Addenda to the FORTRAN Programmer's Reference Manual

I. Title page. For "late '956" read "early 1957".

II. Page 12. The first paragraph of the section of Functions should read as follows.

> As in the above example, ... master FORTRAN tape, or has been defined in a function statement (see below).

Page 14, line 3. Add paragraph.

> Function Statements. A function may also be defined in the source program itself by means of a function statement, and the definition will then persist throughout that one program. Thus it is possible to have the convenience of the function notation even for functions which are not important enough to deserve a place on the master FORTRAN tape.

General Form	Examples
"a=b" where a is a function name followed by parentheses enclosing its arguments (which must be distinct non-subscripted variables) separated by commas, and b is an expression (see next section) which does not involve subscripted variables. Any functions appearing in b must be built-in, or available on the master tape, or already defined by preceding function statements.	FIRSTF(X) = A*X+B SECONDF(X, B) = A*X+B THIRDF(D) = FIRSTF(E)/D FOURTHF(F, G) = SECONDF(F, THIRDF(G)) FIFTHF(I, A) = 3.0*A**I SIXTHF(J) = J+K XSIXTHF(J) = J+K

> Just as with ordinary functions, the answer will be expressed in fixed or floating point according as the name does or does not begin with X.

The right-hand side of a function statement may be any expression not involving subscripted variables and meeting the requirements stated in the next section. In particular, it may involve functions freely, provided that any such function, if it is not built-in or available on the master tape, has been defined in a preceding function statement.

As many as desired of the variables appearing in the expression on the right-hand side may be stated on the left-hand side to be the arguments of the function. Since the arguments are really only dummy variables, their names are unimportant (except as indicating fixed or floating point mode) and may even be the same as names appearing elsewhere in the program.

Those variables on the right-hand side which are not stated as arguments are treated as parameters. Thus, if FIRSTF is defined in a function statement as

```
FIRSTF(X) = A*X+B
```

then a later reference to FIRSTF(Y) will cause ay+b, based on the current values of a, b, and y, to be computed. The naming of parameters, therefore, must follow the normal rules of uniqueness.

A function defined by a function statement may be used just as any other function. In particular, its arguments may be expressions and may involve subscripted variables; thus a reference to FIRSTF(Z+Y(I)) will cause $a(z+y_i)+b$, based on the current values of a, b, y_i, and z, to be computed.

Functions defined by function statements are always compiled as closed subroutines.

NOTE. All the function statements in a program must precede the first executable statement of the program.

III. Page 13. Add to the table of built-in functions.

Type of Function	Definition	No. of Args.	Name	Mode of Argument	Mode of Function
Float	Float fixed number	1	FLOATF	Fixed	Floating
Fix	Same as XINTF	1	XFIXF	Floating	Fixed
Transfer of sign	Sign of Arg_2 times $\lvert Arg_1 \rvert$	2 2	SIGNF XSIGNF	Floating Fixed	Floating Fixed

IV. Page 16, middle of page. Add paragraph

If the variable on the left is floating point and the expression on the right is fixed point, the result will be computed in fixed point, truncated to an integer, and then converted to floating point.

V. Page 21, 3rd line from bottom. Add

(By "same part of the nest" is meant that no DO, and no statement which is a last statement in the range of a DO, shall lie between the exit point and re-entry point.)

VI. Page 22. Replace first paragraph by

Restriction on Assigned GO TO's in the Range of a DO. When an assigned GO TO exists in the range of a DO, all of the statements to which it may transfer which lie in the nest must all be in the exclusive range of a single DO. The exclusive range of a DO are those statements in its range which are not in the range of any DO in its range.

VII. Page 22. Under "Restriction on Calculations in the Range of a DO", add paragraph

> The first statement in the range of a DO must not be a non-executable statement.

VIII. Page 23. At bottom of page add

> WARNING. The last executable statement in the source program must be either a STOP (not a PAUSE) or an IF-type or GO TO-type transfer.

IX. Page 25. Throughout last paragraph replace the phrase "natural order" by "inverse of natural order, i.e., in the order $A_{m,n}, \ldots, A_{m,2}, \ldots, A_{2,2}, A_{1,2}, A_{m,1}, \ldots, A_{2,1}, A_{1,1}$, etc."

X. Page 29, line 3. Add paragraph

> Ending a Format Statement. If a FORMAT statement does not contain any numeric field, any input-output statement which uses it will never come to an end. A decimal input-output operation will be brought to an end when and only when a numeric field is encountered in the FORMAT statement, and there are no items remaining in the list of the input-output statement itself.

XI. Page 37, last sentence. For "8 types" read "7 types".

Page 38. Delete PAUSE from table.

XII. Page 40. Add section.

EXPONENTIATION The appearance of an exponential E**F in an expression will produce one of 5 object situations.

1 and 2. If F is a fixed point constant from +1 to +7 inclusive, then one of two open subroutines will be compiled, depending upon whether E is fixed or floating. One or the other of these subroutines will be compiled for each such exponentiation which occurs. These subroutines perform exponentiation by repeated multiplication; they consist of F-1 multiplications and a maximum of F+2 other instructions. The exponential is computed for any E.

3 and 4. If F is a fixed point constant not in the range +1 to +7 inclusive, or a fixed point variable, then one of two closed subroutines will be compiled, depending upon whether E is fixed or floating. These subroutines will not appear more than once in the object program. They perform exponentiation by forming E^F as the appropriate product of E, E^2, E^4, E^8, ..., followed (if $F < 0$) by division into 1.

The subroutine for fixed E has 34 instructions and takes an average of 10 milliseconds. For $F \leq 0$, $E = 0$ and for $F < 0$, $|E| > 1$, it returns to 0; for all other cases it returns E^F.

The subroutine for floating E has 38 instructions and takes an average of 9 milliseconds. For $F \leq 0$, $E = 0$, it returns to 0; for all other cases it returns E^F.

5. If F, and therefore E, is floating, a closed subroutine is compiled. It will not appear more than once in the object program. It produces E^F

as $e^{F\ln E}$; it has 100 instructions and takes approximately 6 milliseconds. For $F \leq 0$, $E = 0$ it returns 0; for all other cases it returns $/E/^F$.

XIII. Page 42, line 2. Add paragraph

What has just been said applies only when I is referred to as a variable. When it is referred to as a subscript, I is undefined after any normal exit and is the current value after any transfer exit.

Addenda to the FORTRAN Programmer's Reference Manual

I. Page 4 of attached. "X. Page 29, line 3. Add paragraph" should read

Ending a Format Statement. A decimal input-output operation will be brought to an end when and only when a numeric field or the end of the FORMAT statement, is encountered and there are no items remaining in the list of the input-output statement itself.