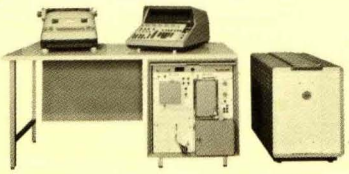
Recomp Recomp Recomp

Recomp II  **computer system**

"SCOPAC"

AUTONETICS Industrial Products
A DIVISION OF NORTH AMERICAN AVIATION, INC. 3400 E. 70TH STREET, LONG BEACH, CALIFORNIA

"S C O P A C"

A COMPILER FOR THE RECOMP II

by

H. D. Goddard

"SCOPAC"

INTRODUCTION

The present trend in computer applications in the scientific and engineering fields is to issue compiler programs which are used to generate computer instructions from source programs defining problem solving procedures. The compiler program produces the instructions (object programs) in a language understandable to the computer, enabling it to perform the necessary operations to solve the problem.

SCOPAC*is a compiler program for the RECOMP II computer. Mathematical expressions and all other pertinent information for a problem are written as SCOPAC statements. In one pass SCOPAC accepts the statements and prepares a program for solving the problem. The statements constitute a source program. As each statement is entered into the computer, SCOPAC generates and punches on paper tape all necessary computer instructions of the object program. It is the object program which is entered into the computer to solve the problem.

In this way the SCOPAC program serves as the intermediary between a computer user and the solutions to a wide range of problems. Once the SCOPAC program is in the computer any number of source programs may be entered for compiling.

* Acknowledgement is gratefully made to Mr. T. J. Tobias for the development and completion of the major portion of the SCOPAC program.

Additions and Revisions

Additions were made to the following sections:

Section 8.1, page 17

A GO TO statement is also used to transfer to a ROUTINE

GO TO routine name s

Section 8.3, page 19, the last sentence

However, the DO, GOTO, or the IF statement must not be the last statement of the DO loop.

Section 8.3.3, page 23

(c) The maximum number of DO loops that may be nested is seven.
(d) A DO, GO TO, or IF statement cannot be the last statement of a DO loop.
(e) The last statement in a DO loop cannot immediately precede the END statement.

The program in Appendix G, page 81, has been revised as follows:

Statements:

```
ARRAY B(100) s
READY ITEMS s
READY A s
DO INPUT N 1(1)ITEMS s
READY B(N) s
INPUT, CONTINUE s
DO OUTPUT N 1(1)ITEMS s
Z:(A+B(N))'2 s
CRR s
PRINT B(N) s
TAB s
OUTPUT, PRINT Z s
HALT s
END s
```

October 8, 1962

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

1. DESCRIPTION OF THE USE OF SCOPAC

The purpose of SCOPAC is to prepare on tape a program, the object program, which will solve a given problem. While SCOPAC is compiling and punching the object program, a listing of that program may be simultaneously typed out if desired. (See Appendix D for type-out of a program compiled by SCOPAC.)

1.1 The SCOPAC tape is placed in the photoreader by the operator and read into memory. A short program at the beginning of the tape will clear the memory to minus zero.

1.2 A SCOPAC statement is then entered either from the typewriter by the operator or from a previously prepared paper tape.

1.3 Computer instructions to execute the statement are generated by SCOPAC and punched on tape. Simultaneously, a listing of the computer instructions is made on the typewriter if desired.

1.4 When the last SCOPAC statement is encountered, the Assignment Table is typed out by SCOPAC. (For detailed description of the Assignment Table, see Appendix C.)

1.5 Following the Assignment Table the words, END OF ASSEMBLY, are typed out and the computer will halt.

1.6 If it is desired to prepare more object programs, the SCOPAC Initializer tape may be read into memory. Its function is to re-establish the SCOPAC program in memory, ready to accept another set of SCOPAC statements.

1.7 A source program on paper tape may be checked for errors without compiling by reading the Check SCOPAC Source Program tape into memory. The tape containing the source program is then placed in the photoreader ready to be checked for errors.

2. THE SCOPAC STATEMENTS

2.1 Kinds of Statements

The SCOPAC statement is a written directive of operations which will be translated into computer instructions and punched on tape by SCOPAC. A series of SCOPAC statements written for a problem constitute a source program. The basic SCOPAC statement is the Arithmetic statement. Each mathematical expression is written as an Arithmetic statement. The rest of the statements provide the means to reserve storage in the computer memory, to input information, to output information, to transfer program control and to enter special instructions.

There are six kinds of statements.

(1) ARRAY STATEMENTS reserve storage space in the computer memory for subscripted variables which are to be used in the program. These statements must precede all other statements in the SCOPAC program.

(2) ARITHMETIC STATEMENTS are the mathematical expressions used to solve the problem. These are the fundamental statements of the program.

(3) INPUT STATEMENTS provide for input of information from the typewriter or photoreader.

(4) OUTPUT STATEMENTS provide for the output of information from the typewriter.

(5) CONTROL STATEMENTS are used for the transfer of control from one SCOPAC statement to another.

(6) SYMBOLIC STATEMENTS permit the input of symbolic and absolute instructions.

Although the statements are written in varying forms, all statements are terminated with a figure shift (F/S) and s. (The FIGURE SHIFT key on the typewriter is depressed before pressing the s key.)

2.2 Location Tags

A location tag is an alphabetic or numeric label assigned by the programmer to a statement as a marker. This enables certain program control statements in another part of the program to refer to this statement by its location tag. Location tags may be used with most statements. In Appendix A, Summary of SCOPAC Statements, there is a tabulation indicating which statements can or cannot have a location tag.

A location tag has either eight (8) or less alphabetic characters or two (2) or less digits. It is to be noted that 00 is a different tag from 0, and 01 is a different tag from 1.

A location tag precedes the statement. It is separated from the statement by a comma.

Example:

PAR, X: A + B s

PAR is a location tag.

X: A + B s is an Arithmetic statement.

Example:

23, V: D/E s

23 is a location tag.

V: D/E s is an Arithmetic statement.

Restrictions:

A location tag should not have the same name as the name of an array prefixed by a K or R. For example, if ARRAY PART is defined in the program, a location tag should not be named KPART or RPART.

A location tag should not have the same name as the name of a routine prefixed by R. For example, if ROUTINE MAXZ is defined in the program, a location tag should not be named RMAXZ.

3. VARIABLES AND SUBSCRIPTED VARIABLES

Two basic components of the SCOPAC statement are variables and subscripted variables.

3.1 Variables

A variable is any quantity that is referred to by a name, and which is able to take on a number of values. In SCOPAC the variable is an alphabetic word consisting of from one to eight letters.

Examples:

B
SUM
CURVE
QUOTIENT

3.2 Subscripted Variables (See Appendix E, Programming of Subscripted Variables)

A variable may be subscripted. The name of the subscripted variable may not consist of more than seven letters. A subscripted variable has the form V(K) or V(K,J) where V is the name of the variable, and K and J are either numbers or variables.

A subscripted variable may refer to elements in a one dimensional or two dimensional array. The subscript of the first element in a one or two dimensional array must not be zero, it must be one.

Examples:

A(1) not A(0) one dimensional array

A(1,1) not A(0,0) two dimensional array.

A two dimensional array is composed of rows and columns. The first subscript refers to the number of rows; the second subscript refers to the number of columns.

Examples of Kinds of Arrays:

LEO (10) - a one dimensional array containing 10 elements.

MATRIX (5,7) - a two dimensional array containing 35 elements arranged in 5 rows and 7 columns.

It is necessary to allocate storage for subscripted variables. This is done by an ARRAY statement.

The use of subscript notation is a valuable technique. A large number of values may be designated by subscripts. This means that in a program a basic calculation can be set up which can be performed with each of these values simply by changing the value of the subscript. (See Example III in Appendix G.)

3.3 Restrictions

The following letter combinations should not be used as names of variables or subscripts.

(a) any Function name defined in SCOPAC. (See Subsection 5.1.3 for Function names.)

(b) the name of any array prefixed by a K or an R. For example, if VECTOR is the name of an array, do not use RVECTOR or KVECTOR.

(c) the name of any subroutine prefixed by an R. For example, if ROUTINE SORT is defined in the program, do not use RSORT.

(d) any location tag.

4. ARRAY STATEMENTS

ARRAY statements allocate storage for subscripted variables. It is mandatory to make all ARRAY statements the first statements in a SCOPAC program.* The purpose is to insure that sufficient consecutive locations in memory will be reserved. Each element of the ARRAY is handled as a floating point value. ARRAY statements always reserve locations for the total number of elements of the array plus one more floating point value. Only one-or two-dimensional arrays are permitted.

ARRAY statements have the form:

ARRAY subscripted variable (items) s

or

ARRAY subscripted variable (rows, columns) s

ARRAY statements must not have location tags.

Examples:

ARRAY LIST (23) s	Storage is reserved for 24 (23+1) floating point values (48 locations)** in a region called LIST.
ARRAY TABLE (5,6) s	Storage is reserved for 31(5x6+1) floating point values (62 locations)** in a region identified by the name, TABLE.

* Only the ORG and BLA instructions (entered with a SYMBOLIC statement) may precede the ARRAY statement. For description of the use of these instructions see Subsection 9.4.1.

** Each floating point value will occupy two locations in memory.

5. ARITHMETIC STATEMENTS

Arithmetic statements are the mathematical expressions of the problem to be solved. These statements may have location tags.

The form of the Arithmetic statement is:

Variable (or subscripted variable): expression s

Example:

D: A + B s

D is a variable

: is the symbol for equal (=)

A + B is an expression

s terminates the statement

Example:

BOB, I: R/E s

BOB is a location tag

I is a variable

: is the symbol for equal (=)

R/E is an expression

s terminates the statement

The equal symbol (:) of an Arithmetic statement does not have the same meaning as the equal sign (=) of ordinary mathematical notation. The equal symbol has the meaning "replace the value of the variable to the left of the equal symbol with the value of the expression to the right of the equal symbol."

Only a single or subscripted variable may be written on the left side of the equal symbol.

The expression is written on the right side of the equal symbol.

5.1 Expressions

An expression is any mathematically meaningful sequence of constants, variables, subscripted variables and functions related by operation symbols.

5.1.1 Operation symbols are defined as follows:

: equality sign, replaced by

\+ addition

\- subtraction

& multiplication

/ division

' exponentiation

() parentheses

The use of two signs in juxtaposition is not permitted, e.g. +-E. When parentheses are used, the number of left parentheses must equal the number of right parentheses.

The following are examples of expressions and the way they should be written in order to be acceptable to the SCOPAC program.

Expression	Expression is written as
X+Y	X+Y
X-Y	X-Y
XY	X&Y
X/Y	X/Y
X/-Y	X/(-Y)
X^{E+2}	X'(E+2)
$X^{E+2}Y$	X'(E+2)&Y
XY/VW	(X&Y)/(V&W)

5.1.2 Numbers

Numbers used in the expressions are entered as decimal numbers. They are restricted to no more than fifteen (15) characters including the sign and the decimal point. Plus signs may be omitted. Neither the integral part nor the fractional part of a number may contain more than eleven (11) characters.

Examples:

```
X:(J&Y)/(P&4.231) s
A:X + .1415 s
G: 425-Y s
```

5.1.3 Functions

Functions have the form F(E), where F is the alphabetic name of a function, and E is an expression. The following functions are defined in SCOPAC.

SQRT(E) - Square root of E

SIN(E) - Sine of E

COS(E) - Cosine of E

TAN(E) - Tangent of E

ARCTAN(E) - Arctangent of E

ARCSIN(E) - Arcsine of E

ARCCOS(E) - Arccosine of E

LOGTWO(E) - $\text{Log}_2 E$

LOG(E) - $\text{Log}_{10} E$

LN(E) - $\text{Log}_e E$

EXP(E) - e^E

EXPTWO(E) - 2^E

EXPTEN(E) - 10^E

ANRED(E) - E modulo π, in radians.

Arguments of trigonometric functions must be in radians. There is automatic angle reduction for SIN, COS, and TAN.

Examples:

```
M: 1.53 & SIN(F-G) s
D(J,K): 2.667 & SQRT(A(J,K)) s
```

6. INPUT STATEMENTS

Input statements provide the means for the input of data from the typewriter or from paper tape.

The input statements are:

READY	(for typewriter or paper tape input)
READZ	(for paper tape input)
ANGLIN	(for typewriter input)

6.1 READY Statements

READY statements permit the entry of one value. Normally, READY statements are used to input information from the typewriter. READY statements may have a location tag.

These statements have the form:

READY variable s

or

READY subscripted variable s

Example:

READY X s

This statement is entered in the source program. The value of X is entered via the typewriter or from data tape, when the READY statement is executed in the object program.

Examples:

READY MATRIX(I,J) s
READY RHO(12,K) s
READY TABLE(ITEM,6) s
READY DATA s
READY PHI(J) s
READY THETA(3) s

The READY statement used with DO loops is a convenient way to enter elements of arrays. A READY statement with one DO loop will enter elements of a one dimensional array. A READY statement with two DO loops provides a choice of reading in a two dimensional array by rows or by columns. Examples of this programming are in Subsection 8.3.2.

The READY statements may be used to read a data tape. Three conditions, however, must be met: (1) The data tape must have at least ~~eight~~ 12 blanks between each character. (2) Program Preparation Package No. 2 (P^3-2) must be read into memory before the object program is read in. (3) Before the execution of the object program two commands in P^3-2 must be changed. (See footnote in Section 17)

6.2 READZ Statements

READZ statements permit the input of an off-line prepared data tape. READZ statements must not have a location tag.

READZ statements have the form:

```
READZ variable s
```

When a READZ statement is executed in the object program one block of data will be read from paper tape. An ARRAY statement, allocating storage for the block of data, must have been given prior to the READZ statement. It is not necessary for the READZ statement to immediately follow the ARRAY statement. There may be any number of statements between ARRAY and READZ.

Example:

```
ARRAY  OPTIC (17) s
READZ  OPTIC s
```

Seventeen elements of a one dimensional array named OPTIC will be read into memory from tape when the READZ statement is executed in the object program.

Example:

```
ARRAY  V(3,4) s
READZ  V s
```

The elements of Array V, 1,2,3,4,5,6,7,8,9,10,11,12, and a -0, are in sequence on tape. When the READZ statement is executed in the object program this data will be stored in memory as follows:

4 Columns

3 rows	1	4	7	10
	2	5	8	11
	3	6	9	12

A minus zero must follow the data in each block of information. The reason is that the data is converted to floating point form in the sequence in which it is entered, and the conversion continues until a minus zero is encountered. There is a further restriction that none of the data should be minus zero, since the data following the minus zero would not be converted to floating point form.

6.3 ANGLIN Statements

ANGLIN statements permit the entry of one angle from the typewriter.

ANGLIN statements have the form:

ANGLIN variable s

or

ANGLIN subscripted variable s

ANGLIN statements may have a location tag.

Example:

INANG, ANGLIN A s

This statement is entered in the source program. The value of A is entered in degrees, minutes, and seconds via the typewriter when called for at the time the ANGLIN statement is executed in the object program. The angle is then converted to radians by a subroutine in P^3-2.

Example:

ANGLIN PAR(K,J) s

Many angles may be input with one ANGLIN statement when the statement is part of a DO loop. (See Subsection 8.3.1 for an example.)

7. OUTPUT STATEMENTS

Output statements provide for output of information via the typewriter. The output will occur at the time the object program is executed.

These statements are:

PRINT
TYPE
CRR (carriage return)
TAB
ANGLOUT

7.1 PRINT Statements

PRINT statements provide for the output of one value.

The statements have the form:

PRINT variable s

or

PRINT subscripted variable s

PRINT statements may have a location tag.

Examples:

PRINT ANSWER s
02, PRINT VECTOR (M) s
PRINT MATRIX (K,J) s

The PRINT statement does not provide for a carriage return or tab. The statements CRR or TAB are used for this purpose.

For Example:

```
PRINT A s
TAB s
PRINT B s
TAB s
PRINT C s
CRR s
```

PRINT statements do not include the format of the type-out. A value will be typed in fixed point format if it is greater than $|.61036 \times 10^{-4}|$ and less than $|.54975 \times 10^{12}|$. Otherwise it is typed in floating point format.

Examples:

+.61035998459-5 (floating point format for $+.61035998459 \times 10^{-5}$)

61.036000000 (fixed point format)

+.20675703017+14 (floating point format for $+.20675703017 \times 10^{14}$)

7.2 TYPE Statements

TYPE statements output alpha-numeric characters. A maximum of 150 characters may be typed using one TYPE statement.

TYPE statements must not have a location tag.

The TYPE statement has the form:

TYPE sp L/S or F/S alphanumeric characters F/S s

The first five characters must be TYPE sp.

sp means space bar on the typewriter.

L/S means the LETTER SHIFT key on the keyboard.

F/S means the FIGURE SHIFT key on the keyboard.

Example:

TYPE sp L/S ABSCISSA sp AT sp Y sp MAX F/S: F/S s

This entry would be typed out as follows:

ABSCISSA AT Y MAX:

If a tab or carriage return is used in a TYPE statement the format of the assembly listing will be destroyed. This can be averted by turning Switch B OFF while SCOPAC compiles a TYPE statement containing a tab or carriage return.

For example to type:*

MAX X MAX Y

as one TYPE statement the following could be done:

TYPE sp L/S MAX sp X F/S tab L/S MAX sp Y L/S C/R F/S s

where C/R means carriage return.

* with Switch B turned OFF

8. CONTROL STATEMENTS

Control statements are used to transfer control from one statement to another.

They are:

GO TO	(Transfers control unconditionally)
IF	(Transfers control conditionally)
DO	(Controls repetition of operations)
CONTINUE	(Dummy statement)
ROUTINE	(Provides the means to write subroutines using SCOPAC statements)
RETURN	(Transfers control to the main program from subroutines written with the ROUTINE statement)
HALT	(Causes object program to halt)
END	(Terminates compiling by the SCOPAC program)

A location tag may be referred to in a control statement. This use of a tag will be included in the following descriptions of the control statements.

8.1 GO TO Statements

The GO TO statement unconditionally transfers program control to a specified tagged statement. This tagged statement may be either before or after the GO TO statement. After control is transferred, another sequence of statements is executed beginning with the tagged statement.

GO TO statements have the form:

```
GO TO tag s
or
GOTO tag s
```

A GO TO statement is also used to transfer to a ROUTINE.

```
GO TO routine name s
```

GO TO statements may have a location tag.

Example :

GO TO DEANE s

DEANE is the tag of the statement to which control is unconditionally transferred. A transfer to the location assigned to the tag, DEANE, will occur when this statement is executed in the object program. (See example in Subsection 8.6)

8.2 IF Statements

IF statements are used to transfer program control conditionally. IF statements have two forms:

(a) IF (expression) minus, zero, plus s

(b) IF (SENSE N) ON, OFF s

These statements may have location tags. In form (a) three location tags follow the expression. They need not be different tags. Control is transferred to one of the tagged statements according to whether the value of the expression is minus, zero, or plus.

Example:

```
REDO, M:M+.333 s
X:.5&M s
IF (X-3.4) REDO, REDO, OUT  s
OUT, PRINT X s
```

If the expression, (X-3.4), is minus or zero, the object program transfers control to the statement tagged REDO.
If (X-3.4) is plus, the object program transfers control to the statement tagged OUT.

In form (b) N stands for Sense Switch B, C, or D. Two location tags follow the Sense Switch. Control is transferred to one of these tagged statements according to the position of the sense switch when the IF statement is executed in the object program.

Example:

IF (SENSE D) OUTPUT, ITERATE s

If Sense Switch D is ON, control is transferred to the statement tagged OUTPUT when the IF statement is executed during the object program.

If Sense Switch D is OFF, control is transferred to the statement tagged ITERATE when the IF statement is executed during the object program.

8.3 DO Statements

DO statements control iteration loops, which are called "DO loops". Basically, a DO loop is a set of SCOPAC statements which are performed several times with changing values or conditions.

DO statements have the form:

DO tag variable I.V. (mod.) F.V. s

DO statements may have a location tag.

The value of the variable is changed by the modifier (mod.) from the initial value (I.V.) to a value not exceeding the final value (F.V.). The modifier (mod.) may be either an increment or a decrement and may be entered as a numeric value or an alphabetic name. If an increment is entered as an alphabetic name, the name must not contain the letter "A". (See Example III of Appendix G.) If a decrement is entered as an alphabetic name, the name must contain at least one letter "A". The initial value (I.V.) and final value (F.V.) must have the same sign.

The tag specifies the last statement in the range of the DO loop. Any statement that permits the use of a location tag will terminate a DO loop if used as the last statement of the loop. However, the DO, GOTO or the IF statement must not be the last statement of the DO loop.

Example:

```
TYPE K s
TAB s
TYPE FUNCTION s
DO ITERATE K 1(2)7 s
FUNCTION: THETA(K)/SIGMA(K) s
CRR s
PRINT K s
TAB s
ITERATE, PRINT FUNCTION s
```

The DO statement means "perform all statements, beginning with the next statement after the DO statement, to and including the statement tagged ITERATE". This DO loop would perform all statements within its range four times: K=1 the first time; K=3 the second time; K=5 the third time; K=7 the fourth time.

The printout would be as follows:

K	FUNCTION (numerical value of FUNCTION will be printed)
1.00000000000	θ_1 / σ_1
3.00000000000	θ_3 / σ_3
5.00000000000	θ_5 / σ_5
7.00000000000	θ_7 / σ_7

Example: Decrementing a variable

```
DO GREG X 100(-10)10 s
MZ:1/SQRT (X'2 + K'4/X'2)
PRINT X s
TAB s
PRINT MZ s
GREG, CRR s
```

The DO loop will solve MZ for 10 values of X, beginning with X=100 and decrementing X by 10 each time.

8.3.1 DO loops contained within the range of another DO loop.

Example:

```
DO OUTPUT K 1(1)3 s
DO OUTPUT J 1(1)4 s
PRINT M(K,J) s
OUTPUT, CRR s
```

Range of inner loop

Range of outer loop

The DO loops will print the elements of the array M.

M is an array with 3 rows and 4 columns
K is the row number
J is the column number

This would print

M_{11}

M_{12}

M_{13}

M_{14}

M_{21}

M_{22}

M_{23}

.
.
.

M_{34}

Note that the inner DO loop is completed before control is transferred to the outer DO loop. The DO loops are said to be "nested".

Example:

```
ARRAY PAR(6,5) s
DO PAM K 1(1)6 s
DO PAM J 1(1)5 s
PAM, ANGLIN PAR(K,J) s
```

The DO loops provide for the entry of 30 angles via the typewriter when these statements are executed in the object program. There may be any number of statements between the ARRAY statement and the DO statements.

8.3.2 Use of DO Loops With READY Statement To Input Elements of An Array

The elements of a two dimensional array may be input either row or column wise. The subscript placed in the outer DO loop determines the way the elements will be entered i.e. if the outer DO loop contains the row subscript, input will be by rows; if the outer DO loop contains the column subscript, input will be by columns. There may be any number of statements between the ARRAY statements and DO statements.

Example: To Input by Rows

```
ARRAY MATRIX (2,4) s
DO TAG I 1(1)2 s
DO TAG J 1(1)4 s
TAG, READY MATRIX (I,J) s
```

The array would be read in row wise because J would be the most rapidly changing subscript.

Example: To Input by Columns

```
ARRAY MATRIX (2,4) s
DO TAG J 1(1)4 s
DO TAG I 1(1)2 s
TAG, READY MATRIX (I,J) s
```

The array would be read in column wise because I would be the most rapidly changing subscript.

Example: To Input a One Dimensional Array

```
ARRAY PHI (4) s
DO TAG J 1(1)4 s
TAG, READY PHI (J) s
```

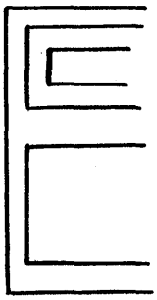
The four elements of ARRAY PHI will be read in during the execution of this DO loop in the object program.

8.3.3 Restrictions

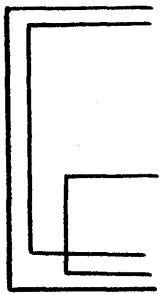
The following restrictions apply to the use of DO loops. In the diagrams, the brackets represent the range of statements under control of a DO statement.

(a) If the range of a DO statement includes another DO statement, all statements in the range of this second statement must also be in the range of the first DO statement.

Permitted

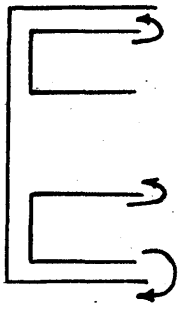


Not Permitted

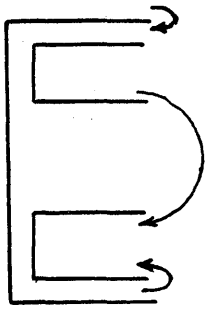


(b) No transfer of control by IF or GOTO statements is permitted <u>into</u> the range of any DO statement from outside its range, since such transfers would not permit the DO loop to be properly indexed. However, an IF statement may be used to transfer <u>out</u> of a DO loop.

Permitted



Not Permitted



(c) The maximum number of DO loops that may be nested is seven.

(d) A DO, GO TO, or IF statement cannot be the last statement of a DO loop.

(e) The last statement in a DO loop cannot immediately precede the END statement.

8.4 CONTINUE Statements

CONTINUE statements are dummy statements.

CONTINUE statements usually have the form:

tag, CONTINUE s

A CONTINUE statement with a tag may be the last statement in a DO loop. When used in this way control passes to the statement following the CONTINUE statement after the operations of the DO loop are completed.

A CONTINUE statement with a tag, may be used to provide a means of transferring control to a statement that cannot have a location tag, e.g. TYPE or SYMBOLIC. The statement that cannot be directly tagged should immediately follow the tagged CONTINUE statement.

Example:

LARRY, CONTINUE s

TYPE Y EQUALS s

A statement such as GO TO LARRY s elsewhere in the program would transfer control to the statement LARRY, CONTINUE s. This would be followed by the execution of the statement TYPE Y EQUALS s.

8.5 ROUTINE Statements

ROUTINE statements provide the means to write subroutines with SCOPAC statements.

ROUTINE statements have the form:

ROUTINE name s

ROUTINE statements must not have a location tag.

The name identifies the subroutine.
The name may consist of not more than seven (7) alpha characters.

Example:

ROUTINE SYSTEM s

A transfer statement to the subroutine should not be the statement immediately preceding the ROUTINE statement. The ROUTINE statement is followed by one or more SCOPAC statements which constitute the subroutine. These statements must be followed by a RETURN statement.

8.6 RETURN Statements

RETURN statements transfer control to the main program after the execution of a subroutine. Control returns to the statement which follows the transfer statement to the subroutine.

RETURN statements have the form:

```
RETURN name s
```

RETURN statements must not have a location tag.

Example:

```
GO TO GUSS s
GO TO RON s
ROUTINE GUSS s
Y:X'3+SIN(M) s
RETURN GUSS s
RON, PRINT Y s
```

In this example Y:X'3+SIN(M) is the subroutine named GUSS. Note that when the statement RETURN GUSS s is executed, the return from the subroutine to the main program will be made to the statement GO TO RON s.

To avoid transferring control around subroutines in the main body of the program, subroutines may be placed after the main body of the program.

8.7 HALT Statements

HALT statements cause a halt and transfer to location 3000_8 to be executed in the object program.

HALT statements have the form:

```
HALT s
```

HALT statements may have a location tag.

8.8 END Statements

An END statement indicates to the SCOPAC program that there is no more information. This is the only purpose of the END statement. Since the END statement does not generate a halt instruction it is advisable to use a HALT statement just prior to the END statement.

END must be the very last SCOPAC statement.

END statements have the form:

```
END s
```

END statements must not have a location tag.

9. SYMBOLIC STATEMENTS

9.1 SYMBOLIC Statements

The SYMBOLIC statements permit input of symbolic coding.

SYMBOLIC statements have the form:

```
SYMBOLIC s
```

SYMBOLIC statements must not have a location tag.

The symbolic coding is entered immediately following the SYMBOLIC statements. (See Subsection 12.8 for examples.)

9.2 COMPILE

COMPILE is used with the SYMBOLIC statement to notify the SCOPAC program that no further symbolic coding is to be entered. Each time symbolic coding is entered in the source program it is preceded by the SYMBOLIC statement and terminated with the word, COMPILE.

9.3 Symbolic Coding Entered With The SYMBOLIC Statement

The SYMBOLIC statement permits the use of RECOMP II operations* and certain special operations. Although the RECOMP II magnitude instructions cannot be entered directly, magnitude operations can be coded with the SYMBOLIC statement as described in Subsection 9.4. In turn, these operations are used to enter decimal and octal numbers, addresses, alpha-numeric information, and instructions in command format.

Functions may be entered as symbolic coding provided the arguments of the Functions are in the A and R registers just preceding the execution of the Functions. Normally, Functions are entered in the Arithmetic statements.

READY, ANGLIN, PRINT, ANGLOUT, CRR, TAB may also be entered with a SYMBOLIC statement. If READY or ANGLIN are entered as symbolic coding, the input must be stored just after the execution of READY or ANGLIN. If PRINT or ANGLOUT are entered as symbolic coding, the output must be in the A and R registers just prior to the execution of PRINT or ANGLOUT.

* See Appendix B for list of acceptable operations.

The special operations are:

ORG (origin)	The first word of the object program is assigned the absolute address specified in the ORG instruction. This operation is used only if an origin other than 3000_8 is desired. The SCOPAC program automatically assigns an origin of 3000_8.
BLA (block allocation)	In the BLA operation the total number of locations for the object program is specified. This operation is used only if the storage allocated for the object program is to be restricted between the origin and a location other than 7757_8.
DEF (definition)	In the DEF operation a symbolic location is assigned an absolute address.
EQU (equal)	In the EQU operation two symbolic addresses are assigned the same absolute address.

Special operations used with the SYMBOLIC statement for output of words in decimal number format are:

DISD (display decimal)	The specified word is displayed as a decimal number in the display register.
TYWD (type word decimal)	The specified word is typed as a decimal number.
PNWD (punch word decimal)	The specified word is punched in decimal number format.
PTWD (punch and type word decimal)	The specified word is typed and punched in decimal number format.

NOTE: The above four operations require that the specified word be in binary coded decimal (BCD) format prior to execution of the operation.

9.4 Writing Instructions In Symbolic Coding

There are four fields used when writing instructions in symbolic coding. They are Location, Command, Type and Address.

The format is:

```
SYMBOLIC s
LOCATION        COMMAND         TYPE        ADDRESS
field           field           field       field
COMPILE
```

LOCATION Field

The Location field may contain a location tag. This tag may be from one to eight alphabetic characters. A numeric tag cannot be used. The Location field may be left blank.

COMMAND Field

The Command field may contain RECOMP II operations, special operations, Functions, or READY, ANGLIN, PRINT, ANGLOUT, CRR, and TAB.

TYPE Field

The Type field specifies the type of entry which will follow in the Address field. A single letter code is used to designate the type of entry.

- S Symbolic (or if the Type field is left blank, it is interpreted as symbolic.)
- N Numeric
- A Alphanumeric
- C Command format
- F Floating point decimal number
- D Fixed point decimal number

ADDRESS Field

The Address field may contain one of the types of entries specified in the Type field, or is left blank.

Symbolic (Address field)

A symbolic address containing from one to eight alphabetic characters may be entered.

Example:

DATA

Addresses relative to the symbolic address may be written:

Examples:

DATA-0001	one word before DATA
DATA+0001	one word after DATA
DATA+27	27 words after DATA

To insure that SCOPAC will not assign these locations for other purposes, it is advisable to reserve these locations with an ARRAY statement, i.e. ARRAY DATA (27) s.

Example:

LOCATION	COMMAND	TYPE	ADDRESS
	FCA		ABC
	FST	S	DATA

Magnitude Operations

Although the RECOMP II codes for magnitude operations cannot be entered, the use of a symbolic address +00001 is a way of specifying the magnitude operations.

Example:

LOCATION	COMMAND	TYPE	ADDRESS
	FCA		ABC+00001
	FST	S	DATA

The absolute value of the contents of ABC will be stored in DATA.

Numeric (Address field)

The numeric entry in the Address field may be from one to five numerals . If five numerals are in this field, they are interpreted as the usual RECOMP II address, with 4 octal digits for channel and sector plus a half-word bit. If there are fewer than five numerals, the address is interpreted as a full word address with a half-word bit of zero. Enough leading zeros are automatically inserted to right justify the numerals.

Examples:

ENTERED AS:	INTERPRETED AS:
7737	7737.0
525	0525.0
17	0017.0
23451	2345.1
1	0001.0
00001	0000.1
0	0000.0

Example:

LOCATION	COMMAND	TYPE	ADDRESS
	CLA	N	6032
	STO	N	5004

Alphanumeric (Address field)

The alphanumeric entry may contain eight or less characters.

Examples:

MAXZ

SPAN1

CURVE 4

Example:

LOCATION	COMMAND	TYPE	ADDRESS
	CLA	A	CURVE 4
	TYA	N	7760

The alphanumeric entry, CURVE 4, will be typed during the execution of the object program.

Command (Address field)

A word in command format may be entered in the Address field.

An example of the command format is:

+0050320+6030300

Example:

LOCATION	COMMAND	TYPE	ADDRESS
	CLA	C	+0050320+6030300
	STO	N	6010

The word, +0050320+6030300 will be stored in 6010 during the execution of the object program.

Floating Point Decimal (Address field)

A floating point decimal number in the Address field may contain a maximum of sixteen (16) characters including the sign and the decimal point.

Neither the integer nor the fraction may contain more than eleven (11) characters. The sign will be interpreted as plus if no sign is given.

Example:

-142857
642.31
.0513

Example:

LOCATION	COMMAND	TYPE	ADDRESS
	FCA	F	1.3
	FMP		X

Fixed Point Decimal (Address field)

If a decimal number is entered as a fixed point number the position of the binary point must be specified.

A fixed point decimal number in the Address field may contain a maximum of sixteen (16) characters including the sign, decimal point and position of the binary point (two or three characters are needed to designate the binary point). Neither the integral part nor the fractional part may contain more than eleven (11) digits. If there is no sign, the number is assumed to be positive.

Either the integral part or the fractional part may be omitted if it is not needed. If the position of the binary point is omitted, the number will be converted incorrectly to a floating point number without an exponent. There will be no error halt for this condition.

If the magnitude of the number is such that it cannot be converted at the specified binary point the number will be incorrectly converted. There will be no error halt for this condition.

Examples of fixed point decimal entries are:

-142857+18	for	-142,857 at binary point 18
642.31+10	for	642.31 at binary point 10
.0513-4	for	.0513 at binary point -4

Example:

LOCATION	COMMAND	TYPE	ADDRESS
	CLA	D	642.31+10
	STO		Z

9.4.1 Writing the Special Operations in Symbolic Coding

Certain special operations, if used, must be placed at the beginning of the source program. The order of entry is:

1. ORG
2. BLA
3. ARRAY Statements (These are not entered by symbolic coding.)
4. DEF and/or EQU

ORG (origin)

This operation is used only if an origin other than 3000_8 is desired. The SCOPAC program automatically assigns an origin of 3000_8. If used, this operation must be the first entry in the source program.

The first word of the object program is assigned the location specified in the Address field. All other words in the object program will be assigned locations greater than the origin. (Origin must be an even number ≥ 0004. The Type field must contain N.

Generally, Program Preparation Package No. 2 (P^3-2) is used in conjunction with the object program. P^3-2 occupies locations 0003_8 and 0010_8 through 2777_8. Therefore, if an origin $< 3000_8$ is specified, part of P^3-2 will be destroyed.

Example:

LOCATION	COMMAND	TYPE	ADDRESS
	ORG	N	5000

An origin of 5000_8 is assigned to the object program.

BLA (block allocation) This operation is used only if the storage allocated for the object program is to be restricted between the origin and a location other than 7757_8.

The total number of locations allocated for the object program is specified in the Address field. These locations are consecutive. The BLA operation must precede all ARRAY statements in a source program.

The number of locations is written as an octal number.

The Type field must contain an N.

Example:

LOCATION	COMMAND	TYPE	ADDRESS
	BLA	N	100

100_8 locations will be reserved for the object program. Assuming an origin of 3000_8 the storage allocated extends from 3000_8 to 3077_8.

DEF* (definition)

The symbolic address in the Location field is assigned the absolute address in the Address field.

The Type field must contain N.

This operation enables the programmer to assign an absolute address to a symbolic address.

Example:

LOCATION	COMMAND	TYPE	ADDRESS
ABC	DEF	N	3200

ABC is the symbolic address.

DEF is the operation.

N signifies a numeric address follows.

3200_8 is the absolute address assigned to the symbolic address, ABC.

EQU* (equal)

The symbolic address in the Location field is assigned the same absolute address as the symbolic address in the Address field.

Example:

LOCATION	COMMAND	TYPE	ADDRESS
DOG	EQU	S	PUP

DOG is a symbolic address.

EQU is the operation.

S means a symbolic address will follow.

PUP is the symbolic address which will have the same absolute address as DOG.

* It is advisable that DEF and EQU operations immediately follow the ARRAY statements to insure that all symbols with defined locations and all equivalences are known before reference is made to them.

The following four operations require that the word be in BCD (Binary Coded Decimal) format prior to execution. BCD format must be arranged by programming. (See the description of the Display operation in the RECOMP II Operating Manual for details regarding BCD format).

DISD (Display Decimal)	The word specified in the Address field is displayed as a decimal number. The Type field must be either N or blank depending on whether the Address field contains a numeric or symbolic entry.
TYWD (Type Word Decimal)	The word specified in the Address field is typed as a decimal number. The Type field must be either N or blank depending on whether the Address field contains a numeric or symbolic entry.
PNWD (Punch Word Decimal)	The word specified in the Address field is punched in decimal number format. The Type field must be either N or blank depending on whether the Address field contains a numeric or symbolic entry.
PTWD (Punch and Type Word Decimal)	The word specified in the Address field is typed and punched in decimal number format. The Type field must be either N or blank depending on whether the Address field contains a numeric or symbolic entry.

10. PREPARATION OF SOURCE PROGRAM ON PAPER TAPE

SCOPAC statements and/or symbolic coding may be prepared on paper tape.

[For Flexowriter offline tape preparation of SCOPAC source language and data, see RECOMP Technical Bulletin No. 21. The bulletin was written for tape preparation of source programs for SALT (Program No. 1034) but applies as well to SCOPAC.]

10.1 SCOPAC statements and/or symbolic coding must be prepared on paper tape in exactly the same format that is used for typewriter input. There must be at least eight (8) blanks between each character, including figure shift (F/S), letter shift (L/S), space (sp), etc.*

10.2 When preparing a statement on paper tape, each name or number must be terminated by a space, figure shift, or letter shift.

10.3 Terminate each statement by a figure shift followed by an s.

10.4 When preparing symbolic instructions on paper tape terminate the Location, Command, and Type fields by a tab. Terminate the Address field by a carriage return.

10.5 In the following examples at least 8 blanks must be inserted between each character.*

b means 8 blanks

t means a tab

Example:

X:3 s

This would be entered on paper tape as:

XbF/Sb:b3bF/Sbspbs

Example:

32,CRRs

This would be entered on paper tape as:

F/Sb3b2bF/Sb,bL/SbCbRbRbF/Sbs

* The requirement that at least eight blanks be between each character applies only if the tape is to be read in via the photoreader.

Example:

SYMBOLICs

ORG N 3400

COMPILE

This would be entered on paper tape as:

L/SbSbYbMbBbObLbIbCbF/SbsbtbL/SbObRbGbF/SbtbL/SbNbF/Sbtb3b4bObObL/SbC/R

L/SbCbObEbPbIbLbEbF/SbtbtbtbL/SbC/R

10.6 A source program may be on more than one tape. In this case the following precautions must be taken: (1) Each tape must have a complete statement at the end (i.e. a statement may not be continued on another tape); (2) All symbolic instructions to be entered by a SYMBOLIC statement must be punched on one tape together with the SYMBOLIC statement and the word, COMPILE.

11. CORRECTION OF ERRORS WHILE PREPARING SOURCE PROGRAM ON TAPE

11.1 If an error is discovered before termination of a statement or symbolic instruction, depress LINE FEED key and re-type the statement or instruction. If an error occurs, such as an incorrect character, it is possible to correct without depressing the LINE FEED key by manually moving the tape back to a point one space ahead of the incorrect character and entering the correct character. The tape is moved so that the correct character is just ahead of the incorrect character.

11.2 If an error is discovered after termination of a statement or symbolic instruction, it may be corrected by manually moving the tape back to a point one space ahead of the termination character s or C/R and depressing the LINE FEED key. The statement or symbolic instruction may then be re-typed.

12. INPUT OF SOURCE PROGRAM FROM THE TYPEWRITER

12.1 For operations prior to entering the SCOPAC statements see Compiling under Operating Instructions, Section 14.

12.2 A statement may be entered only when the ALPHA light is ON and the COMPUTE light is OFF.

12.3 No statement may consist of more than 150 characters. This includes letter shift (L/S), figure shift (F/S), space (sp), carriage return (C/R), etc.

12.4 The expression in an Arithmetic statement may contain a maximum of 70 characters of the following type: All parentheses, operators (+, -, &, /, '), Functions, variables and numbers.

The entire Arithmetic statement may contain a maximum of 150 characters including letter shifts, figure shifts, and all other characters.

12.5 A blank is a character in typewriter input.

12.6 Typing A Statement*

All location tags, variables, subscripts, numbers and names in a statement must be terminated. The termination may be done by a letter shift (L/S), figure shift (F/S) or a space (sp).

Each statement must be terminated by a figure shift (F/S) followed by an s.

At the beginning of any statement the typewriter is in L/S mode. Therefore, if the first character requires L/S mode it is not necessary to depress the LETTER SHIFT key.

* NOTE: 1. If an error is discovered before termination of a statement (prior to entering the s, or prior to depressing the CARRIAGE RETURN key if symbolic coding), depress the LINE FEED key and retype the statement or the line of symbolic coding.

2. If an error is discovered after termination of a statement (after entering the s, or after depressing the CARRIAGE RETURN key if symbolic coding) depress the STOP key as soon as possible. For following steps see Correction of Errors While Compiling a Source Program, Subsection 14.6.

Example:

DLD,TAB s

This is typed as:

DLD F/S, L/S TAB sp F/S s [The space (sp) before F/S is not necessary.]

Example:

01,CRR s

This is typed as:

F/S 01 F/S, L/S CRR sp F/S s

Example:

READY V(2,3) s

This is typed as:

READY sp V F/S (2 F/S , 3 F/S) sp s

Example:

ABC,Z(Q,J):2&X'3+W/2 s

This is typed as:

ABC F/S, L/S Z F/S (L/S Q F/S , L/S J F/S):2 F/S &
L/S X F/S ' 3 F/S + L/S W F/S / 2 F/S sp s

12.7 Entering TYPE Statements

The five characters, TYPE sp, must precede information to be entered with a TYPE statement.

If a tab or carriage return is to be part of the TYPE statement, turn Switch B OFF while SCOPAC compiles the TYPE statement. (Otherwise, the format of the assembly listing will be destroyed.) The alternative would be to use the statements TAB and CRR separately from the TYPE statement.

The TYPE statement may be corrected by either depressing the LINE FEED key and re-entering the statement or by using the h character in figure shift (F/S) mode. When the latter method is used, characters may be removed in the reverse order of their input, a character at a time, by depressing the FIGURE SHIFT key once and then by depress-

ing the h key for each of the characters to be removed. Thus, if three characters in succession were incorrectly entered, F/S hhh followed by the three correct characters would be typed.

12.8 Typing Symbolic Instructions

When typing symbolic instructions, terminate the Location, Command and Type fields by a tab. Terminate the Address field by a carriage return (C/R). The last character in the Address field should be followed by a carriage return only.

The formats are:

SYMBOLIC s

LOCATION		COMMAND		TYPE		ADDRESS	
1-8 Alpha characters or left blank	tab	3-7 Alpha characters	tab	1 char-acter	tab		
				S or left blank		Symbolic 1-8 characters	C/R
				N		Numeric 1-5 numerals	C/R
				A		Alphabetic 1-8 alpha-numeric characters	C/R
				C		Command Format such as: +3030620+5736700	C/R
				F		Floating Point Decimal 1-16 characters including sign and decimal point	C/R
				D		Fixed Point Decimal 1-16 characters including sign, decimal point & location of binary point	C/R

Although READY, ANGLIN, PRINT, ANGLOUT, CRR, and TAB are usually entered as statements, and the Functions are usually entered in Arithmetic statements, they may be entered as symbolic coding.

LOCATION	COMMAND	TYPE	ADDRESS
1-8 characters or left blank	Functions	left blank	left blank
	READY		
	ANGLIN		
	PRINT		
	ANGLOUT		
	CRR		
	TAB		

Example:

SYMBOLIC s

ABC	tab	FCA	tab	S	tab	X	C/R
tab		COS	tab	tab		C/R	
tab		FST	tab	tab		Y	C/R
tab		FMP	tab	F		3	C/R
BBC	tab	PRINT	tab	tab		C/R	
tab		CRR	tab	tab		C/R	
COMPILE	tab	tab		tab		C/R	

Example:

SYMBOLIC s

CCD	tab	CLA	tab	C	tab	+3030620+5736700	C/R
tab		STO	tab	N	tab	2142	C/R
tab		ADD	tab	D	tab	+2+18	C/R
tab		STO	tab	N	tab	2170	C/R
COMPILE	tab	tab		tab		C/R	

13. CHECKOUT OF RECOMP II READINESS

13.1 Turn computer POWER Switch ON and wait for READY light above switch to go ON.

13.2 Turn Input-Output POWER Switch ON (on the front control panel).

13.3 Turn the COMPUTER-MANUAL PUNCH Switch to COMPUTER.

13.4 If tape is to be read into memory:

a. Turn Tape Reader POWER Switch ON.

b. Turn Tape Reader MOTOR Switch ON.

c. Turn the TAPE ADVANCE Switch to OFF.

13.5 If tape is to be punched:

a. Turn Tape Punch POWER Switch ON.

b. Turn the PUNCH-EXTERNAL Switch to PUNCH.

13.6 Turn typewriter ON-OFF Switch (under lower right corner) to ON.

13.7 Turn the OPERATION Switch (on the Console) to CONTINUOUS.

13.8 Turn the PRE-SET STOP Switch to OFF.

13.9 Turn TRANSFER STOP Switch OFF (down).

The computer is now ready to use.

14. OPERATING INSTRUCTIONS FOR COMPILING

14.1 Preliminary Procedure for Compiling One or More Source Programs

a. Load the SCOPAC tape. (A short program at the beginning of the tape will clear memory to minus zero.) SCOPAC will use all memory locations.

To verify the SCOPAC tape:

(1) Place the tape in the photoreader past the short "zero memory" program.

(2) Press the VERIFY button.

b. Set typewriter margins and tabs.

(1) Set typewriter margins at 6 and 97.

(2) Set tab stops at 23, 33, 53, and 60.

(3) Set tab override switch in the OFF position.

14.2 Input of Source Program

a. Advance at least two folds of blank tape.

b. If the object program is to be listed, turn Sense Switch B ON.

c. If input is via the typewriter:

(See Section 12, Input of Source Program From the Typewriter and Subsection 14.6, Correction of Errors While Compiling a Source Program.)

(1) Depress START 1 button.

(2) After entering a statement or symbolic instruction wait for the computer to halt with the ALPHA light on.

(3) Type the next statement or symbolic instruction.

(4) After all of the source program has been entered, proceed to Step e.

d. If input is to be entered from paper tape:

(See Section 10, Preparation of a Source Program on Paper Tape and Subsection 14.6, Correction of Errors While Compiling a Source Program)

(1) Place input tape in photoreader.

(2) Depress START 2 button if all input statements are to be typed.

(3) Depress START 3 button if only those input statements in error are to be typed.

(4) After the source program is processed, proceed to Step e (1).

(5) If the source program is on more than one tape:

(a) Place each tape in the photoreader.

(b) Press the desired START button (START 2 or 3) and compiling will continue.

e. After the END statement is entered:

(1) Wait for the Assignment Table to be typed.

(2) Wait for END OF ASSEMBLY to be typed.

f. Remove the object program from the punch canister.

14.3 To Initialize the SCOPAC Program

a. Instead of reloading the SCOPAC program, it may be initialized (re-establish SCOPAC in memory without reloading it) at any time during compiling by the SCOPAC Initializer tape.

(1) Press FILL button on the Console.

(2) Load SCOPAC Initializer tape. The SCOPAC program will be re-established in memory and the computer will halt at 1413.1.

(3) Return to Step 14.2.

14.4 To Interrupt Compiling

To interrupt compiling while entering SCOPAC statements via the typewriter:

a. When the typewriter is ready for input of the next statement (ALPHA light on), turn the computer OFF.

b. To resume compiling with typewriter input:

(1) Turn the computer ON.

(2) Turn POWER switch ON for the tape punch unit.

(3) Check Switch B to be sure it is in the same position as it was at the time compiling was interrupted.

(4) Press START 1 button.

(5) When the ALPHA light comes on, resume compiling.

To interrupt compiling while SCOPAC statements are being entered via paper tape:

a. Wait for a time when the tape is not being read by the photoreader; then press STOP key on the Console.

b. Set Channel and Sector dials to 4720.

c. Turn PRE-SET STOP Switch to 1st.

d. Press START key.

e. After the program halts at location 4720, turn the PRE-SET STOP switch OFF.

f. Turn the computer OFF.

g. Either leave tape in photoreader or mark tape and remove it. (Mark tape so that it may be placed in the same position on photo reader when compiling is resumed.)

h. To resume compiling with input from paper tape:

(1) Turn the computer ON.

(2) Turn the POWER switch ON for the tape unit.

(3) Check Switch B to be sure it is in the same position as it was at the time compiling was interrupted.

(4) If necessary, place input tape in photoreader in the same position it was at the time compiling was interrupted.

(5) Turn on photoreader (by turning the POWER switch and the MOTOR switch ON).

(6) Set Location Counter to 4720.0 by pressing L key, typing the location, and depressing the ENTER key.

(7) Press START key. (The type-out specified originally by START 2 or START 3 will be resumed automatically.)

14.5 Check of Source Program on Tape Without Compiling

The Check SCOPAC Source Program tape is used to run a check of the statements in a source program on tape without compiling.

a. Load the SCOPAC tape.

b. Load the Check SCOPAC Source Program tape.

c. Place the source program in the photoreader.

d. Press either START 2 or START 3 button.

(1) START 2 - every statement on the tape will be listed. If an error is detected, ERROR or EQUATION ERROR will be typed.

(2) START 3 - only those statements in error will be listed. If an error is detected, ERROR or EQUATION ERROR will be typed followed by a type-out of the statement in error.

e. If errors in symbolic coding are detected, SYMBOLIC CODE ERROR will be typed. If the error was such that SCOPAC fails to interpret the word COMPILE as the termination of the symbolic coding, statements that follow will be processed as symbolic coding, and SYMBOLIC CODE ERROR will be typed.

f. Any number of off-line prepared tapes containing source programs may be checked without reloading SCOPAC and the Check SCOPAC Source Program tapes. Just place the tape in the photoreader and depress START 2 or START 3.

g. After checking all tapes for errors, it is only necessary to load the SCOPAC Initializer tape and a source program tape in order to begin compiling.

14.6 Correction of Errors While Compiling a Source Program (typewriter or tape input)

a. If an error is discovered by the programmer before termination of a statement or symbolic instruction (prior to entering the s, or prior to depressing the CARRIAGE RETURN key if symbolic coding), depress the LINE FEED key and retype the statement or the symbolic instruction (this applies only to input via the typewriter).

b. If an error is discovered by the programmer after termination of a statement or symbolic instruction:

(1) Depress the STOP button as soon as possible.

(2) Set Channel and Sector dials at 5000.

(3) Set PRE-SET STOP switch on 1st.

(4) Turn Switch B ON.

(5) Depress the START key.

(6) Wait until the halt at 5000 occurs.

(7) Turn PRE-SET switch OFF.

(8) (a) For all statements except the SYMBOLIC statement:
- Depress START 1 button, and wait for the ALPHA light to come on.
- Type the correct statement.

(b) For a line of symbolic coding:
- Set the Location Counter at 2665.1.
- Press the START key and wait for the ALPHA light to come on.
- Type the correct symbolic instruction.

(9) An examination of the incorrect statement or symbolic instruction may indicate that even if all or part of it should be executed by the object program, it would not affect the program. If this is so, continue with the compiling. If it is not so, either continue with the compiling, noting where a transfer can be made around the incorrect instructions when executing the object program, or initialize the SCOPAC program and begin compiling again starting at Step 14.2 of Operating Instructions.

If an error is discovered while a source program on tape is being compiled, follow the instructions in Step b(1)-(7). In Step b(8)(a) and (b) the instructions are the same, but another START button is used.
- Depress START 2 button if all the statements are to be typed out.
- Depress START 3 button if only the statements in which the SCOPAC program finds errors are to be typed out.

c. Errors detected by the SCOPAC program:

(1) ERROR
If the SCOPAC program finds an error in any statement except the Arithmetic statement such as an illegal word or an incorrect format, the word ERROR will be printed and the ALPHA light will come on ready for a statement to be entered. Type the correct statement.

(2) EQUATION ERROR
If the SCOPAC program detects an error in an Arithmetic statement the word EQUATION ERROR will be printed. This indicates that parentheses have been improperly used, that termination of names or numbers is incorrect, or illegal entries have been made. When the ALPHA light comes on, type the correct statement.

(3) SYMBOLIC CODE ERROR
If an error occurs while typing in symbolic coding, the words SYMBOLIC CODE ERROR will be typed. When the ALPHA light comes on, type the correct line of symbolic coding.

(4) Incorrect Entry of ARRAY Statement
If an ARRAY statement is not entered at the beginning of the source program as specified in the SCOPAC manual, the following statement will be typed:

TOO LATE TO ENTER ARRAYS

The computer will halt at 2610.0. Depress the START key to continue with the <u>next</u> statement.

(5) Assembly Error
If ASM ER is printed as a result of an illegal operation (such as COW instead of CLA being input in symbolic coding) it cannot be corrected by any means other than to start the entire operation over again, beginning with initializing the SCOPAC program.

14.7 Procedure When the Type-out "ASSIGNMENT TABLE IS FULL" or "NO STORAGE LEFT FOR YOUR PROGRAM" Occurs During Compiling

a. ASSIGNMENT TABLE IS FULL

During compiling if all storage in the Assignment Table is used, the following is typed:

ASSIGNMENT TABLE IS FULL

and the computer will halt at 2610.0. When this occurs, the SCOPAC Assignment Table Full tape and Sense Switch C may be used to eliminate certain types of entries from the Assignment Table. In this way storage is provided for additional entries into the Table. The position of Sense Switch C will determine the type of entries to be eliminated from the Table. With Sense Switch C ON fewer types of entries are removed than with Sense Switch C OFF.

(1) Set Sense Switch C.

(a) If Sense Switch C is ON, Alpha literals and location tags of the form TAG NN (except those needed in DO loops) will be eliminated from the Assignment Table.

(b) If Sense Switch C is OFF, Alpha literals, Floating literals, Fixed Point literals, Command literals, and location tags of the form TAG NN (except those needed in DO loops) will be eliminated from the Assignment Table.

(2) Load the Assignment Table Full tape.

Computing will begin (at location 1412.0), and those words and their assigned locations that are eliminated from the Assignment Table will be typed out. When the operation is completed, a word in command format will be typed. The computer will halt at location 6447.1. This last word indicates the number of locations now available in the Assignment Table. For example, if the last word was:

+0007240+0007720

all locations from 0724.0 to 0772.0 inclusive are now available for use in the Assignment Table. (Remember that each word of eight or less characters will require two locations in the Assignment Table. Those with more than eight require three locations.)

(3) Press START key to resume compiling after the halt at 6447.1. Compiling will resume at the point where it was interrupted when the Assignment Table became full.

(a) NO STORAGE LEFT FOR YOUR PROGRAM
If the program being compiled by SCOPAC has used all the storage available, the following statement will be typed:

NO STORAGE LEFT FOR YOUR PROGRAM

The computer will halt at location 2610.0. It may be possible to initialize SCOPAC, change the origin to allocate more storage, and recompile the entire program.

15. DESCRIPTION OF THE EXECUTION OF THE OBJECT PROGRAM

A summary of the operations during the execution of the object program are:

15.1 Loading of Program Preparation Package No. 2 (P^3-2) into memory.* (P^3-2 is normally used in conjunction with the object program.)

15.2 Loading of object program tape into memory.

15.3 Placing data tape (if used) in photoreader.

15.4 Execution of the object program.

15.5 Entry of data (if used) via typewriter during the execution of the program.

* If the statements READY, READZ, ANGLIN, PRINT, or ANGLOUT, or any Function (except SQRT) were in the source program, P^3-2 must be used.

P^3-2 occupies locations 0003_8, and 0010_8 through 2777_8.

16. PREPARATION OF DATA TAPE

16.1 Input Via READZ Statements

If the READZ statement is used to enter data, the following must be entered in the order shown for each block of data.

a. An N code (the letter N, at least 8 blanks ahead of the first number).

b. The data, with each data word consisting of a sign, integer, a decimal point code, a fraction, and an enter code.

c. Minus zero (-0.0) followed by an enter code.

d. A location of 17740 (17731 may also be used).

e. A start code.

As many blocks of data as desired may be on one tape. It is only necessary that Steps a through e be followed and an ARRAY and READZ be given in the program for each block of data. At least two inches of blank tape must separate each block of data.

A minus zero must follow the data because the data is converted to floating point form in the sequence in which it is entered and the conversion continues until a minus zero is encountered. This means that none of the data should be minus zero since the data following it would not be converted to floating point form.

16.2 Preparation of Input on VERSATAPE

All the information except the N code may be put on tape by the VERSATAPE.

a. Plug in integer - fraction, (decimal), cartridge at rear of Punch Unit.

b. Turn the Selector switch on keyboard unit to the N/C position.

c. Enter the integer, consisting of 10 digits or less, on the keyboard. Leading zeros need not be entered as they are automatically supplied by the Punch.

d. Press the sign bar (plus or minus).

e. Enter the fraction and then enter the number of zeros needed for a total of 10 digits. (Ten digits must always be entered to prevent unwanted leading zeros from being punched.)

f. Press ENTER bar.

g. Repeat Steps c through f for remaining data and the -0.

h. Turn the Selector switch to the S position.

i. Press the SET L bar.

j. Enter 17740 (17731 may also be used).

k. Press LOC bar, - location is punched, and a start code is automatically punched.

l. Via either the computer typewriter or a Flexowriter, the N code is entered at least 8 blanks ahead of the first number of <u>each block</u> of <u>data</u> on tape.*

16.3 Input Via READY Statements

The READY statement may be used to input the data. The order in which the variables are listed on tape must be the same as the order in which the corresponding variables appear in the READY statements. The data may be prepared on the Flexowriter. There must be at <u>least eight (8) blanks</u> between each character.* (The Flexowriter may be wired to automatically enter at least 8 blanks between each character as described in Technical Bulletin No. 21, Flexowriter Offline Tape Preparation of SALT Source Language and Data.) Numbers are entered in the format described in Section 18.

* The requirement that at least eight blanks be between each character applies only if the tape is to be read in via the photoreader.

17. OPERATING INSTRUCTIONS FOR EXECUTION OF THE OBJECT PROGRAM

17.1 Load P3-2.*

17.2 Load object program.

17.3 Set typewriter margins and tabs to desired settings.

17.4 If sense switches are used, set to desired settings.

17.5 If data is to be entered by tape*, place the data tape in the photo-reader. (See Section 18 for preparation of data tape.)

a. Set Location Counter to origin.** This will be 3000.0_8 unless an ORG command was used in the source program to change the origin.

b. Press the START key (not START 1, 2, or 3 buttons) to begin execution of the object program.

17.6 If input of data is from the typewriter:

a. Set Location Counter to origin (see Step 17.5 a).

b. Press the START key (not START 1, 2, or 3 buttons) to begin execution of the object program.

c. When the ALPHA light comes on, type one word of data. (See Section 18, Input of Data Via Typewriter During the Execution of the Object Program.)

* It is essential to read in P3-2 prior to the object program if a READZ statement is used in the source program. Normally data is read from tape by using a READZ statement in the source program. However, it is possible to use a READY statement in the source program for this purpose. If a READY statement is used to read data from tape, the following command changes are made in P3-2:

LOCATION	COMMAND
2315.0	+3524450+4000000
2360.0	+7377610+4100010

** To change a location: Press L key on the Console. Type in location on Console keyboard. Press ENTER key.

To change commands: Press C key on the Console. Type in commands on Console keyboard. Press ENTER key.

18. INPUT OF DATA VIA TYPEWRITER DURING EXECUTION OF THE OBJECT PROGRAM

When the ALPHA light is on, one number may be entered via the typewriter. The entry of a number may consist of several parts:

A. The sign of the number

B. The integral part of the number

C. A decimal point followed by the fractional part of the number

D. The sign and (integral) value of the power of ten by which the number is to be multiplied

E. One of the following termination characters: carriage return, space, tab, or blank.

Not all of these parts are required for every input. Any one of the following combinations is acceptable.

BCE, BE, CE, ABE, ACE, ABCE, BDE, CDE, ABDE, ACDE, ABCDE, E (yields plus zero) AE (yields signed zero).

Examples of proper formats are:

.959

-4

+3.9599

698.7634

Extremely large or small values may be entered in the following notation:

21+16 means 21×10^{16}

14-10 means 14×10^{-10}

1.6+6 means 1.6×10^{6}

The total number of decimal digits (integer and fraction) must be ≤ 12. The absolute value must be $\leq 2^{39}-1$. The exponent must be ≤ 511 in absolute value.

If an error is made prior to entering the termination character, press the LETTER SHIFT key. Type the correct number.

APPENDIX A

SUMMARY OF SCOPAC STATEMENTS

APPENDIX A

	Is Location Tag Allowed?	Page No.
CONTROL STATEMENTS (continued)		
END (terminates the Source Program)	NO	26
END s		
SYMBOLIC (enters symbolic coding)		27
SYMBOLIC s	NO	

APPENDIX B

R E C O M P I I

O P E R A T I O N S B Y A L P H A B E T I C C O D E S

and list of additional operational codes acceptable in SCOPAC programs

Alpha Code	Operation
ADD	Add
ALS	Accumulator Left Shift
ARS	Accumulator Right Shift
CFL	Copy from L Loop
CFV	Copy from V Loop
CLA	Clear and Add
CLS	Clear and Subtract
CTL	Copy to L Loop
CTV	Copy to V Loop
DIS	Display
DIV	Divide
DSL	Divide Single Length
DSR	Divide Single Length and Round
DVR	Divide and Round
EXT	Extract
FAD	Floating Add
FCA	Floating Clear and Add
FCS	Floating Clear and Subtract
FDV	Floating Divide
FMP	Floating Multiply
FNM	Floating Normalize

APPENDIX B

Alpha Code	Operation
FSB	Floating Subtract
FSQ	Floating Square Root
FST	Floating Store
HTR	Halt and Transfer
MPR	Multiply and Round
MPY	Multiply
PNA	Punch N Characters in Alphabetic Mode
PNC	Punch Character
PNW	Punch Word Command
PTA	Punch and Type N Characters in Alphabetic Mode
PTC	Punch and Type Character
PTW	Punch and Type Word
RDY	Read N Characters from Typewriter
RDZ	Read N Characters from Photoreader
SAX	Store A and Exchange A and X
SQR	Square Root
STA	Store Address
STO	Store
SUB	Subtract
TMI	Transfer on Minus
TOV	Transfer on Overflow
TPL	Transfer on Plus
TRA	Transfer
TSB	Transfer on Sense Switch B
TSC	Transfer on Sense Switch C

APPENDIX B

Alpha Code	Operation
TSD	Transfer on Sense Switch D
TYA	Type N Characters in Alphabetic Mode
TYC	Type Character
TYW	Type Word
TZE	Transfer on Zero
XAR	Exchange A and R

ADDITIONAL OPERATION CODES

ORG	Origin
BLA	Block Allocation
DEF	Definition
EQU	Equal
ALF	Alphanumeric
COM	Command
DISD	Display Decimal (BCD Format)
TYWD	Type Word Decimal (BCD Format)
PNWD	Punch Word Decimal (BCD Format)
PTWD	Punch and Type Word Decimal (BCD Format)

APPENDIX C

ASSIGNMENT TABLE

The Assignment Table is always printed after the END statement is recognized by the SCOPAC program. The Assignment Table extends from 0100_8 through 0777_8 in the SCOPAC program. The Table consists of a pair of words for each unique tag, variable, subscript, and literal constant (Floating, Fixed Point, Alpha) which has been entered in the source program or was generated by SCOPAC. Three words of the Table are used if an entry of more than eight characters is made such as a Command Literal.

The type-out of the Assignment Table consists of two columns. The first column contains the entries. The second column contains information related to the entries in the first column.

Following is an example of an Assignment Table:

START	+0000000-0030000
MAX. X	+1000000-0030640
MAX. Y	+1000000-0030220
+1.25	+4400000-0030460
+0053240+5742340	+2007770-0031220
+1+39	+4000000-0030320
+123.456789	+4407760-0030160
BACK	+0000000-0030700
END OF ASSEMBLY	

The first two digits of each word in the second column identify the type of entry in the first column.

00 means symbolic

10 means alpha literal

20 means command literal

40 means fixed point literal

44 means floating literal

The next five digits are zero unless there are more than eight characters. All the characters in excess of eight are stored in the Assignment Table at the location specified by the five digits.

The next three characters are always -00.

The last five digits specify the absolute address in the object program assigned to the entry. If the entry is a floating point number, this address will be the first of two consecutive locations which will be required by the number in the object program.

Example:

FIRST WORD	SECOND WORD
+123.456789	+4407760-0030160

In the second word:

+44 identifies the first word as a floating point number.

07760 is the location in the Assignment Table where the characters 789 of the floating point number +123.456789 are stored.

30160 is the address of the first of two consecutive locations in the object program assigned to the number +123.456789.

APPENDIX D

DESCRIPTION OF THE SCOPAC LISTING

A listing of the object program is obtained if Sense Switch B is ON. The listing consists of the symbolic and absolute coding of the object program.

Each line of the listing contains in this order:

1. one symbolic instruction or literal constant

2. the absolute address assigned to (1)

3. the contents of the location specified by the absolute address in (2).

   The contents consist of a word in command format containing:

   a. the instruction and a transfer to the next instruction

   or

   b. a literal constant.

	symbolic instruction or literal constant		absolute address	contents of the absolute address
READY FIRST s				
	TRA	READY	3000	+5722140+5730050
READY INCR s				
	FST	FIRST	3005	+3530470+5730150
	TRA	READY	3015	+5722140+5730220
READY LAST s				
	FST	INCR	3022	+3530640+5730320
	TRA	READY	3032	+5722140+5730370
DO OUTPUT X FIRST (INCR) LAST s				
	FST	LAST	3037	+3530010+5731470
	FCA	FIRST	3147	+3030470+5730550
Y:SQRT(X/3.5) s				
TAG 01	FST	X	3055	+3530170+5731650
	FCA	X	3165	+3030170+5730250
FLOATING	LITERAL	(+3.5)	3027	+7000000-0000000+0000000-0000010
	FDV	(+3.5)	3025	+0530270+5730130
	FST	STORE01	3013	+3531550+5730230
	FSQ	STORE01	3023	+4431550+5730360
CRR s				
	FST	Y	3036	+3531000+5730460
	TYC	+37	3046	+7200370+5730060
OUTPUT,PRINT Y s				
	TYC	+10	3006	+7200100+5731170
OUTPUT	FCA	Y	3117	+3031000+5731060
	TRA	PRINT	3106	+5722531+5731130
	FCA	X	3113	+3030170+5731250
	FAD	INCR	3125	+0430640+5730040
	FST	X	3004	+3530170+5732650
	FSB	LAST	3265	+0630010+5730210
	TMI	TAG 01+00001	3021	+5130551+5730600
HALT s				
	TZE	TAG 01+00001	3060	+5030551+5731600
END s				
	HALT	+3000	3160	+7730000+5732010

assignment table

FIRST	+0000000-0030470
INCR	+0000000-0030640
LAST	+0000000-0030010
TAG 01	+0000000-0030550
X	+0000000-0030170
+3.5	+4400000-0030270
STORE01	+0000000-0031550
Y	+0000000-0031000
OUTPUT	+0000000-0031170

END OF ASSEMBLY

APPENDIX E

PROGRAMMING OF SUBSCRIPTED VARIABLES

Subscripted Variables

If a reference is made to a subscripted variable, the number +65535 will be generated by SCOPAC. The use of this number assumes that the subscript of the first element is one, not zero. For example, the first element of A must be written as A_1 or $A_{1,1}$ not A_0 or $A_{0,0}$.

Example:

Correct	Incorrect
ARRAY A (9) s	ARRAY A (9) s
DO ABC Z 1 (1) 9 s	DO ABC Z 0 (1) 8 s
READY A (Z) s	READY A (Z) s
ABC, CRR s	ABC, CRR s

A reference to a subscripted variable requires that SCOPAC generate the coding necessary to compute the address of the specified element. This increases compiling time, causes location tags to be generated by SCOPAC and stored in the Assignment Table and increases the computing time of the object program. Thus it is advantageous to minimize references to subscripted variables.

For example, consider the following three equations:

```
X:A(2,3)&B(4,5)/A(1,2) s
Y:A(1,2)'2+A(2,3)&B(4,5) s
Z:B(4,5)/(A(2,3)&A(1,2)) s
```

In the three equations there are nine references to subscripted variables. This requires nine separate computations of the address of the specified elements. Each of the nine computations will have a location tag generated by SCOPAC and stored in the Assignment Table. To minimize the references to subscripted variables, the three equations could be written in the following manner:

```
U:A(2,3) s
V:B(4,5) s
W:A(1,2) s
X:U&V/W s
Y:W'2+U&V s
Z:V/(U&W) s
```

In this way there are only three references to subscripted variables. The compiling time will be less; there will be only three separate computations of the specified elements, and instead of nine, there will be only three location tags generated by SCOPAC. The variables U, V, W will be in the Assignment Table but the total number of words in the table will be less.

APPENDIX F

USE OF THE SAME MEMORY AREA FOR STORING SEVERAL ARRAYS

The same memory area may be utilized several times during the execution of the object program for storing the elements of different arrays.

There are two requirements pertaining to the size of the arrays:

1. An ARRAY statement must reserve sufficient storage for the size of the largest array that will be sharing the same memory space.

2. If both one and two dimensional arrays are to be stored in the same memory area, the maximum number of elements to be entered should be written in an ARRAY statement as subscripts of a two dimensional array. This means that even if a one dimensional array is the largest array which will be stored in a given memory space, the number of elements must be expressed as subscripts of a two dimensional array.

   Example:

   Given: Three arrays to share the same memory area,

   V (100), A (2,3), J (10,5)

   The largest array could be written as follows:

   ARRAY V (10,10) s

   If all the arrays that are to be stored in the same memory area are one dimensional it will not be necessary for the number of elements of the largest array to be expressed as subscripts of a two dimensional array.

APPENDIX F

If Input of Source Program is Via the Typewriter

Only the SCOPAC statements needed for replacing in memory the elements of one array with those of another array are given in the following example:

Given: Arrays A(25,4), CC(20,3), DDD(40)*

During the execution of the object program the elements of these arrays will occupy the same storage area in memory in the order:

Array CC, Array A, Array DDD

In the following statements, b signifies a blank.

Statements For
Source Program

ARRAY A(25,4) s	This statement reserves storage for 100 floating point values (25 x 4) - the size of the largest array. SCOPAC will generate: (1) a constant which contains the absolute address assigned to the first element of Array A. (2) a constant which is the value of the row subscript for Array A.
ARRAY ROW(1) s	Reserves storage for the values of the row subscripts of the two dimensional arrays that will occupy the same memory space.
KbbbbbCC: KbbbbbbA s	This statement insures that the location assigned to the first element of Array CC is the same as the location assigned to the first element in Array A. There must be 8 character positions in these variables, typed in the order K, blanks, and name of the array. (Blanks are entered by depressing the Blank Key.)
READZ CC s	The elements of Array CC will be read from tape into the common storage area during the execution of the object program. The first element of Array CC will be stored in the location assigned to the first element of Array A.

* In this example if all the arrays were one dimensional the statement ARRAY ROW(1) s and all READZ ROW s statements would not be used.

READZ ROW s	The row subscript (20) of Array CC will be read from tape into memory during the execution of the object program.
RbbbbbCC: ROW s	During the execution of the object program this statement will cause RbbbbbCC to be the value, 20, the row subscript of Array CC.
READZ A s	The elements of Array A will be read from tape into the common storage area during the execution of the object program.
KbbbbDDD: KbbbbbbA s	This statement insures that the location assigned to the first element of Array DDD is the same as the location assigned to the first element of Array A.
READZ DDD s	The elements of Array DDD will be read from tape into the common storage area during the execution of the object program.

The elements of the arrays, and the row subscript of Array CC, must be on a data tape in the order in which they will be called for in the object program.

The order would be:

elements of Array CC

20 (row subscript of Array CC)

elements of Array A

elements of Array DDD.

APPENDIX F

If Input of Source Program is From Paper Tape

The statements in a source program on paper tape to input arrays in the same storage, differ from those statements in a source program entered via the typewriter.

A READZ statement containing only the array name of the largest array is used to enter the elements of each array into the same storage area. (The statements such as KbbbbbCC: KbbbbbbA which may be used in source programs entered via the typewriter cannot be used in source programs on tape as the blanks would not be recognized as characters by the computer.)

The absolute location for the row subscript must be used. This location cannot be punched as a symbolic address because blanks would be required between the first character, R, and the name of the array. Blanks on paper tape are not interpreted as characters when read into the computer.

In the following example one storage area is to be utilized for several arrays. The general formula* for obtaining the absolute location of the row subscript is:

$$ABLRS = (ORG+2)_8 + \left[2(N_1+2)_{10}\right]_8$$

where

ABLRS is the absolute location of row subscript

ORG is the origin of the object program

N_1 is the total number of elements of the largest of the arrays which will occupy the same storage area.

* Formulas for determining the absolute location for the row subscript are summarized at the end of Appendix F.

APPENDIX F

Statements required for entering the elements of several arrays in the same memory area when the source program is prepared on tape are as follows:

Example:

Given: Three arrays to share the same storage area

Array A(25,4), Array CC(20,3), Array DDD(40)*

During the execution of the object program the elements of these arrays will occupy the same storage area in memory in the order:

Array CC, Array A, Array DDD.

Statements For
Source Program

ARRAY A(25,4) s	This statement reserves storage for 100 floating point values (25x4) - the size of the largest array. SCOPAC will generate: (1) a constant which contains the absolute address assigned to the first element of Array A. (2) a constant which is the value of the row subscript for Array A.
ARRAY ROW(1) s	Reserves storage for the values of the row subscripts of the two dimensional arrays that will occupy the same memory space.
READZ A s	The elements of Array CC will be read from tape into the common storage area during the execution of the object program. The first element of Array CC will be stored in the location assigned to the first element of the Array A.
READZ ROW s	The row subscript (20) of Array CC will be read from tape into memory during the execution of the object program.

* In this example if all the arrays were one dimensional the statement ARRAY ROW(1) s and all READZ ROW s statements would not be used.

SYMBOLIC s

FCA		ROW	The row subscript, 20, will be stored in the
FST	N	XXXX	absolute location XXXX during the execution of the object program. If the origin is at 3000_8, XXXX would be 3316_8.

COMPILE

READZ A s — The elements of Array A will be read from tape into the common storage area during the execution of the object program.

READZ ROW s — The row subscript, (25), of Array A will be read from tape into memory during the execution of the object program.

SYMBOLIC s

FCA		ROW	The row subscript, 25, will be stored in the
FST	N	XXXX	absolute location XXXX, (3316_8), during the execution of the object program.

COMPILE

READZ A s — The elements of Array DDD will be read into the common storage area during the execution of the object program.

All references in the source program to the arrays sharing a common storage area must be made by using only the name of the largest array. Therefore a statement such as:

Z:CC(7,2) s

would be written as

Z:A(7,2) s

The elements of the arrays and the row subscript of each two dimensional array must be on a data tape in the order in which they will be called for in the object program. The order would be:

elements of Array CC

20 (row subscript of Array CC)

elements of Array A

25 (row subscript of Array A)

elements of Array DDD

APPENDIX F

The SCOPAC program reserves storage for arrays according to certain formulas. It is possible for the programmer to use these formulas to obtain the absolute location where the value of the row subscript will be stored.

If only one storage area is to be used for several arrays, the general formula is:

$$ABLRS = (ORG+2)_8 + \left[2(N_1+2)_{10}\right]_8$$

The specific formula with the standard origin of 3000_8 and one storage area is:

$$ABLRS = 3002_8 + \left[2(N_1+2)_{10}\right]_8$$

If more than one storage area are to be used for several sets of arrays, the general formula is:

$$ABLRS = (ORG+2)_8 + \left[2(N_1+2)_{10}\right]_8 + \sum_{k=2}^{n} \left[2(N_k+3)_{10}\right]_8$$

where

ABLRS is absolute location of row subscript.

ORG is the origin of the object program.

N_1 is the total number of elements of the largest of the arrays that are to occupy the first storage area.

N_k is the total number of elements of the largest of the arrays that are to occupy the k_{th} storage area. $(k \geq 2)$

Example:

Assume the origin is 3000_8.

Provision is to be made for three storage areas, each to be occupied by more than one array. The largest arrays to be stored in each of the three storage areas are 200, 140 and 220 elements respectively. The absolute locations of the row subscripts for these arrays would be computed as follows:

<u>First Storage Area</u>

$$ABLRS = (ORG+2)_8 + \left[2(N_1+2)_{10}\right]_8$$

$$ABLRS = 3000+2+\left[2(200+2)_{10}\right]_8 = 3002+\left[(404)_{10}\right]_8$$

$$= 3002+624=3626_8$$

Second Storage Area

$$ABLRS = (ORG+2)_8+\left[2(N_1+2)_{10}\right]_8+\sum_{k=2}^{N}\left[2(N_k+3)_{10}\right]_8$$

$$ABLRS = 3000+2+\left[2(200+2)_{10}\right]_8+\left[2(140+3)_{10}\right]_8$$

$$= 3002+624+\left[(286)_{10}\right]_8$$

$$= 3626+436=4264_8$$

Third Storage Area

$$ABLRS = (ORG+2)_8+\left[2(N_1+2)_{10}\right]_8+\sum_{k=2}^{N}\left[2(N_k+3)_{10}\right]_8$$

$$ABLRS = 3000+2+\left[2(200+2)_{10}\right]_8+\left[2(140+3)_{10}\right]_8+\left[2(220+3)_{10}\right]_8$$

$$= 3002+624+436+676$$

$$= 4264+676=5162_8$$

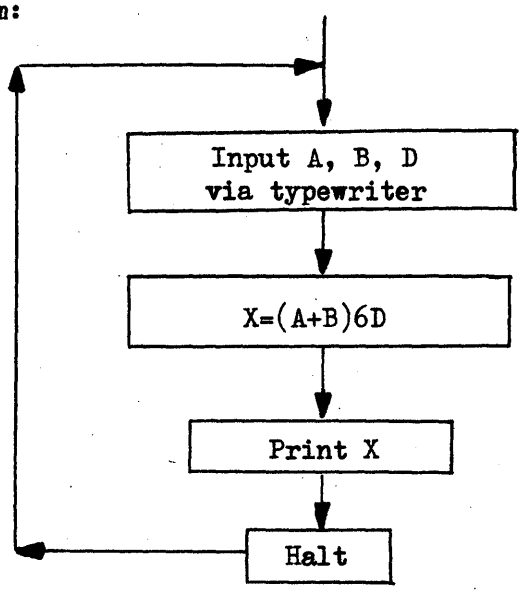
APPENDIX G

Following are examples of source programs.

EXAMPLE I

Solve: X=(A+B)6D

Flow Diagram:



Statements:

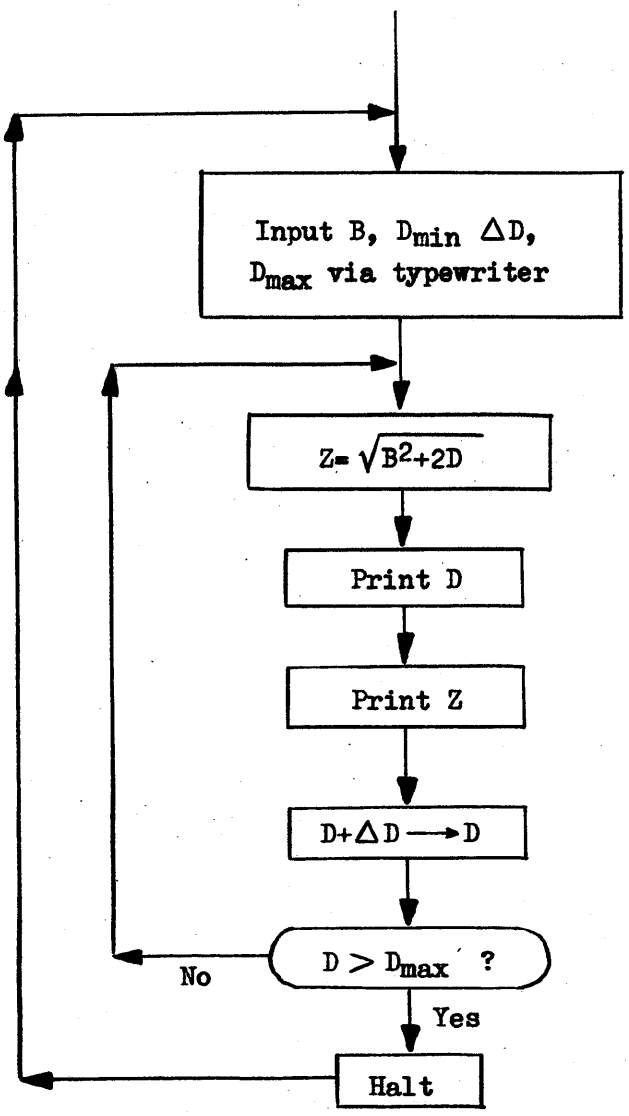
```
READY A s
READY B s
READY D s
X:(A+B)&6&D s
CRR s
PRINT X s
HALT s
END s
```

APPENDIX G

EXAMPLE II

Solve: $Z=\sqrt{B^2+2D}$ where D goes from D_{min} to D_{max} by increments of ΔD.

Flow Diagram:



APPENDIX G

Statements:

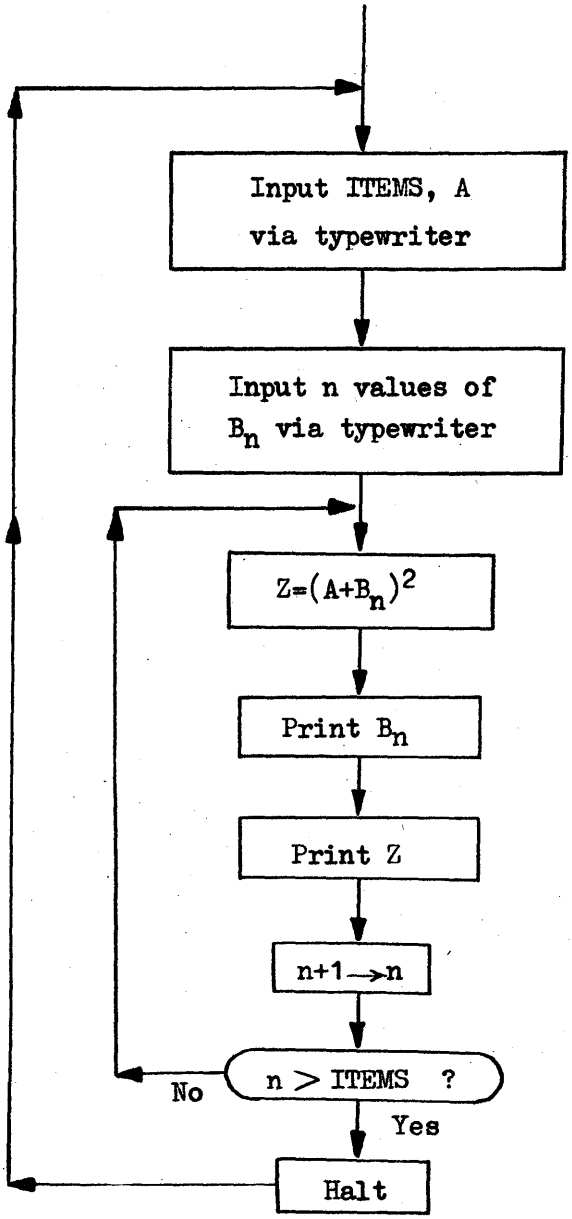
```
READY B s
READY DMIN s
READY DELTD s
READY DMAX s
CRR s
TYPE D s
TAB s
TYPE Z s
DO LOOP D DMIN (DELTD) DMAX s
Z:SQRT(B'2+2&D) s
CRR s
PRINT D s
TAB s
LOOP, PRINT Z s
HALT s
END s
```

APPENDIX G

EXAMPLE III

Solve: $Z_n=(A+B_n)^2$ Where $1 \leq n \leq 100$ and maximum n = ITEMS

Flow Diagram:



APPENDIX G

Statements:

```
ARRAY B(100) s

READY ITEMS s

READY A s

DO INPUT N 1(1)ITEMS s

READY B(N) s

INPUT, CONTINUE s

DO OUTPUT N 1(1)ITEMS s

Z:(A+B(N))'2 s

CRR s

PRINT B(N) s

TAB s

OUTPUT, PRINT Z s

HALT s

END s
```

GLOSSARY

Address - A label which identifies a register, location, or device in which information is stored.

Absolute Address - A numeric label permanently assigned to a specific location in the memory of the computer.

Symbolic Address - A label that identifies a particular word independent of the location of the word in memory.

Binary Point - An implicit point separating the integral and fractional parts of a binary number. The position of the binary point is described in terms of the number of bit positions right or left of a point (defined as binary point 0) between the sign bit and the first bit. Thus a binary point +2 defines the point as being between bit positions 2 and 3. A binary point -2 defines the point as being two bit positions to the left of binary point 0, even though bit positions to the left of the sign position do not exist.

Block - An unspecified number of words considered or transported as a unit.

Character - An elementary symbol which a computer recognizes. The symbols usually include the decimal digits 0 through 9, the letters A through Z, punctuation marks, operation symbols and any other single symbol which the computer recognizes.

Fixed-Point Decimal Number - A decimal number expressed in a system of arithmetic that requires the position of the binary point to be specified so that the number may be converted to a binary number.

Floating-Point Decimal Number - A decimal number expressed in a system of arithmetic that does <u>not</u> require the position of the binary point to be specified in order to be converted to a binary form. The decimal number is converted to a binary form consisting of a fraction and an exponent. In the RECOMP II the fraction and exponent occupy two consecutive locations.

Format - A specified method of arranging information.

Instruction - A set of characters which defines a computer operation and an address. The address specifies the word or register(s) in memory upon which the operation is to be performed.

Symbolic Instruction - An instruction consisting of an operation written in an alphabetic code, and a symbolic address.

GLOSSARY

Literal

Alpha Literal - A word to be interpreted as alphanumeric characters.

Command Literal - A word expressed in the command word format of the RECOMP II.

Fixed-Point Literal - A decimal number converted to a binary number at a specified binary point.

Floating Literal - A decimal number converted to a binary form consisting of a fraction and an exponent. In the RECOMP II the fraction and exponent occupy two consecutive locations.

Location - One storage position in the computer.

Absolute Location - Same as absolute address.

Symbolic Location - Same as symbolic address.

Memory - A storage facility forming an integral physical part of the computer. In the RECOMP II the memory consists of 4096_{10} words.

Octal - Pertaining to the number base of eight; e.g. in octal notation 603_8 is $6x8^2+0x8^1+3x8^0=387_{10}$.

Off-Line Operation - Information is prepared or processed by equipment that is <u>not</u> under the control of the computer.

Subroutine - A set of instructions which direct the computer to carry out mathematical or logical operations. It is generally included as a subunit of a program. SCOPAC statements may be used to write a subroutine.

Word - A set of characters occupying one storage location and treated by the computer as a unit.