

Assume that 100 ops. will not exceed 240
sep. wd where each pseudo op, each symbol etc
takes one word. Math-
Matic

If user is pos. - get ATL (Trf. on >)
neg. - ? ALL (Trf. on <)
Never get ALL?
If user is var. - get ATL - we think

Jean
Sammet
File AT-3

Preliminary Manual
for
MATH-MATIC and ARITH-MATIC
Systems
for
ALGEBRAIC TRANSLATION and COMPILATION
for
UNIVAC I and II

Prepared by:

- R. Ash
- E. Broadwin
- V. Della Valle
- M. Greene
- A. Jenny
- C. Katz
- L. Yu

Automatic Programming Development
REMINGTON RAND UNIVAC
Philadelphia, Penna.
19 April, 1957

TABLE OF CONTENTS

<u>Section</u>	<u>Page No.</u>
1. The Math-Matic Programming System (Introduction)	1
2. General Rules for Writing Sentences	3
3. Functional Call Words Used in Math-Matic Equations (List)	10
4. Repertoire of Math-Matic Control Statements (List)	11
5. Functional Call Words (Descriptive)	13
6. Control Statements (Descriptive)	17
7. Compiler and Computer Sections	33
8. Example of Compiler Section and Directory	36
9. Example of Computer Section and Directory	37
10. The Math-Matic System (Descriptive) <i>Phases + Sweeps</i>	38
11. How to Increase the Repertoire of Math-Matic Call Words and Control Statements	43
12. Subroutines for All Glossaries	47
13. Subroutines for Equation Glossaries	51
14. Subroutines for Statement Glossaries	60
15. Constant Pool for Glossary Use	70
16. Examples of a Glossary with Sentence File 1 and 2	72
17. Operating Instructions for Math-Matic Librarian	75
18. How to Write Arith-Matic Pseudo-code	80
19. Repertoire of Arith-Matic Pseudo-code Operations	82
20. Use of Univac C-10 Code with the Arith-Matic Compiler	92
21. List of Arith-Matic Constants	94
22. Rules for Subroutine Construction	95
23. Operating Instructions for Math-Matic	97
24. Sample Problem 1	99
25. Sample Problem 2	102
26. Math-Matic Print-outs	104
27. Arith-Matic Print-outs	117
28. Problem Run Print-outs	120

THE MATH-MATIC PROGRAMMING SYSTEM

Before the development of automatic programming techniques, it was necessary that the user of high-speed electronic digital computers either require his technical staff to learn the intricacies of machine coding and debugging, or that he employ qualified programmers, who then had to learn the technicalities of his business or profession. In either event, a considerable expenditure of time and money occurred before a given program could "get on the air".

It soon became apparent that nothing was better suited to performing accurately and tirelessly the repetitive and routine clerical aspects involved in coding than the computer itself. From this basic concept have come many automatic programming routines (assembly systems, interpretive routines, and compilers).

In addition to the basic translation into a machine-coded program, executable by the computer, of a problem expressed in a pseudo-code easily learned by personnel unfamiliar with the computer, compilers make it possible to re-use subroutines, previously coded, which may be applicable to many different parts of the same or different problems. Such pieces of coding are stored on a library tape, from which the master compiler routine selects them, as needed, and integrates them into the final running tape, with suitable modifications.

Math-matic (AT-3) is an automatic coding system designed for mathematicians, engineers, and all others who may have problems expressible in mathematical symbology and English sentences. The system makes it possible for the user to program directly for the machine without any necessity for briefing a coder who may not be technically qualified to understand the

problem. Standard mathematical notation, combined with control statements expressed in English, form the input language of the system. Thus, it is only necessary for the user to analyze his problem logically and to express this analysis directly in Math-Matic pseudo-code.

The already significant reduction in programming costs of scientific problems through the use of the Arithmetic (A-3) Compiler will be further reduced through the use of Math-matic. The elapsed time between the definition of a problem and its production running also will be reduced. A problem which normally would require six weeks of programmer time to code and debug (including hours of computer time) will require less than a day of programmer time (including one hour of computer time), a contrast of better than 30/1 man-days.

To sum up, Math-matic pseudo-code (the combined equations and control statements) describes the problem, from the user's standpoint, rather than the program required by the hardware of the computer. Conversion of the problem, expressed in pseudo-code, into the necessary program, in machine code, is performed entirely automatically and internally, greatly reducing programming costs.

GENERAL RULES FOR WRITING SENTENCES

I. NOTATION

- A. Sentence - any command which can be interpreted by AT-3. AT-3 sentences are of two types:
1. Equation - combination of operands and symbols of operations having a mathematical meaning.
 2. Control Statement - English sentence performing a specifically stated control function for the problem.
- B. Character - any of the letters of the alphabet, cardinal numbers, punctuation marks, or other typewriter symbols recognized by the compiler.
1. Alphabetic character - any of the letters of the alphabet A through Z.
 2. Numeric character - any of the digits zero through nine.
 3. Exponential character - any of the superscript digits zero through nine, plus superscript decimal point, division slash, and minus sign.
- C. Symbol - any combination of twelve or less characters.
- D. Operations - are designated as "unary" if they operate on a single operand; "binary" if they operate on two operands.

Unary

<u>Symbol</u>	<u>Operation</u>	<u>Example</u>
+ (optional)	plus	+47.28
-	minus	-246.92
∇	absolute value	-(A + B) A + B

Can write $(a+b)/c + d$

<u>Binary</u>	<u>Symbol</u>	<u>Operation</u>	<u>Example</u>
	+	addition	$4 + y$
	-	subtraction	$X - 372.18$
	*	multiplication	$A * B * 7$
	/	division	$X/3$
	Δ	functional operation	$\cos \Delta A$
	Superscript symbol	exponentiation	$x^{-1/2}$

E. Argument - see operand.

F. Operand - any "symbol" representing a quantity which is to become the subject of a mathematical or logical operation.

G. Functional Operation - any standard call-word which refers to a subroutine in the Math-matic (AT-3) library. This library contains frequently used routines (trigonometric, hyperbolic, exponential, and logarithmic functions).

H. Relations -

= is equal to $W = X + Y$

> is greater than $X > Y$

< is less than $Y < X$

I. Constant - a "symbol" consisting of a series of decimal digits (numeric characters) which may include a decimal point and which may be preceded by a plus or a minus sign.

1. If a pure decimal constant is used, it must not be typed as a space and a period ($\Delta.$), since this combination indicates the end of a statement. Therefore, decimal numbers must be typed as zero point ($0.$) if there is no integral part.

3.

2. Since the system uses the two-word floating decimal mode of computation, all constants will be converted to two word floating decimal form with eleven digit accuracy. This also pertains to constants written in power of ten notations.

$$300 = 030000000000 \\ 000000000003$$

3. As noted above, constants may be written in power of ten form

$$123.95 \\ 0.12395 * 10^3 \\ 1.2395 * 10^2 \\ 12395 * 10^{-2}$$

Each of the above will be converted to exactly the same two-word floating decimal form:

- J. Variable - any "symbol" of alphabetic or numeric characters, at most 12 digits in length. The first digit must be an alphabetic character in order to distinguish it from a constant X, ABC, AD1776.
- K. Expression - a group of one or more variables and/or constants which has a mathematical meaning. Every adjacent pair of variables and/or constants must be separated by an operation symbol.

$$X * \text{SIN}A$$

$$\text{BAPOWA}2-4 * A * C$$

- L. Grouping - parentheses are used to group variables and operations in order to sequence the computation. Any number of parentheses may be used, provided they are paired. Whenever there is a question of ambiguity, always use parentheses.

$$A/B+C \quad \text{means} \quad C + \frac{A}{B}$$

$$A/(B+C) \quad \text{means} \quad \frac{A}{B+C}$$

M. Subscripts of a Variable - a sequence of symbols or expressions enclosed within parentheses which follow the symbol used for the variable $X(j)$

1. No operation symbol is placed between the variable and the left parentheses.
2. Multiple subscripts within the same pair of parentheses are separated by commas.
3. Signs of operation may be used in subscripts for notational purposes. They will not be interpreted.
4. Subscripts of subscripts are indicated by parentheses within parentheses.
5. A variable and its subscripts may not exceed 12 digits.

$X(i,j)$ means $X_{i,j}$

$X(i(p),j(q))$ means X_{i_p, j_q}

$Y(i+5, j+p)$ means $Y_{i+5, j+p}$

II. GENERAL FORMAT

- A. Every sentence must begin with a sentence number.
 1. The sentence number must be enclosed in parentheses followed by a space.
 2. The sentence numbers must be in ascending order--not necessarily sequential.
 3. The sentence number must consist of one or two numeric digits (0-99) and a possible appended alphabetic--to aid insertion of new sentences in previously defined problems.

NOT TRUE
9-19-57

- B. A maximum of one hundred sentences are allowed in a problem.
- C. Each sentence must end with a space followed by a period.
- D. Any number of spaces may appear between the end of the sentence and the period, and between the period and the parentheses of the next sentence. *what is "end of sentence" last valid entry*
- E. Any pure decimal number (.35621) must be preceded by a zero (0.35621). Otherwise a space-period sequence (Δ .35621) would occur, signifying the end of a sentence.

III. RULES FOR EQUATIONS

- A. Equations must be explicit; only a single variable (of as many as twelve characters) may appear as the left member of an equation.
- B. Any variable appearing in the right member of an equation, must have been previously defined as the left member of an equation or have appeared in an input statement.
- C. Each equation may contain up to 100 signs of operation (+, -, *, /, (), Δ) and Exponentiation.
- D. Math-matic analyzes and processes three classes of operation symbols in order of priority.
1. Functional call-words and exponents.
 2. Multiplication and division (* and /)
 3. Addition and subtraction (+ and -)
- Ambiguities may arise in the grouping of arguments in expressions. Such ambiguities must be removed by using parentheses to enclose the desired expressions.
- E. Exponents other than literal should be shown in the normal superscript manner. If the exponent is literal, it must be expressed as $\Delta\Delta\Delta\Delta\Delta\Delta$.

- F. Any argument involved with a functional call-word which appears as an expression other than a single argument, must be enclosed in parentheses. C^D is written $C\Delta P O W \Delta D$.
- G. Only literal quantities may be enclosed in absolute signs.
- H. Use of Spaces:
1. Use a space after the parentheses which encloses the sentence number.
 2. Every sentence ends with a space and a period.
 3. A space must be inserted between any functional call-word and its argument, and multiple argument call-words, must be preceded and followed by a space, - separating the call word from both arguments.
 4. Spaces must be inserted before and after the equal sign, and before and after "greater than" ($>$) or "less than" ($<$) signs.
- I. The user is responsible for seeing that the variables associated with functional call-words do not take on values which will lead, in computation, to infinite or undefined results (e.g., $\log A$, $A \leq 0$). Such values will lead to error print-outs during the problem run, which are discussed in the Arithmetic section of the manual.

IV. RULES FOR STATEMENTS

A list of available statements will follow:

1. Spaces or commas may be used to delete arguments.
2. A space must be placed after the parentheses which enclose sentence numbers.
3. End each sentence with a space and a period.
4. Greater than, (>), less than, (<), and equal (=), signs must be preceded and followed by a space. These relations are not confused with those in an equation sentence.
5. A space must be placed after every word.

EXAMPLE: (n)ΔVARYΔAAΔ.

(n)ΔJUMPΔTOΔSENTENCEΔFAΔ.

6. When using one of the statements in the available list, the user must adhere exactly to the format given. Spaces and commas may be interchanged.
 - n - refers to the number of the sentence.
 - F - refers to the first sentence in the range.
 - L - refers to the last sentence in the range.
7. If the word "REWRITE" is typed out in the supervisory control during the Math-matic run, it signifies that an error has been made in the pseudo-code or data which cannot be corrected with a type-in. Hit the start bar and all tapes will rewind.
8. If a type-out offers the operator the option of typing in and he does not wish to exercise this option, then the tapes must be rewound manually.

FUNCTIONAL CALL WORDS USED IN MATHEMATIC EQUATIONS

1. TRIGONOMETRIC FUNCTIONS:

SIN Δ	COT Δ
COS Δ	SEC Δ
TAN Δ	CSC Δ

2. HYPERBOLIC FUNCTIONS:

SINH Δ
COSH Δ
TANH Δ

3. GENERAL:

A Δ POW Δ B
N Δ ROCT Δ A
SQRA Δ
EXPA Δ
LOG Δ A
LN Δ A

4. INVERSE FUNCTIONS:

ARCTAN Δ A

REPertoire OF MATH-MATIC CONTROL STATEMENTS

- 1. (A) READ Δ X Δ Y Δ Z Δ .
 (B) READ Δ X Δ Y Δ Z Δ IF Δ SENTINEL Δ JUMP Δ TO Δ SENTENCE Δ F Δ .

2. TYPE-IN Δ X Δ Y Δ Z Δ .

3. PRINT-OUT Δ X Δ Y Δ Z Δ .

4. (A) EDIT Δ X Δ Y Δ Z Δ .

(B) EDIT Δ CONVERTED Δ X Δ Y Δ Z Δ .

(C) EDIT Δ FOR Δ UNI PRINTER Δ X Δ Y Δ Z Δ .

5. WRITE Δ X Δ Y Δ Z Δ .

6. (A) EDIT Δ AND Δ WRITE Δ X Δ Y Δ Z Δ .

(B) EDIT Δ AND Δ WRITE Δ CONVERTED Δ X Δ Y Δ Z Δ .

(C) EDIT Δ CONVERTED Δ AND Δ WRITE Δ X Δ Y Δ Z Δ .

(D) EDIT Δ AND Δ WRITE Δ FOR Δ UNI PRINTER Δ X Δ Y Δ Z Δ .

(E) EDIT Δ FOR Δ UNI PRINTER Δ AND Δ WRITE Δ X Δ Y Δ Z Δ .

NOTE 1. IN THE ABOVE 5 SENTENCES EDIT AND WRITE MAY BE TRANSPOSED.

7. (A) VARY Δ X Δ Y Δ 0 Δ (Δ X) Δ LX, Δ SENTENCES Δ F Δ THRU Δ L Δ .

(B) VARY Δ X Δ X Δ 0 Δ (Δ X) Δ LX, Δ Y Δ Y Δ 0 Δ (Δ Y) Δ LY, Δ SENTENCES Δ F Δ THRU Δ L Δ .

(C) VARY Δ X Δ X Δ 0 Δ (Δ X) Δ LX, Δ Y Δ Y Δ 0 Δ (Δ Y) Δ LY, Δ Z Δ Z Δ 0 Δ (Δ Z) Δ LZ, Δ SENTENCES Δ F Δ THRU Δ L Δ .

8. (A) VARY Δ X Δ X Δ 0 Δ X Δ 1 Δ X Δ 2 Δ X Δ 3 Δ X Δ 4 Δ ... Δ X Δ N Δ SENTENCES Δ F Δ THRU Δ L Δ .

(B) VARY Δ X, Δ Y, Δ Z, Δ X Δ 0, Δ Y Δ 0, Δ Z Δ 0, Δ X Δ 1, Δ Y Δ 1, Δ Z Δ 1... Δ X Δ n, Δ Y Δ n, Δ Z Δ n,SENTENCES Δ F Δ THRU Δ L Δ .

9. JUMP Δ TO Δ SENTENCE Δ F Δ .

10. (A) IF Δ X Δ r' Δ Y Δ JUMP Δ TO Δ SENTENCE Δ F Δ .

(E) IF Δ X Δ r' Δ Y Δ JUMP Δ TO Δ SENTENCE Δ F Δ 1, Δ IF Δ

X Δ r' Δ Y Δ JUMP Δ TO Δ SENTENCE Δ F Δ 2, Δ IF Δ

X Δ r' Δ Y Δ JUMP Δ TO Δ SENTENCE Δ F Δ 3 Δ .

r' = Any of the three relations " equal to (=), greater than (>), less than (<).

11. (A) EXECUTE SENTENCE A.
(B) EXECUTE SENTENCES A THROUGH L.

12. STOP.

13. (A) COMPILER-A.
(B) COMPILER-A INPUT STORAGE-B SERVO-2.
(C) COMPILER-A OUTPUT STORAGE-B SERVO-3.
(D) COMPILER-A INPUT STORAGE-A SERVO-2, OUTPUT STORAGE-B SERVO-4.

NOTE 2: The above sentences may be followed by any amount of English description.

14. (A) COMPUTER-4.
(B) COMPUTER-5A INPUT STORAGE-E SERVO-5.
(C) COMPUTER-15A OUTPUT STORAGE-A SERVO-3.
(D) COMPUTER-8A INPUT STORAGE-B SERVO-4,
OUTPUT STORAGE-C SERVO-6.

SEE NOTE 2 ABOVE.

FUNCTIONAL CALL-WORDS

Trigonometric functions

SINAX	COTAX
COSAX	SECAX
TANAX	CSCAX

1. Each of the above functional call-words will produce the one or two Arithmetic operations necessary to compute the function.
2. The argument may be in any allowable alphabetic or numeric form, and is assumed to be in radians.
3. $|X|$ must be less than 10^{11} ; otherwise an error printout will occur during the Univac run.

Hyperbolic functions

SINHAX

COSHAX

TANHAX

1. Each of the above functional call words will produce the Arithmetic operation necessary to compute the function.
2. The argument may be in any allowable alphabetic or numeric form.

NOTE: This uses LAD in A-3. Hence if A is
req. will have trouble.

General

A) AAPQWAB (A^B)

1. The above functional call-word will, depending on its parameters, select the most suitable Arith-matic operations to compute A^B .
2. A and B may be any allowable alphabetic or numeric form.
3. If either A or B requires more than one symbol in order to be stated (e.g., power of ten form) it must be enclosed in parentheses.
4. If B is numeric, A^B should be written in ordinary mathematical form. If B is alphabetic, the above functional call-word must be used.

B) NAROOTAA ($\sqrt[N]{A}$)

1. The above functional call word will, depending on its parameters, produce the most suitable Arith-matic operations to compute $A^{1/N}$; (e.g., 3AROOTAA will give the result $A^{1/3}$ or $\sqrt[3]{A}$).
2. A and N may be any allowable alphabetic or numeric form.
3. If either A or N requires more than one symbol in order to be stated (e.g., power of ten form), it must be enclosed in parentheses.
4. If N is numeric but not integral, $A^{1/N}$ should be written in ordinary mathematical form. Otherwise, the above call word, or the call word AAPQWAB, may be used.

C) SQRAA (\sqrt{A})

1. The above functional call word will produce the Arithmetic operation necessary to compute \sqrt{A} .
2. A may be any allowable alphabetic or numeric form.
3. If A requires more than one symbol in order to be stated, (e.g., power of ten form), it must be enclosed in parentheses.

D) EXPAA (e^A)
LOGAA $(\log_{10} A)$
LNAA $(\log_e A)$

1. The functional call-word in column I will produce the Arithmetic operations necessary to compute the function in column II.
2. A may be any allowable alphabetic or numeric form.
3. If A requires more than one symbol in order to be stated (e.g., power of ten form), it must be enclosed in parentheses.
4. If LOGAA or LNAA is desired, the user must ensure that $A > 0$. If $A \leq 0$, an error print-out will occur during the Univac run.

E) ARCTANAA

1. The above functional call-word will produce the Arithmetic operation necessary to calculate the arc-tangent of A in radians.
2. A may be any allowable alphabetic or numeric form.

(n) ΔREADΔXΔYΔZΔ.

(n) ΔREADΔXΔYΔZΔIFΔSENTINELΔJUMPΔTOΔSENTENCEΔFA.

Functions: This sentence will assign a new set of values to the listed variables each time it is executed. These set of values must be prepared by the coder in two-word floating decimal form and stored on an input tape, from which a new block is read each time the previous block is exhausted. When the Sentinel (Z-----Z) in the first word of the set of values is reached, control will be transferred to sentence F, if the jump phrase has been included. During the Math-matic run, instructions will be typed out on the supervisory control telling the operator where to mount his data during the problem run.

Conventions:

1 - The following numbers of variables are acceptable in this sentence:

1, 2, 3, (4), 5, 6, 10, 15, and 30. *Note: For 4 variables, each block must be padded after 7th set*

2 - To read an unacceptable number of variables less than 30 (over 30 is not permitted) each input item on the tape must be padded to the next highest acceptable item size.

3 - The data must be prepared in two-word floating decimal form. Therefore, each allowable item size is double the corresponding number of variables. (e.g., An eleven variable item must be padded to a fifteen variable, or thirty word item). This will require eight words of padding.

4 - If an unacceptable number of variables is desired, the read sentence may be padded to the next highest number of acceptable variables. This eliminates a print-out and type-in during the Math-matic run.

5 - The variable must be listed in the order in which the values for them appear on tape.

6 -- A maximum of eight input or output statements is permitted. Since "read" is an input statement, care must be exercised not to exceed the limit.

Error Printouts: See Print-outs #7, 8, 9, and 10 under heading, "Phase II Error Print-outs".

(n) ATYPE-INAKAYAZA.

Function: This sentence will cause type-ins to be set up during the program run, for all of the listed variables.

Conventions:

- 1 - Any number of variables may be listed in the sentence.
- 2 - Two type-in instructions will be set up for each variable in the statement. The pseudo-coder should prepare to type-in each value in normalized two-word floating decimal form.

Error Printouts: None

(n) PRINT-OUTXAYAZA.

Function: This sentence will cause the values of the listed variables to be typed-out on the Supervisory control. These values will appear in two-word floating decimal form.

Conventions:

1 - Any number of variables may be listed in the sentence.

Error Printouts: None

- 1 (n) Δ EDIT Δ A,B,C Δ .
- 2 (n) Δ EDIT Δ CONVERTED Δ A,B,C Δ .
- 3 (n) Δ EDIT Δ FOR Δ UNIPRINTER Δ A,B,C Δ .

Function: The three statements above represent the permissible options for producing the desired type of editing. For the High-Speed Printer, Statements 1 or 2 should be used. Statement 1 (as shown above) will edit variables A, B, and C for High-speed Printer, in unconverted form (See definitions below). Statement 2 will also edit for High-speed Printer, in converted form. For the Uniprinter, no converted form is available and Statement 3 should be used.

Definitions:

1 - Unconverted: In this form, the value of a given variable is shown, correct to eleven significant digits, with a decimal point on the left, and followed by a three digit, positive or negative power-of-ten exponent in parentheses. If this exponent is greater than 999, the value is left unedited in normal two-word floating decimal form.

The edited form is as follows: Δ .XXXXXXXXXX X Δ (+XXX) Δ AAA Δ .

2 - Converted: Here, the positive or negative value of a given variable is shown, with its decimal point properly placed, correct to eleven significant digits. If the power-of-ten exponent in the original two-word floating decimal form was greater than ten, the value is left unedited.

Edited form: Δ XXXXXXXXXX Δ +XX.XXXXXXXXXX.

Conventions:

1 - A maximum of 30 variables, each of which may consist of as many as 12 characters may be contained in an edit statement.

2 - Any sequence of all or part of the variables involved in a given problem may appear in an Edit statement. The Write statement corresponding to a given Edit statement must contain the same variables in the same order (See Convention No. 4)

3 - If it is desired, for any reason, to edit the same variables for two or more different outputs in the same problem, they should be listed in different order in the two Edit statements.

4 - Each Edit statement must have an associated Write statement which may appear anywhere in the same problem, unless the Edit and Write statements have been written as a single statement (See Edit write glossary).

Error Printouts: See Print-outs 11, 12, 13, and 14, under the heading "Phase II Print-outs".

1. (n)ΔWRITEAA,B,CA.

Function: These sentences will produce instructions which will write unedited values of A, B, and C to tape, for High-speed Printer (See Edit Glossary), if used alone. If used in conjunction with an Edit statement, appearing elsewhere in the same problem, it will produce instructions in conformity with the requirements of the associated Edit statement (High-speed Printer unconverted or converted, or Uniprinter). In such instances, its variables must be identical with and in the same order as those of the associated Edit statement.

Conventions:

1 - A Write statement must be written somewhere in the same problem for every Edit statement, and its variables must be identical with and in the same order as those in the associated Edit statement.

2 - A Write statement may contain a maximum of 30 variables each of which may consist of as many as 12 characters.

3 - If a Write statement is associated with an Edit statement in the same problem, no other Write statement may contain the same variables in the same order.

Error Printouts: See Print-outs # 12, 13, 14, and 15, under the heading, "Phase II Print-outs".

1. (n)ΔEDITΔANDΔWRITEΔA,B,CΔ.
2. (n)ΔEDITΔANDΔWRITEΔCONVERTEDΔA,B,CΔ.
3. (n)ΔEDITΔCONVERTEDΔANDΔWRITEΔA,B,CΔ.
4. (n)ΔEDITΔANDΔWRITEΔFORΔUNIPRINTERΔA,B,CΔ.
5. (n)ΔEDITΔFORΔUNIPRINTERΔANDΔWRITEΔA,B,CΔ.

Function: The five statements above represent the permissible variations and options for combining the three Edit statements with the Write statement, producing the same results as the corresponding Edit statements would alone, with an associated Write statement located somewhere else in the same problem. The difference between Statements 2 and 3, and between Statements 4 and 5 is simply one of word-order, and the user can choose either one of each pair to achieve the same result. Another option makes it possible to transpose the words "EDIT" and "WRITE" in any of the above five statements without altering their function.

Conventions:

1 - Any of the five combined Edit and Write statements may contain a maximum of 30 variables each of which may consist of as many as 12 characters.

2 - Multiple combined Edit and Write statements may be used, provided they contain either different variables or the same variables in different order.

Error Printouts: See Print-outs # 11, 12, 13, and 14 under the heading "Phase II Print-outs".

(n) VARY A, A₀, Δ(ΔA) ΔA SENTENCES F THROUGH L*.

Function: This sentence will cause the calculations indicated by Sentences F through L to be repeated, each time for a different value of A. The values of A will vary from A₀ up to and including limit A in increments of delta A. When the limit A is exceeded, this sequence of execution will be broken and control will be transferred to the sentence following M.

Conventions:

- 1 - The initial value, increment and limit of A may be either alphabetic or numeric (any acceptable numeric form).
- 2 - The range (limit A - A₀) need not include an integral number of increments.
- 3 - Sentence L must follow Sentence F in order of execution; however, L need not necessarily be greater than F.
- 4 - Sentence n must precede sentence F in order of execution.

Error Printouts: See Print-outs # 16, 17, and 18 under heading, "Phase II Error Print-outs".

* FORMAT for "vary" statement which will handle two or three variables at one time will be found immediately following.

(n) Δ VARY $\Delta A_0 \Delta (\Delta A) \Delta A \Delta B \Delta B_0, (\Delta B), LB \Delta C$
 $C_0 \Delta (\Delta C) \Delta C$ SENTENCES F THRU L.

Function: This sentence may include either two or three variables e.g., A, B, or A, B and C. This sentence will cause the calculations indicated by Sentences F through L to be repeated, each time for a different Set of values for A and B, or A, B, and C. The sets of values of A, B, and C will vary from A_0, B_0, C_0 up to and including limit A, limit B, limit C, A being incremented by delta A, B by delta B, and C by delta C for each repetition. When the limit of C is exceeded by C, this sequence of execution will be broken and control will be transferred to the sentence following L.

Conventions:

1 - The initial value, increment and limit of the variables may be either alphabetic, or numeric and may be absolute.

2 - All variables will be incremented simultaneously, but the repetition will cease only when the limit of the last variable is exceeded. However, dummy limits of the other variables must be included in the format of the sentence.

3 - The range of the last variable, (e.g., limit C--- C_0) need not include an integral number of increments.

4 - Sentence L must follow sentence F in order of execution; however, L need not necessarily be greater than F.

5 - Sentence n must precede sentence F in order of execution.

Error Printouts: See Print-outs # 16, 17, 18, and 19 under heading, "Phase II Error Print-outs".

(n) Δ VARY Δ X Δ X₀ Δ X₁ Δ X₂ Δ X₃ Δ ... Δ X_n Δ SENTENCES Δ F Δ THRU Δ L Δ .

(n) VARY Δ X Δ Y Δ Z Δ X₀ Δ Y₀ Δ Z₀ Δ X₁ Δ Y₁ Δ Z₁ Δ X₂ Δ Y₂ Δ Z₂ Δ
 ... Δ X_n Δ Y_n Δ Z_n Δ SENTENCES Δ F Δ THRU Δ L Δ .

Functions: This sentence, called a vary of the "list type", will cause the calculations indicated by sentences F thru L to be repeated, each time for a different set of values for X, Y, Z, ... After the final set of values (X_n, Y_n, Z_n, ...) has been used, the sequence of execution will be broken and control will be transferred to the sentence following L.

Conventions:

- 1 - A maximum of ten variables is allowed.
- 2 - All the values of the variables must be numeric, not alphabetic.
- 3 - In each set of values there must be one value for each variable; that is, the total number of values must be evenly divisible by the number of variables.
- 4 - A maximum of 220 such values is allowed. This maximum is reduced to 110 if the values are expressed in power-of-ten form.
- 5 - The sequences of execution involved in different vary sentences of the list type must not overlap. Since they use the same working storage area and the same input tape, only one Set of variables may be processed at a time.
- 6 - If the statement of the problem is going to violate rule 5, a READ Sentence should be used.
- 7 - Sentence L must follow sentence F in order of execution; however, L need not necessarily be greater than F.
- 8 - Sentence n must precede sentence F in order of execution

Error Printouts: See Print-outs # 20, 21, 22, 23, and 24 under heading "Phase II Error Print-outs".

(n)ΔJUMPΔTOΔSENTENCEΔF

This sentence will break the sequence of execution and transfer control to Sentence F.

Function: F may be the number of any sentence in the problem.

Error Printouts: See Print-out #19 under heading "Phase II Error Print-outs".

(n)ΔEXECUTESENTENCEΔFΔ.

(n)ΔEXECUTESENTENCEΔFΔTHRUΔLΔ.

Function: These Sentences will cause Sentences F, or F through L to be executed, after which control is transferred the Sentence following n.

Conventions:

1 - Sentence L must follow sentence F in order of execution; however, L need not necessarily be greater than F.

2 - Sentence n must lie outside the range of execution, F thru L.

Error Printouts: See Print-out #26 under heading, "Phase II Error Print-outs".

(n)ΔSTOPΔ.

Function: This sentence must be the last sentence in the problem, numerically as well as in the order of execution. It indicates to Mathematic that the end of the program has been reached.

Conventions:

1 - A STOP Sentence may be followed by directory and computer and/or compiler sections. (See Instructions for writing computer and compiler sections.)

Error Printouts: None

(n)ΔCOMPILER-1Δ.*

(n)ΔCOMPILER-99ΔINPUTΔSTORAGE-BΔ.
SERVO-4Δ.

(n)ΔCOMPILER-11AΔOUTPUTΔSTORAGE-AAΔ
SERVO-6Δ.

(n)ΔCOMPILER-2DΔINPUTΔSTORAGE-LΔ
SERVO-8ΔOUTPUTΔSTORAGE-J
SERVO-5Δ.

Function: This sentence informs the Math-matic system that a section of Arith-matic code has been written on the input tape following the pseudo-code. If the compiler section employs any read or write orders, the corresponding storage block and servo number must be specified in the Sentence.**

Conventions:

1 - The label of the compiler Section may have as many as three digits, either numeric or alphabetic or mixed. The label on the statement must match the label on the section.

2 - The Storage blocks, if any, may be labeled with any letter of the alphabet from A thru J.

3 - The servo-numbers assigned may be any number 2 through 9 and minus.

4 - Any amount of English description may follow the compiler sentence. In this case the Δ (space).(period) which indicates the end of the sentence will come after the description. The purpose of this description is to complete the clear statement of the problem in pseudo-code.

Error Printouts: See Print-outs # 30 and 31 under heading, "Phase II Error Print-outs".

* See special instructions for writing Compiler of Arith-matic code.

** See special instructions for writing the directory and compiler sections.

(n)△COMPUTER-1△.

(n)△COMPUTER-99△INPUT△STORAGE-B△
SERVO-4△.

(n)△COMPUTER-11A△OUTPUT△STORAGE-A△
SERVO-6△.

(n)△COMPUTER-2D△INPUT△STORAGE-I△
SERVO-8△OUTPUT△STORAGE-J△
SERVO-5△.

Function: This sentence informs the Math-matic System that a section of Univac code has been written on the input tape following the pseudo-code. If the computer section employs any read or write orders, the corresponding storage block and servo must be specified in the Sentence.*

Conventions:

1 - The label of the computer Section may be up to three digits; either numeric or alphabetic or mixed. The label in pseudo-code must match the label of the section.*

2 - The storage blocks, if any, may be labelled with any letter of the Alphabet from A thru J.

3 - The Servo numbers assigned may be any number 2 through 9 and minus.

4 - Any amount of English description may follow the Computer sentence. In this case, the △ (space).(period) which indicates the end of the sentence will come after the description. The purpose of this description is to complete the clear statement of the problem in pseudo-code.

Error Printouts: See Print-outs # 32 and 33 under heading, "Phase II Error Print-outs".

* See special instructions for writing directory and computer sections.

2.544
6.25
W/101

COMPILER AND COMPUTER SECTIONS

Should the coder wish to include some function in his program that is not yet available in the Math-matic library, he may code this function in either Arith-matic pseudo-code (Compiler sections) or Univac code (Computer sections). When using either compiler or computer sections, the following conventions must be observed:

- 1) Each section must begin a new block and must follow the Math-matic sentences on the input tape.
- 2) Corresponding to each section there must be a computer or compiler statement included among the Math-matic sentences. The label used in the sentence must be the same as that used in the header of the section. (e.g., COMPILER-1; COMPUTER-XYZ).
- 3) If any input or output instructions are used in the section, the storage block letters and servo numbers must be specified in the associated Math-matic sentence, and coding in the section may directly address entries in these storage areas.
- 4) Should a section make reference to any variables that are employed elsewhere in the program, it is necessary that the various usages be identified. This function is performed by the "Directory".

The directory consists of a list of variables and constants that are used in the computer and compiler sections, and is subject to the following conventions:

- a) The directory must begin in the first block after the "STOP" sentence, and must precede the computer and compiler sections.
- b) The directory must have a header (DIRECTORY) and an end sentinel (END).

COMPILER AND COMPUTER SECTIONS (Cont'd)

- c) The entries in the directory may be variables or constants, expressed in any of the acceptable forms. They should appear on the left of space-filled words.
 - d) Computer and compiler sections make reference to directory entries by "W" addresses.
 - e) The directory may have a maximum of 99 entries (W01 to W99)
- 5) The end sentinel "END△COMPILER" (END△COMPUTER) must follow the last valid instruction in each section.
- 6) "Own code" sections of Arithmetic pseudo-code ~~should~~ ^{must} be written as separate computer sections rather than as part of a compiler section.
- 7) Symbology used in Compiler Sections:
- a) Fourth, seventh and tenth operation digits.
 - 1. "A" thru "J" refers directly to input or output storage blocks.
 - 2. "W" - refers to an entry in the directory (e.g., AA0; B02; W15; C04).
 - b) All jump operations are in the form $OXCNOO \begin{matrix} O \\ M \end{matrix} Onn\Delta$.
If "M" is in the seventh digit "nn" refers to an operation within the section. If zero is used "nn" refers to a sentence number.
- 8) Symbology used in Computer Sections:
- a) Third and ninth instruction digits.
 - 1. "A" thru "J" - refers directly to input or output storage blocks.

COMPILER AND COMPUTER SECTIONS (Cont'd)

2. "M" - refers to an instruction within the computer section.
3. "W" - refers to an entry in the directory. A "zero" immediately following the "W", refers to the mantissa of the entry, a "one" to the characteristic of the floating decimal value of the entry. (e.g., BOW003; LGW100).

b) All jumps to outside the computer section are indicated by "XXCN" in the 9th thru 12th instruction digits. The sentinel "END△OWN△△△△△" follows the computer code, after which the XXCN's are listed. Each XXCN word contains the sentence number to which control is to be transferred.

*Can you refer to all A-3 operation?
NO*

e.g.,

000000U001CN

⋮

END△OWN△△△△△ ←

01CN000012△△

END△COMPIER

- 9) For further information on writing computer and compiler code see "Rules for Writing Arithmetic Pseudo-Code".

EXAMPLE OF COMPILER SECTION AND DIRECTORY:

(4) COMPILER-1ΔINPUTΔSTORAGE-AΔSERVO-5Δ.

Directory - first block after "STOP" Sentence.

```

000)  DIRECTORYΔΔΔ
      XΔ-----Δ
      YΔ-----Δ
      ZΔ-----Δ
      7Δ-----Δ
      .....
      .....
      ENDΔIRECTRY
    
```

W00 header.

```

W01 }
W02 }   Variables
W03 }
    
```

W04

} associated with another computer
or compiler section

} end sentinel.

Compiler Section

```

X      COMPILER-1ΔΔ
M00    GMIO5A002A00
M01    AACW01W04W01
M02    QTOW01W02000
      X      01CN00M00500
M03    GMMW01002W03
M04    U000000000000
      X      01CN00001000
M05    AMGW02A02W03
M06    U000000000000
      X      01CN00000900
      X      ENDΔCOMPILER
    
```

Header = Matcher reference in Sentence 4

Reads a block from tape 5 into storage blocks A in accordance with Sentence 4.

$X + 7 = X.$

Test X : Y

If $X > Y$, go to opn. # 5

If $X \leq Y$, go to opn. # 3

Move the floating decimal value of X to Z.

Control is unconditionally transferred to sentence 10 in the program.

Multiply Y by the quantity in word 02 of the storage block filled by operation one. Place the result in Z.

Control is unconditionally transferred to sentence nine in the program.

EXAMPLE OF COMPUTER SECTION AND DIRECTORY:

X Δ

(4) ΔCOMPUTER-1 ΔINPUT STORAGE-A, SERVO-5 Δ.

Directory - first block after "STOP" Sentence.

DIRECTORY Δ Δ Δ	<i>no of</i>	W000 -- header	
X Δ ----- Δ		W001	} Variables
Y Δ ----- Δ		W002	
Z Δ ----- Δ		W003	
7 Δ ----- Δ		W004 --	Constant
.....		} associated with another	computer or compiler section
.....			
ENDDIRECTORY		End sentinel	

NOTE: W 1 is not
nec. before W 2
nos. are not consec.

Computer Section

all addresses should be increased by 1

X	COMPUTER-1 Δ Δ	Header: Matches reference in Sentence 4
M001	15000030A000	Reads a block from Tape 5 into storage block A in accordance with Sentence 4.
M002	B0W100L0W101	W101 and W102 refer to the characteristics of the variables, (X and Y)
M003	V2W003Q0M006	M006 refers to the sixth line of the computer section.
M004	V2W001T0M006	
M005	V2W000000000	W000 refers to the mantissa (and the characteristic, in the case of a V instruction) of X.
M006	W2W002B0A010	A010 refers to the tenth word in storage block A.
M007	H0W102T001CN	1CN is a reference to a sentence number in the program.
M008	W0W102U001CN	
X	END Δ W N Δ Δ Δ Δ Δ	
X	01CN000 Δ 005 Δ Δ	transfer control to sentence 5
X	END Δ COMPUTER	End Sentinel.

THE MATH-MATIC SYSTEM

Math-Matic consists of two major parts; the Translator (AT-3), and the Arith-Matic Compiler (A-3). The translator is made up of four phases and a library of Glossaries. The primary means of communication between the phases is the Sentence File which houses the program in its various states of partial translation. The final output of the translator is a symbolically allocated Arith-Matic program, a data tape, and an unedited record of translation.

The four phases of the compiler make use of the static and dynamic sub-routines, and the generators of the library to produce the C-10 running program. The compiler makes all decisions on segmenting, performs the actual storage allocation, and produces a complete edited record of both translation and compilation.

The following is a brief description of the various phases of the system and the functions they perform. For a complete coverage of this topic, see the Math-Matic Technical Manual.

TRANSLATION PHASE I

By a left to right digital analysis of the pseudo-code, phase I separates the various elements (symbols of operation, arguments, pseudo-words, etc.) into individual space-filled Univac words. All words thus derived from each Math-Matic sentence, together with the appropriate header item and end sentinel comprise one section of sentence file 1. A discrimination between statements and equations is made part of the header.

Each statement call-word is verified by being matched in the Math-Matic catalog, as are all functional call-words used in equations. The word GLOSSARYAAAA is entered in sentence file 1 preceding each valid functional call-word.

Phase I, also converts all superscript symbols to their numerical equivalents, and enters these in sentence file 1 together with their glossary identification.

TRANSLATION PHASE II

It is in phase II that the translation into a modified form of Arithmetic pseudo-code takes place. Each A-3 operation that is produced is in an expanded four word packet. The first word of each packet is the appropriate Arith-Matic all-word. The other three words contain the names of the arguments or result of the operation, or the associated sentence numbers, if jumps are involved.

Equation sections of sentence file 1 are analyzed, and the order in which packets are delivered to sentence file 2 is determined by the following conventions.

1. Parenthesized groupings.
2. Functions and exponentiation (handled by glossaries).
3. Multiplication and division.
4. Addition and subtraction.

Statement sections are processed by the appropriate glossaries. They may produce not only packets for sentence file 2, but also entries to:

1. Control List 1: Statements involving a range of sentences (Vary, Execute, Etc.)
2. Storage List 1: Statements concerned with input or output (Read, Write, etc.)
3. Data List 1: Statements containing data (Vary list).

TRANSLATION PHASE III

Using the information contained in storage list 1, phase II assigns the necessary input and output storage blocks and servos, and operating

instructions for the problem run are printed out on the supervisory control printer.

Storage addresses are assigned to all variables and constants that appear in the packets of sentence file 2, the packets with the addresses replacing the names of the operands are contracted into their normal Arith-Matic form, and entered in sentence file 3. A complete record of the allocation of storage is produced as storage list 2. All constants in sentence file 2 and data list 1 are converted to normal two word floating decimal form to produce data list 2 for the problem run.

Control list 1 is sorted according to the range components of its entries and after addresses are supplied, for its operands, is written as control list 2.

TRANSLATION PHASE IV

Phase IV completes the allocation of storage, assigning addresses for the partial results of the various operations of each equation. Sentence numbers, in transfer instructions, are converted to operation numbers, and redundant operations are eliminated. Operations contained in control list 2 are integrated with those from sentence file 3, and any computer or compiler sections contained in the pseudo-code are processed and inserted in the Arith-Matic program tape. This program tape comprises a complete statement of the problem in Arith-Matic pseudo-code.

Another output of Phase IV is the sentence list. This is a cross reference table of Math-Matic sentence numbers, and the associated Arith-Matic operation numbers. It will become part of the final record of compilation.

ARITH-MATIC SWEEP 1

Each Arith-matic pseudo-code operation is checked in the catalog of subroutines, and if valid is expanded into a more digestible form, and entered into operation list 1. Any generator or molecular subroutines requested by the pseudo-code are activated at this time, and the generated subroutines are entered in the generated library. Molecular subroutines add their atomic operations to operation list 1. Own code sections of pseudo-code are copied to the generated library.

ARITH-MATIC SWEEP 2

Sweep 2 deals primarily with the problem of segmenting the program. As operation list 1 is copied to operation list 2, a tally is kept of the number of lines of code required by the operations. Governed by the loop sentinels in the list, sweep 2 uses this tally to decide when to send the segment sentinel to operation list 2.

All necessary read instructions to bring in the segments of the running program are generated at this time.

ARITH-MATIC SWEEP 3

Each segment of operation list 2 is inspected for repeated operations. Any repeated operations that are listed in the substitution catalog will be replaced by the appropriate substitution operation in operation list 3.

Sweep 3 also performs the function of allocating storage locations for each subroutine. The call word of each subroutine that is required to solve the problem, along with the starting address and ending address of the subroutine, are entered in the Subroutine List.

ARITH-MATIC SWEEP 4

All operation call-words in operation list 3 are examined during this sweep. Each generator or "own code" operation calls forth a subroutine from the generated library. All other operations find their subroutines in the Arith-matic library. The proper address modifications are made to each subroutine, and the subroutines are linked together to form the final running program.

RECORD EDIT

Following the completion of Arith-matic Sweep-4, the Sentence List, Storage List, and Subroutine List are brought in and edited into a concise, easily legible format, providing a record of the transition from Math-matic pseudo-code sentences through Arith-matic pseudo-code operations to the final Univac program. These stages of translation and compilation are cross-referenced and the record provides an invaluable aid in debugging any logical errors which may have occurred in setting up a given problem. The record also shows the layout of the memory during the problem run.

HOW TO INCREASE THE REFERTOIRE
OF MATH-MATIC CALL-WORDS AND
CONTROL STATEMENTS

* * * * *

Each Math-matic functional call-word or control statement activates a glossary during Phase II of the Math-matic translation. Each glossary knows the permissible formats of the control statement or call-word associated with it. The range of allowable formats depends on the degree of flexibility the programmer puts in the glossary. Based on the parameters in the sentence, the glossary will determine the appropriate Arith-matic operations and special list entries internal to the Math-matic system. The glossary directs this translation by selecting and using the sub-routines of phase II to turn out the four word packets of Sentence File 2. (See sample problem).

The repertoire may be augmented in two ways; new Arith-matic sub-routines may be added to the Arith-matic library (see special instructions on Arith-matic library) or new glossaries may be written corresponding to the desired call words and control statements.

New glossaries may produce any combination of Arith-matic pseudo-code operations, and if any of these are not already in the Arith-matic library they must be added.

There are two types of sentences: equations and control statements. In an equation, whenever a special function denoted by a functional call-word appears, a glossary is called for which will create the Arith-matic operations necessary to calculate the function.

In every control statement the first word of the sentence brings in a glossary which produces the Arithmetic operations for the entire sentence and makes any necessary Control, Storage, and Data-list entries. Whenever a statement deals with input or output, (e.g., Read; Write; Edit and Write) or includes data (e.g., Vary of the list type) it must make appropriate entries in the Storage list. Any statement which refers to a range of one or more sentences (e.g., Execute; Vary) must make entries in the control list. Any statement which includes data in its format (e.g., Vary of the list type) must make entries in the Data-list. The rules governing these entries will be found in the list of sub-routines for Phase II.

The two types of glossaries are similar in their function of producing Arithmetic pseudo-code, and the general rules discussed below apply equally to both.

I. - Each glossary must have a two word header.

1st word: GLOSSARY $\Delta\Delta\Delta\Delta$

2nd word: (NAME OF GLOSSARY) $\Delta\text{-----}\Delta$

In a control statement, this name is the first word of the statement (e.g., Vary, IF). In an equation this name is the functional call-word. (e.g., SIN; COS; LOG).

II. - A control statement glossary may use up to four blocks of the memory; a functional call-word glossary may use up to two. If the statement includes data in its format, its glossary must create a data list. Since this requires a block of memory, the available blocks would be reduced to three. An equation glossary has two blocks of memory available to it.

III - The first block of the glossary is read in by the main routine during Phase II. Succeeding blocks must be read in by the glossary itself. Any overlay must also be handled by the glossary.

IV - Glossaries should be written in symbolic form and entered in the library by means of the Math-matic librarian. The following conventions must be observed:

A. Reference to subroutines in phase II are made by use of the three digit names of the routines (See "Subroutines for Glossary Use") (e.g., ROROCNUOJOCN is a return jump to subroutine OCN).

B. In the third or ninth digit positions:

1. "M" refers to a line in the Glossary

2. "R" refers to the exit line of the subroutine indicated by the address portion of the instruction.

(e.g., ROROOC records in the exit line of sub-routine OOC).

3. "J" refers to the entrance line of the subroutine indicated by the address portion of the instruction.

(e.g., UOJRWJS transfers control to subroutine RWS).

4. "K" refers to a line in the constant pool of phase II. A list of these constants follows this section.

5. "W" refers to a storage location used as input or output by the Phase II subroutines. The descriptions of these sub-routines indicate the "W" storages used by each.

6. "A", "B", "C" and "D" refer to Sentence File 1. Sentence File 2, the Control list and the Storage list respectively.

C. The sentinel "ENDΔGLOSSARY" must follow the last valid word in the glossary.

- D. Use of the sentinel "OVERLAY#####" will cause the librarian to fill and write the block of processed glossary, and read in the next block of symbolic glossary.
- E. The word "CONSTANTSOXX" (OXX is the number of such constants) must precede any constants included in the glossary. These constants will not be modified by the librarian.
- F. The sentinel words used in the symbolic glossaries should not be counted as instructions since they will not appear in the C-10 glossary that is entered in the library. The two word header, however, is counted.
- G. In statement glossaries the "M address" may range from M000 to M239, in equation glossaries to M119. If glossary overlays are used, duplication of addresses will be necessary.

SUBROUTINES FOR ALL GLOSSARIES

Exit COG

Function: This is the normal exit from the glossaries. When the glossary has been entered from a control statement, upon exiting, end sentinels are entered in Sentence File 2, and the pertinent block of Sentence File 2 is written on tape. Control is then transferred to the master routine of Phase 2, which continues the processing of Sentence File 1. In any event, the glossary writer need not be concerned with affixing end sentinels to Sentence File 2 or turning out the final block of a given section, as this is taken care of automatically by the main body of the system in all cases.

Error Printouts: None

Exit RMS

Function: This is the error exit from the glossaries. Should an error in pseudo-code require a rewrite of the problem, a transfer to this routine will rewind all serves and stop the computer. It is suggested that all glossaries offer this exit optionally to the coder, so that he may exercise his discretion upon analyzing the error.

Error Printouts: None

Subroutine 000

Function: Pick up the four word output packet from working storage and deliver it to Sentence File=2. When the output block is filled, it will be written on tape.

PICKS UP FROM	DELIVERS TO	REGISTERS UNAFFECTED
W ₀ : - A-3 Call Word	SF-2: - Four Word Packet	rF, rY
W ₁ : - First Argument	rA: - Zeros.	
W ₂ : - Second Argument		
W ₃ : - Result.		

Error Printouts: None

Subroutine OFD

Function: Creates a "Floating Decimal Define Packet" and enters it in Sentence File-2. This routine expects to find the mantissa and characteristic of the number to be defined in working storage. It supplies the title "DEFINE", advances the floating decimal count by one, delivers the next floating decimal symbol to working storage and to Sentence File 1 to replace the present entry, and delivers the four word packet to Sentence File-2. If the count should exceed 99, an error print-out will result.

PICKS UP FROM	DELIVERS TO	REGISTERS UNAFFECTED
W ₁ : - Mantissa of number	W ₀ : - DEFINE△△△△△△△△	rF, rY
W ₂ : - Characteristic of number	W ₃ : - SF-1 } FLOAT△DEC△△△	
	SF-2: Four Word Packet.	
	rA: Zeros	

Error printouts: See Print-out #1, under heading "Phase II Error Print-outs".

Subroutines for Equation Glossaries

Subroutine OWF

Function: Get next non-space entry from Sentence File 1 and deliver it to working storage. If the limit of the grouping is reached, the word "LIMITAAAAAAAA" is delivered.

PICKS UP FROM	DELIVERS TO	REGISTERS UNAFFECTED
Sentence File 1	$\left. \begin{matrix} W_4 \\ rA \end{matrix} \right\} \begin{matrix} \text{Next entry from} \\ \text{Sent. File 1} \end{matrix}$	rF, rV, rY
	$\left. \begin{matrix} W_4 \\ rA \end{matrix} \right\} \begin{matrix} \text{If Limit} \\ \text{"LIMITAAAAAAAA"} \end{matrix}$	

Error Printouts: None

Subroutine OWP

Function: Get previous non-space entry from Sentence File 1 and deliver it to working storage. If the limit of the grouping is reached, the word "LIMIT~~~~~" is delivered.

PICKS UP FROM	DELIVERS TO	REGISTERS UNAFFECTED
Sentence File 1	$\left. \begin{array}{l} W_4 \\ rA \end{array} \right\} \begin{array}{l} \text{Previous entry} \\ \text{from Sent. File 1} \end{array}$	rF, rV, rY
	<p style="text-align: center;"><u>If Limit</u></p> $\left. \begin{array}{l} W_4 \\ rA \end{array} \right\} \text{"LIMIT~~~~~"}$	

Error Printouts: None

Subroutine 00A

Function: Deliver a word of spaces to Sentence File 1 to replace the last entry that had been procured from it.

<u>PICKS UP FROM</u>	<u>DELIVERS TO</u>	<u>REGISTERS UNAFFECTED</u>
CONSTANT <u>POOL</u>	SF-1: - spaces rA: - zeros.	rV, rY

Error Printouts: None

Subroutine OOR

Function: Steps the partial result counter ahead by one, and delivers the next partial result symbol to working storage. Should the count exceed 99, an error print-out will result.

PICKS UP FROM	DELIVERS TO	REGISTERS UNAFFECTED
	W ₃ : { PARARESΔΔΔnn 5A: }	rF, rV, rY

Error Printouts: See Print-out #2, under heading "Phase II Error Printouts".

Subroutine OOA

Function: Delivers the next partial result symbol to working storage, and replaces the last received entry in Sentence File 1 by it.

PICKS UP FROM	DELIVERS TO	REGISTERS UNAFFECTED
	SF-1 } W3 } PARARES $\Delta\Delta\Delta$ nn rA: - zeros	rT, rY

Error Printouts: See Print-out #2 under heading "Phase II Error Print-outs".

Subroutine PFR

Function: Delivers the previous entry and the following entry from Sentence File 1 to working storage. Replaces the present and previous entries in Sentence File 1 by spaces, and the following entry by the next partial result symbol. The partial result symbol is also delivered to working storage.

PICKS UP FROM	DELIVERS TO	REGISTERS UNAFFECTED
Sentence	W ₁ : - Previous entry	rV, rY
File 1	W ₂ : - Following entry	
	W ₃ : - } SF-1 } PARARESAAAAn	
	rA: Zeros	

Error Printouts: See Print-outs #2, 3, and 4 under heading, "Phase II Error Print-outs".

Subroutine JFR

Function: Deliver the following entry from Sentence File 1 to working storage. Replace the present entry in Sentence File 1 by spaces, and the following entry by the next partial result symbol. The partial result symbol is also delivered to working storage.

PICKS UP FROM	DELIVERS TO	REGISTERS UNAFFECTED
Sentence File 1	W ₁ : - Following entry W ₂ : - Zeros <i>Spaces</i> W ₃ } SF-1 } PARAPRES/ΔΔΔnn rA : Zeros	rV, rY

Error Printouts: See Print-outs # 2, 3, and 4 under heading, "Phase II Error Print-outs".

Subroutine LPR

Function: Deliver the previous entry from Sentence File 1 to working storage. Replace the previous entry in Sentence File 1 with spaces, and the present entry with the next partial result symbol. The partial result symbol is also delivered to working storage.

PICKS UP FROM	DELIVERS TO	REGISTERS UNAFFECTED
Sentence	W ₁ : - Previous entry	rV, rY
File 1	W ₂ : - Zeros	
	W ₃ } PARARES AAAA nn	
	SFL }	
	rA : Zeros	

Error Printouts: See Print-outs # 2, 3, and 4 under heading, "Phase II Error Print-outs".

Subroutine 2FR

Function: Deliver the next two following entries from Sentence File 1 to working storage. Replace the present and first following entries in Sentence File 1 with spaces, and the second following entry with the next partial result symbol. The partial result symbol is also delivered to working storage.

PICKS UP FROM	DELIVERS TO	REGISTERS UNAFFECTED
Sentence	W ₁ : - First following entry	rV, rY
File 1	W ₂ : - Second " "	
	W ₃ } PARΔRESΔΔΔΔn	
	SF1 }	
	rA: - Zeros	

Error Printouts: See Print-outs # 2, 3, and 4 under heading, "Phase II Error Print-outs".

Subroutines for Statement Glossaries

Subroutine OCW

Function: Get the next one word entry from Sentence File 1, and deliver it to working storage. Should entries beyond the "ENDSECTION" sentinel be requested, an error printout will result.

PICKS UP FROM	DELIVERS TO	REGISTERS UNAFFECTED
Sentence File 1	W_4 } { Next entry rA } { from Sentence File 1.	rF, rV, rY

Error Printouts: See Print-out #5 under heading, "Phase II Error Print-outs".

Subroutine CON

Function: Get the next entry from Sentence File 1. If the entry is numeric, it will be delivered to working storage as a two-word item, the second word of the item will contain the power of ten of the number if this notation was used in pseudo-code; otherwise it is a word of zeros. If the entry is alphabetic, it will be delivered as a one word item to working storage.

PICKS UP FROM	DELIVERS TO	REGISTERS UNAFFECTED
Sentence File 1	<u>If Alphabetic:</u> W_4 } Next entry from rA } Sent. File 1	rV, rY
	<u>If Numeric:</u> W_1 : - Mantissa of number W_2 : - Characteristic of number. rA : Zeros	

Error Printouts: See Print-out #5 under heading, "Phase II Error Print-outs".

Subroutine CWN

Function: Get the next numerical entry from Sentence File 1, skipping any alphabets, and deliver it as a two word item to working storage. The second word of this item contains the power of ten of the number if that notation was used in pseudo-code, otherwise it is a word of zeros. If no more numerics exist in Sentence File 1, the end sentinel of S.F. 1 will be delivered.

PICKS UP FROM	DELIVERS TO	REGISTERS UNAFFECTED
Sentence File 1	<u>If Numeric:</u> W ₁ : - Mantissa of number W ₂ : - Characteristic of number rA: - Zeros	rV, rY
	<u>If End of Entries:</u> W ₄ : - ENDSECTIONA rA: - " "	

Error Printouts: None

Subroutine OOB

Function: Creates a "Begin Loop Packet" and enters it in Sentence File 2. Advances the loop counter by one and delivers it to working storage. Should the count exceed 99, an error print-out will result.

PICKS UP FROM	DELIVERS TO	REGISTERS UNAFFECTED
-----	W : - BEGINLOOPApp W ₁ } W ₂ } - Δ-----Δ W ₃ } W ₁₄ : ~~~~~ pp SF-2: Four Word Packet. rA: - Zeros	rF, rY

Error Printouts: See Print-out #6 under heading, "Phase II Error Print-outs".

Subroutine OOE

Function: Creates an "End Loop Packet" and enters it in Sentence File 2. Advances the loop counter by one and delivers it to working storage. Should the count exceed 99, an error printout will result.

PICKS UP FROM	DELIVERS TO	REGISTERS UNAFFECTED
-----	W ₀ : - ENDΔLOOPΔΔpp W ₁ } W ₂ } - Δ————Δ W ₃ } W ₁₄ : ~~~~~ pp SF-2: Four Word Packet. rA: - Zeros	rF, rY

Error Printouts: See Print-out #6 under heading, "Phase II Error Print-outs".

Subroutine OCN

Function: Creates an "A-3 Control Transfer Reference" from a sentence number picked up from working storage. This sentence number is properly positioned and combined with a "OCN" counter. Each time this routine is entered, the counter is stepped ahead by one, counting from 01CN to 03CN. The counter is reset to one by "Subroutine 000".

PICKS UP FROM	DELIVERS TO	REGISTERS UNAFFECTED
$W_4: (\Delta\Delta\Delta\Delta\Delta\Delta\Delta\Delta nnn)$ Sentence Number.	$W_4: - 0 \begin{cases} 1 \\ 2 \\ 3 \end{cases} CNCOOOnnnO$	rV, rY

Error Printouts: None

Subroutine 00C

Function: Form the two word control list entry and enter it in Control List 1. This routine expects to find the sentence numbers defining the range of the sentence, the A-3 operation call word, (or the name of the argument in the case of an "add to a limit" operation), and the control indicator (zero or +) in working storage. The sentence numbers, the indicator, and the present loop number are combined to form the control word.

PICKS UP FROM	DELIVERS TO	REGISTERS UNAFFECTED
W ₆ : - First sent. no. of range (F)	W ₈ : - Control word	rY
W ₇ : - Last sent. no. of range (L)	SF-2: Control word A-3 Call word	
W ₉ : - A-3 Call word		
V ₁₈ : Control indicator	RA: - Zeros.	

Error Printouts: None

Subroutine OST

Function: Forms four word header for Storage List 1. If a particular storage block is specified in the pseudo-code sentence, the glossary must change the header (DOOO)

PICKS UP FROM	DELIVERS TO		REGISTERS UNAFFECTED
-----	Stg. Lst. 1	- Header	rF, rV, rY
	D ₀	STORAGE△△△△△	
	D ₁	△-----△	
	D ₂	SENT.△NO△nnn	
	D ₃	(Sentence Name)	
	rA: -	△-----△	

Error Printouts: None

Subroutine S

Function: Make one word entry in Storage List 1. The first entry that each glossary must make is the servo number (if a particular servo is specified in the pseudo-code sentence), or a word of spaces.

<u>PICKS UP FROM</u>	<u>DELIVERS TO</u>	<u>REGISTERS UNAFFECTED</u>
W ₁₀ : - entry for Storage List 1	STG LIST 1: Contents of W ₁₀ rA: - Zeros	rF, rV, rY

Error Printouts: None

Subroutine STZ

Function: Deliver the end sentinel to Storage List 1 and write the block of Storage List to tape.

<u>PICKS UP FROM</u>	<u>DELIVERS TO</u>	<u>REGISTERS UNAFFECTED</u>
-----	STG LIST 2: - ENDSECTIONΔ rA: - Zeros	rF, rV, rY

Error Printouts: None

CONSTANT POOL FOR GLOSSARY USE

0	0---01 0---0
1	0----- 01
2	10-----0
3	0010-----0
4	000111 0---0
5	GMMA-----Δ
6	0----- 0111
7	111111 111111
8	ΔΔΔΔΔ ΔΔΔΔΔ
9	0----- 04
10	0----- 02
11	01CNC-----0
12	0---02 0---0
13	0----- 01000
14	ENDΔSE CTIONΔ
15	0----- 0
16	999999 999999
17	***** *****
18	/Δ-----Δ
19	+Δ-----Δ
20	-Δ-----Δ
21	Δ-----Δ
22	PSEUDO ΔWORDS
23	OPERAT IONΔΔΔ
24	VARIAB LEΔ--Δ
25	DEFINE Δ---Δ

26	0---02 0---02	54	OPERAT IONSΔΔ
27	FIRSTΔ ----Δ	55	REWRIT EΔ---Δ
28	SECOND Δ----Δ	56	STORAG EΔ---Δ
29	ARGUME NTΔ--Δ	57	OIO--- ----0
30	MISSINGΔ----Δ	58	IFΔ--- ----Δ
31	SENTAF ILEΔ01	59	002Δ-- ----Δ
32	STATEM ENTΔ-Δ		
33	0--060 0--060		
34	SENTAF ILEΔ02		
35	EQUATI ONΔ--Δ		
36	0----- 0+		
37)Δ-----Δ		
38	(Δ-----Δ		
39	ABSΔ-----Δ		
40	LEFTΔ-----Δ		
41	PRENT HESISΔ		
42	GLOSSA RYΔ--Δ		
43	*Δ-----Δ		
44	ANDΔ-- ----Δ		
45	ADOΔ-- ----Δ		
46	AAOΔ-- ----Δ		
47	ANIΔ-- ----Δ		
48	ASOΔ-- ----Δ		
49	FNDALQ OPΔ000		
50	TOQAMA NYΔ--Δ		
51	INCORR ECTΔ-Δ		
52	ENDALI STΔ--Δ		
53	ENDAPH ASEΔ02		

EXAMPLES OF A GLOSSARY WITH SENTENCE FILE 1 AND 2

The following example illustrates how the glossary creates Sentence File 2 out of Sentence File 1. Sentence File 2 is composed of four-word packets of which the first word is the Arithmetic call word and the other three specify the parameters. The glossary checks the input parameters for conformity to the allowable formats, and uses the sub-routine of phase 2 to produce the four word packets.

SENT△FILE△01

SENT.△NO.06△

STATEMENT△△△

JUMP△△△△△△△△

TO△△△△△△△△△

SENTENCE△△△△

5△△△△△△△△△

END△SECTION△

SENT△FILE△02

SENT.△NO.06△

STATEMENT△△△

JUMP△△△△△△△△

UOO△△△△△△△△

△-----△

△-----△

OICNO△0005△0

END△SECTION△

000	GLOSSARY△△△△	
001	JUMP△△△△△△△△	name of glossary
002	ROROWNUO ^J OWN	get the next numeric from Sentence File 1.
003	LOMO22QOM011	if the sentence reference is missing the word LIMIT is in the A register.
004	ROROCNUO5OCN	Form 01CNG0000C 500 → W ₄
005	BOWO04COWO03	move W ₄ → W ₃
006	BOKO08HOWO01	spaces → W ₁ and W ₂
007	COWO02BOM021	UOO△△△△△△△△ → W ₀ (This is the Arith-matic call-word)
008	COWO00000000	
009	ROROOUOJOO0	Turn out four word packet W ₀ W ₄ to Sentence File 2.
010	000000UOJOO0	Transfer Control out of Glossary
011	5OWO165OWO17	If there is no Sentence reference in the pseudo code a type out and type-in is set up.
012	5OM0205OM016	
013	5OM0175OM018	"SENT.△NO△n JUMP
014	5OM0191OWO04	SENTENCE REFERENCE OMITTED TYPE-IN"
015	000000UOMO04	The glossary continues using the information typed in.
X	CONSTANTS007	
016	REFERENCE△△△	
017	OMITTED△△△△△	
018	TYPE-IN△△△△△	
019	CORRECTION△△	
020	SENTENCE△△△△	
021	UOO△△△△△△△△	
022	LIMIT△△△△△△	
X	END△GLOSSARY	

OPERATING INSTRUCTIONS FOR MATH-MATIC LIBRARIAN

*Will also be used
for A-3*

The Math-matic Librarian is contained as the first section of the Math-matic library tape. This routine is made for the express purpose of deleting, inserting, replacing and changing the catalog and glossaries contained in the library tape.

Servo allocations will be as follows:

At Beginning of *Librarian Run*	Servo * * *	At End of Librarian * * * * Run * * * * *
Library	5	Old Library
Blank Tape	6	New Complete Library
New Symbolic Glossaries With or Without correc- tion list.	4	New symbolic glossar- ies.

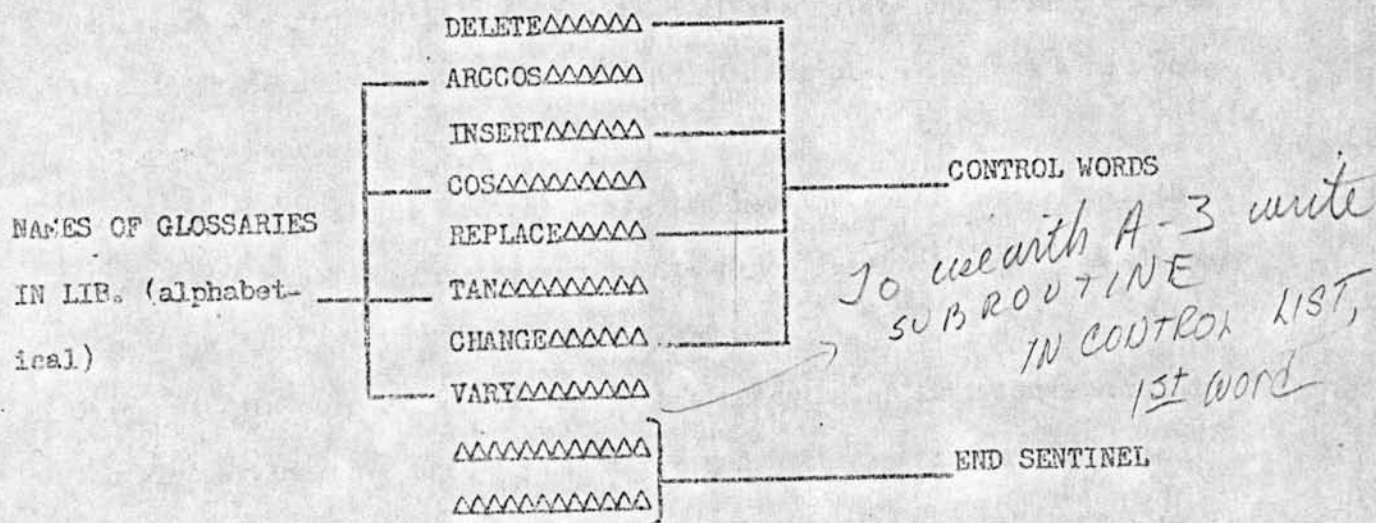
Supervisory Control

1. Force transfer on break point #1 if corrections are on tape four.
2. Block subdivide Servo six for High-speed printing later.
3. Initial-read Servo five.

Each installation should maintain a tape library of symbolic glossaries as well as the functioning library. Should any changes be required in the Math-matic system or should the user wish to adapt his glossaries to the Univac II Math-matic system, it would be sufficient merely to process the symbolic library through the librarian. A librarian to maintain the symbolic library will be supplied with the Math-matic system, and the procedures to be followed are the same as those for the functioning library.

Explanation of use of Librarian:

The control list: This is a list of words naming the library changes, and their associated correction words, for different types of changes. The following illustrates a correction list. Note that the call-words of glossaries in the control list are in alphabetical order, as are the new glossaries.



This correction list can either be typed in or be placed on tape using break-point one option. If the control list is to be typed in, the routine will print-out on supervisory control, "TYPE△OF△CHNG" and a type-in is set up. The four possible type-ins at this point are, Delete, Insert, Replace, and Change. The next type-out will be, "NAME△OF△GLOS" on supervisory control and a type-in will be set up for a glossary name with space fill. After typing in desired changes, two words of spaces will terminate the type-in.

Explanation of Instructions:

Delete - The delete instruction will eliminate a given function from the catalog and glossary library. As in the example correction list, the function ARCCOS would be deleted from the library.

Insert - The insert instruction will enter the new glossary and its associated catalog listing in their proper places in the Mathematic library. The new glossary should appear on the tape on Servo-four. The symbolic coding of the new glossary will be processed into C-10 as it is inserted into the library.

Replace - The replace is a combination of the delete and insert changes in that it deletes the old glossary and in its place inserts a new one from Servo-four with the same name.

As shown in the sample control list, the routine would delete from the library the old TAN glossary and in its place insert the new TAN glossary.

Change - This correction will allow the changing of individual words in the glossary named. A separate change and glossary name must be used for every glossary which requires word corrections. When the routine reaches a change order in the control list, it will type out on supervisory control, "BLK $\Delta\Delta\Delta$ WORD Δ ". This Block (BLK) count refers to the block count relative the glossary only and not the library tape. The form of the type in OOOBBB $\Delta\Delta\Delta$ WWW. The next type-out will be the old word, and the librarian waits for the new symbolic word to be typed-in. In the case of constants, that the glossary writer does not want modified, instead of the new word, the type-in should be, "CONSTANTS Δ NN.", "NN" standing for the number of consecutive constants to follow. A word of twelve Z's will terminate the type-ins of word corrections when "BLK $\Delta\Delta\Delta$ WORD Δ " type-in is ready.

NORMAL PRINTOUTS:

PRINT-OUT	DESCRIPTION	PROCEDURE
1. TYPE OF CHNG	System requires specification of desired change.	Type in one of the following, as required: INSERT△△△△△ DELETE△△△△△ REPIACE△△△△△ CHANGE△△△△△ Type in a word of spaces if changes are completed.
2. NAME OF GLOS	System requires specification of Glossary name.	Type in space-filled name of glossary. Type in a word of spaces if changes are completed.
3. BIK△△△WORD	System requires specification of the block of the glossary and word of that block to be corrected. If the correction is to be skipped, type in a word of spaces.	Type-in BBB and WWW OOOBBFOOOWWW Where B = Block No. Where W = Word No. Type word of Z's if finished.
4. END CORRECTIONS	Corrections have all been made.	Remove and label tapes.
5. TYPE-IN [NN] CONSTANTS	Type-in the constants in the order wanted and number named.	Type in N consecutive words. These words will not be modified.

ERROR PRINT-OUTS:

PRINT-OUTS	DESCRIPTION	PROCEDURE
1. INCORRECT TYPE OF CHNG (TYPE OF CHANGE) (NAME OF GLOSSARY)	Typed in word naming type correction is incorrect.	Type in correct word as in print-out # 1, of normal print-outs.
2. INCORRECT NAME OF GLOSS (TYPE OF CHANGE) (NAME OF GLOSSARY)	Name of glossary not in catalog, and, therefore, the change named cannot be executed.	Restart librarian corrections from the beginning.
3. ALREADY IN CATALOG (INSERT ----) (TYPE OF CHANGE) (NAME OF GLOSSARY)	The glossary being inserted is contained in the catalog.	Type in a new change request, as in normal print-out # 1, or restart the librarian corrections from the beginning.
4. LAST CORRECTION	The limit has been reached on the amount of corrections permitted. The last correction has been put in and spaces have been supplied to indicate the end of corrections.	If further corrections are desired, a new run must be made.
5. INCORRECT ADD- RESS (NAME OF GLOSSARY)	The symbolic address used in the word correction is in error.	Type in correct symbolic address.
6. OUT OF SEQUENCE WITH CORRECTIONS (TYPE OF CHANGE) (NAME OF GLOSSARY)	The new symbolic glossary on the tape on Servo 4 is not in sequence with control list corrections.	Restart librarian corrections from the beginning.
7. CORRECTION WRONG ORDER (BLOCK & WORD)	The block number is wrong because it is <u>less</u> than the previous block correction made.	Type in correct block and word, after block and word type-out.
8. INCORRECT BLOCK AND WORD END GLOSSARY	The block count has extended beyond the end of the glossary being processed.	Restart librarian, or leave Glossary uncorrected by transferring control to word 144 (~ UOOL44)

HOW TO WRITE ARITH-MATIC PSEUDO-CODE

Arith-matic pseudo-code consists of a series of pseudo-instructions which the Arith-matic system will compile into Univac code ready for the computer. Certain beginning sentinels must precede the first pseudo-instruction, and an end sentinel must follow the last one. These will be listed below:

```
XXABLXANKAST
YYYWDSASTA
A00000000000
B00000000000
.....
ENDASTORAGEA
(1st WD OF PSEUDO CODE)
.....
(1st WD OF PSEUDO CODE)
ENDACODINGAA
```

XX is the number of blocks of storage reserved for input and output during the Univac run.

YYY is the number of words of storage reserved in memory for temporary and working storage.

If $YYY = 0$, the system will ignore "n" and will reserve no storage. If $YYY \neq 0$, the system will read the words into the memory from servo "n" until it reaches the sentinel "ENDADATAAAA".

Regardless of how many words are read from tapes, the system will reserve exactly YYY words of memory for working storage.

A, B, etc. are the symbolic labels of all the blocks needed to cover blocks and words of working storage. These labels need not be in alphabetical order. However, they correspond to blocks (XX) and words (YYY) of working

storage in the order in which both are listed.

If (XX) and (YYY) both equal zero, there is no need for block labels. Nevertheless, the words including XX and YYY must appear at the beginning of the pseudo-code followed by "ENDASTORAGEΔ". None of these beginning and end sentinels is given an operation number.

The first pseudo-instruction is called operation #0. The succeeding pseudo-instructions are numbered in order, except that beginning, end and other non-operational sentinels, and operation references in the form of XXCN0000YY00, are not considered operations and are not given operation numbers. Any pseudo-instruction which transfers control must refer to other operations by operation number, by using a pseudo-instruction beginning with XXCN. The XX is determined by the transfer operation itself. All available pseudo-instructions, and the method of writing new ones, are discussed elsewhere in this section.

REPertoire OF ARITH-MATIC PSEUDO-CODE OPERATIONS

ARITHMETIC SUBROUTINES	DESCRIPTIONS
AAO(A)(B)(C)	$A \div B = C$
ASO(A)(B)(C)	$A - B = C$
AMO(A)(B)(C)	$A \times B = C$
ADO(A)(B)(C)	$A \div B = C$

NOTES:

- (1) (A) is the address of A
- (2) (B) is the address of B
- (3) (C) is the address of C

TRIGONOMETRIC SUBROUTINES	DESCRIPTION
TSO(A)OOO(B)	Sin A = B
TCO(A)OOO(B)	Cos A = B
TTO(A)OOO(B)	Tan A = B
TAT(A)OOO(B)	Arctan A = B

NOTES:

- (1) (A) is the address of A, which is the angle to be calculated, expressed in radians.
- (2) (B) is the address of B.

HYPERBOLIC SUBROUTINES	DESCRIPTION
HSO(A)OOO(B)	Sin h A = B
HCO(A)OOO(B)	Cos h A = B
HTO(A)OOO(B)	Tan h A = B

NOTES:

(1) See NOTES #1 and #2 of TRIGONOMETRIC Subroutines.

GENERAL MATHEMATICAL SUBROUTINES	DESCRIPTION
ANI(A)OOO(B)	$-A = B$
EXP(A)OOO(B)	$C^A = B$
'APN(A)(N)(B)	$A^N = B$ ("N" must be integral) $-99 \leq N \leq 99$.
GPN(A) N (B)	$A^N = B$ ("N" must be positive integral)
X+A(N)(log ₁₀ A)(B)	$A^N = B$ (The product, $N \log_{10} A$, must not have an exponent exceeding + 10).
SQR(A)OOO(B)	$\sqrt{A} = B$
RNA(A)(N)(B)	$\sqrt[N]{A} = B$ $-9 \leq N \leq 9$.
GRN(A) N (B)	$\sqrt[N]{A} = B$ ("N" must be positive integral)
LAU(A)(Log ₁₀ B)(C)	$\text{Log}_p A = C$ ("A" must be greater than zero.)

NOTES:

- (1) In the Subroutines TAT, EXP, APN, X+A and RNA the (N) is the address of N.
- (2) In the Subroutines GPN and GRN the "N" is the power itself or the root itself rather than the address of N.
- (3) When coding a problem, using the above pseudo-code, there are a number of decisions which can be made in order to produce a better running program --
 - (a) If the mathematical problem involves raising a number of quantities to certain powers which are integral, it would be more efficient to use APN rather than use GPN since APN will not be repeated within the same segment.
 - (b) Similarly, if it is necessary to take the square root of a number of quantities, then it would be more efficient to use the SQR routine rather than the GRN routine since, as with APN, SQR will not appear more than once within the same segment.
 - (c) Also, if it is found that there are GRN routines which lie entirely within the same segment, it would be preferable to use the RNA subroutine for the same reason as that described in (a) and (b) above with reference to SQR and APN routines.

CONTROL SUBROUTINES	DESCRIPTION
<p><i>old AAL</i> = AAL(X_1)(ΔX)(L_X) O1CNOO(K)OO O2CNOO(N)OO</p>	<p>Add to a limit $X_i + \Delta X \rightarrow X_i$ If $X_i < L_X$ go to opn #K If $X_i \geq L_X$ go to opn #N</p>
<p><i>Like AT L except for neg. incr.</i> ALL(X_1)(ΔX)(L_X) O1CNOO(K)OO O2CNOO(N)OO</p>	<p>Add to a limit: $X_i + \Delta X \rightarrow X_i$ If $X_i \geq L_X$ go to opn #K If $X_i < L_X$ go to opn #N</p>
<p><i>Old AAL except for end pt.</i> ATL(X_1)(ΔX)(L_X) O1CNOO(K)OO O2CNOO(N)OO</p>	<p>Add to a limit: $X_i + \Delta X \rightarrow X_i$ If $X_i \leq L_X$ go to opn #K If $X_i > L_X$ go to opn #N</p>
<p>QUO(A)(B)OOO O1CNOO(K)OO</p>	<p>Algebraic equality test. If $A = B$ go to opn. #K If $A \neq B$ go to next operation.</p>
<p>QUA(A)(B)OOO O1CNOO(K)OO</p>	<p>Absolute equality test. If $A = B$ go to opn #K If $A \neq B$ go to next operation.</p>
<p>QTO(A)(B)OOO O1CNOO(K)OO</p>	<p>Algebraic greater than test. If $A > B$ go to opn #K. If $A \leq B$ go to next operation.</p>
<p>QTA(A)(B)OOO O1CNOO(K)OO</p>	<p>Absolute greater than test. If $A > B$ go to opn. #K If $A \leq B$ go to next operation.</p>
<p>QZO(A)OOOOOO O1CNOO(K)OO</p>	<p>Sentinel test. If A equals a word of Z's, than go to opn. #K. If A is not (#) a word of Z's go to next operation.</p>
<p>UOOOOOOOOOOO O1CNOO(K)OO</p>	<p>Unconditional transfer Go to opn. #K.</p>

CONTROL SUBROUTINES	DESCRIPTION
JTC000000000 01CN00(K)00 02CN00(N)00 03CN00(P)00	Generalized Return Jump See NOTE #1
R00000000000	See NOTE #1

NOTES:

1.

This routine acts as a generalized return jump instruction permitting the sequential execution of any specified portion of coding to be followed by a jump to some specified operation.

Upon entering JTC, control is transferred to operation K. Computation then proceeds to operation N. Control is then transferred to operation P. Operation N must be a special pseudo-operation whose call word is R00. This operation serves figuratively as a spring-board for the jump to operation P. This R00 operation will act as a skip unless it is first set by the execution of a JTC referring to the R00 in its 2CN parameter. An operation involving R00 that has been set in this manner will reset itself upon execution and will on subsequent re-entry, act as a skip until being set again by a JTC. The same R00 operation may be referred to by multiple JTC operations. A JTC operation will set only that R00 operation that is specified by its 2CN parameter and will affect no other R00 operation that may appear in the program.

When using JTC, operations K, N, and P need not be in the same segment.

INPUT-OUTPUT ROUTINES	DESCRIPTION
$\begin{matrix} R \\ GTHWt \\ S \end{matrix} \begin{matrix} B \\ NNNMM \\ F \end{matrix}$	Tape-handling generator (Note #1),
GMIOtXOSSMM	Input-generator (Note #2)
$GMOO \begin{matrix} H \\ X \\ L \end{matrix} SSMM$	Output-generator (Notes #3 and 5).
$FILXO \begin{matrix} H \\ L \end{matrix} t \begin{matrix} Z \\ \Sigma \end{matrix} KKKK$	Ending sentinè fill routine (Notes #4 and 5)

NOTES:

#1. GTH

(R) read, (W) write, or (S) skip (N) blocks
 (B) backward or (F) forward on servo (t), starting in storage location (M).

#2 GMI

Move an (S) word item from input block (X) to working storage beginning at location (M). If the block is exhausted, read the next block from servo (t) into block (X).

#3 GMO

Move an (S) word item from working storage beginning in location (M) to output block (X). When the output block is filled, write the block to servo (t) at (H) high or (L) low density.

#4 FIL

Enter a word of Z's or Σ 's (if output is edited) as the next item in output block (X). Write the block to servo (t) at (H) high or (L) low density. (K) refers to the operation number of the associated GMO.

#5

Each GMO and its associated FIL must agree in servo (t), block (X), and density (H/L) parameters.

EDIT ROUTINES	DESCRIPTION
EDF(m ₁)(C)(NN)(m ₂)	Exponential edit for high speed printer SEE NOTES #1, #2, #3.
EDT(m ₁)(C)(NN)(m ₂)	Conversion and Edit for high speed printer SEE NOTES #1, 2, and #4.
EDU(m ₁)(C)(NN)(m ₂)	Exponential edit for uniprinter. SEE NOTES 1, 2, and 3.

#1 "C" is the number of columns the coder desires in the printed form. If the number of columns is greater than five, a space should be used in this position. If a zero is used, the values will be edited, but they will not be put into columns.

#2 "NN" values (2 words to each value) are picked up, starting from memory position "m₁" edited and deposited, starting in memory position "m₂".

#3 Edited form for EDF and EDU is = ±.XXXXXXXXXXΔ(±XXX)ΔΔΔΔ

#4 Edited form for EDT is =
 For exponent of 0 thru -10 (i.e., -4)
±.0000XXXXXXXXXX0000.
 For exponent of 1 thru +10 (i.e., +4)
±XXXX.XXXXXXXXXX00000000.

GENERAL ROUTINES	DESCRIPTION
GMM(m ₁)OSS(m ₂)	Move generator See note #1.
YTO(XXX)(YYY)(ZZZ)	Print-out Subroutine
BTI(XXX)(YYY)OOO	Type-in Subroutine.
RWS(tape numbers)	Rewind tapes and stop.

NOTE: #1

GMM will move SS words from Memory position "m₁" to memory position "m₂".

NON-OPERATIONAL SENTINELS	DESCRIPTION
XXΔBLKΔWKΔST YYYΔWDSΔSTAN	Must be first two words of pseudo-code (Note 3)
BEGINLOOPXXX	Indicates start of iterative loop. (Notes 1 and 2)
ENDΔLOOPΔXXX	Indicates end of iterative loop (Notes 1 and 2)
SEGMENTΔΔΔΔΔ	Indicates that a new segment is to be begun with the next pseudo-code opera- tion.
ENDΔCODINGΔΔ	Indicates end of Arith-matic pseudo- code.

NOTES:

- #1 In any problem involving an iterative loop that is traversed many times, it is desirable, when possible from the standpoint of eliminating waste-ful tape motion, to keep all of the operations which constitute such a loop within the same segment. This purpose is accomplished in Arith-matic by the sentinel "BEGINLOOPXXX" preceding the first operation con-stituting the loop and "ENDΔLOOPΔXXX" following the last operation in the loop.
- #2 XXX represents the loop number. Loop sentinels must be used in pairs. Elements of a pair are identified by having the same loop number.
- #3 $0 \leq X \leq 10$
 $0 \leq Y \leq 999$
 $N \neq 1$ (servo number)

USE OF UNIVAC C-10 CODE WITH THE ARITH-MATIC COMPILER

It is possible to include actual Univac C-10 instructions along with pseudo-code in an arithmetic program. These detailed machine instructions called "own code" are used to perform operations not covered by the existing subroutines.

Each set of C-10 instructions required to perform a necessary operation has one operation number. The operation consists of three parts:

(1) A word, "OWN Δ CODE Δ XXX", indicating to the compiler that the following lines are C-10 coding. The form of the word is fixed except for the last three positions which contain the number of lines of C-10 coding to be executed.

(2) The actual C-10 instructions. All addresses can be written in absolute form or in relative form. If there is an "M" in the third digit position of the instruction, it is possible to refer to a line within the C-10 coding. If the own code section is written in relative form, such as, in line one of example, "EOACOO" refers to the second word of the "A" block or location.

Transfer of control from the "own code" section to other operations in the program is accomplished by writing 1CN, 2CN, 3CN, etc., in the place reserved for the three digit address as shown in example line. A word of skips must follow this type of Q, T or U instruction and also a skip instruction must precede this type of instruction.

(3) The end sentinel for the "end code" section is "END Δ OWN Δ XXXX" where "XXXX" is the operation number of the own code section.

Following this, any other operation numbers referred to in the C-10 coding by 1CN, 2CN, 3CN, etc., are referenced in the form:

01CN Δ XXXX00

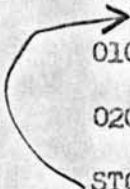
02CN Δ XXXX00

XXXX is the operation number. . . .

The pseudo instruction STORAGEOOZZZ following ENDΔOWNΔXXXX will allocate ZZZ lines of fixed storage (not disturbed by segmentation) for the use of the own code section. These locations are referred to within the section as 00ST, 01ST, CTC. ?

EXAMPLE:

```
OWNΔCODEΔ008
001 BOA000LOBO02
002 ~~~~~ Q002CN
003 ~~~~~
004 BCM001A0M008
005 HOM001C000ST
006 ~~~~~ U001CN
007 ~~~~~
008 -----1-----
ENDΔOWNΔXXXX
01CN00XXXX00
02CN00XXXX00
STORAGE00001
```



LIST OF ARITH-MATIC CONSTANTS

180	R00184 U00181	201	050 000 000 000
181	C00185 B00184	202	099 999 999 999
182	A-0197 C00184	203	000 000 000 060
183	B00185 000000	204	016 666 666 667
184	000000 U00111	205	041 666 666 667
185	000000 000000	206	083 333 333 333
186	010000 C00000	207	013 888 888 889
187	000000 000001	208	019 841 269 841
188	020000 000000	209	024 801 587 302
189	000000 000001	210	000 000 U00 000
190	040000 000000	211	027 557 319 224
191	000000 000000	212	078 539 816 340
192	-000000 000000	213	015 915 494 309
193	010000 000000	214	025 052 108 385
194	-100000 000000	215	020 876 756 988
195	001000 000000	216	000 111 000 000
196	000001 000000	217	000 000 000 111
197	000000 000001	218	043 429 448 190
198	000001 000001	219	000 000 000 000
199	020000 000000		
200	Z7ZZZZ Z7ZZZZ		

RULES FOR SUBROUTINE CONSTRUCTION

The subroutine is the logical unit or building block. It operates in a specific way on input data to produce results. The compiler forces certain limitations on the form of subroutines.

The requirements for subroutines to be used with the A-2 Compiler are stated below.

1. Subroutines are written in relative coding starting in line 000 of the subroutine which contains the call-word. The call-word consists of four 3-digit fields.

XXX 111 F00 00X

XXX The identification code (alphabetic) of the subroutine.
111 The number of lines in the subroutine, exclusive of the call-word and sentinel word (word of ignores) at the end.
F00 A floating decimal point subroutine.
00X A sentinel which serves to identify the start of a subroutine.

2. Line 001 of the subroutine is the only entrance (exclusive of R U or generalized overflow re-entries).
3. In referring to the input to the subroutine, 1RG, 2RG, etc. are used instead of memory locations; i.e., B 1RG. The output of the subroutine; the results, are referred by 1RS, 2RS, etc.
4. Normal exit from a subroutine is considered to occur when the line preceding the ending sentinel is the last line in the subroutine to be executed, or when control is transferred from any line in the subroutine to the ending sentinel. Under these conditions the next line to be executed will be the entrance line of the next compiled subroutine.
5. Any exit from the subroutine other than one of the types described above (4), is called a controlled exit and is written T 1CN or U 2CN, etc. A skip must precede such instructions (i.e., 00000 Q 3CN), and a word of skips must follow such instructions.
6. References to other lines within the subroutine require the use of an "M" in the digit position (digit 3 or 9) preceding the 3-digit relative line number.
7. A set of the most used constants will be in the memory in locations 180-219 when the problem is being run, and these may be referred to by fixed memory locations (see list of Permanent Constants). Constants not on this list must be included in the subroutine.
8. There are ten temporary storage locations in 170-179 which may be used by any subroutine.
- 8a. If it is desired to use 00ST, 01ST in the subroutine, then the number of these storage positions used by the subroutine must be placed in the ninth, and tenth digit position of the call-word.

9. The floating decimal arithmetic routines are fixed in the memory locations 60-119. Any subroutine can use this section by placing A in 170, 171 and B in 172, 173 and by selecting the proper R U instruction. A and B are floating-decimal operands with integers in even, and exponents in odd-numbered locations.

A + B	C	R	90	U	60
A - B	C	R	90	U	108
A x B	C	R	90	U	127
A ÷ B	C	R	90	U	110

C is the result in 174, 175.

10. This section is also available to normalize numbers if the operand is placed in 174, 175, and the instruction R 90 U 159 is given. The result will be placed in 174, 175.
11. All subroutines must consist of an even number of lines excluding call-number and ending sentinel.

Example: (non-functional)

000	XYZ012	F00001	Call-word
001	V 1RG	W 890	1RG = A
002	V 3RG	W 892	3RG = B
003	R 810	U 847	A x B = C
004	V 894	W 1RS	1RS = C
005	B 894	L 911	
<u>006</u>	<u>00 000</u>	<u>T 1CN</u>	if C > 0 → Controlled exit
007	A=MO10	HOM011	Internal subroutine references
<u>008</u>	<u>V 894</u>	<u>OOM012</u>	
<u>009</u>	<u>W 1RG</u>	<u>UOM013</u>	Normal exit
010	000000	000009	Constant
011	[]	Temporary storage
012	W 3RS	00 000	Normal exit
013	####	####	Ending sentinel

OPERATING INSTRUCTIONS FOR MATH-MATIC

Mount tapes on the following servos:

<u>SERVO</u>	<u>TAPE</u>
1	MATH-MATIC MASTER
2	BLANK
3	BLANK
4	BLANK
5	BLANK
6	MATH-MATIC LIBRARY
7	BLANK
8	INPUT MATH-MATIC PSEUDO CODE
9	BLANK

No Breakpoints

Block Sub-divide on Servo -2

Initial Read Taps 1

During the MATH-MATIC run instructions will be typed out on the supervisory control printer specifying which servos tapes are to be mounted on during the problem run.

Tape 4 will be rewound with interlock and "REMOVE SERVO-4 MOUNT BLANK" will be printed out.

This tape contains data that was created from the pseudo-code to be used in the problem run. Remove this tape and save. Mount a blank on Servo 4.

After the print out "END△COMPILE△", all tapes will rewind. Remove tape from Servo 2 for printing. This tape contains the edited record. Remove the tape from servo 7 and mount it on Servo 1. This tape contains the running program. Mount the tape removed from servo 4 on the servo specified in the Working storage print-out. Mount any other tapes, as specified in the SCP print-outs. Block sub-dkvide on output servos. Initial read servo 1.

SAMPLE PROBLEM 1

$$n = \frac{e^{X\left[\frac{1-y}{y}\right]} - 1}{\frac{e^{X\left[\frac{1-y}{y}\right]} - 1}{y}} \quad (1)$$

For equation (1), take values of y from 0.01 to 0.99 in steps of 0.01 (i.e., 0.01, 0.02, 0.03, etc.) and values of X from 0.025 to 2.00 in steps of 0.025 (i.e., 0.025, 0.050, 0.075, etc.).

$$n = \frac{y[e^{X\left[\frac{y-1}{y}\right]} - 1]}{ye^{X\left[\frac{y-1}{y}\right]} - 1} \quad (2)$$

For equation (2), take values of y from 1.01 to 2.00 in steps of 0.01 (i.e., 1.01, 1.02, 1.03, etc.) and values of X the same as before.

MATH-MATIC CODE

- (1) $\Delta VARY \Delta Y \Delta 0.01 \Delta (0.01) \Delta 0.99 \Delta SENTENCES \Delta 2 \Delta THRU \Delta 4 \Delta.$
- (2) $\Delta VARY \Delta X \Delta 0.025 \Delta (0.025) \Delta 2 \Delta SENTENCES \Delta 3 \Delta THRU \Delta 4 \Delta.$
- (3) $\Delta N \Delta = \Delta (\text{EXP} \Delta (X * (1 - Y) / Y) - 1) / (\text{EXP} \Delta (X * (1 - Y) / Y) / Y - 1) \Delta.$
- (4) $\Delta EDIT \Delta AND \Delta WRITE \Delta N, Y, X \Delta.$
- (5) $\Delta VARY \Delta Y \Delta 1.01 \Delta (0.01) \Delta 2 \Delta SENTENCES \Delta 6 \Delta THRU \Delta 8 \Delta.$
- (6) $\Delta VARY \Delta X \Delta 0.025 \Delta (0.025) \Delta 2 \Delta SENTENCES \Delta 7 \Delta THRU \Delta 8 \Delta.$
- (7) $\Delta N \Delta = \Delta Y * (\text{EXP} \Delta (X * (Y - 1) / Y) - 1) / (Y * \text{EXP} \Delta (X * (Y - 1) / Y) - 1) \Delta.$
- (8) $\Delta EXECUTE \Delta SENTENCE \Delta 4 \Delta.$
- (9) $\Delta STOP \Delta.$

ARITH-MATIC PSEUDO CODE PRODUCED BY THE TRANSLATOR

	01A8LKAWKAST	28	A1B02B08B10
	044AWDSOWKA3		01CN00001600
	A00000000000		02CN00002900
	B00000000000		ENDALOOPA002
	ENDASTORAGEA	29	F1LA0H2E00L1
	BEGINLOOP004	30	RWS123000000
0	GMMB06002B02		ENDACODING
	BEGINLOOP003		
1	GMMB12002B004		
2	ASOB18B02B32		
3	AMOB04B32B34		
4	ADOB34B02B36		
5	EXOB36000B38		
6	ASOB38B18B40		
7	ADOB38B02B42		
8	ASOB42B18B42		
9	ADOB40B42B00		
	BEGINLOOP000		
10	EDFB000A3A50		
11	GMOO2AH10A50		
12	R00000000000		
13	A+LEQ4B14B16		
	01CN00000200		
	02CN00001400		
	ENDALOOPA003		
14	A+LB02B08B10		
	01CN00000100		
	02CN00001500		
	ENDALOOPA004		
	BEGINLOOP002		
15	GMMB06002B02		
	BEGINLOOP001		
16	GMMB12002B04		
17	ASOB02B18B32		
18	AMOB04B32B34		
19	ADOB34B02B36		
20	EXOB36000B38		
21	ASOB38B18B40		
22	AMOB02B38B42		
23	ASOB42B18B42		
24	AMOB02B40B40		
25	ADOB40B42B00		
26	JTC000000000		
	01CN00001000		
	02CN00001200		
	03CN00002700		
	ENDALOOPA000		
27	A+LEQ4E14B16		
	01CN00001700		
	02CN00002800		
	ENDALOOPA001		

SAMPLE PROBLEM 2

SOLVE:

$$Y = \frac{X^3(2+X)}{3 \cos A} - \sqrt[4]{3P}$$

FOR $0.2 \leq P \leq 0.8$ where $\Delta P = 0.2$

$0.35 \leq A \leq 1.5$ where $\Delta A = 0.175$

$1.8 \leq X \leq 3.8$ where $\Delta X = 0.5$

MATH-MATIC PSEUDO CODE

- (1) $\Delta VARY \Delta P \Delta 0.2 \Delta (0.2) \Delta 0.8 \Delta SENTENCES \Delta 2 \Delta THRU \Delta 5 \Delta$
- (2) $\Delta VARY \Delta A \Delta 0.35 \Delta (0.175) \Delta 1.05 \Delta SENTENCES \Delta 3 \Delta THRU \Delta 5 \Delta$
- (3) $\Delta VARY \Delta X \Delta 1.8 \Delta (0.5) \Delta 3.8 \Delta SENTENCES \Delta 4 \Delta THRU \Delta 5 \Delta$
- (4) $\Delta Y \Delta = \Delta X^3 * (2 + X) / (3 * \cos \Delta A) - \Delta \sqrt[4]{\Delta 3 * P} \Delta$
- (5) $\Delta EDIT \Delta AND \Delta WRITE \Delta Y, X, A, P \Delta$
- (6) $\Delta STOP \Delta$

ARITH-MATIC PSEUDO CODE PRODUCED BY THE TRANSLATOR

	01ABLKANKAST	14	ATLBO2B22B24
	038ANDSANKA3		01CN00000300
	A00000000000		02CN00001500
	B00000000000		ENDAL <u>LOOP</u> A000
	ENDASTORAGEA	15	A+LB04B16B18
	BEGIN <u>LOOP</u> 002		01CN00000200
0	GMMB08002B06		02CN00001600
	BEGIN <u>LOOP</u> 001		ENDAL <u>LOOP</u> A001
1	GMMB14002B04	16	A+LB06B10B12
	BEGIN <u>LOOP</u> 000		01CN00000100
2	GMMB20002B02		02CN00001700
3	AAOB26B02B30		ENDAL <u>LOOP</u> A002
4	TCOB04000B32	17	FTIA OH2Σ0013
5	AMOB28B32B32	18	RWS123000000
6	AMOB28B06B34		ENDACODING
7	GFNB02003B36		
8	GRNB34000B34		
9	AMOB36B30B30		
10	ADOB30B32B30		
11	ASOB30B34B00		
12	EDFB00A04A50		
13	GMQ002AH10A50		

MATH-MATIC PRINT-OUTS

The error print-outs which may occur throughout all the phases of translation and compilation or in the problem run are listed in this section. The exact form of each print-out is given in the left-hand column, followed in the middle column by a description of the error which caused the print-out.

Before following any of the alternative procedures listed in the third column, it is advisable to examine the pseudo-code to ascertain whether or not other portions of the problem will be affected by any changes made.

Phase 1 Error Print-outs

Normal Print-outs;
END PHASE 01

<u>ERROR PRINT-OUT</u>	<u>DESCRIPTION</u>	<u>PROCEDURE</u>
1. <u>NO LEFT PAREN</u>	<u>FIRST</u> Sentence of Pseudo-Code has no left parentheses.	HIT START BAR TO CONTINUE
2. <u>NO RIGHT PAREN</u>	<u>FIRST</u> Sentence of Pseudo-Code has no right parentheses.	HIT START BAR TO CONTINUE
3. <u>SENT.ΔNO.nnn</u> <u>NON-NUMERIC TYPE-</u> <u>IN CORRECTION</u>	Sentence Number must be Numeric. Alphabetic are used only as Appendages for insertion purposes.	TYPE-IN NEW SENTENCE NUMBER. FORM: XΔ-----Δ
4. <u>SENT.ΔNO.nnn</u> <u>NOT IN ORDER TYPE-</u> <u>IN CORRECTION</u>	Sentence Number is less than previous Sentence Number.	TYPE-IN NEW SENTENCE NUMBER. FORM: XΔ-----Δ
5. <u>SENT.ΔNO.nnn</u> <u>NO SPACE AFTER</u> <u>PAREN.</u>	Space missing after the parentheses following the Sentence Number.	HIT START BAR. TO CONTINUE
6. <u>MORE THAN 100</u> <u>SENTENCES RE-</u> <u>WRITE</u>	Limit of Input Sentences is 100.	HIT START BAR TO REWIND
7. <u>SENT.ΔNO.nnn</u> <u>NO SPACE BE-</u> <u>FORE EQUAL.</u>	Δ = Δ CORRECT FORMAT	HIT START BAR. TO CONTINUE
8. <u>SENT.ΔNO.nnn</u> <u>NO SPACE AFTER</u> <u>EQUAL</u>	Δ = Δ CORRECT FORMAT.	HIT START BAR. TO CONTINUE
9. <u>SENT.ΔNO.nnn</u> <u>PSEUDO-WORD</u> <u>(pseudowords)</u> <u>TOO LONG RE-</u> <u>WRITE</u>	Pseudo-word longer than 12 digits.	Rewrite. HIT START BAR. TO REWIND SERVOS.

ERROR PRINT-OUT	DESCRIPTION	PROCEDURE
10. <u>SENT.ΔNO.nnn</u> <u>PUNCTUATION IN-</u> <u>CORRECT (pseudo-</u> <u>word) TYPE-IN</u> <u>CORRECTION</u>	Decimal point before alpha- betic Pseudo-word.	TYPE-IN CORRECT PSEUDO- WORD FORM: XΔ-----Δ.
11. <u>SENT.ΔNO.nnn</u> <u>TOO LONG RE-</u> <u>WRITE</u>	Sentence longer than 230 words. (<i>pseudowords</i>)	REWIND. HIT START BAR TO REWIND TAPES.
12. <u>SENT.ΔNO.nnn</u> <u>PSEUDO-WORD IN-</u> <u>CORRECT (pseudo-</u> <u>word-1st) TYPE-</u> <u>IN CORRECTIONS.</u> <u>(2nd pseudo-word)</u>	The <u>functional</u> call-word printed out is not in the Math-matic catalog of call-words <i>3 ROOT</i>	IF <u>FIRST</u> , WORD IS IN- CORRECT TYPE-IN NEW WORD AND THEN A WORD OF SPACES. FORM: XΔ-----Δ Δ-----Δ IF SECOND WORD IS IN- CORRECT, TYPE SPACES IN FIRST WORD. IN THE SECOND WORD TYPE NEW WORD. FORM: ^^^^^^^^^^^ XΔ-----Δ
13. <u>SENT.ΔNO.nnn</u> <u>PSEUDO-WORD IN-</u> <u>CORRECT (pseudo-</u> <u>word) TYPE-IN</u> <u>CORRECTION.</u>	The <u>statement</u> call-word is not in the Math-matic catalog of call-words.	TYPE-IN CORRECTION. FORM: XΔ-----Δ.
14. <u>SENT.ΔNO.nnn</u> <u>PARENS INCORRECT-</u> <u>LY PAIRED REWRITE</u>	The number of open paren- theses differs from the number of closed paren- theses.	REWRITE. HIT START BAR TO REWIND TAPES.
15. <u>SENT.ΔNO.nnn</u> <u>ABSOLUTES INCORR-</u> <u>ECTLY PAIRED RE-</u> <u>WRITE.</u>	There are an odd number of absolute signs in the sen- tence.	REWRITE. HIT START BAR TO REWIND SERVOS.
16. <u>SENT.ΔNO.nnn</u> <u>(pseudo words)</u> <u>SPACE PERIOD</u> <u>WITH NO LEFT</u> <u>PAREN.</u>	A space-period signifies the end of the sentence and should be followed by spaces or a left par- entheses to signify the start of the next sen- tence. The user may not have wished to end the sentence. 106	HIT THE START BAR TO BE- GIN PROCESSING A NEW SEN- TENCE. SET BREAKPOINT 8 AND FORCE NO TRANSFER TO CONTINUE PROCESSING THE SAME SENTENCE.

ERROR PRINT-OUT	DESCRIPTION	PROCEDURE
16. (Cont'd)	Therefore, he has a break-point option.	
17. <u>WRONG TAPE ON SERVO-6 MOUNT LIBRARY</u>	The wrong tape was mounted on servo six.	CHANGE THE TAPE. -- RE- WIND ALL TAPES -- START MATH-MATIC AGAIN.

PHASE II - Normal Printouts

1. END PHASE 2

PHASE II - Error Printouts

1.	ERROR PRINT-OUT	DESCRIPTION	PROCEDURE
	SENT.ΔNO. nnn TOO MANY FLOATADEC.0- REWRITE".	There are more than 100 numbers expressed in "power of ten" form in the pseudo-code. This count does not include the numbers in the "VARY (LIST)" instruction.	Rewrite the problem, keeping the number of arguments expressed in "power of ten" form within the limit of 100. (Hit the start bar to rewind all servos.)
2.	"SENT.ΔNO. nnn TOOMANYΔΔΔΔ OPERATIONS REWRITE".	More than 100 partial results have been created for the equation being processed.	Rewrite the problem. Break the equation up into smaller ones. (Hit the start bar to rewind all servos.)
3.	"FIRST ARGUMENT MISSING SENT.ΔNO. nnn REWRITE".	The desired entry from Sentence File 1 is not there.	Rewrite pseudo-code. (Hit the start bar to rewind servos.)
4.	"SECOND ARGU- MENT MISSING SENT.ΔNO. nnn REWRITE".	The desired entry from Sentence File 1 is not there.	Rewrite pseudo-code. (Hit start bar to rewind servos.)
5.	PSEUDOWORDS MISSINGΔΔΔΔ SENT.ΔNC. nnn REWRITEΔΔΔΔ	Too many words have been requested from Sentence File 1.	Rewrite. (Hit start bar to rewind servos.)
6.	SENT.ΔNO. nnn TOOMANY LOOPS REWRITE.	The pseudo-code calls for more than 100 loops.	Rewrite the pseudo-code (Hit the start bar to rewind all servos.)

PHASE II - Error Printouts (Cont'd)

ERROR PRINT-OUT	DESCRIPTION	PROCEDURE
7. SENT.ΔNO.n.. READ ITEM SIZE (NUMBER) NOT ALLOWED TYPE-IN CORRECTED NUMBER	The number of variables listed in the sentence is not acceptable.	If the input data has been appropriately padded. Type in number of words in whole item including padding. FORM: XXΔ-----Δ -otherwise rewrite.
8. SENT.ΔNO.n.. READ TOO MANY VARIABLES REWRITE	More than 30 variables have been listed in the Sentence.	Rewrite pseudo-code. (Hit Start bar to rewind tapes).
9. SENT.ΔNO.n.. READ SENTENCE REFERENCE MISSING TYPE-IN	The stated sentence reference for the jump option does not conform to pseudo-code format.	Type-in the sentence reference FORM: FΔ-----Δ.
10. SENT.ΔNO.n.. READ INCORRECT FORMAT TYPE-IN	The Sentence does not conform to pseudo-code format.	Type-in the entire Sentence omitting the word read. Type-in one word at a time. FORM: XΔ-----Δ When finished, type-in a word of spaces.
11. SENT.ΔNO.... WORD MISSING AFTER-- AND.ΔΔ TYPE-INΔΔΔΔΔ.	The word "AND" is not followed by "WRITE", indicating either unintentional omission of the word "WRITE", or incomplete deletion.	If WRITE is desired, type-in "WRITEΔΔΔΔΔΔΔΔ". If WRITE is not desired, type-in "ΔΔΔΔΔΔΔΔΔΔΔΔΔΔ".
12. SENT.ΔNOΔ.... TYPE-INΔΔΔΔΔ CORRECTLY.ΔΔ	Incorrect procedure has been followed, regarding Print-outs 1 or 3. Wrong word has been typed-out.	Same as for Print-out 1 or Print-out 3, whichever immediately preceded this print-out.

PHASE II - Error Printouts (Cont'd)

ERROR PRINT-OUT	DESCRIPTION	PROCEDURE
13. SENT.ΔNO.nnn WORD ΔMISSING AFTER- FOR.ΔΔ TYPE-INΔΔΔΔΔ	The word "FOR" is not followed by "UNIPRINTER", indicating an omission in pseudo-code. It can not be determined whether Uniprinter, High-speed Printer, or High-speed Printer converted editing is desired.	If Uniprinter is desired, type-in "UNIPRINTERΔΔ". If High-speed Printer is desired, type-in "HSPΔΔΔΔΔΔΔΔ". If CONVERTED is desired, type-in "CONVERTEDΔΔΔ".
14. SENT.ΔNO.nnn TOO ΔMANYΔΔΔΔ VARIABLES.ΔΔ REWRITEΔΔΔΔΔ	More than 30 variables are involved in an Edit Statement or Write Statement.	Rewrite. (Hit start-bar to rewind all tapes).
15. SENT.ΔNO.nnn WORD ΔMISSING AFTER- AND.ΔΔ TYPE-INΔΔΔΔΔ	The word "AND" is not followed by "EDIT", indicating either unintentional omission of the word "EDIT", or incomplete deletion.	If "EDIT" is desired, type-in "EDITΔΔΔΔΔΔΔΔ". If "EDIT" is not desired, type in "ΔΔΔΔΔΔΔΔΔΔ".
16. SENT.ΔNOΔnnn ABSOLUTE SIGN MISSING (PSEUDO-WORD)	The left absolute sign has been omitted from pseudo-code.	To continue with the variable not in absolute form hit the start bar. Otherwise rewrite.
17. SENT.ΔNO.nnn ABSOLUTE VALUE MUST BE ALPHABETIC (PSEUDO-WORD) TYPE-IN	The absolute value of a numeric instead of an alphabetic was specified in pseudo-code.	Type in alphabetic variables. FORM: XΔ-----Δ or rewrite.
18. SENT.ΔNO.nnn INCORRECTLY STATED TYPE-IN SENTENCE RANGE.	The stated sentence range does not conform to pseudo-code format.	Type-in Sentence range -- F and L. FORM: FΔ-----Δ LΔ-----Δ
19. SENT.ΔNO.nnn TOO MANY VAR- IABLES REWRITE	More than 3 variables have been listed in the Sentence.	Rewrite. (Hit the Start-bar and all tapes will rewind).

PHASE II - Error Printouts (Cont'd)

ERROR PRINT-OUT	DESCRIPTION	PROCEDURE
20. SENT.ΔNO.nnn ABSOLUTE SIGN MISSING. (PSEUDO-WORD)	The left Absolute sign has been omitted from pseudo-code.	To continue with the variable not in Ab- solute form, hit the Start bar. Otherwise rewrite.
21. SENT.ΔNO.nnn ABSOLUTE VALUE MUST BE ALPHA- BETIC (PSEUDO- WORD) TYPE-IN	The absolute value of a numeric instead of an alphabetic was spec- ified in pseudo-code.	Type-in alphabetic variables. FORM: XΔ-----Δ or rewrite.
22. SENT.ΔNO.nnn INCORRECTLY STATED TYPE-IN SENT. RANGE	The Stated Sentence range does not conform to pseudo-code format	Type-in Sentence FORM: FΔ-----Δ LΔ-----Δ
23. SENT.NO.nnn "VARY" TOO MANY VARIABLES REWRITE	More than ten var- iables were listed in this Sentence.	Hit Start bar and all tape will re- wind.
24. SENT.NO.nnn VARY (PSEUDO- WORD) INCORRECT FORMAT	The glossary has finished processing the values of the variables, and ex- pects the typed- out pseudo word to be "Sentences."	If all the values of the variables have been processed, hit the start bar to begin the search for the sentence range --- Otherwise rewrite.
25. SENT.NO.nnn VARY INCORRECT NUMBER OF VALUES FOR GIVEN NUMBER OF VARIABLES REWRITE.	The number of values is not divisible by the number of var- iables.	Hit the start bar and all tape will rewind.
26. SENT.NO.nnn VARY INCORRECT FORMAT TYPE-IN SENTENCE RANGE	The stated sentence range does not con- form to pseudo-code format.	Type-in sentence range FORM: FΔ-----Δ LΔ-----Δ

PHASE II - Error Printouts (Cont'd)

ERROR PRINT-OUT	DESCRIPTION	PROCEDURE
27. SENT.ΔNO.nnn VARY INCORRECT FORMAT TYPE-IN SECOND SENTENCE OF RANGE.	The stated Sentence range does not conform to pseudo-code format.	Type-in the sentence number for L. FORM: LΔ-----Δ
28. SENT.ΔNO.nnn JUMP SENTENCE REFERENCE OMITTED TYPE-IN CORREC- TION	The Sentence number to which control is to be transferred has either been omitted or stated incorrectly.	Type-in the sentence reference. FORM: FΔ-----Δ
29. SENT.ΔNO.nnn EXECUTE SENT RANGE IN-- CORRECT TYPE-- IN	The stated Sentence range does not conform to pseudo-code format.	If there are 2 sentences in the range Type-in FΔ-----Δ LΔ-----Δ If there is only one sentence, type spaces in the second word.
30. SENT.ΔNO.nnn COMPILER IN- CORRECT TYPE-- IN CORRECTED STORAGE AND SERVO NO.	The word input or output is not followed by storage and servo allocation in proper pseudo-code format.	Type-in storage block and servo number. FORM: STORAGE-XAAA SERVO-XAAAA If storage and servo information is not included in the sentence, type-in a word of spaces.
31. SENT.ΔNO.nnn COMPILER LABEL MISSING TYPE-IN	The compiler section is not labeled correctly EXAMPLE: COMPILER-2 label must match label assigned to the compiler section.	Type-in the correct label FORM: XΔ-----Δ a 1, 2 or 3, digit label may be typed-in.

PHASE II - Error Printouts (Cont'd)

ERROR PRINT-OUT	DESCRIPTION	PROCEDURE
<p>32. SENT.ΔNO.nnn COMPUTER IN- CORRECT TYPE- IN CORRECTED STORAGE AND SERVO NO.</p>	<p>The word input or out- put is not followed by storage and servo allo- cation in proper pseudo- code format.</p>	<p>Type-in the storage block and servo no. FORM: STORAGE-XAAA SERVO-XXXXXX If storage and servo information is not included in the com- puter section, type in a word of spaces.</p>
<p>33. SENT.ΔNO.nnn COMPUTER LABEL MISSING TYPE-IN</p>	<p>The computer section is not labeled correct- ly. EXAMPLE: COMPUTER=2 label must match label assigned to the computer sec- tion.</p>	<p>Type-in the correct label. FORM: XΔ-----Δ a 1, 2 or #, digit label may be typed- in.</p>
<p>34. LEFT PAREN- THESES SENT.ΔNO.nnn MISSING REWRITE.</p>	<p>No left Parentheses in equation.</p>	<p>Rewrite pseudo-code. (Hit start bar to re- wind all tapes).</p>
<p>35. OPERATION missing SENT.ΔNO.nnn REWRITE.</p>	<p>Binary operation missing in equations.</p>	<p>Rewrite pseudo-code (Hit start bar to re- wind all tapes).</p>

PHASE III NORMAL PRINT-OUTS

1. SENT.ΔNO.nnn
(Sentence Name)
STORAGEΔΔXnn
SERVO-nΔΔΔΔΔ

2. ΔΔΔΔΔΔΔΔΔΔ
WKG.ΔSTG.ΔΔΔ
STORAGEΔΔXnn
SERVO-nΔΔΔΔΔ

For every input or output storage and servo allocation during Phase III, print-out No. 1 is typed on the Supervisory Control Printer. For each block of working storage assigned, print-out No. 2 appears.

3. REMOVEΔΔΔΔΔΔ
SERVO-ΔΔΔΔΔΔ
MOUNTΔBLANKΔ

Data has been prepared by the Math-Matic System and must be mounted during the Univac run on the Servo specified in Working Storage print-out. A blank must be mounted on Servo-4 for the Arith-Matic run.

4. END PHASE 3

PHASE VII ERROR PRINT-OUTS

<u>PRINT-OUT</u>	<u>DESCRIPTION</u>	<u>PROCEDURE</u>
1. TOOΔMUCHΔWSΔ	The number of variables and constants to be stored in working storage has exceeded the limits of the storage blocks available for allocation, or the number of blocks to be allocated for input or output statements has exceeded the number of blocks available	Re-examine problem so that it can be broken down into shorter problems. Rewind all tapes. Rewrite.
2. TOOΔMNYΔSRVO	The number of Servos to be assigned has exceeded the number available.	Re-examine problem so that it can be broken down into shorter problems. Rewind all tapes. Rewrite.

PRINT-OUT	DESCRIPTION	PROCEDURE
3. SL2QATQQALNG	The number of variables and constants to be stored in the output area of Storage List 2 has exceeded the available space.	Re-examine problem so that it can be broken down into shorter problems. Rewind all tapes. Rewrite.
4. SL2IATQQALNG	The number of variables and constants to be stored in the Input area of Storage List 2 has exceeded the available space.	Re-examine the problem so that it can be broken down into shorter problems. Rewind all tapes. Rewrite.
5. SENT.ANO. nnn NOAWRITEAAAA	An edit statement has been written without a corresponding write statement.	Rewrite.
6. VARMISSINGA	A literal variable has appeared on the right side of an equation or in a statement without having been previously defined.	Rewrite.

*Prints out "Equation"
should give no.*

PHASE IV Normal Print-outs:

1. END AT-3

PHASE IV Error Print-outs:

<u>ERROR PRINT-OUT</u>	<u>DESCRIPTION</u>	<u>PROCEDURE</u>
1. <u>TOO MANY OPERATIONS. REWRITE!</u>	More than 30 redundant operations within a single equation.	REWRITE. (HIT START BAR TO REWIND ALL TAPES).
2. <u>TOO MANY PARTIAL RESULTS. REWRITE!</u>	More than 100 operations within a single equation.	AS ABOVE.
3. <u>COMP. SECTION NOT STORED. REWRITE!</u>	The Computer or Compiler Section cannot be located on Tape on Servo 8.	AS ABOVE.
4. <u>"NO SUCH SENT. SENT. ΔNO. nnn"</u>	No such sentence could be found in Pseudo-code.	REWIND TAPES MANUALLY.

ARITH-MATIC PRINT-OUTS

The error print-outs contained in this section can occur only when the coder uses Univac code or Arith-matic pseudo-code (computer or compiler sections) directly. The Arith-matic operations produced by the translator from Math-matic sentences are of course, correct, and will not contain any of the errors discussed herein.

SWEEP I NORMAL PRINT-OUT:

END SWEEP I

SWEEP I ERROR PRINT-OUTS:

PRINT-OUT	DESCRIPTION	PROCEDURE
1. ENDAOWNAXXX SHOULD BE ENDAOWNAYYY	Operation Number of own code section XXXX is not equal to YYY or ENDAOWN is written incorrectly.	If the number of operations of pseudo-code are miscounted, then recompiled. If, however, ENDAOWN was written improperly or if the error was in the number "XXXX", and not in miscounting the operations, then hit start bar and type in ENDAOWNAYYY.
2. AAA()() NOT STORED	The first three alphabetic characters of pseudo-instruction have been improperly written, and, thus, it is not found in catalogue	Make correction of pseudo-word and recompile.
3. ERROR IN SERVO SPEC. (GMIOtXOSSMM) (GTHt ^f NNNMM) S O	The fifth position which contains the servo number, which will be the input for the running program, cannot be one since the running program will be on servo one.	Type in different servo number. FORM: 0000X000000
4. ERROR IN IN- PUT BLOCK. (GMIOtXOSSMM)	The sixth digit position does not contain an alphabetic character for the input location.	Type in block letter FORM: 00000000X00

PRINT-OUT	DESCRIPTION	PROCEDURE
5. INCORRECT ITEM SIZE (GMIO ^H XOSSMM) (GM ^O O ^L X ^H SSMM)	In the eighth and ninth digits a non-allowable item size is specified. See the permissible item sizes in the report for GMI and GMO.	Type in a correct item size FORM: (1) for GMI 0000000000XX (2) for GMO 00000000XX00
6. OUTPUT AREA NOT DIV TEN (GMIO ^H XOSSMM) (GM ^O O ^L X ^H SSMM)	For the item size specified output (working storage location) area must be divisible by ten.	Type in a working storage location which is divisible by ten. FORM: 0000000000XX
7. INCORRECT DENSITY SPEC. (GMIO ^H X ^L SSMM)	The density specification for the high-speed printer is an "H" in the seventh digit position, while for the unprinter it is an "L" in the seventh digit position.	Type in an "H" or an "L" FORM: 000000000000
8. INCORRECT SERVO NO. (GM ^O O ^L X ^H SSMM) (GTHW ^R _S ^B _F NNNMM)	See Printout #4.	Type in servo number greater than one. FORM: 0X0000000000
9. ITEM LOCAT NOT EVEN (GMIO ^H X ^L SSMM)	The item location must either be in or start in an even location for item sizes ² 2, 4, 6, 8, 10, 12, 20, 30.	Type in another address which is an even location. FORM: 0000000000XX
10. OUTPUT AREA ODD GM ^O O ^L X ^H SSMM	The working storage address must be even for item sizes 2, 4, 6, 8, 10, 12, 20 and 30.	Type in an even area FORM: 0000000000MM
11. BACKWARD OR FORWARD (GTHW ^R _S ^B _F NNNMM)	The sixth digit must contain an "F" or "B". In this case an "F" or "B" is not in the sixth digit.	Type in "F" or "B" FORM: 00000X000000
12. READ WRITE OR SKIP (GTHW ^R _S ^B _F NNNMM)	The fourth digit location does not have one of the letters "R" "W" or "S".	Type in "R" "W" or "S" FORM: 000X00000000

SWEEP II NORMAL PRINT-OUT

END SWEEP II

SWEEP II ERROR PRINT-OUTS:

NONE

SWEEP III NORMAL PRINT-OUT

END SWEEP III

SWEEP III ERROR PRINT-OUTS:

NONE

SWEEP IV NORMAL PRINT-OUTS

1. END SWEEP IV

2. END COMPILE

SWEEP IV ERROR PRINT-OUTS:

PRINT-OUT	DESCRIPTION	PROCEDURE
1. KEY IN Σ OR Z END SENTINEL.	The end sentinel has been omitted from the 8th digit of the FIL Routine or it is not a "Σ" or a "Z".	Type in either a "Σ" or a "Z". FORM: Z-----Z Σ-----Σ
2. DENSITY SPEC OOOOOOOOOOH or OOOOOOOOOOL	Density Specification is missing from the FIL Routine.	Type in an "H" or an "L" FORM: H-----H L-----L
3. EDF BAD C. N. O3RG00000CNN	The number of values to be edited is in error, or the number of columns is in error.	Type in acceptable number of values and also column number in the FORM: OOOOOOOO0CNN
4. EDT BAD C. N. O3RG00000CNN	The number of values to be edited is in error, or the number of columns is in error.	Type in acceptable number of values and also column number in the form: OOOOOOOO0CNN

PROBLEM RUN PRINT-OUTS

Subroutine	Print-out	Description	Procedure
1. GTHO($\frac{1}{2}$ CSX(M))	None.		
2. GRN(A)HFN(B)	2a. 50XXXX 9OMEGA 2b. EXP TOO BIG DIVIDE 000000U00XXX	Extraction of an even root (i.e., 2nd, 4th, etc.) of a negative number (A) is being attempted. See ADO	Hit the start bar to insert zero in the result.
	2c. EXP TOO BIG "0" DIVISOR 000000 U00XXX	See ADO	
	2d. EXP TOO BIG ADDITION 000000 U00XXX	See A50	
	2e. EXP TOO BIG MULTIPLY 000000 U00XXX	See A60	
	2f. EXP TOO BIG ADDITION 000000 U00XXX	See A40	

PROBLEM RUN PRINT-OUTS (Cont)

Subroutine	Print-out	Description	Procedure
3. TTO(A)000(B)	3a. TTO on XXX NO SIG DIGIT 3b. EXP TOO BIG DIVIDE 000000 U00XXX 3c. EXP TOO BIG DIVISCR 000000 U00XXX	Input value "A" exceeds machine capacity. See ADO See ADO	Hit start bar to send zeros to output for result "B". XXX is the address in which the TPO subroutine begins in memory. If zero value is not acceptable correct "A" in data and SCICR to XXX in print-out.
4. TSO(A)000(B)	TSO on XXX NO SIG FIG OK TO GO ON (Also prints input number A).	Input value (A) exceeds machine capacity.	If computation is allowed to proceed a zero value will be sent to the output for result (B). XXX is the address in which the TSO subroutine begins in memory. If zero value is not acceptable correct (A) in data and SCICR to XXX in printout.
5. TCO(A)000(B)	TCO on XXX NO SIG FIG OK TO GO ON (Also prints input number A).	Input value (A) exceeds machine capacity.	Same as above.

PROBLEM RUN PRINT-OUTS (Cont)

Subroutine	Print-out	Description	Procedure
6. TAT(A)000(B)	None.		
7. AAL(X)(ΔX)(LX)	None for AAL (AAL uses AAO)	If print-out occurs see AAO.	Consult "RECORD" to see if XXX of the printout is anywhere in the AAL routine. Type correct values into 170, 171, 172, 173 and SCICR to 1 less than address XXX of printout. If nothing can be done, correct data or pseudo-code and recompile.
8. QUD(A)(B)000	None.		
9. QUA(IAI)(IB)000	None.		
10. QTO(A)(B)000	None.		
11. QTA(IAI)(IB) 000	None.		
12. UCCCCCCCC000	None.		
13. QZO(m)000000	None.		
14. CST000000000	None.		
15. BFI(m _A)(m _B)000	None.		
16. ANI(A)000(B)	None.		
17. APN(A)(N)(B)	None. 17a. LARGE EXP (and the power N)	N > ± 99	Correct N in data and pick up in program at a convenient point. Consult record for address where APN begins in the memory. If an error is in the program correct pseudo-code and recompile.

PROBLEM RUN PRINT-OUTS (Con't)

Subroutine	Print-out	Description	Procedure
17b.	FRACT EXP (and the power N)	N is not a whole number	
17c.	G O divis _r	A = 0 and N < 0	
17d.	INDEF form	A = 0 and N = 0	
18. X+A(N)(LOG ₁₀ A)(B)	X+A ARCS out of range starts (XXX)	N times A has an exponent beyond the routine's capacity	Correct N or A in data and pick up in program at XXX (in printout). If it is an error in logic of program which produced wrong N or A, correct pseudo-code and recompile
19. SQR(A)000(B)	9SQR0900XXX (and the value) "A"	A is negative	XXX in the print out is the address in memory where SQR begins. Printout this line. It will read EGyyy K00000 yyy and yyy-1 are the addresses where (A) is stored. Type into those locations the correct value of A. SCICR to address XXX.(in printout).
20. CM10(t)($\frac{1}{2}$)0(S)(m)	None		
21. CMM(m ₁)0(n)(m ₂)	None		
22. WVO(m ₁)(n)(m ₂)	None		
23. CM0 0(t)X($\frac{1}{L}$)(m)	None		
24. GZ0(S)0($\frac{Z}{E}$)0($\frac{H}{L}$)0(t)	None		
25. AAO(A)(B)(C)	Exp too big addition 000000 U00xxx	Exponent of the sum is probably 1011, which exceeds machine capacity (See note 1 below)	Type out from 174 and 175. Increasing the second of these by 1 in the least significant digit yields the number the routine wanted to put out. If you can replace this with an acceptable number type it into 174 and 175. SCICR by typing in the last word of the printout: 000000 U00xxx

PROBLEM RUN PRINT-OUTS (Con't)

Subroutine	Print-out	Description	Procedure
26. ASO(A)(B)(C)	Exp too big addition 000000 U00xxxx	Same as for AAO(A)(B)(C)	Same as for AAO(A)(B)(C)
27. AMO(A)(B)(C)	Exp too big multiply 000000 U00xxxx	Exponent of product is ≥ (1011-1) or ≤ (-1011)	Either the result legitimately exceeds machine capacity or there is an error in the program or the data.
28. ADO(A)(B)(C)	28a. Exp too big divide 000000 U00xxxx	Exponent of quotient (or some times dividend) exceeds machine capacity	Same for AMO(A)(B)(C)
	28b. Exp too big 0 divisor 000000 U00xxxx	Divisor = zero or is not normalized	Same for AMO(A)(B)(C)
29. EDF(m ₁)(C)(n)(m ₂)	29a. EDF LG EXP (and the value) in "m." 29b. "EDF incorrect #"	Raw F.D. number has exponent exceeding 3 digits n improperly stated	None. UNIVAC proceeds after putting RAW F.D. number in edited output. A type-in to correct location has already been set up. Type in correct n for EDX 000000000(n)
30. EDT(m ₁)(C)(n)(m ₂)	EDT LG EXP (and the value) in "m."	Raw F.D. number has exponent exceeding 1101	None. UNIVAC proceeds after putting RAW F.D. number in edited output.
31. EDU(m ₁)(C)(n)(m ₂)	"EDU exponent out of range" plus None.	Raw RFD. number has exponent exceeding 3 digits	None. UNIVAC proceeds after putting RAW F.D. number in edited output.
32. EWS(Tape nos. in order)	None.		

NOTE: For the operations AAO, ASO, AMO, ADO, the input quantities (A) and (B) are in memory locations 170, 171 (A) and 172, 173 (B). The last word of the printout (000000000xxx) indicates the address to which control was to be transferred when this floating decimal operation was finished.

PROBLEM RUN PRINT-OUTS (Con't)

Subroutine	Print-out	Description	Procedure
33. RNA(A)(N)(B)	RNA root B00NNN ; 20000	N is not an integer ≤ 9 in absolute value	NNN of the printout shows the memory location of (N). Type in correct N and SCICR to yyy. Where yyy is 4 less than control counter reading when computer stopped.
	Even neg RNA Root B00NNN ; 20000	A is negative and N is even	Remember control counter reading when computer stopped. Add 11 to control counter reading and printout contents of this address. It will read W00(A) W00(B) where (A) and (B) are the addresses of these values. The address of the value (N) is NNN of the printout. Correct (A) or (N) as required and SCICR in the same manner described above.
	Exp too big multiply 000000 U00xxx	See AMO	
	Exp too big divide 000000 U00xxx	See ADO	
	Exp too big divisor 000000 U00xxx	See ADO	
34. LAU(A)(LOG ₁₀ B) (C)	LOG of non-positive U starts Mxxx	U is (A) in this subroutine .. A \leq 0	Print out address xxx. It will read "B00yyy100191" yyy is the address of (A) type in correct. Value for A and SCICR to xxx.