return) signals the System to ignore the remaining elements of the array currently being filled.

Example 1, with numeric variables:

```
5 DIM A(2), B(3), C(3, 4)
10 MAT INPUT A,B(2),C(2,4)
```

When this program is run, key in on the keyboard the values, separated by commas,

−3, −5, .612, .41

Touch the RETURN key to enter these values for array elements A(1), A(2),B(1) and B(2). Enter the values

−6.4, −5.6, 98

separated by commas; touch the RETURN key to enter these values for the array elements C(1,1), C(1,2), and C(1,3). Touch the RETURN key without entering further values to enter a carriage return and ignore the rest of possible values for the array C.

Example 2, with alphanumeric string variables:

```
10 DIM C$(2), A$(4)4, B(3)
100 MAT INPUT A$(4)3, B(2), C$
```

Enter RAD,DEG,MIN,SEC,2.5,5.6,LAST RESULT, "ROTATE X,Y" and touch the RETURN key.

MAT INV (MAT INVerse, D)

General Form:	MAT c = INV(a) [,d] where c and a are numeric array names. d = numeric variable; the value of the determinant of the array a .

Purpose:

This statement causes matrix c to be replaced by the inverse of matrix a. Array c can appear on both sides of the equation. Matrix c is redimensioned to have the same dimensions as matrix a. Matrix a must be a square, non-singular matrix; otherwise, an error message is displayed and program execution is terminated.
After inversion, the variable d (if specified) equals the value of the determinant of matrix a.

This statement uses the Gauss-Jordan Elimination Method done in-place; as with any matrix inversion technique, results can be inaccurate if the determinant (or normalized determinant) of the matrix is close to zero. It is therefore good practice to check the determinant after any inversion. Additionally with large matrices, some round-off accumulation error is to be expected.

The Gauss-Jordan Elimination Method also works best when values on the main diagonal are in the same range as other values in the matrix; in particular, numbers with large negative exponents on the main diagonal should be avoided when other values are not in this range. When in doubt, it is a good plan to check your data before inversion and adjust or rearrange it accordingly (for example, elements that are close to zero set equal to zero or rearrange data so that elements on the main diagonal are as large as possible).

Example 1, illustration of statement syntax:

```
10 MAT A = INV(B)
400 MAT Z1=INV(P), X2
700 MAT F = INV(C), J3
800 MAT C = INV(C)
```

Example 2:

This program takes the 4x4 matrix A from the keyboard input, calculates the inverse of it, and prints both the result and the value of the determinant of A.

```
10 DIM A(4,4)
20 PRINT "ENTER ELEMENTS OF A 4x4 MATRIX"
30 MAT INPUT A
40 MAT B=INV(A),D
50 MAT PRINT B
60 REM B IS THE INVERSE OF A, D IS THE DETERMINANT OF A
70 PRINT "VALUE OF DET.A=";D
```

If array A = $\begin{bmatrix} 0 & 2 & 4 & 8 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 4 & 8 & 16 & 32 \end{bmatrix}$ then array B = $\begin{bmatrix} -1 & 0 & 0 & .25 \\ -3.5 & -2 & -4 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & -.25 \end{bmatrix}$

and the value of the determinant of A = –8

If the input matrix is singular (i.e., non-invertible) the error message ERR 93 is displayed.

MAT * (MAT multiplication)

General Form:	MAT c = a * b where c, a, and b are numeric array names

Purpose: The product of arrays a and b is stored in array c. Array c cannot appear on both sides of the equation. If the number of columns in matrix a does not equal the number of rows in matrix b, an error message is printed and execution is terminated. The resulting dimension of c is determined by the number of rows in a and number of columns in b.

Example of statement syntax:

```
10 DIM A(5,2),B(2,3),C(4,7)
20 DIM E(3,4),F(4,7),G(3,7)
40 MAT G = E * F
50 MAT C = A * B
```

If the rows and columns are not compatible, the error message ↑ERR 90 is displayed. For example:

```
10 DIM A(2,2), B(4,4)
20 MAT C = A * B
              ↑ ERR 90
```

Example of usage in a program:

```
10 DIM A(2,3),B(3,4)
20 MAT INPUT A,B
30 MAT C = A * B
40 MAT PRINT C
```

Let $A = \begin{bmatrix} 0 & 1 & 4 \\ 7 & 7 & 7 \end{bmatrix}$, $B = \begin{bmatrix} 5 & 1 & 0 & 4 \\ 4 & 1 & 0 & 4 \\ 3 & 4 & 3 & 4 \end{bmatrix}$

When the program is executed and arrays A and B are keyed in, array C is displayed as:

```
16    17    12    20
84    42    21    84
```

MAT PRINT

General Form:	MAT PRINT array name [t array name . . .] [t] where t is a comma or semicolon

Purpose: The MAT PRINT statement prints arrays in the order given in the statement. Each matrix is printed row by row. All the elements of a row are printed on as many lines as required. The first element of a row always starts at the beginning of a new print line. An array is printed in zoned format unless the array name is followed by a semicolon, in which case, the array is printed in packed format. However, for alphanumeric arrays the zone length is set equal to the maximum length defined for each array element (not always 16). A vector (a one-dimensional array) is printed as a column vector.

MAT PRINT (Continued)

Examples of statement syntax:

```
10 DIM A(4),B(2,4),B$(10),C$(6)
100 MAT PRINT A; B, C$
200 MAT PRINT A, B$
```

Examples of usage in a program:

This program takes as input nine alphanumeric quantities, each up to 16 characters long, and prints them as a 3 x 3 array in packed format.

```
10 DIM Z$ (3,3)
20 MAT INPUT Z$
30 MAT PRINT Z$;
```

MAT READ

General Form:

MAT READ { numeric array name [(d_1 [,d_2])] / alpha array name [(d_1 [,d_2]) [length]] } [,...]

where: d = expression specifying a new dimension ($1 \leqslant d_1, d_2 \leqslant 255$;)

length = expression specifying maximum length of each alpha array element ($1 \leqslant$ length < 65) (default = 16)

Purpose:

The MAT READ statement is used to assign values contained in DATA statements to array variables without referencing each member of the array individually. The MAT READ statement causes the referenced arrays to be filled sequentially with the values available from the DATA statement(s). Each array is filled row by row. Values are retrieved from a DATA statement in the order they occur on that program line. If a MAT READ statement references beyond the limit of existing numbers in a DATA statement, the system searches for the next sequential DATA statement. If no more DATA statements are in the program, an error message is printed and execution is terminated.

Alphanumeric string variable arrays can also be used in the list. The information entered in the data statement must be compatible with the array (i.e., numeric values for numeric arrays, alphanumeric literal strings for alphanumeric arrays).

The dimensions of the array(s) are as last specified in the program (by a COM, DIM, or MAT statement) unless the user redimensions the array(s) by specifying new dimension(s) after the array name(s) in the MAT READ statement. The maximum length for alphanumeric array elements can be specified by including the length after the dimension specification; if no length is specified, a default of 16 is used.

The RESTORE statement can be used with MAT READ in the same manner as with the READ statement.

MAT READ (Continued)

Example 1:

```
5 DIM A(1),B(3,3)
10 MAT READ A,B(2,3)
100 DATA 1,-.2,315,-.398,6.21,0,0
110 MAT PRINT A,B
```

Example 2:

```
10 DIM A(2,2),B$(3,2)
15 DIM C(3), D$(4) 7
20 MAT READ A,B$,C(2),D$(4) 6
100 DATA 1,2,3,-3.4E12,5
110 DATA "ABC","DEFG","HI","J","K"
120 DATA .2345,1E-12, "AB", "CD", "EFGH","IJK"
130 MAT PRINT A, B$, C, D$
```

MAT REDIM (MAT REDIMension)

General Form:	MAT REDIM array name (d_1 [,d_2]) [length] [,...] where: d = expression specifying new dimension ($1 \leqslant d_1, d_2 \leqslant 255;$) length = expression specifying the maximum length of each alpha array element (1 ⩽ integer < 65) (default = 16)

Purpose

The MAT REDIM statement redimensions the specified arrays. The new dimension(s) are enclosed in parentheses immediately following the array name. The maximum length of each element in an alphanumeric array can be specified by including the length as the last parameter of the dimension specification. If a maximum length is not specified, it is set at 16. A vector cannot be redimensioned as a matrix, and a matrix cannot be redimensioned as a vector. Redimensioned arrays cannot be larger than the originally defined array (if this occurs, error 92 is displayed).

Examples of statement syntax:

```
10 MAT REDIM A(4,5)
20 MAT REDIM B$(20)8,C1$(4,4),D$(8)
```

Example of a working program:

```
10 DIM A(3,3)
20 MAT INPUT A
30 MAT PRINT A;
40 MAT REDIM A(2,2)
50 MAT PRINT A;
```

When this program is executed, enter the array A as

9 8 7 6 5 4 3 2 1

The first MAT PRINT statement displays the above data as the matrix

```
9 8 7
6 5 4
3 2 1
```

and the second MAT PRINT statement displays it as

```
9 8
7 6
```

both MAT PRINT statements use packed format.

MAT ( ) * (MAT scalar multiplication)

General Form:	MAT c = (k) * a where c and a are numeric array names and k is an expression.

Purpose: Each element of matrix or vector a is multiplied by the value of expression k and the product is stored in array c. Array c can appear on both sides of the equation. Array c is redimensioned to the same dimensions as array a.

Example of statement usage:

```
20 MAT C = (SIN(X))*A
30 MAT D = (X+Y↑2)*A
40 MAT A = (5)*A
```

Example of program:

This program inputs a 3x3 array and a scalar. It then performs scalar multiplication and displays the result.

```
10 PRINT "ENTER DATA FOR A 3 x 3 ARRAY"
20 MAT INPUT C(3,3)
30 PRINT "ENTER SCALAR"
40 INPUT K
50 MAT A = (K) * C
60 MAT PRINT A;
```

Let $C = \begin{bmatrix} 5 & 3 & 1 \\ 2 & 2 & 2 \\ 1 & 1 & 1 \end{bmatrix}$, K = 5 then $A = \begin{bmatrix} 25 & 15 & 5 \\ 10 & 10 & 10 \\ 5 & 5 & 5 \end{bmatrix}$

MAT - (MAT subtraction)

General Form:	MAT c = a – b where a, b, and c are numeric array names.

Purpose: Subtracts matrices or vectors of the same dimension. The difference of each element is stored in the corresponding element of c. Array c can appear on both sides of the equation. An error message is displayed and execution is terminated if the dimensions of a and b are not the same. Array c is redimensioned to have the same dimensions as arrays a and b.

Example of statement syntax:

```
10 DIM A(6,3),B(6,3),C(6,3),D(4),E(4)
20 MAT C = A – B
30 MAT C = A – C
40 MAT D = D – E
```

Example of program:

```
10 DIM D(3,3),E(3,3)
20 MAT INPUT D
30 MAT INPUT E
40 MAT F = D – E
50 MAT PRINT F
```

If you let $D = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 2 & 2 & 2 \end{bmatrix}$, $E = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \end{bmatrix}$ Then $F = \begin{bmatrix} -2 & -2 & -2 \\ -2 & -2 & -2 \\ -1 & -1 & -1 \end{bmatrix}$

MAT TRN (transpose)

General Form:	MAT c = TRN(a) where a and c are numeric array names.

Purpose: This statement causes array c to be replaced by the transpose of array a. Array c is redimensioned to the same dimensions as the transpose of array a. Array c cannot appear on both sides of the equation.

Example of statement syntax:

```
10 MAT C = TRN(A)
```

Example of program usage:

```
10 DIM A(3,3)
20 MAT INPUT A
30 MAT C = TRN(A)
40 MAT PRINT C
```

$$\text{Let } A = \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$$

When the program is executed, C is displayed as:

$$\begin{bmatrix} 9 & 6 & 3 \\ 8 & 5 & 2 \\ 7 & 4 & 1 \end{bmatrix}$$

NOTE:

With any 32K System 2200, an array to be transposed must not be the first variable or array defined in the program; it must be preceded in the program by a variable or array defined with at least 8 (column-dimension-of-array-to-be-transposed −1) bytes. In the above example, on a 32K System 2200, line 10 should read: 10 DIM A$16,A(3,3).*

MAT ZER (MAT ZERo)

General Form:	MAT c = ZER [(d_1 [,d_2])] where c is a numeric array name and d_1, d_2 are expressions specifying new dimensions. ($1 \leqslant d_1, d_2 \leqslant 255$; default: $d_1 = d_2 = 10$)

Purpose: This statement sets all elements of the specified array equal to zero. Using (d_1,d_2) causes the matrix to be redimensioned. If (d_1, d_2) are not used, the matrix has the dimensions specified in a previous COM, DIM, or MAT statement.

Example:

```
10 MAT C = ZER(5,2)
20 MAT B = ZER
30 MAT A = ZER(F,T+2)
40 MAT D = ZER(20)
```

Appendix A Matrix Statement Error Messages

↑ERR 89 MATRIX NOT SQUARE

Cause: The dimensions of the operand in a MAT inversion or identity are not equal.

Action: Correct the array dimensions.

Example:

```
:10 MAT A=IDN(3,4)
:RUN

 10 MAT A=IDN(3,4)
                    ↑ERR 89
:10 MAT A=IDN(3,3)               (Possible Correction)
```

↑ERR 90 MATRIX OPERANDS NOT COMPATIBLE

Cause: The dimensions of the operands in a MAT statement are not compatible; the operation cannot be performed.

Action: Correct the dimensions of the arrays.

Example:

```
:10 MAT A = CON(2,6)
:20 MAT B=IDN(2,2)
:30 MAT C=A+B
:RUN

 30 MAT C=A+B
              ↑ERR 90
:10 MAT A=CON(2,2)               (Possible Correction)
```

↑ERR 91 ILLEGAL MATRIX OPERAND

Cause: The same array name appears on both sides of the equation in a MAT multiplication or transposition statement.

Action: Correct the statement.

Example:

```
:10 MAT A=A*B
             ↑ERR 91
:10 MAT C=A*B                    (Possible Correction)
```

↑ERR 92 ILLEGAL REDIMENSIONING OF ARRAY

Cause: The space required to redimension the array is greater than the space initially reserved for the array; or a matrix is being redimensioned into a vector; or a vector is being redimensioned into a matrix.

Action: Reserve more space for array in DIM or COM statement.

Example:

```
:10 DIM A(3,4)
:20 MAT A=CON(5,6)
:RUN

 20 MAT A=CON(5,6)
                  ↑ERR 92
:10 DIM A(5,6)                      (Possible Correction)
```

↑ERR 93 SINGULAR MATRIX

Cause: The operand in a MAT inversion statement is singular and cannot be inverted.

Action: Correct the input values or the program.

Example:

```
:10 MAT A=ZER(3,3)
:20 MAT B=INV(A)
:RUN

 20 MAT B=INV(A)
                ↑ERR 93
```

↑ERR 94 MISSING ASTERISK

Cause: An asterisk (*) was expected.

Action: Correct statement text.

Example:

```
:10 MAT C=(3)B
             ↑ERR 94
:10 MAT C=(3)*B                     (Possible Correction)
```

Appendix B Specifications

Speed: For most matrix operations a Matrix Statement runs about 8 to 10 times faster than equivalent BASIC programs without the MAT statement.

MAT statement	Approximate Speed for a 10 x 10 matrix (in seconds)
MAT A = B + C	0.11
MAT A = (B) * D	0.53
MAT A = B – C	0.11
MAT A = INV(B), D	5.00
MAT A = TRN(B)	0.02
MAT PRINT A; (to CRT)	0.27
MAT A = B * C	4.20

Size: The maximum number of numeric array elements which can be accommodated in a machine of given memory size can be determined by evaluating the following expression:

$$\frac{M - H}{8}$$

where M = machine size (4K = 4096 bytes),
H = number of bytes in RAM for housekeeping (~ 700), program statements, and other variables.

Since eight bytes are needed for each numeric matrix element, take the square root of the result to find the size of a square matrix.

$$\left[\frac{M - H}{8}\right]^{1/2}$$

A 4K machine can accommodate a 20x20 matrix or a matrix of 420 elements; an 8K machine, a 30x30 matrix or a 960 element matrix, to a maximum of 63x63 elements (a 3969 element matrix) in a 32K machine.

For alphanumeric arrays, the denominator of the previous expressions would change from eight (8) to the desired element length (1 to 64 bytes inclusive).

INDEX

To help us to provide you with the best manuals possible, please make your comments and suggestions concerning this publication on the form below. Then detach, fold, tape closed and mail to us. All comments and suggestions become the property of Wang Laboratories, Inc. For a reply, be sure to include your name and address. Your cooperation is appreciated.

700-3332E

TITLE OF MANUAL: **MATRIX STATEMENTS REFERENCE MANUAL**

COMMENTS:

Fold

Fold

(Please tape. Postal regulations prohibit the use of staples.)

WANG

Fold

FIRST CLASS
PERMIT NO. 16
Lowell, Mass.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

— POSTAGE WILL BE PAID BY —

WANG LABORATORIES, INC.
ONE INDUSTRIAL AVENUE
LOWELL, MASSACHUSETTS 01851

Attention: Technical Writing Department

Fold

Cut along dotted line.

Printed in U.S.A.
13-1019