HEWLETT hp PACKARD

HP

HP FORTRAN

reference manual

HP 2000 COMPUTER SYSTEMS

HP FORTRAN

reference manual

HEWLETT hp PACKARD

HEWLETT-PACKARD COMPANY
11000 WOLFE ROAD, CUPERTINO, CALIFORNIA, 95014

MANUAL PART NO. 02116-9015
MICROFICHE PART NO. 02116-91784

Printed: MAR 1974
Printed in U.S.A.

First Edition, Feb. 1968
Revised, April 1970
Revised, June 1971
Revised, March 1974

Printed in the U.S.A.

Fourth Edition

PREFACE

This publication is the reference manual for the HP FORTRAN programming language for the 2100 family of computers. Since Hewlett-Packard provides FORTRAN Compilers for all of its operating systems, this manual covers only the features of language, not operating procedures for the compiler. The user should refer to the appropriate system manual or operator's guide listed below:

SOFTWARE OPERATING PROCEDURES SIO SUBSYSTEMS Module (5951-1390)

DISC OPERATING SYSTEM (02116-91748)

BASIC CONTROL SYSTEM (02116-9017)

MOVING-HEAD DISC OPERATING SYSTEM (02116-91779)

MAGNETIC TAPE SYSTEM (02116-91752)

In addition, the Formatter and other relocatable subroutines used by FORTRAN programs are described in full in the *RELOCATABLE SUBROUTINES* manual (02116-91780).

NEW AND CHANGED INFORMATION

For this edition, all known errors in the HP FORTRAN manual have been corrected and some information has been eliminated to avoid repetition in several manuals. The operating procedures (previously found in Section IX) have been deleted and are now described in the *SOFTWARE OPERATING PROCEDURES SIO SUBSYSTEMS* Module (5951-1390). The Basic Control System Relocating Loader listings have been deleted from Appendix D.

CONTENTS

CONTENTS

SECTION VI
MAIN PROGRAM, FUNCTIONS, AND SUBROUTINES (cont)

CONTENTS

TABLES

ILLUSTRATIONS

INTRODUCTION

The FORTRAN compiler accepts as input, a source program written according to American Standard Basic FORTRAN specifications; it produces as output, a relocatable binary object program which can be loaded and executed under control of an HP operating system.

In addition to the ASA Basic FORTRAN language, HP FORTRAN provides a number of features which expand the flexibility of the system. Included are:

> Free Field Input: Special characters included with ASCII input data direct its formatting; a FORMAT statement need not be specified in the source program.
>
> Specification of heading and editing information in the FORMAT statement through use of the "..." notation; permits alphanumeric data to be read or written without giving the character count.
>
> Array declaration within a COMMON statement.
>
> Redefinition of its arguments and common areas by a function subprogram.
>
> Interpretation of an END statement as a RETURN statement.
>
> Basic External Functions which perform masking (Boolean) operations.
>
> Two-branch IF statement.
>
> Octal constants.

There are several versions of the HP FORTRAN Compiler; each is designed to run in a different operating environment: Software Input/Output System, Disc Operating System, etc. The operating system manuals contain descriptions of any features limited to special versions of the compiler.

SECTION I
PROGRAM FORM

A FORTRAN program is constructed of characters grouped into lines and statements.

CHARACTER SET

The program is written using the following characters:

Alphabetic:	A through Z
Numeric:	0 through 9
Special:	
	Space
=	Equals
+	Plus
-	Minus
*	Asterisk
/	Slash
(	Left Parenthesis
)	Right Parenthesis
,	Comma
.	Decimal Point
$	Dollar Sign
"	Quotation mark

Spaces may be used anywhere in the program to improve appearance; they are significant only within heading data of FORMAT statements and, in lieu of other information, in the first six positions of a line.

In addition to the above set which is used to construct source language statements, certain characters have special significance when appearing with ASCII input data.

They are the following:

space,	Data item delimiters
/	Record terminator
+ -	Sign of item
.E+-	Floating point number
@	Octal integer
"..."	Comments
←	Suppress CR-LF (output)

Details on the input data character set are given in Section VII.

LINES

A line is a sequence of up to 72 characters. On paper tape, each line is terminated by a *return*,CR, followed by a *line-feed*, LF. This terminator may be in any position following the statement information or comment contained in the line. If an error is punched on a paper tape, a *rubout* before the *return* and *linefeed* causes the entire line containing the error to be ignored.

STATEMENTS

A statement may be written in an initial line and up to five continuation lines. The statement may occupy positions 7 through 72 of these lines. The initial line contains a zero or blank in position 6. A continuation line contains any character other than zero or space in position 6 and may not contain a C in position 1.

STATEMENT LABLES

A statement may be labeled so that it may be referred to in other statements. A label consists of one to four numeric digits placed in any of the first five positions of a line. The number is unsigned and in the range of 1 through 9999. Imbedded spaces and leading zeros are ignored. If no label is used, the first five positions of the statement line must be blank. The statement label or blank follows the CR LF terminator of the previous line.

COMMENTS

Lines containing comments may be included with the statement lines; the comments are printed along with the source program listing. A comment line requires a C in position 1 and may occupy positions 2 through 72. If more than one line is used, each line requires a C indicator. Each comment line is terminated with a CR and LF.

CONTROL STATEMENT

The first statement of a program is the control statement; it defines the output to be produced by the FORTRAN compiler. The following options are available:

Relocatable binary -- The output can be loaded by the relocating loader and run.

Source Listing output -- A listing of the source program is produced.

Object Listing output -- A list of the object program is produced.

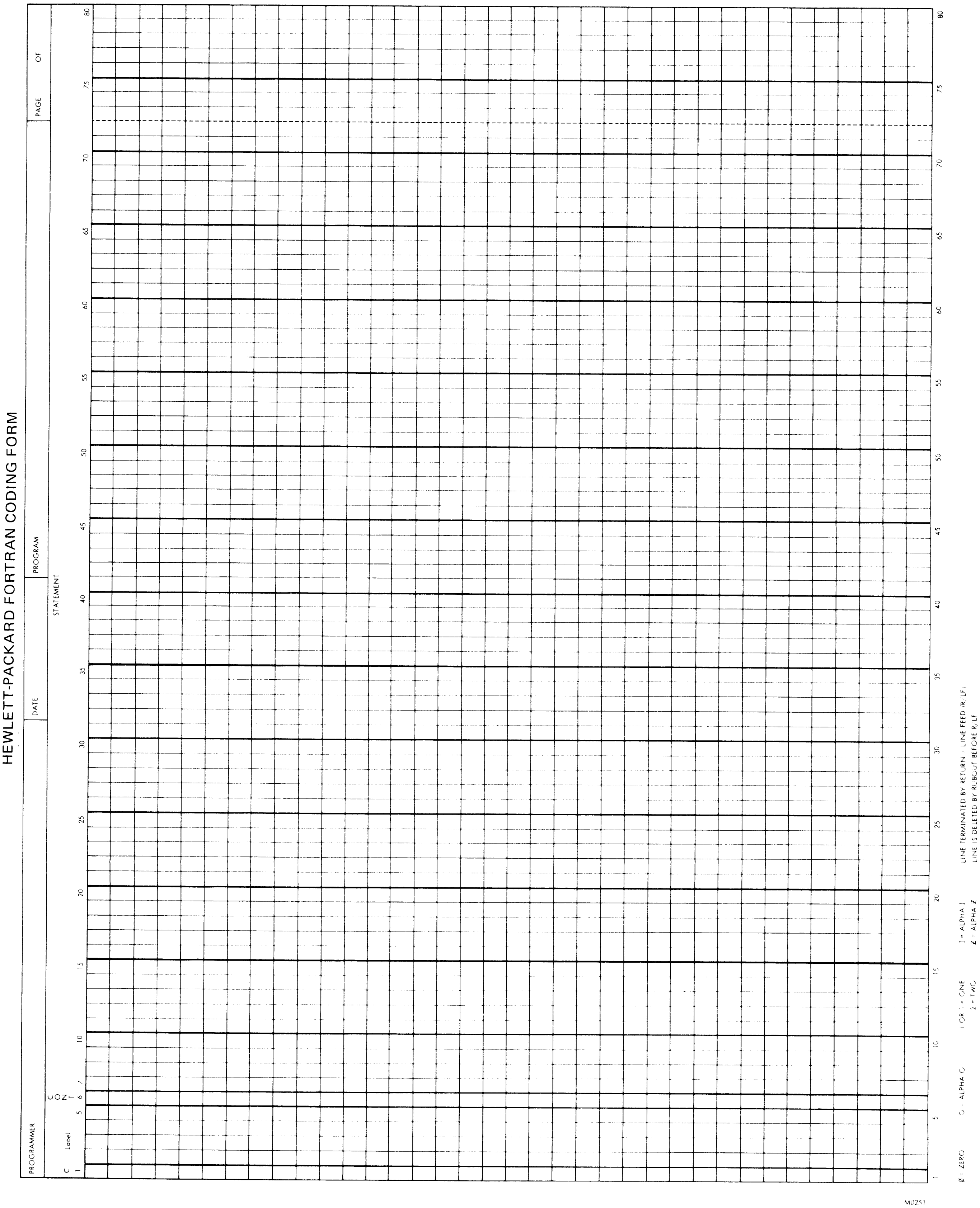
The control statement must be followed by the CR LF terminator.

END LINE

Each subprogram is terminated with an end line which consists of blanks in positions 1 through 6 and the letters E, N, and D located in any of the positions 7 through 72. The special end line, END$, signifies the end of five or less programs being compiled at one time. The end line is terminated by CR LF .

CODING FORM

The FORTRAN coding form is shown in Figure 1-1. Columns 73-80 may be used to indicate a sequence number for a line; they must not be punched on paper tape. All other columns of the form conform with line positions for paper tape.



HEWLETT-PACKARD FORTRAN CODING FORM

PROGRAMMER | DATE | PROGRAM | PAGE OF

C Label CONT STATEMENT

1
5
6
7
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80

Ø = ZERO O - ALPHA O I OR 1 = ONE I = ALPHA I LINE TERMINATED BY RETURN / LINE FEED (R/LF)
2 = TWO Z = ALPHA Z LINE IS DELETED BY RUBOUT BEFORE R/LF

M0251

Figure 1-1. Sample Coding Form (Actual Size 11 x 13-1/2)

SECTION II
ELEMENTS OF HP FORTRAN

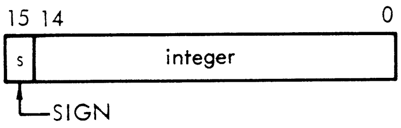
HP FORTRAN processes two types of data -- real and integer quantities. They differ in mathematical significance, constant format, and symbolic representation.

DATA TYPE PROPERTIES

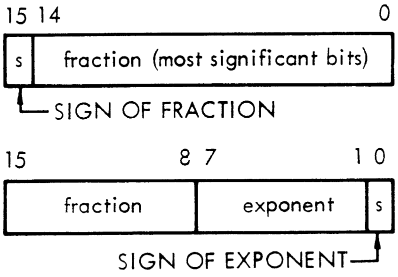
Integer and real data quantities have different ranges of values.

An integer quantity has an assumed fixed decimal point. It is represented by a 16-bit computer word with the most significant bit as the sign and the assumed decimal point on the right of the least significant bit.

An integer quantity has a range of -2^{15} to 2^{15} -1.



A real quantity has a floating decimal point; it consists of a fractional part and an exponent part. It is represented by two 16-bit computer words; the exponent and its sign are eight bits; the fraction and its sign are twenty-four bits.



It has a range in magnitude of approximately 10^{-38} to 10^{38} and may assume positive, negative, or zero values. If the fraction is negative, the number is in two's complement form. A zero value is stored as all zero bits. Precision is approximately seven decimal digits.

CONSTANTS

A constant is a value that is always defined during execution and may not be redefined. Three types of constants are used in HP FORTRAN: integer, octal (treated as integer), and real. The type of constant is determined by its form and content.

Integer

An integer constant consists of a string of up to five decimal digits. If the range -32768 to 32767 (-2^{15} to 2^{15} -1) is exceeded, a diagnostic is provided by the compiler.

Examples:

8364

1720

1872

125

3653

5932

9

31254

1

30000

Octal

Octal constants consist of up to six octal digits followed by the letter B. The form is:

$n_1 n_2 n_3 n_4 n_5 n_6$ B

n_1 is 0 or 1

n_2 - n_6 are 0 through 7

If the constant exceeds six digits, or if a non-octal digit appears, the constant is treated as zero and a compiler diagnostic is provided.

Examples:

7677B
3270B
3520B
175B
567B

7631B
5B
75026B
177776B
177777B

Real

Real constants may be expressed as an integer part, a decimal point, and a decimal fraction part. The constant may include an exponent, representing a power of ten, to be applied to the preceding quantity. The forms of real constants are:

n.n   n.   .n   n.nE±e   n.E±e   .nE±e

n is the number and e is the exponent to the base ten. The plus sign may be omitted for a positive exponent. The range of e is 0 through 38. When the exponent indicator E is followed by a + or - sign, then all digits between the sign and the next operator or delimiter are assumed to be part of the exponent expression, e.

If the range of the real constant is exceeded, the constant is treated as zero and a compiler diagnostic message occurs.

Examples:

4.512
4.
.512
4.0
4.E-10
1.

4.5E2
.45E+3
4.5E-5
0.5
.5E+37
10000.0

VARIABLES

A variable is a quantity that may change during execution; it is identified by a symbolic name. Simple and subscripted variables are recognized. A simple variable represents a single quantity; a subscripted variable represents a single quantity (element) within an array of quantities. Variables are identified by one to five alphanumeric characters; the first character must be alphabetic.

The type of variable is determined by the first character of the name. The letters I, J, K, L, M, and N, indicate an integer (fixed point) variable; any other non-numeric character indicates a real (floating point) variable. Spaces imbedded in variable names are ignored.

Simple Variable

A simple variable defines the location in which values can be stored. The value specified by the name is always the current value stored in that location.

Examples:

Integer	Real
I	ALPHA
JAIME	G13
K9	DOG
MIL	XP2
NIT	GAMMA

Subscripted Variable

A subscripted variable defines an element of an array; it consists of an alphanumeric identifier with one or two associated subscripts enclosed in parentheses. The identifier names the array; the subscripts point to the particular element. If more than two subscripts appear, a compiler diagnostic message is given.

Subscripts may be integer constants, variables, or expressions; they may have the form (exp_1, exp_2), where exp_i is one of the following:

c*v+k
c*v-k
c*v
v+k
v-k
v
k

where c and k are integer constants and v is a simple integer variable.

Examples:

Integer	Real
I(J, K)	A(J)
LAD(3, 3)	BACK(M+5, 9)
MAJOR (24*K, I+5)	OPA45(4*I)
NU (K+2)	RADI (IDEG)
NEXT (N*5)	VOLTI (,J)

ARRAYS

An array is an ordered set of data of one or two dimensions; it occupies a block of successive memory locations. It is identified by a symbolic name which may be used to refer to the entire array. An array and its dimensions must be declared at the beginning of the program in a DIMENSION or COMMON statement. The type of an array is determined by the first letter of the array name. The letters I, J, K, L, M, and N, indicate an integer array; any other letter indicates a real array.

Each element of an array may be referred to by the array name and the subscript notation. Program execution errors may result if subscripts are larger than the dimensions initially declared for the array, however, no diagnostic messages are issued.

Array Structure

Elements of arrays are stored by columns in ascending order of storage locations. An array declared as SAM(3,3), would be structured as:

Columns

	SAM(1,1)	SAM(1,2)	SAM(1,3)
Rows	SAM(2,1)	SAM(2,2)	SAM(2,3)
	SAM(3,1)	SAM(3,2)	SAM(3,3)

and would be stored as:

m	SAM(1,1)
m+1	SAM(2,1)
m+2	SAM(3,1)
m+3	SAM(1,2)
m+4	SAM(2,2)
m+5	SAM(3,2)
m+6	SAM(1,3)
m+7	SAM(2,3)
m+8	SAM(3,3)

The location of an array element with respect to the first element is a function of the subscripts, the first dimension, and the type of the array. Addresses are computed modulo 2^{15}.

Given DIMENSION A(L,M), the memory location of A(i,j) with respect to the first element, A, of the array, is given by the equation:

$$\ell = A + [i - 1 + L(j - 1)] *s$$

The quantity in brackets is the expanded subscript expression. The element size, s, is the number of storage words required for each element of the array: for integer arrays, s = 1; for real arrays, s = 2.

Array Notation

The following subscript notations are permitted for array elements:

For a two-dimensional array, A(d_1, d_2):

A(I,J)	implies A(I,J)
A(I)	implies A(I,1)
A	implies A(1,1)*

For a single-dimension array, A(d)

A(I)	implies A(I)
A	implies A(1)

The elements of a single-dimension array, A(d), however, may not be referred to as A(I,J). A diagnostic message is given by the compiler if this is attempted.

EXPRESSIONS

An expression is a constant, variable, function or a combination of these separated by operators and parentheses, written to comply with the rules for constructing the particular type of instruction. An arithmetic expression has numerical value; its type is determined by the type of the operands.

Examples:

```
A+B-C        .4+SIN(ALPHA)
X*COS(Y)     A/B+C-D*F
RALPH-ALPH   4+2*IABS(LITE)
```

**In an Input/Output list, the name of a dimensioned array implies the entire array rather than the first element.*

STATEMENTS

Statements are the basic functional units of the language. Executable statements specify actions; non-executable statements describe the characteristics and arrangement of data, editing information, statement functions, and classification of program units.

A statement may be given a numeric label of up to four digits (1 to 9999); a label allows other statements to refer to a statement. Each statement label used must be unique within the program.

SECTION III
ARITHMETIC EXPRESSIONS AND ASSIGNMENT STATEMENTS

ARITHMETIC EXPRESSIONS

An arithmetic expression may be a constant, simple or subscripted variable, or a function. Arithmetic expressions may be combined by arithmetic operators to form complex expressions.

Arithmetic operators are:

+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Exponentiation

If α is an expression, (α) is an expression. If α and β are arithmetic expressions, then the following are expressions:

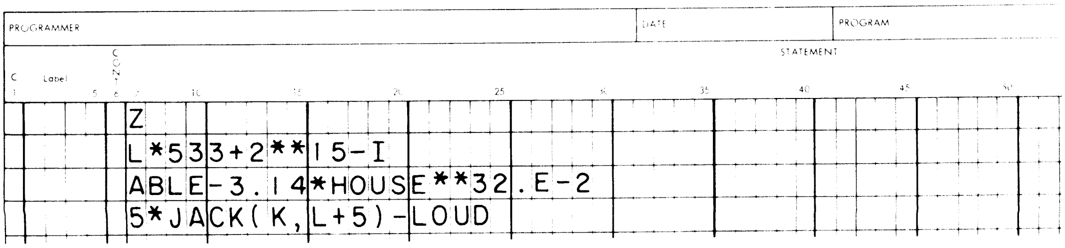
```
α +  β          α - β        α/β
α *  β            + α          - α
α ** β
```

An arithmetic expression may not contain adjoining arithmetic operators, α op op β.

Expressions of the form α**β and α**β and α**(-β) are valid; α**β**γ is not valid.

Integer overflow resulting from arithmetic operations is not detected at execution time.

Examples:



PROGRAMMER | DATE | PROGRAM

C Label CONT STATEMENT

```
Z
L*533+2**15-I
ABLE-3.14*HOUSE**32.E-2
5*JACK(K,L+5)-LOUD
```

Order of Evaluation

In general, the hierachy of arithmetic operation is:

**	exponentiation	class 1
/	division	class 2
*	multiplication	
-	subtraction	class 3
+	addition	

In an expression with no parentheses or within a pair of parentheses, evaluation basically proceeds from left to right, or in the above order if adjacent operators are in a different class.

When writing an integer expression it is important to remember not only the left to right scanning process, but also that dividing an integer quantity by an integer quantity yields a truncated result; thus 11/3 = 3. The expression I*J/K may yield a different result than the expression J/K*I. For example, 4*3/2 = 6; but 3/2*4 = 4.

Expressions enclosed in parentheses and function references are evaluated as they are encountered from left to right.

Examples:

In the examples below, s_1, s_2,..., s_n indicate intermediate results during the evaluation of the expression; the symbol → can be interpreted as "goes to".

a) Evaluation of class 1 precedes class 3

A+B**C-D

$B**C \rightarrow s_1$

$s_1+A \rightarrow s_2$

$s_2-D \rightarrow s_3$ $\qquad$ s_3 is the evaluated expression

b) Evaluation of class 2 precedes class 3

A*B*C/D+E*F-G/H

$A*B \rightarrow s_1$

$s_1*C \rightarrow s_2$

$s_2/D \rightarrow s_3$

$E*F \rightarrow s_4$

$s_4+s_3 \rightarrow s_5$

$G/H \rightarrow s_6$

$-s_6 \rightarrow s_7$

$s_7+s_5 \rightarrow s_8$ s_8 is the evaluated expression

c) Evaluation of an expression including a function is performed.

A+B**C+D+COS(E)

$B**C \rightarrow s_1$

$A+s_1 \rightarrow s_2$

$s_2+D \rightarrow s_3$

$COS(E) \rightarrow s_4$

$s_4+s_3 \rightarrow s_5$ s_5 is the evaluated expression

d) Parentheses can control the order of evaluation

A*B/C+D

$A*B \rightarrow s_1$

$s_1/C \rightarrow s_2$

$s_2+D \rightarrow s_3$ s_3 is the evaluated expression

A*B/(C+D)

$A*B \rightarrow s_1$

$C+D \rightarrow s_2$

$s_1/s_2 \rightarrow s_3$ s_3 is the evaluated expression

e) If more than one pair of parentheses or if an exponential expression appears, evaluation is performed left to right.

A+B**C-(D*E+F)+(G-H*P)

$B**C \to s_1$

$s_1+A \to s_2$

$D*E \to s_3$

$s_3+F \to s_4$

$-s_4 \to s_5$

$s_5+s_2 \to s_6$

$H*P \to s_7$

$-s_7 \to s_8$

$s_8+G \to s_9$

$s_9+s_6 \to s_{10}$ $\quad s_{10}$ is the evaluated expression

Type of Expression

With the exception of exponentiation and function arguments, all operands within an expression must be of the same type. An expression is either real or integer depending on the type of all of its constituent elements.

If either an integer or real operand is exponentiated by an integer operand, the resultant element is of the same type as that of the operand being exponentiated. If both operands are real, the resultant element is real.

Examples:

J**I	integer
A**I	real
A**B	real

An integer exponentiated by a real operand is not valid.

ASSIGNMENT STATEMENTS

An arithmetic assignment statement is of the form:

v = e

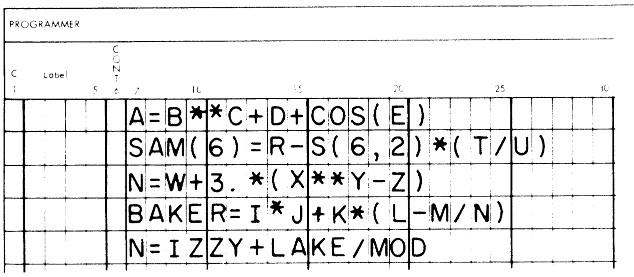
The variable, v , may be simple or subscripted; e is an expression. Execution of this statement causes the evaluation of the expression, e , and the assignment of the value to the variable.

Type of Statement

The processing of the evaluated expression is performed according to the following table:

Type of v	Type of e	Assignment rule
Integer	Integer	Transmit e to v without change.
Integer	Real	Truncate and transfer as integer to v.
Real	Integer	Transform integer form of e to floating decimal and transfer to v.
Real	Real	Transmit e to v without change.

Examples:


PROGRAMMER
C
Label
CONT

`A=B**C+D+COS(E)`	Transmit without change
`SAM(6)=R-S(6,2)*(T/U)`	Transmit without change
`N=W+3.*(X**Y-Z)`	Truncate
`BAKER=I*J+K*(L-M/N)`	Convert to real
`N=IZZY+LAKE/MOD`	Transmit without change

MASKING OPERATIONS

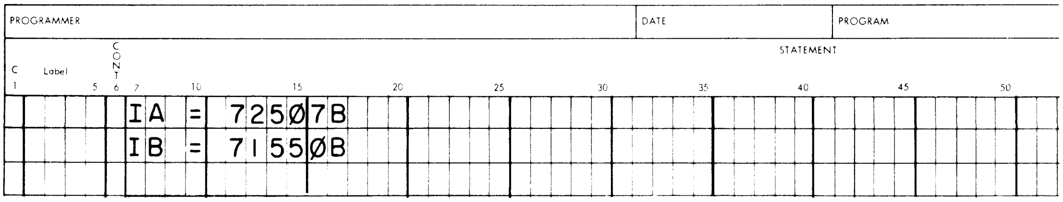
In HP FORTRAN, masking operations may be performed using the Basic External Functions IAND, IOR, and NOT. (See Section VI.) These functions are as follows:

IAND	Form the bit-by-bit logical product of two operands
IOR	Form the bit-by-bit logical sum of two operands
NOT	Complement the operand

The operations are described by the following table:

Value of Arguments		Value of Function		
a_1	a_2	IAND (a_1 , a_2)	IOR (a_1 , a_2)	NOT (a_1)
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

Examples:



PROGRAMMER | DATE | PROGRAM

C Label CONT STATEMENT

1 5 6 7 10 15 20 25 30 35 40 45 50

```
      IA = 725Ø7B
      IB = 7155ØB
```

IAND (IA, IB) is 70500B

IOR (IA, IB) is 73557B

NOT (IA) is 105270B

SECTION IV
SPECIFICATIONS STATEMENTS

The Specifications statements, which include DIMENSION, COMMON, and EQUIVALENCE, define characteristics and arrangement of the data to be processed. These statements are non-executable; they do not produce machine instructions in the object program. The statements must all appear before the first executable statement in the following order: DIMENSION, COMMON, and EQUIVALENCE.

DIMENSION

The DIMENSION statement reserves storage for one or more arrays.

DIMENSION v_1 (i_1), v_2 (i_2),..., v_n (i_n)

An array declarator, $v_j(i_j)$; defines the name of an array, v_j, and its associated dimensions, (i_j). The declarator subscript, i, may be an integer constant or two integer constants separated by a comma. The magnitude of the values given for the subscripts indicates the maximum value that the subscript may attain in any reference to the array.

The number of computer words reserved for a given array is determined by the product of the subscripts and the type of the array name. For integer arrays, the number of words equals the number of elements in the array. For real arrays, two words are used for each element; the storage area is twice the product of the subscripts.

A diagnostic message is printed if an array size exceeds 2^{15} -1 locations.

Examples:

```
DIMENSION SAM (5, 10), ROGER (10, 10), NILE (5, 20)

Area reserved for SAM            5*10*2 = 100 words
Area reserved for ROGER         10*10*2 = 200 words
Area reserved for NILE           5*20*1 = 100 words
```

COMMON

The COMMON statement reserves a block of storage that can be referenced by the main program and one or more subprograms. The areas of common information are specified by the statement form:

COMMON $a_1, a_2, \ldots, a_n$

Each area element, a_i, identifies a segment of the block for the subprogram in which the COMMON statement appears. The area elements may be simple variable identifiers, array names, or array declarators (dimensioned array names).

If dimensions for an array appear both in a COMMON statement and a DIMENSION statement, those in the DIMENSION statement will be used.

Any number of COMMON statements may appear in a subprogram section (preceding the first executable statement). The order of the arrays in common storage is determined by the order of the COMMON statements and the order of the area elements within the statements. All elements are stored contiguously in one block.

At the beginning of program execution, the contents of the common block are undefined; the data may be stored in the block by input/output or assignment statements.

Examples:

```
COMMON I(5), A(6), B(4)
```

Area reserved for I =	5 words
Area reserved for A =	12 words
Area reserved for B =	8 words
Common area	25 words

	Common Block
Origin	I (1)
	I (2)
	I (3)
	I (4)
	I (5)
	A (1)
	A (1)
	A (2)
	A (2)
	A (3)
	A (3)
	A (4)
	A (4)
	A (5)
	A (5)
	A (6)
	A (6)
	B (1)
	B (1)
	B (2)
	B (2)
	B (3)
	B (3)
	B (4)
	B (4)

Correspondence of Common Blocks

Each subprogram that uses the common block must include a COMMON statement. Each subprogram may assign different variable and array names, and different array dimensions, however, if corresponding quantities are to agree, the types should be the same for corresponding positions in the block.

Examples:

```
MAIN PROG COMMON I(5), A(6), B(4)
      .
      .
      .
SUBPROG1 COMMON J(3), K(2), C(5), D(5)
```

MAIN PROG reference	Common Block	SUBPROG1 reference
I (1)	integer 1	J (1)
I (2)	integer 2	J (2)
I (3)	integer 3	J (3)
I (4)	integer 4	K (1)
I (5)	integer 5	K (2)
A (1)	real 1	C (1)
A (1)	real 1	C (1)
A (2)	real 2	C (2)
A (2)	real 2	C (2)
A (3)	real 3	C (3)
A (3)	real 3	C (3)
A (4)	real 4	C (4)
A (4)	real 4	C (4)
A (5)	real 5	C (5)
A (5)	real 5	C (5)
A (6)	real 6	D (1)
A (6)	real 6	D (1)
B (1)	real 7	D (2)
B (1)	real 7	D (2)
B (2)	real 8	D (3)
B (2)	real 8	D (3)
B (3)	real 9	D (4)
B (3)	real 9	D (4)
B (4)	real 10	D (5)
B (4)	real 10	D (5)

If portions of a common block are not referred to by a particular subprogram, dummy variables may be used to provide correspondence in reserved areas.

Examples:

```
MAIN PROG COMMON I(5), A(6), B(4)
          .
          .
          .
SUBPROG2 COMMON J(17), B(4)
```

MAIN PROG reference	Common Block	SUBPROG2 reference
I (1)	integer 1	J (1)
I (2)	integer 2	J (2)
I (3)	integer 3	J (3)
I (4)	integer 4	J (4)
I (5)	integer 5	J (5)
A (1)	real 1	J (6)
A (1)	real 1	J (7)
A (2)	real 2	J (8)
A (2)	real 2	J (9)
A (3)	real 3	J (10)
A (3)	real 3	J (11)
A (4)	real 4	J (12)
A (4)	real 4	J (13)
A (5)	real 5	J (14)
A (5)	real 5	J (15)
A (6)	real 6	J (16)
A (6)	real 6	J (17)
B (1)	real 7	B (1)
B (1)	real 7	B (1)
B (2)	real 8	B (2)
B (2)	real 8	B (2)
B (3)	real 9	B (3)
B (3)	real 9	B (3)
B (4)	real 10	B (4)
B (4)	real 10	B (4)

J (17) is a dummy array. It is not referenced in SUBPROG 2 but provides proper correspondence in reserved areas so that SUBPROG 2 can refer to array B.

The length of the common block may differ in different subprograms, however, the subprogram (or main program) with the longest common block must be the first to be loaded at execution time.

EQUIVALENCE

The EQUIVALENCE statement permits sharing of storage by two or more entities. The statement has the form:

$$\text{EQUIVALENCE } (k_1), (k_2), \ldots, (k_n)$$

in which each k is a list of the form:

$$a_1, a_2, \ldots, a_m$$

Each a is either a variable name or a subscripted variable; the subscript of which contains only constants. The number of subscripts must correspond to the number of subscripts for the related array declarator.

All names in the list may be used to represent the same location. If an equivalence is established between elements of two or more arrays, there is a corresponding equivalence between other elements of the arrays; the arrays share some storage locations. The lengths may be different or equal.

Examples:

```
DIMENSION A(5), B(4)
EQUIVALENCE (A (4), B (2))
```

Array 1 Name	Array 2 Name	Quantity Element
A (1)		real 1
		real 1
A (2)		real 2
		real 2
A (3)	B (1)	real 3
		real 3
A (4)	B (2)	real 4
		real 4
A (5)	B (3)	real 5
		real 5
	B (4)	real 6
		real 6

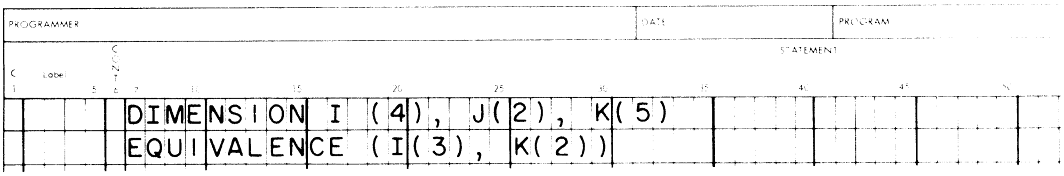
The EQUIVALENCE statement establishes that the names A(4) and B(2) identify the fourth real quantity. The statements also establish a similar correspondence between A(3) and B(1), and A(5) and B(3).

An integer array/or variable may be made equivalent to a real array or variable; equivalence may be established between different types. The variables may be with or without subscripts.

The effect of an EQUIVALENCE statement depends on whether or not the variables are assigned to the common block. When two variables or array elements share storage, the symbolic names of the variables or arrays may not both appear in COMMON statements in the same subprogram. The assignment of storage to variables and arrays declared in a COMMON statement is determined on the basis of their type and the array declarator. Entities so declared are always contiguous according to the order in the COMMON statement. The EQUIVALENCE statement must not alter the origin of the common block, but arrays may be defined so that the length of the common block is increased.

Examples:

a) Effect of EQUIVALENCE, variables not in common block:

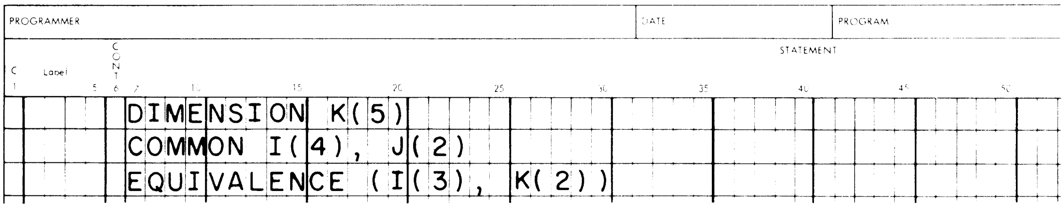

PROGRAMMER
DATE
PROGRAM
STATEMENT

```
      DIMENSION I(4), J(2), K(5)
      EQUIVALENCE (I(3), K(2))
```

storage is assigned as follows:

Arrays		Quantities
I(1)		integer 1
I(2)	K(1)	integer 2
I(3)	K(2)	integer 3
I(4)	K(3)	integer 4
	K(4)	integer 5
	K(5)	integer 6
J(1)		integer 7
J(2)		integer 8

b) Effect of EQUIVALENCE, some variables in common block:



PROGRAMMER DATE PROGRAM

C Label CONT STATEMENT

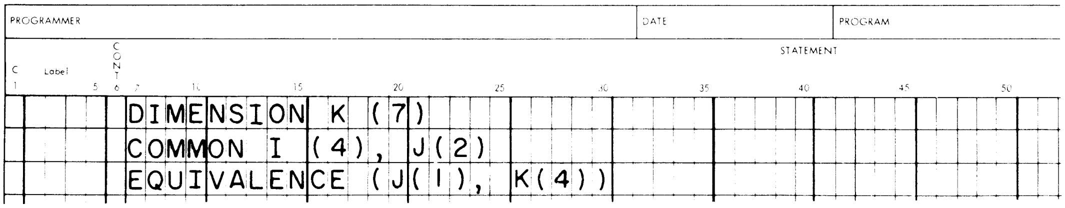
1 5 6 7 10 15 20 25 30 35 40 45 50

```
      DIMENSION K(5)
      COMMON I(4), J(2)
      EQUIVALENCE (I(3), K(2))
```

storage is assigned as follows:

Arrays		Quantities	
I(1)		integer 1	Common block
I(2)	K(1)	integer 2	
I(3)	K(2)	integer 3	
I(4)	K(3)	integer 4	
J(1)	K(4)	integer 5	
J(2)	K(5)	integer 6	

c) Effect of EQUIVALENCE on the length of the common block:



PROGRAMMER DATE PROGRAM

C Label CONT STATEMENT

1 5 6 7 10 15 20 25 30 35 40 45 50

```
      DIMENSION K (7)
      COMMON I (4), J(2)
      EQUIVALENCE (J(1), K(4))
```

storage is assigned as follows:

Arrays		Quantities	
I(1)		integer 1	common block
I(2)	K(1)	integer 2	
I(3)	K(2)	integer 3	
I(4)	K(3)	integer 4	
J(1)	K(4)	integer 5	
J(2)	K(5)	integer 6	
	K(6)	integer 7	
	K(7)	integer 8	

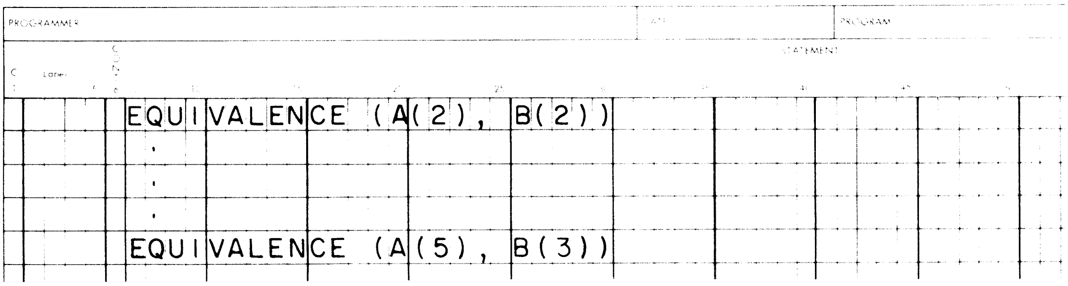
The value of the subscripts for an array being made equivalent to another array should not be such that the origin of the common block is changed (for example, EQUIVALENCE (I(3), K(4)).

	Arrays		Quantities
	K(1)	← origin changed	integer 1
origin →	I(1) K(2)		integer 2
	I(2) K(3)		integer 3
	I(3) K(4)		integer 4
	I(4) K(5)		integer 5
	J(1) K(6)		integer 6
	J(2) K(7)		integer 7

If contradictory EQUIVALENCE relationships are specified, a diagnostic message is printed.

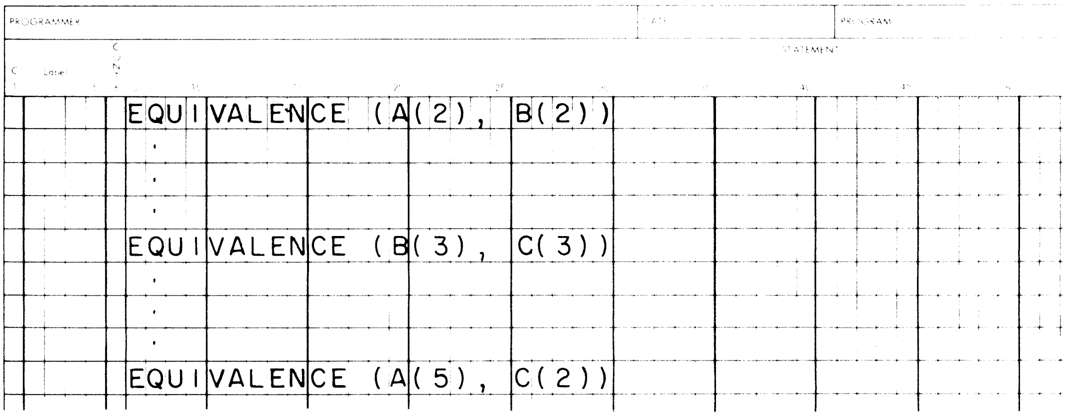
Example:

a)



```
      EQUIVALENCE (A(2), B(2))
      .
      .
      .
      EQUIVALENCE (A(5), B(3))
```

b)



```
      EQUIVALENCE (A(2), B(2))
      .
      .
      .
      EQUIVALENCE (B(3), C(3))
      .
      .
      .
      EQUIVALENCE (A(5), C(2))
```

SECTION V
CONTROL STATEMENTS

Program execution normally proceeds from statement to statement as they appear in the program. Control statements can be used to alter this sequence or cause a number of iterations of a program section. Control may be transferred to an executable statement only; a transfer to a non-executable statement will result in a program error which is usually recognized during compilation as a transfer to an undefined label.* With the DO statement, a predetermined sequence of instructions can be repeated a number of times with the stepping of a simple integer variable after each iteration.

Statements are labelled by unsigned numbers, 1 through 9999, which can be referred to from other sections of the program. A label up to four digits long precedes the FORTRAN statement and is separated from it by at least one blank or a zero. Imbedded blanks and leading zeros in the label are ignored: 1, 01, 0 1, 0001 are identical.

GO TO STATEMENTS

GO TO statements provide transfer of control.

GO TO k

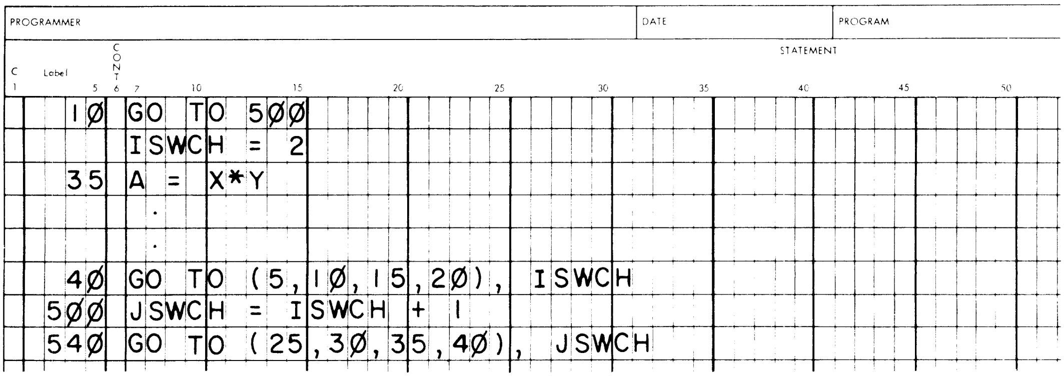
This statement, an unconditional GO TO, causes the transfer of control to the statement labelled k.

GO TO $(k_1,k_2,\ldots,k_n)$,i

This statement, a computed GO TO, acts as a many-branched transfer. The k's are statement labels and i is a simple integer variable. Execution of this statement causes the statement identified by the label k_j to be executed next, where j is the value of i at the time of execution, and $1 \le j \le n$. If $i < 1$, a transfer to k_1 occurs; if $i > n$, a transfer to k_n occurs.

**A transfer to a FORMAT statement is not detectable during compilation; if such an error occurs, no diagnostic message is produced.*

Examples:



PROGRAMMER | DATE | PROGRAM

C Label CONT | STATEMENT

```
1    5 6 7  10   15   20   25   30   35   40   45   50
   1Ø  GO TO 5ØØ
       ISWCH = 2
   35  A = X*Y
       .
       .
   4Ø  GO TO (5,1Ø,15,2Ø), ISWCH
  5ØØ  JSWCH = ISWCH + 1
  54Ø  GO TO (25,3Ø,35,4Ø), JSWCH
```

At statement 40, control transfers to statement 10, which is an unconditional transfer to statement 500. At 540 control transfers to statement 35.

IF STATEMENTS

The arithmetic IF statement provides conditional transfer of control

$$\text{IF (e)}k_1,k_2,k_3$$

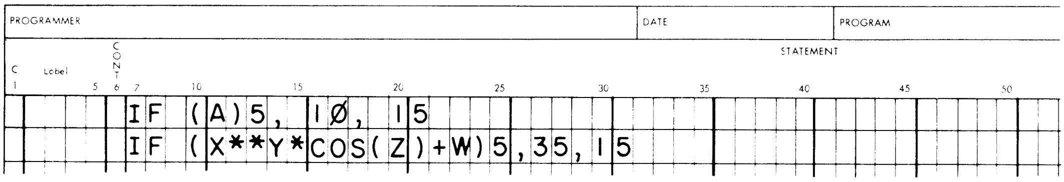
The e is an arithmetic expression and the k's are statement labels. The arithmetic IF is a three-way branch. Execution of this statement causes evaluation of the expression and transfer of control depending on the following conditions:

$e < 0$, go to k_1

$e = 0$, go to k_2

$e > 0$, go to k_3

Examples:



PROGRAMMER | DATE | PROGRAM

C Label CONT | STATEMENT

```
1    5 6 7  10   15   20   25   30   35   40   45   50
       IF (A)5, 1Ø, 15
       IF (X**Y*COS(Z)+W)5,35,15
```

The logical IF statement provides conditional transfer of control to either of two statements:

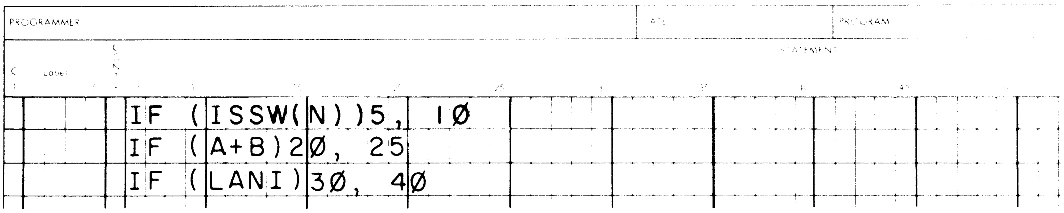
$$\text{IF } (e)k_1,k_2$$

The e is an arithmetic expression that may yield a negative or non-negative (positive or zero) value. Execution of this statement causes evaluation of the expression and transfer of control under the following conditions:

$e < 0$, go to k_1

$e \geq 0$, go to k_2

Examples:



```
IF (ISSW(N))5, 1Ø
IF (A+B)2Ø, 25
IF (LANI)3Ø, 4Ø
```

DO STATEMENTS

A DO statement makes it possible to repeat a group of statements.

$$\text{DO } n\ i = m_1,m_2,m_3$$

or

$$\text{DO } n\ i = m_1,m_2$$

The n is the label of an executable statement which ends the group of statements. The statement, called the terminal statement, must physically follow the DO statement in the source program. It may not be a GO TO of any form, IF, RETURN, STOP, PAUSE, or DO statement.

The i is the control variable; it may be a simple integer variable.

The m's are indexing parameters: m_1 is the initial parameter; m_2, the terminal parameter; and m_3, the incrementation parameter. They may be unsigned integer constants or simple integer variables. At time of execution, they all must be greater than zero. If m_3 does not appear (second form), the incrementation value is assumed to be 1.

A DO statement defines a loop, as shown in the flowchart of Figure 5-1. Associated with each DO statement is a range that is defined to be those executable statements following the DO, to and including the terminal statement associated with the DO. At time of execution, the following steps occur:

1. The control variable is assigned the value of the initial parameter.
2. The range of the DO is executed.
3. The terminal statement is executed and the control variable is increased by the value of the incrementation parameter.
4. The control variable is compared with the terminal parameter. If less than or equal to the terminal parameter, the sequence is repeated starting at step 2. If the control variable exceeds the terminal parameter, the DO loop is satisfied and control transfers to the statement following n. The control variable becomes undefined.

Should m_1 exceed m_2 on the initial entry to the loop, the range of the DO is executed and control passes to the statement after n. If a transfer out of the DO loop occurs before the DO is satisfied, the current value of the control variable is preserved. The control variable, initial parameters, terminal parameter, and incrementation parameters may not be redefined during the execution of the range of the DO loop.

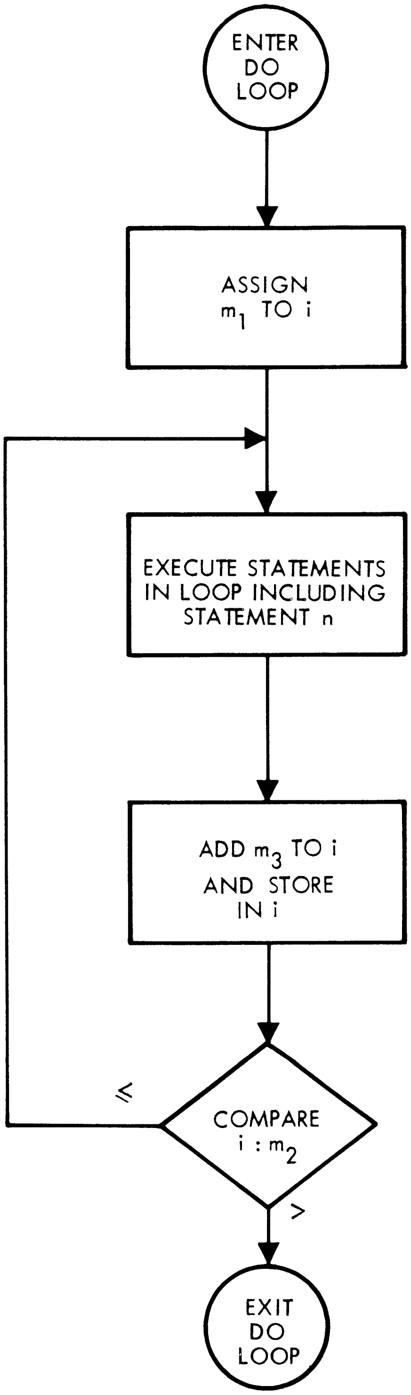
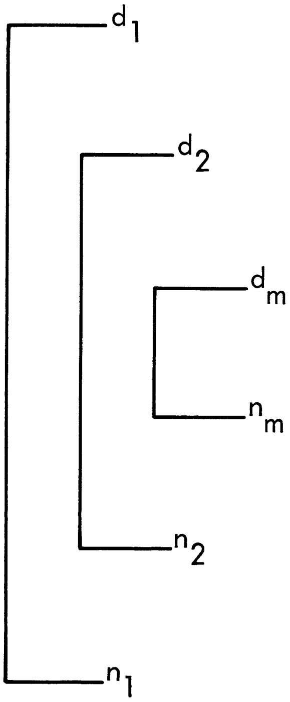


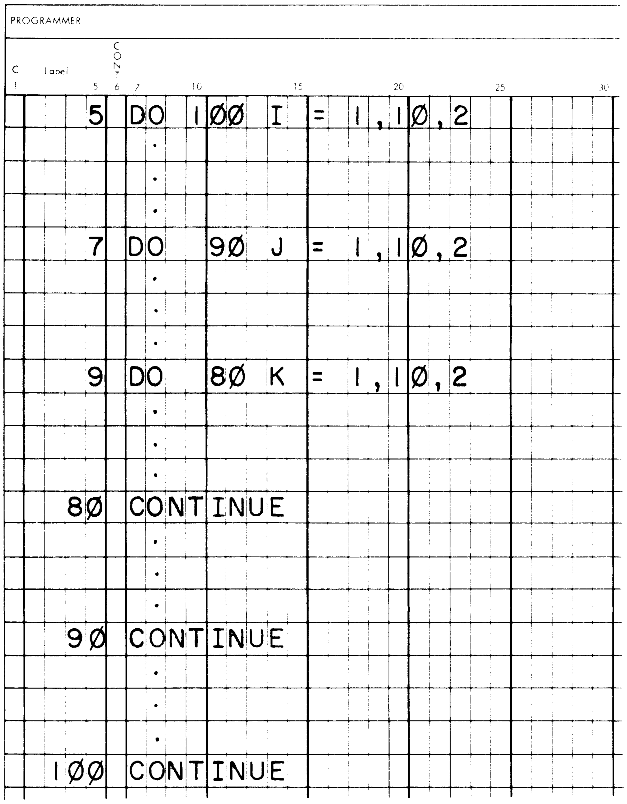
Figure 5-1. Example of a DO Loop

DO Nests

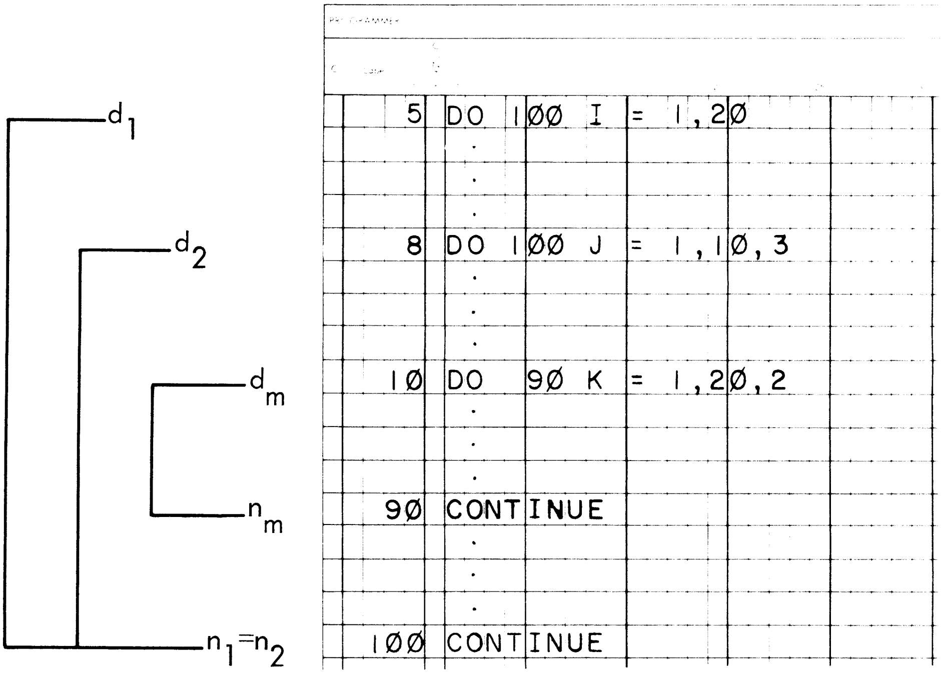
When the range of a DO loop contains another DO loop, the latter is said to be nested. DO loops may be nested 10 deep. The last statement of a nested DO loop must be the same as the last statement of the outer loop or occur before it. If $d_1, d_2, \ldots, d_n$ are DO statements, which appear in the order indicated by the subscripts; and if n_1, $n_2, \ldots, n_m$ are the respective terminal statements, then n_m must appear before or be the same as n_{m-1}, n_{m-1} must appear before or be the same as n_2, and n_2 must appear before or be the same as n_1.

Examples:

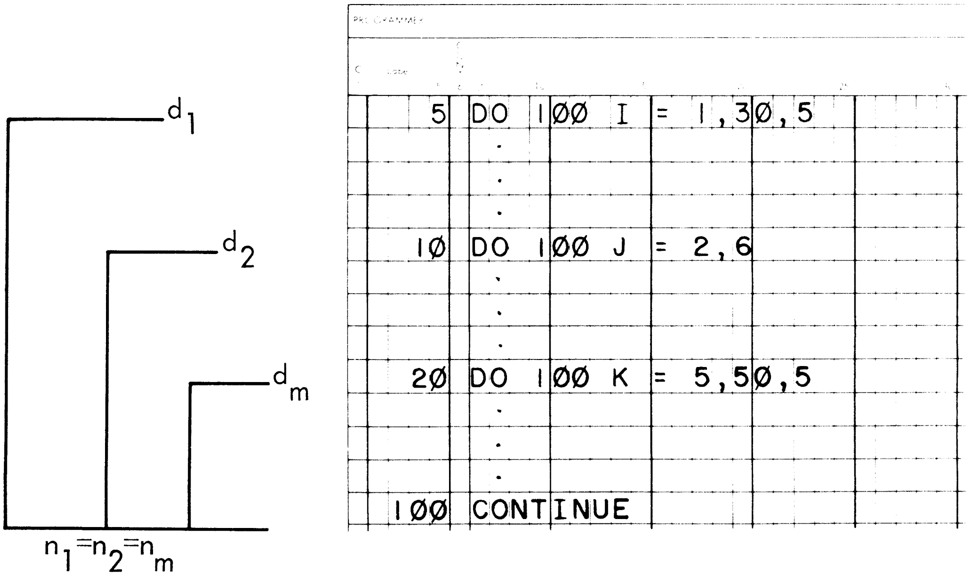



PROGRAMMER
C
Label
CONT
1
5
6
7
10
15
20
25
30

```
    5 DO 1ØØ I = 1,1Ø,2
      .
      .
      .
    7 DO 9Ø J = 1,1Ø,2
      .
      .
      .
    9 DO 8Ø K = 1,1Ø,2
      .
      .
      .
   8Ø CONTINUE
      .
      .
      .
   9Ø CONTINUE
      .
      .
      .
  1ØØ CONTINUE
```



```
    5 DO 1ØØ I = 1,2Ø
      .
      .
      .
    8 DO 1ØØ J = 1,1Ø,3
      .
      .
      .
   1Ø DO 9Ø K = 1,2Ø,2
      .
      .
      .
   9Ø CONTINUE
      .
      .
      .
  1ØØ CONTINUE
```

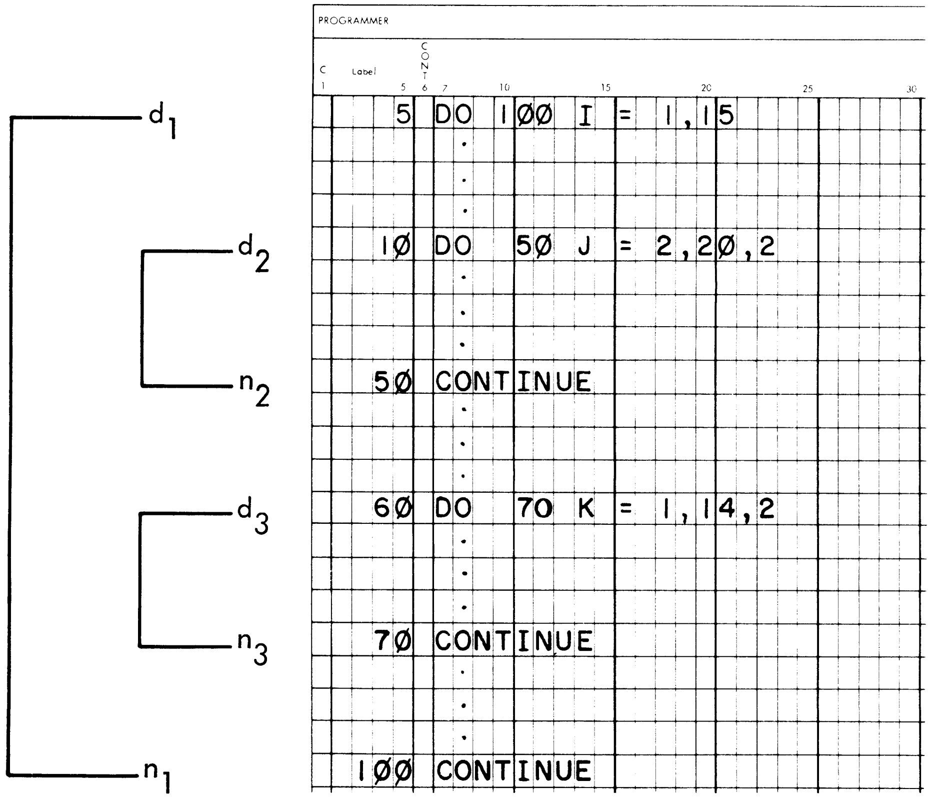


```
    5 DO 1ØØ I = 1,3Ø,5
      .
      .
      .
   1Ø DO 1ØØ J = 2,6
      .
      .
      .
   2Ø DO 1ØØ K = 5,5Ø,5
      .
      .
      .
  1ØØ CONTINUE
```

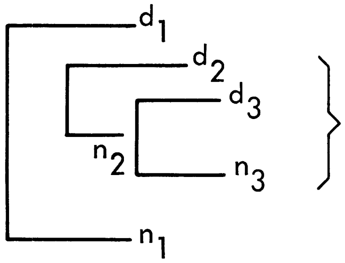
If one or more nested loops have the same terminal statement, when the inner DO is satisfied, the control variable for the next outer loop is incremented and tested against its associated terminal parameter. Control transfers to the statement following the terminal statement only when all related loops are satisfied.

DO loops may be nested in common with other loops as long as their ranges do not overlap.

Examples:



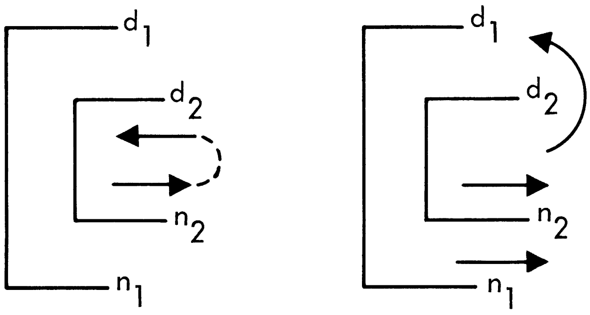
```
    5 DO 1ØØ I = 1,15
        .
        .
        .
   1Ø DO 5Ø J = 2,2Ø,2
        .
        .
        .
   5Ø CONTINUE
        .
        .
        .
   6Ø DO 7Ø K = 1,14,2
        .
        .
        .
   7Ø CONTINUE
        .
        .
        .
  1ØØ CONTINUE
```



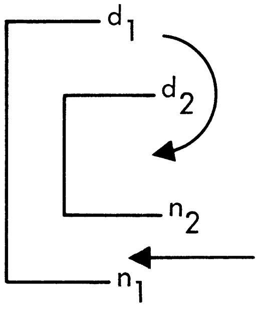
Invalid, ranges overlap

In a DO nest, a transfer may be made from an inner loop into an outer loop, and transfer is permissible outside of the loop. It is illegal, however, for a GO TO or IF to initiate a transfer of control from outside of the range of a DO into its range.

When nested DO loops have the same terminal statement, a transfer to that terminal statement causes a transfer to the innermost logs of the nest. When this transfer occurs, the current value of the control variable for the innermost loop is incremented and that loop is executed until its range is satisfied, etc.



VALID
TRANSFERS



INVALID
TRANSFERS

CONTINUE

This statement acts as no-operation instruction.

```
CONTINUE
```

The CONTINUE statement is most frequently used as the last statement of a DO loop to provide a loop termination when a GO TO or IF would normally be the last statement of the loop. If used elsewhere in the source program, it acts as a do-nothing instruction and control passes to the next sequential program statement.

PAUSE

This statement provides a temporary program halt.

```
PAUSE n
  or
PAUSE
```

n may be up to four octal digits (without a B suffix) in the range 0 to 7777. This statement halts the execution of the program and types PAUSE on the standard output unit. The value of n, if given is in the A-Register. Program execution resumes at the next statement.

STOP

The STOP statement terminates the execution of the program.

```
        STOP n
          or
        STOP
```

n may be up to four octal digits (without a B suffix) in the range 0 to 7777. This statement halts the execution of the program and types STOP on the standard output unit. The value of n, if given, is in the A-Register.

END

The END statement indicates the physical end of a program or subprogram. It has the form:

```
        END name
```

The END statement is required for every program or subprogram. The name of the program can be included, but it is ignored by the compiler. The END statement is executable in the sense that it will effect return from a subprogram in the absence of a RETURN statement. An END statement may be labelled and may serve as a junction point.

END$

The END$ statement indicates the physical end of five or less programs or subprograms that are to be compiled at one time. If there are four or less programs, the statement is printed on the source program listing. If there are exactly five, the statement is not printed. If more than five programs are on the same tape, the END$ may be omitted after the fifth program; the compiler stops accepting input after the fifth is processed.

SECTION VI
MAIN PROGRAM, FUNCTIONS, AND SUBROUTINES

A FORTRAN program consists of a main program with or without subprograms. Subprograms, which are either functions or subroutines, are sets of statements that may be written and compiled separately from the main program.

The main program calls or references subprograms; and subprograms may call or reference other subprograms as long as the calls are non-recursive. That is, if program A calls program B, subprogram B may not call program A. Furthermore, a program or subprogram may not call itself. A calling program is a main program or subprogram that refers to another subprogram.

In addition to multi-statement function subprograms, a function may be defined by a single statement in the program (statement function) or it may be defined as basic external function. A statement function definition may appear in a main program or subprogram body and is available only to the main program or subprogram containing it. A statement function may contain references to function subprograms, basic external functions, or other previously defined statement functions in the same subprogram. Basic external function references may appear in the main program, subprogram, and statement functions.

Main programs, subprograms, statement functions, and basic external functions communicate by means of arguments (parameters). The arguments appearing in a subroutine call or function reference are actual arguments. The corresponding entities appearing with the subprogram, statement function, or basic external function definition are the dummy arguments.

ARGUMENT CHARACTERSITICS

Actual and dummy arguments must agree in order, type, and number. If they do not agree in type, errors may result in the program execution, since no conversion takes place and no diagnostic messages are produced.

Within subprograms, dummy arguments may be array names or simple variables; for statement functions, they may be variables only. Dummy arguments are local to the subprogram or statement function containing them and, therefore, may be the same as names appearing elsewhere in the program. A maximum of 63 dummy arguments may be used in a function or subroutine.

No element of a dummy argument list may appear in a COMMON or EQUIVALENCE statement within the subprogram. If it does, a compiler diagnostic results. When a dummy argument represents an array, it should be declared in a DIMENSION statement within the subprogram. If it is not declared, only the first element of the array will be available to the subprogram and the array name must appear in the subprogram without subscripts.

Actual arguments appearing in subroutine calls and function references may be any of the following:

A constant
A variable name
An array element name
An array name
Any other arithmetic expression

MAIN PROGRAM

The first statement of a main program may be the following:

```
PROGRAM name
```

The name is an alphanumeric identifier of up to five characters. If the PROGRAM statement is omitted, the compiler assigns the name "FTN."

SUBROUTINE SUBPROGRAM

An external subroutine is a computational procedure which may return none, one, or more than one value through its arguments or through common storage. No value or type is associated with the name of a subroutine.

The first statement of a subroutine subprogram gives its name and, if relevant, its dummy arguments.

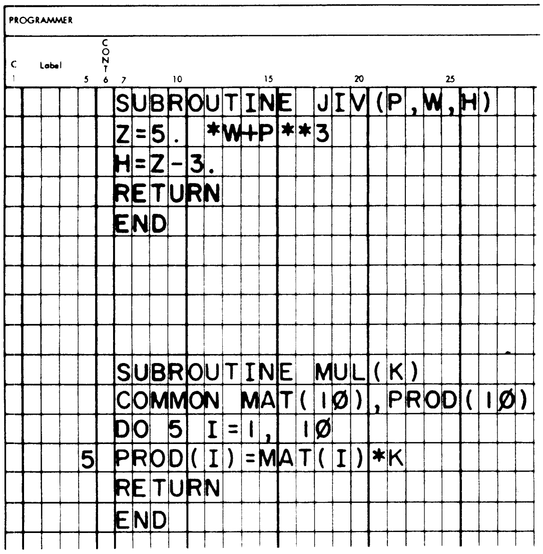
```
SUBROUTINE s(a1, a2,..., an)
   or
SUBROUTINE s
```

The symbolic name, s, is an alphanumeric identifier of up to five characters by which the subroutine is called. If the subroutine is unnamed the compiler will assign the name of "." (period). The a's are the dummy arguments of the subroutine.

The name of the subroutine must not appear in any other statement within the subprogram.

The subroutine may define or redefine one or more of its arguments and areas in common so as to effectively return results. It may contain any statements except FUNCTION, another SUBROUTINE statement, or any statement that directly or indirectly references the subroutine being defined. It must have at least one RETURN or END statement which returns control to the calling program.

Examples:


PROGRAMMER
C Label
CONT
5 6 7 10 15 20 25

```
      SUBROUTINE JIV(P,W,H)
      Z=5.  *W+P**3
      H=Z-3.
      RETURN
      END



      SUBROUTINE MUL(K)
      COMMON MAT(1Ø),PROD(1Ø)
      DO 5 I=1, 1Ø
    5 PROD(I)=MAT(I)*K
      RETURN
      END
```

P,W and H are the dummy parameters. Actual values supplied by a calling program are to be substituted for P and W. The variable name supplied for H would contain the result on return to the calling program.

MUL multiplies the array supplied for MAT by the single value supplied for K to produce values to be stored in array PROD.

SUBROUTINE CALL

The executable statement in the calling program for referring to a subroutine is:

CALL s $(a_1, a_2, \ldots, a_n)$

or

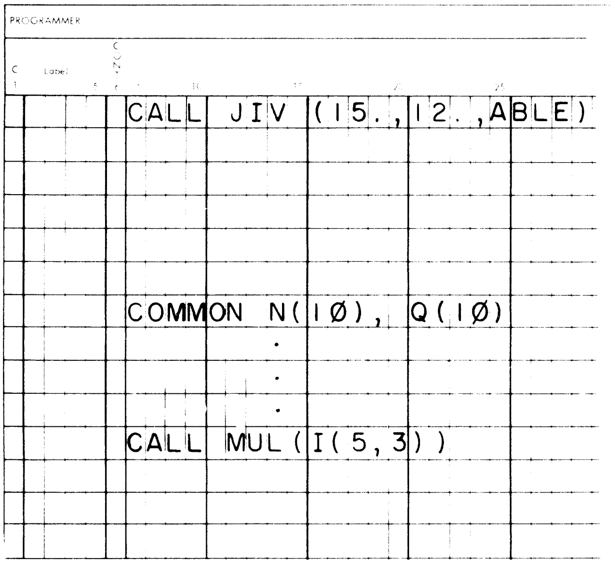
CALL s

The symbolic name, s, identifies the subroutine being called; the a's define the actual arguments. The name may not appear in any specification statements in the calling program.

If an actual argument corresponds to a dummy argument that is defined or redefined in the called subprogram, the actual argument must be a variable name, an array element name, or an array name.

The CALL statement transfers control to the subroutine. Execution of the subroutine results in an association of actual arguments with all appearances of dummy arguments in executable statement and function definition statements. If the actual argument is an expression, the association is by value rather than by name. Following these associations, the statements of the subprogram are executed. When a RETURN or END statement is encountered, control is returned to the next executable statement following the CALL in the calling program. If the CALL statement is the last statement in a DO loop, looping continues until satisfied.

Examples:



```
      CALL JIV (15.,12.,ABLE)





      COMMON N(1Ø), Q(1Ø)
               .
               .
               .
      CALL MUL (I(5,3))
```

These calls provide actual arguments for the subroutines defined in the previous example. In subroutine JIV, 15. is substituted for P; 12., for W; and ABLE, for H. For subroutine MUL, the data is passed via COMMON. The value supplied for the dummy argument K is element (5,3) of matrix I of the calling program.

FUNCTION SUBPROGRAM

A function subprogram is a computational procedure which returns a single value associated with the function name. The type of the function is determined by the name; an integer quantity is returned if the name begins with I, J, K, L, M, or N, otherwise it will be a real quantity.

The first statement of a function subprogram must have the following form:

FUNCTION f $(a_1, a_2, \ldots, a_n)$

The symbolic name, f, is an alphanumeric identifier of up to five characters by which the function is referenced. If the function is unnamed the compiler will assign the name of "." (period). The a's are the dummy arguments of the function.

The name of the function cannot appear in any non-executable statement within the subprogram. It must be used in the subprogram, however, at least once as any of the following:

The left-hand identifier of an assignment statement

An element of an input list

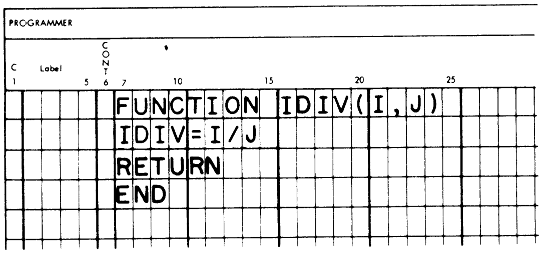
An actual parameter of a subprogram reference

The value of name at the time of execution of a RETURN or END statement in the subprogram is called the value of the function.

The function subprogram may define or redefine one or more of its arguments and areas in common so as to effectively return results in addition to the value of the function. If the subprogram redefines variables contained in the same expression as the function reference, the evaluation sequence of the expression must be taken into account. Variables in the portion of the expression that is evaluated before the function reference is encountered and the values of variable subscripts are not affected by the execution of the function subprogram. Variables that appear following the function reference are modified according to the subprogram processing.

Examples:

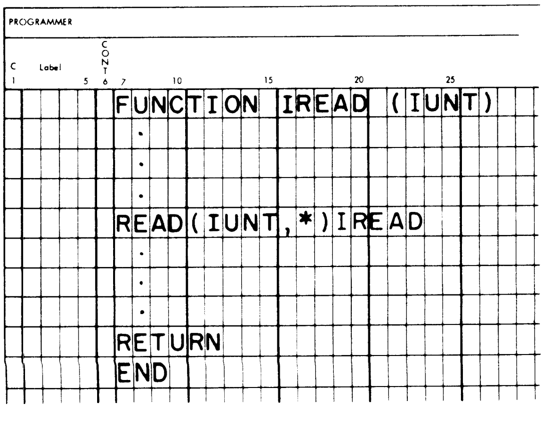
a)



```
      FUNCTION IDIV(I,J)
      IDIV=I/J
      RETURN
      END
```

The function IDIV calculates the value of I divided by J. On return to the calling program the result provided is the value of IDIV.

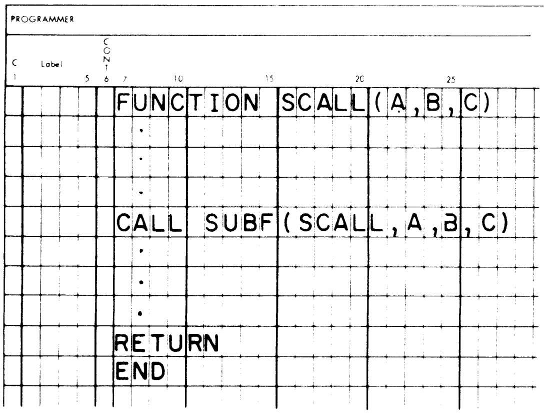
b)



```
      FUNCTION IREAD (IUNT)
      .
      .
      .
      READ(IUNT,*)IREAD
      .
      .
      .
      RETURN
      END
```

The function IREAD reads a value from the unit IUNT (specified as an actual parameter in the calling program.) IREAD has this value on return to the calling program.

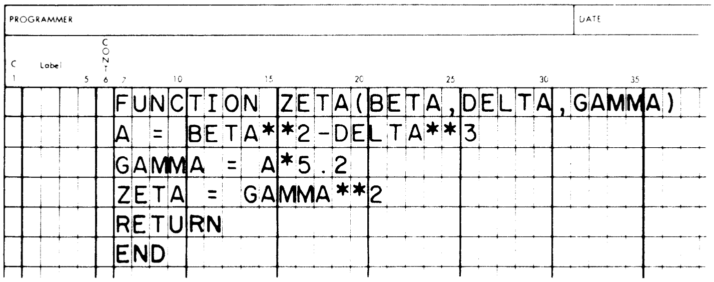
c)


PROGRAMMER
C Label
CONT
1 5 6 7 10 15 20 25

```
      FUNCTION SCALL(A,B,C)
      .
      .
      .
      CALL SUBF(SCALL,A,B,C)
      .
      .
      .
      RETURN
      END
```

SCALL is both the function name and an actual parameter of a subroutine call. The value of SCALL is provided by SUBF and returned to the calling program.

d)


PROGRAMMER
DATE
C Label
CONT
1 5 6 7 10 15 20 25 30 35

```
      FUNCTION ZETA(BETA,DELTA,GAMMA)
      A = BETA**2-DELTA**3
      GAMMA = A*5.2
      ZETA = GAMMA**2
      RETURN
      END
```

The function defines the value of GAMMA as well as finding the value of ZETA.

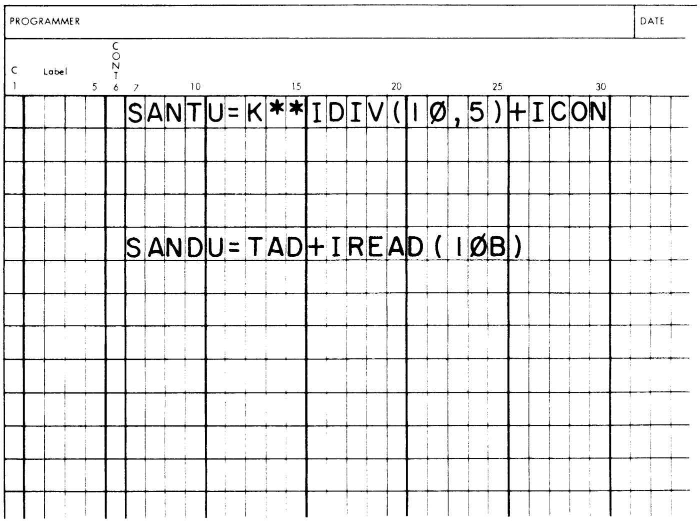
FUNCTION REFERENCE

A function subprogram is referenced by using the name and arguments in an arithmetic expression:

$$f(a_1, a_2, \ldots, a_n)$$

The type of function depends on the first letter of the name of the function referenced; the a's are the actual arguments. The reference may appear any place in an expression as an operand. The evaluated function will have a single value associated with the function name. When a function reference is encountered in an expression, control is transferred to the function indicated. Execution of the function results in an association of actual arguments with all appearances of dummy arguments in executable statements and function definition statements. If the actual argument is an expression, this association is by value rather than by name. Following these associations, the statements of the subprogram are executed. When a RETURN or END statement in the function subprogram is encountered, control returns to the statement containing the function reference. During execution the function also may define or redefine one or more of its arguments and areas in common.

Example:



PROGRAMMER DATE

C Label CONT

a)

```
      SANTU=K**IDIV(1Ø,5)+ICON
```

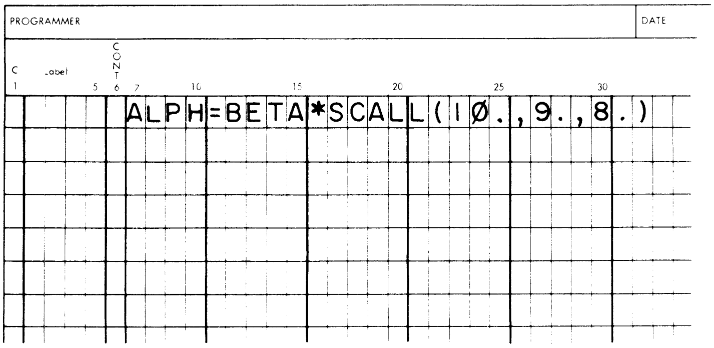
b)

```
      SANDU=TAD+IREAD(1ØB)
```

a) The values of 10 and 5 are provided for I and J: The resulting value of IDIV would be 2.

b) The function IREAD is called with 10B as the unit number. The value of IREAD would be the value of the item read from the device with unit reference number 10_8.

c)



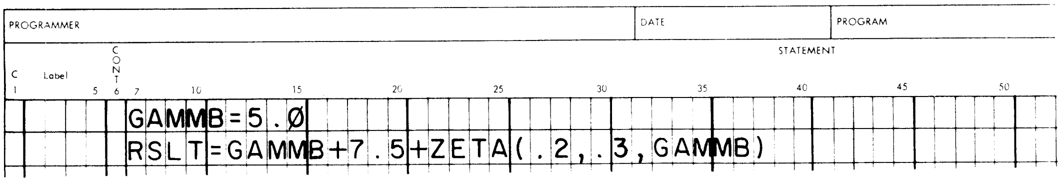
PROGRAMMER DATE

C Label CONT

```
      ALPH=BETA*SCALL(1Ø.,9.,8.)
```

The actual parameters SCALL are 10., 9., and 8. The value of SCALL would depend on the value supplied by the subroutine SUBF.

d) The program,



PROGRAMMER DATE PROGRAM

C Label CONT STATEMENT

```
      GAMMB=5.Ø
      RSLT=GAMMB+7.5+ZETA(.2,.3,GAMMB)
```

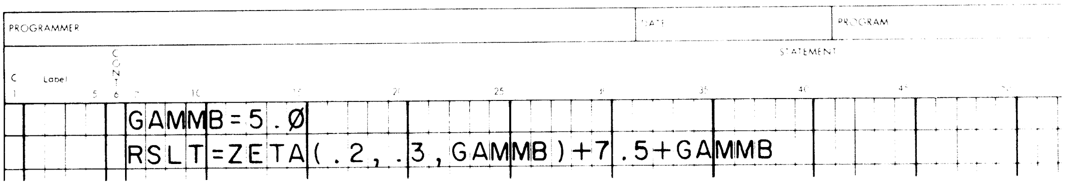
would result in the following calculation:

RSLT = 5.0 + 7.5 + ZETA

where ZETA would be determined as:

```
    A = .2**2 - .3**3 = .04 - .027 = .013
GAMMA = .013*5.2 = .0676 (GAMMB is not altered)
 ZETA = .0676**2 = .00456976
 RSLT = 5.0 + 7.5 + .00456976
      = 12.50456976
```

But, the program,



PROGRAMMER
DATE
PROGRAM
STATEMENT
C Label
CONT

```
      GAMMB=5.Ø
      RSLT=ZETA(.2,.3,GAMMB)+7.5+GAMMB
```

would result in the following calculations for ZETA and GAMMB:

A = .2**2 - .3**3 = .04 - .027 = .013

GAMMA = .013*5.2 = .0676 = GAMMB

ZETA = .0676**2 = .00456976

RSLT = .00456976 + 7.5 + .0676

= 7.57216976

When referring to a function which redefines an argument which appears as a variable elsewhere in the same expression, the order of evaluation (i.e., the order in which the expression is stated) is significant.

STATEMENT FUNCTION

A statement function is defined internally to the program or subprogram in which it is referenced and must precede the first executable statement. The definition is a single statement similar in form to an arithmetic assignment statement.

$$f\ (a_1,\ a_2,\ldots,\ a_n) = e$$

The name of the statement function, f, is an alphanumeric identifier; a single value is associated with the name. The dummy arguments, a's, must be simple variables. One to ten arguments may be used. The expression, e, may be an arithmetic expression and may contain references to basic external functions, previously defined statement functions, or function subprograms. The dummy arguments must appear in the expression. Other variables appearing in the expression have the same values as they have outside the statement function.

The statement function name must not appear in any specification statements in the program or subprogram containing it.

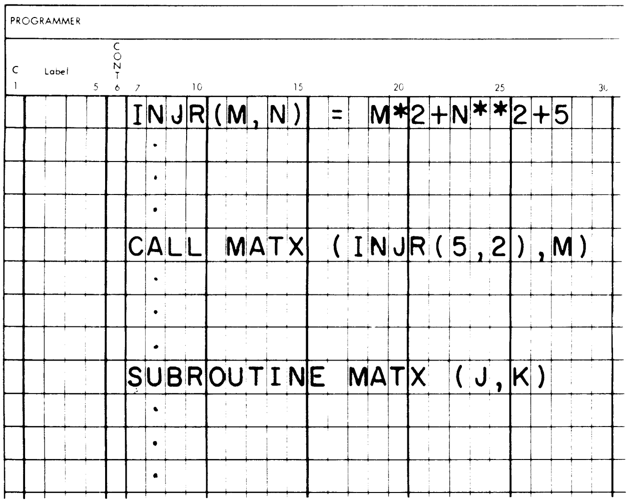
Statement functions must precede the first executable statement of the program or subprogram, but they must follow all specification statements.

A statement function reference has the form:

$$f\ (a_1,\ a_2,\ldots,\ a_n)$$

f is the function name and the a's are the actual arguments. A function reference with its appropriate actual arguments may be used to define the value of an actual argument in a subroutine call or function subprogram reference.

Example:


PROGRAMMER
C
Label
CONT
1
5
6
7
10
15
20
25
30

```
      INJR(M,N) = M*2+N**2+5
      .
      .
      .
      CALL MATX (INJR(5,2),M)
      .
      .
      .
      SUBROUTINE MATX (J,K)
      .
      .
      .
```

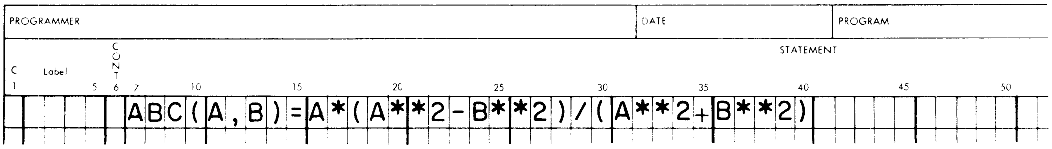
Statement function definition.

Subroutine call using statement function reference.

Execution of a statement function reference results in an association of actual argument values with the corresponding dummy arguments in the expression of the function definition, and evaluation of the expression. Following this, the resultant value is made available to the expression that contained the function reference and control is returned to that statement.

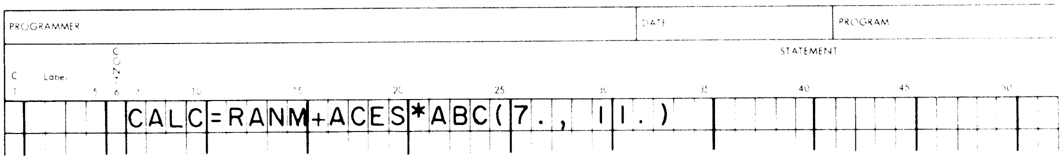
Example:

Statement function:


PROGRAMMER
DATE
PROGRAM
STATEMENT
C
Label
CONT
1
5
6
7
10
15
20
25
30
35
40
45
50

```
      ABC(A,B)=A*(A**2-B**2)/(A**2+B**2)
```

Function reference:


PROGRAMMER
DATE
PROGRAM
STATEMENT
C
Label
CONT
1
5
6
7
10
15
20
25
30
35
40
45
50

```
      CALC=RANM+ACES*ABC(7., 11.)
```

BASIC EXTERNAL FUNCTIONS

Certain basic functions are defined in FORTRAN. When one of these appears as an operand in an expression, the Compiler generates the appropriate calling sequence within the object program.

The types of these functions and their arguments are defined. The compiler recognizes the basic function and associates the type with the results. The actual arguments must correspond to the type required for the function; if not, a diagnostic message is issued. The functions available are shown in Table 6-1.

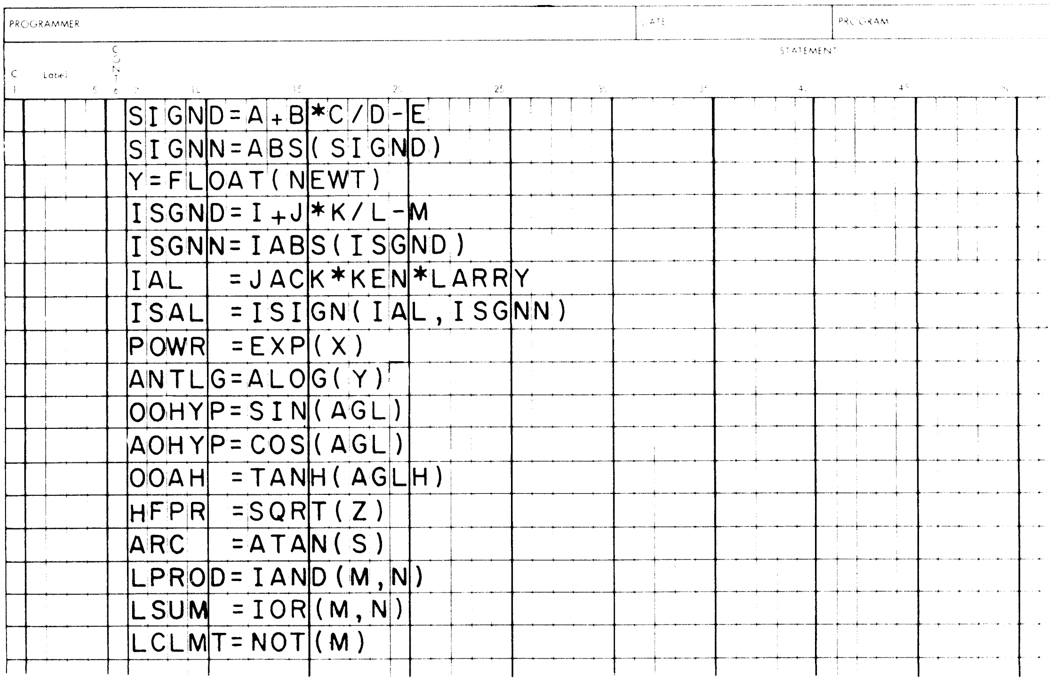
Table 6-1

FORTRAN Functions and Arguments

Function Name	Definition	Symbolic Name	No. of Arguments	Type of	
				Argument	Function
Absolute Value	$\|a\|$	ABS	1	Real	Real
		IABS	1	Integer	Integer
Float	Conversion from integer to real	FLOAT	1	Integer	Real
Fix	Conversion from real to integer	IFIX	1	Real	Integer
Transfer sign	Sign of a_2 times $\|a_1\|$	SIGN	2	Real	Real
		ISIGN	2	Integer	Integer
Exponential	e^a	EXP	1	Real	Real
Natural Logarithm	$\log_e$ (a)	ALOG	1	Real	Real
Trigonometric Sine	sine (a)†	SIN	1	Real	Real
Trigonometric Cosine	cos (a)†	COS	1	Real	Real
Trigonometric Tangent	tan (a)†	TAN	1	Real	Real
Hyperbolic Tangent	tanh (a)	TANH	1	Real	Real
Square Root	$(a)^{1/2}$	SQRT	1	Real	Real
Arctangent	arctan (a)	ATAN	1	Real	Real
And (Boolean)	$a_1 \wedge a_2$	IAND	2	Integer	Integer
Or (Boolean)	$a_1 \vee a_2$	IOR	2	Integer	Integer
Not (Boolean)	$\neg a$	NOT	1	Integer	Integer
Sense Switch	Sense Switch Register switch (n)	ISSW	1	Integer	Integer

†a is in radians

Examples:



```
      SIGND=A+B*C/D-E
      SIGNN=ABS(SIGND)
      Y=FLOAT(NEWT)
      ISGND=I+J*K/L-M
      ISGNN=IABS(ISGND)
      IAL  =JACK*KEN*LARRY
      ISAL =ISIGN(IAL,ISGNN)
      POWR =EXP(X)
      ANTLG=ALOG(Y)
      OOHYP=SIN(AGL)
      AOHYP=COS(AGL)
      OOAH =TANH(AGLH)
      HFPR =SQRT(Z)
      ARC  =ATAN(S)
      LPROD=IAND(M,N)
      LSUM =IOR(M,N)
      LCLMT=NOT(M)
```

RETURN AND END STATEMENTS

A subprogram normally contains a RETURN statement that indicates the end of logic flow within the subprogram and returns control to the calling program. It must always contain an END statement.

In function subprograms, control returns to the statement containing the function reference. In subroutine subprograms, control returns to the next executable statement following the CALL. A RETURN statement in the main program is interpreted as a STOP statement.

The END statement marks the physical end of a program, subroutine subprogram, or function subprogram. If the RETURN statement is omitted, END causes a return to the calling program. The END$ is required in additon to END statements when five or less subprograms are being compiled at one time.

SECTION VII
INPUT/OUTPUT LISTS AND FORMAT CONTROL

Data transmission between internal storage and external equipment requires an input/output statement and, for ASCII character strings, either a FORMAT statement or format control symbols with the input data. The input/output statement specifies the input/output process, such as READ or WRITE; the unit of equipment on which the process is performed; and the list of data items to be moved. The FORMAT statements or control symbols provide conversion and editing information between the internal representation and the external character strings. If the data is in the form of strings of binary values, format control is unnecessary.

INPUT/OUTPUT LISTS

The input list specifies the names of the variables and array elements to which values are assigned on input. The output list specifies the references to the variables, array elements, and constants whose values are transmitted. The input and output lists are of the same form. The list elements consist of variable names, array elements, and array names separated by commas. The order in which the elements appear in the list is the sequence of transmission. If FORMAT statements are used, the order of the list elements must correspond to the order of the format descriptions for the data items. In array elements buffer length is limited to a maximum output of 60 computer words.

Supscripts in an input/output list may be of the form (exp_1, exp_2), where exp_i is one of the following:

c*v+k

c*v-k

c*v

v+k

v-k

v

k

where c and k are integer constants and v is a simple integer variable previously defined or defined within an implied DO loop.

DO-IMPLIED LISTS

A DO-implied list consists of one or more list elements and indexing parameters. The general form is

$$(\ldots(\text{list},\ i = m_1,\ m_2,\ m_3)\ldots)$$

list — Any series of arrays, array elements, or variables separated by commas

i — Control variable

m's — Index parameters in the form of unsigned integer constants or predefined integer variables

Data defined by the list elements is transmitted starting at the value of m_1 in increments of m_3 until m_2 is exceeded. If m_3 is omitted it is assumed to be one.

An implied DO loop may be used to transmit a simple variable or a sequence of variables more than one time.

Two-dimensional arrays may appear in the list with values specified for the range of the subscripts in an implied DO loop. The general form for an array is:

$$((a(d_1,d_2),i_1 = m_1,\ m_2,\ m_3),\ i_2 = n_1,\ n_2,\ n_3)$$

where,

a — An array name

d_1, d_2 — Subscripts of the array in one of the preceding forms

i_1, i_2 — Control variables representing either of the variables subscripts d_1 and d_2

m's, n's — Index parameters in the form of unsigned integer constants or predefined integer variables. If m_3 or n_3 is omitted, it is construed as 1.

The input/output list may contain nested implied DO loops. During execution, the control variables are assigned the values of the initial parameters ($i_1 = m_1$, $i_2 = n_1$). The first control variable defined in the list is incremented first. When the first control variable reaches the maximum value, it is reset; the next control variable to the right is incremented and the process is repeated until the last control variable has been incremented.

If the name of a dimensioned array appears in a list without subscripts, the entire array is transmitted.

Examples:

a) The DO-implied list

```
((A(I,J), I=1, 20, 2), J=1, 50,5)
```

replaces the following:

```
    DO x J=1, 50, 5
    DO x I=1, 20, 2
    transmit A (I,J)
x   CONTINUE
```

b) Other implied DO loops might be:

```
((ABLE(5*KID-3, 100*LID), KID=1, 100), LID=1, 10)
((A(I,J), I=1, 5),J=1, 5) Transmit elements by column
((A(I,J), J=1, 5),I=1, 5) Transmit elements by row.
```

c) Nested implied DO loops:

```
((((A(I,J), B(K,L),K=1,10),L=1,15),I=1,20),J=1,25)
 (((A(I,J), B(K),K=1,10),I=20,100,10),K=9,90,10)
```

d) Simple variable transmission:

```
(A,K=1, 10) Transmits 10 values of A.
```

e) Dimensioned array transmission:

```
    DIMENSION A(50,20)
         .
         .
         .
    ... A ...    list element
    is equivalent to:
    DO x I = 1,20
    DO x J = 1,50
    transmit A(J,I)
x   CONTINUE
```

FORMAT STATEMENT

ASCII input/output statements may refer to a FORMAT statement which contains the specifications relating to the internal-external structure of the corresponding input/output list elements.

FORMAT (spec$_1$,..., r(spec$_m$,...), spec$_n$,...)

The spec's are format specifications and r is an optional repetition factor which must be an unsigned integer constant. FORMAT specifications may be nexted to a depth of one level. The FORMAT statement is non-executable and may appear anywhere in the program.

FORMAT Statement Conversion Specifications

The data elements in the input/output lists may be converted from external to internal and from internal to external representation according to FORMAT conversion specifications. (If the type of a variable in the input/output list does not correspond to the type specified in the FORMAT statement, the Formatter insures that the proper conversion from one type to the other will take place.) FORMAT statements may also contain editing codes.

Conversion Specifications

rEw.d	Real number with exponent
rFw.d	Real number without exponent
rIw	Decimal integer
r@w	Octal integer
rKw	
rAw	Alphanumeric character

Editing Specification

nX	Blank field descriptor
nHh_1 h_2...h_n	Heading and labeling descriptors Specification should not be on more than one line. If continuation is necessary, specification should be broken up in two specifications.
r"h_1 h_2...h_n"	
r/	Begin new record

Both w and n are nonzero integer constants representing the width of the field in the external character string; n may be omitted if the width is one. d is an integer constant representing the number of digits in the fractional part of the string. r, the repeat count, is an optional nonzero integer constant indicating the number of times to repeat the succeeding basic field descriptor. Each h is one character.

Ew.d Output

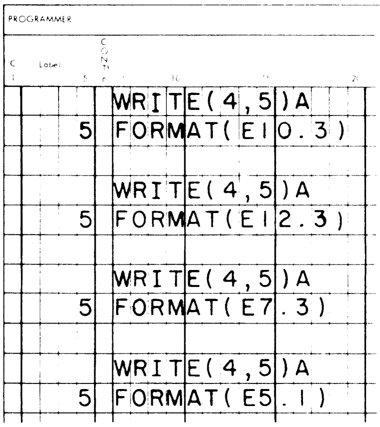
The E specification converts numbers in storage to character form for output. The field occupies w positions in the output record; the number appears in floating point form right justified in the field as:

$\underline{\wedge}.X_1 \ldots X_d$ E$\underline{+}$ee[†]

$X_1 \ldots X_d$ are the most significant digits of the value of the data to be output. ee are the digits in the exponent. Field w must be wide enough to contain significant digits, signs, decimal point, E, and exponent. Generally, w should be greater than or equal to d + 4.

If the field is not long enough to contain the output value, an attempt is made to adjust the value of d (i.e., truncating part or all of the fraction) so that a number is written in the field. If the remaining value is still to large for the field, dollar signs ($) are inserted in the entire field. If the field is longer than the output value, the quantity is right-justified with spaces to the left.

Examples:



```
      WRITE(4,5)A
    5 FORMAT(E10.3)
```

A contains +12.34 or -12.34
Result is ∧∧.123E+02 or ∧-.123E+02

```
      WRITE(4,5)A
    5 FORMAT(E12.3)
```

A contains +12.34 or -12.34
Result is ∧∧∧∧.123E+02 or
∧∧∧-.123E+02

```
      WRITE(4,5)A
    5 FORMAT(E7.3)
```

A contains +12.34 or -12.34
Result is .12E+02 or -.1E+02

```
      WRITE(4,5)A
    5 FORMAT(E5.1)
```

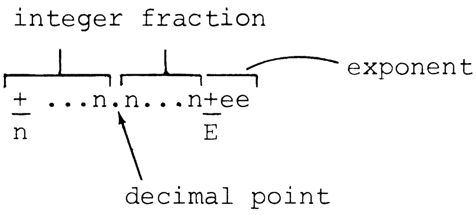
A contains +12.34
Result is $$$$$

[†] The caret symbol, ^, indicates the presence of a space.

Ew.d Input

The E spcification converts the number in the input field (specified by w) to a real number and stores it in the appropriate storage locations.

The input field may consist of integer, fraction, and exponent subfields:



The integer subfield begins with a + or - sign, or a digit and may contain a string of digits terminated by a decimal point, an E, +, -, or the end of the input field.

The fraction subfield begins with a decimal point and may contain a string of digits terminated by an E, +, -, or the end of the input field.

The exponent field may begin with a sign or an E and contains a string of digits. When it begins with E, the + is optional between E and the string. The value of the string of digits should not exceed 38. The number may appear in any positions within the field; spaces in the field are ignored.

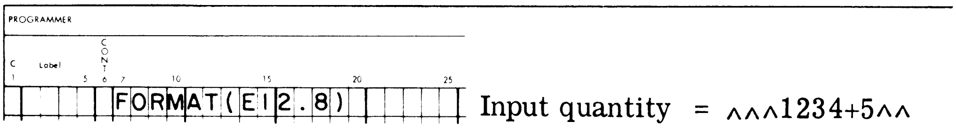
Examples:

```
+1.2345E2
123.456+9
-0.1234-6
.12345E-3
1234
+12345
+12345E6
```

When no decimal point is present in the input quantity, d acts as a negative power of ten scaling factor. The internal representation of the input quantity will be:

$$(\text{integer subfield})\text{x}10^{-d}\text{x}10^{(\text{exponent subfield})}$$

Example:


PROGRAMMER
C
Label
CONT
1
5
6
7
10
15
20
25

```
      FORMAT(E12.8)
```

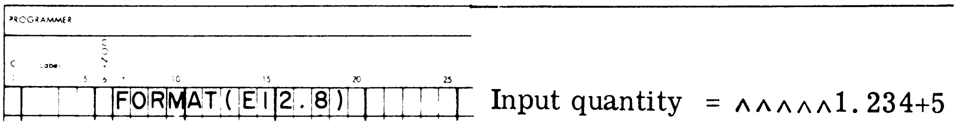
Input quantity = ∧∧∧1234+5∧∧

Conversion performed: $1234\text{x}10^{-8}\text{x}10^{5}$

Result: 1.234

If a d value in the specification conflicts with the a decimal point appearing in an input field, the actual decimal point takes precedence.

Example:


PROGRAMMER
C
Label
CONT
5
6
10
15
20
25

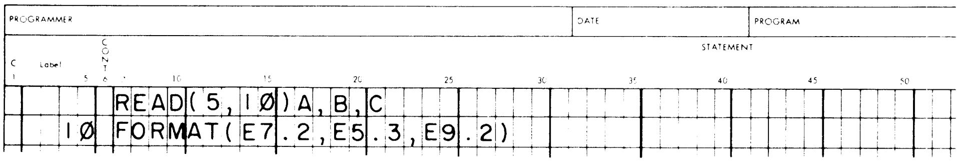
```
      FORMAT(E12.8)
```

Input quantity = ∧∧∧∧∧1.234+5

Quantity stored: $1.234\text{x}10^{5}$

The field width specified by w should always be the same as the width of the input field. When it is not, incorrect data may be read, converted and stored. The value of w should include positions for signs, the decimal point, the letter E, as well as the digits of the subfields:

Example:


PROGRAMMER
DATE
PROGRAM
C
Label
CONT
STATEMENT
1
5
6
10
15
20
25
30
35
40
45
50

```
      READ(5,1Ø)A,B,C
   1Ø FORMAT(E7.2,E5.3,E9.2)
```

Assuming input data in contiguous fields:

```
-12.3E1+1234123.46E-3
|←7—*— 5    *———9 →|
```

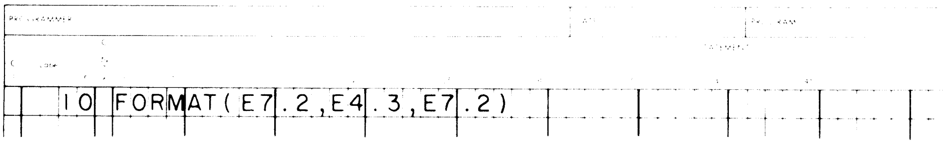
The fields read would be:

```
-12.3E1
+1234
123.46E-3
```

and converted as:

```
-123.
1.234
.12346
```

However, if specifications were:



```
   10 FORMAT(E7.2,E4.3,E7.2)
```

The fields read would be:

```
-12.3E1
+123
4123.46
```

The effects of possible FORMAT specification errors such as the above may not be detected by the system.

Examples:

FORMAT Specification	Input Field	Converted Value
E9.2	+1.2345E2	123.45
E9.4	-0.1234-6	-.0000001234
E4.2	1234	12.34

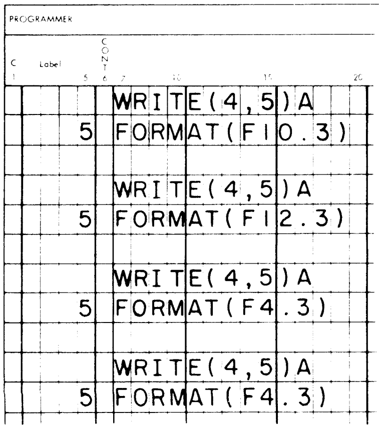
Fw.d Output

The F specification converts real numbers in storage to character form for output. The field occupies w positions and will appear as a decimal number, right justified in the field.

∧x...x.x...x

The x's are the most significant digits. The number of decimal places to the right of the decimal point is specified by d. If d is zero, no digits appear to the right of the decimal point. The field must be wide enough to contain the significant digits, sign, and decimal point. If the number is positive, the + sign is suppressed. If the field is not long enough to contain the output value, an attempt is made to adjust the value of d (i.e., truncating part or all of the fraction) so that a number is written in the field. If the remaining value is still too large for the field, dollar signs ($) are inserted in the entire field. If the field is longer than the output value, the number is right-justified with spaces occupying the excess positions on the left.

Examples:


PROGRAMMER
C
Label
CONT
1
5
6
7
10
15
20

```
      WRITE(4,5)A
    5 FORMAT(F10.3)
```

A contains +12.34 or -12.34
Result: ∧∧∧∧12.340 or ∧∧∧-12.340

```
      WRITE(4,5)A
    5 FORMAT(F12.3)
```

A contains +12.34 or -12.34
Result: ∧∧∧∧∧∧12.340 or∧∧∧∧∧ -12.340

```
      WRITE(4,5)A
    5 FORMAT(F4.3)
```

A contains +12.34
Result: 12.3

```
      WRITE(4,5)A
    5 FORMAT(F4.3)
```

A contains +12345.12
Result: $$$$

Fw.d Input

The F specification input is identical to the E specification input. Although the fields are generally assumed to contain only a sign, integer, decimal point, and fraction; they may also contain an exponent subfield. All restrictions for Ew.d input apply.

Iw

The Iw specification converts internal values to output character strings, or input character strings to internal numbers. The output external field occupies w record positions and appears right justified (spaces on left) as:

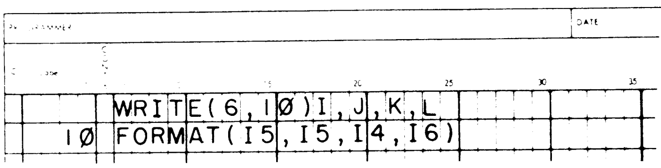
$$\overset{\wedge}{-}x_1 \ldots x_d$$

During input conversion, if a value is less than -32768_{10}, the value is converted to a positive 32767.

The x's represent the decimal digits (maximum of 5) of the integer. When the integer is positive on output, the sign is suppressed. If an output field is too short, dollar signs ($) will be placed in the output record.

The Iw specification, when used for input, is identical to an Fw.0 specification.

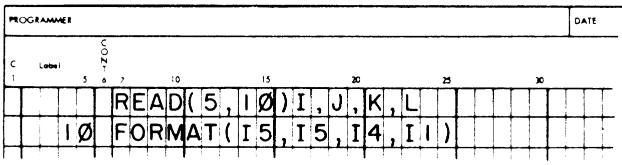
Examples:



```
      WRITE(6,1Ø)I,J,K,L
   1Ø FORMAT(I5,I5,I4,I6)
```

I contains -1234
J contains +12345
K contains +12345
L contains +12345

```
Result:  -123412345$$$$‸12345

         |←5→|←5→|←4→|←6→|
```



```
      READ(5,1Ø)I,J,K,L
   1Ø FORMAT(I5,I5,I4,I1)
```

PROGRAMMER
DATE
C
Label
CONT
1
5
6
7
10
15
20
25
30

Input contains:

-‸12312‸‸3‸1‸23

|←5→|←5→|←4→| 1 |

I contains -0123

J contains 12003

K contains 0102

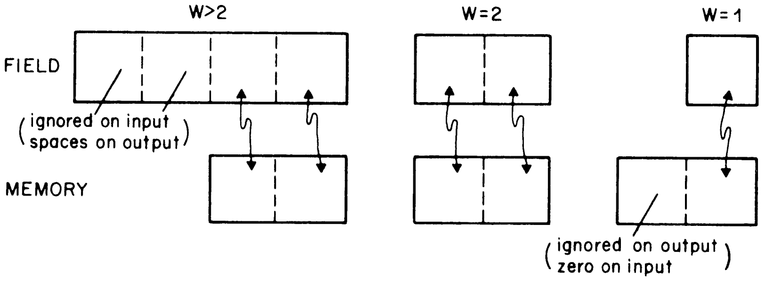
L contains 3

Aw

This specification (not available in the 4K version of FORTRAN) causes alphanumeric data on an external medium to be translated to or from ASCII form in memory. The associated list element must be of type integer.

On input, if the field, as indicated by w, is greater than 2, the first w-2 characters are ignored; only the last two characters are read. When w equals 2, the two characters are read. If w equals 1, one character is read and stored in the right half of a computer word; zero is entered in the left half.

On output, if the field is greater than 2, two characters are written with right justification in the field; the leading positions are filled with spaces. If w equals 2, the two characters are written. If w equals 1, the character in the right half of the computer word is written.



Example:

```
        Input data:  AZZ213-ABCXABC137 - ZZ9 CR LF

        DIMENSION ID (5)
        READ (5, 1Ø) 12, I1, ID
10      FORMAT (A1Ø, A1, 5A2)

        Result:  12 BC
                 I1 ØX
                 ID AB
                    C1
                    37
                    -Z
                    Z9
```

r @ w, rKw

Octal integer values are converted under either the @ or the K specification. The field is w octal digits in length; the corresponding list element must be of type integer. (Not available in the 4K version of FORTRAN.)

On input, if w is greater than or equal to 6, up to six octal digits are stored; non-octal digits appearing within the field are ignored. If the value of the octal digits within the field is greater than 177777, the results are unpredictable. If w is less than 6, or if less than six octal digits are encountered in the field, the number is right justified in the computer word with zero fill on the left.

On output, if the field is greater than 6, six octal digits are written with right justification in the field; the leading positions are filled with spaces. If w equals 6, the six octal digits are written. If w is less than 6, the w least significant octal digits are written.

Example:

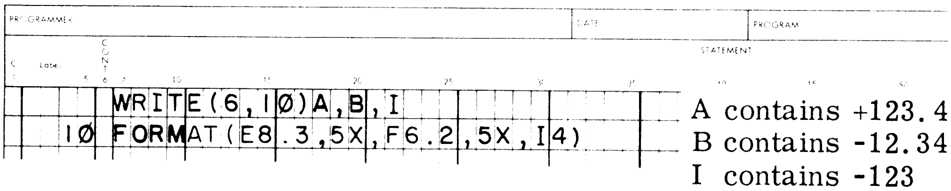
```
        Input data:  123456-1234562342342342, 396-Ø5 CR LF

        DIMENSION ID(2), IE(2)
        READ (5,10) IB, IC, ID, IE
10      FORMAT (@6, @7, 2@5, 2@4)
```

```
Result:  IB 123456
         IC 123456
         ID Ø23423
            Ø42342
         IE ØØØØ36
            ØØØØØ5
```

nX

The X specification may be used to include n blanks in an output record or to skip n characters on input to permit spacing of input/output quantities. In the specifications list, the comma following X is optional. ^X is interpreted as 1X. 0X is not permitted.

Examples:


PROGRAMMER
DATE
PROGRAM
STATEMENT
C
Label
CONT

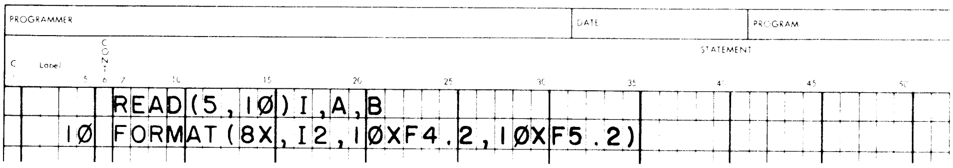
```
      WRITE(6,1Ø)A,B,I
   1Ø FORMAT(E8.3,5X,F6.2,5X,I4)
```

A contains +123.4
B contains -12.34
I contains -123

```
Result:  ^.1234E2^^^^^-12.34^^^^^-123
```

Input:

```
WEIGHT^^10^^PRICE^^$1.98^^TOTAL^^$19.80
```


PROGRAMMER
DATE
PROGRAM
STATEMENT
C
Label
CONT

```
      READ(5,1Ø)I,A,B
   1Ø FORMAT(8X,I2,1ØXF4.2,1ØXF5.2)
```

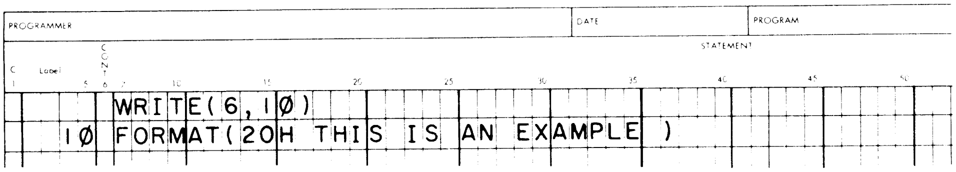
```
Result:  I contains 10
         A contains 1.98
         B contains 19.80
```

$nHh_1h_2...h_n$

The H specification provides for the transfer of any combination of 8-bit ASCII characters, including blanks. n is an unsigned integer specifying the number of characters to the right of the H that are to be transmitted. The comma following the H specification is optional. ‸H is interpreted as 1H. OH is not permitted. An H-specification should not span more than one line. If continuation is necessary the H specification should be broken off in 2H specifications, one on each line.

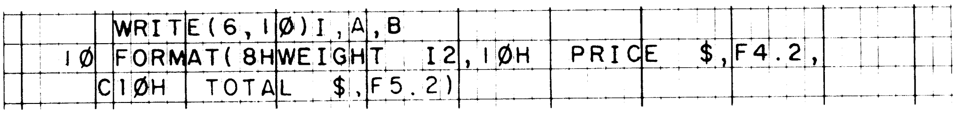
On output, the ASCII data in the FORMAT statement is written on the unit in the form of comments, titles, and headings.

Example:


PROGRAMMER
DATE
PROGRAM
STATEMENT
C
Label
CONT

```
      WRITE(6,1Ø)
   1Ø FORMAT(20H THIS IS AN EXAMPLE )
```

Result: THIS IS AN EXAMPLE



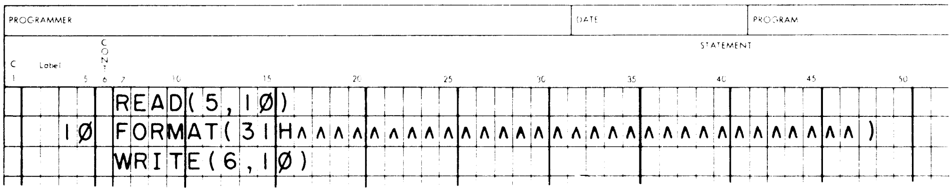
```
      WRITE(6,1Ø)I,A,B
   1Ø FORMAT(8HWEIGHT  I2,1ØH  PRICE   $,F4.2,
     C1ØH  TOTAL   $,F5.2)
```

I contains 10
A contains 1.98
B contains 19.80

Result: WEIGHT 10 PRICE $1.98 TOTAL $19.80

On input, the data is transmitted from the unit to the FORMAT statement. A subsequent output statement transfers the new data to the output record.

Examples:


PROGRAMMER
DATE
PROGRAM
STATEMENT
C
CONT
Label

```
      READ(5,1Ø)
   1Ø FORMAT(31HΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛ)
      WRITE(6,1Ø)
```

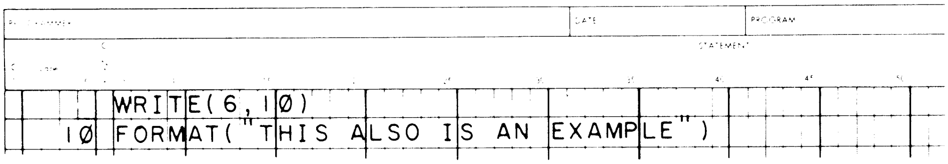
Input: H INPUT ALLOWS VARIABLE HEADERS

Result: H INPUT ALLOWS VARIABLE HEADERS

$r"h_1h_2...h_n"$

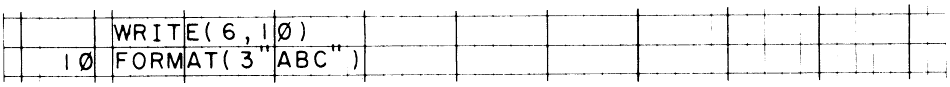
This specification also provides for the transfer of any combination of ASCII characters (except the quotation marks.). The number of characters transmitted is the number of positions between the two quotation marks; field length is not specified. If r, an optional repeat count, is present, the character string within the quotation marks is repeated that number of times. Commas preceding the initial quotation mark and following the closing quotation are optional. As with H, the specification must be contained on one line.

Examples:


PROGRAMMER
DATE
PROGRAM
STATEMENT

```
      WRITE(6,1Ø)
   1Ø FORMAT("THIS ALSO IS AN EXAMPLE")
```

Result: THIS ALSO IS AN EXAMPLE



```
      WRITE(6,1Ø)
   1Ø FORMAT(3"ABC")
```

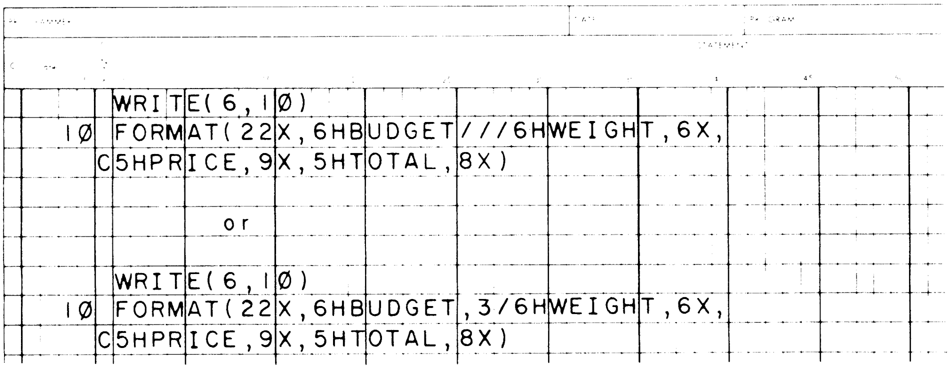
Result: ABCABCABC

On input, the number of characters within the quotation marks is skipped on the input field.

New Record

The slash, /, terminates the current record and signals the beginning of a new record of formatted data. It may occur anywhere in the specifications list and need not be separated from the other list elements by commas. Several records may be skipped by indicating consecutive slashes or by preceding the slash with a repetion factor; r-1 records are skipped for r/. On output the slash is used to skip lines, cards, or tape records; on input, it specifies that control passes to the next record or card.

Examples:



```
      WRITE(6,1Ø)
   1Ø FORMAT(22X,6HBUDGET///6HWEIGHT,6X,
     C5HPRICE,9X,5HTOTAL,8X)

          or

      WRITE(6,1Ø)
   1Ø FORMAT(22X,6HBUDGET,3/6HWEIGHT,6X,
     C5HPRICE,9X,5HTOTAL,8X)
```

Result:

```
line 1    ^^^^^^^^ ^^^^^^^^^^ ^^^^BUDGET
line 2
line 3
line 4    WEIGHT^^^^^^PRICE^^^^^^^^^TOTAL^^^^^^^^
```

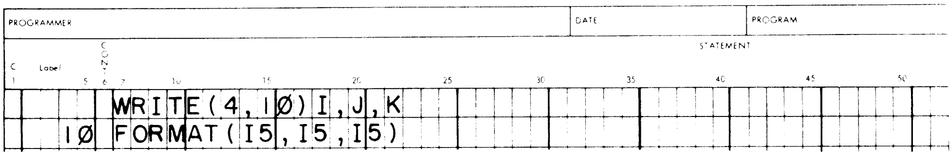
Repeat Specifications

Repetition of the field descriptors (except nH) is accomplished by preceding the descriptor with a repeat count, r. If the input/output list warrants, the conversion is interpreted repetitively up to the specified number of times.

Repetition of a group of field descriptors, including nH is accomplished by enclosing the group in parentheses and preceding the left parenthesis with a group in parentheses and preceding the left parenthesis with a group repeat count.

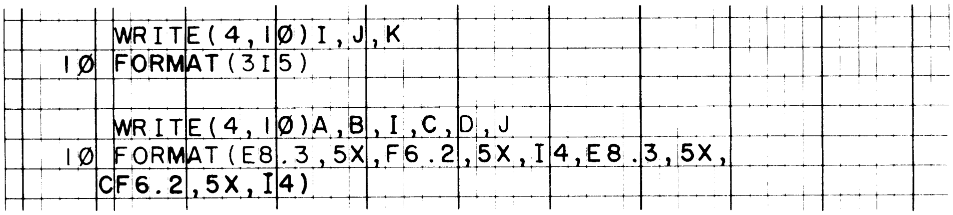
If no group repeat count is specified, a value of one is assumed. Grouped field descriptors may be nested to a depth of one level.

Examples:



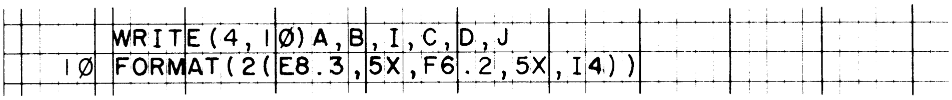
```
PROGRAMMER                                  DATE            PROGRAM
      WRITE(4,1Ø)I,J,K
   1Ø FORMAT(I5,I5,I5)
```

can be written as



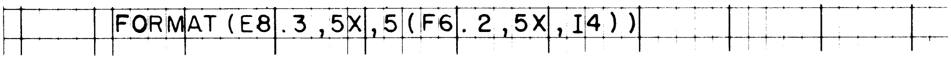
```
      WRITE(4,1Ø)I,J,K
   1Ø FORMAT(3I5)

      WRITE(4,1Ø)A,B,I,C,D,J
   1Ø FORMAT(E8.3,5X,F6.2,5X,I4,E8.3,5X,
     CF6.2,5X,I4)
```

can be written as



```
      WRITE(4,1Ø)A,B,I,C,D,J
   1Ø FORMAT(2(E8.3,5X,F6.2,5X,I4))
```

A nested repetition specification would be:



```
      FORMAT(E8.3,5X,5(F6.2,5X,I4))
```

The group F6.2, 5X, I4 would be written five times, and the entire group, once.

Unlimited Groups

FORMAT specifications may be repeated without use of the repetition factor. If list elements remain after all specifications in a FORMAT statement are processed, the rightmost group of repeated (enclosed in parentheses) specifications is used. If there is no repeated group, processing resumes with the first specification in the statement. On output, each time the rightmost parenthesis in the statement, or in the unlimited group, is reached, the current record is terminated.

FREE FIELD INPUT

By following certain conventions in the preparation of the input data, a 2116A FORTRAN program may be written without use of FORMAT statements. Special symbols included with the ASCII input data items direct the formatting:

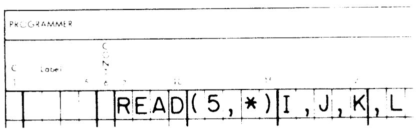
space or,	Data item delimiters
/	Record terminator
+ -	Sign of item
. E + -	Floating point number
@	Octal integer
"..."	Comments

All other ASCII non-numeric characters are treated as spaces (and delimiters). Free field input may be used for numeric data only. Free field input is indicated in the FORTRAN READ statement by using an asterisk rather than a number of a FORMAT statement.

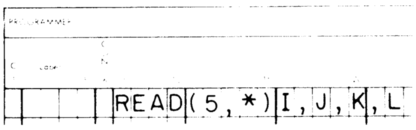
Data Item Delimiters

Any contiguous string of numeric and special formatting characters occurring between two commas, a comma and a space, or two spaces, is a data item whose value corresponds to a list element. A string of consecutive spaces is equivalent to one space. Two consecutive commas indicate that no data item is supplied for the corresponding list element; the current value of the list element is unchanged. An initial comma causes the first list element to be skipped.

Example:

1) 
PROGRAMMER

```
      READ(5,*)I,J,K,L
```

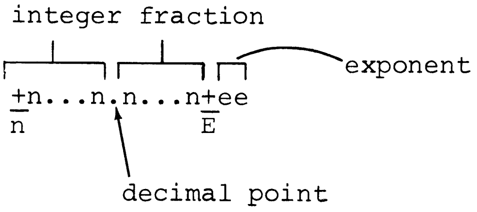
2) 
PROGRAMMER

```
      READ(5,*)I,J,K,L
```

```
Input data:  1720, 1966            Input data:  1266,,1794,2000
             1980  1492


Result:  I contains 1720           Result:  I contains 1266
         J contains 1966                    J contains 1966
         K contains 1980                    K contains 1794
         L contains 1492                    L contains 2000
```

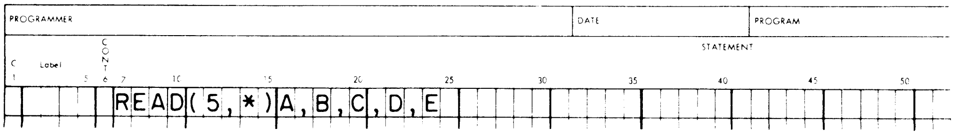
Floating Point Input

The symbols used to indicate a floating point data item are the same as those used in representing floating point data for FORMAT statement directed input:



If the decimal point is not present, it is assumed to follow the last digit.

Examples:


PROGRAMMER | DATE | PROGRAM

C Label CONT STATEMENT

1 5 6 7 10 15 20 25 30 35 40 45 50

```
      READ(5,*)A,B,C,D,E
```

Input Data: 3.14, 314E-2, 3140-3, .0314+2, .314E1

All are equivalent to 3.14

Octal Input

An octal input item has the following format:

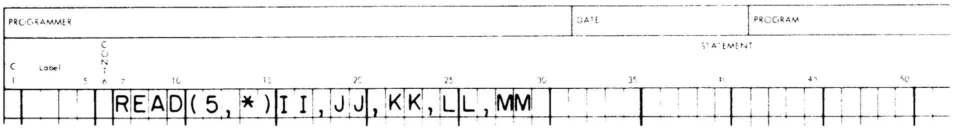
$@x_1...x_d$

The symbol @ defines an octal integer. The x's are octal digits each in the range of 0 through 7. List elements corresponding to the octal data items must be type integer.

Record Terminator

A slash within a record causes the next record to be read immediately; the remainder of the current record is skipped.

Example:



PROGRAMMER | DATE | PROGRAM

C Label | CONT | STATEMENT

```
      READ(5,*)II,JJ,KK,LL,MM
```

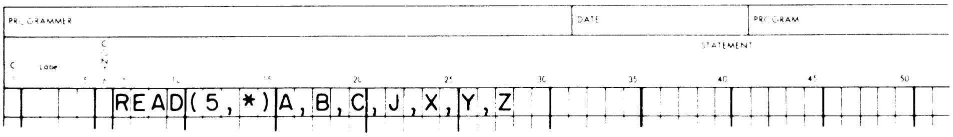
Input data: 987, 654, 321, 123/DESCENDING CR LF
456

```
Result:  II  contains 987
         JJ  contains 654
         KK  contains 321
         LL contains 123
         MM contains 456
```

List Terminator

If a line terminates (with a CR LF) and a slash has not been encountered, the input operation terminates even though all list elements may not have been processed. The current values of remaining elements are unchanged.

Examples:



PROGRAMMER | DATE | PROGRAM

C Label | CONT | STATEMENT

```
      READ(5,*)A,B,C,J,X,Y,Z
```

Input Data:

```
  A=7.987   B=5E2   C=4.6859E-3  CR  LF
  J=3456  CR  LF
```

```
Result:  A contains 7.987
         B contains 5E2
         C contains 4.6859E-3

         J, X, Y, Z are unchanged.
```

Comments

All characters appearing between a pair of quotation marks in the same line are considered to be comments and are ignored.

Examples:

```
"6.7321"   is a comment and ignored
 6.7321    is a real number
```

SECTION VIII
INPUT/OUTPUT STATEMENTS

Input/output statements transfer information between memory and an external unit. The logical unit is specified as an integer variable that is defined elsewhere in the program or an integer constant.

Each statement may include a list of names of variables, arrays, and array elements. The named elements are assigned values on input and have their values transferred on output.

Records may be formatted or unformatted. A formatted record consists of a string of ASCII characters. The transfer of such a record requires the specification of a FORMAT statement or free field input data. An unformatted record consists of a string of binary values.

LOGICAL UNIT NUMBERS

FORTRAN input/output statements refer to logical unit numbers (1 to 63) whose meaning varies depending upon the operating system used. Refer to the appropriate manual. The operating system relates the logical unit number to a physical unit through system tables. Logical unit 4 always refers to a punch device, 5 to an input device, and 6 to a list output device.

FORMATTED READ, WRITE

A formatted READ statement is one of the forms:

```
READ (u, f)k
READ (u, *)k
READ (u, f)
```

Execution of this statement causes the input of the next ASCII records from unit u. The information is scanned and converted according to the FORMAT specification statement, f, and assigned to the elements of list k. If the input is free field, an asterisk is specified in the READ statement rather than the label of a FORMAT statement. If the list is absent, the FORMAT statement should contain editing specifications only.

A formatted WRITE statement may have one of the following forms:

```
WRITE (u, f)k
    or
WRITE (u, f)
```

This statement transfers ASCII information from locations given by names in the list k to output unit u. The values are converted and positioned as specified by the FORMAT statement f. If the list is absent, the FORMAT statement should contain editing specifications only.

UNFORMATTED READ, WRITE

An unformatted READ statement has one of the forms:

```
READ (u)k
    or
READ (u)
```

This statement transfers the next binary input record from the unit u to the elements of list k. The sequence of values required by the list may not exceed the sequence of values from the record. If no list is specified, READ (u) skips the next record.

An unformatted WRITE statement has the form:

```
WRITE (u)k
```

Execution of this statement creates the next record on unit u from the sequence of values represented by the list k.

AUXILIARY INPUT/OUTPUT STATEMENTS

There are three types of auxiliary input/output statements:

```
REWIND
BACKSPACE
ENDFILE
```

A REWIND statement has the form:

```
REWIND u
```

This statement causes the unit u to be positioned at its initial point. If the unit is currently at this position, the statement acts as a CONTINUE.

A BACKSPACE statement is as follows:

```
BACKSPACE u
```

BACKSPACE positions the unit u so that what had been the preceding record becomes the next record. If the unit is currently at its initial point, the statement acts as a CONTINUE.

An ENDFILE statement is of the form:

```
ENDFILE u
```

Execution of this statement causes the recording of an end-of-file record on the output unit u. If given for an input unit, the statement acts as a CONTINUE.

SECTION IX
COMPILER INPUT AND OUTPUT

The FORTRAN Compiler accepts as input, paper tape containing a control statement and a source language program. The output produced by the Compiler may include a punched paper tape containing the object program; a listing of the source language program with diagnostic messages, if any; and a listing of the object program in assembly level language.

CONTROL STATEMENT

The control statement must precede the first statement of the source program; it directs the compiler.

FTN, p_1, p_2, p_3

FTN is a free field control statement. Following the comma are one to three parameters, in any order, which define the output to be produced. The control statement must be terminated by an end-of-statement mark, (CR) (LF). Spaces embedded in the statement are ignored.

The parameters may be a combination of the following:

B Binary output: A program is to be punched in relocatable binary format suitable for loading by the Relocating Loader.

L List output: A listing of the source language program is to be produced as the source program is read in.

A Assembly listing: A listing of the object program in assembly level language is to be produced in the last pass.

T Symbol table only: A listing of the symbol table only is produced; in MTS, if both T and A are specified, only the last used will be decisive.

SOURCE PROGRAM

The source program follows the control statement. Each statement is followed by the end-of-statement mark, CR LF. Specifications statements must precede executable statements. The last statement in each program submitted for compilation must be an END statement. Up to five source programs may be compiled at one time. The last program must be followed by an END$ statement, if less than six programs are to be compiled.

The control statement, each of the five programs, and the END$ terminator may be submitted on a single tape or on separate tapes. If more than five programs are contained on a tape, the compiler processes only the first five. The remaining programs must be compiled separately.

BINARY OUTPUT

The punch output produced by the compiler is a relocatable binary program. It does not include system subroutines introduced by the compiler, or library subroutines referred to in the program.

LIST OUTPUT

If the List Output parameter is specified, the first 72 characters of each line of the source program is printed on the List Output device. The END$ is the last statement printed. If exactly five programs are compiled, however, the END$ is omitted from the list.

If the Assembly listing parameter is specified, the program is printed in assembly level language on the List Output device. If the Symbol Table option is specified, the program listing is followed by a Symbol Table for the assembly level program.

The format for the assembly level listing is as follows:

Columns	Content
1-5	Zero-relative location (octal) of the instruction
6-7	Blank
8-13	Object code word in octal
14	Relocation or external symbol indicator
15	Blank
16-18	Mnemonic operation code
19	Blank
20-25	Operand address in octal or external symbol name.
26-27	The indicator ",I" if indirect addressing is used.

The Symbol Table listing has the following format:

Columns	Content
1-5	Symbol, statement label, or numeric symbol assigned by the compiler.
6	Blank
7	Relocation indicator
8	Blank
9-14	The zero-relative value of the symbol

The characters that designate an external symbol or type of relocation for the operand address or a symbol in the Symbol Table are:

Character	Relocation Base
Blank	Absolute
R	Program relocatable
X	External symbol
C	Common relocatable

NOTE: The operating procedures for the FORTRAN Compiler are contained in the *SOFTWARE OPERATING PROCEDURES SIO SUBSYSTEMS* Module (5951-1390).

APPENDIX A
HP CHARACTER SET

ASCII CHARACTER FORMAT

b_7				0	0	0	0	1	1	1	1
b_6				0	0	1	1	0	0	1	1
b_5				0	1	0	1	0	1	0	1
b_4	b_3	b_2	b_1								
0	0	0	0	NULL	DC_0	ƀ	0	@	P	UNASSIGNED	UNASSIGNED
0	0	0	1	SOM	DC_1	!	1	A	Q		
0	0	1	0	EOA	DC_2	"	2	B	R		
0	0	1	1	EOM	DC_3	#	3	C	S		
0	1	0	0	EOT	DC4 (STOP)	$	4	D	T		
0	1	0	1	WRU	ERR	%	5	E	U		
0	1	1	0	RU	SYNC	&	6	F	V		
0	1	1	1	BELL	LEM	' (APOS)	7	G	W		
1	0	0	0	FE_0	S_0	(	8	H	X		
1	0	0	1	HT / SK	S_1	)	9	I	Y		
1	0	1	0	LF	S_2	*	:	J	Z		
1	0	1	1	V_{TAB}	S_3	+	;	K	[		
1	1	0	0	FF	S_4	, (COMMA)	<	L	\		ACK
1	1	0	1	CR	S_5	—	=	M	]		①
1	1	1	0	SO	S_6	.	>	N	↑		ESC
1	1	1	1	SI	S_7	/	?	O	←		DEL

Standard 7-bit set code positional order and notation are shown below with b_7 the high-order and b_1 the low-order, bit position.

Example: The code for "R" is: $1^{b_7}\ 0^{b_6}\ 1^{b_5}\ 0^{b_4}\ 0^{b_3}\ 1^{b_2}\ 0^{b_1}$

LEGEND

NULL	Null/Idle
SOM	Start of message
EOA	End of address
EOM	End of message
EOT	End of transmission
WRU	"Who are you?"
RU	"Are you...?"
BELL	Audible signal
FEo	Format effector
HT	Horizontal tabulation
SK	Skip (punched card)
LF	Line feed
V_{TAB}	Vertical tabulation

LEGEND (cont)

FF	Form feed
CR	Carriage return
SO	Shift out
SI	Shift in
DCo	Device control reserved for data link escape
DC_1-DC_3	Device Control
DC_4(Stop)	Device control (stop)
ERR	Error
SYNC	Synchronous idle
LEM	Logical end of media
So-S_7	Separator (information)
ƀ	Word separator (space, normally non-printing)
<	Less than
>	Greater than
↑	Up arrow (Exponentiation)
←	Left arrow (Implies/Replaced by)
\	Reverse slant
ACK	Acknowledge
⦶	Unassigned control
ESC	Escape
DEL	Delete/Idle

APPENDIX B
ASSEMBLY LANGUAGE SUBPROGRAMS

A FORTRAN program can refer to a subprogram that has been prepared using Assembler source language. The subprogram may be treated as a subroutine or as a function. The object code programs generated by FORTRAN and by the Assembler are then linked together by the Relocating Loader when the programs are loaded.

FORTRAN REFERENCE

In the FORTRAN program, a subroutine is called using the following statement:

CALL s $(a_1, a_2,...,a_n)$

The symbolic name, s, identifies the subroutine and the a's are the actual arguments.

If the subprogram is a function, it is referenced by using the name and the actual arguments in an arithmetic expression:

$f(a_1, a_2,...,a_n)$

As a result of either the call or the reference, FORTRAN generates the following coding sequence:

```
JSB s/f     Transfers control to subroutine or function
DEF*+n+1    Defines return location
DEF a1      Defines address of a1
DEF a2      Defines address of a2
  .
  .
  .
DEF an      Defines address of an
```

The words defining the addresses of the arguements may be direct or indirect depending on the actual arguments. For example, an integer constant as an actual argument would yield a direct reference; an integer variable might yield an indirect reference.

If the subprogram being referenced is a subroutine, it may return none, one, or more than one value through its arguments or through common storage. If the subprogram is a function, it is assumed to return a single value in the accumulators: a function of type integer returns a value in the A-Register; a function of type real returns a value in the A- and B-Registers.

The subprogram may transfer values directly by accessing the words in the calling sequence or it may make use of the FORTRAN library subroutine .ENTR to aid in the transfer.

DIRECT TRANSFER OF VALUES

Any suitable technique may be used to obtain or deliver values for the arguments and to return control to the calling program. If address arithmetic is used in conjunction with an argument (e.g., to process elements of an array), the base location must be a direct reference; the location given in the calling sequence must be checked to determine if it is a direct or indirect reference. If it is an indirect reference the location to which it points must also be checked, and so forth.

Example:

```
PROGRAMMER                                   DATE        PROGRAM
                                                      STATEMENT
Label    Operation   Operand                        Comments
       NAM AMSUB
       ENT AMSUB
AMSUB  NOP          AMSUB TO CONTAIN ADDR OF "*+N+1"
       LDA AMSUB,I  A CONTAINS VALUE OF "*+N+1".
       STA RETRN    RETRN CONTAINS VALUE OF "*+N+1".
NXTAG  ISZ AMSUB    AMSUB CONTAINS ADDR OF LOCATION
       LDA AMSUB    OF ARGUMENT. TEST IF ALL ARGU-
       CPA RETRN    MENTS PROCESSED: COMPARE VALUE
       JMP RETRN,I  OF "*+N+1" WITH ADDR OF CURRENT
                    LOCATION OF ARGUMENT. IF EQUAL
PRSAG               RETURN TO CALLING PROGRAM, IF NOT,
        .           PROCESS ARGUMENT AS REQUIRED.
        .
        .
       LDA AMSUB,I  A CONTAINS LOCATION OF ARGUMENT.
       LDA Ø,I      LOAD ONE-WORD (FIXED POINT)
        .           VALUE INTO A.
        .
        .
       LDA AMSUB,I  LOAD TWO-WORD (FLOATING POINT)

       DLD Ø,I      VALUE INTO A AND B.
        .
        .
        .
       LDA AMSUB,I  STORE ONE-WORD VALUE IN ARGUMENT
       STA OUTAD    LOCATION.
       LDA W1VAL
       STA OUTAD,I
        .
        .
        .
       LDA AMSUB,I  STORE TWO-WORD IN ARGUMENT
       STA OUTAD    LOCATIONS.
       DLD W2VAL
       DST OUTAD,I
        .
        .
        .
       LDA AMSUB,I  A CONTAINS ADDR OF LOCATION OF
       SSA          ARGUMENT. TO DETERMINE IF REF IS
       JMP *+2      INDIRECT, TEST BIT 15. IF ONE,
       JMP *+5      SET TO ZERO WITH AND, THEN LOAD
       AND ANMSK    A WITH REFERENCED LOCATION.
       LDA Ø,I      REPEAT TEST WITH NEXT REF. WHEN
       JMP *-5      DIRECT REF ENCOUNTERED, PROCEED
ANMSK  OCT Ø77777   WITH PROCESSING.
        .
        .
        .
       JMP NXTAG    RETURN THROUGH HERE WHEN NEXT
RETRN  BSS 1        ARGUMENT IS REQUIRED.
OUTAD  BSS 1
W1VAL  BSS 1
W2VAL  BSS 2
       END
```

The preceding example assumes that each argument is processed or partially processed before the next is obtained or delivered. Control returns to the calling program when all arguments have been picked up or delivered.

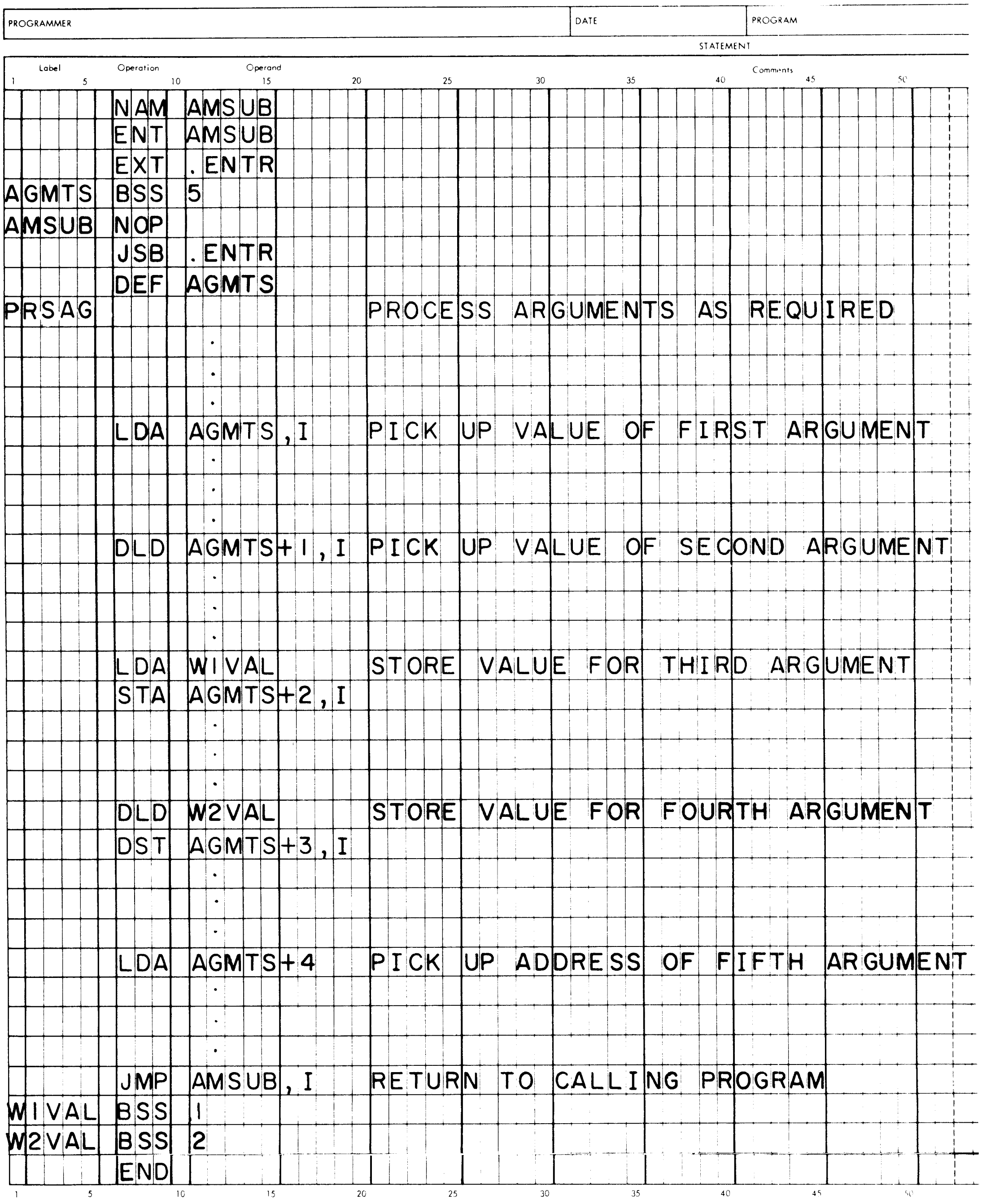
TRANSFER VIA .ENTR

The transfer of values to or from the locations listed in the calling sequence may be facilitated through use of the FORTRAN library subroutine .ENTR . This subroutine moves the addresses of the arguments into an area reserved within the Assembly language subroutine. The addresses stored in the reserved area are all direct references; .ENTR performs all the necessary direct/indirect testing, etc. It also sets the correct return address in the entry point location.

The general form of the subroutine is:

```
        NAM s           The subroutine name is s.
        ENT s
        EXT .ENTR       .ENTR must be declared as external.
a       BSS n           Reserves n words of storage for the
s       NOP             addresses of the arguments; this pseudo
                        instruction must directly precede the entry
                        point location, s.
        JSB .ENTR
        DEF a           Defines first location of area used to
 (First instruction)    store argument addresses.
           .
           .
           .
        JMP s, I
        END
```

Example:



PROGRAMMER | DATE | PROGRAM

STATEMENT

Label | Operation | Operand | Comments

```
      NAM  AMSUB
      ENT  AMSUB
      EXT  .ENTR
AGMTS BSS  5
AMSUB NOP
      JSB  .ENTR
      DEF  AGMTS
PRSAG                PROCESS ARGUMENTS AS REQUIRED
           .
           .
           .
      LDA  AGMTS,I   PICK UP VALUE OF FIRST ARGUMENT
           .
           .
           .
      DLD  AGMTS+1,I PICK UP VALUE OF SECOND ARGUMENT
           .
           .
           .
      LDA  W1VAL     STORE VALUE FOR THIRD ARGUMENT
      STA  AGMTS+2,I
           .
           .
           .
      DLD  W2VAL     STORE VALUE FOR FOURTH ARGUMENT
      DST  AGMTS+3,I
           .
           .
           .
      LDA  AGMTS+4   PICK UP ADDRESS OF FIFTH ARGUMENT
           .
           .
           .
      JMP  AMSUB,I   RETURN TO CALLING PROGRAM
W1VAL BSS  1
W2VAL BSS  2
      END
```

APPENDIX C
SAMPLE PROGRAM

Using Simpson's rule, calculate the value of the integral:

$$\int_a^b \frac{\cos x}{x}\, dx$$

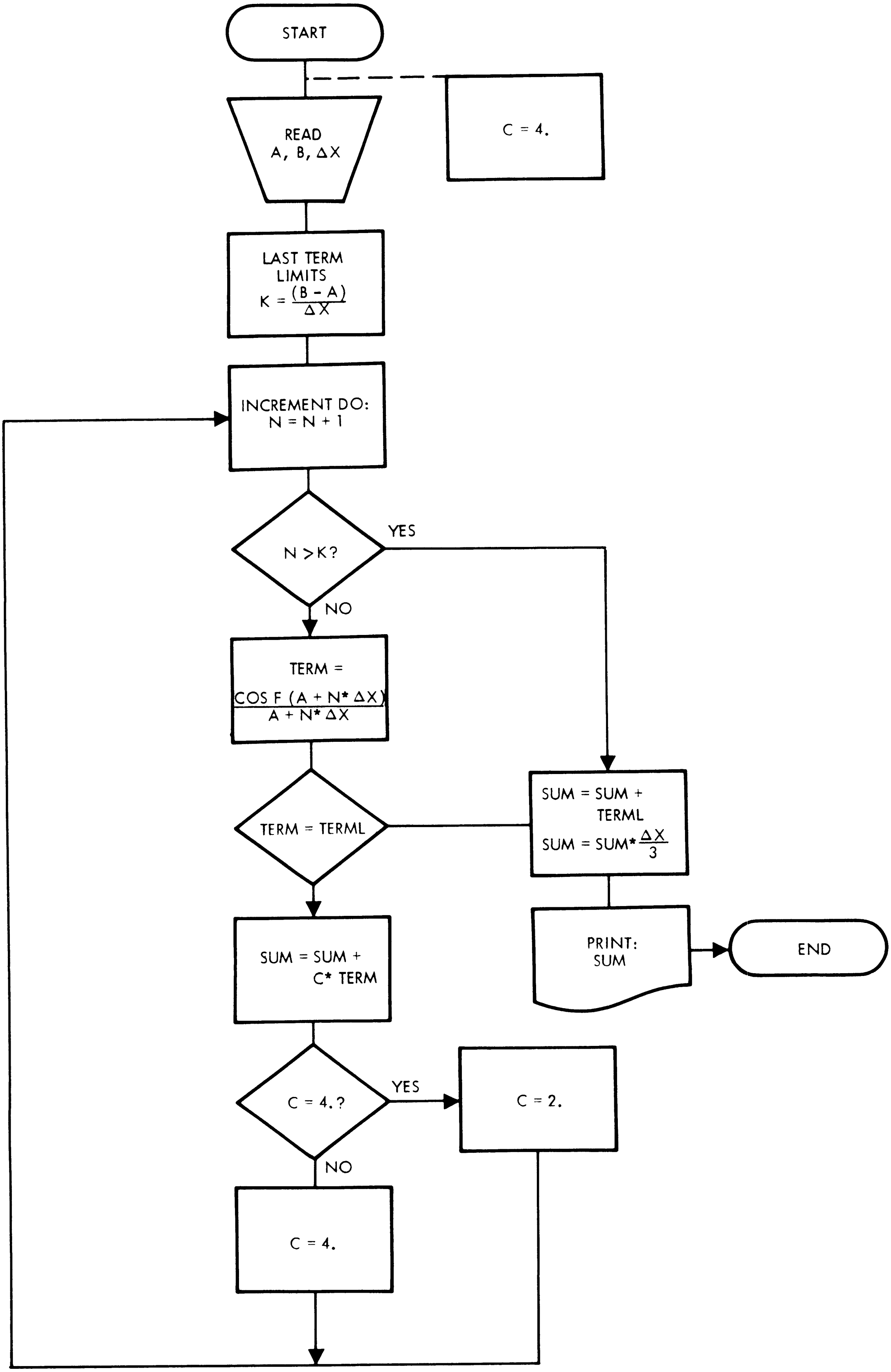
for the following possible values:

Variable	Range of Values
a	-6.99 to +6.99
b	-6.99 to +6.99
Δx	-.25 to +.25

Simpson's rule for approximating a definite integral is:

$$\int_a^b f(x)\,dx = \frac{\Delta x}{3}(f(a)+4f(a+\Delta x)+2f(a+2\Delta x)+4f(a+3\Delta x)+\ldots+f(b)$$

The last term is reached when $(a+k\Delta x)=b$, and when neither a 2 nor a 4 appears in front of the first or last term.



SAMPLE PROGRAM FLOWCHART

PROGRAMMER | DATE | PROGRAM

C Label CONT STATEMENT

```
FTN,B,L,A
      PROGRAM SMPSN
      READ(5,1Ø) A,B,DELTX
   1Ø FORMAT(2E8.2,E7.2)
      TERML=COS(B)/B
      SUM=COS(A)/A
      K=(B-A)/DELTX
      C=4.
      I=K+1
      DO 6Ø N=1,I
      FN=N
      IF(N-K)2Ø,2Ø,7Ø
   2Ø TERM=COS(A+FN*DELTX)/(A+FN*DELTX)
      IF(TERM-TERML)3Ø,7Ø,3Ø
   3Ø SUM=SUM+C*TERM
      IF(C-4.)5Ø,4Ø,5Ø
   4Ø C=2.
      GO TO 6Ø
   5Ø C=4.
   6Ø CONTINUE
   7Ø SUM=SUM+TERML
   8Ø SUM=(SUM*DELTX)/3.
      WRITE(6,9Ø) SUM
   9Ø FORMAT("SUM=",E8.2)
      STOP
      END
      END$
```

M0251

Ø = ZERO   O = ALPHA O   1 OR 1 = ONE   I = ALPHA I   LINE TERMINATED BY RETURN / LINE FEED (R/LF)
2 = TWO   Z = ALPHA Z   LINE IS DELETED BY RUBOUT BEFORE R/LF

OBJECT PROGRAM

Input and Output Data

```
    1.23     4.72     .25
SUM=-.63E+00
STOP
    1.23     2.01     .10
SUM=-.12E-01
STOP
    0.34     1.01     .02
SUM= .88E+00
STOP
    0.00     1.00     .01
SUM= .57E+36
STOP
    1.00     1.25     .05
SUM= .92E-01
STOP
```

APPENDIX D
FORTRAN ERROR MESSAGES

Errors detected in the source program are indicated by a numeric code inserted before or after the statement in the List Output.

The format is as follows:

E-eeee:	ssss + nnnn
eeee	The error diagnostic code shown below.
ssss	The statement label of the statement in which the error was detected. If unlabeled, 0000 is typed.
nnnn	Ordinal number of the erroneous statement following the last labeled statement. (Comment statements are not included in this count.)

Error Code — Description

0001 Statement label error:

a) The label is in positions other than 1-5.

b) A character in the label is not numeric.

c) The label is not in the range 1-9999.

d) The label is doubly defined.

e) The label indicated is used in a GO TO, DO, or IF statement or in an I/O operation to name a FORMAT statement, but it does not appear in the label field for any statement in the program (printed after END).

0002 Unrecognized Statement

a) The statement being processed is not recognized as a valid statement.

b) A specifications statement follows an executable statement.

Error Code | Description

c) The specification statements are not in the following order:

DIMENSION
COMMON
EQUIVALENCE

d) A statement function precedes a specification statement.

0003 Parenthesis error: There are an unequal number of left and right parentheses in a statement.

0004 Illegal character or format:

a) A statement contains a character other than A through Z, 0 through 9, or space =+-/(),.$".

b) A statement does not have the proper format.

c) A control statement is missing, misspelled, or does not have the proper format.

d) An indexing parameter of a DO-loop is not an unsigned integer constant or simple integer variable or is specified as zero.

0005 Adjacent operators: An arithmetic expression contains adjacent arithmetic operators.

0006 Illegal subscript: A variable name is used both as a simple variable and a subscripted variable.

0007 Doubly defined variable:

a) A variable name appears more than once in a COMMON statement.

b) A variable name appears more than once in a DIMENSION statement.

c) A variable name appears more than once as a dummy argument in a statement function.

Error Code — Description

d) A program subroutine, or function name appears as a dummy parameter; in a specifications statement of the subroutine or function; or as a simple variable in a program or subroutine.

0008 — Invalid parameter list:

a) The dummy parameter list for a subroutine or function exceeds 63.

b) Duplicate parameters appear in a statement function.

0009 — Invalid arithmetic expression:

a) Missing operator

b) Illegal replacement

0010 — Mixed mode expression: integer constants or variables appear in an arithmetic expression with real constants or variables.

0011 — Invalid subscript:

a) Subscript is not an integer constant, integer variable, or legal subscript expression.

b) There are more than two subscripts (i.e., more than two dimensions.)

c) Two subscripts appear for a variable which has been defined with one dimension only.

0012 — Invalid constant:

a) An integer constant is not in the range of -2^{15} to 2^{15} -1.

b) A real constant is not in the approximate range of 10^{38} to 10^{-38}.

c) A constant contains an illegal character.

Error Code	Description

0013 Invalid EQUIVALENCE statement:

a) Two or more of the variables appearing in an EQUIVALENCE statement are also defined in the COMMON block.

b) The variables contained in an EQUIVALENCE cause the origin of COMMON to be altered.

c) Contradictory equivalence; or equivalence between two or more arrays conflicts with a previously established equivalence.

0014 Table overflow: Too many variables and statement labels appear in the program.

0015 Invalid DO loop:

a) The terminal statement of a DO loop does not appear in the program or appears prior to the DO statement.

b) The terminal statement of a nested DO loop is not within the range of the outer DO loop.

c) DO loops are nested more than 10 deep.

d) Last statement in a loop is a GO TO, arithmetic IF, RETURN, STOP, PAUSE, or DO.

0016 Statement function name is doubly defined.

During execution of the object program, diagnostic messages may be printed on the output unit by the input/output system supplied for FORTRAN programs. When a halt occurs, the A-register contains a code which further defines the nature of the error:

Message	A-register	Explanation	Action
*FMT	000001	FORMAT error: a) w or d field does not contain proper digits. b) No decimal point after w field. c) w - d ≤ 4 for E specification.	Irrecoverable error; program must be recompiled.
*FMT	000002	a) FORMAT specifications are nested more than one level deep. b) A FORMAT statement contains more right parentheses than left parentheses.	Irrecoverable error; program must be recompiled.
*FMT	000003	a) Illegal character in FORMAT statement. b) Format repetition factor of zero. c) FORMAT statement defines more character positions than possible for device.	Irrecoverable error; program must be recompiled.
*FMT	000004	Illegal character in fixed field input item or number not right-justified in field.	Verify data.
*FMT	000005	A number has an illegal form (e.g., two Es, two decimal points, two signs, etc.).	Verify data.

INDEX

READER COMMENT SHEET

HP FORTRAN

HP 2116-9015 March 1974

We welcome your evaluation of this manual. Your comments and suggestions help us improve our publications. Please use additional pages if necessary.

Is this manual technically accurate?

Is this manual complete?

Is this manual easy to read and use?

Other comments?

FROM:

Name ______________________________

Company ______________________________

Address ______________________________

____

____

FOLD FOLD

FIRST CLASS
PERMIT NO. 141
CUPERTINO
CALIFORNIA

BUSINESS REPLY MAIL

No Postage Necessary if Mailed in the United States Postage will be paid by

Manager, Technical Publications
Hewlett-Packard
Data Systems Development Division
11000 Wolfe Road
Cupertino, California 95014

FOLD FOLD

HEWLETT hp PACKARD

SALES & SERVICE OFFICES

UNITED STATES

ALABAMA
8290 Whitesburg Dr., S.E.
P.O. Box 4207
Huntsville 35802
Tel: (205) 881-4591
TWX: 810-726-2204

Medical Only
228 W. Valley Ave.,
Room 220
Birmingham 35209
Tel: (205) 942-2081

ARIZONA
2336 E. Magnolia St.
Phoenix 85034
Tel: (602) 244-1361

2424 East Aragon Rd.
Tucson 85706
Tel: (602) 294-3148

***ARKANSAS**
Medical Service Only
P.O. Box 5646
Brady Station
Little Rock 72205
Tel:)501) 664-8773

CALIFORNIA
1430 East Orangethorpe Ave.
Fullerton 92631
Tel: (714) 870-1000

3939 Lankershim Boulevard
North Hollywood 91604
Tel: (213) 877-1282
TWX: 910-499-2170

6305 Arizona Place
Los Angeles 90045
Tel: (213) 649-2511
TWX: 910-328-6147

***Los Angeles**
Tel: (213) 776-7500

3003 Scott Boulevard
Santa Clara 95050
Tel: (408) 249-7000
TWX: 910-338-0518

***Ridgecrest**
Tel: (714) 446-6165

2220 Watt Ave.
Sacramento 95825
Tel: (916) 482-1463

9606 Aero Drive
P.O. Box 23333
San Diego 92123
Tel: (714) 279-3200

COLORADO
5600 South Ulster Parkway
Englewood 80110
Tel: (303) 771-3455

CONNECTICUT
12 Lunar Drive
New Haven 06525
Tel: (203) 389-6551
TWX: 710-465-2029

FLORIDA
P.O. Box 24210
2806 W. Oakland Park Blvd.
Ft. Lauderdale 33307
Tel: (305) 731-2020
TWX: 510-955-4099

***Jacksonville**
Medical Service only
Tel: (904) 725-6333

P.O. Box 13910
6177 Lake Ellenor Dr.
Orlando 32809
Tel: (305) 859-2900
TWX: 810-850-0311

P.O. Box 12826
Pensacola 32575
Tel: (904) 434-3081

GEORGIA
P.O. Box 105005
Atlanta 30348
Tel: (404) 434-4000
TWX: 810-766-4890

Medical Service Only
***Augusta** 30903
Tel: (404) 736-0592

HAWAII
2875 So. King Street
Honolulu 96814
Tel: (808) 955-4455

ILLINOIS
5500 Howard Street
Skokie 60076
Tel: (312) 677-0400
TWX: 910-223-3613

***St. Joseph**
Tel: (217) 469-2133

INDIANA
7301 North Shadeland Ave.
Indianapolis 46250
Tel: (317) 842-1000
TWX: 810-260-1796

IOWA
1902 Broadway
Iowa City 52240
Tel: (319) 338-9466
Night: (319) 338-9467

***KANSAS**
Derby
Tel: (316) 267-3655

KENTUCKY
Medical/Calculator Only
Atkinson Square
3901 Atkinson Dr.,
Suite 207
Louisville 40218
Tel: (502) 456-1573

LOUISIANA
P.O. Box 840
3239 Williams Boulevard
Kenner 70062
Tel: (504) 721-6201
TWX: 810-955-5524

MARYLAND
6707 Whitestone Road
Baltimore 21207
Tel: (301) 944-5400
TWX: 710-862-9157

2 Choke Cherry Road
Rockville 20850
Tel: (301) 948-6370
TWX: 710-828-9684

MASSACHUSETTS
32 Hartwell Ave.
Lexington 02173
Tel: (617) 861-8960
TWX: 710-326-6904

MICHIGAN
23855 Research Drive
Farmington Hills 48024
Tel: (313) 476-6400
TWX: 810-242-2900

MINNESOTA
2400 N. Prior Ave.
Roseville 55113
Tel: (612) 636-0700
TWX: 910-563-3734

MISSISSIPPI
***Jackson**
Medical Service only
Tel: (601) 982-9363

MISSOURI
11131 Colorado Ave.
Kansas City 64137
Tel: (816) 763-8000
TWX: 910-771-2087

148 Weldon Parkway
Maryland Heights 63043
Tel: (314) 567-1455
TWX: 910-764-0830

NEBRASKA
Medrcal Only
7171 Mercy Road
Suite 110
Omaha 68106
Tel: (402) 392-0948

NEW JERSEY
W. 120 Century Rd.
Paramus 07652
Tel: (201) 265-5000
TWX: 710-990-4951

NEW MEXICO
P.O. Box 11634
Station E
11300 Lomas Blvd., N.E.
Albuquerque 87123
Tel: (505) 292-1330
TWX: 910-989-1185

156 Wyatt Drive
Las Cruces 88001
Tel: (505) 526-2485
TWX: 910-983-0550

NEW YORK
6 Automation Lane
Computer Park
Albany 12205
Tel: (518) 458-1550
TWX: 710-441-8270

New York City
Manhattan, Bronx
Contact Paramus, NJ Office
Tel: (201) 265-5000
Brooklyn, Queens, Richmond
Contact Woodbury, NY Office
Tel: (516) 921-0300

201 South Avenue
Poughkeepsie 12601
Tel: (914) 454-7330
TWX: 510-248-0012

39 Saginaw Drive
Rochester 14623
Tel: (716) 473-9500
TWX: 510-253-5981

5858 East Molloy Road
Syracuse 13211
Tel: (315) 455-2486
TWX: 710-541-0482

1 Crossways Park West
Woodbury 11797
Tel: (516) 921-0300
TWX: 710-990-4951

NORTH CAROLINA
P.O. Box 5188
1923 North Main Street
High Point 27262
Tel: (919) 885-8101
TWX: 510-926-1516

OHIO
16500 Sprague Road
Cleveland 44130
Tel: (216) 243-7300
TWX: 810-423-9431

330 Progress Rd.
Dayton 45449
Tel: (513) 859-8202
TWX: 810-474-2818

1041 Kingsmill Parkway
Columbus 43229
Tel: (614) 436-1041

OKLAHOMA
P.O. Box 32008
Oklahoma City 73132
Tel: (405) 721-0200
TWX: 910-830-6862

OREGON
17890 SW Lower Boones
Ferry Road
Tualatin 97062
Tel: (503) 620-3350

PENNSYLVANIA
111 Zeta Drive
Pittsburgh 15238
Tel: (412) 782-0400
Night: 782-0401
TWX: 710-795-3124

1021 8th Avenue
King of Prussia Industrial Park
King of Prussia 19406
Tel: (215) 265-7000
TWX: 510-660-2670

SOUTH CAROLINA
6941-0 N. Trenholm Road
Columbia 29260
Tel: (803) 782-6493

TENNESSEE
***Memphis**
Medical Service only
Tel: (901) 274-7472

***Nashville**
Medical Service only
Tel: (615) 244-5448

TEXAS
P.O. Box 1270
201 E. Arapaho Rd.
Richardson 75080
Tel: (214) 231-6101
TWX: 910-867-4723

P.O. Box 27409
6300 Westpark Drive
Suite 100
Houston 77027
Tel: (713) 781-6000
TWX: 910-881-2645

205 Billy Mitchell Road
San Antonio 78226
Tel: (512) 434-8241
TWX: 910-871-1170

UTAH
2160 South 3270 West Street
Salt Lake City 84119
Tel: (801) 487-0715

VIRGINIA
Medical Only
P.O. Box 12778
No. 7 Koger Exec. Center
Suite 212
Norfolk 23502
Tel: (804) 497-1026/7

P.O. Box 9854
2914 Hungary Springs Road
Richmond 23228
Tel: (804) 285-3431
TWX: 710-956-0157

WASHINGTON
Bellefield Office Pk.
1203-114th Ave. S.E.
Bellevue 98004
Tel: (206) 454-3971
TWX: 910-443-2446

***WEST VIRGINIA**
Medical/Analytical Only
Charleston
Tel: (304) 345-1640

WISCONSIN
9004 West Lincoln Ave.
West Allis 53227
Tel: (414) 541-0550

FOR U.S. AREAS NOT LISTED:
Contact the regional office nearest you: Atlanta, Georgia; North Hollywood, California; Rockville, Maryland; Skokie, Illinois. Their complete addresses are listed above.

***Service Only**

CANADA

ALBERTA
Hewlett-Packard (Canada) Ltd.
11748 Kingsway Ave.
Edmonton T5G 0X5
Tel: (403) 452-3670
TWX: 610-831-2431 EDTH

Hewlett-Packard (Canada) Ltd.
915-42 Avenue S.E. Suite 102
Calgary T2G 1Z1
Tel: (403) 287-1672
TWX: 610-821-6141

BRITISH COLUMBIA
Hewlett-Packard (Canada) Ltd.
837 E. Cordova Street
Vancouver V6A 3R2
Tel: (604) 254-0531
TWX: 610-922-5059 VCR

MANITOBA
Hewlett-Packard (Canada) Ltd.
513 Century St.
St. James
Winnipeg R3H 0L8
Tel: (204) 786-7581
TWX: 610-671-3531

NOVA SCOTIA
Hewlett-Packard (Canada) Ltd.
800 Windmill Road
P.O. Box 931
Dartmouth B2Y 3Z6
Tel: (902) 469-7820
TWX: 610-271-4482 HFX

ONTARIO
Hewlett-Packard (Canada) Ltd.
1785 Woodward Dr.
Ottawa K2C 0P9
Tel: (613) 225-6530
TWX: 610-562-8968

Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
Mississauga L4V 1L9
Tel: (416) 678-9430
TWX: 610-492-4246

QUEBEC
Hewlett-Packard (Canada) Ltd.
275 Hymus Blvd.
Pointe Claire H9R 1G7
Tel: (514) 697-4232
TWX: 610-422-3022
TLX: 05-821521 HPCL

Hewlett-Packard (Canada) Ltd.
2376 Galvani Street
Ste-Foy G1N 4G4
Tel: (418) 688-8710
TWX: 610-571-5525

FOR CANADIAN AREAS NOT LISTED:
Contact Hewlett-Packard (Canada) Ltd. in Mississauga.

CENTRAL AND SOUTH AMERICA

ARGENTINA
Hewlett-Packard Argentina
S.A.C.e.I
Lavalle 1171-3° Piso
Buenos Aires
Tel: 35-0436, 35-0627, 35-0341
Telex: Public Booth No. 9
Cable: HEWPACK ARG

BOLIVIA
Stambuk & Mark (Bolivia) Ltda.
Av. Mariscal, Santa Cruz 1342
La Paz
Tel: 40626, 53163, 52421
Telex: 3560014
Cable: BUKMAR

BRAZIL
Hewlett-Packard Do Brasil
I.E.C. Ltda.
Rua Frei Caneca, 1140/52 Bela Vista
01307-**São Paulo**-SP
Tel: 288-71-11, 287-81-20,
287-61-93
Telex: 391-112-3602 HPBR-BR
Cable: HEWPACK São Paulo

Hewlett-Packard Do Brasil
I.E.C. Ltda.
Praca Dom Feliciano, 78-8°
andar (Sala 806/8)
90000-**Pôrto Alegre**-RS
Tel: 25-84-70-DDD (0512)
Cable: HEWPACK Pôrto Alegre

Hewlett-Packard Do Brasil
I.E.C. Ltda.
Rua Siqueira Campos, 53, 4°
andar-Copacabana
20000-**Rio de Janeiro**-GB
Tel: 257-80-94-DDD (021)
Telex: 391-212-1905 HEWP-BR
Cable: HEWPACK
Rio de Janeiro

CHILE
Calcagni y Metcalfe Ltda.
Alameda 580-Of. 807
Casilla 2118
Santiago, 1
Tel: 398613
Telex: 350001 CALMET
Cable: CALMET Santiago

Medical Only
General Machinery Co., Ltda.
Paraguay 494
Casilla 13910
Santiago
Tel: 31123, 31124
Cable: GEMCO Santiago

COLOMBIA
Instrumentación
Henrik A. Langebaek & Kier S.A.
Carrera 7 No. 48-75
Apartado Aéreo 6287
Bogotá, I.D.E.
Tel: 69-88-77
Cable: AARIS Bogotá
Telex: 044-400

COSTA RICA
Cientifica Costarricense S.A.
Calle Central, Avenidas 1 y 3
Apartado 10159
San José
Tel: 21-86-13
Cable: GALGUR San José

ECUADOR
Medical Only
A.F. Viscaino Compañía Ltda.
Av. Rio Amazonas No. 239
P.O. Box 2925
Quito
Tel: 527-088,527-804
Cable: ASTOR Quito

Calculators Only
Computadoras y Equipos
Electrónicos
P.O. Box 2695
Avda. 12 De Octubre No. 2207
Quito
Tel: 233869, 236783
Telex: 02-2113 Sagita Ed
Cable: Sagitra-Quito

EL SALVADOR
IPESA
Bulevar de los Heroes 11-48
San Salvador
Tel: 252787

GUATEMALA
IPESA
Avenida La Reforma 3-48,
Zona 9
Guatemala City
Tel: 63627, 64786
Telex: 4192 Teletro Gu

MEXICO
Hewlett-Packard Mexicana,
S.A. de C.V.
Torres Adalid No. 21, 11° Piso
Col. del Valle
Mexico 12, D.F.
Tel: (905) 543-42-32
Telex: 017-74-507

Hewlett-Packard Mexicana,
S.A. de C.V.
Ave. Constitución No. 2184
Monterrey, N.L.
Tel: 48-71-32, 48-71-84
Telex: 038-843

NICARAGUA
Roberto Terán G.
Apartado Postal 689
Edificio Terán
Managua
Tel: 25114, 23412,23454
Cable: ROTERAN Managua

PANAMA
Electrónico Balboa, S.A.
P.O. Box 4929
Calle Samuel Lewis
Cuidad de Panama
Tel: 64-2700
Telex: 3431103 Curunda,
Canal Zone
Cable: ELECTRON Panama

PARAGUAY
Z.J. Melamed S.R.L.
División: Aparatos y Equipos
Médicos
División: Aparatos y Equipos
Científicos y de Investigación
P.O. Box 676
Chile-482, Edificio Victoria
Asuncion
Tel: 4-5069, 4-6272
Cable: RAMEL

PERU
Compañía Electro Médica S.A.
Los Flamencos 145
San Isidro Casilla 1030
Lima 1
Tel: 413485
Cable: ELMED Lima

PUERTO RICO
Hewlett-Packard Inter-Americas
Puerto Rico Branch Office
P.O. Box 29081
65th Inf. Station
San Juan 00929
Calle 272, Urb. Country Club
Carolina 00639
Tel: (809) 762-7355/7455/7655
Telex: 3450514

URUGUAY
Pablo Ferrando S.A.
Comercial e Industrial
Avenida Italia 2877
Casilla de Correo 370
Montevideo
Tel: 40-3102
Cable: RADIUM Montevideo

VENEZUELA
Hewlett-Packard de Venezuela
C.A.
Apartado 50933, Caracas 105
Edificio Segre
Tercera Transversal
Los Ruices Norte
Caracas 107
Tel: 35-01-07, 35-00-84,
35-00-65, 35-00-31
Telex: 25146 HEWPACK
Cable: HEWPACK Caracas

FOR AREAS NOT LISTED, CONTACT:
Hewlett-Packard
Inter-Americas
3200 Hillview Ave.
Palo Alto, California 94304
Tel: (415) 493-1501
TWX: 910-373-1260
Cable: HEWPACK Palo Alto
Telex: 034-8300, 034-8493

EUROPE

AUSTRIA
Hewlett-Packard Ges m b H
Handelska 52/3
P O Box 7
A-1205 **Vienna**
Tel (0222) 35 16 21 to 32
Cable HEWPAK Vienna
Telex 75923 hewpak a

BELGIUM
Hewlett-Packard Benelux
S A /N V
Avenue de Col-Vert, 1
(Groenkraaglaan)
B-1170 **Brussels**
Tel (02) 672 22 40
Cable PALOBEN Brussels
Telex 23 494 paloben bru

DENMARK
Hewlett-Packard A/S
Datavej 52
DK-3460 **Birkerød**
Tel (02) 81 66 40
Cable HEWPACK AS
Telex 166 40 hpas

Hewlett-Packard A/S
Navervej 1
DK-8600 **Silkeborg**
Tel (06) 82 71 66
Telex 166 40 hpas
Cable HEWPACK AS

FINLAND
Hewlett-Packard OY
Nahkahousuntie 5
P.O Box 6
SF-00211 **Helsinki** 21
Tel 6923031
Cable HEWPACKOY Helsinki
Telex 12-1563

FRANCE
Hewlett-Packard France
Quartier de Courtaboeuf
Boite Postale No 6
F-91401 **Orsay**
Tel (1) 907 78 25
Cable HEWPACK Orsay
Telex 600048

Hewlett-Packard France
"Le Saquin"
Chemin des Mouilles
Boite Postale No 12
F-69130 **Ecully**
Tel (78) 33 81 25,
Cable HEWPACK Eculy
Telex 310617

Hewlett-Packard France
Agence Régionale
Péricentre de la Cépière
Chemin de la Cépière, 20
F-31300 **Toulouse-Le Mirail**
Tel:(61) 40 11 12
Telex 510957F

Hewlett-Packard France
Agence Régionale
Aéroport principal de
Marseille-Marignane
F-13721**Marignane**
Tel (91) 89 12 36
Cable HEWPACK MARGN
Telex 410774F

Hewlett-Packard France
Agence Régionale
63, Avenue de Rochester
F-35000 **Rennes**
Tel (99) 36 33 21
Cable HEWPACK 74912
Telex 740912F

Hewlett-Packard France
Agence Régionale
74, Allée de la Robertsau
F-67000 **Strasbourg**
Tel (88) 35 23 20/21
Telex 890141
Cable HEWPACK STRBG

Medical/Calculator Only
Hewlett-Packard France
Agence Régionale
Centre Vauban
201, rue Colbert
Entrée A2
F-59000 **Lille**
Tel (20) 51 44 14
Telex 820744

GERMAN FEDERAL REPUBLIC
Hewlett-Packard GmbH
Vertriebszentrale Frankfurt
Bernerstrasse 117
Postfach 560 140
D-6000 **Frankfurt** 56
Tel (0611) 50 04-1
Cable HEWPACKSA Frankfurt
Telex 04 13249 hpFFmd

Hewlett-Packard GmbH
Technisches Buero Böblingen
Herrenbergerstrasse 130
D-7030 **Böblingen** Wurttemberg
Tel (07031) 667-1
Cable HEPAK Böblingen
Telex 07265739 bbn

Hewlett-Packard GmbH
Technisches Buero Dusseldorf
Emanuel-Leutze-Str 1 (Seestern)
D-4000 **Dusseldorf**
Tel (0211) 59711
Telex 85/86 533 hpdd d

Hewlett-Packard GmbH
Technisches Buero Hamburg
Wendenstrasse 23
D-2000 **Hamburg** 1
Tel (040) 24 13 93
Cable HEWPACKSA Hamburg
Telex 21 63 032 hphh d

Hewlett-Packard GmbH
Technisches Buero Hannover
Mellendorfer Strasse 3
D-3000 **Hannover-Kleefeld**
Tel (0511) 55 60 46
Telex 092 3259

Hewlett-Packard GmbH
Technisches Buero Nuremberg
Neumeyer Str 90
D-8500 **Nuremberg**
Tel (0911) 56 30 83/85
Telex 0623 860

Hewlett-Packard GmbH
Technisches Buero Munchen
Unterhachinger Strasse 28
ISAR Center
D-8012 **Ottobrunn**
Tel (089) 601 30 61 7
Telex 52 49 85
Cable HEWPACKSA Munchen
Telex 0524985

(West Berlin)
Hewlett-Packard GmbH
Technisches Buero Berlin
Keith Strasse 2-4
D-1000 **Berlin** 30
Tel (030) 24 90 86
Telex 18 3405 hpbln d

GREECE
Kostas Karayannis
18 Ermou Street
GR-**Athens** 126
Tel 3237731
Cable RAKAR Athens
Telex 21 59 62 rkar gr

Analytical Only
INTECO G Papathanassiou & Co
Marini 17
GR - **Athens** 103
Tel 521 915
Cable INTEKNIKA
Telex 21 5329 INTE GR

Medical Only
Technomed Hellas Ltd
52 Skoufa Street
GR - **Athens** 135
Tel 626 972 663 930 614 959
Cable ETALAK Athens
Telex 21-4693 ETAL GR

ICELAND
Medical Only
Elding Trading Company Inc
Hafnarhovoli - Tryggvatotu
IS-**Reykjavik**
Tel 1 58 20
Cable ELDING Reykjavik

IRAN
Hewlett-Packard Iran Ltd
Mir-Emad Avenue
14th Street No 19
IR-**Tehran**
Tel 85 10 82 86

IRELAND
Hewlett-Packard Ltd
King Street Lane
Winnersh Wokingham
GB-**Berkshire** RG11 5AR
Tel (0734) 78 47 74
Telex 847178/848179

ITALY
Hewlett-Packard Italiana S p A
Casella postale 3645
I 20124 **Milano**
Tel (2) 6251 (10 lines)
Cable HEWPACKIT Milano
Telex 32046

Hewlett-Packard Italiana S p A
Via Pietro Maroncelli 40
(ang Via Visentin)
I-35100 **Padova**
Tel (49) 66 48 88
Telex 32046 via Milano

Medical only
Hewlett-Packard Italiana S p A
Via d Aghiardi 7
I-56100 **Pisa**
Tel (050) 2 32 04
Telex 32046 via Milano

Hewlett-Packard S p A
Via G Armellini 10
I-00143 **Roma**-Eur
Tel (06) 54 69 61
Telex 61514
Cable HEWPACKIT Roma

Hewlett-Packard Italiana S p A
Via San Quintino 46
I-10121 **Turino**
Tel 53 82 64 54 84 68
Telex 32046 via Milano

Medical Calculators Only
Hewlett-Packard Italiana S p A
Via Principe Nicola 43 G/C
I-95126 **Catania**
Tel (095) 37 05 05

Hewlett-Packard Italiana S p A
Via Amerigo Vespucci 9
I-80142 **Napoli**
Tel (81) 33 77 11

Hewlett-Packard Italiana S p A
Via E Masi 9 B
I-40137 **Bologna**
Tel (051) 30 78 87

LUXEMBURG
Hewlett-Packard Benelux
S A N V
Avenue du Col-Vert 1
(Groenkraaglaan)
B 1170 **Brussels**
Tel (02) 672 22 40
Cable PALOBEN Brussels
Telex 23 494

NETHERLANDS
Hewlett-Packard Benelux N V
Van Heuven Goedhartlaan 121
P O Box 667
NL- **Amstelveen** 1134
Tel (020) 47 20 21
Cable PALOBEN Amsterdam
Telex 13 216 hepa nl

NORWAY
Hewlett-Packard Norge A S
Nesveien 13
Box 149
N 1344 **Haslum**
Tel (02) 53 83 60
Telex 16621 hpnas n

POLAND
BIURO INFORMACJI TECHNICZNEJ
Hewlett-Packard
U1 Stawki 2 6P
00-950**Warsaw**
Tel 39 67 43
Telex 81 24 53 hepa pl

PORTUGAL
Telectra-Empresa Técnica de
Equipamentos Eléctricos S a r l
Rua Rodrigo da Fonseca 103
P O Box 2531
P-**Lisbon** 1
Tel (19) 68 60 72
Cable TELECTRA Lisbon
Telex 12598

Mundinter
Intercambio Mundial de Comércio
S a r l
Av A A de Aguiar 138
P O Box 2761
P - **Lisbon**
Tel (19) 53 21 31 7
Cable INTERCAMBIO Lisbon

RUMANIA
Hewlett-Packard Technical Office
BD N Balcesci 16
Bucharest
Tel 1580 23 138885
Telex 10440

SPAIN
Hewlett-Packard Española S A
Jerez No 3
E-**Madrid** 16
Tel (1) 458 26 00 (10 lines)
Telex 23515 hpe

Hewlett-Packard Española, S A
Milanesado 21-23
E-**Barcelona** 17
Tel (3) 203 6200 (5 lines)
Telex 52603 hpbe e

Hewlett-Packard Española, S A
Av Ramón y Cajal 1-9
(Edificio Sevilla I)
E-**Seville 5**
Tel 64 44 54 58

Hewlett-Packard española S A
Edificio Albia II 7 B
E-**Bilbao**
Tel 23 83 06/23 82 06

Calculators Only
Hewlett-Packard Española S A
Gran Via Fernando El Católico, 67
E-**Valencia**-8
Tel 326 67 28/326 85 55

SWEDEN
Hewlett-Packard Sverige AB
Enighetsvägen 3
Fack
S-161 20 **Bromma** 20
Tel (08) 730 05 50
Cable MEASUREMENTS
Stockholm
Telex 10721

Hewlett-Packard Sverige AB
Frotallsgatan 30
S-421 32 **Vastra Frolunda**
Tel (031) 49 09 50
Telex 10721 Via Bromma Office

SWITZERLAND
Hewlett-Packard (Schweiz) AG
Zurcherstrasse 20
P O Box 307
CH-8952 Schlieren **Zurich**
Tel (01) 98 18 21
Cable HPAG CH
Telex 53933 hpag ch

Hewlett-Packard (Schweiz) AG
9 Chemin Louis-Pictet
CH-1214 Vernier-**Geneva**
Tel (022) 41 49 50
Cable HEWPACKAG Geneva
Telex 27 333 hpag ch

TURKEY
Telekom Engineering Bureau
P O Box 437
Bevglu
TR-**Istanbul**
Tel 49 40 40
Cable TELEMATION Istanbul

UNITED KINGDOM
Hewlett-Packard Ltd
King Street Lane
GB-**Winnersh**, Wokingham
Berks RG11 5AR
Tel (0734) 78 47 74
Cable Hewpie London
Telex 847178/9

Hewlett-Packard Ltd
"The Graftons"
Stamford New Road
GB-**Altrincham**
Cheshire WA14 IDJ
Tel (061) 928 9021
Cable Hewpie Manchester
Telex 668068

Hewlett-Packard Ltd
Lygon Court
Dudley Road
GB-**Halesowen**, Worcs
Tel (021) 550 7053
Telex (021) 550 7273

Hewlett-Packard Ltd
4th Floor
Wedge House
799 London Road
GB-**Thornton Heath** CR4 6XL
Surrey
Tel (01) 684 0103/0105
Telex 946825

Hewlett-Packard Ltd
c/o Makro
South Service Wholesale Centre
Wear Industrial Estate
Washington
GB-**New Town** County Durham
Tel Washington 464001 ext 57/58

Hewlett-Packard Ltd s registered
address for V A T purposes
only
70, Finsbury Pavement
GB-**London** EC2A1SX
Registered No 690597

USSR
Hewlett-Packard Representative
Office USSR
Pokrovsky Boulevard 4/17, Suite 12
Moscow101000
Tel 294 2024
Telex 7825 hewpak SU

YUGOSLAVIA
Iskra-standard Hewlett-Packard
Office
Miklosiceva 38/VII
61000 **Ljubljana**
Tel 315-879/321-674
Telex 31583 YU HEWPAK

AFRICA, ASIA, AUSTRALIA

AMERICAN SAMOA
Calculators Only
Oceanic Systems Inc
P.O Box 777
Pago Pago Bayfront Road
Pago Pago 96799
Tel 633-5513
Cable OCEANIC-Pago Pago

ANGOLA
Telectra
Empresa Técnica de
Equipamentos
Eléctricos, S A R L
R Barbosa Rodrigues, 42-I°DT
Caixa Postal, 6487
Luanda
Tel 35515/6
Cable TELECTRA Luanda

AUSTRALIA
Hewlett-Packard Australia
Pty Ltd
31-41 Joseph Street
Blackburn, Victoria 3130
P.O Box 36
Doncaster East, Victoria 3109
Tel 89-6351
Telex 31-024
Cable HEWPARD Melbourne

Hewlett-Packard Australia
Pty. Ltd.
31 Bridge Street
Pymble
New South Wales, 2073
Tel 449-6566
Telex 21561
Cable HEWPARD Sydney

Hewlett-Packard Australia
Pty Ltd
153 Greenhill Road
Parkside, 5063, S A
Tel 27-2591
Telex 82536 ADEL
Cable HEWPARD ADELAIDE

Hewlett-Packard Australia
Pty Ltd
141 Stirling Highway
Nedlands, W A 6009
Tel 86-5455
Telex 93859 PERTH
Cable HEWPARD PERTH

Hewlett-Packard Australia
Pty Ltd
121 Wollongong Street
Fyshwick, A C T 2609
Tel 95-3733
Telex 62650 Canberra
Cable HEWPARD CANBERRA

Hewlett Packard Australia
Pty Ltd
5th Floor
Teachers Union Building
495-499 Boundary Street
Spring Hill, 4000 Queensland
Tel 29-1544
Telex 42133 BRISBANE

CYPRUS
Kypronics
19 Gregorios & Xenopoulos Rd
P O Box 1152
CY-**Nicosia**
Tel 45628/29
Cable KYPRONICS PANDEHIS

GUAM
Medical/Pocket Calculators Only
Guam Medical Suply, Inc
Jay Ease Building, Room 210
P O Box 8383
Tamuning 96911
Tel 646-4513
Cable EARMED Guam

HONG KONG
Schmidt & Co (Hong Kong) Ltd
P O Box 297
Connalight Centre
39th Floor
Connaught Road, Central
Hong Kong
Tel H-255291-5
Telex 74766 SCHMC HX
Cable SCHMIDTCO Hong Kong

INDIA
Blue Star Ltd
Kasturi Buildings
Jamshedji Tata Rd
Bombay 400 020
Tel 29 50 21
Telex 2156
Cable BLUEFROST

Blue Star Ltd
Sahas
414/2 Vir Savarkar Marg
Prabhadevi
Bombay 400 025
Tel 45 78 87
Telex 4093
Cable FROSTBLUE

Blue Star Ltd
Band Box House
Prabhadevi
Bombay 400 025
Tel 45 73 01
Telex 3751
Cable BLUESTAR

Blue Star Ltd
14/40 Civil Lines
Kanpur 208 001
Tel 6 88 82
Telex 292
Cable BLUESTAR

Blue Star Ltd
7 Hare Street
P O Box 506
Calcutta 700 001
Tel 23-0131
Telex 7655
Cable BLUESTAR

Blue Star Ltd
Blue Star House
34 Mahatma Gandhi Rd
Lajpatnagar
New Delhi 110 024
Tel 62 32 76
Telex 2463
Cable BLUESTAR

Blue Star Ltd
Blue Star House
11/11A Magarath Road
Bangalore 560 025
Tel 55668
Telex 430
Cable BLUESTAR

Blue Star Ltd
Meeakshi Mandiran
xxx/1678 Mahatma Gandhi Rd
Cochin 682 016 Kerala
Tel 32069, 32161, 32282
Telex 046-514
Cable BLUESTAR

Blue Star Ltd
1-1-117/1
Sarojini Devi Road
Secunderabad 500 003
Tel 70126, 70127
Cable BLUEFROST
Telex 459

Blue Star Ltd
23/24 Second Line Beach
Madras 600 001
Tel 23954
Telex 379
Cable BLUESTAR

Blue Star Ltd
Nathraj Mansions
2nd Floor Bistupur
Jamshedpur 831 001
Tel 7383
Cable BLUESTAR
Telex 240

INDONESIA
BERCA Indonesia P T
P.O Box 496
1st Floor JL Cikini Raya 61
Jakarta
Tel 56038, 40369, 49886
Telex 42895
Cable BERCACON

BERCA Indonesia P T
63 JL Raya Gubeng
Surabaya
Tel 44309

IRAN
Hewlett-Packard Iran Ltd
Mir-Emad Ave
I4th Street No I9
IR-**Tehran**
Tel 851082 86
Telex 21 25 74

ISRAEL
Electronics & Engineering Div
of Motorola Israel Ltd
16 Kremenetski Street
P O Box 25016
Tel-Aviv
Tel 38973
Telex 33569
Cable BASTEL Tel-Aviv

JAPAN
Yokogawa-Hewlett-Packard Ltd
Ohashi Building
1-59-1 Yoyogi
Shibuya-ku **Tokyo**
Tel 03-370-2281/92
Telex 232-2024YHP
Cable YHPMARKET TOK 23-724

Yokogawa-Hewlett-Packard Ltd
Nissei Ibaraki Building
2-8 Kasuga 2-chrome Ibaraki-shi
Osaka 567
Tel (0726) 23-1641
Telex 5332-385 YHP OSAKA

Yokogawa-Hewlett-Packard Ltd
Nakamo Building
24 Kami Sasajima-cho
Nakamura-ku **Nagoya** 450
Tel (052) 571-5171

Yokogawa-Hewlett-Packard Ltd
Tanigawa Building
2-24-1 Tsuruya-choo
Kanagawa-ku
Yokohama 221
Tel 045-312-1252
Telex 382-3204 YHP YOK

Yokogawa-Hewlett-Packard Ltd
Mito Mitsui Building
105 1-chrome San-no-maru
Mito Ibaragi 310
Tel 0292-25-7470

Yokogawa-Hewlett-Packard Ltd
Inoue Building
1348-3 Asahi-cho, 1-chome
Atsugi Kanagawa 243
Tel 0462-24-0452

KENYA
Technical Engineering
Services(E A)Ltd
P O Box 18311
Nairobi
Tel 557726/556762
Cable PROTON

Medical Only
International Aeradio(E A)Ltd
P O Box 19012
Nairobi Airport
Nairobi
Tel 336055/56
Telex 22201/22301
Cable INTAERID Nairobi

KOREA
American Trading Company
Korea
C P O Box 1103
Dae Kyung Bldg , 8th Floor
107 Sejong-Ro
Chongro-Ku **Seoul**
Tel (4 lines) 73-8924-7
Telex K-28338
Cable AMTRACO Seoul

LEBANON
Constantin E Macridis
Clemenceau Street 34
P O Box 7213
RL-**Beirut**
Tel 36 63 97/8
Telex 21114 Leb
Cable ELECTRONUCLEAR Beirut

MALAYSIA
Teknik Mutu Sdn Bhd
2 Lorong 13/6A
Section 13
Petaling Jaya **Selangor**
Tel 773455(5 lines)

MOZAMBIQUE
A N Goncalves Lta
162 1 Apt 14 Av D Luis
Caixa Postal 107
Lourenco Marques
Tel 27091 27114
Telex 6-203 Negon Mo
Cable NEGON

NEW ZEALAND
Hewlett Packard (N Z) Ltd
4-12 Cruickshank Street
Kilbirnie, Wellington 3
Mailing Address Hewlett-Packard
(N Z) Ltd
P O Box 9443
Courtney Place
Wellington
Tel 877-199
Telex NZ 3839
Cable HEWPACK Wellington

Hewlett-Packard (N Z) Ltd
Pakuranga Professional Centre
267 Pakuranga Highway
Box 51092
Pakuranga
Tel 569-651
Telex NZ 3839
Cable HEWPACK Auckland

Analytical/Medical Only
Medical Supplies N Z Ltd
Scientific Division
79 Carlton Gore Rd , Newmarket
P O Box 1234
Auckland
Tel 75-289
Telex 2958 MEDISUP
Cable DENTAL Auckland

Analytical/Medical Only
Medical Supplies N Z Ltd
P O Box 1994
147-161 Tory St
Wellington
Tel 850-799
Telex 3858
Cable DENTAL Wellington

Analytical Medical Only
Medical Supplies N Z Ltd
P O Box 309
239 Stanmore Road
Christchurch
Tel 892-019
Cable DENTAL Christchurch

Analytical Medical Only
Medical Supplies N Z Ltd
303 Great King Street
P O Box 233
Dunedin
Tel 88-817
Cable DENTAL Dunedin

NIGERIA
The Electronics
instrumentations Ltd
N6B 770 Oyo Road
Oluseun House
P M B 5402
Ibadan
Tel 61577
Telex 31231 TEIL Nigeria
Cable THETEIL Ibadan

The Electronics Instrumentations Ltd
144 Agege Motor Road, Mushin
P O Box 6645
Lagos
Cable THETEIL Lagos

PAKISTAN
Mushko & Company, Ltd
Oosman Chambers
Abdullah Haroon Road
Karachi-3
Tel 511027 512927
Telex KR894
Cable COOPERATOR Karachi

Mushko & Company Ltd
38B Satellite Town
Rawalpindi
Tel 41924
Cable FEMUS Rawalpindi

PHILIPPINES
The Online Advanced Systems
Corporation
6th Floor Yujuico Building
560 Quintin Paredes Street
Binondo **Manila**
Tel 40-05-41, 40-05-31
Telex 3327 GENBANK

RHODESIA
Field Technical Sales
45 Kelvin Road North
P O Box 3458
Salisbury
Tel 705231 (5 lines)
Telex RH 4122

SINGAPORE
Hewlett-Packard Singapore
(Pte) Ltd
Blk 2, 6th Floor, Jalan
Bukit Merah
Redhill Industrial Estate
Alexandra P O Box 58
Singapore 3
Tel 633022
Telex HPSG RS 21486
Cable HEWPACK, Singapore

SOUTH AFRICA
Hewlett-Packard South Africa
(Pty) Ltd
Private Bag Wendywood
Sandton Transvaal 2144
Hewlett-Packard House
Daphne Street, Wendywood,
Sandton, Transvaal 2144
Tel 802-104016
Telex SA43-4782JH
Cable HEWPACK JOHANNESBURG

Hewlett-Packard South Africa
(Pty), Ltd
P O Box 120
Howard Place Cape Province,
7450
Pine Park Center, Forest Drive,
Pinelands, Cape Province, 7405
Tel 53-7955 thu 9
Telex 57-0006

Hewlett-Packard South Africa
(Pty) Ltd
P O Box 37099
Overport, Durban 4067
641 Ridge Road Durban
Durban, 4001
Tel 88-7478/88-1080/88-2520
Telex 6-7954
Cable HEWPACK

TAIWAN
Hewlett-Packard Far East Ltd ,
Taiwan Branch
39 Chung Shiao West Road
Sec 1, 7th Floor
Taipei
Tel 389160 1 2 3
Telex 21824 HEWPACK
Cable HEWPACK TAIPEI

Hewlett-Packard Taiwan
38, Po-Ai Lane, San Min Chu,
Kaohsiung
Tel (07) 242318

Analytical Only
San Kwang Instruments Co , Ltd ,
No 20, yung Sui Road
Taipei 100
Tel 3713171-4
Telex 22894 SANKWANG
Cable SANKWANG TAIPEI

TANZANIA
Medical Only
International Aeradio (E A), Ltd
P O Box 861
Daressalaam
Tel 21251 Ext 265
Telex 41030

THAILAND
UNIMESA Co , Ltd
Elcom Research Building
Bangjak Sukumvit Ave
Bangkok
Tel 932387 930338
Cable UNIMESA Bangkok

UGANDA
Medical Only
International Aeradio(E A), Ltd ,
P O Box 2577
Kampala
Tel 54388
Cable INTAERIO Kampala

ZAMBIA
R J Tilbury (Zambia) Ltd
P O Box 2792
Lusaka
Tel 73793
Cable ARJAYTEE, Lusaka

MEDITERRANEAN AND MIDDLE EAST COUNTRIES NOT SHOWN PLEASE CONTACT:
Hewlett-Packard S A
Mediterranean and Middle
East Operations
35, Kolokotroni Street
Platia Kefallariou
GR-Kifissia-**Athens**
Telex 21-6588
Cable HEWPACKSA Athens

OTHER AREAS NOT LISTED, CONTACT:
Hewlett-Packard Intercontinental
3200 Hillview Ave
Palo Alto, California 94304
Tel (415) 493-1501
TWX 910-373-1267
Cable HEWPACK Palo Alto
Telex 034-8300, 034-8493

HEWLETT hp PACKARD

MANUAL PART NO. 02116-9015
MICROFICHE PART NO. 02116-91784

Printed: MAR 1974
Printed in U.S.A.