ATRAN PROGRAMMING SYSTEM
Users Manual

PBC 1020

pb 250

pb Packard Bell Computer

ATRAN PROGRAMMING SYSTEM
Users Manual

PBC 1020

pb Packard Bell Computer

A DIVISION OF PACKARD BELL ELECTRONICS
1905 ARMACOST AVENUE • LOS ANGELES 25, CALIFORNIA

June 30, 1962

NOTICE

TABLE OF CONTENTS

I. INTRODUCTION

ATRAN (Algebraic Translator) is a programming system which allows PB-250 users to write programs in a language that resembles the familiar algebraic notation associated with engineering calculations. It accepts a source program written in the ATRAN language and produces an object program in the CINCH language. When the ATRAN program is finished, the object program compiled by it is left in memory, ready to be executed by the CINCH interpreter.

Before presenting the reader with a mass of detailed rules, it is the plan of this manual to present in a general way, by means of examples, what the ATRAN language is and its relationship to familiar algebraic expressions. We will then give detailed rules governing the use of the various elements of the language. The manual will be concluded with a summary of the ATRAN language, some illustrations of ATRAN programs, and operating instructions for running the ATRAN system.

II. THE ATRAN LANGUAGE AND ITS USE

Example 1

The algebraic equation,

$$y = ax^3 + bx^2 + cx - 10 - \frac{1}{x} + 3.5\sqrt{x}$$

is a string of constants and two kinds of symbols, which denote (1) operators, and (2) data (i. e. parameters, variables). As an equation, it means, "y is equivalent to the expression on the right of =". As a procedural statement, it means, "Compute y by evaluating the expression on the right for given values of x, a, b, and c". It is this meaning with which we are concerned here.

The following ATRAN procedural statements will instruct the PB-250 to evaluate the equation using the values 5. 25, 3. -3, and . 5 for A, B, C, and X, respectively, and print out the result. Notice that all statements must end with $.

```
A = 5. 25$                                          (1)
B = 3$                                              (2)
C = -3. $                                           (3)
X = 0.5$                                            (4)
Y = A*X↑3+B*X↑2+C*X-10-1/X+3. 5*√X$                 (5)
PRA:Y = $                                           (6)
PRV:(5. 3C)Y$                                       (7)
```

The first five statements are Arithmetic Statements. The first four of these are the simplest possible type; their purpose is to set the variables on the left of = to the values ("constants") on the right. The fifth statement evaluates the expression on the right of = and replaces y with the result, using the previously set values of A, B, C, and X. This statement uses all seven ATRAN operators. The order of operations is left to right within this hierarchy:

1. Exponentiation and square root, ↑ , √
2. Multiply and divide, *, /
3. Plus and minus, +, -

Parentheses, ( ), and brackets, [ ], may be used to help control the sequence of calculation; this will be discussed later.

The last two statements are Output Statements. Line 6 causes "Y =" to be typed. "Y =" is called a "Literal". Line 7 causes the value of Y to be typed. The information in (5. 3C) is for format control; five digits will be printed to the left of the decimal point and three will be printed to the right. "C" causes a carriage return after typing.

Example 2 - Introducing Decision Statements

$$y = ax^2 + bx - \frac{1}{x} - 10$$

Evaluate this equation for progressive values of x, from x = . 5 to x = 10. 0, in increments of x = . 5. Type out x and y. Use a = 5. 25, b = 3.

ATRAN Statements

```
A = 5. 25$ B = 3$                          (1)
X = . 5$                                   (2)
PNT:1$                                     (3)
Y = A*X*X+B*X-1/X-10$                      (4)
PRA:X = $ PRV:(2. 2T)$                     (5)
PRV:Y = $ PRV:(5. 3C)$                     (6)
X = X+. 5$                                 (7)
FL = X-10. 5$                              (8)
TST:FL$                                    (9)
MJP:1$                                     (10)
HLT:99$                                    (11)
```

Comment

Line 9 sets up the quantity FL for testing. Line 10 is a conditional jump statement; if FL is negative, a jump to jump point number 1 will be executed; otherwise the next statement in sequence will be executed. Line 4, PNT:1, establishes the immediately following statement as Jump-Point #1. ("PNT:" is not a procedural statement; its function is labeling or identification.) Thus, a decision actually involves three ATRAN statements: (1) TST:--, which is immediately followed by (2) a conditional jump or skip statement, and (3) a PNT:-- statement, which identifies the destination of the conditional jump. There are six conditional jump and skip commands in ATRAN.

Example 3 - Introducing Use of Subscripts and Data Input

$$y = x^2 + 5x + 3.4$$

Evaluate y for the following group of x values, successively, and type x and y. x = 1, 3, 7, 0, -1, -.5, +.5.

ATRAN Statements

```
DIM:X(7)$                                   (1)
LOD:X$                                      (2)
I = 1$                                      (3)
PNT:22$                                     (4)
Y = X(I)*X(I)+5*X(I)+3.4$                   (5)
PRA:X = $ PRV:(2.2T)X(I)$                   (6)
PRA:Y = $ PRV:(2.2C)Y$                      (7)
I = I+1$                                    (8)
TST:I$                                      (9)
XGR:7$                                      (10)
JMP:22$                                     (11)
HLT:99$                                     (12)
```

General Comments

Up to now, each symbol denoting a variable has referred to a single quantity. In this example, however, we have one variable, X, which refers to a group of seven different numbers. This group is usually called a list or a vector; a group of vectors may be called an array or a matrix. In order to refer to a particular value within the list or array, a subscript is added to the name of the variable. In the ATRAN language, a subscript is denoted by writing it in parentheses, ( ), e. g. X_i is written X(I). Previous examples have had their data built into the program in the form of constants; this one has been designed to read its data from paper tape at execution time.

Specific Comments

Line 1 is not a procedural statement; it is a dimension statement. Its purpose is to cause seven locations to be reserved for the list X. Dimension statements must be at the head of the program.

Line 2 is an input statement; it will cause the list X to be filled with seven numbers from a data tape.

Line 3: The subscript I is set to 1. The first element of any variable which refers to subscript I will now be referenced.

Line 8: The subscript I is increased by 1.

Lines 9 and 10: I is tested. If I is greater than 7, the next sequential statement is skipped; otherwise, the one following it will be executed.

III. DETAILED DESCRIPTION OF THE LANGUAGE

A. Constants

Constants used in ATRAN statements consist of numbers of up to five digits to the left and to the right of the decimal point. Negative numbers must be written with a minus sign to the left of the number. Positive numbers may be written either with a plus sign or with no sign. If the number is an integer, the decimal point may be omitted. An integer written in this manner may have no more than five digits.

Examples:

12345.12345	0	467
-54321.54321	0.	467.
+24.62	-.5	456.0
-.00005	0.5	

The largest constant allowed is 99999.99999; the smallest is -99999.99999; the smallest fraction is .00001 (in magnitude).

Examples of incorrect constants:

12.123456

123456

72.4E-7

B. Variables

There are two kinds of variables in the ATRAN language: undimensioned variables, which refer to single quantities, and dimensioned variables, which refer to lists or arrays of quantities. In addition, there is a third type of variable, called a subscript variable, or simply a subscript, which behaves like an undimensioned variable but also has another use.

1. Undimensioned Variables

Undimensioned variables may be expressed by either one or two alphabetic characters, the first (or only) character being any letter except I, J, K, L, M, or N. Examples of undimensioned variables are:

C

HI

PK

X

AA

2. Subscript Variables

Subscripts are represented by either one or two alphabetic characters, the first (or only) character being I, J, K, L, M, or N. Examples of subscript variables are:

I

IA

KZ

K

Subscripts may be used in the same context as undimensioned variables to refer to a single quantity, such as in:

```
I = I+1
KA = 3*J
```

However, once the value of a subscript variable has been defined by a previous ATRAN statement, it may be used truly as a subscript, as in:

```
A(I) = B(I)+C(I)
A(I, J) = B(I, K)*C(K, J)
```

When used in such a manner, its value must be an integer from 1 to 99.

If two subscripts are necessary to refer to a quantity, the individual subscripts must be separated by a comma, and the result is called a

subscript combination. Examples of subscript combinations are:

I, J

KZ, LL

M, NF

NE, K

3. Dimensioned Variables

ATRAN accepts only two types of dimensioned variables: one-dimensioned subscripted variables (called lists or vectors), and two-dimensioned subscripted variables (called arrays or matrices).

An item in a list is referred to by the name of the list and a subscript. The name of a list is expressed by either one or two alphabetic characters, the first (or only) character of which is any letter except I, J, K, L, M, or N. The subscript follows the name of the list and is enclosed in parentheses. Examples are:

A(I)

CK(M)

QL(K)

An element of an array is referred to by the name of the array and a subscript combination. The name of an array is expressed by either one or two alphabetic characters, the first (or only) character of which is any letter except I, J, K, L, M, or N. The subscript combination follows the name of the array and is enclosed in parentheses. Examples are:

D(K, J)

FI(KA, MN)

HL(LN, N)

SN(M, JI)

Examples of incorrect variable expressions are:

B(A)

I(J)

Z(2, 3)

C. Operators

There are seven operators in ATRAN:

+	Add
-	Subtract
*	Multiply
/	Divide
√	Square root
↑	Exponentiation
=	Is replaced by

ATRAN operators are placed between constants or variables to indicate arithmetic operation. For instance, in the ATRAN statement I = I+2, the "+" is the arithmetic operator placed between the variable "I" and the constant "2". It causes 2 to be added to the value of the variable designated by I, and causes this sum to replace the previous value of I. Remember to insert the multiply operator between two operands to be multiplied.

D. Non-Procedural Statements

1. Dimension Statement - DIM:(xx)$ or DIM:(xx, yy)$

The purpose of the dimension statement is to define the size of a list or an array, and to instruct the compiler to reserve the proper number of memory locations for such variables. All DIM statements must be written at the beginning of the ATRAN program, and there must be one DIM statement for each dimensioned variable used in the program. No list may be dimensioned to contain more than 99 items, nor may any array contain more than 99 rows or 99 columns.

Examples:

DIM:A(7)$	7 items in list A
DIM:ZI(14)$	14 items in list ZI
DIM:C(4, 3)$	An array with 4 rows and 3 columns
DIM:XI(9, 3)$	An array with 9 rows and 3 columns

2. Control Point Statement - PNT:xx$

This statement causes the statement immediately following it to be given a label enabling that statement to be referred to by a jump or conditional jump statement. Every statement to which a transfer of control is to be made must be labeled by a PNT statement. Control points may be numbered 1 through 99, but not necessarily in any order or sequence. A PNT statement may not preceed the first executable statement in a program, since the compiler unconditionally labels the first executable statement as PNT zero.

Example: PNT:46$

3. End Statement - END:xx$.

The last statement of every ATRAN program must be an END statement. The number xx indicates that execution of the object program will begin with the ATRAN statement at PNT:xx. If, however, END:0$. or END:00$. is written, execution of the object program begins with the first ATRAN procedural statement, regardless of whether a PNT statement precedes it or not. The $ terminating an END statement must always be followed by a period.

Examples: END:5$.
END:0$.

E. Procedural Statements

1. Arithmetic

ATRAN expressions are any sequence of constants or variables (with or without subscripts) separated by operators, parentheses, and brackets to form a mathematical expression. Parentheses and brackets have the same meaning in ATRAN as in ordinary algebra. ATRAN accepts one level of brackets and one level of parentheses within each bracket. Further nesting of parentheses within a bracket is not permitted. Parentheses may not enclose brackets. Brackets may not enclose brackets; parentheses may not enclose parentheses, with the exception of those parentheses which are used to enclose subscripts. Such parentheses are not in the scope of these rules because they are not used to affect the

arrangement in which an expression is evaluated.

The only item permitted to the left of the equals sign is a variable, such as "I = __", "X = __", or "A(J) = __". Such things as "A↑Z = __", or "Y+5 = __", or "16 = __" are never permitted.

The expression to the right of the equals sign is evaluated from left to right, while observing the rules of algebra concerning the interpretation of parentheses and brackets and the hierarchy of operators. Square root and exponentiation will be performed first; then multiplication and division; and, finally, addition and subtraction. Following are some examples to illustrate this hierarchy:

a. The expression A+B*C is evaluated as A+(B*C), since multiplication takes precedence over addition.

b. The expression A*B↑C is evaluated as A*(B↑C), since exponentiation takes precedence over multiplication.

c. The expression A/B*C is evaluated as (A/B)*C, since multiplication and division are of equal hierarchy and the left to right rule takes precedence.

Exponentiation (↑) is a relational operator and must be preceded and followed by a variable, constant, or expression in parentheses or brackets; if the item to be exponentiated or the power is a complex expression (e. g. A*B or B+C/D), the expression must be enclosed in parentheses or brackets. For example, one would write X↑(A*B) or X↑(B+C/D). If X↑A*B would appear, it would be interpreted as (X↑A)*B.

Square root (√) is a unitary operator and must not be preceded by a variable, constant, or expression in parentheses or brackets; A√B is not permitted; it must be written A*√B. The operand of √, if complex, must be enclosed in parentheses or brackets; thus, √(A+B). If one wrote √A+B it would be interpreted as (√A)+B.

Operations on subscripts should be written like any other arithmetic operations. The only caution to be made is to be sure that the values of all subscripts will always be integers from 1 to 99. Examples of arithmetic statements are:

```
I = 2
A = B(I)+37.6
X = [C+(D(J)+2.5)] / Z(I,JJ)
```

2. Functions

A function expression may contain one and only one function, selected from the list of available ATRAN functions. The function statement is of the form,

```
A = FUN:B
```

where A represents any dimensioned or undimensioned variable, but not a subscript variable, and B represents any variable or constant. The nine functions in ATRAN are:

SIN	Sine
COS	Cosine
ASN	Arc sine
ACS	Arc cosine
ATN	Arc tangent
EXP	Exponential
LOG	Logarithm, base 10
LGN	Logarithm, base e
ABS	Absolute value

Examples of function statements:

```
A = SIN:B$
A = SIN:B(I)$
A(J) = COS:X(K)$
A(I,J) = ATN:1.587
```

Incorrect examples:

```
A = SIN:2*X$
A = SIN:(A+B)$
I = EXP:J$
```

3. Decision and Control Statements

a. Unconditional Jump Statement - JMP:nn$

The JMP statement causes an unconditional transfer to be taken to control point nn, where nn is defined by a PNT statement.

b. Test Statement - TST:xx$

Two procedural statements are required to make a decision. The first is the TST statement. This statement obtains the specified variable xx and saves it for testing by the one or more decision statements that immediately follow. These statements may be any of the three conditional jump or three conditional skip statements. The conditional jump or skip statements must always immediately follow either a TST statement or another jump or skip statement. Some examples of TST statements are:

```
TST:A$
TST:A(I)$
TST:A(I, J)$
TST:I$
```

c. Conditional Jump Statements

The conditional jump statements compare the tested variable to zero and alter the sequence of control if a condition is met. They are:

PJP:nn$ Jump to control point nn if the variable specified by the immediately preceding TST statement is positive. Otherwise, execute the next statement in sequence (nn must be defined by a PNT statement).

ZJP:nn$ Jump to control point nn if the variable specified by the immediately preceding TST statement is zero. Otherwise, execute the next statement in sequence (nn must be defined by a PNT statement).

MJP:nn$ Jump to control point nn if the variable specified by the immediately preceding TST statement is negative. Otherwise, execute the next statement in sequence (nn must be defined by a PNT statement).

Conditional jump statements are illustrated by the following examples:

```
TST:V$
ZJP:20$
PJP:30$
MJP:10$
```

d. Conditional Skip Statements

The conditional skip statements compare the tested variable against an integer from 1 to 99, and alter the sequence of control if a condition is met. They are:

XGR:xx$ Skip the next statement if the variable specified by the immediately preceding TST statement is greater than the integer xx; otherwise execute the next statement in sequence.

XEQ:xx$ Skip the next statement if the variable specified by the immediately preceding TST statement is equal to the integer xx; otherwise execute the next statement in sequence.

XLS:xx$ Skip the next statement in sequence if the variable specified by the immediately preceding TST statement is less than the integer xx; otherwise execute the next statement in sequence.

Conditional skip statements are illustrated by the following examples:

```
TST:AA(I, JB)$        TST:MA$
XGR:50$               XEQ:10$
JMP:1$                JMP:2$
. . . . .             . . . . .
```

4. Input Statements

a. Paper Tape Input

```
LOD:D$
LOD:JZ$
```

The above ATRAN statements will load the variables D and JZ into the computer from paper tape. If a variable is defined by a dimension statement as a list or array, then the entire list or array will be read in, with the size being determined by the dimension statement. If the variable is not dimensioned, then a single variable will be read in.

```
LOD:A(I)$
LOD:B(K, M)$
```

The above statements will allow one element of the list or array to be read into the computer from paper tape. In these examples, the i-th word of A, or the k, m-th word of array B will be loaded with data from the paper tape. A dimension statement for A and B must be at the head of the program.

b. Flexowriter Keyboard Input

```
PUT:D$
PUT:JZ$
```

The above ATRAN statements will load the variables D or JZ into the computer as they are typed on the keyboard. If a variable is defined by a dimension statement as a list or array, then the entire list or array will be read in with the size being determined by the dimension

statement. If the variable is not dimensioned, then a single variable will be read in.

PUT:A(I)$

PUT:B(K, L)$

The above statements will load one element of the list or array into the computer as they are typed on the keyboard. A dimension statement for A or B must, of course, be at the head of the program.

4. Output Statements

Provisions are available in ATRAN to print or punch data in either fixed point or floating point (CINCH format) form. Provisions are also available for printing or punching of literal information.

a. Fixed Point Data Output

Examples:

PRV:(n. mf)K$

PRV:(n. mf)A(I)$

PRV:(n. mf)B(I, J)$

The first example will cause the single variable, K, to be printed in the format indicated in the parentheses. The second example will cause the i-th element of list A to be printed. The third example will cause the i, j-th element of the array B to be printed. The "n" represents the number of decimal digits to be printed to the left of the decimal point; the "m" represents the number of digits to be printed to the right of the decimal point. The "f" represents either a "C", for carriage return, of "T", for tab. The tab or carriage return will be executed after printing the number.

b. Floating Point Data Output

Examples:

PRF:(af)X$

PRF:(af)A(I)$

PRF:(af)B(I, J)$

The first example will cause the single variable, K, to be printed in the floating point format indicated in the parentheses. The second example will cause the i-th element of list A to be printed. The third example will cause the i, j-th element of the array B to be printed. The "a" may be any digit 0 through 9. If a = 0, the mantissa of the floating point variable will be printed to ten places and rounded; otherwise, the mantissa will be truncated and printed to "a" places.

c. Printing Literals

Example:

PRA:PB 250$

This statement will cause the literal message "PB 250" to be printed on the Flexowriter. When using literal statements, any characters except semi-colon, colon, and dollar sign may be used within the text of the literal.

6. Punched Output

A full complement of "Punch Output Statements" is provided. These statements operate identically to the Print Output Statements described above, by changing a print statement, PR__: to a punch statement, PN__:. That is,

PRV becomes PNV
PRF becomes PNF
PRA becomes PNA

7. HALT Statement

HLT:LITERAL$

This statement causes the computer to halt. Just before stopping, the message "LITERAL" will be typed. Restarting, if desired, causes the next sequential statement to be executed.

IV. RESTRICTIONS

Due to a finite machine size, there are several restrictions that must be placed upon the size of ATRAN programs. A list of the more important ones follows:

A. Limit on Characters in a Statement

The number of characters in one ATRAN statement must not exceed 100. (This does not include tape feed, delete code, space, carriage return, or tab.)

B. Limit on Statements in the Program

The total number of statements in an ATRAN program must not exceed 254.

C. Limit on Variables

The combined total of undimensioned variables, subscript variables, and one-dimensioned lists in an ATRAN program must not exceed 175.

D. Limit on Constants

The combined total of constants and two-dimensioned lists in an ATRAN program must not exceed 127.

E. Limit on Literals

The total number of characters contained in all literal messages appearing in an ATRAN program must not exceed 240. Nor must the number of messages exceed 64.

F. Limit on Control Points

The combined total of PNT statements and subscript variables which appear on the left side of an equals sign must not exceed 125.

G. Limit on Subscripts

The number of subscripts and distinct subscript combinations appear-

ing in an ATRAN program must not exceed 31.

The number of subscripts appearing between PNT statements also must not exceed 31. If more than 31 subscripts are required between two PNT statements, the programmer may insert a "dummy PNT statement" thereby satisfying this restriction.

H. Limit on Program Size

The total CINCH memory required by the compiled program must not exceed 923 PB250 memory locations.

I. Limit on Type 1 Statements

The combined total of CF1 type statements (defined to be all jumps and literal output statements) must not exceed 63.

J. Limit on Type 2 Statements

The combined total of CINCH instructions generated by CF2 type statements (defined to be all statements that are neither CF1 type nor arithmetic) must not exceed 125.

K. Limit on Arithmetic Statements

The number of CINCH instructions generated by arithmetic statements must not exceed 512.

V. SUMMARY OF ATRAN STATEMENTS

Constants

Use up to five digits to left and to right of the decimal. Write the sign to the left; the plus sign is optional. Decimal point in integer is also optional.

Variables

Use one or two alphabetic characters to denote variables; do not use I, J, K, L, M, or N as the first or only character. Variables may have one or two subscripts, in parentheses. Use one or two alpha characters for subscripts; use only I, J, K, L, M, or N as the first or only character. Subscripts are variables themselves, may be operated on by arithmetic statements, but must be integers from 1 to 99.

Operators

+, -, *, /, $\sqrt{}$, $\Uparrow$(exponentiation), =

Non-Procedural Statements

DIM:(x)$ or DIM:(x, x)$, where $1 \leq x \leq 99$

PNT:___$ Establishes a control point.

END:___$. Ends program.

Procedural Statements

Arithmetic

I = I+1$

X = (any algebraic expression) No functions are allowed.

Functions

SIN, COS, ASN, ACS, ATN, EXP, LOG, LGN, ABS

Decisions

TST:xx$	Follow logically by any of the following.	
PJP:nn$	Positive jump to nn.	
ZJP:nn$	Zero jump to nn.	
MJP:nn$	Minus jump to nn.	
XGR:yy$	Skip if xx > yy	$1 \leq yy \leq 99$
XEQ:yy$	Skip if xx = yy	
XLS:yy$	Skip if xx < yy	

Input

Paper Tape

LOD:x$	x may have a subscript. Without a subscript, it will load entire list in row-wise order, if x is mentioned in DIM: statement.

Keyboard

PUT:x$	Same logic as LOD:, except data is typed on-line.

Typed Output

Fixed Point - PRV: (n. mf)v$

n = number of places left of decimal

m = number of places right of decimal

f = T for Tab or C for Carriage Return

v is any variable, and may have a subscript

Floating Point - PRF:(af)v$

a = number of places in mantissa; $0 \leq a \leq 9$. If a = 0, the mantissa will have 10 places, rounded. If $a \neq 0$, the mantissa will have "a" places, truncated.

Literals - PRA:LITERAL$

:, ;, $ may not be in text of literal

Punched Output

The same rules apply as for Typed Output, by changing PRV to PNV, PRF to PNF, and PRA to PNA.

Halt

HLT:LITERAL$ Restarting executes next statement.

VI. SAMPLE PROBLEMS

We present three sample programs to illustrate the use of ATRAN. The first example is a program for matrix multiplication. Find the product of two matrices, A and B, given that:

1. Matrix A consists of 5 rows and 6 columns.
2. Matrix B consists of 6 rows and 8 columns.

Read in the elements of A and B from paper tape, and print the elements of the product matrix C = AB, where:

$$c_{ij} = \sum_{K=1}^{6} a_{iK} b_{Kj} \qquad (i = 1, \ldots, 6;\ j = 1, \ldots, 8)$$

The ATRAN program follows on the next page.

```
ATRAN SAMPLE PROBLEM; MATRIX MULTIPLY$$
DIM:A(5, 6)$ DIM:B(6, 8)$ DIM:C(5, 8)$
LOD:A$
LOD:B$
I = 0$
PNT:05$  J = 0$
I = I+1$
PNT:06$  K = 0$
J = J+1$
C(I, J) = 0.0$
PNT:07$  K = K+1$
C(I, J) = C(I, J)+A(I, K)*B(K, J)$
TST:K$  XEQ:06$
JMP:07$
TST:J$  XEQ:08$
JMP:06$
TST:I$  XEQ:05$
JMP:05$
I = 0$
PNT:08$  J = 0$
I = I+1$
PNT:09$  J = J+1$
PRV:(3.2T)C(I, J)$
TST:J$  XEQ:07$
JMP:09$
J = J+1$
PRV:(3.2C)C(I, J)$
TST:I$  XEQ:05$
JMP:08$
HLT:DONE$  END:0$.
```

The second example is that of calculating data for plotting the curve of the safe loading of a column as a fraction of slimness ratio. The safe load (S) is given by two formulas, each of which holds over a different range of the slimness ratio (R).

$$S = \begin{cases} 17{,}000 - .435\,R^2 & R \leq 120 \\ \dfrac{18{,}000}{1 + \dfrac{R^2}{18{,}000}} & R \geq 120 \end{cases}$$

Calculate S from R = 20 to R = 200, where R varies in steps of 5, and print out the results.

The ATRAN program follows on the next page.

```
ATRAN TEST CASE
COLUMN DESIGN$$
R = 20$
PRA:
$
PNT:10$
TA = R-120$
TST:TA$
MJP:20$
JMP:30$
PNT:20$
S = 17000-.485*R↑2$
JMP:40$
PNT:30$
S = 18000/(1+R↑2/18000)$
PNT:40$
PRA:R = $
PRF:(8T)R$
PRA:S = $
PRF:(8C)S$
TA = R-200$
TST:TA$
MJP:50$
JMP:60$
PNT:50$
R = R+5$
JMP:10$
PNT:60$
HLT:01$
END:0$.
```

The last example is that of calculating the roots of a set of quadratic equations. Read in the coefficients of four quadratics of the form $a_i x^2 + b_i x + c_i = 0$, and store them in an array

$$\begin{array}{ccc} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \\ a_4 & b_4 & c_4 \end{array}$$

Then calculate the roots

$$X_i = \frac{-b_i \pm \sqrt{b_i^2 - 4 a_i c_i}}{2 a_i}$$

If the roots are imaginary, print a message to that effect.

The ATRAN program follows on the next page.

```
QUADRATIC$$
DIM:GG(4. 3)$
LOD:GG$
I = 1$
PNT:1$
JK = 1$
JL = 2$
JM = 3$
Y = GG(I, JL)↑2-4*GG(I, JK)*GG(I, JM)$
TST:Y$
MJP:3$
XA = (-GG(I, JL)+Y↑. 5)/(2*GG(I, JK))$
PRA:
X1 = $
PRF:(8C)XA$
XB = (-GG(I, JL)-Y↑. 5)/(2*GG(I, JK))$
PRA:X2 = $
PRF:(8C)XB$
PNT:4$
TST:I$
XEQ:4$
JMP:2$
HLT:END$
PNT:2$
I = I+1$
JMP:1$
PNT:3$
PRA:
THE ROOTS ARE IMAGINARY
$
JMP:4$ END:0$.
```

VII. OPERATING INSTRUCTIONS FOR ATRAN

There are two versions of ATRAN, one of which is designed to run on a PB-250 that is equipped with a high speed photoreader, and one which is designed to run on a PB-250 that has no photoreader attached. In the latter case, the Flexowriter is the sole mode of input. We shall refer to the two versions as PHOTOREADER ATRAN and FLEXOWRITER ATRAN, respectively. These two versions differ only in the manner in which the sections of the program tape are read into the computer; they are identical in other respects.

A. Operation of FLEXOWRITER ATRAN

1. Load the Octal Utility Package into memory (see Appendix C of the PB-250 Programming Manual[1] for detailed instructions).

2. Mount Section 1 of ATRAN in the Flexowriter. When the tape is mounted, type an F. (This action initiates the reading of Section 1.)

3. The message "TYPE P OR K" will be printed and the keyboard light will come on.

a. If the source program is on tape, mount that tape in the Flexowriter and type P.

b. If the source program is to be typed directly, type K and when the keyboard light comes on, proceed to type the program.

When Section 1 is complete, the keyboard light will come on again. If there have been any diagnostic error messages at this time, do not continue. (You will have to correct the errors, re-punch the source program tape, and start again.) However, if there are no diagnostics, proceed to step 4.

4. Mount Section 2 of ATRAN in the Flexowriter. When the tape is mounted, type an F. (This action initiates the reading of Section 2.)

5. After the entire ATRAN tape has been read and processing has been completed, the computer will halt in line 5, displaying a line address of 25.

6. Load CINCH into memory (see the PB250 CINCH Interpreter Manual[2] for detailed instructions). CINCH is then used in the standard manner to print, punch, or execute the program compiled by ATRAN.

7. Error Procedure

A halt in line 1, displaying a line address of 37, indicates an error in reading a portion of the ATRAN tape. If this condition occurs, backspacethe ATRAN tape one binary line, depress the Enable and Breakpoint switches, type I, and then raise the two switches. When the Flexowriter light comes on, type an F to re-read the offending line.

The beginning of a binary line can be recognized by a blank space on the tape, followed by a two-digit line number, a $ and a G, in that order.

B. Operation of PHOTOREADER ATRAN

1. Load the ATRAN HSR II tape into memory using the standard bootstrap procedure. (See Appendix C of the PB250 Programming Manual for detailed instructions.)

2. When the keyboard light comes on, mount the ATRAN tape in the photoreader. When the tape is mounted, type an F. (This action initiates the reading of ATRAN.)

3. The message "TYPE P OR K" will be printed and the keyboard light will come on.

a. If the source program is on tape, mount that tape in the Flexowriter and type P.

b. If the source program is to be typed directly, type K and when the keyboard light comes on, proceed to type the program.

After the source program has been read and processed, the photoreader will be activated to read the next section of the ATRAN tape. If there have been any diagnostic error messages printed, stop the computer manually by depressing the Enable switch. (You will have to correct the errors, re-punch the source program tape, and start again.) However, if there are no diagnostics, do not disturb the operation of the computer.

4. After the entire ATRAN tape has been read and processing has been completed, the computer will halt in line 5, displaying a line address of 25.

5. Load CINCH into memory (see Appendix 3, Instructions for Loading ATRAN CINCH, page 40). CINCH is then used in the standard manner to print, punch, or execute the program compiled by ATRAN.

6. Error Procedures

a. A halt in line 1, displaying a line address of 37, indicates an error in reading a portion of the ATRAN tape, before the input of the source program. If this condition occurs, turn off the photoreader, backspace the ATRAN tape one binary line, depress the Enable and Breakpoint switches, type I, and then raise the two switches. When the Flexowriter light comes on, turn on the photoreader and type an F to re-load the offending line.

b. A halt in line 3, displaying a line address of 37, indicates an error in reading a portion of the ATRAN tape, after the input of the source program. If this condition occurs, turn off the photoreader, backspace the ATRAN tape one binary line, and depress the Enable and Breakpoint switches. While the Enable and Breakpoint switches are still down, turn on the photoreader. Upon raising the two switches, the tape will start to read again.

c. If, for any reason, you should wish to re-start from the beginning of ATRAN, rewind the ATRAN tape, mount it in the photoreader,

depress the Enable and Breakpoint switches, type I, and then raise the two switches. When the Flexowriter light comes on, type an F as you would in step 3 of the normal operating procedure.

The beginning of a binary line can be recognized by a blank space on the tape, followed by a two digit line number, a $ and a G, in that order.

C. Source Program Input

ATRAN initially types out:

TYPE P OR K

If the ATRAN source program is to be input on paper tape, first mount the input tape and then type a lower-case P; Phase I will then be in complete control. If the ATRAN source program is to be keyed in, type a lower-case K; when the Flexowriter light goes on, begin typing in the program. As Phase I does not process input at maximum Flexowriter speed, it is essential to be sure the light is on before typing each character.

D. Character Interpretation

In the ATRAN source program, upper and lower case are significant according to the following rule: unless case is set when Phase I is not accepting input, a character will be properly interpreted if and only if it prints on the carriage paper (or would print, in the case of paper tape) as the character it is intended to be. Thus, alphabetic characters, comma (,), period (.), and minus (-) may be in either upper or lower case. (Since upper-case minus is really an underscoring, it is a minor exception.) Numerals and all punctuation (except as noted above) must be in the proper case. Delete and tape feed are always ignored. Tab, carriage return, stop, and space are ignored, except in the text of a literal. Upper and lower case do not increment the character count of a statement, except in the text of a literal. Colon (:) and semi-colon (;) are forbidden in the text of a literal, because of the way CINCH handles alpha-numeric text, but will not cause an error message in Phase I.

Comments may be included in an ATRAN program if followed by $$. It is essential to wait for the light to go on before typing the next statement. Note also that some character must be typed (or punched on paper tape) after the $ of an END statement; a period would be satisfactory for this purpose, but a carriage return, tab, etc. would be by-passed.

If an error is noticed in a statement as it is being typed, the statement may be deleted by typing two dollar signs ($$). No provision has been made for deleting a statement once it has been processed. Particularly when input is on paper tape, however, it may be best to let Phase I process the entire source program, regardless of the number of error messages. Virtually all formal errors will be detected in Phase I.

E. Error Messages

All error messages are of the form,

C/R ERROR XX, STATEMENT XXX

where XX and XXX are numbers. Statement numbers are assigned by Phase I to each statement sequentially, starting with 001; these numbers have no relation to reference points assigned by PNT statements. Statements deleted by $$ are not assigned numbers. Error numbers have the meanings shown on page 34.

NOTE: One restriction (that of more than 31 subscripts between PNT statements, see Page 19) is not detected in Phase I. However, it will result in a HLT operating out of line 12 and displaying line 22 in Phase III.

01 Statement exceeds 100 characters.

02 Illegal operation.

03 CF1 excess.

04 CF2 excess.

05 Other than floating variable in DIM statement.

06 Other format error in DIM statement.

07 In DIM statements: two lists have the same name, or two arrays have the same name.

08 Illegal operand in XLS, XEQ, XGR.

09 Table A excess.

10 Table B or Table E excess.

11 Illegal subscripting format.

12 (Not used)

13 Other than variable in context requiring variable.

14 Subscripted variable, no DIM statement.

15 Table D excess.

16 Illegal reference point designation.

17 More than 5 integer digits or more than 5 fractional digits in floating constant.

18 Format error in PRF, PNF, PRV, PNV.

19 Illegality in algebra not covered by other error numbers.

20 Algebra excess.

21 $ not at apparent end of statement.

22 Table C excess.

23 LITTEXT excess

In general, Phase I will process a statement up to the first formal error. Excess data for Tables A, B, D, and E will inhibit entry of data in the full table but will not terminate the statement scan. Error 14 will not terminate the statement scan. All other errors will terminate the scan at the point of occurrence. Dummy entries will be made in CF1 or CF2 for erroneous CF1 or CF2 statements, unless the appropriate table is full. In case of premature termination of the scan, error message 21 will <u>not</u> be

printed out.

At times, error messages may be difficult to interpret. For example,

XLS:102$

is illegal because the operand is more than two digits. Phase I, however, will process the statement as

XLS:10$

and print out error message 21.

The statement

XLS:A$

is illegal because the operand is alphabetic. In this case Phase I will print out error message 08 but not error message 21. In the first case, the erroneous nature of the statement is detected after an apparently legal operand has been scanned. In the second case, the actual error is detected immediately after the colon.

In the case of a PNT statement preceeding the first executable statement error message, 02 is printed.

APPENDIX 1

A BRIEF DESCRIPTION OF THE ATRAN COMPILER

The following discussion is intended to give the user a little more insight into ATRAN. Basically, the ATRAN compiler is divided into four phases, each of which performs a separate task in the compiling process.

Phase I reads the source program into the computer, decodes it, makes appropriate table entries for each variable, constant, literal, control point, and subscript combination that it encounters, and expresses each statement in one of several compact canonical forms. The END statement of the source program indicates to Phase I that all statements of the program have been entered.

Phase II of ATRAN reduces the previously stored canonical forms to one form, called PRECINCH. The canonical forms were characterized by the fact that there was one entry per ATRAN statement; the PRECINCH form has one entry corresponding to each CINCH instruction that is to be developed. Section A of Phase II transforms all ATRAN statements, with the exception of arithmetic statements. Section B of Phase II transforms arithmetic statements to the PRECINCH form.

Phase III of ATRAN scans the subscript appearances in the PRECINCH form and assigns CINCH index registers. It generates the necessary coding required to properly load and restore these index registers.

Phase IV of ATRAN transforms the PRECINCH entries into the final CINCH instructions. It allocates memory, not only for the CINCH object program, but also for constants, data, and variables, and converts the constants into the CINCH floating point binary format. When Phase IV has been completed, the object program is in memory, ready to be executed by the CINCH interpreter.

At this point, the CINCH tape is read into the computer, without

disturbing the object program. Then the CINCH program can be used to read the data and execute the object program. The CINCH program can also be used to list the object program and punch it out for later use.

A brief outline of the arithmetic scan may prove to clear up any questions the user might have about the left to right and hierarchy rules in ATRAN.

A left to right scan is made to locate any parentheses that appear in a statement, excluding parentheses enclosing subscripts. Arithmetic operations within the left-most (or only) pair of parentheses are evaluated first. The contents of this pair of parentheses are scanned from left to right for square root or exponentiation operators. When either operator is encountered, ATRAN compiles a set of instructions to evaluate the operation indicated. Having compiled instructions necessary for the evaluation of these two types of operators, a left to right scan of the contents of the parentheses is made for multiply and divide operators. Again, instructions are compiled to evaluate the results of these operations. Finally, another left to right scan is made for addition and subtraction operations and instructions are compiled to evaluate them.

After evaluating the contents of the left-most parentheses, a left to right scan is made for another pair of parentheses. Its contents are processed in the manner described above, and the scan continues until all parentheses are exhausted. Then a similar left to right scan is made for brackets and the contents of the brackets are processed in the same manner. Finally, after all parentheses and brackets have been removed, a similar left to right scan is made of the entire statement.

APPENDIX 2

MODIFICATIONS TO CINCH

The version of CINCH to be used with ATRAN is a modification of CINCH II, which shall be referred to as ATRAN CINCH. The procedures for using ATRAN CINCH are identical in every manner to those for CINCH II. Therefore, the user can refer to the CINCH Manual for any question about using ATRAN CINCH.

The only difference between the two versions of CINCH is that ATRAN CINCH has two extra commands, FIX (octal CINCH code 61) and FLO (octal CINCH code 62). These two instructions are used to signal the output routines of CINCH to enable the handling of fixed point output. A description follows:

FIX M — Fixed Output Mode — (61)

Set CINCH to the fixed point output mode. All succeeding output commands will allow no more than M digits before the decimal point. CINCH will remain in this mode until reset by a FLO command or another FIX command.

FLO — Floating Output Mode — (62)

Set CINCH to the floating point output mode. All succeeding output commands will function exactly as in CINCH II. CINCH will remain in this mode until reset by a FIX command.

In addition, the SFL command takes on a different meaning in the fixed point output mode. In the floating point output mode, it is unchanged.

SFL M — Set Fraction Length — (35)

If CINCH is in the floating point output mode, the SFL command will function exactly as in CINCH II.

If CINCH is in the fixed point output mode, the following output com-

mands will assume that M ($1 \leq M \leq 10$) is the total number of digits to be typed or punched, both before and after the decimal point. M is permanently fixed until changed by another SFL command.

For example, if we give the sequence of commands,

FIX 5

SFL 8

all succeeding output will be punched or typed with 5 digits to the left of the decimal point and 3 digits to the right.

APPENDIX 3

INSTRUCTIONS FOR LOADING ATRAN CINCH

For loading ATRAN CINCH after compilation by FLEXOWRITER ATRAN, the standard procedure as detailed by the CINCH Interpreter Manual is to be used. However, if PHOTOREADER ATRAN has been used, the following procedure is recommended for more expedient loading of ATRAN CINCH.

The ATRAN CINCH tape is divided into two sections: the larger one contains lines 2-13 of CINCH and the smaller one contains line 1, preceded by a bootstrap.

1. Mount the larger section of ATRAN CINCH in the photoreader. Depress the Enable and Breakpoint switches, type I, and raise the two switches. When the Flexowriter light comes on, type an F (this will initiate the reading of the tape).

2. After the tape has been read successfully, the computer will halt in line 1, displaying a line address of 26. If there has been an error in reading, the halt will again be in line 1, but the line address will be 37 (see Error Procedure, below).

3. Mount the smaller section of ATRAN CINCH in the Flexowriter. With the Enable and Breakpoint switches down, turn the FILL switch on the Computer Control Panel to the ON position. After the tape starts moving, be sure to raise at least one of the Enable and Breakpoint switches.

4. When tape movement stops, turn the FILL switch to the OFF position. Depress Enable and Breakpoint, type I, and then raise these switches. After the remainder of the tape has been read, the Flexowriter light will come on. At this point you will be under external control of CINCH.

Error Procedure

A halt in line 1, displaying a line address of 37, indicates an error in

reading a portion of the ATRAN CINCH tape through the photoreader. If this condition occurs, turn off the photoreader, backspace the tape one binary line, depress the Enable and Breakpoint switches, type I, and then raise the two switches. When the Flexowriter light comes on, turn on the photoreader and type an F to re-load the offending line.

APPENDIX 4

EXECUTION OF ATRAN PROGRAMS WITH CINCH

In running an ATRAN program with CINCH, the user must know the location in CINCH memory at which the ATRAN program begins. This location will always be CINCH location 65. Thus, to start execution, the user should type:

```
$ 0065
G
```

If the user wishes to use CINCH to punch out a binary program tape, it is suggested that he punch out all of CINCH memory, from location 0065 to location 1023. If, however, the user knows that his program is small, he can use a CINCH memory print out to determine where his commands and data are located, if he is familiar with CINCH.

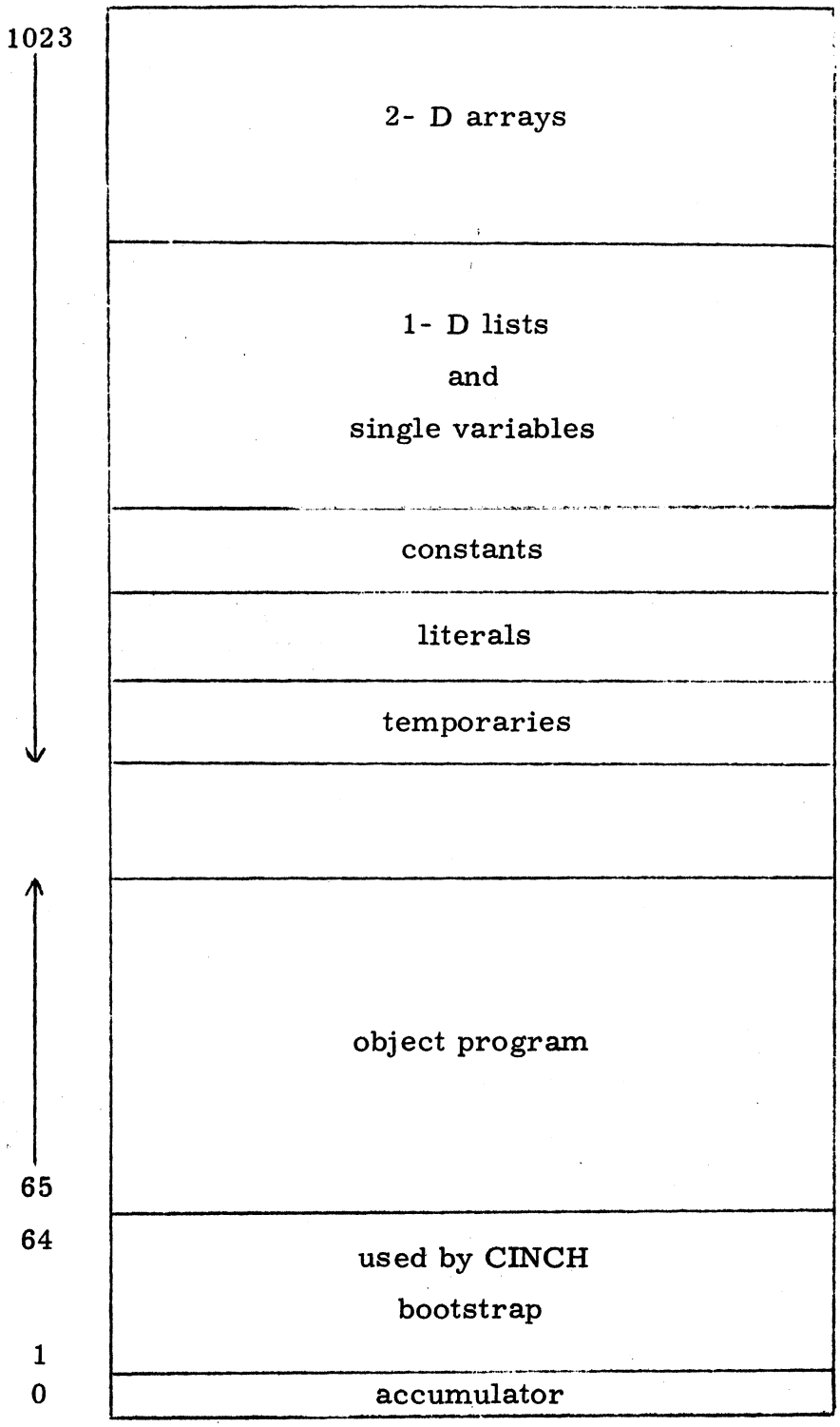
The commands begin at location 0065 and proceed sequentially upwards in CINCH memory. The last two commands in the program will always be:

```
46 XXXX
00 0000
```

no matter which ATRAN statements are in the user's program. These commands can easily be recognized because of the fact that no other commands follow. The address XXXX (of the 46 command) is the lowest address of the literal messages. The constants immediately follow the literals and the end of the constant area can be recognized because the last five constants will always be 2, , /2, 1, and 0.

The commands, literals, and constants are the only portions of CINCH memory that need be punched out. The contents of the remainder of memory, if not zero, are either read from a data tape or computed by the program at execution time.

A simplified diagram of how CINCH memory is laid out follows.


1023
2- D arrays
1- D lists
and
single variables
constants
literals
temporaries
object program
65
64
used by CINCH
bootstrap
1
0
accumulator

REFERENCES

1. PB-250 Programming Manual, PBC1004, Revision 1
   Packard Bell Computer Corporation, March 15, 1961

2. PB-250 CINCH Interpreter, PBC1006
   Packard Bell Computer Corporation, March 15, 1961

3. A Guide to FORTRAN Programming
   Daniel D. McCracken, Wiley, 1961