1963 seconda serie

Pubblicazioni del

CENTRO STUDI CALCOLATRICI ELETTRONICHE

del C. N. R.

presso l'Università degli studi di Pisa

n. 13

LITO-OFFSET FELICI S.
PISA

O. G. MANCINO

FORTRAN CEP language

a FORTRAN II version for the C.E.P.

OTTOBRE 1963 - PISA

1. INTRODUCTION

The FORTRAN CEP language differs from FORTRAN II, as described in [1], mainly because:

(1) it extends the variety of the modes for real quantities;

(2) it allows the addressing of a greater number of input-output equipments;

(3) it removes the limitations on the list of quantities to be transmitted between the magnetic core memory and the drum or the magnetic tape units.

A program, written in FORTRAN II language, may be compiled by the FORTRAN CEP translator, which has been already realized, with only a few modifications.

The definition of the FORTRAN CEP language is the object of the present paper.

2. CHARACTERISTIC FEATURES OF THE CEP COMPUTER

The CEP has been built at the University of Pisa. It is a parallel binary computer with a 36 bit word lenght. Instructions are single address with the possibility of two modifications [2]. Every instruction occupies one word. The main (magnetic core) memory has 8192 words. The auxiliary memory is composed of a magnetic drum with 16384 words and of eight magnetic tape units.

There are 128 instructions and 219 pseudo-instructions which refer to arbitrary closed subroutines [2].

Arithmetic is fixed or floating point.

The input is performed by three photo-electric paper tape readers, and the output is realized by means of one on-line typewriter, three paper tape high speed punches and one on-line printer. In addition, the CEP is equipped with one off-line printer.

3. DEFINITION OF FORTRAN CEP LANGUAGE

FORTRAN CEP language will be defined stressing the differences between it and FORTRAN II. Therefore parts and chapters of [1] will be referred to respectively by two numbers between brackets and separated by a comma. When such a reference is associated with the title of a section of this paper, everything in [1], related to that title and not redefined in the same section or in others, must be intended as in [1] changing at most the word FORTRAN to FORTRAN CEP.

3.1. A PROGRAM AND ITS CONSTITUENT CHARACTERS

A program consists of a sequence of the statements described in the course of this paper.

The characters that can be used to compose such statements are:

(1) the capital letters of the English alphabet
(2) the decimal digits
(3) the special characters: + * - / ( ) . : , =
(4) the "blank" character.

3.2. PUNCHING A PROGRAM

Each statement of a program is punched into a portion

of a paper tape with a maximum of 60 characters; however, if a statement is longer, it can be continued till 660 characters, dividing the portions with the quatern of characters # γ ν #, where γ and ν are respectively the carriage-return character and the space-line character.

The beginning of a statement is signed by # and the end by the pair of characters γ ν.

The beginning character # may be preceded by a statement number, greater than zero and less than 10^5, or by a letter (cf. sections 3.6, 3.9.4 and 3.12) followed or not by a statement number up to a maximum of six characters. The statements must be punched in the same order as they are in the program.

Blank characters may be used and they are ignored everywhere except in the alphanumeric fields.

3.3. QUANTITIES

A "quantity" is a constant, a variable, a subscripted variable, or a function.

Generally the mode of one or more quantities is specified by a "mode indicator". This may appear on the right or on the left of the beginning character #, and is composed of a single letter, respectively followed or not by the character ":".

Some mode indicators may be used without any special provision; they are those containing one of the following letters:

B, V, W, S, D, where

B specifies boolean quantities

V specifies floating point single precision quantities with an absolute value equal to zero or included between the approximate limits of 10^{-38} and 10^{+38}.

W specifies floating point double precision quantities with an absolute value equal to zero or included between the approximate limits of 10^{-4932} and 10^{+4932}.

S specifies fixed point single precision quantities with values included in the interval $-1 \vdash\!\!-\!\!- 1$.

D specifies fixed point double precision quantities with values included in the interval $-1 \vdash\!\!-\!\!- 1$.

Other mode indicators may be used for real quantities provided they are introduced by the MODE statement (cf. section 3.11.4) but they will be effective only if a suitable group of pseudo-instructions is built to perform the operations required in the indicated modes.

A mode indicator cannot be given more than one meaning in the main program and in its subprograms.

The scope of the mode indicators will be specified later.

3.3.1. CONSTANTS

Let δ be a decimal digit and ω be an octal digit. In a program one may use:

(1) the integer constants having the general form:

$$\pm\delta\delta \cdots \delta$$

where the sign is optional and the maximum number of digits is 5.

(2) the constants in the B mode having the general form:

$$\omega\ \omega \cdots \omega.$$

where the maximum number of digits is 12.

(3) the constants in the V or W mode having the general form:

$$\pm\,\delta\,\delta\cdots\delta.\delta\delta\cdots\delta E\pm\delta\,\delta\cdots\delta$$

where the sign is optional and parts of the general form may be omitted as in [1, 2].

The maximum number of significant digits, considered in the conversion from the decimal to the binary system, is 8 or 16 according to whether the mode of the constant is V or W.

(4) the constants in the S or D mode having the general form:

$\pm\,0.\delta\delta\cdots\delta$ if the value is in -1——1

or

$-1.00\cdots 0$ if the value is -1

In the former case the sign and the zero before the decimal point are optional and the maximum number of digits after the decimal point, in the decimal to binary conversion, is 10 or 20 depending on the constant being in S or D mode. In the latter case the zeros after the decimal point may be omitted.

(5) constants in other modes provided they have the decimal point.

The mode of a constant, not of type (1), is described by the rules in section 3.4.

3.3.2. VARIABLES

Variables are referred to by names, composed of letters and digits, and they may be integers or non-integers.

The general form of integer variables is the same as indicated in [1,2] for fixed point variables (restricted to integer values). An integer variable, that does not appear

in a subscript, may assume any integer value included in the interval -2^{35} ⊢——— 2^{35}.

The general form of non-integer variables is the same as indicated in [1,2] for floating point (single precision) variables. A non-integer variable may assume any value expressible as a constant in the variable mode.

The matrices may be integer or not and they are indicated as integer or non-integer variables respectively.

The indices of a matrix element are called "subscripts" and they may be at most three.

The general form and values of subscripts are as in [1,2].

Matrix elements are called "subscripted variables". They may be integers or non-integers and they are referred to respectively with integer or non-integer variable names (the same names of matrices to which they belong), each one followed by parentheses that enclose the subscript or the subscripts separated by commas.

The subscripted integer or non-integer variables may assume any value which may be assumed by integer or non-integer variables respectively.

From now on, a variable or a subscripted variable will be called a "generical variable".

When it is necessary, the mode of a generical non-integer variable is specified by the mode indicator or, if such indicator is missing, by the [1,2] rules (cf. yet section 3.4 for an exception).

A generical variable may be in only one mode in the same main program or subprogram. In particular, a subscripted variable must be in the mode of the matrix to which it belongs (cf. section 3.11.1).

Restrictions on variable names given in [1,2] also ap

ply to variables in any other mode.

3.3.3. FUNCTIONS

The functions are divided into Library, Arithmetic Statement and Fortran functions, and they are defined in section 3.5.

A function is "called", namely used, by writing its name in an arithmetic (cf. section 3.4.1) or boolean expression (cf. section 3.4.2); parentheses must follow the function name enclosing the actual argument or arguments of the function. If the actual arguments are more than one, they must be separated by commas.

Only one value is produced by every function by applying the calculation rules, specified in the definition of the function, to the given set of actual arguments.

Functions may be integers or non-integers.

Integer function names have general forms as indicated in [1,3] for fixed point function names (restricted to integral values).

Non-integer function names have general forms as indicated in [1,3] for floating point (single precision) function names.

An integer function may assume any integer value included in the interval -2^{35} —— 2^{35}.

A non-integer function may assume any value which may be expressed as a constant in the mode of the function.

In an expression, the mode of a non-integer function is explained by the rules of section 3.4.

3.3.3.1. ACTUAL ARGUMENTS AND MODES

Each actual argument of a Library or Arithmetic Statement function is an arithmetic expression of type 3.4.1

or boolean of type 3.4.2 according to whether the function is real or boolean.

Each actual argument of a Fortran function may be: an arithmetic expression of type 3.4.1 or a boolean of type 3.4,2, a matrix name, the name of Library function without the terminal F, the name of a Fortran function, the name of a SUBROUTINE subprogram (cf. section 3.9.2) or a Hollerith field.

The mode of an expression, when it is an actual argument, will be dealt with in section 3.4.

The mode of a matrix name, when it is an actual argument, is the same as in the DIMENSION statement (cf. section 3.11.1).

The mode of a Library function name, when it is an actual argument, is the same as in the definition of the function (cf. section 3.5.1).

The mode of a Fortran function, when it is an actual argument, is that of the same name in the body of the FUNCTION subprogram which defines the function (cf. section 3.9.1).

The mode of a name of a SUBROUTINE or of a Hollerith field, when they are arguments, does not make sense.

3.3.4. NOTES

3.3.4.1. Previously, the word "integer" has been used instead of "single precision integer".

Later it will be written with the same meaning "integer", "integer of single precision", or "in J mode".

3.3.4.2. The mode of an integer constant, generical variable, function, or matrix is implicit in its form or in that of its name as in [1]; therefore we agree that mode indica-

tors have effect only on quantities or matrices which are not in the J mode, and that, if they are generical variables, functions or matrices, they have names in floating point as in [1].

3.4. EXPRESSIONS

Expressions may be arithmetic or boolean and they are substantially made up of operands (constants, generical variables or functions) operators (arithmetic or logical) and parentheses according to the respective formation rules.

Since an expression may be an argument of a function contained in a longer expression, it is useful to distinguish an expression not contained in another from an expression which is argument of a function, referring to the first one as a "total expression" and to the second one as an "argument expression".

3.4.1. ARITHMETIC EXPRESSIONS

The arithmetic expressions are made up in the usual way.

Arithmetic operators are symbols + , - , * , / and * * which denote respectively addition, subtraction, multiplication, division and exponentiation.

Operands may be integers or not.

Integer operands may appear everywhere in an arithmetic expression (cf. section 3.4.1.3 for some exceptions).

Every dividend must lie on a line with its divisor.

Every base must lie on a line with its exponent.

Two operators cannot appear consecutively.

3.4.1.1. HIERARCHY OF OPERATORS

On the same parenthsis level, the hierarchy of operators is defined in the following way:

first: * *

second: * and /

third: + and −

If the hierarchy is not specified by these rules or by the use of parentheses, it is assumed from left to right.

For istance A * * B * * C means (A * * B) * * C.

3.4.1.2. OPERANDS AND MODES

The value of a total or argument arithmetic expression may be obtained in any real mode that will be called the "mode of the total or argument arithmetic expression".

This mode must be generally specified with the suitable mode indicator immediately preceding the total or argument arithmetic expression.

If $\mathcal{A}$ is a total or argument arithmetic expression immediately preceded by a mode indicator, the non-integer constants, generical variables and functions, contained in $\mathcal{A}$ but not in an argument expression inside $\mathcal{A}$, are thought in the mode specified by that indicator.

Mode indicators are not necessary before or inside a total arithmetic expression, when it is valid in [1] and it is intended as in [1], except for the extension to the value of integer operands.

Moreover, it is not necessary to indicate the mode of a total arithmetic expression $\mathcal{A}_1$, when the constants, generical variables and functions, contained in $\mathcal{A}_1$ but not in an argument expression inside $\mathcal{A}_1$, are intended as in [1], except for the extension to the value of integer operands.

Finally, it is not necessary to indicate the mode

of an argument arithmetic expression $\mathcal{A}_2$ when the non-integer constants, generical variables and functions, contained in $\mathcal{A}_2$ but not in an argument expression inside $\mathcal{A}_2$, are thought in the mode of the function of which $\mathcal{A}_2$ is an argument.

For istance, in the expressions:

```
L + 15
C(I,J) + A(I,K) * B(K,J)
13.25+L * * F-ACCAF(X,L+M,W:13.275476543889E40+RIGAF(Y(K)),Z)
D:G(I)+0.000256324753297699 * FUN(OMEGAF(E(I)))
```

the number 15 is assumed as an integer constant;

the variables L, I, J, K, M are assumed as integer quantities;

the number 13.25 is assumed as a constant in the V mode;

the generical variables C(I,J), A(I,K), B(K,J), F, X, Z are assumed as quantities in the V mode;

the function ACCAF(X,··· Z) is assumed as a quantity in the V mode;

the number 13.275476543889E40 is assumed as a constant in the W mode;

the subscripted variable Y(K) is assumed as a quantity in the W mode;

the function RIGAF (Y(K)) is assumed as a quantity in the W mode;

the number 0.000256324753297699 is assumed as a constant in the D mode;

the subscripted variables G(I), E(I) are assumed as quantities in the D mode;

the functions FUN(OMEGAF(E(I))), OMEGAF(E(I)) are assumed as quantities in the D mode.

3.4.1.3. OPERATIONS AND MODES

Every arithmetic operation is executed in such a manner that the result is in the J mode only if both the operands are in the J mode; otherwise the result of the operation is in the mode of the operand or operands not in the J mode.

The result of the division of an integer quantity by another is the integer part of the real quotient.

The addition and the subtraction between operands, one of which being in the S or D mode and the other one integer, are not allowed.

The division of a quantity in the J mode by a quantity in S or D mode is not allowed.

If an operation allowed is not mathematically defined for certain values of the operands, or if one of the Library functions listed in Table I (cf. section 3.5) is not mathematically defined for certain values of its argument, the computer stops signalling the reason why the operation or the function is not mathematically defined.

Thus, for example, if n is the value of the variable N in the J mode and g is the value of the variable G in the V mode, the operation G * * N will cause the computer to stop if $n \leq 0$ and $g = 0$.

3.4.2. BOOLEAN EXPRESSIONS

Boolean expressions are made up with rules similar to those of the arithmetic expressions.
More precisely:

(1) the following may be interpreted as boolean expressions:

(a) a positive octal constant with no more than twelve digits;

(b) a generical variable with name indicating a non-integer mode;

(c) a function with name indicating a non-integer mode.

(2) If $\mathcal{B}_1$ and $\mathcal{B}_2$ are interpretable as boolean expressions and if the first character of $\mathcal{B}_2$ is not the symbol -, then the following are also interpretable as boolean expressions:

(d) $-\mathcal{B}_2$

(e) $\mathcal{B}_1 * \mathcal{B}_2$

(f) $\mathcal{B}_1 + \mathcal{B}_2$

(g) $(\mathcal{B}_1)$

(3) A total or argument expression, interpretable as boolean and immediately preceded by the mode indicator "B:", is treated as a boolean expression in which symbols as -, * and + are respectively the logical operators "not", "and", "or".

3.4.2.1. OPERANDS AND MODES

If $\mathcal{B}$ is a total or argument boolean expression, the constants, generical variables and functions, contained in $\mathcal{B}$ but not contained in an argument expression inside $\mathcal{B}$, are thought as boolean.

If an expression, interpretable as boolean and not immediately preceded by the mode indicator "B:", is argument of a boolean function, it is also treated as boolean.

In an arithmetic statement (cf. section 3.6), the generical variable on the left side and the total expression on the right side are treated as boolean if they are both in

terpretable as such and the letter B precedes the beginning character # of the statement.

3.4.2.2. HIERARCHY OF OPERATORS

On the same parenthesis level, the hierarchy of logical operators is defined in the following way:

first: .-

second: *

third: .+

If the hierarchy is not determined by these rules or parentheses, it is assumed from left to right.

3.4.2.3. OPERATIONS

All logical operations are performed upon the full 36-bit words [2].

3.5 DEFINITION OF FUNCTIONS

3.5.1. LIBRARY FUNCTIONS

These functions are predefined by means of closed subroutines which may be on library paper tapes or magnetic tape.

Twenty-six library functions are already available.

They are listed in Table I where x denotes the value of the sole dummy argument.

Other functions may be added to the library.

3.5.2. ARITHMETIC STATEMENT FUNCTIONS

These functions are defined by a single statement

TABLE I

Function name	Description of the function	Mode of	
		funct.	argum.
SQRTF	Square root of x	V	V
CRTF	Cubic root of x	V	V
LOGF	Logarithm, on base e, of x	V	V
LOGXF	Logarithm, on base 10, of x	V	V
EXPF	Exponential, on base e, of x	V	V
EXPXF	Exponential, on base 10, of x	V	V
SINF	Sine of x (x in radians)	V	V
COSF	Cosine of x (x in radians)	V	V
TANF	Tangent of x (x in radians)	V	V
CTANF	Cotangent of x (x in radians)	V	V
ASINF	Arc in radians, included in the interval $-\pi/2 \vdash\dashv \pi/2$, which has x as sine	V	V
ACOSF	Arc in radians, included in the interval $0 \vdash\dashv \pi$, which has x as cosine	V	V
ATANF	Arc in radians, included in the interval $-\pi/2 \vdash\dashv \pi/2$, which has x as tangent	V	V
SINHF	Hyperbolic sine of x	V	V
COSHF	Hyperbolic cosine of x	V	V
TANHF	Hyperbolic tangent of x	V	V
ASINHF	Argument of the hyperbolic sine of x	V	V
ACOSHF	Argument of the hyperbolic cosine of x	V	V
ATANHF	Argument of the hyperbolic tangent of x	V	V
ABSF	Absolute value of x	V	V
EXPSF	Exponential, on base e, of x	S	S
LOGSF	Logarithm, on base e, of x	S	S
EXPDF	Exponential, on base e, of x	D	D
LOGDF	Logarithm, on base e, of x	D	D
EXPWF	Exponential, on base e, of x	W	W
LOGWF	Logarithm, on base e, of x	W	W

which has the general form:

$$\mathcal{F} = \mathcal{E}$$

where $\mathcal{F}$ is the name of the function followed by parentheses, which enclose the dummy argument or arguments of the same function, and $\mathcal{E}$ is an arithmetic expression of the type 3.4.1. or boolean of the type 3.4.2 which involves the same arguments.

Each argument of $\mathcal{F}$ must be a non-subscripted variable. If the arguments of $\mathcal{F}$ are more than one they must be separated by commas. The argument names of $\mathcal{F}$ may be used to indicate other variables in another part of the program. The expression $\mathcal{E}$ must not contain subscripted variables, but may use variables, library functions, and functions defined by a preceding statement of the above said form, freely.

The function $\mathcal{F}$ assumes the mode of the expression $\mathcal{E}$.

The mode of each dummy argument is such as pertains to it, in $\mathcal{E}$, by virtue of the rules on the mode of the operands of an expression (cf. section 3.4).

A function defined in the above said form may be called only in the particular main program or subprogram where its definition appears which must precede the first executable statement of that program or subprogram.

Examples:

```
ALFAF(X) = B * TANF(A+X)+LOGF(X)
XBETAF(L) = N+M * * L
GAMMAF(X) = D: .27847570190872322567+25 * X + LOGDF(X)
DELTAF(X,Y) = W: 4517 * D - 230.556897465999E+54/X+EXPWF(Y)
ETAF(X,Y) = W: -.20175437421 35E-40+X/J+K/Y-X * * Y
TETAF(F,G,I) = W:ETAF(F,DELTAF(F,G))/XBETAF(I)
EQUIVAF(P,Q) = B:P * Q + (-P) * (-Q)
```

3.5.3. FORTRAN FUNCTIONS

See page 16 of [1], replacing the word FORTRAN with the words FORTRAN CEP, and compare [2,3] with section 3.9.

3.5.4. NOTES

3.5.4.1. Every function must be used in the mode it has in its definition.

3.5.4.2. Dummy arguments are replaced, at execution time, by the actual arguments taken in the same order; therefore, there must be agreement in number, order and mode between the two sets of arguments.

3.6. ARITHMETIC STATEMENT

An arithmetic statement has, excluding the mode indicators, the general form:

$$\mathcal{V} = \mathcal{E}$$

where $\mathcal{V}$ is a generical variable, $\mathcal{E}$ is an expression and the sign = does not indicate equality between the two sides but an assignment of the value of $\mathcal{E}$ to $\mathcal{V}$.

If the letter B precedes, as said in 3.2, the beginning character # of the arithmetic statement, $\mathcal{V}$ and $\mathcal{E}$ (supposed interpretable as boolean) are treated as boolean; otherwise:

(a) $\mathcal{V}$ is a generical real variable and may be preceded immediately by a mode indicator to specify its mode;

(b) $\mathcal{E}$ is an arithmetic expression of the type 3.4.1 with a mode equal to, or different from, that of $\mathcal{V}$ but such that neither of the members is in the J mode if the other is in the S or in the D mode.

If the mode of $\mathcal{E}$ is different from that of $\mathcal{V}$ and the value of $\mathcal{E}$ does not exceed the limits for $\mathcal{V}$, this value is expressed in the mode of $\mathcal{V}$ and then assigned to $\mathcal{V}$. In this case, if $\mathcal{V}$ is in J mode and $\mathcal{E}$ is in V or W mode, to $\mathcal{V}$ it is assigned the value of the integer part of the result of $\mathcal{E}$.

If the mode of $\mathcal{E}$ is different from that of $\mathcal{V}$ but the result of $\mathcal{E}$ exceeds the limits for $\mathcal{V}$, to $\mathcal{V}$ it is assigned the best approximate value for $\mathcal{E}$ between the limits for $\mathcal{V}$, expressed in the mode of $\mathcal{V}$.

If $\mathcal{V}$ and $\mathcal{E}$ have the same mode specified immediately before each of them by the suitable mode indicator, it may be substituted, on the left side as on the right side, by the letter denoting the common mode, placed, as said in 3.2, before the beginning character # of the arithmetic statement. Examples:

```
L = L+15
C(I,J) = C(I,J) + A(I,K) * B(K,J)
A = 13.25+L* *F+ACCAF(X,L+M,W:13.275476543889E40+RIGAF(Y(K),Z)
S: AH53(I+3)=D:G(I) + 0.0002563247532976 99 * FUN(OMEGAF(E(I)))
W 12#D(K,J) = LOGWF(R) - A1(J)/A2(K)+L* *M * R+M-L
B # U = EQUIVAF (S,T)
B  30 # V * 377
B  15 # FIMP(I) = - P(I)+Q(I)
```

3.7. CONTROL STATEMENTS [2,2]

3.7.1. UNCONDITIONAL GO TO, COMPUTED GO TO, ASSIGNED GO TO and ASSIGN

There are no restrictions on Assigned GO TO in the range of a DO.

3.7.2. IF

The expression may be arithmetic of type 3.4.1 or boolean of type 3.4.2.

Examples:

```
IF(D:A(I,J)-A(J,I))12, 15, 24
IF(B:-P+Q) 5, 7, 7
```

3.7.3. SENSE LIGHT, IF(SENSE LIGHT), IF(SENSE SWITCH), IF ACCUMULATOR OVERFLOW, IF QUOTIENT OVERFLOW and IF DIVIDE CHECK

There are no changes.

3.7.4. DO

No IF-type or GO TO-type transfer is allowed into the range of a DO from outside its range.

Not the first one, but the last statement in the range of a DO must not be one of the following statements: FORMAT, DIMENSION, EQUIVALENCE, COMMON, FREQUENCY and MODE.

3.7.5. CONTINUE, PAUSE and STOP

There are no changes.

3.8. END

It assumes the simple form END and it is used only to signal to the FORTRAN CEP translator that the end of the program has been reached.

The general form indicated in [2,2] is accepted, but the reference to the sense switches is ignored.

END must always be the last statement of a program and must never be omitted.

3.9. SUBPROGRAM STATEMENTS [2,3]

A subprogram may call other subprograms except itself or one that refers to it.

A FUNCTION subprogram may be referred to by an arithmetic expression of type 3.4.1 or boolean of type 3.4.2, which contains the name of the Fortran function; a SUBROUTINE subprogram may be referred to only by a CALL statement (cf. section 3.9.3).

Hand-coded subprogram are briefly treated in section 4.

3.9.1. FUNCTION

In a FUNCTION statement, each dummy argument may be a name of a non-subscripted variable, the name of a SUBROUTINE subprogram, the name of a Fortran function, or the name, without the terminal F, of a Library function.

The mode of the function referred to is that belonging to the name of it in the body of the FUNCTION subprogram.

The mode of a dummy argument is that belonging to it in the body of the FUNCTION subprogram.

A dummy argument, which is the name of a matrix, must appear in a DIMENSION statement with the same dimensions and the same mode of the name of the matrix that corresponds to it as actual argument.

Example:

The subprogram

```
  # FUNCTION ZETA (X,Y) ;;
    .
    .
W # ZETA=((X+G)/(X-G))**Y ;;
    .
    .
  # END ;;
```

could be referred to by the arithmetic statement

```
  # F=W: ZETA(A+C, B*S-T)  +K*Z/N γν
```

3.9.2. SUBROUTINE

In a SUBROUTINE statement, each dummy argument may be as in 3.9.1.

The mode of a dummy argument is that belonging to it in the body of the SUBROUTINE subprogram. A dummy argument, which is the name of a matrix, must appear in a DIMENSION statement as in 3.9.1.

Example:

The subprogram

```
      # SUBROUTINE SPUR (A, N, R) γν
      .
      .
D     # R=0. γν
      # DO 10 I=1, Nγν
D10   # R=R+A(I, I)γν
      .
      .
      # END γν
```

could be referred to by the statement given as second example in 3.9.3, if C appears in a DIMENSION statement as a matrix in the D mode.

3.9.3. CALL

The actual arguments are set as those of the FORTRAN functions (cf. section 3.3.3.1).

Examples:

```
CALL RIS (SIN, X, A, B, 6HRESULT)
CALL SPUR (C, 50, D: T)
CALL QUIBUS (B: G*E+F, W: 253.256764357856, LOGW, S: A+B/J, X1)
```

3.9.4. SUBROUTINE NAMES AS ARGUMENTS

A list of names of Fortran functions, of Library functions without the terminal F and of SUBROUTINE subprograms, which are actual arguments of FUNCTION or SUBROUTINE subprogram, must be punched, as said in 3.2, with the letter F before the beginning character #. Such a list may be punched between any two statements of a program containing references to the FUNCTION and SUBROUTINE subprograms.

Example:

F # SIN, COS, MAP

3.9.5. RETURN

There are no changes.

3.10. INPUT/OUTPUT STATEMENTS

Input and output statements are used for the transmission of information, at execution time, between magnetic core memory on one side and magnetic tapes, magnetic drum, paper tape readers, high speed punches, on-line typewriter and on-line printer on the other side.

3.10.1. LIST OF QUANTITIES

Some of the input/output statements contain a list of quantities to be transmitted. This list may be, in each of the input/output statements to which it is associated, as described in pages 37 and 38 of [1].

In the list several mode indicators may appear, each of which may be introduced before any list element and also before any variable which is not part of a subscript or of an indexing information.

A mode indicator specifies the mode of the generical

non-integer variables, that follow it, until another mode indicator occurs. The variables of the list not preceded by a mode indicator have the mode as said in [1].

Extra-parentheses are not required.

To transmit an entire matrix it is only necessary to mention the name of the matrix in the list.

A matrix, which is simply listed by its name, is intended in the mode it would have if it is to be considered as a variable of the list. This mode must be equal to that of the matrix in its DIMENSION statement.

The elements of a matrix are stored in order of increasing storage locations.

For instance, in the list:

T, K, B(K), D:((CAM(I,2*J), I=1,3), J=1,5),(((A(I,J,L), J=1,7),I=1,5),L=1,K)
W:(AH55(I,5),I=1,4),V:RMS,C,(D:E(I),S:F(I,9),I=1,100,2),G

K, I, J, L are integer variables;
T, B(K), RMS, C are generical variables in the V mode;
CAM(I,2 * J), A(I,J,L), E(I) are subscripted variables in the D mode;
AH55 (1,5) is a subscripted variable in the W mode;
F(I,9) is a subscripted variable in the S mode;
G is the matrix in the S mode with dimensions defined in 3.11.1.

3.10.2. FORMAT [2,4]

3.10.2.1. UNIT RECORD

A unit record may be:

(1) a printed line with a maximum of 68 or 102 characters depending on whether the output is performed by means of typewriter or printer;

(2) a portion of punched paper tape ending with the pair of characters γν and composed by 68 characters at the most;

(3) a record of a BCD magnetic tape with a maximum of 132 characters.

3.10.2.2. NUMERICAL FIELDS

The conversions already available are listed in Table II. Each conversion depends on the mode of the quantity that is to be transmitted and on the corresponding numerical format specification; yet the format specification of type O over-rides any other mode.

Numerical format specifications may be in the forms indicated in [2,4] and they describe the external form of the numerical information as in [2.4].

The number of position to the right of the decimal point, in a numerical field controlled by an E- or F- type specification, is not treated modulo.

3.10.2.3. ALPHANUMERICAL FIELDS

The format specification of type A over-rides any other mode.

3.10.2.4. BLANK FIELDS

The number of blank characters provided in an output record or characters skipped in an input record may not exceed the record maximum length.

3.10.2.5. REPETITION OF FIELD FORMAT. REPETITION OF GROUPS.

A FORMAT statement may include up to three levels of parentheses. For example, the statement:

TABLE II

Inner numerical information	Character of mode	Character of control	Outer numerical information
Binary integer	J	I	Decimal integer in the interval $-2^{35} \longmapsto 2^{35}$
(1) Binary integer	J S D V W	O	Octal integer in the interval $-400000000000 \longmapsto +777777777777$
Fixed-point single precision binary number	S	F	Decimal fixed-point number included in the interval $-1 \longmapsto 1$. The maximum number of decimal digits (considered in the decimal to binary system conversion) after the point, is 10.
Fixed-point double precision binary number	D	F	Decimal fixed-point number included in the interval $-1 \longmapsto 1$. The maximum number of decimal digits (considered in the decimal to binary system conversion) after the point, is 20.
Floating-point single precision binary number	V	F	Decimal number, without exponent, equal to zero or with absolute value included between the approximate limits of 10^{-38} and 10^{+38} (2). The maximum number of significant digits, considered in the decimal to binary system conversion, is 8.
Floating-point double precision binary number	W	F	Decimal number, without exponent, equal to zero or with absolute value included between the approximate limits of 10^{-4932} and 10^{+4932} (2). The maximum number of significant digits considered in the decimal to binary system conversion, is 16.
Floating-point single precision binary number	V	E	Decimal number with exponent on line with the mantissa, separated from it by the letter E. The magnitude of such a number must be zero or included between the approximate limits of 10^{-38} and 10^{+38}. The maximum number of significant digits, considered in the decimal to binary system conversion, is 8.
Floating-point double precision binary number	W	E	Decimal number with exponent on line with the mantissa, separated from it by the letter E. The magnitude of such a number must be zero or included between the approximate limits of 10^{-4932} and 10^{+4932}. The maximum number of significant digits, considered in the decimal to binary system conversion, is 16.

(1) Octal numbers in the interval $400000000000 \longmapsto 777777777777$ are converted in the binary system as the opposites of the complement of 8.

(2) It is obvious that those limits meet uniformity requirements and not practical requirements when the decimal number is without exponent.

FORMAT (2(3(2(F14.7,E16.6),I3/),3E20.8//))
is allowed.

3.10.2.6. SCALE FACTORS. MULTIPLE - RECORD FORMATS. FORMAT AND INPUT/OUTPUT STATEMENT LISTS. ENDING A FORMAT STATEMENT. FORMAT STATEMENTS READ AT EXECUTION TIME. CARRIAGE CONTROL OF THE PRINTER OFF OR ON LINE.

There are no changes.

3.10.2.7. DATA INPUT AT EXECUTION TIME

The blank characters in numerical fields are ignored.

The numbers, controlled by the numerical specification of type I, are not treated modulo.

Relaxations, in input data format, are permitted in any precision. Numbers affected by writing errors or by magnitude errors will not be converted and the computer will stop signalling the error type.

3.10.3. READ

The READ statement has the general form:
READ $\mathcal{U}$/$\mathcal{R}$,$\mathcal{L}$
where:
(a) $\mathcal{U}$ is a positive integer constant or an integer variable which specifies the number of the paper tape reader to be used;
(b) $\mathcal{R}$ is the number 0 or a reference to a FORMAT statement, made by means of a statement number or a matrix name;
(c) $\mathcal{L}$ is the list of quantities as is described in 3.10.1.
$\mathcal{U}$ may assume the values 1, 2 or 3.

The general form indicated in [2,4] is allowed and it is equivalent to a READ statement for the paper tape reader No1.

The READ statement causes the reading and the conversion (if it is necessary) of data punched on paper tape, up to the end of the list.

If $\mathcal{R}$ is the number 0, reading is not done under the control of a FORMAT statement; otherwise reading is controlled by the FORMAT statement referred to. In the first case, to the variables of the list may correspond only numerical data of type indicated in table II and ended by blank characters and/or by the pair of characters γ ν.

Examples:

```
READ 12, (FRM (K), K = 1,8)
READ I/O ,T, K,B(K), D:((CAM(I,2 * J), I=1,3), J=1,5),
                      (((A(I,J,L), J=1,7), I=1,5),L=1,K)
READ LF/FRM, W: (AH55(I,5),I=1,4); V: RMS,C,(D:E(I),S:F(I,9),
                                          I=1,100,2),G
```

3.10.4. READ INPUT TAPE, READ TAPE, READ DRUM, PRINT, WRITE OUTPUT TAPE, WRITE TAPE AND WRITE DRUM [2,4]

The reference to a FORMAT statement, if any, may be done through a name of a matrix as well as by means of a statement number.

The list is that indicated in 3.10.1.

The magnetic tape units are numbered from 1 to 8.

There are not 8 magnetic drums but there are 8 regions of the same magnetic drum.

3.10.5. PUNCH

The PUNCH statement has the general form:

PUNCH $\mathcal{U}$ / $\mathcal{R}$, $\mathcal{L}$

where:

(a) $\mathcal{U}$ is a positive integer constant or an integer variable that specifies the number of the high speed punch to be used;

(b) $\mathcal{R}$ is a reference to a FORMAT statement, made by means of a statement number or a matrix name;

(c) $\mathcal{L}$ is a list of quantities as said in 3.10.1.

$\mathcal{U}$ may assume the values 1,2 or 3.

The general form indicated in [2,4] is allowed and it is equivalent to a PUNCH statement for the high speed punch No. 1.

The PUNCH statement causes the conversion (if it is necessary) and the punching of the list quantities on a paper tape.

Successive records are punched in agreement with the FORMAT statement referred to, until the list is exhausted.

Examples:

```
PUNCH 10, (((A(I,J,K),B(I,J,K), K=1,5), C(I,J), J=1,3),
                                              D(I), I=M,40)
PUNCH 3/7, ((W:A(I,J), D:B(I+2,3 * J), I=1,15,2),J=1,4)
PUNCH K/12, I, S : (C(I,J-2), J=4,7), D(I)
```

3.10.6. TYPE

TYPE is a statement which has the general form:

TYPE $\mathcal{R}$, $\mathcal{L}$

where $\mathcal{R}$ and $\mathcal{L}$ are as in 3.10.5.

The TYPE statement causes the conversion (if it is necessary) and the printing of the quantities of the list on the on-line typewriter.

Successive lines are printed accordingly with the referred FORMAT statement until the list is exhausted.

Example:
TYPE 12, A, B, J, D: C(J), D(J), G(J)

3.10.7. END FILE, REWIND AND BACKSPACE

Magnetic tape units are numbered from 1 to 8.

3.11 SPECIFICATION STATEMENTS [2,5]

3.11.1 DIMENSION

In the list of matrix names, each subscripted with one, two or three positive integer constants which denote in the given order the number of rows, columns and layers of the matrix, may appear several mode indicators, each of which may be introduced before any matrix name.

A mode indicator specifies the mode of non-integer matrices, referred to with names following it, until another mode indicator occurs.

Matrices, the names of which are not preceded by a mode indicator, have the mode as in [1].

The elements of a matrix are all in the mode of the matrix.

For example, in the following statements:
DIMENSION W: A(5), B(4,20), D: C(3,17,6), INT(12)
DIMENSION JOB(7), F(14,8), S: G(20,4,50), H(11,5), W: TAB(50,50)
the elements of the matrices referred to with INT, JOB are all integers;
the elements of the matrix referred to with F are all in the V mode;
the elements of the matrices referred to with A, B, TAB are all in the W mode;
the elements of the matrices referred to with G, H are all in

the S mode;
the elements of the matrix referred to with C are all in the D mode.

3.11.2. EQUIVALENCE AND COMMON.

In an EQUIVALENCE statement the subscripted variable C(p), with p > 0, means the (p-1)th location after the higher-order word of the variable C or the first element of the matrix C.

If p is not specified it is taken to be 1.

The COMMON area is assigned beginning at location 8000 and continuing downward in the main memory.

The elements of each matrix are stored in order of increasing storage locations.

Dummy variable names, placed in a COMMON statement to force correspondence in main memory locations between two variables which otherwise will occupy different relative positions, must also appear in a DIMENSION statement.
For example, the statements:

```
DIMENSION W: E(3)
COMMON A, B, C, D
EQUIVALENCE (C, E(2))
```

will cause storage to be assigned in the following way:

```
D
B
A
                E^I (1)
C               E^II(1)
                E^I (2)
                E^II(2)
                E^I (3)
                E^II(3)
```

where $E^{I}(i)$ and $E^{II}(i)$ represent the first and second part respectively of the element E(i) in double precision and the locations increase, one by one, from D to $E^{II}(3)$.

3.11.3. FREQUENCY

May be used, but it is simply ignored.

3.11.4. MODE

MODE is a statement which has the general form:

$$\text{MODE } \mathcal{M}_1, \mathcal{M}_2, \ldots \mathcal{M}_n$$

where each $\mathcal{M}_i$ is one of the letters G, K, L, M, N, P, Q, R, T, U, Y followed by a positive integer constant between parentheses.

Every letter indicates the real mode of which the integer constant, following it, makes explicit the number of words a quantity occupies under that mode.

The MODE statement introduces real modes different from that indicated in section 3.3 and must precede the other statements in which they appear.

For example, if the letter K is assumed to specify integer quantities in double precision, the statement which introduces the mode K is:

MODE K(2)

3.12. COMMENT

A comment must be punched, as said in 3.2, with the letter C before the beginning character #.

4. HAND-CODED SUBPROGRAMS

Subprograms written in symbolic language LPSC (Linguaggio Programmativo Simbolico CEP [3]) may be called by a FORTRAN CEP program as FORTRAN CEP subprograms of type FUNCTION or SUBROUTINE provided that the coding is done in the suitable way.

REFERENCES

[1] IBM Reference Manual 704 FORTRAN Programming System, Form C 28 - 6106.

[2] Manuale delle istruzioni CEP - Centro Studi Calcolatrici Elettroniche, Pisa (Italia) 1960.

[3] La programmazione simbolica per la CEP - Centro Studi Calcolatrici Elettroniche, Pisa (Italia), 1961.