TAM is an interactive program which performs arithmetic computations on constants, variables, and one- or two-dimensional arrays. TAM operates from two-dimensional, hand-written input and generates two-dimensional output.

TAM is fundamentally a "one statement at a time" system. Each statement is accepted and, if legal, executed before the next statement is requested. However, statements can modify the TAM environment so that subsequent statements are affected. The program can recall and re-execute previous statements and has a powerful editing capability.

Wherever possible, TAM frees the user from explicit or ordered declaration. Storage is acquired through usage; unknown quantities are requested when needed.

TAM incorporates a powerful set of arithmetic operators on constants, variables, and one- and two-dimensional arrays. It also provides looping facilities, single statement functions, and user-defined input and output.

Some built-in functions, such as logarithm, and some built-in constants, such as π and e, are provided.
The work reported herein was supported by the Advanced Research Projects Agency of the Department of Defense under Contract DAHC15-67-C-0149, ARPA Order No. 1327, Amendment No. 3, Program Code No. 1D30, and 1P10, also supported by NASA Contract NAS12-526.

THE ASSISTANT MATHEMATICIAN

by

Joan Bebb

23 September 1971

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U.S. Government.

Distribution of this document is unlimited.
<table>
<thead>
<tr>
<th>Section</th>
<th>Table of Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td><strong>INTRODUCTION</strong></td>
</tr>
<tr>
<td>2.</td>
<td><strong>TAM ENTITIES</strong></td>
</tr>
<tr>
<td>2.1</td>
<td>Quantities</td>
</tr>
<tr>
<td>2.2</td>
<td>Identifiers</td>
</tr>
<tr>
<td>3.</td>
<td><strong>TAM OPERATORS</strong></td>
</tr>
<tr>
<td>3.1</td>
<td>Operators</td>
</tr>
<tr>
<td>3.2</td>
<td>Operational Hierarchy</td>
</tr>
<tr>
<td>3.3</td>
<td>Operator Definition</td>
</tr>
<tr>
<td>4.</td>
<td><strong>TAM STATEMENTS</strong></td>
</tr>
<tr>
<td>4.1</td>
<td>Assignment Statement</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Form</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Use</td>
</tr>
<tr>
<td>4.2</td>
<td>Function Definition Statement</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Form</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Use</td>
</tr>
<tr>
<td>4.3</td>
<td>Input Statement</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Form</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Use</td>
</tr>
<tr>
<td>4.4</td>
<td>Output Statement</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Form</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Use</td>
</tr>
<tr>
<td>4.5</td>
<td>Loop Control</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Forms</td>
</tr>
<tr>
<td>4.5.1.1</td>
<td>Form 1</td>
</tr>
<tr>
<td>4.5.1.2</td>
<td>Form 2</td>
</tr>
<tr>
<td>4.5.1.3</td>
<td>Form 3</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Use</td>
</tr>
<tr>
<td>4.6</td>
<td>Built-in Functions</td>
</tr>
</tbody>
</table>

Page numbers: 1, 1, 1, 2, 2, 3, 3, 4, 5, 5, 5, 6, 6, 7, 7, 7, 8, 8, 8, 8, 9, 9, 9, 9, 9, 10, 10
TABLE OF CONTENTS (cont'd)

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.</td>
<td>INPUTTING AND EDITING TAM STATEMENTS</td>
<td>11</td>
</tr>
<tr>
<td>5.1</td>
<td>Overview</td>
<td>11</td>
</tr>
<tr>
<td>5.2</td>
<td>Pushbuttons</td>
<td>12</td>
</tr>
<tr>
<td>6.</td>
<td>UNCONSTRAINED TAM</td>
<td>14</td>
</tr>
<tr>
<td>6.1</td>
<td>Minimum Declaration</td>
<td>14</td>
</tr>
<tr>
<td>6.2</td>
<td>Demand Input</td>
<td>15</td>
</tr>
<tr>
<td>6.3</td>
<td>Natural Expression of Operation</td>
<td>15</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Implicit Multiplication</td>
<td>16</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Imbedded Exponentiation</td>
<td>16</td>
</tr>
<tr>
<td>7.</td>
<td>ERROR CONDITIONS</td>
<td>17</td>
</tr>
<tr>
<td>7.1</td>
<td>User Errors</td>
<td>17</td>
</tr>
<tr>
<td>7.2</td>
<td>Interpreter Errors</td>
<td>19</td>
</tr>
<tr>
<td>7.3</td>
<td>Capacity-Overload Errors</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td><strong>APPENDIX A</strong> BUILDING A TAM DICTIONARY</td>
<td>22-26</td>
</tr>
<tr>
<td></td>
<td><strong>APPENDIX B</strong> CHARACTER CODES - STRING-PARSER-GENERATED CHARACTERS</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td><strong>APPENDIX C</strong> CHARACTER CODES - LEGAL USER CHARACTERS</td>
<td>28-30</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

TAM is an interactive program which performs arithmetic computations on constants, stored variables, and one- or two-dimensional arrays. TAM operates from two-dimensional, hand-written input and generates two-dimensional output.

TAM is fundamentally a "one statement at a time" system. Each statement is accepted and, if legal, executed before the next statement is requested. However, statements can modify the TAM environment so that subsequent statements are affected. The program can recall and re-execute previous statements and has a powerful editing capability.

Wherever possible, TAM frees the user from explicit or ordered declaration. Storage is acquired through usage; unknown quantities are requested when needed.

TAM incorporates a powerful set of arithmetic operators on constants, variables, and one- and two-dimensional arrays. It also provides looping facilities, single statement functions, and user-defined input and output.

Some built-in functions, such as logarithm, and some built-in constants, such as $\pi$ and $e$, are provided.

2. TAM ENTITIES

2.1 QUANTITIES

Quantities in TAM are either positive or negative, integral or mixed numbers. Internally, mixed numbers are carried in double-precision, floating-point form; integers are carried in one IBM/360 word (4 bytes). Therefore, effective precision is that defined for the IBM/360. Quantities may be contained in variables or arrays or expressed as constants. Most storage declaration is implied by usage. Arrays are dimensioned either implicitly or explicitly.
2.2 IDENTIFIERS

Variables and array identifiers are single-letter names. The legal alphabet of TAM consists of Greek and Roman uppercase and lowercase letters. An identifier may be made unique through the use of overscoring or underscoring. Legal overscore and underscore characters are:

Examples of legal identifiers are:

```
 a       A       a
 A       A       a
```

3. TAM OPERATORS

3.1 OPERATORS

Quantities may be manipulated through the use of operators as follows:

```
 a + b   addition
 a - b   subtraction
 ab, a*b, a\,*b  multiplication
 a      division
 a/b    exponentiation
 x^y    \quad n^{th} root \quad \sqrt[n]{}\rightarrow \sqrt[n]{x}
 !      factorial
 { }    absolute value
```
Note that implicit multiplication is allowed because all identifiers are single letters (possibly qualified). A special operator, $\mathbf{V}$, is used in conjunction with setting arrays and is explained subsequently.

### 3.2 OPERATIONAL HIERARCHY

Parenthesization is allowed to control the parsing hierarchy. In the absence of parenthesization, the hierarchy of operation is (from least to most binding):

- $\Sigma$ : summation
- $\Pi$ : product
- $+, -$ : addition, subtraction
- $\ast, /$ : multiplication, division
- $+, -$ : unary plus, unary minus
- $!$ : factorial
- $\{, [, ], \{, \}, \sqrt{\#}, T$ : floor, ceiling, function call, absolute value, root, exponentiation, transpose

### 3.3 OPERATOR DEFINITION

Each operator is usable when meaningful. With few exceptions (for example, transpose applies to matrices only; $(-3)!$ is signaled as an error), all operators are usable to manipulate single constants or variables. The operators...
are legal when applied to arrays where an acceptable matrix or vector operation is defined. For example,

\[
\begin{bmatrix}
1 & 2 \\
3 & 4
\end{bmatrix}^{-3} = \left( \begin{bmatrix}
1 & 2 \\
3 & 4
\end{bmatrix}^3 \right)^{-1}
\]

is defined if the matrix is square;

\[
\begin{bmatrix}
1 & 2 \\
3 & 4
\end{bmatrix}^{1/3}
\]

is not defined;

\[
\begin{bmatrix}
1 & 2 \\
3 & 4
\end{bmatrix} = \begin{bmatrix}
1 & 2 \\
3 & 4
\end{bmatrix} \times \begin{bmatrix}
5 & 6 \\
7 & 8
\end{bmatrix}^{-1}
\]

is defined if the divisor is a square matrix and the interior dimensions of the two matrices are the same;

and

\[
\begin{bmatrix}
1 & 5 \\
7 & 9
\end{bmatrix}!
\]

is not defined.

One-dimensional arrays are stored and treated as row vectors, with one exception. In multiplication, if one or both operands are vectors, the operand on the left (if a one-dimensional array) is treated as a row vector and the operand on the right (if a one-dimensional array) is treated as a column vector. The multiplication performed is the dot product. Therefore,

\[
\begin{bmatrix}
1 & 2 & 3
\end{bmatrix} \times \begin{bmatrix}
1 \\
2 \\
3
\end{bmatrix} = 14
\]

4. TAM STATEMENTS
There are five distinct TAM statements: assignment, function definition, input, output, and loop.
4.1 ASSIGNMENT STATEMENT

The assignment statement is used to set identifiable variables or arrays, presumably for use in subsequent statements.

4.1.1 Form

identifier + expression

The expression may consist of any legal manipulation of quantities.

4.1.2 Use

For example:

\[ \hat{A} \leftarrow 3 \]
Set variable \( \hat{A} \) to 3. (1)

\[ B \leftarrow \begin{bmatrix} 1 & 2 \\ \hat{A} \\ 4 & 5 & 6 \end{bmatrix} \]
Set array \( B \) to the given matrix (also dimension \( B \) as 2, 3). Each element of the matrix \( B \) may be any legal expression that yields a single numeric value. (2)

\[ C \leftarrow \mathbf{0}_{3,5} \]
Set all elements of array \( C \) to \( \mathbf{0} \) dimension \( C \) 3, 5. (3)

\[ D_1 \leftarrow 25.5 \]
Set first element of array \( D \) to 25.5. The dimensionality of \( D \) is unknown as yet. (4)

\[ X \leftarrow \sum_{j=1}^{\hat{A}} B_{1,j} \]
Set \( X \) to minus the sum of the elements of the first row (\( \hat{A}=3 \)) of \( B \). (5)
$Y = D_1^{-X} + [X \sqrt[3]{B_{2,2}}]$  \hspace{1cm} \text{(6)}

Set $Y$ to the sum of $D_1$ to the $-X^{th}$ power and the product of the absolute value of $X$ and the square root of $B_{2,2}$.

The third example illustrates the use of the special operator $\mathbf{V}$. It is used to declare, dimension, and preset an array (of one or two dimensions). The form of the operator is:

$$\mathbf{V}_{s,t}^p$$

where $s$ and $t$ are dimensions ($t$ and the preceding comma are optional) and $p$ (also optional) is the presetting value. $P$ may be any legal expression that yields a single numeric value.

4.2 FUNCTION DEFINITION STATEMENT

The TAM user may define frequently used arithmetic expressions as functions; he may then call upon these functions when necessary. Functions, of course, return values. The function definition and call may contain parameters. Both the function expression and the actual parameters of the call may contain calls to other functions.

4.2.1 Form

$$f_n(p_1,p_2,...,p_m) = \text{expression}$$

where $f$ is a legal identifier, $n$ is an optional alphabetic or numeric qualifier, and the $p_i$ are optional parameters. The expression may involve any legal manipulation of quantities. The identifier $f$, once it has been used as a function name, defines a class of functions $f_n$ and cannot be later used as a variable or array identifier. The various functions in class $f$ are distinguished from one another through the use of the qualifier $n$. For example $\bar{G}_1(X) = X^2$ and $\bar{G}_2(X) = X$ are two functions in class $\bar{G}$; $\bar{G} + 3$ is an illegal statement. $\tilde{G}$ and $G$ are not in class $\bar{G}$. As many function classes as desired
may be defined. The optional parameters, $p_i$, must be legal identifiers. The same identifier may be used as a parameter in many function definitions and also as a variable or array name or as a function class.

4.2.2 Use

An example of a function definition and call:

Function definition:

$$Z_3(a,b) = \frac{a}{\sqrt[3]{b}}$$

Function call:

$$\Theta = Z_3(27.2,3.14159) + y$$

To exponentiate the value returned by a function, it is permissible to write (for function $a$, parameters $b, c$, exponent 2):

$$a^2(b,c)$$

or

$$a(b,c)^2$$

4.3 INPUT STATEMENT

Because TAM requests values for undefined quantities as they are encountered, user-directed input is seldom necessary. However, an input statement is provided so that the user may guide the order of quantity setting in a direct fashion and so that he may reset quantities easily.

4.3.1 Form

$$[] \rightarrow \text{list}$$

where the list consists of legal identifiers of simple variables or arrays separated by commas.
4.3.2 Use
The statement results in a request for input(s) from the user. A value for each element in the list is requested in turn. For example:

\[ \Pi \rightarrow \Omega, \alpha \]

results in the user being prompted with:

\[ \Omega \rightarrow \]

Hopefully, the user will then respond with a value for \( \Omega \).

TAM will then request a value for \( \alpha \) by writing:

\[ \alpha \rightarrow \]

Once again, the user is expected to input a value for \( \alpha \).

4.4 OUTPUT STATEMENT

4.4.1 Form
list
where the list consists of expressions or identifiers separated by commas.

4.4.2 Use
The statement:

\[ B, 3 + 5 \]

results in the output

\[
\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6
\end{bmatrix}
\]

assuming \( B \) has been defined as the given matrix. If \( B \) has not yet been defined, the following, rather unusual, result occurs. The user is first asked to input a value for \( B \) by the prompt:

\[ B \rightarrow \]

After receiving \( B \), TAM outputs it.
4.5 LOOP CONTROL

An assignment statement, input statement, or output statement may be iterated by following the statement with loop-control information.

4.5.1 Forms

Loop control may be specified in three forms. Loops may be nested to any level, but each loop variable in the nest must be unique.

4.5.1.1 Loop Control, Form 1

Statement: \( i = m_0, \ldots, n \)

where \( i \) is the loop variable (an identifier of a simple variable whose value will be incremented by one for each iteration of the statement), \( m \) is the initial value for \( i \), and \( n \) is the terminal value. \( m \) and \( n \) may be any legal expressions that yield single numeric values. The iteration is complete when \( i \) exceeds \( n \). The statement iterated may, but need not, contain references to \( i \).

4.5.1.2 Loop control, Form 2

Statement: \( i = m_1, m_2, \ldots, n \)

where \( i \) is the loop variable, \( m_1 \) and \( m_2 \) are the first two values for \( i \) as the statement is iterated, \( m_2 - m_1 \) defines the loop increment (or decrement), and \( n \) is the terminal value. \( m_1, m_2 \), and \( n \) may be any legal expressions which yield single numeric values. The iteration is complete when \( i \) exceeds (or becomes less than) \( n \). The statement iterated may, but need not, contain references to \( i \).

4.5.1.3 Loop Control, Form 3

Statement: \( i = m_1, m_2, m_3, \ldots, n \)

where \( i \) is the loop variable and the \( m_j \) are successive settings for \( i \) each time the statement is iterated. The ellipsis \( (\ldots) \) shown is not a part of the

loop-control form, as it is in the two previous forms, but is included to indicate that the list \( m_j \) is of user-determined length. The loop terminates after the statement has been executed for \( i = n \). The statement may, but need not, contain references to \( i \).

4.5.2 Use

For example:

\[ X_{i,j} = Y_{i} + 1; \quad i = 1, \ldots, 5; \quad j = 2, 4, \ldots, 8 \]

is an instance of a two-level nest of loop control. The level of nesting proceeds from right to left, i.e., \( j \) is the outer control variable, \( i \) is the inner control variable.

Another example:

\[ A = AB; B = 86, \sqrt{4.7, 4.7} \]

\( B \) assumes the values \( 86, \sqrt{4.7, 4.7} \) for successive iterations of the loop. The final value for \( A \) is the sum of these values plus the original contents of \( A \).

4.6 BUILT-IN FUNCTIONS

TAM includes a set of built-in functions that the user can activate by including one of the names given below (along with an appropriate parameter) within any context in which a function call is permissible. (In expressing the name, any combination of uppercase and lowercase Roman letters is permissible; e.g., \( \ln : \ln = \ln \).

The available functions are:

<table>
<thead>
<tr>
<th>Name and Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sin(x) )</td>
<td>( \sin(x) )</td>
</tr>
<tr>
<td>( \cos(x) )</td>
<td>( \cos(x) )</td>
</tr>
<tr>
<td>( \tan(x) )</td>
<td>( \tan(x) )</td>
</tr>
<tr>
<td>( \cot(x) )</td>
<td>( \cot(x) )</td>
</tr>
<tr>
<td>( \arctan(x) )</td>
<td>( \arctan(x) )</td>
</tr>
<tr>
<td>( \tan^{-1}(x) )</td>
<td>( \arctan(x) )</td>
</tr>
<tr>
<td>( \ln(x) )</td>
<td>natural logarithm(x)</td>
</tr>
</tbody>
</table>

The arguments are in radians for the trigonometric functions.
4.5 LOOP CONTROL

An assignment statement, input statement, or output statement may be iterated by following the statement with loop-control information.

4.5.1 Loop control may be specified in three forms. Loops may be nested to any level, but each loop variable in the nest must be unique.

4.5.1.1 Loop Control, Form 1
Statement: \( \text{L} = \text{i} \rightarrow \text{m}, \text{m}, \ldots, \text{n} \)
where \( \text{i} \) is the loop variable (an identifier of a simple variable whose value will be incremented by one for each iteration of the statement), \( \text{m} \) is the initial value for \( \text{i} \), and \( \text{n} \) is the terminal value. \( \text{m} \) and \( \text{n} \) may be any legal expressions that yield single numeric values. The iteration is complete when \( \text{i} \) exceeds \( \text{n} \). The statement iterated may, but need not, contain references to \( \text{i} \).

4.5.1.2 Loop Control, Form 2
Statement: \( \text{L} = \text{i} \rightarrow \text{m} \rightarrow \text{n} \)
where \( \text{i} \) is the loop variable, \( \text{m} \) and \( \text{n} \) are the first two values for \( \text{i} \) as the statement is iterated, \( \text{m} - \text{n} \) defines the loop increment (or decrement), and \( \text{n} \) is the terminal value. \( \text{m} \) and \( \text{n} \) may be any legal expressions that yield single numeric values. The iteration is complete when \( \text{i} \) exceeds \( \text{n} \). The statement iterated may, but need not, contain references to \( \text{i} \).

4.5.1.3 Loop Control, Form 3
Statement: \( \text{L} = \text{i} \rightarrow \text{m}, \text{m}, \ldots, \text{n} \)
where \( \text{i} \) is the loop variable and the \( \text{m} \) are successive settings for \( \text{i} \) each time the statement is iterated. The ellipsis (\( \ldots \)) shown is not a part of the loop control. The loop terminates after the statement has been executed for \( \text{i} = \text{mn} \).

The statement may, but need not, contain references to \( \text{i} \).

For example:
\( \text{L} = \text{i} \rightarrow \text{m}, \text{m}, \ldots, \text{n} \)

Another example:
\( \text{L} = \text{i} \rightarrow \text{m}, \text{m}, \ldots, \text{n} \)

4.6 BUILT-IN FUNCTIONS

The available functions:
- \( \text{sin}(\text{x}) \)
- \( \text{cos}(\text{x}) \)
- \( \text{tan}(\text{x}) \)
- \( \text{cot}(\text{x}) \)
- \( \text{arctan}(\text{x}) \)
- \( \text{ln}(\text{x}) \)
- \( \text{arcsin}(\text{x}) \)
- \( \text{arccos}(\text{x}) \)
- \( \text{arctan}(\text{x}) \)

The arguments are in radians for the trigonometric functions.

The arguments are in radians for the trigonometric functions.
5. INPUTTING AND EDITING TAM STATEMENTS

5.1 OVERVIEW

TAM statements are input in hand-written two-dimensional form through a RAND-Tablet display device. TAM uses a previously constructed dictionary of patterns describing the user's individual handwriting characteristics and defining the recognizable character set (see Appendix A).

Once active at the terminal, TAM asks for a file description of this dictionary:

OLD DICTIONARY =

The user then responds with the appropriate file description.

TAM then displays a set of "pushbuttons" (e.g., the words "Exit" and "Erase") that allow the user to direct subsequent operations. The TAM pushbuttons and their meanings are given in Section 5.2.

To start operating, the user writes a TAM statement on the face of the display. The line will be parsed as recognized and an encoded form of the line will appear (see Appendix B for encoding characters). The user can then operate on his statement through use of the pushbuttons. As TAM operates, a history display is built of the last four legal lines and any associated output. Any of the last four lines entered can be recalled for editing and/or reoperation.
5.2 PUSHBUTTONS

Figure 1. TAM Pushbuttons

<table>
<thead>
<tr>
<th>Button</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Erase</td>
<td>The most recently input user characters, their related parsed display, and any associated record in the interpreter interface are deleted. A new or recalled line is now expected.</td>
</tr>
<tr>
<td>Move</td>
<td>The designated user characters are moved as specified by the editing stroke(s). The move operations and how they are indicated are:</td>
</tr>
</tbody>
</table>

**Open-up**—a single horizontal line drawn on or between previously scripted characters. A space equal to the length of the line is made between the selected characters. The initial point of the line determines the location of the space and the direction of the line determines the direction in which the affected characters are moved.

**Close-up**—two horizontal lines, similar to an equal sign, drawn in a space between previously scripted characters. The
**Button**

**Function**

Move (cont'd)  characters to the right of the editing strokes are moved to the left a distance equal to the length of the longest of the editing stroke pair.

Move character—a circle or a rectangle, drawn as one stroke, followed by a single line. The characters enclosed by the circle or rectangle are moved the length and direction of the line.

**Context**

This button may be used in conjunction with the Move-character operation and the character scrub (the latter consists of rubbing out characters with the pen). If the Context button is pushed immediately prior to the scrub action, any characters logically attached to the scrubbed character, such as a superscript, are erased. If the Context button is used in place of the Move button, any characters logically attached to a character enclosed by the first stroke of the move character are also moved.

**TAM**

The current user characters are processed by the interpreter. If no characters exist, the action is taken as a cancel signal. When the current line becomes the history display, the program is ready for new input. The interpreter's output, if not an error message, is joined with the parsed representation of the user's scripted input and added to the history display, and the previous entry in the history display is deleted. An internal program history pointer is set to the most recent history entry.

**Latest**

The user characters in the history display are regenerated and displayed.
**Button**  
**Function**

Backward  
The history pointer is moved from its current position to the next older entry. The entry's user characters are then regenerated and displayed.

Forward  
The history pointer is moved from its current position to the next more recent entry. The entry's user characters are then regenerated and displayed.

Quit  
Terminate TAM.

6. **UNCONSTRAINED TAM**

6.1 **MINIMUM DECLARATION**

One of the important design goals of TAM is that the user be freed from unnecessary or ordered declaration of storage, quantities, or functions. One manifestation of this freedom is that an array of elements need not be explicitly dimensioned until the user wishes to operate with the array as an entity, i.e., as a matrix or vector; so long as the user deals with his array element by element, no explicit dimensioning is necessary. Before the array is first used as a whole, the user must dimension it. Then, when the array is used, the user is asked to set elements that do not yet have values. In setting array values, the user may either override or maintain existing values.

TAM gives the user this freedom by incorporating the following techniques. An identifier—a single-letter name along with any overscore and/or underscore characters expressed with the letter—is classified into one of two classes, function or variable, on the basis of its first usage. That is, the four distinct identifiers

\[ A \bar{A} \bar{A}\bar{A} \]

will be separately classified according to how
they are first used. Subscripts used with identifiers discriminate between
class members by type. Each member of a class has associated with it a type:
list, vector, array, or function.

For example:

\[ A_1 \leftarrow 3 \text{ sets } A \text{ to class 'variable' and } A_1 \text{ to type 'list'.} \]

Before \( A \) can be used as a vector, dimensioning information must be given about
\( A \). The type of \( A \) there will be set to 'vector'. If necessary, TAM will
demand values for unset members of class \( A \) when it is used.

6.2 DEMAND INPUT

TAM further frees the user from ordered statements in that quantities to be
manipulated need have values only when used. TAM automatically provides
the necessary prompting. For example:

\[ A \leftarrow 2 + Y \]

as the first TAM statement would result in the request

\[ 2 \leftarrow \]

to which the user would be expected to respond with a value or expression.
If the user's response contains unknown quantities, TAM also requests values
for these. Once a value for \( 2 \) is ascertained, TAM requests \( Y \):

\[ Y \leftarrow \]

Once again, a value or expression is mandatory. Once all quantities are defined,
\( A \) is set to the value of the original expression \( 2 + Y \).

6.3 NATURAL EXPRESSION OF OPERATION

TAM allows the user to express standard mathematical operations in a natural
way. Two-dimensional input is TAM's outstanding facility in this regard. In
addition, two features of standard notation usually disallowed in programming languages are permitted in TAM: implicit multiplication and embedded exponentiation in function calls.

6.3.1 Implicit Multiplication
TAM allows the user to imply the multiplication operator. The nature of TAM identifiers eliminates contextual confusion. The following are examples of implied multiplication:

\[
\begin{align*}
ab & \Rightarrow a \times b \\
c(de) & \Rightarrow c \times (d \times e) \\
5 & \Rightarrow 5 \\
e \sum_{i=3}^{4} \Rightarrow e^{i=3} \times 4 \\
(ij)! & \Rightarrow i \times (j)! \\
(ij)! & \Rightarrow (i \times j)! \\
\end{align*}
\]

Note: \(ij! \Rightarrow i \times (j!\)\)

and

\[
\begin{align*}
ab^{2} & \Rightarrow a \times (b^{2}) \\
(ab)^{2} & \Rightarrow (a \times b)^{2} \\
\end{align*}
\]

6.3.2 Embedded Exponentiation
Exponentiation of the value returned by a function can be expressed in either of two ways:

\[
\begin{align*}
\sin(a)^{2} & \Rightarrow (\sin(a))^{2} \\
or \sin^{2}(a) & \Rightarrow (\sin(a))^{2} \\
\end{align*}
\]

That is, the exponent may appear immediately after the full function call (which includes parameters) or immediately adjacent to the function name, before the parameters.
7. **ERROR CONDITIONS**

TAM legality checks each statement before attempting to execute it. If an illegality is discovered, the statement is rejected and an appropriate message is generated.

7.1 **USER ERRORS**

The most frequent error message generated by TAM is:

**SYNTAX ERROR AT CHARACTER XX**

where XX is the relative character position (in the parsed string) of the illegality. Other possible error messages are:

<table>
<thead>
<tr>
<th>Message</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE ERR</td>
<td>1. An identifier defined as a function name is being used as a variable.</td>
</tr>
<tr>
<td></td>
<td>2. An identifier defined as a variable or array name is being used as a function.</td>
</tr>
<tr>
<td></td>
<td>3. The control variable in a loop statement is an array or function name.</td>
</tr>
<tr>
<td>X! X ≠ INTEGER ≥ 0</td>
<td>The factorial operator is defined for positive integers only.</td>
</tr>
</tbody>
</table>

**CLASS ERR**

1. An identifier which is a member of a class of functions is being used as a variable.

2. An identifier which is in a variable or array class is being defined as a function.
<table>
<thead>
<tr>
<th>Message</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPERATION ERR - ARRAY/VECTOR</td>
<td>An operator is being used which is meaningless if one or both operands is an array or vector.</td>
</tr>
<tr>
<td>X&lt;sub&gt;Y&lt;/sub&gt;, Y = INTEGER</td>
<td>For negative X, only exponentiation to an integral power is defined.</td>
</tr>
<tr>
<td>PARAMETER ERR</td>
<td>The number of parameters in a function call does not match the number of parameters in the definition.</td>
</tr>
<tr>
<td>≠ COLUMNS/ROWS</td>
<td>In a block input of an array, the columns are of unequal length or the rows are of unequal width.</td>
</tr>
<tr>
<td>SUBSCRIPTS MUST BE INTEGERS</td>
<td>An attempt to invert a singular matrix has been made.</td>
</tr>
<tr>
<td>SINGULAR MATRIX, NO SOLUTION</td>
<td>Ln(X), X &lt; Q returns a very small number.</td>
</tr>
<tr>
<td>LOG OF - OR Q ARG RETURNS - INFINITY</td>
<td></td>
</tr>
<tr>
<td>SYMBOL OR STRING TOO LONG</td>
<td></td>
</tr>
</tbody>
</table>
Message
MISMATCHED DIMENSIONS

Meaning
An arithmetic operation on two arrays has been attempted. The nature of the arrays precludes the operation.

EXPONENT OVERFLOW EXCEPTION

EXPONENT UNDERFLOW EXCEPTION

SIGNIFICANCE EXCEPTION

FLOATING DIVIDE EXCEPTION

FLOAT /= NO.

An array or vector has been specified in a context that demands a simple variable, e.g., as a preset value for an array or vector.

7.2 INTERPRETER ERRORS

The following messages should not, but may, appear. The program will stop. The messages signal errors in the TAM interpreter. If one of the following messages is encountered, a register and core dump should be taken and the TAM maintenance personnel contacted.

AN UNDEFINED ROUTINE HAS BEEN CALLED
BAD CAR
BAD CDR
GENERATOR DOES NOT MATCH ITS PARAMETERS
ALL BOOLEANS OF AN EXPRESSION ARE FALSE
BAD CADD
BAD CADD2
7.3 CAPACITY OVERLOAD ERRORS

The following messages signal that the capacities of TAM have been exceeded. The program will stop. If one of these messages is encountered and the excess seems unwarranted or ill defined, a core and register dump should be taken and TAM maintenance personnel contacted.
FULL UNDEFINED STACK
FULL BACKUP STACK
FULL ARGUMENT STACK
NO MORE ARRAY SPACE
NOT ENOUGH ARRAY SPACE
NOT ENOUGH LIST SPACE
NO MORE LARGE NUMBER SPACE
NO MORE REAL NUMBER SPACE
NO MORE GENSYM SPACE
NOT ENOUGH STRING SPACE
APPENDIX A

BUILDING A TAM DICTIONARY

To operate TAM, the user must have, at some previous time, built a dictionary of recognizable characters. This dictionary provides the information to allow TAM to recognize and accept the user's individual handwriting and character set. Each character to be used with TAM must be entered into the dictionary.

A program, BUILDER, is provided for dictionary construction. The program will either create a new dictionary or allow the user to modify and amplify an existing one.

Using the Dictionary Builder
After signing on, the program asks:

CONTINUE W/ OLD DICT ? Y/N

A "y" answer will cause the program to request file description information and open an old dictionary file. An "n" answer causes the program to assume a new dictionary is being built.

Function pushbuttons will then appear on the display. The pushbuttons and their functions are:

<table>
<thead>
<tr>
<th>Pushbutton</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save</td>
<td>Save the dictionary on disc or tape</td>
</tr>
<tr>
<td>Build</td>
<td>Build a dictionary</td>
</tr>
<tr>
<td>Test</td>
<td>Test the dictionary definitions</td>
</tr>
<tr>
<td>Sumry</td>
<td>Print a summary of the dictionary contents</td>
</tr>
<tr>
<td>Print</td>
<td>Print a detailed description of the dictionary</td>
</tr>
<tr>
<td>Quit</td>
<td>Quit the program</td>
</tr>
</tbody>
</table>

(Only the Build and Save functions will be described.)
Build Function
The user builds a dictionary containing stroke information and the definition associated with the strokes. The stroke information consists of the user's own individual characteristics in drawing characters.

Sampling Mode:
The program asks for user input by displaying "INPUT" at the top of the display. The user then inputs the strokes he wants to define. Note that the strokes need not be similar to each other.

Keyboard
Input strokes
Dictionary matches/no matches

Program/Control Pushbuttons
Threshold pushbuttons

Keyboard pushbuttons
Keyboard, keyboard pushbuttons. There are five keyboards, each consisting of a unique set of definitions. The user selects the set containing the definitions he needs by touching one of the keyboard pushbuttons: "S" for special characters, such as the asterisk and plus sign, "G" for Greek characters, "N" for numerals 0 through 9, "U" for upper case Roman alphabet, and "L" for lower case Roman alphabet.

The uppercase Roman alphabet is the first keyboard set displayed; the last set selected will be displayed at the top of the display until another set is selected.

Dictionary matches/no matches. Each stroke is analyzed and compared to definitions already in the dictionary. If a match has been found (matches are subject to threshold values described below), the character associated with that match is displayed directly below the stroke. This character will either belong to one of the keyboard sets or be a backward "S". The backward "S" represents a match in the dictionary that has no character definition, i.e., is only a part of a defined character.

If no match has been found, a backward (to distinguish it from the character "?") question mark is displayed directly below the stroke.

These defines/undefines are displayed about an inch below the string of input strokes, each define/undefine aligned with its stroke, at successively lower levels to prevent overlapping of definitions.

If the strokes are input below the bottom third of the screen, or if the layering of defines/undefines results in the overlapping of pushbuttons, then the defines/undefines appear above the threshold pushbuttons in the order in which their corresponding strokes were drawn.
Threshold pushbuttons. These buttons appear at the bottom center of the display, "RT" for reduced thresholds and "IT" for increased thresholds. A box appears around the threshold button last selected; the reduced thresholds are in effect first. These thresholds are used in determining whether or not a match exists for a given stroke/strokes.

Program/Control pushbuttons.

- **Redefn**
  - Redefine button, allows the user to change a dictionary definition.
- **Forget**
  - Forget button, ignore a previous action.
- **Erase**
  - Clear the display and return to the sampling mode ("INPUT" displayed.)
- **Exit**
  - Leave the Build function and return to function selecting mode.

Defining Mode.

A definition is entered in the dictionary by circling the stroke/strokes or either end point of a stroke and touching the corresponding character in the keyboard.

These definitions may be entered one at a time or as many as the user wishes, subject to program limitations.
The matches/no matches appear again below the input strokes. There may be all matches or there may still be some failures to match, as shown at the left.

The user may define the strokes for which no matches exist until all have matches in the dictionary.

Note: The possible matches are done using the thresholds in effect. Increasing the threshold values by touching the "IT" pushbutton may result in more matches.

To change the current definition in the dictionary for a certain stroke or strokes, the user should first touch the "Redefn" pushbutton and then define the stroke or strokes in the manner described above.

When satisfied that the current set of strokes has been properly defined, the user should touch either the "Erase" pushbutton to input a new set of strokes or the "Exit" pushbutton to leave the Build mode.

Save Function
To save the dictionary just built (for future use with TAM or for subsequent amplification), the user should touch the "Save" pushbutton. The program then requests a file identification (on the terminal). If a new file is to be opened, 'N' must be included in the file identification. Once the dictionary has been saved successfully, the message "DICT SAVED" appears on the terminal.
## APPENDIX B

**CHARACTER CODES - STRING-PARSER-GENERATED CHARACTERS**

<table>
<thead>
<tr>
<th>HEX</th>
<th>Octal</th>
<th>Decimal</th>
<th>Parse</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>301</td>
<td>193</td>
<td></td>
<td>Overscore; Upper Integral limit</td>
</tr>
<tr>
<td>C2</td>
<td>302</td>
<td>194</td>
<td></td>
<td>Superscript</td>
</tr>
<tr>
<td>C3</td>
<td>303</td>
<td>195</td>
<td></td>
<td>Subscript</td>
</tr>
<tr>
<td>C4</td>
<td>304</td>
<td>196</td>
<td></td>
<td>Underscore, Lower Integral limit</td>
</tr>
<tr>
<td>C5</td>
<td>305</td>
<td>197</td>
<td></td>
<td>Sq. Rt. index</td>
</tr>
<tr>
<td>D1</td>
<td>327</td>
<td>209</td>
<td>[</td>
<td>Left Bracket</td>
</tr>
<tr>
<td>D2</td>
<td>322</td>
<td>210</td>
<td>]</td>
<td>Right Bracket</td>
</tr>
<tr>
<td>D3</td>
<td>323</td>
<td>211</td>
<td>{</td>
<td>Numerator and Denominator enclosures</td>
</tr>
<tr>
<td>D4</td>
<td>324</td>
<td>212</td>
<td>}</td>
<td></td>
</tr>
<tr>
<td>D5</td>
<td>325</td>
<td>213</td>
<td>{</td>
<td>Multiple overscore enclosures</td>
</tr>
<tr>
<td>D6</td>
<td>326</td>
<td>214</td>
<td>}</td>
<td></td>
</tr>
<tr>
<td>D7</td>
<td>327</td>
<td>215</td>
<td>{</td>
<td>Matrix enclosures</td>
</tr>
<tr>
<td>D8</td>
<td>330</td>
<td>216</td>
<td>}</td>
<td></td>
</tr>
<tr>
<td>D9</td>
<td>331</td>
<td>217</td>
<td>{</td>
<td>Combinatorial enclosures</td>
</tr>
<tr>
<td>DA</td>
<td>332</td>
<td>218</td>
<td>}</td>
<td></td>
</tr>
<tr>
<td>EO</td>
<td>340</td>
<td>224</td>
<td></td>
<td>Square Root Bar</td>
</tr>
<tr>
<td>FO</td>
<td>360</td>
<td>240</td>
<td></td>
<td>Matrix Element Separator</td>
</tr>
<tr>
<td>F1</td>
<td>361</td>
<td>241</td>
<td>/</td>
<td>Fraction line</td>
</tr>
<tr>
<td>F2</td>
<td>362</td>
<td>242</td>
<td>\</td>
<td>Multiple overscore indicator</td>
</tr>
<tr>
<td>F3</td>
<td>363</td>
<td>243</td>
<td>x</td>
<td>Multiply cross</td>
</tr>
<tr>
<td>F4</td>
<td>364</td>
<td>244</td>
<td>\</td>
<td>Multiple underscore indicator</td>
</tr>
<tr>
<td>F6</td>
<td>366</td>
<td>246</td>
<td>;</td>
<td>Matrix Row Separator</td>
</tr>
<tr>
<td>FD</td>
<td>375</td>
<td>253</td>
<td></td>
<td>String Start</td>
</tr>
<tr>
<td>FE</td>
<td>376</td>
<td>254</td>
<td></td>
<td>String End</td>
</tr>
</tbody>
</table>
## APPENDIX C

### USER INPUT CHARACTERS

<table>
<thead>
<tr>
<th>Octal</th>
<th>Hex</th>
<th>Character</th>
<th>Octal</th>
<th>Hex</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>00</td>
<td>n/a</td>
<td>040</td>
<td>20</td>
<td>Blank</td>
</tr>
<tr>
<td>001</td>
<td>01</td>
<td>a</td>
<td>041</td>
<td>21</td>
<td>!</td>
</tr>
<tr>
<td>002</td>
<td>02</td>
<td>β</td>
<td>042</td>
<td>22</td>
<td>&quot;</td>
</tr>
<tr>
<td>003</td>
<td>03</td>
<td>α</td>
<td>043</td>
<td>23</td>
<td>#</td>
</tr>
<tr>
<td>004</td>
<td>04</td>
<td>δ</td>
<td>044</td>
<td>24</td>
<td>$</td>
</tr>
<tr>
<td>005</td>
<td>05</td>
<td>ε</td>
<td>045</td>
<td>25</td>
<td>%</td>
</tr>
<tr>
<td>006</td>
<td>06</td>
<td>₪</td>
<td>046</td>
<td>26</td>
<td>§</td>
</tr>
<tr>
<td>007</td>
<td>07</td>
<td>η</td>
<td>047</td>
<td>27</td>
<td>•</td>
</tr>
<tr>
<td>010</td>
<td>08</td>
<td>λ</td>
<td>050</td>
<td>28</td>
<td>•</td>
</tr>
<tr>
<td>011</td>
<td>09</td>
<td>μ</td>
<td>051</td>
<td>29</td>
<td>\</td>
</tr>
<tr>
<td>012</td>
<td>0A</td>
<td>γ</td>
<td>052</td>
<td>2A</td>
<td>*</td>
</tr>
<tr>
<td>013</td>
<td>0B</td>
<td>π</td>
<td>053</td>
<td>2B</td>
<td>+</td>
</tr>
<tr>
<td>014</td>
<td>0C</td>
<td>ρ</td>
<td>054</td>
<td>2C</td>
<td>,</td>
</tr>
<tr>
<td>015</td>
<td>0D</td>
<td>σ</td>
<td>055</td>
<td>2D</td>
<td>-</td>
</tr>
<tr>
<td>016</td>
<td>0E</td>
<td>τ</td>
<td>056</td>
<td>2E</td>
<td>•</td>
</tr>
<tr>
<td>017</td>
<td>0F</td>
<td></td>
<td>057</td>
<td>2F</td>
<td>\</td>
</tr>
<tr>
<td>020</td>
<td>10</td>
<td></td>
<td>060</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>021</td>
<td>11</td>
<td></td>
<td>061</td>
<td>31</td>
<td>1</td>
</tr>
<tr>
<td>022</td>
<td>12</td>
<td></td>
<td>062</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>023</td>
<td>13</td>
<td></td>
<td>063</td>
<td>33</td>
<td>3</td>
</tr>
<tr>
<td>024</td>
<td>14</td>
<td></td>
<td>064</td>
<td>34</td>
<td>4</td>
</tr>
<tr>
<td>025</td>
<td>15</td>
<td></td>
<td>065</td>
<td>35</td>
<td>5</td>
</tr>
<tr>
<td>026</td>
<td>16</td>
<td></td>
<td>066</td>
<td>36</td>
<td>6</td>
</tr>
<tr>
<td>027</td>
<td>17</td>
<td></td>
<td>067</td>
<td>37</td>
<td>7</td>
</tr>
<tr>
<td>030</td>
<td>18</td>
<td></td>
<td>070</td>
<td>38</td>
<td>8</td>
</tr>
<tr>
<td>031</td>
<td>19</td>
<td></td>
<td>071</td>
<td>39</td>
<td>9</td>
</tr>
<tr>
<td>032</td>
<td>1A</td>
<td></td>
<td>072</td>
<td>3A</td>
<td>;</td>
</tr>
<tr>
<td>033</td>
<td>1B</td>
<td>{</td>
<td>073</td>
<td>3B</td>
<td>;</td>
</tr>
<tr>
<td>034</td>
<td>1C</td>
<td>}</td>
<td>074</td>
<td>3C</td>
<td>&lt;</td>
</tr>
<tr>
<td>035</td>
<td>1D</td>
<td>)</td>
<td>075</td>
<td>3D</td>
<td>=</td>
</tr>
<tr>
<td>036</td>
<td>1E</td>
<td>}</td>
<td>076</td>
<td>3E</td>
<td>&gt;</td>
</tr>
<tr>
<td>037</td>
<td>1F</td>
<td>-</td>
<td>077</td>
<td>3F</td>
<td>?</td>
</tr>
<tr>
<td>Octal</td>
<td>Hex</td>
<td>Character</td>
<td>Octal</td>
<td>Hex</td>
<td>Character</td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
<td>----------</td>
<td>-------</td>
<td>-----</td>
<td>----------</td>
</tr>
<tr>
<td>100</td>
<td>40</td>
<td>@</td>
<td>140</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>41</td>
<td>A</td>
<td>141</td>
<td>61</td>
<td>a</td>
</tr>
<tr>
<td>102</td>
<td>42</td>
<td>B</td>
<td>142</td>
<td>62</td>
<td>b</td>
</tr>
<tr>
<td>103</td>
<td>43</td>
<td>C</td>
<td>143</td>
<td>63</td>
<td>c</td>
</tr>
<tr>
<td>104</td>
<td>44</td>
<td>D</td>
<td>144</td>
<td>64</td>
<td>d</td>
</tr>
<tr>
<td>105</td>
<td>45</td>
<td>E</td>
<td>145</td>
<td>65</td>
<td>e</td>
</tr>
<tr>
<td>106</td>
<td>46</td>
<td>F</td>
<td>146</td>
<td>66</td>
<td>f</td>
</tr>
<tr>
<td>107</td>
<td>47</td>
<td>G</td>
<td>147</td>
<td>67</td>
<td>g</td>
</tr>
<tr>
<td>110</td>
<td>48</td>
<td>H</td>
<td>150</td>
<td>68</td>
<td>h</td>
</tr>
<tr>
<td>111</td>
<td>49</td>
<td>I</td>
<td>151</td>
<td>69</td>
<td>i</td>
</tr>
<tr>
<td>112</td>
<td>4A</td>
<td>J</td>
<td>152</td>
<td>6A</td>
<td>j</td>
</tr>
<tr>
<td>113</td>
<td>4B</td>
<td>K</td>
<td>153</td>
<td>6B</td>
<td>k</td>
</tr>
<tr>
<td>114</td>
<td>4C</td>
<td>L</td>
<td>154</td>
<td>6C</td>
<td>l</td>
</tr>
<tr>
<td>115</td>
<td>4D</td>
<td>M</td>
<td>155</td>
<td>6D</td>
<td>m</td>
</tr>
<tr>
<td>116</td>
<td>4E</td>
<td>N</td>
<td>156</td>
<td>6E</td>
<td>n</td>
</tr>
<tr>
<td>117</td>
<td>4F</td>
<td>O</td>
<td>157</td>
<td>6F</td>
<td>o</td>
</tr>
<tr>
<td>120</td>
<td>50</td>
<td>P</td>
<td>160</td>
<td>70</td>
<td>p</td>
</tr>
<tr>
<td>121</td>
<td>51</td>
<td>Q</td>
<td>161</td>
<td>71</td>
<td>q</td>
</tr>
<tr>
<td>122</td>
<td>52</td>
<td>R</td>
<td>162</td>
<td>72</td>
<td>r</td>
</tr>
<tr>
<td>123</td>
<td>53</td>
<td>S</td>
<td>163</td>
<td>73</td>
<td>s</td>
</tr>
<tr>
<td>124</td>
<td>54</td>
<td>T</td>
<td>164</td>
<td>74</td>
<td>t</td>
</tr>
<tr>
<td>125</td>
<td>55</td>
<td>U</td>
<td>165</td>
<td>75</td>
<td>u</td>
</tr>
<tr>
<td>126</td>
<td>56</td>
<td>V</td>
<td>166</td>
<td>76</td>
<td>v</td>
</tr>
<tr>
<td>127</td>
<td>57</td>
<td>W</td>
<td>167</td>
<td>77</td>
<td>w</td>
</tr>
<tr>
<td>130</td>
<td>58</td>
<td>X</td>
<td>170</td>
<td>78</td>
<td>x</td>
</tr>
<tr>
<td>131</td>
<td>59</td>
<td>Y</td>
<td>171</td>
<td>79</td>
<td>y</td>
</tr>
<tr>
<td>132</td>
<td>5A</td>
<td>Z</td>
<td>172</td>
<td>7A</td>
<td>z</td>
</tr>
<tr>
<td>133</td>
<td>5B</td>
<td>[</td>
<td>173</td>
<td>7B</td>
<td></td>
</tr>
<tr>
<td>134</td>
<td>5C</td>
<td>\</td>
<td>174</td>
<td>7C</td>
<td></td>
</tr>
<tr>
<td>135</td>
<td>5D</td>
<td>]</td>
<td>175</td>
<td>7D</td>
<td></td>
</tr>
<tr>
<td>136</td>
<td>5E</td>
<td>+</td>
<td>176</td>
<td>7E</td>
<td></td>
</tr>
<tr>
<td>137</td>
<td>5F</td>
<td>+</td>
<td>177</td>
<td>7F</td>
<td>EOM</td>
</tr>
<tr>
<td>Octal</td>
<td>Hex</td>
<td>Character</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
<td>-----------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>80</td>
<td>reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>201</td>
<td>81</td>
<td>[</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>202</td>
<td>82</td>
<td>]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>203</td>
<td>83</td>
<td>[</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>204</td>
<td>84</td>
<td>]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>205</td>
<td>85</td>
<td>]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>206</td>
<td>86</td>
<td>]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>207</td>
<td>87</td>
<td>≈</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>210</td>
<td>88</td>
<td>^</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>211</td>
<td>89</td>
<td>□</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>212</td>
<td>8A</td>
<td>\</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>