GC28-6884-0

Program Product

IBM FORTRAN Program Products for OS and the CMS Component of VM/370 General Information

Program Nos: 5734-FO1
5734-FO2
5734-FO3
5734-LM1
5734-LM3
5734-CP3
5734-FO5

This publication provides general information about the functions, capabilities, and system requirements of the following program products:

- Code and Go FORTRAN Processor
- FORTRAN IV (G1) Processor
- FORTRAN IV (H Extended) Processor
- FORTRAN IV Library (Mod I)
- FORTRAN IV Library (Mod II)
- TSO FORTRAN Prompter
- FORTRAN Interactive Debug

These products, with the exception of the TSO Prompter, operate under both OS and the CMS component of VM/370.

This publication is intended as an aid to evaluation and planning and is not meant for the terminal user or applications programmer.

IBM

PREFACE

This publication, directed to data processing system planners and analysts, is intended as an aid in evaluating and planning for the use of large-system FORTRAN program products available for OS and VM/370-CMS. Included are discussions of the Code and Go FORTRAN, FORTRAN IV (G1), and FORTRAN IV (H Extended) processors, the FORTRAN IV (Mod I) and FORTRAN IV (Mod II) libraries, and the TSO FORTRAN Prompter and FORTRAN Interactive Debug.

To assist the reader in using this publication, its organization is outlined below:

- *Introduction:* This section describes, in broad terms, the interrelationships of the program products available to the user, and provides comparative information about them. It informs the reader of the function of each product and of the applications each is best suited for.

- *Language Summary:* This section summarizes the IBM FORTRAN IV language as implemented by the program product processors. Intended primarily to introduce the language to users unfamiliar with IBM FORTRAN IV, the section contains tables of statement usage, math functions, and language features that represent IBM extensions to ANS FORTRAN.

- *Program Product Processors:* Using the Introduction as background material, this section provides a more detailed description of the capabilities of the Code and Go, FORTRAN IV (G1), and H Extended processors.

- *Libraries:* This section, closely related to the section on processors, concentrates on the facilities made available to the user via the Mod I and Mod II libraries.

- *Application Development Support Products:* This section contains information on the TSO FORTRAN Prompter and on FORTRAN Interactive Debug. A table summarizing the subcommands that can be used with FORTRAN Interactive Debug is included.

- *Reference Material:* This section describes the publications currently available to support the installation and use of the FORTRAN program products.

FIRST EDITION (July 1972)

Changes are periodically made to the information herein; any such changes will be reported in subsequent revisions.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

Address comments concerning the contents of this publication to IBM Corporation, Programming Publications, 1271 Avenue of the Americas, New York, New York 10020.

CONTENTS

TABLES

IBM offers a wide range of FORTRAN program products for OS and the CMS component of VM/370 that provide comprehensive support to meet the needs of all FORTRAN users, from the most experienced system analyst to engineers, mathematicians, and others who may not be full-time, professional programmers. These program products are:

- Three language processors designed to meet the varying needs of compilation speed and execution performance: the Code and Go FORTRAN processor; the FORTRAN IV (G1) processor; and the FORTRAN IV (H Extended) processor.

- Two libraries: FORTRAN IV (Mod I) and FORTRAN IV (Mod II).

- Two application development support facilities created to simplify the programmer's activities in a time-sharing environment: The TSO FORTRAN Prompter for the G1 processor and FORTRAN Interactive Debug for the Code and Go and G1 processors.

FORTRAN Interactive Debug, the most recent addition to the FORTRAN support, gives the terminal user powerful new tools for program checkout. Through the use of simple commands, he can dynamically monitor and control the execution of his program in terms that are meaningful to him (i.e., the symbols, labels, and line numbers of his source program). Because the user is in an interactive mode, and can decide his next action based on the results of preceding ones, he can quickly isolate the area of his program that is giving trouble. Because he can also adjust the value of variables during execution, he can very often check out his "fix" at the same time. He then returns to the edit mode of his system (TSO or CMS) to make permanent corrections to his program.

See the section "FORTRAN Interactive Debug" for a discussion of the scope of debugging capabilities available with this product.

As for the processors themselves, all three can be invoked under OS -- either as background (batch) processors or as foreground processors under TSO -- and under CMS.[1] The FORTRAN language supported by the processors encompasses and is compatible with the American National Standard FORTRAN language. The processors also provide support for IBM extensions to the language.

Code and Go FORTRAN, as a time-sharing tool, has been designed to meet the specific needs of two types of users: (1) the *problem solving* programmer, who writes, debugs, and executes relatively short programs at the terminal, and (2) the *production* programmer who debugs components of a large program on-line before running the program through a production-oriented processor, such as FORTRAN IV (H Extended). Thus, design emphasis has been placed on rapid compilation-execution turnaround and on ease of use. Code and Go supports free-form input format -- which considerably

[1] Throughout this publication, the term foreground is used to refer both to the TSO time-sharing foreground and to real-time jobs submitted under CMS; the term background is used to refer both to the TSO background and to batch jobs submitted under CMS.

reduces the programmer's concern with terminal-typing tasks, such as tab settings and margin stops -- and includes options for obtaining short- or long-form diagnostic messages. Support is also provided for FORTRAN Interactive Debug and for the use of list-directed input/output, which frees the programmer from having to code FORMAT statements.

FORTRAN IV (G1), an extended version of FORTRAN IV (G), offers the added capabilities of directing error diagnostics and/or compiler output to a terminal and of using list-directed input/output. Additionally, the processor supports FORTRAN Interactive Debug.

Under OS, the usability of G1 is enhanced by the TSO FORTRAN Prompter, a TSO command processor that sets up the command procedure to invoke the compiler. The foreground prompter function, available separately for G1, is built into the Code and Go processor for TSO use.

Code and Go and G1 are supported by the FORTRAN IV Library (Mod I), which provides mathematical, service, and input/output routines (including, for OS, support of ASCII data sets) needed by the processors. Additionally, the Mod I library and the processors incorporate the same data conversion routines, which round real constants and real data items on input rather than truncate them (as was the case with the predecessor FORTRAN library and the FORTRAN IV (G) compiler). This provides finer resolution and greater accuracy of results.

FORTRAN IV (H Extended), besides providing extended language capability for computational power, is a true production compiler, utilizing advanced optimization technology to produce efficient object code. The H Extended compiler is designed primarily for background use, although it can, like any program, be invoked for execution in the foreground.

The extended language capabilities of FORTRAN IV (H Extended) include:

- Support for extended precision arithmetic via REAL*16 and COMPLEX*32 data types or via use of a compiler option.
- Support for asynchronous input/output for overlapping reading and writing of unformatted sequential data with processing, thus offering significant execution-time performance improvements for programs involving transmission of large unformatted arrays.
- Automatic function selection to simplify references to built-in and library functions.

As a compilation-time option, the user may specify automatic precision increase, allowing for conversion of floating-point calculations from single to double and double to extended precision.

FORTRAN IV (H Extended) is supported by the FORTRAN IV Library (Mod II), which, in addition to specifically providing routines required by the processor, encompasses all of the functions of the Mod I library; thus, an installation equipped with the Mod II library does not need the Mod I library to support Code and Go or G1.

Selection of FORTRAN processor support to meet an installation's needs is dependent upon the characteristics of the installation's workload. In many instances, maximum effectiveness is achieved with a processor "pair" -- one used for short, one-shot problems and for program checkout, such as Code and Go, and one for object-code optimization for production jobs, such as H Extended. An installation may further require the greater computational power offered by the H Extended compiler.

The FORTRAN program products are supported by a full complement of user publications. A complete description of the supporting publications and a guide to publications' selection for a particular installation's needs are given in the section, "Reference Material."

A detailed description of the facilities provided by TSO and the svstem requirements for it can be found in the publication *TSO Guide,* Order No. GC28-6690. For CMS, this information is contained in the publication *IBM Virtual Machine Facility/370: Planning and System Generation Guide,* Order No. GC20-1801.

LANGUAGE SUMMARY

The language levels supported by Code and Go FORTRAN and FORTRAN IV (G1) are identical and include the Debug facility. [1] The language level supported by FORTRAN IV (H Extended), with the exception of the Debug facility, is a superset of the Code and Go, and G1 levels. For a complete description of the IBM FORTRAN IV language, see the publication *IBM System/360 and System/370 FORTRAN IV Language,* Order No. GC28-6515.

This section summarizes the IBM FORTRAN IV language implemented by the program product processors. In addition, FORTRAN-supplied mathematical function subprograms as well as service subprograms available to the programmer are listed. Language syntax is not shown, only statement usage. Table 1 describes the language elements supported; Table 2 summarizes the statements that are available; Table 3 shows mathematical function subprograms; Table 4 lists service subroutines. A listing of IBM FORTRAN IV features not in ANS FORTRAN is also provided.

Discussions of list-directed input/output, of free-form input format, and of the language features unique to FORTRAN IV (H Extended) are also included. (For users familiar with the predecessor FORTRAN G and H processors, these are the major new language capabilities not available with those processors.)

Language Compatibility

The source language supported by all three processors encompasses and is compatible with the American National Standard (ANS) FORTRAN language (X3.9-1966).

All valid source programs that compiled under the OS FORTRAN IV (E), (G), and (H) processors can be compiled under Code and Go, G1, and, exclusive of the Debug facility, H Extended. Note that the improved accuracy of data conversion may yield different, more accurate numerical results.

Code and Go programs in standard format are interchangeable with G1 programs (a sift utility is available to convert Code and Go free-form input format code into standard format) and, exclusive of the Debug facility, Code and Go or G1 programs can be compiled with H Extended. (Of course, H Extended programs using language features unique to that processor can not be compiled under G1 or Code and Go.)

Object modules compiled by any of the three processors can be combined into an executable program, which can also include object modules compiled by FORTRAN IV E, G, or H. The user should be aware, however, that for execution of these combined modules, the Mod I or Mod II libraries may be required. As an example, to execute a program consisting of G1 modules combined with H Extended modules containing asynchronous input/output statements, the Mod II library is required.

[1] This refers to the compilation-time Debug facility of Code and Go, and G1, and should not be confused with FORTRAN Interactive Debug, which is also discussed in this book.

FORTRAN LANGUAGE ELEMENTS

Table 1. FORTRAN Language Elements

Constants	The following constants are supported: • Numerical: integer, real, and complex numbers Integer constants have a maximum magnitude of $2^{31} - 1$. Real constants have a magnitude range of from 0 or 16^{-65} (approximately 10^{-78}) through 16^{63} (approximately 10^{75}). Depending upon associated length specifications, precision is assured for at least 7 or 16 significant digits -- with extended precision, for approximately 34 significant digits. Complex constants are an ordered pair of real constants. Magnitude range and precision are the same as for real constants. • Logical: a constant that specifies the logical value true or false. • Literal: a character string of from 1 to 255 characters long. • Hexadecimal: a value, of from 2 to 32 hexadecimal digits, used for data initialization.
Symbolic Names	The maximum length of a symbolic name is six characters.
Variables	Variables are supported for representation of numerical and logical data.
Arrays	Arrays, which are identified by symbolic names, may have up to seven dimensions.
Operators and Expressions	Arithmetic, logical, and relational expressions are supported. Operators permitted in these expressions are shown below: • Arithmetic: + - * / ** • Logical: .NOT. .AND. .OR. • Relational: .GT. .GE. .LT. .LE. .EQ. .NE.

FORTRAN STATEMENTS

Table 2 lists FORTRAN statements and their functions. Processor names are shown across the last three column headings. An X in a column indicates that a statement is implemented for that processor; a dash (–) indicates that it is not.

Table 2. FORTRAN Statements (Part 1 of 4)

Statement	Function	Code & Go	G1	H Ext
ASSIGN	Associates a statement number with a symbolic label in an assigned GOTO statement.	X	X	X
assignment	Assigns an arithmetic or logical expression to either a variable or array element.	X	X	X
AT (debug facility)	Identifies the beginning of a debug packet and indicates the point in the program where debugging is to begin.	X	X	–
BACKSPACE	Causes a data set to backspace one record.	X	X	X
BLOCK DATA	Marks the beginning of a BLOCK DATA subprogram.	X	X	X
CALL	Calls a SUBROUTINE subprogram.	X	X	X
CONTINUE	Primarily used in a DO loop to assign an end-of-range label to the loop.	X	X	X
DATA	Defines initial values of variables, array elements, and arrays.	X	X	X
DEBUG (debug facility)	Sets conditions for operation of the debug facility and designates applicable debugging operations.	X	X	–
DEFINE FILE	Describes the characteristics of a data set to be used during direct-access operation.	X	X	X
DIMENSION	Provides information necessary for the system to allocate storage for arrays.	X	X	X
DISPLAY (debug facility)	Displays data in NAMELIST output format.	X	X	–
DO	Initializes a DO loop – the repeated execution of a specified number of statements following the DO statement.	X	X	X
DOUBLE PRECISION	Specifies that the variables named be of type double precision.	X	X	X
END	Defines the end of a main program or subprogram.	X	X	X
ENTRY	Establishes supplementary entry points for subprograms.	X	X	X
EQUIVALENCE	Controls the allocation of data storage within a single program unit.	X	X	X

Table 2. FORTRAN Statements (Part 2 of 4)

Statement	Function	Code & Go	G1	H Ext
explicit	Declares the type of a particular variable or array by its name, rather than its initial character.	X	X	X
EXTERNAL	Passes subprogram names as arguments to other subprograms. For H Extended, the user may declare names of FORTRAN library functions or subroutines to be the names of user-supplied functions or subroutines.	X	X	X
FIND	Causes the next required input record to be found while the current record is being processed, thus increasing object program execution speed (used with direct-access READ).	X	X	X
FORMAT	Specifies the structure of FORTRAN records and the form of the data fields within the reocrds (used with sequential or direct access READ or WRITE).	X	X	X
FUNCTION	Marks the beginning of a FUNCTION subprogram.	X	X	X
GENERIC	Allows the use of a single generic name in requesting a FORTRAN-supplied function that has several names depending on argument type.	–	–	X
GO TO (assigned)	Specifies transfer of control based on a statement number specified in an ASSIGN statement.	X	X	X
GO TO (computed)	Specifies conditional transfer of control.	X	X	X
GO TO (unconditional)	Specifies unconditional transfer of control.	X	X	X
IF (arithmetic)	Tests an arithmetic condition and transfers control based on results of test.	X	X	X
IF (logical)	Tests a logical expression and transfers control on basis of its being true or false.	X	X	X
IMPLICIT	Specifies the type of all variables, arrays, and user-supplied functions whose names begin with a particular letter.	X	X	X
NAMELIST	Declares a name to refer to a particular list of variables or array names (used with sequential READ or WRITE).	X	X	X
PAUSE	Interrupts program execution and, optionally, displays information to the operator.	X	X	X
PRINT	Specifies that a printed listing be produced.	X	X	X
PUNCH	Specifies that a card deck be produced.	X	X	X

Table 2. FORTRAN Statements (Part 3 of 4)

Statement	Function	Code & GO	G1	H Ext
READ (asynchronous)	Provides high-speed transmission of unformatted data from an external data set to an array in main storage.	—	—	X
READ (direct-access)	Transfers data from a direct-access device into main storage.	X	X	X
READ (formatted)	Retrieves data sequentially from a data set in accordance with FORMAT statement specifications.	X	X	X
READ (list-directed)	Retrieves data sequentially in the order specified in the program, and the format used in the data.	X	X	X
READ (unformatted)	Retrieves in sequential order a single unformatted record from a data set (used with unformatted WRITE).	X	X	X
READ (using NAMELIST)	Retrieves data sequentially from a data set according to lists of variables or arrays declared in a NAMELIST statement (used with WRITE using NAMELIST).	X	X	X
RETURN	Returns control to calling program.	X	X	X
REWIND	Causes a subsequent READ or WRITE statement to read data from or write data into the first record of a data set.	X	X	X
statement function definition	Specifies operations to be performed whenever the definition is referred to.	X	X	X
STOP	Terminates program execution and, optionally, displays up to five decimal digits to the operator.	X	X	X
SUBROUTINE	Marks the beginning of a SUBROUTINE subprogram.	X	X	X
TRACE OFF (debug facility)	Stops the recording of program flow by statement number.	X	X	—
TRACE ON (debug facility)	Initiates recording of program flow by statement number.	X	X	—
WAIT	Redefines a receiving area and makes it available for reference, or makes a transmitting area available for redefinition (used with asynchronous READ and WRITE statements).	—	—	X
WRITE (asynchronous)	Provides high-speed transmission of unformatted data from an array in main storage to an external data set.	—	—	X
WRITE (direct-access)	Transfers data from main storage to a direct-access device.	X	X	X
WRITE (formatted)	Writes data sequentially into a data set in accordance with FORMAT statement specifications.	X	X	X

Table 2. FORTRAN Statements (Part 4 of 4)

Statement	Function	Code & GO	G1	H Ext
WRITE (list-directed)	Writes data sequentially into a data set in the order and from the locations that the user specifies.	X	X	X
WRITE (unformatted)	Writes in a sequential order a single unformatted record into a data set (used with unformatted READ).	X	X	X
WRITE (using NAMELIST)	Writes data from lists of variables or arrays declared in a NAMELIST statement (used with READ using NAMELIST).	X	X	X

MATHEMATICAL FUNCTION SUBPROGRAMS

Table 3 lists mathematical function subprograms and their entry names. Entry names starting with the letter D (except for DBLE and DIM) indicate double precision; starting with Q, extended precision (H Extended only); starting with C (except for CMPLX, CONJG, COS, COSH, COTAN), complex functions. Entry names starting with CD indicate complex double precision; starting with CQ, complex extended precision (H Extended only).

Table 3. Mathematical Function Subprograms

General Function	Entry Name
Natural and common logarithm	LOG,ALOG,DLOG,QLOG,CLOG,CDLOG,CQLOG, LOG10,ALOG10,DLOG10,QLOG10
Exponential	EXP,DEXP,QEXP,CEXP,CDEXP,CQEXP
Square	SORT,DSQRT,QSQRT,CSQRT,CDSQRT,CQSQRT
Arc sine and arc cosine	ASIN,ARSIN,DARSIN,QARSIN,ACOS,ARCOS, DARCOS,QARCOS
Arc tangent	ATAN,DATAN,QATAN2
Sine and cosine	SIN,DSIN,QSIN,COS,DCOS,QCOS
Absolute value	IABS,ABS,DABS,QABS,CABS,CDABS,CQABS
Error function	ERF,DERF,QERF
Maximum and minimum values	MAX,MAX0,AMAX0,MAX1,AMAX1,DMAX1,QMAX1, MIN,MIN0,AMIN0,MIN1,AMIN1,DMIN1,QMIN1
Truncation	AINT,DINT,QINT,INT,IDINT,IQINT
Obtain real part of a COMPLEX argument	REAL,DREAL,QREAL
Obtain imaginary part of a COMPLEX argument	IMAG,AIMAG,DIMAG,QIMAG
Precision increase	DBLE,QEXT,QEXTD
Express two REAL arguments in complex form	CMPLX,DCMPLX,QCMPLX
Obtain conjugate of a COMPLEX argument	CONJG,DCONJG,QCONJG

SERVICE SUBROUTINES

Table 4. Service Subroutines

Name	Purpose
DUMP	Dump selected portions of storage on the output data set and terminate execution.
PDUMP	Dump selected portions of storage on the output data set and continue execution.
DVCHK	Test for divide check exception.
OVERFL	Test for exponent overflow or underflow.
EXIT	Terminate execution.
SLITE	Alter status of sense lights.
SLITET	Test and record status of sense lights (After the test, the sense light that was tested is turned off.).

IBM FORTRAN IV FEATURES NOT IN ANS FORTRAN

Asynchronous input/output statements (H Extended only)
Direct access input/output statements
Dummy arguments enclosed in slashes
ENTRY
ERR and END parameters in a READ
EXTERNAL statement extension for user-supplied subprograms (H Extended only)
Function name in explicit specification statements
Generalized subscripts
GENERIC statement (H Extended only)
Hexadecimal constant
IMPLICIT
Initial data values in explicit specification statements
Length of variables and arrays as part of type specifications
Length specification in FUNCTION statement
List-directed input/output statements
Literal as actual argument in function reference
Literal enclosed in apostrophes
Mixed-mode expressions
More than three dimensions in an array
Multiple exponentiation without parentheses to indicate order of computation
NAMELIST
Object-time dimensions transmitted in COMMON
PAUSE 'message'
PRINT b, list
PUNCH b, list
READ b, list
RETURN i
T and Z format codes, and extensions to G format code

COMPLEX*16
INTEGER*2
LOGICAL*1 } data types

COMPLEX*32
REAL*16 } data types (H Extended only)

LANGUAGE CAPABILITIES

The following section describes list-directed input/output, free-form input format, and the language features unique to the FORTRAN IV (H Extended) processor. (For users familiar with the predecessor FORTRAN IV G and H processors, these are the major new language capabilities not available with those processors.)

List-Directed Input/Output

List-directed input/output, supported by the Code and Go, G1, and H Extended processors, simplifies data entry and output by freeing the user from having to code FORMAT statements. Use of list-directed input/output requires an installation to have either of the program product FORTRAN IV libraries, Mod I or Mod II.

List-directed processing of READ, WRITE, PUNCH, or PRINT statements is specified by replacing the FORMAT statement label with an asterisk. No FORMAT statement is used. Data to be read at execution of a statement may be entered without regard to column boundaries. Individual data items may not be split between cards or lines, except for complex items and literals in quotation marks. Additional information on list-directed input and output is provided below:

Input

Input entries are separated by blanks or commas, with successive commas indicating values to be omitted. The input list may be cut short with a slash. For example, the statement

```
10 READ (5,*)  (A(I),I=1,5)
```

will read data from the device associated with data set reference number 5 (which may refer to the user's terminal or any other FORTRAN-supported I/O device.) If that number refers to a terminal, the user is prompted for input with a question mark (unless prompting has been suppressed). In addition, if -- as in the example -- the statement is labeled, the statement label is printed out following the question mark. Prompt and reply for the required input might look like:

```
? 00010
5.,33.44,5.E-9,2. 6.
```

A blank serves as the delimiter between the last two entries. Alternatively, the user may enter each value on a separate line or card (if the READ statement is not directed to the terminal). If the user wishes to retain the current values of the last two elements in array A, he may simply type in:

```
5.,33.44,5.E-9/
```

The user may set k items to the same value by entering k* value, as for NAMELIST input. For example, the elements of array A can be set to the value 56.3 by the input line:

```
5*56.3
```

Integer, real, complex, literal, and logical constants may be entered as input for list-directed READ statements.

Output

The list-directed WRITE statement

```
WRITE(6,*)X,L,N
```

will write out the current values of X,L, and N on the device defined by data set reference number 6 (which may refer to the user's terminal or any other FORTRAN supported input/output device). Real and complex numbers are written using E exponents only. Integer and logical values are also written as output. For example, the output line might be:

```
3.1E + 11 T 205
```

List-directed WRITE or PRINT statements will not produce output in the form of literal constants.

Advantages of List-Directed Input/Output

The advantages of list-directed over conventional, formatted input/output in terms of time and effort saved can best be seen by example. In the following example -- first shown for formatted I/O and then repeated using list-directed I/O -- a READ statement is used to read a record containing constants of various types into main storage. When the READ is executed, the value 1.2 is read into each of the first 10 elements of ARRAY; text is read into the locations HEAD1 and HEAD2 (for formatted I/O, locations HEAD 1 and HEAD 2 are initialized as a result of the REAL*8 specification statement); a complex value is read into location A; a double precision number is read into location B; and the logical value false is placed into location P.

Formatted Input/Output

```
REAL*8 HEAD1/'HEADING'/,HEAD2/'FOOTING'/,B
READ(5,10)  (ARRAY(I),I=1,10),A,B,P
10 FORMAT(10F2.1,E8.2,E6.2,D8.3,L5)
```

The data associated with these statements follow:

Card Column or Typing Position

```
1                    20        28      34        42   47
1212121212121212122.17E+153.14E00.125D-3FALSE
```

List-Directed Input/Output

```
REAL*8 HEAD1,HEAD2,B
READ(5,*)  (ARRAY(I),I=1,10),HEAD1,HEAD2,A,B,P
```

The data associated with these statements follow:

```
10*1.2 "HEADING' 'FOOTING' (2.17E+15,3.14E0)
0.125D-3 .FALSE.
```

As is obvious, the programmer saves an appreciable amount of coding time by using list-directed I/O for this single read operation. Consider, then, the time that is saved by using list-directed I/O for an entire program. Note, too, that with list-directed I/O the programmer's concern with data layout and specific card-column positions is eliminated, as, of course, are all errors associated with this tedious process.

FORTRAN IV (H Extended) Language Features

This section describes the language extensions uniquely available with FORTRAN IV (H Extended).

Extended Precision

Todays scientists, mathematicians, and engineers are increasingly concerned with problems whose solutions demand greater accuracy than in the past. These demands are met by the extended-precision capability, which provides mathematical results in 112 significant bits -- the equivalent of 33 to 34 decimal digits. The processor recognizes and can process two new lengths for existing data types: REAL*16 and COMPLEX*32.

As a consequence, program results previously limited by insufficient precision can now be improved using the new data lengths, and migration difficulties caused by precision are eliminated.

For real and complex data items, the maximum number of storage locations that are allocated per data item is twice the previous maximum. Also, the FORTRAN-supplied functions required to support the extended-precision data types are provided, with two exceptions. Extended precision equivalents of the GAMMA and ALGAMA functions are not included.

For information related to extended precision, see the discussion of automatic precision increase under the heading "FORTRAN IV (H Extended)" in the section "Program Product Processors."

Asynchronous Input/Output

A high-speed asynchronous input/output capability is provided, by which unformatted sequential data is transmitted between external data sets (residing on tape, disk, drum, or data cell) and arrays in main storage; while such transmission is taking place, other program statements may be executed. Because of the method of data transfer implemented, as well as the ability to maximize the extent of I/O overlap with computation, significant performance improvement is achieved for programs involving transmission of large unformatted arrays. Performance improvements vary according to program type. Maximum improvement is shown, for both elapsed time and CPU

time, for programs using large unformatted arrays for input/output.

The FORTRAN IV language for H Extended includes special forms of READ and WRITE statements for initiating a transmission, and a WAIT statement for completing the transmission cycle. (Note that under CMS, programs using asynchronous input/output statements can be compiled, but not executed. Execution must take place under OS. Through the facilities of VM/370, the user can readily compile under CMS and then switch to OS for execution.)

Automatic Function Selection

The automatic function selection facility provides a concise set of generic names (usually the names of the single-precision form of the function) for built-in library functions; these names can be used in place of the larger set of specific (data-type dependent) function names. Automatic function selection is requested by the specification statement, GENERIC. The user's task of referring to built-in and library functions is thus simplified because the same name can always be used for a function, even though the type of the function and the type of its arguments may vary with each use. Without automatic function selection, different names would have to be coded, depending upon the type of the function and its arguments.

An example of illustrating the advantages of GENERIC follows:

Without GENERIC

```
REAL*8   C,D
REAL*16  E,F
COMPLEX*8  P,R
COMPLEX*16 S,T
COMPLEX*32 U,V
    .
    .
    .

C=DCOS(D)
E=QCOS(F)
P=CCOS(R)
S=CDCOS(T)
U=CQCOS(V)
```

With GENERIC

```
GENERIC
REAL*8   C,D
REAL*16  E,F
COMPLEX*8  P,R
COMPLEX*16 S,T
COMPLEX*32 U,V
    .
    .
    .
C=COS(D)
E=COS(F)
P=COS(R)
S=COS(T)
U=COS(V)
```

When not using GENERIC, the programmer must refer back to his specification statements to determine the type and the length of the variables in his program -- information he needs to specify the appropriate function name.

When GENERIC is used, the need to refer back to specification statements is eliminated. The programmer simply specifies the function name, and the appropriate form of the function is automatically called during compilation.

EXTERNAL Statement Extension

An extension to the EXTERNAL statement enables the user to "detach" the names of FORTRAN library subprograms. Detachment of a subprogram name causes that name

to be no longer associated with the FORTRAN-supplied library subprogram; instead, it is considered to be the name of a user-supplied subprogram. The extension consists of the special character "&" prefixed to the subprogram name when it appears in the EXTERNAL statement. The extension enables the user to supply his own subprograms in place of identically named FORTRAN library subprograms, with the assurance that the compiler will interpret all subprogram references correctly.

Code and Go Free-Form Input Format

Code and Go FORTRAN will accept source input in either of two forms: (1) standard fixed-form (80-character) FORTRAN input records (see the publication *IBM System/360 and System/370: FORTRAN IV Language;* or (2) variable- or fixed-length records prepared in free-form input format (sometimes called *free-form source*), which frees the user from all card column restrictions in writing his FORTRAN source program.

With free-form source, the user begins a statement in any typing position or card column and indicates continuation by ending his line with a hyphen. Contrasted with conventional source coding, the advantages are many -- especially for the terminal user. Typing is made quicker and easier because there is no need to be concerned with setting tabs or, under OS, with setting a margin stop at column 72, since free-form statements, under OS, may extend beyond column 72.

As an example, the standard-form statement:

Typing position or card column:

```
1      67
C       SAMPLE TEXT
   10   D=10.5
        GO TO 56
  150   A=B+C*(D+E**F+
       1G+H-2.*(G+P))
        C=3
```

can be written in free-form as the following:

Typing position or card column:

```
1     67
"SAMPLE TEXT
10 D=10.5
GO TO 56
150 A=B+C* (D+E**F+-
G+H-2* (G+P))
C=3.
```

In general, the rules for free-form coding are similar to those for conventianal coding. Some minor differences do exist though. For example, note that a double quotation mark (") replaces a C to indicate a comment card (an asterisk may also be used). For a full discussion of free-form input format rules, see the User's Guide appropriate to your system.

A sift utility supplied with Code and Go provides for two-way conversion between free-form and standard-form source statements under OS, and for one-way conversion from free-form to standard-form under CMS. Standard-form records can be submitted to other compilers for processing. The sift utility can be invoked by the TSO or CMS CONVERT command or can be run in a batch mode.

PROGRAM PRODUCT PROCESSORS

The following sections describe the principal features of the Code and Go, G1, and H Extended processors.

CODE AND GO FORTRAN

Code and Go FORTRAN is a compile-and-go processor that compiles at a fast rate and then invokes the system loader (either OS or CMS) to link to library subprograms and initiate execution. Performance is one of the two major considerations in the design of Code and Go. The other, particularly for the foreground user, is its ease of use.

First, Code and Go accepts source statements in free-form input format, simplifying program entry from the terminal.

Second, Code and Go, when operating under TSO or CMS, incorporates a time-sharing command processing function. Since Code and Go is also able to invoke the loader after compilation has completed successfully, a single command can effect the complete compilation-execution cycle, including allocation of required data sets.

This simplicity of use makes Code and Go particularly suitable for the problem-solver, since it reduces his concern with system functions to a minimum. Similarly, the production programmer who is using Code and Go in the foreground for debugging can concentrate his attention on his source program without the effort of setting up a command procedure. Moreover, he can use free-form source even if he intends to compile later under a compiler that does not accept it, since Code and Go also includes a sift utility that will convert his source statements to standard form.

In the foreground, easy and rapid debugging of a Code and Go program is made possible by a combination of quick compilation and the debugging aids provided by compiler diagnostics and FORTRAN Interactive Debug. [1] When no compilation errors are found, the compiler -- in conjunction with the loader -- automatically starts execution of the compiled code. (Under CMS, the user can request that execution be delayed.)

For both foreground and background, Code and Go provides two levels of diagnostic messages: one aimed for the experienced programmer; the other for those who are not full-time programmers and will need additional assistance for error correction. Diagnostics for the experienced programmer are short and concise, e.g., INVALID LOG CONSTANT. Those for the less experienced are more tutorial -- as an example, the more tutorial form of INVALID LOG CONSTANT is LOGICAL CONSTANTS CAN ONLY BE TRUE. OR .FALSE.

---

[1] See the section "Application Development Support Products" for information on FORTRAN Interactive Debug.

If the user has access to the FORTRAN IV Library (Mod I), he can reduce and simplify his source coding and data entry requirements by using list-directed input/output statements, described previously in the section "Language Review" under the heading "List-Directed Input/Output."

Under OS, Code and Go foreground programs compiled without error go into execution automatically; in the background, a compile-only option is provided. Additionally for background or batch jobs, users can:

- Obtain source listings and object module decks
- Invoke the linkage editor rather than the loader prior to execution
- Indicate the maximum number of lines per page for a source listing.

Under CMS, in both foreground and background, Code and Go operates as in the OS background; users can:

- Obtain source listings and object module decks
- Request compilation only
- Indicate the maximum number of lines per page for a source listing

In summary, as a foreground tool, Code and Go is specifically designed for users who place a premium on quick compilation and execution, on simplicity of use, and on easy and rapid debugging.

System Requirements and Considerations

Operation of Code and Go requires the minimum area of contiguous main storage shown in Table 5. The floating-point instruction set is required and, for OS foreground processing, the Loader. (The Loader is also required if use is to be made of the immediate-execution option in the background.) Under OS, the processor can handle most program units consisting of 230 or fewer statements in the minimum region. (For compilation with the RUN subcommand of the TSO EDIT command, optimal performance is obtained for compilations of approximately 50 standard-form or 100 free-form statements.) Code and Go can handle larger source programs in larger areas; for each additional 10K bytes of main storage provided, about 75 additional statements can be processed.

Execution of the Code and Go processor under OS must be performed with a system that has been generated with a SUPRVSOR macro-instruction that (1) specifies the IDENTIFY function as an option and (2) specifies the WAIT operand as MULTIPLE.

Operation of Code and Go under TSO requires the minimum TSO configuration. See the publication *TSO Guide,* Order No. GC28-6608, for details. Operation under CMS requires the minimum CMS configuration. See the publication *IBM Virtual Machine Facility/370: Planning and System Generation Guide,* Order No. GC20-1801.

Table 5. Main Storage Requirements

	For OS Background or CMS Virtual Storage	For OS (TSO) Foreground [1]		
		Without EDIT	With EDIT [2] Resident in Link Pack Area	With EDIT Not Resident in Link Pack Area
FORTRAN IV (G1)	90K	110K	138K	150K
Code and Go FORTRAN Compiler	88K	112K	140K	152K
Sift Utility	8K	22K [3]	-	-

[1] These region requirements assume that the TMP and the service routines GETLINE, PUTLINE, and PUTGET are resident in the TSO Time Sharing Link Pack Area.

[2] Refers to the 12K "main line" of EDIT, consisting of control, service, and special access method routines.

[3] Fits in minimum TSO region.

FORTRAN IV (G1)

The FORTRAN IV (G1) processor was developed for installations that do not reqauire the extremely fast compile-to-execute time of the Code and Go processor. Its object code is more efficient than that of the Code and Go processor, particularly with regard to DO loops, thus making it more suitable for "production" programs -- ones that will be executed frequently.

In providing the ability to produce object listings and storage maps, FORTRAN IV (G1) complements Code and Go in the TSO foreground. In addition, FORTRAN IV (G1) can produce source listings and can store object modules in both foreground and batch operation, under OS and CMS. Like Code and Go, the G1 processor supports the use of list-directed input/output and of FORTRAN Interactive Debug.

The processor provides a terse form of output, comprising error messages and compiler statistics, suitable for terminal display. This output is normally suppressed when the compiler is operating in batch mode. Source and object listings and storage maps may be directed to the terminal (when compiling in the foreground) or to any other output device.

Use of FORTRAN IV G1 under TSO is simplified by the TSO FORTRAN Prompter; [1] the processor, as previously noted, also supports the use of the FORTRAN Interactive Debug program product (see the section "Application Development Support Products").

[1] When using G1 under CMS, the command processing functions of the Prompter are not required; they have been built into the processor itself.

System Requirements and Considerations

Operation of FORTRAN IV (G1) requires the minimum area of contiguous main storage shown in Table 5. The floating-point instruction set is required. In a minimum area, the processor can handle most program units consisting of 400 or fewer statements; larger units can be handled in larger areas.

Operation of FORTRAN IV (G1) under TSO requires the minimum TSO configuration. See the publication *TSO Guide,* Order No. GC28-6698, for details. Operation under CMS requires the minimum CMS configuration. See the publication *IBM Virtual Machine Facility/370: Planning and System Generation Guide,* Order No. GC20-1801.

FORTRAN IV (H EXTENDED)

FORTRAN IV (H Extended), a true production compiler, utilizes advanced optimization technology to produce efficient object code.

Compilation-time options are available to optimize execution speed. In the process, object module size is often reduced. The user can select optimization facilities at several levels. These will improve object code by -- to cite a few examples -- increasing the program's efficiency in its handling of nested DO loops and subscripts, and in its use of registers and branch instructions.

In addition to the greater computational power provided by language extensions for extended precision, asynchronous input/output, automatic function selection, and EXTERNAL statement usage, the H Extended processor offers an automatic precision increase facility. This facility, specified at compilation-time, provides an automatic procedure for converting all single–precision floating-point variables, constants, and functions to double precision and/or double-precision floating-point variables, constants, and functions to extended precision.

The automatic precision increase facility is an aid in the conversion of programs written for other computers whose precision is greater than that afforded by the System/360 32 bit word size. The facility helps to solve the loss-of-precision problems that occur when such programs are run on System/360 Without it, the System/360 user experiencing loss of precision has to convert his programs to double precision manually, a process that is complex, time-consuming, and prone to error.

Also, automatic precision increase can be used for programs in which double-precision calculations are inadequate to meet the user's precision requirements. Such programs can be converted automatically to extended precision.

The automatic precision increase facility should be considered as a tool for precision conversion; use of the facility does not assure that every FORTRAN source progam will be executed correctly in the higher precision.

System Requirements and Considerations

FORTRAN IV (H Extended) operates under CMS and on all machines supported by OS that have at least 256K bytes of amin storage. The floating-point instruction set is required, as is the FORTRAN IV (Mod II) library. For machines with extended

precision capabilities, the extended floating-point feature is required for optimum performance. Under OS, sufficient devices to support the SYSIN, SYSPRINT, SYSLIN, SYSPUNCH, SYSUT1, and SYSUT2 data sets are required.

FORTRAN IV (H Extended) requires a minimum of 160K bytes of main storage under OS; for CMS, 450K of virtual storage is required. Approximately 200 to 300 source statements can be compiled when this amount of storage is available to the compiler. Up to 160 tracks on an IBM 2311 Disk Storage Unit may be required for secondary storage.

Storage requirements can be affected by the use of extended-precision variables, constants, and calculations in the source program or by use of the automatic precision increase facility.

Calculations for extended-precision variables require approximately four times the number of instructions that would be required for the same calculations on single-precision or double-precision variables. On machines requiring use of the extended-precision simulator, additional storage will be needed for the simulator. Automatic precision increase involves promotion of data from one type to another (for example, single precision to double precision); it can also involve padding of variables. For the promotion operation, data storage is doubled for each variable that is promoted. For the padding operation, data storage is doubled for each variable that is padded. There is an additional increase in storage requirements when double-precision data is promoted to extended precision, because the compiler must generate a greater number of instructions for extended-precision calculations, as noted above.

FORTRAN IV LIBRARY (MOD I) AND FORTRAN IV LIBRARY (MOD II)

The FORTRAN IV Library (Mod I) interfaces with the Code and Go processor and the G1 processor to (1) direct PAUSE and STOP statement messages to the terminal, (2) provide the routines necessary for use of list-directed input/output, and (3) permit, for OS, the use of tape data sets written in the American National Standard Code for Information Interchange (also referred to as ASCII).

FORTRAN IV OS processors other than G1 and Code and Go, (excluding H Extended, which requires the Mod II library) can use the routines of the Mod I library, taking advantage of the data conversion routines and, for OS but not CMS, ASCII support. Use of the list-directed input/output function requires specific list-directed I/O support from the compiler.

Conversly, Code and Go or G1 users not requiring list-directed I/O or ASCII support can use the predecessor FORTRAN IV library.

The FORTRAN IV Library (Mod II), which provides specific support of the mathematical, service, and input/output routines needed by the H Extended processor, also encompasses all of the FORTRAN IV Library (Mod I) support. Consequently, an installation equipped with the Mod II library need not have the Mod I library to support Code and Go or G1.

ASCII Data Sets

For OS, the Mod I and Mod II libraries accept as input and can create as output magnetic tapes written in ASCII code. Only tapes read or written by list-directed input/output statements or under format control at execution time are acceptable. ASCII tape data sets that are unlabeled or that have standard ASCII labels are processed for F, U, or D type records.

Conversion Routines

The input/output conversion routines in the Mod I and Mod II libraries produce more accurate results than the routines of the OS FORTRAN IV library they replace. This improvement is achieved by use of an algorithm that rounds values instead of truncates, on input, and that maintains more accuracy in intermediate values while performing the conversion. A REAL*4 or REAL*8 number that is written and then read will produce the same internal representation that existed prior to the write and read. This is called an onto mapping for an out/in conversion.

The same input conversion routines are contained in the three program product processors to ensure compatibility.

STORAGE REQUIREMENTS

Table 6 lists the increase in storage requirements for the Mod I and Mod II libraries over the predecessor, OS FORTRAN IV library.

Table 6. FORTRAN IV Library (Mod I) and (Mod II) Storage Increases

Function	Storage Increase	
	Mod I	Mod II
Secondary Storage	18K bytes [1]	22K bytes [1]
Primary Storage [2]		
Base	900 bytes	2100 bytes
List-directed I/O	2600 bytes	2600 bytes
Extended-precision conversion	-	700 bytes
Asynchronous I/O	-	5000 bytes [3]
Subprograms	-	[4]

[1] 52 tracks on an IBM 2311 for Mod I; 73 tracks for Mod II.

[2] Total primary storage increase for the Mod I library is the sum of the base increase, the increase for list-directed I/O, and the length of each subprogram called by the main program; for Mod II, it is the sum of the base increase, increases for list-directed I/O, extended precision, and asynchronous I/O (as applicable), and the length of each subprogram called by the main program.

[3] Base storage increase only. Add 8 bytes for each FORTRAN logical unit assigned to the system; add 250 bytes for each unit processing asynchronous I/O requests.

[4] Extended-precision mathematical subprograms require 3.5 to 6 times the main storage of the corresponding double-precision subprograms.

The TSO FORTRAN Prompter, offered for use with the G1 processor, and FORTRAN Interactive Debug, which can be used with Code and Go and with G1, are described in this section. Both products are designed to assist the time-sharing user in developing his program from his terminal with a maximum of ease.

TSO FORTRAN PROMPTER FOR FORTRAN IV (G1)

The TSO FORTRAN Prompter provides the most convenient means of invoking the FORTRAN IV (G1) processor in the TSO foreground.[1] When the prompter has been added to a TSO installation, FORTRAN IV (G1) may be invoked by the TSO FORT command or by the TSO RUN command. Operands of the FORT command allow the specification of various compiler options: whether or not a listing is to be produced; the contents of the listing, and where it is to be printed or stored; whether or not an object module is to be produced; and whether or not diagnostic messages are to be sent to the terminal. All operands, except the input data set name, can default to standard values. (When the RUN command is used, parameters for the processor may not be specified; the prompter will supply default values for these parameters.)

FORTRAN IV (G1), the terminal user, and TSO interact through the TSO FORTRAN Prompter. The prompter reads and interprets the FORT or RUN command parameter string used to invoke it, prompts the terminal user for any information that has been omitted or incorrectly entered, allocates required data sets, and passes parameters to the G1 processor. Thus, the prompter provides the terminal user with a conversational means of setting up the proper parameter lists and data sets that are to be used by the processor.

Storage Requirements

The prompter is transient in the FORTRAN IV (G1) region and is overlayed by the compiler. Hence, the storage requirements are satisfied by any region capable of running the compiler via the TSO CALL command.

FORTRAN INTERACTIVE DEBUG

FORTRAN Interactive Debug, which operates under both TSO and CMS, allows the programmer to debug both Code and Go and G1 programs from his terminal through the use of simple conversational subcommands and system responses. The use of FORTRAN Interactive Debug requires compilation, with the specification of TEST as a compilation option.

[1] When using G1 under CMS, the command processing functions of the Prompter are not required; they have been built into the CMS interface module.

When a program is compiled with the TEST option, an object program containing all necessary linkages to the debugging routines is produced and stored. (Code and Go programs compiled with TEST are not automatically loaded and executed the way ordinary TSO Code and Go programs are.) After compiling with the TEST option, the programmer invokes Interactive Debug by issuing a TESTFORT command. TESTFORT provides parameters for calling private user libraries during execution and naming a print data set for debugging output.

Once within TESTFORT, the programmer has access to TESTFORT subcommands. Using the subcommands, and referring to program data by line numbers, statement labels, and symbolic FORTRAN names (instead of internal sequence numbers, internal registers, or machine code), the programmer has complete control over the execution of his program. He can start and stop execution, establish breakpoints, display and alter values of variables, control the logical flow of his program, monitor the frequency of execution of program statements, monitor logical and arithmetic conditions, dynamically trace and monitor execution, and control error conditions. Once his program is debugged, the programmer returns to EDIT mode to make the necessary permanent program corrections. Table 7 lists TESTFORT subcommands and their facilities.

Any Code and Go FORTRAN or FORTRAN G1 program that will run under EDIT can be debugged with FORTRAN Interactive Debug, except for programs containing Debug facility statements (i.e., compilation-time Debug packet statements). Once these statements are removed, however, these programs can also be debugged using FORTRAN Interactive Debug.

System Requirements and Considerations

FORTRAN Interactive Debug operates with Code and Go and G1 under TSO or CMS. Interactive Debug requires the FORTRAN IV Library (Mod I) or (Mod II) *with* the Extended Error Handling feature. The latest release of the Code and Go or G1 processor available at Interactive Debug release time will be required. For installations using the TSO FORTRAN Prompter, the latest release of the prompter will also be required.

For operation under TSO, Interactive Debug requires the minimum TSO configuration and a minimum region size of 142K bytes.

Table 7. FORTRAN Interactive Debug (TESTFORT) Subcommands (Part 1 of 2)

Subcommand Name	Function
AT	The AT subcommand sets breakpoints in the FORTRAN program and allows the programmer to assume control at the breakpoint or to supply a list of subcommands for execution at the breakpoint.
END	The END subcommand discontinues testing and returns the user to command mode.
ERROR	The ERROR subcommand allows the user to dynamically control the extended error handling facility from the terminal.
FIXUP	The FIXUP subcommand lets the user specify his own corrective action when execution errors occur, and resumes program execution after performing the fixup.
GO	The GO subcommand resumes execution of the FORTRAN program after it has stopped for a breakpoint, an attention interruption, or an execution error. Execution may be resumed at any point in the executing program unit.
HALT	The HALT subcommand allows the user to get control of his program if conditions specified with an IF subcommand are true.
HELP	The HELP subcommand provides information about the function, syntax, and operands of any of the TESTFORT subcommands.
IF	The IF subcommand allows the user to define an arithmetic or logical condition, and performs a user-specified operation (by subcommand) if the condition is true.
LIST	The LIST subcommand allows the user to display values of any program variables at the terminal or on a print data set. The display can be produced in a variety of output formats.
LISTBRKS	The LISTBRKS subcommand provides a list of all breakpoints set in a program, and a list of all WHEN condition monitoring (active and inactive) that is defined for the program.
LISTFREQ	The LISTFREQ subcommand produces a listing of the frequency of execution of any statement in the program, or a listing of any statements that have not been executed at all.
NEXT	The NEXT subcommand sets a temporary breakpoint at the next FORTRAN statement that is to be executed.
OFF	The OFF subcommand removes breakpoints that have been set with an AT subcommand.
OFFWN	The OFFWN subcommand turns off condition monitoring that has been activated by a WHEN subcommand.
PURGE	The PURGE subcommand is used in an attention interruption to suppress listing of printed output from a single executing subcommand without aborting any other subcommands that may be pending.
QUALIFY	The QUALIFY subcommand is used to refer subcommands to program units other than the program unit in execution when the subcommand is given.

Table 7. FORTRAN Interactive Debug (TESTFORT) Subcommands (Part 2 of 2)

Subcommand Name	Function
RUN	The RUN subcommand removes all program breakpoints and resumes execution of the program at a point specified by the user. Execution is completed without further testing.
SET	The SET subcommand allows the user to change the value of any variable in his program.
SOURCE	The SOURCE subcommand produces a listing of any of the FORTRAN source code at the terminal.
TRACE	The TRACE subcommand allows the user to trace transfers of control within his program by line number; he can trace either (1) subroutine entries and exits, or (2) transfers of control by statement and line number.
WHEN	The WHEN subcommand is used to set up monitoring of any arithmetic or logical condition, or to reinitiate monitoring of a previously defined condition.
WHERE	The WHERE subcommand informs the programmer of the line number of the next statement to be executed. Optionally, it can provide a traceback of all program transfers that led to his current statement.

The Code and Go FORTRAN Processor, the FORTRAN IV (G1) Processor, the FORTRAN IV (H Extended) Processor, the FORTRAN IV Library (Mod I), the FORTRAN IV Library (Mod II), the TSO FORTRAN Prompter, and FORTRAN Interactive Debug are supported by the publications described below.

Many of these publications pertain to more than one of the products, since most users will be using them in some combination. In these cases, the function being described is identified with the product or product combination that implements it (e.g., list-directed I/O requires the Mod I library and one of the processors), so that if a user does not have the product(s) available to him, he is steered around the function.

To determine which publications are needed for a given product or combination of products, consult Table 8, following the descriptions.

An installation using TSO or CMS will have additional information needs that are provided through other publications not described here. A description of the available TSO and CMS literature will be found in *IBM System/360 and System/370 Bibliography,* GA22-6822.

IBM SYSTEM/360 and SYSTEM/370
FORTRAN IV LANGUAGE
GC28-6515

This publication provides complete reference information for the full FORTRAN IV language supported by the Code and Go FORTRAN, FORTRAN IV (G1), and FORTRAN IV (H Extended) processors. This language encompasses the American National Standard FORTRAN, with IBM extensions such as the Debug facility, list-directed input/output, etc.

The publication also includes information on use of the mathematical and service subprograms provided in the FORTRAN library.

Readers of this publication are assumed to have some prior knowledge of programming techniques and of the FORTRAN language. Users new to FORTRAN can use the set of programmed instruction texts, *FORTRAN IV for IBM System/360,* Order Numbers SR29-0080 through SR29-0087, as an introduction to the language.

OS
FORTRAN IV MATHEMATICAL AND SERVICE SUBPROGRAMS
GC28-6818

This publication provides comprehensive information about the mathematical and service subprograms in the FORTRAN IV Library of which the Mod I and Mod II libraries are extensions.

This book includes mathematical subprogram performance data, argument ranges and accuracy data, the effect of argument error on the accuracy of results, and the mathematical algorithms used in the implementation.

For many users, the information in the FORTRAN IV language manual cited above will be sufficient for the use of these subprograms. This publication, which has a mathematical orientation, is intended for users who require specific, detailed information about the mathematical algorithm implemented, the effect of argument error, etc.

OS
FORTRAN IV MATHEMATICAL AND SERVICE
SUBPROGRAMS SUPPLEMENT FOR MOD I AND MOD II LIBRARIES
SC28-6864

This publication, a supplement to the Mathematical and Service Subprograms manual (GC28-6818), contains performance data, argument information, and implementation algorithms for the extended precision mathematical subprograms contained in the Mod II library, as well as storage estimates for modules containing imput/output, error handling, and data conversion routines.

The mathematical orientation of this publication is similar to that of the Mathematical and Service Subprograms manual, and for many users the information in the FORTRAN IV language manual (GC28-6515) will be sufficient for use of the subprograms.

OS (TSO)
CODE AND GO FORTRAN PROCESSOR
TERMINAL USER'S GUIDE
SC28-6842

This publication is for the user who is using the Code and Go FORTRAN processor in the TSO foreground. The reader of this manual is assumed to know the FORTRAN IV language (as described in the language manual cited above) but is not required to have any prior knowledge of TSO or any other time-sharing experience.

This publication is designed to both instruct the new user and serve as a reference for the experienced user of Code and Go. In addition to information about the processor itself, this guide is a primary source of information for that subset of the TSO command language that is directly supported by or directly relevant to the use of Code and Go, including source program creation and editing (using line numbers), syntax checking, and compilation and execution. It also describes the use of the terminal I/O facilities available through the FORTRAN IV Library (Mod I).

This publication assumes that the installation has established a TSO log-on procedure for the Code and Go user and has advised him of the procedure's name and its limits; the minimum requirements for this procedure are given in the Code and Go installation reference material publication, described later.

For the user whose scope of work is limited to Code and Go foreground processing, no other TSO publications are required. (An exception to this would be the TSO publication that describes the supported TSO terminals for installations that do not provide their own terminal usage instructions.) Note that this publication does not cover Code and Go use in the background or batch environment; this information is provided in the Code and Go and G1 Progammer's Guide, described later.

OS
FORTRAN IV (H EXTENDED) PROGRAMMER'S GUIDE
SC28-6852

This publication is directed to the applications programmer using the FORTRAN IV (H Extended) Compiler. A knowledge of the FORTRAN IV language is assumed, but the reader is not required to have had any prior experience with OS.

In addition to containing information about the use of the FORTRAN IV (H Extended) Compiler, the FORTRAN IV Library (Mod II), and special features such as automatic precision increase, this publication is a primary source of information for that subset of the OS job control language directly relevant for the compilation, link-editing or loading, and execution of a FORTRAN program.

For most users of the FORTRAN IV (H Extended) compiler, this publication and the FORTRAN IV language manual (GC28-6515) are the only publications normally required. Users who require specific detailed information about the mathematical and service subprograms provided in the Mod II library should, additionally, consult the Mathematical and Service Subprogram manual (GC28-6818), and its Mod II supplement (SC28-6864).

OS
CODE AND GO AND FORTRAN IV (G1) PROGRAMMER'S GUIDE
SC28-6853

This publication is for the FORTRAN applications programmer who is using either the Code and Go FORTRAN processor or the FORTRAN IV (G1) processor in conjunction with the FORTRAN IV Library (Mod I), in the OS background or batch environment. The reader of this publication is assumed to know the FORTRAN IV language (as described in the language manual cited earlier) but is not required to have any prior knowledge of OS.

In addition to information about the two processors and the library, this manual is a primary source of information for that subset of the OS Job Control Language and the OS Linkage Editor statements that is directly supported by or directly relevant to the use of either Code and Go or G1, including compilation, linkage editing or loading, and execution.

For those users whose scope of work is limited to Code and Go or G1 processing in the OS background or batch environment, no other OS publications are required. Note that this publication doesn't cover Code and Go or G1 use in the TSO foreground; this information is provided in the appropriate terminal user's guides.

Users who wish to submit Code and Go or G1 jobs through the TSO foreground (i.e., via the TSO SUBMIT command) should use this publication in conjunction with the background command information in the two TSO publications, *Terminal User's Guide,* Order Number GC28-6763, and *Command Language Reference,* Order Number GC28-6732.

OS (TSO)
TERMINAL USER'S SUPPLEMENT FOR FORTRAN IV (G1) PROCESSOR AND TSO FORTRAN PROMPTER
SC28-6855

This publication supplements the two TSO publications, *Terminal User's Guide,* Order Number GC28-6763, and *Command Language Reference,* Order Number GC28-6732 and provides the terminal user with specific information on how to use the TSO FORTRAN Prompter and the FORTRAN IV (G1) processor in the TSO foreground. For installations not equipped with the TSO FORTRAN Prompter, this publication describes the TSO commands necessary to invoke the G1 processor directly. It also describes the terminal I/O facilities available through the FORTRAN IV Library (Mod I).

For information on using the TSO command language and editing facilities, the linkage editor or loader, etc., the user should have available the two TSO publications.

This publication assumes that the installation has established a TSO log-on procedure for the G1 user and has advised him of the procedure's name and its limits; the minimum requirements for this procedure are given in the G1 and Prompter installation reference material manual described below.

Note that this manual doesn't cover G1 use in the background or batch environments; this information is provided in the Code and Go and G1 Programmer's Guide, described above.

FORTRAN INTERACTIVE DEBUG FOR OS (TSO)
CODE AND GO FORTRAN
FORTRAN G1
TERMINAL USER'S GUIDE
SC28-6885

This publication contains all the information necessary to debug a Code and Go or FORTRAN IV (G1) program using FORTRAN Interactive Debug under TSO. The TESTFORT command and its subcommands are explained in detail, and numerous examples are provided throughout the text.

This publication is intended to be used in conjunction with either the Code and Go Terminal User's Guide (SC28-6842) or the Terminal User's Supplement for G1 (SC28-6855), as appropriate. Both these publications are described above.

CMS
TERMINAL USER'S SUPPLEMENT FOR CODE AND GO FORTRAN FORTRAN IV (G1), AND FORTRAN IV (H EXTENDED) PROCESSORS

This publication will supplement the CMS command language user's guide and associated publications to provide specific information on how to use the program product processors in a CMS environment. (The exact title and order number of this Supplement will be announced with the availability of CMS.)

OS
FORTRAN IV (G1) PROCESSOR AND TSO FORTRAN PROMPTER
INSTALLATION REFERENCE MATERIAL
SC28-6856

OS
FORTRAN IV LIBRARY (MOD I)
INSTALLATION REFERENCE MATERIAL
SC28-6858

OS
CODE AND GO FORTRAN PROCESSOR
INSTALLATION REFERENCE MATERIAL
SC28-6859

OS
FORTRAN IV (H EXTENDED) COMPILER AND LIBRARY
(MOD II) INSTALLATION REFERENCE MATERIAL
SC28-6861

OS (TSO)
FORTRAN INTERACTIVE DEBUG
INSTALLATION REFERENCE MATERIAL
SC28-6886

These publications provide the installation with information on how to install and use the program products indicated in the titles. Included are program installation procedures (with system generation dependencies), log-on procedure requirements for TSO foreground use, storage estimates, system programmer considerations, and diagnostic messages.

These publications are essentially supplemental, in that they assume the availability at the installation of other OS system publications pertaining to the installation and use of OS itself (e.g., System Generation, System Programmer's Guide, etc.).

OS
FORTRAN IV (H EXTENDED) COMPILER AND LIBRARY
(MOD II) MESSAGES
SC28-6865

This publication contains the text of all diagnostic messages that may be produced by the FORTRAN IV (H Extended) compiler and FORTRAN IV Library (Mod II) during compilation and execution of a FORTRAN program. Also included is a detailed explanation of each message and suggested corrective action.

OS
FORTRAN IV (H EXTENDED) COMPILER
PROGRAM LOGIC
LY28-6403

OS
FORTRAN IV LIBRARY (MOD II)
PROGRAM LOGIC
LY28-6409

OS
CODE AND GO FORTRAN PROCESSOR
PROGRAM LOGIC
LY28-6846

OS
FORTRAN IV (G1) PROCESSOR
PROGRAM LOGIC
LY28-6856

OS
FORTRAN IV LIBRARY (MOD I)
PROGRAM LOGIC
LY28-6408

OS (TSO)
TSO FORTRAN PROMPTER
PROGRAM LOGIC
LY28-6410

These publications describe the internal structure, method of operation, and implementation of the program products in the titles. They are meant to be used in conjunction with the program listing for purposes of program maintenance.

Program logic manuals are not necessary for the installation and use of the product; they are provided for use by program maintenance personnel. Program logic manuals are available to licensees only.

PUBLICATIONS SELECTION GUIDE

Table 8 lists the publications described in the preceding text, indicating the intended audience of each and the product to which each applies. In the table, publication titles have been abbreviated; full titles and order numbers are given in the preceding descriptions.

AUDIENCE / PRODUCT	ALL LANGUAGE USERS	TERMINAL USERS		BACKGROUND (BATCH) USERS	INSTALLATION PERSONNEL	MAINTENANCE PERSONNEL
		TSO	CMS			
Code and Go FORTRAN	FORTRAN IV Language	Code and Go Terminal User's Guide	Terminal User's Supplement for Code and Go, G1 and H Extended	Code and Go and G1 Programmer's Guide (See Note 1)	Code and Go Installation Reference Material	Code and Go Logic
FORTRAN IV (G1)		G1 and TSO Prompter Supplement (See Note 2)			G1 and TSO Prompter Installation Reference Material	FORTRAN IV (G1) Logic
FORTRAN IV (H Extended)		—		(1) H Extended Programmer's Guide (2) H Extended Messages	H Extended and Library (Mod II) Installation Reference Material	H Extended Logic
TSO Prompter	—	G1 and TSO Prompter Supplement (See Note 2)	—	—	G1 and TSO Prompter Installation Reference Material	Prompter Logic
FORTRAN Interactive Debug	—	Interactive Debug Terminal User's Guide		—	Interactive Debug Installation Reference Material	—
FORTRAN IV Library (Mod I)	(See Note 3)	(See Note 4)			Library (Mod I) Installation Reference Material	Library (Mod I) Logic
FORTRAN IV Library (Mod II)					H Extended and Library (Mod II) Installation Reference Material	Library (Mod II) Logic

1. Users who are going to submit jobs for background processing through the TSO foreground should have other terminal user's publications available for use in conjunction with this publication (see description).
2. This publication assumes the availability of other terminal user publications (see description).
3. Although not essential for writing FORTRAN programs or for using the facilities of the FORTRAN libraries, the following publications are available for users who require comprehensive library subprogram information: *OS FORTRAN IV Mathematical and Service Subprograms,* Order No. GC28-6818, and *OS FORTRAN IV Mathematical and Service Subprogram Supplement for Mod I and Mod II Libraries,* Order No. SC28-6864 (see descriptions).
4. I/O and error handling facilities for the libraries are described in the user's guides for the program product processors.

INDEX