60483900

CONTROL DATA CORPORATION

FORTRAN VERSION 5 INSTANT

CDC® OPERATING SYSTEMS:
NOS 1
NOS/BE 1
SCOPE 2

CONTROL DATA CORPORATION

FORTRAN VERSION 5 INSTANT

CDC® OPERATING SYSTEMS:
NOS 1
NOS/BE 1
SCOPE 2

REVISION RECORD	
REVISION	DESCRIPTION
A	Original Release.
(01-15-81)	
Publication No. 60483900	

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
215 MOFFETT PARK DRIVE
SUNNYVALE, CALIFORNIA 94086

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

Page	Rev
Front Cover	-
Title Page	-
ii	A
iii/iv	A
v thru ix	A
1 thru 66	A
Back Cover	-

PREFACE

This instant outlines the FORTRAN Version 5 language. FORTRAN Version 5 complies with the American National Standards Institute FORTRAN language described in document X3.9-1978 and known as FORTRAN 77. FORTRAN Version 5 extensions to FORTRAN 77 are indicated by shading. Detailed information is contained in the FORTRAN Version 5 reference manual.

The FORTRAN 5 compiler operates in conjunction with the COMPASS 3 assembly language processor under control of:

- NOS 1 for the CONTROL DATA® CYBER 170 Series; CYBER 70 Models 71, 72, 73, and 74; and 6000 Series Computer Systems
- NOS/BE 1 for the CDC® CYBER 170 Series; CYBER 70 Models 71, 72, 73, and 74; and 6000 Series Computer Systems
- SCOPE 2 for CONTROL DATA CYBER 170 Model 176, CYBER 70 Model 76, and 7600 Computer Systems.

This instant provides a brief description of the major FORTRAN 5 language features. It is intended for programmers familiar with FORTRAN 5.

Detailed information can be found in the listed publications. The publications are listed alphabetically within groupings that indicate relative importance to readers of this manual.

The NOS manual abstracts and the NOS/BE manual abstracts are instant-sized manuals containing brief descriptions of the contents and intended audience of all NOS and NOS product set manuals, and NOS/BE and NOS/BE product set manuals, respectively. The abstracts manuals can be useful in determining which revision level of software documentation corresponds to the Programming Systems Report (PSR) level of installed site software.

The following publications are of primary interest:

Publication	Publication Number
FORTRAN Version 5 Reference Manual	60481300
NOS Version 1 Reference Manual Volume 1 of 2	60435400

NOS/BE Version 1 Reference Manual	60493800
SCOPE Version 2 Reference Manual	60497800

The following publications are of secondary interest:

Publication	Publication Number
CYBER Interactive Debug Version 1 Reference Manual	60481400
CYBER Record Manager Advanced Access Methods Version 2 Reference Manual	60499300
CYBER Record Manager Basic Access Methods Version 1.5 Reference Manual	60495700
NOS Version 1 Manual Abstracts	84000420
NOS/BE Version 1 Manual Abstracts	84000470
Software Publications Release History	60481000
Sort/Merge Version 4 and 1 Reference Manual	60497500

CDC manuals can be ordered from Control Data Corporation, Literature and Distribution Services, 308 North Dale Street, St. Paul, Minnesota 55103.

CONTENTS

FIGURES

TABLES

NOTATIONS

Certain notations are used throughout the manual with consistent meaning. These notations are:

UPPERCASE	Uppercase words in language syntax indicate statement keywords or symbols to be written as shown.
lowercase	Lowercase words in language syntax indicate names, numbers, symbols, or entities to be supplied by the programmer.
[]	Brackets in language syntax enclose optional items that can be used or omitted.
{}	Braces in language syntax indicate that only one of the vertically stacked items can be used.
...	Ellipsis in language syntax indicates that the preceding optional item in brackets can be repeated as necessary.
Δ	Delta in language syntax indicates a blank character.
Shading	Shading in language syntax, language descriptions, and program examples, indicates extensions to FORTRAN 77.

LANGUAGE ELEMENTS

This section describes the language elements of FORTRAN 5.

SYMBOLIC NAMES

Symbolic names are 1-6 or 7 letters and digits; the first character must be a letter.

FORTRAN CHARACTER SET

The FORTRAN character set is composed of the following symbols:

Alphabetic: A to Z

Numeric: 0 to 9

Special:

=	equal	,	comma
+	plus	:	colon
-	minus	'	apostrophe (CDC graphic ↑)
*	asterisk	.	decimal point
/	slash	$	currency symbol
(	left parenthesis		blank
)	right parenthesis	"	quote (CDC graphic ≠)

Any character acceptable to the operating system can be used in Hollerith or character information and comments. Blanks are significant only in character and Hollerith constants and character and Hollerith edit descriptors.

FORTRAN STATEMENTS

Column	Contents
1	C or * indicates comment line.
1-2	C$ indicates compiler directives.
1-5	Statement label (any 1- to 5-digit positive nonzero integer).
6	Any character other than blank or zero denotes continuation (columns 1 through 5 must be blank).

7-72	FORTRAN statement.
73-80	Identification field; not processed by the compiler, but printed with the source listing.

Statements are written in columns 7 through 72; blanks are ignored except in Hollerith or character fields. All 80 columns can be used for data input. Statements can be labeled by an integer constant in the range 1 through 99999.

PRINTER CONTROL CHARACTERS

Character	Action
Blank	Space vertically one line, then print.
0	Space vertically two lines, then print.
1	Eject to first line of next page before printing.
+	Do not advance before printing; allows overprinting.
Any other character	Refer to operating system reference manual.

CONSTANTS

A constant is a fixed quantity. Table 1 lists the types of constants and gives a brief description of each.

VARIABLES

A variable is a quantity that can be changed. It is composed of 1-6 or 7 letters or digits; the first character must be a letter.

DEFAULT IMPLICIT TYPING

A variable not defined in a type declaration is:

A-H,O-Z	Real
I-N	Integer

Table 2 lists the types of variables and gives a brief description of each.

ARRAYS

A FORTRAN array is a set of elements identified by a single name. The name is composed of 1 through 6 or 7 letters and digits and begins with a letter. The array name and its dimensions must be declared in a DIMENSION, COMMON, or type statement. Arrays can have one to seven dimensions. The declaration of array dimensions takes the following form:

```
array(d[,d]...)
```

array — Is the symbolic name of the array.

d — Specifies the bounds of an array dimension and takes the form:

```
[lower:]upper
```

lower — Specifies the lower bound of the dimension. The lower bound can be an integer expression with a positive, zero, or negative value. If omitted, the lower bound is assumed to be 1.

upper — Specifies the upper bound of the dimension. The upper bound can be an integer expression with a positive, zero, or negative value. The upper bound must be greater than or equal to the lower bound.

Example:

```
REAL
TABLE(4,3)
INTEGER X(-10:10)
```

References to complete arrays or to specific array elements can be made. A reference to a complete array is to the array name. A reference to a specific element involves the array name followed by a subscript specification. The format for array element references is as follows:

```
array(e[,e]...)
```

TABLE 1. LIST OF CONSTANTS

Constants	Format	Examples
Integer	[+] d [d] . . . d Is a decimal digit, 1 to 18 digits without decimal point. Range $-(2^{59}-1)$ to $2^{59}-1$ An integer and the results of addition and subtraction. $-(2^{48}-1)$ to $2^{48}-1$ Integers used in and the results of multiplication, division, and exponentiation. $-(2^{17}-2)$ to $2^{17}-2$ Integers used as DO indexes or as subscripts except: when DO=LONG is selected or a DO(LONG=1) directive is in effect, a DO index can exceed $2^{M}-1$. When LCM=G is selected, the range for subscripts is $-(2^{20}-8)$ to $2^{20}-8$.	2 247

Real	[+] coeff [+]coef E [+]exp [+]n E[+]exp coeff Is a coefficient in the form of a real or integer constant: n. , n.n, .n, n n Is an unsigned integer constant. exp Is an unsigned integer exponent (base 10). Range 10^{-293} to 10^{+322}. Accurate to approximately 14 decimal digits.	7.5 3.22 42.E1 314.E05 -14.5 700.E-2 0.5 0.
Double-Precision	[+] coeff D [+] exp , [+] n D [+]exp coeff Is a coefficient in the form of a real or integer constant: n., n., .n, n n Is an unsigned integer constant. exp Is an unsigned integer exponent(base 10). Range 10^{-293} to 10^{+322}. Accurate to approximately 29 decimal digits.	5.834D2 7.D2 9.2D03 14.D-5 3120D4 -1.D0

TABLE 1. LIST OF CONSTANTS (Contd)

Constants	Format	Examples
Complex	(real, imag) real — Is a real or integer constant for the real part. imag — Is a real or integer constant for the imaginary part. Range 10^{-293} to 10^{+322}.	(1.,7.54) (-2.1E1,3.24) imag (4.0.5.0)
Octal	0"o" o — Is a string of 1 to 20 octal digits.	0"77" 0"57" 0"234" 0"124" 0"156"
Hollerith	nHs L"s" R"s" "s" nHs — Indicates left-justified with blank fill. L"s" — Indicates right-justified with binary zero fill.	3HBAT L"BAT" R"BAT"

	R"s" Indicates right-justified with binary zero fill. n Is an unsigned nonzero integer constant in the range 1 to 10. s Is a string of 1 to 10 characters.	
Logical	.TRUE. .FALSE. .TRUE. Represents the logical value true. .FALSE. Represents the logical value false.	LOGICAL X1,Z2 X1=.TRUE Z2=.FALSE.
Hexadecimal	Z"z" z Is a string of 1 to 15 hexadecimal digits.	Z"1A" Z"0E" Z"235"
Character	's' s Is a string of characters enclosed by apostrophes. Range 1 to 32767 characters. Blanks are significant.	'CAT' '278' 'CAN"T'

TABLE 2. LIST OF VARIABLES

Variables	Description	Examples
Integer	Range $-(2^{59}-1)$ to $2^{59}-1$. As the subscript or index of a DO statement, maximum value is $2^{17}-2$. As a result of multiplication or division or conversion between real and integer, maximum value is $2^{48}-1$.	INTEGER X,ITEM,JSUM
Real	Range 10^{-293} to 10^{+322}. Contains approximately 14 significant digits.	REAL I,AVAR,TUF
Double-Precision	Range 10^{-293} to 10^{+322}. Must be defined explicitly or implicitly in type declaration. Contains approximately 29 significant digits. Occupies two storage words.	DOUBLE PRECISION OMEGA,X,B DOUBLE PRECISION,X,Y
Logical	Must be defined explicitly or implicitly in type declaration.	LOGICAL L3,C LOGICAL L2,R
Character	Must be defined explicitly or implicitly in type declaration. The length of the character variable is specified when the variable is typed as character.	CHARACTER JAM*15,C3*3
Boolean	Must be defined explicitly or implicitly as Boolean. A Boolean variable occupies one storage word. Hollerith, octal, or hexadecimal values are generally assigned to Boolean variable.	BOOLEAN HVAL,ZZZ,R34

array — Is the symbolic name of the array.

e — Is a subscript expression that is an integer, real, double-precision, complex, or Boolean expression.

Example:

```
X=DZ(12)
```

CHARACTER SUBSTRINGS

Specific parts of a character string can be defined or referenced with character substring references. A character substring reference has the following format:

char([first]:[last])

char — Is the name of a character variable or an array element.

first — Specifies an integer, real, double-precision, Boolean, or complex expression for the position of the first character of the substring. If first is omitted, the value is one.

last — Specifies an integer, real, double-precision, Boolean, or complex expression for the position of the last character in the substring. If last is omitted, the value is the length of the string.

Example:

```
CHARACTER SIX6,S2*3,S3*4
DATA S1/'STRING'/
.
.
.
S2=S(1:3)
S3=S(3:)
```

In this example, S2 has the value 'STR' and S3 has the value 'RING'.

STATEMENT FORMS

The following symbols are used in the description of statements:

v	Variable, array element, or substring
sl	Statement label
iv	Integer variable
name	Symbolic name
u	Input/output unit: 1- to 3-digit decimal integer constant; integer variable with value of: 0 through 999 or a Hollerith value denoting the file name, left-justified with zero fill
fn	Format designator, statement label, or array name
ios	Integer variable into which one of the following values is placed after the input/output operation is complete: ≤0 End-of-file =0 Operation completed normally >0 Number of error conditions detected
n	String of 1 to 5 digits or character constants
con	Constant, constant expression, or extended constant expression
c...c	String of 1 to 70 characters
eam	Arithmetic or masking expression
erl	Relational or logical expression
ce	Character expression
a	Variable or array name
iolist	Input/output list specifying items to be transmitted
recn	Record number

ASSIGNMENT STATEMENTS

An assignment statement assigns a value to a variable. Table 3 lists the format and an example for each of the assignment statements.

TABLE 3. LIST OF ASSIGNMENT STATEMENTS

Type	Format	Example
Arithmetic	v = eam	A = B + C
Logical	v = erl	LOGICAL L,M,N L = M .AND. N M = B .EQ. C
Character	v = ce	CHARACTER A*2,B*4 A = B
Multiple	v = [v=] ... expression	X = Y = Z = (10. +B)
Boolean	v = expression (arithmetic or Boolean)	A = C*D

CONTROL STATEMENTS

The control statements provide a means of altering, interrupting, terminating, or modifying the normal flow of a program. Table 4 lists the type, the format, and an example for each of the control statements.

SPECIFICATION STATEMENTS

Specification statements are nonexecutable and are used to specify the characteristics of symbolic names used in the program. (See table 5.)

PROGRAM UNITS AND PROCEDURES

A program unit is a group of FORTRAN statements with optional comments, terminated by an END statement. A procedure can be a function subprogram, a subroutine subprogram, or a statement function. (See table 6.)

INPUT/OUTPUT

Input and output involve reading records from files, and writing records to files. (See table 7.)

EXTENDED INTERNAL FILES

The internal transfer of data is accomplished through the ENCODE and DECODE statements. ENCODE is the extended internal file output statement and DECODE is the extended internal file input statement. The formats are:

```
DECODE(c,fn,v) iolist

Example:

    DECODE(77,17,CARD)INK


ENCODE(c,fn,v)iolist

Example:

    ENCODE(40,1,ALPHA)A,B,C
```

TABLE 4. LIST OF PROGRAM CONTROL STATEMENTS

Type	Format	Example
Unconditional GO TO	GO TO sl	GO TO 30
Computed GO TO	GO TO(sl [,sl] ...) [,] eam	GO TO(1,2,9),A + B GO TO(1,2,9) A + B
Assigned GO TO	GO TO iv [[,] (sl [,sl] ...)]	GO TO LUB, (1,2,3)
ASSIGN	ASSIGN sl TO iv	ASSIGN 1 TO LUB
Arithmetic IF	IF(eam)sl_1,sl_2,sl_3	IF (I-N) 3,4,6
Logical IF	IF(erl) statement	IF (P.AND.Q)RES = 7.2
Block IF	IF(erl) THEN if-block END IF	IF(I.EQ.0)THEN X = X + DX END IF
ELSE IF (never stands alone)	ELSE IF (erl) THEN	IF(N.EQ.1)THEN CALL ASUB(X,R) CALL BSUB(X,S)

TABLE 4. LIST OF PROGRAM CONTROL STATEMENTS (Contd)

Type	Format	Example
ELSE (never stands alone)	ELSE	ELSE IF(N.EQ.2)THEN DO 6 I = 1,1000 6 X(I) = 0.0

END IF
(never stands alone)
	END IF	ELSE IF(N.EQ.3) THEN
		X = 3.1
		ELSE
		X = 0.0
		END IF

DO	DO sl [,] iv = m_1,m_2[,m_3] m_1 initial parameter m_2 terminal parameter m_3 increment parameter (optional)	DO 10 I = 1,20,2 DO 20 J = 1,10
CONTINUE	CONTINUE	100 CONTINUE
PAUSE	PAUSE [n]	PAUSE 'MOUNT TAPE'
STOP	STOP [n]	STOP 25
END	END	END

TABLE 5. LIST OF SPECIFICATION STATEMENTS

Description	Example
INTEGER name [,name] ...	INTEGER A,B,C(10)
REAL name [,name] ...	REAL NEXT,X(5)
COMPLEX name [,name] ...	COMPLEX CC,J
DOUBLE PRECISION name [,name] ...	DOUBLE PRECISION DP1, DP2
LOGICAL name [,name] ...	LOGICAL L1,L2
BOOLEAN name [,name] ...	BOOLEAN ALABEL, Q MASK
CHARACTER * len name [,name] len Specifies the length and can be: An unsigned nonzero integer constant. An integer constant expression, enclosed in parenthesis. or (*)	CHARACTER*3 CC,A(4)

TABLE 5. LIST OF SPECIFICATION STATEMENTS (Contd)

Description	Example
IMPLICIT type(ac [,ac] ...)[,type(ac [,ac] ...)] ...	IMPLICIT REAL (I-L),CHARACTER*20(M,X-Z)
type — Is INTEGER, CHARACTER [*len] , REAL, BOOLEAN, COMPLEX, DOUBLE PRECISION or LOGICAL.	
ac — Is a single letter, or range of letters represented by the first and last letter separated by a hyphen, indicating which letters imply which type.	
DIMENSION array(d [,d] ...[,array(d [,d] ...)] ...	DIMENSION BETA(2,3)
array — Is an array name.	
d — Specifies the bounds of a dimension; upper or lower:upper.	
PARAMETER (name=con [,name=con]...)	PARAMETER (ITER=20,START=5.3)

COMMON [/ [cb] /] nlist[[,] / [cb] /nlist] ... cb — Is a common block name identifying a named common block containing the entities in nlist. nlist — Is a list of entities to be included in the common block.	COMMON /ST/C,D,E
EQUIVALENCE (nlist) [,(nlist)] ... nlist — Is a list of variable names, array names, array element names, or character substring names.	EQUIVALENCE(C),(A,D(1))
LEVEL 1,name [,name] ... 1 — Is an unsigned integer constant or symbolic constant, with the value 0, 1, 2, or 3 indicating the storage level. name — Is either a common block designator of the form /(cb)/ or a dummy argument name.	LEVEL 3,/ECSBLK/
SAVE [a [,a] ...] a — Is a variable name, array name, or common block name enclosed in slashes.	SAVE /X/ SAVE/(cb)/

TABLE 5. LIST OF SPECIFICATION STATEMENTS (Contd)

Description	Example
INTRINSIC fun [,fun] ... fun — Is an intrinsic function name.	INTRINSIC SIN,COS
DATA nlist/clist/[[,] nlist/clist/] ... nlist — Is a list of names to be initially defined. Each name in the list can take the following forms: variable, array, array element, substring, or implied-DO. clist — Is a list of constants or symbolic constants specifying the initial values. Each item in the list can take the form: c, r*c, r(c[,c] ...).	DATA JR/4/ DATA K/1,2,3,4/ DATA AT/5.0/,AQ/7.5/
EXTERNAL proc [,proc] ... proc — Is the name of an external procedure dummy procedure, or block data subprogram.	EXTERNAL SQRT

TABLE 6. LIST OF PROGRAM UNITS AND PROCEDURES

Name	Format	Example
Block data Subprogram	BLOCK DATA [name]	BLOCK DATA ANAME
Program Statement	PROGRAM name [(fpar [,fpar] ...)] fpar Is the file name in one of the following forms: file, file=n, file=/r,file=n/r, altunit=file. See the FORTRAN 5 reference manual for the definition of file names.	PROGRAM A (INPUT,OUTPUT)
Subroutine Subprogram	SUBROUTINE name [([d [,d]...])] name Is the name of the subroutine subprogram. d Is a dummy argument; a name or an asterisk.	SUBROUTINE ERROR2 SUBROUTINE AY(*,*)

TABLE 6. LIST OF PROGRAM UNITS AND PROCEDURES (Contd)

Name	Format	Example
Function Subprogram	[type] FUNCTION name ([d [,d]...]) type Is INTEGER, REAL, DOUBLE PRECISION, BOOLEAN, COMPLEX, LOGICAL, or CHARACTER* len. name Is the name of the function subprogram. d Is a dummy argument name.	FUNCTION JOR (X,Y)
Entry Statement	ENTRY ep[([d [,d]...])] ep Is an entry point in a function or subroutine. d Is a dummy argument name (can also be an asterisk in a subroutine).	ENTRY A(X,Y,Z)

Statement Function	fun([d [,d]...]) = expr fun Is the function name. expr Is an expression. d Is a dummy argument name.	ADD(A,B,C,D) = A+B+C+D
Subroutine Call	CALL sub[([a [,a]...])] sub Is the name of subroutine or dummy procedure. a Is an actual argument.	CALL VAL (A,B,C)
RETURN Statement	RETURN [eam] eam Is an arithmetic expression.	RETURN

TABLE 7. LIST OF INPUT/OUTPUT STATEMENTS

Type	Description	Example
Formatted	PRINT fn [,iolist]	PRINT 4,A,B,N
	PUNCH fn [,iolist]	PUNCH 5,A,B,C
	READ fn [,iolist]	READ 10,A,B,C
	READ ([UNIT=]u, [FMT=] fn [,IOSTAT=ios] [,ERR=sl][,END=sl])[iolist]	READ (2,ERR=16,END=18) A,B
	WRITE ([UNIT=]u, [FMT=] fn [,IOSTAT=ios][,ERR=sl])[iolist]	WRITE (6,100,ERR=200) X,Y,M
	READ† ([UNIT=]u, [FMT=] fn [,IOSTAT=ios] [,ERR=sl] [,REC=recn]) [iolist]	READ (2,99,REC=I,ERR=20)(A(J)),J=1,6)
	WRITE† ([UNIT=]u, [FMT=] fn [,IOSTAT=ios] [,ERR=sl] [,REC=recn]) [iolist]	WRITE (5,FMT=10,REC=1)A,B,IA,C,IB,E,D,ID,F

Unformatted	READ ([UNIT=]u [,IOSTAT=ios] [,ERR=sl]) [,END=sl])[iolist] WRITE ([UNIT=]u [,IOSTAT=ios] [,ERR=sl] [iolist]) READ† ([UNIT=]u, [,IOSTAT=ios] [,ERR=sl] [,REC=recn])[iolist] WRITE† ([UNIT=]u, [,IOSTAT=ios] [,ERR=sl] [,REC=rcn])[iolist]	READ(2,END=40,ERR=50)X,Y,Z WRITE(10,ER=16)A,B READ (2,REC=I,ERR=30)A,B,C WRITE (5,REC=1,ERR=5)I,J,K
Buffer	BUFFER IN (u,p)(a,b) BUFFER OUT (u,p)(a,b) a,b Are the first and last words of the data block to be transferred. p Is an integer constant or integer variable: 0 even parity 1 odd parity	BUFFER IN (1,1)(R(1),R(5,12)) BUFFER OUT(1,J)(B(M),B(N))

TABLE 7. LIST OF INPUT/OUTPUT STATEMENTS(Contd)

Type	Description	Example
Status Statements	CLOSE ([UNIT=]u [,IOSTAT=ios] [,ERR=sl] [,STATUS=sta]) sta Is a character expression that determines the disposition of the file associated with the specified unit. Valid values are: 'KEEP','DELETE'. OPEN ([UNIT=]u [,IOSTAT=ios] [,ERR=sl] [,FILE=fin] [,STATUS=sta] [,ACCESS=acc][,FORM=fm] [,RECL=rl] [,BLANK=blnk] [,BUFL=bl]) sta Is a character expression specifying file status. Valid values are: 'OLD', 'NEW', 'SCRATCH', 'UNKNOWN'.	CLOSE (2,ERR=25,STATUS='DELETE') OPEN(2,STATUS='NEW',ERR=12, FILE='NEWFL',*ACCESS='SEQUENTIAL')

fin	Is a character expression whose value is the name of the file to be opened.
acc	Is a character expression specifying the access method of the file. Valid values are: 'SEQUENTIAL', 'DIRECT'.
rl	Is an integer variable or positive integer constant specifying the record length for a direct access file.
blnk	Is a character expression having one of the following values: 'NULL', 'ZERO'.
fm	Is a character expression having one of the following values: 'FORMATTED', 'UNFORMATTED'.
bl	Is a nonnegative integer expression specifying the buffer length.

TABLE 7. LIST OF INPUT/OUTPUT STATEMENTS(Contd)

Type	Description	Example
	INQUIRE({[UNIT=]u / FILE=fin} [,IOSTAT=ios] [,ERR=sl] [EXIST=ex] [,OPENED=od] [,NUMBER=num] [NAMED=nmd] [,NAME=fn] [,ACCESS=acc] [,SEQUENTIAL=seq] [,DIRECT=dir] [,FORM=fm] [,FORMATTED=fmt] [,UNFORMATTED=unf] [,RECL=rcl] [,NEXTREC=nr] [,BLANK=blnk]) fin Is a character expression specifying the name of the file for which information is to be returned. ex Is a logical variable: .TRUE. (file or unit exists) or .FALSE. (file or unit does not exist).	INQUIRE(FILE='AFILE',ERR=100, EXIST=EX, ACCESS='SEQUENTIAL')

od — Is a logical variable:
.TRUE. (file or unit is connected to a unit or file)
or
.FALSE. (file or unit is not connected to a unit or file).

num — Is an integer variable containing the external unit number of the unit currently associated with the file.

nmd — Is a logical variable:
.TRUE. (file has a name) or
.FALSE. (file does not have a name).

fn — Is a character variable containing the name of the file associated with unit u.

acc — Is a character variable indicating the access method of the file: 'SEQUENTIAL', 'DIRECT'.

TABLE 7. LIST OF INPUT/OUTPUT STATEMENTS(Contd)

Type	Description	Example
	seq Is a character variable indicating whether the file can be opened for sequential input/output: 'YES', 'NO', 'UNKNOWN'. dir Is a character variable indicating whether the file can be opened for direct access input/output: 'YES', 'NO', 'UNKNOWN'. fm Is a character variable indicating formatted input/output: 'FORMATTED', 'UNFORMATTED'. fmt Is a character variable specifying whether the file can be opened for formatted input/output: 'YES', 'NO', 'UNKNOWN'.	

	unf Is a character variable specifying whether the file can be opened for unformatted input/output: 'YES', 'NO', 'UNKNOWN'. rcl Is an integer variable containing the record length of a file opened for direct access. nr Is an integer variable indicating the record number of the next record to be read or written to a direct access file. Undefined for sequential files. blnk Is a character variable: return values can be 'NULL' or 'ZERO'.	
List Directed	PRINT* [,iolist]	PRINT*,Z,Q,B
	PUNCH* [,iolist]	PUNCH*,A,B,C
	READ ([UNIT=] u, [FMT=] * [,IOSTAT=ios] [,ERR=sl] [,END=sl] [iolist]	READ(*,*,END=99) J,K
	READ* [,iolist]	READ*,CAT,BIRD,DOG

TABLE 7. LIST OF INPUT/OUTPUT STATEMENTS(Contd)

Type	Description	Example
	WRITE ([UNIT=] u, [FMT=] * [,IOSTAT=ios] [,ERR=sl]) [iolist]	WRITE(6,*,ERR=20) C,F
Namelist	NAMELIST/name/a[,a] ... [/name/a[,a] ...]...	NAMELIST/SHIP/I1,I2,A,B
	PRINT name	PRINT SHIP
	PUNCH name	PUNCH SHIP
	READ ([UNIT=] u, [FMT=] name [,IOSTAT=ios] [,ERR=sl] [,END=sl])	READ(*,SHIP,END=10)
	READ name	READ SHIP
	WRITE ([UNIT=] u, [FMT=] name [,IOSTAT=ios] [,ERR=sl])	WRITE(6,SHIP,ERR=30)

†Direct Access

FILE POSITIONING STATEMENTS

File positioning statements are used to position files connected for sequential access. Table 8 lists the name, the format, and an example of each statement.

TABLE 8. LIST OF FILE POSITIONING STATEMENTS

Format	Example
REWIND([UNIT=] u [,IOSTAT=ios] [,ERR=sl])	REWIND(3,ERR=40)
BACKSPACE([UNIT=] u [,IOSTAT=ios] [,ERR=sl])	BACKSPACE(LUN,ERR=50)
ENDFILE([UNIT=] u, [IOSTAT=ios] [,ERR=sl])	ENDFILE(UNIT=IOUT,ERR=100)

FORMAT SPECIFICATION

Format specifications are used to produce output or read input that consists of strings of display coded characters:

sl FORMAT(flist)

flist Is a list of items, separated by commas.

Example:

```
100 FORMAT(I6,F7.3, 214)
```

DATA CONVERSION

The data conversion to be performed is specified by format specifications. Format specifications are composed of repeatable and nonrepeatable edit descriptors. Tables 9 and 10 summarize the edit descriptors.

TABLE 9. LIST OF REPEATABLE EDIT DESCRIPTORS

Format	Definition	Example
srEw.d	Single-precision floating-point with exponent.	2E13.3
srEw.dEe	Floating-point with specified exponent length.	E10.2E1
srFw.d	Single-precision floating-point without exponent.	F7.3
srGw.d	Single-precision floating-point with or without exponent.	G14.6
srGw.dEe	Single-precision floating-point with or without explicitly specified exponent length.	G12.4E2
srDw.d	Double-precision floating-point with exponent.	2D10.4
rIw	Decimal integer conversion.	4I9
rIw.m	Integer with specified minimum digits.	I6.2
rLw	Logical conversion.	2L5
rA	Character with data dependent length.	A
rAW	Character conversion with specified length.	A7

rRw	Rightmost characters with binary zero fill.	4R10
rOw	Octal integer conversion.	O5
rOw.m	Octal integer with specified minimum digits.	O24.16
rZw	Hexadecimal conversion.	Z8
rZw.m	Hexadecimal with leading zeros and minimum number of digits.	Z5.3

Note:

s Optional scale factor of the form kP

r Optional repetition factor

w Integer constant indicating the field width

d Integer constant indicating digits to the right of the decimal point

e Integer indicating digits in the exponent field

m Integer specifying minimum number of digits

TABLE 10. LIST OF NONREPEATABLE EDIT DESCRIPTORS

Format	Definition	Example
BN	Blanks ignored.	READ(6,'(I3,BZ,I3,BN,I3)')I,J,K
BZ	Blanks treated as zeros.	
SP	Plus signs (+) produced.	
SS	Plus signs (+) suppressed.	WRITE(2,'(1X,F6.2,SP,F6.2,S5,F6.2)')A,B,C
S	Plus signs (+) suppressed.	
nX	Position forward n characters.	10 FORMAT(6X,F7.2)
Tn	Position to column n.	
TRn	Position forward n characters.	40 FORMAT(T2,F5.2,TR5,F6.2)
TLn	Position backward n characters.	READ(2,'(F5.2,TL5,F5.2)')A,B
nH	Hollerith.	5 FORMAT(4X,7HHEADING)
	"..." Output character string.	5 FORMAT(4X,"HEADING")
	'...' Output character string.	5 FORMAT(4X,'HEADING')
:	Terminate format control.	30 FORMAT (4(F4.1,:,','))
/	End of current input or output record.	50 FORMAT(2X,4HDOGS///1X,3HCAT)
kP	Scaling for numeric editing.	15 FORMAT(2P,E14.3)

OVERLAYS

Overlays reduce the amount of storage required and make efficient use of the available field length. An overlay is identified by the OVERLAY statement and is called by the OVERLAY call:

```
CALL OVERLAY(fname,i,j[,recall[,k]])
```

fname — Is either the file name of the file containing the (i,j) overlay to be executed (if k is absent or zero), or the overlay name (if k is nonzero).

i,j — Are the overlay level numbers and can be an integer expression.

recall — Is the recall parameter and can be an arithmetic expression with the value 6HRECALL, or a character expression with the value 'RECALL'.

k — Is an indicator affecting the interpretation of fname.

Example:

```
CALL OVERLAY(3HSUB,1,0,6HRECALL)
```

The OVERLAY statement is:

```
OVERLAY([fname,]i,j[,orig] [,OV=n])
```

fname — Is the name of the file in which the generated overlay is to be written.

i,j — Are the overlay level numbers in octal (0 through 77).

orig — Is an optional parameter specifying the origin of the overlay; not allowed for (0,0) overlay. The loader accepts any of the following forms: Cnnnnnn, 0=nnnnnn, 0=ept, 0=ept+nnnnnn.

OV=n — Is an optional parameter specifying that the overlay generator is to generate an overlay structure suitable for Fast Overlay Loader(FOL).

Example:

```
OVERLAY(XFILE,0,0,OV=2)
```

DEBUGGING

Two debugging aids are available to help the user find execution-time errors in a FORTRAN program. They are CYBER Interactive Debug (CID) and Post Mortem Dump (PMD).

CYBER Interactive Debug is a supervisory program that allows the user to monitor and control the execution of a FORTRAN program from a terminal. To use all of the capabilities of CID, the user must specify the DEBUG control statement prior to compilation or the FTN5 control statement must specify the DB=ID parameter. For further details on CID, see the CYBER Interactive Debug Version 1 reference manual.

Post Mortem Dump is a program that analyzes the cause of execution-time errors in FORTRAN programs. To use PMD, the DB=PMD parameter must be specified on the FTN5 control statement. PMD is invoked by hardware/software errors and can also be invoked intentionally by the user.

PMD has the following optional calls:

- CALL PMDARRY(i,j,k,l,m,n,o)

  Causes dumps of arrays to be limited to elements where subscripts do not exceed i,j,k,l,m,n, and o for their respective dimensions; the integers i through o represent the first through seventh dimensions respectively.

  Example:

  ```
  CALL PMDARRY(3,4,1)
  ```

- CALL PMDDUMP

  Causes a dump of variables in the calling routine, not at once, but when an abort occurs or when PMDLOAD or PMDSTOP is called. PMDDUMP and PMDLOAD or PMDSTOP need not be called from the same routine.

  Example:

  ```
  CALL PMDDUMP
  ```

- CALL PMDLOAD

  Causes an immediate dump of variables in the calling routine and in any routines that have called PMDDUMP.

Example:

```
CALL PMDLOAD
```

- CALL PMDSTOP

  Causes an immediate dump of variables in the calling routine, all routines in the trace-back chain, and any routines that have called PMDDUMP.

  Example:

```
CALL PMDSTOP
```

See the FORTRAN 5 reference manual for further details on PMD.

C$ DIRECTIVES

A C$ directive is a special form of comment line that controls compiler processing.

LISTING CONTROL

A listing control directive modifies the state of list option switches that have been initially enabled on the FTN5 control statement. It has the following format:

```
C$ LIST(p[=c] [,p[=c]]...)
```

p	Is S, O, R, A, M, or ALL.
S	Source lines
O	Object code
R	Symbol references
A	Symbol attribute list
M	Symbol map list
ALL	Equivalent to S, O, R, A, M

Examples:

```
C$  LIST(S=O)
C$  LIST(A)
C$  LIST(ALL)
```

CONDITIONAL COMPILATION

A conditional compilation directive controls whether the lines immediately following the directive are to be processed or ignored by the compiler. The formats of the categories of the conditional compilation directives are:

```
C$ IF(e)[[,]lab]
```

e Is a logical constant expression

lab Is a label. It must be a symbolic name.

Example:

```
C$ IF(M .EQ. 0)
```

```
C$ ELSE [,lab]
```

lab Is a label. It must be a symbolic name.

Example:

```
C$ ELSE
```

```
C$ ENDIF [,lab]
```

lab Is a label. It must be a symbolic name.

Example:

```
C$ ENDIF
```

COLLATION CONTROL

A collation control directive directs the interpretation of character relational expressions in the lines following the directive and preceding either another collation control directive or the END statement of the program unit. It has the following format:

```
C$ COLLATE(p)
```

p Is FIXED or USER.

Example:

```
C$ COLLATE(USER)
```

DO LOOP CONTRO

A DO loop control directive modifies the state of on
or both DO loop switches. It affects th
interpretation of only those DO loops whos
DO statements follow the directive in the same progra
unit. The format is as follows:

```
C$ DO(p[=c] [,p[=c]])
```

p — Is either OT or LONG. OT=minimum tri
count. LONG=maximum trip count.

c — Is a constant or the symbolic name of
constant.

Example:

```
C$ DO(LONG=1)
```

FORTRAN LIBRARY: INTRINSIC FUNCTION

Table 11 gives a list and brief description of th
FORTRAN library intrinsic functions.

LIBRARY SUBROUTINES AND FUNCTION

Library subroutines and functions are utilit
subprograms supplied by the system:

- DATE( )

  Returns current date in format mm/dd/yy.

- JDATE( )

  Returns current date in format yyddd. No
  available on SCOPE 2.

- TIME( ) or CLOCK( )

  Returns current time in format hh.mm.ss.

- CALL DISPLA(h,k)

  Places a name and a value in the dayfile
  Parameter h is a character expression to b
  displayed. Parameter k is a real or integer
  variable or expression whose value is to b
  displayed.

TABLE 11. LIST OF INTRINSIC FUNCTIONS

Definition	Specific Name	Type of Argument	Example
Conversion to integer, int(a)	---	Integer	J=INT(I)
	INT	Real	I=INT(X)
	IFIX	Real	K=IFIX(I)
	IDINT	Double	L=IDINT(Z)
	---	Complex	M=IDINT(W)
Conversion to real	FLOAT	Integer	A=FLOAT(K)
	REAL	Integer	B=REAL(*)
	---	Real	C=REAL(D)
	---	Complex	D=REAL(B,C)
	SNGL	Double	E=SNGL(Z)
	---	Complex	F=SNGL(C,D)
Conversion to double	---	Integer	Y=DBLE(I)
	---	Real	Z=DBLE(H)
	---	Double	B=DBLE(W)
	---	Complex	C=DBLE(B)

Conversion to complex	--- --- --- ---	Integer Real Double Complex	S=CMPLX(J) T=CMPLX(A) F=CMPLX(R) G=CMPLX(V,V)
Character conversion to integer	ICHAR	Character	K=ICHAR(I)
Integer conversion to character	CHAR	Integer	I=CHAR(K)
Conversion to Boolean	---	Any type except logical	P=BOOL(X)
Defined as int(a)	AINT DINT	Real Double	Y=AINT(A) Z=DINT(B)
int(a + .5) if a is positive or zero; int(a - .5) if a is negative	ANINT DNINT	Real Double	W=ANINT(T) XI=DNINT(H)
int(a + .5) if a is positive or zero; int(a - .5) if a is negative	NINT IDNINT	Real Double	J=NINT(B) M=IDNINT(C)
a ; if a is complex, square root$_2$ of ((real a)^2 + (imaginary a)^2)	IABS ABS DABS CABS	Integer Real Double Complex	I=IABS(K) F=ABS(T) U=DABS(T) V=CABS(A,B)
$a_1-int(a_1/a_2)*a_2$	MOD AMOD DMOD	Integer Real Double	N=MOD(I,K) G=MOD(C,D) H=MOD(U,V)

TABLE 11. LIST OF INTRINSIC FUNCTIONS (Contd)

Definition	Specific Name	Type of Argument	Example
a_1 if a_2 is positive or zero; $-a_1$ if a_2 is negative	ISIGN SIGN DSIGN	Integer Real Double	K=ISIGN(J,L) P=SIGN(A,B) W=DSIGN(X,Y)
a_1-a_2 if a_1 is greater than a_2; 0 if a_1 is not greater than a_2	IDIM DIM DDIM	Integer Real Double	K=IDIM(I,J) F=DIM(B,C) B=DDIM(S,T)
a_1*a_2	DPROD	Real	R=DPADD(Y,Z)
max (a_1 ,a_2 ,a_n ...)	MAXO AMAX1 DMAX1	Integer Real Double	N=MAXD(J,K,L) A=AMAX1(Q,R) V=DMAX1(O,V)
min (a_1 ,a_2 ,a_n ...)	MINO AMIN1 DMIN1	Integer Real Double	J=MINO(M,N) H=MIN1(B,C,D) XI=DMIN1(X,Y,Z)
Length of character string	LEN	Character	J=LEN(I)
Location of substring a_2 in string a_1	INDEX	Character	K=INDEX(M,N)

Imaginary part of (ar,ai) = ai	AIMAG	Complex	C=AIMAG(B,E)
Negation of imaginary part (ar,-ai)	CONJG	Complex	F=CONJG(V)
Square root of (a)	SQRT	Real	Y=SQRT(Z)
	DSQRT	Double	Z=DSQRT(V)
	CSQRT	Complex	X=CSQRT(H)
e**a	EXP	Real	D=EXP(Y)
	DEXP	Double	A=DEXP(S)
	CEXP	Complex	C=CEXP(T)
$\log_e$ (a)	ALOG	Real	A1=ALOG(X)
	DLOG	Double	A2=DLOG(Y)
	CLOG	Complex	A3=CLOG(Z)
$\log_{10}$ (a)	ALOG10	Real	B1=ALOG10(A)
	DLOG10	Double	B2=DALOG10(C)
sin (a), where a is in radians	SIN	Real	C=SIN(X)
	DSIN	Double	D=DSIN(Y)
	CSIN	Complex	E=CSIN(Z)
sin (a), where a is in degrees	SIND	Real	A=SIND(B)
cos (a), where a is in radians	COS	Real	X1=COS(H)
	DCOS	Double	X2=DCOS(B)
	CCOS	Complex	X3=CCOS(C)

TABLE 11. LIST OF INTRINSIC FUNCTIONS (Contd)

Definition	Specific Name	Type of Argument	Example
cos (a), where a is in degrees	COSD	Real	Y1=COSD(B)
tan (a), where a is in radians	TAN DTAN	Real Double	Y=TAN(A) N=DTAN(B)
tan (a), where a is in degrees	TAND	Real	Z=TAND(X)
arcsin (a)	ASIN DASIN	Real Double	B=ASIN(Y) C=DASIN(Z)
arccos (a)	ACOS DACOS	Real Double	A=ACOS(B) D1=DACOS(C)
arctan (a)	ATAN DATAN	Real Double	F=ATAN(Z) G=DATAN(X)
arctan (a_1/a_2)	ATAN2 DATAN2	Real Double	X=ATAN2(A,B) Y=DATAN1(C,D)
sinh (a)	SINH DSINH	Real Double	A=SINH(X) B=DSINH(Y)

cosh (a)	COSH DCOSH	Real Double	C=COSH(B) E=DCOSH(F)
tanh (a)	TANH DTANH	Real Double	U=TANH(X) V=DTANH(Z)
arctanh (a)	ATANH	Real	X1=ATANH(B)
erf (a)	ERF	Real	X2=ERF(C)
1-erf (a)	ERFC	Real	X3=ERFC(D)
True if a_1 follows a_2, or $a_1=a_2$, in ASCII collating sequence	LGE	Character	J1=LGE(I,K)
True if a_1 follows a_2 in ASCII collating sequence	LGT	Character	J2=LGT(M,N)
True if a_1 precedes a_2, or $a_1=a_2$, in in ASCII collating sequence	LLE	Character	J3=LLE(L,M)
True if a_1 precedes a_2 in ASCII collating sequence	LLT	Character	J4=LLT(L,N)
Boolean result of a_1 shifted a_2 bit positions	SHIFT	Any type but character for a_1; integer for a_2	K=SHIFT(A,I)

TABLE 11. LIST OF INTRINSIC FUNCTIONS (Contd)

Definition	Specific Name	Type of Argument	Example
Boolean result of left-justified 1 bits	MASK	Integer	I=MASK(I,J)
Random number in range (0,1)	RANF	None	H=RANF
Address of variable, array element, sub-string, subroutine, or external function	LOCF	Any type	I=LOCF(A)
CPU time in seconds from start of job	SECOND	None	F=SECOND
Boolean result of .AND. operator	AND	Any type but character	K=AND(J,M,N)
Boolean result of .OR. operator	OR	Any type but character	I=OR(K,L,M)
Boolean result of .XOR. operator	XOR	Any type but character	J=XOR(I,J,K)
Same as exclusive OR	NEQV	Any type but character	A=NEQV(C,D)
Boolean result of .EQV. operator	EQV	Any type but character	B=EQV(U,V,W)
Boolean result of .NOT. operator	COMPL	Any type but character	W=COMPL(D)

- CALL REMARK(h)

  Places a message in the dayfile. Parameter h is a character expression.

- CALL SSWTCH(i,j)

  Tests sense switches. If sense switch i is on, j is set to 1; if sense switch i is off, j is set to 2. Parameter i is a sense switch number. Parameter j is an integer return variable.

- CALL GETPARM(c1,c2,i)

  Accesses user parameters that have been declared on the execution control statement. Parameter c1 is the character variable or array element to receive the parameter name. Parameter c2 is the character variable or array element to receive the parameter value. Parameter i is the integer return code.

- CALL RANSET(n)

  Initializes the seed of RANF. Parameter n is a one-word-bit pattern.

- CALL RANGET(n)

  Obtains the current seed of RANF between 0 and 1. Parameter n is the symbolic name to receive the random number seed.

INPUT/OUTPUT STATUS CHECKING

Input/output status checking can be done by using the following functions:

- UNIT(u)

  Checks the status of a BUFFER IN or BUFFER OUT operation for an end-of-file or parity error condition on logical unit u.

  Example:

```
IF(UNIT(5))12,14,16
```

- EOF(u)

  Tests for an end-of-file condition on unit u following a formatted, list directed, NAMELIST, or unformatted sequential read.

Example:

```
IF(EOF(5).NE.0)GO TO 20
```

- IOCHEC(u)

  Tests for a parity error on unit u following a formatted, list directed, NAMELIST, or unformatted read.

  Example:

  ```
  J=IOCHEC(6)
  ```

- LENGTH(u)
  CALL LENGTHX(u,nw,ubc)

  Returns information regarding the previous BUFFER IN or READMS call of the file designated by u. Parameter nw is the argument set to the number of words read. Parameter ubc is the argument set to the number of unused bits in the last word of the transfer.

- CALL LABEL(u,labinfo)

  Passes label information to the operation system. Parameter labinfo specifies the array for label information.

- CALL MOVLEV(a,b,n)

  Transfers n consecutive words of data between a and b. Parameter a specifies a variable that is the starting address of the data to be moved. Parameter b specifies a variable that is the starting address of the receiving location. This function is normally used to move data to and from ECS or LCM.

  Example:

  ```
  CALL MOVLEV(01,02,1000)
  ```

- CALL MOVLCH(a,b,n)

  Transfers n consecutive characters between a and b. Parameter a specifies a variable, array element, or substring that represents the starting location on the character string to be moved. Parameter be specifies a variable, array element, or substring that represents the starting location of the receiving area.

Example:

```
CALL MOVLCH(CH1(8),CH2(3),369)
```

- CALL CONNEC(u,cs)

  Connects files to the terminal. Parameter cs is the character set designator (0=Display,1=ASCII 96 2=ASCII 256).

- CALL DISCON(u)

  Disconnects a file from within a program

RANDOM FILE ACCESS

The following mass storage input/output subroutines provide random file access capability:

- CALL OPENMS(u,ix,lngth,t)

  Opens the mass storage file and informs the system that it is a random file. Parameter ix is the name of the array containing the master index. Parameter lngth is the length of the master index. Parameter t is the type of index.

  Example:

  ```
  CALL OPENMS(5,I,11,0)
  ```

- CALL WRITMS(u,fwa,n,k,r,s)

  Transmits data from central memory to the file. Parameter fwa is the name of an array in central memory or LCM. Parameter n is the number of 60-bit word to be transferred. Parameter k is the record key. Parameter r specifies rewrite. Parameter s specifies subindex flag.

  Example:

  ```
  CALL WRITMS(3,DATA,25,6,1)
  ```

- CALL READMS(u,fwa,n,k)

  Transmits data from the file to central memory. Parameter u, fwa, n, k are the same as in CALL WRITMS.

Example:

```
CALL READMS(3,MORDAT,25,2)
```

- CALL CLOSMS(u)

  Writes the master index from central memory to the file and closes the file.

  Example:

  ```
  CALL CLOSMS(2)
  ```

- CALL STINDX(u,ix,lngth,t)

  Selects a different array to be used as the current index to the file. Parameter ix is the name of the array in central memory containing the subindex. Parameter lngth is the length of the subindex. Parameter t is the type of subindex.

  Example:

  ```
  CALL STINDX(3,SUBIX,10,0)
  ```

DEBUGGING ROUTINES

The following is a list of miscellaneous debugging routines:

- CALL STRACE

  Provides traceback information from the subroutine calling STRACE back to the main program.

- LEGVAR(a)

  Checks the value of variable a and returns the result -1 if the variable is indefinite, +1 if out of range, and 0 otherwise. Parameter a is the argument.

- CALL SYSTEM(ernum,mesg)

  Enables the user to issue an execution-time error message. Parameter ernum is the error number. Parameter mesg is the error message.

  Example:

  ```
  CALL SYSTEM(3,'CHECKDATA')
  ```

- CALL SYSTEMC(errnum,speclist)

  Enables the user to alter the contents of the error table, which contains specifications that regulate error processing. Parameter errnum is the error number. Parameter speclist is an integer array containing error processing specifications in consecutive locations.

  Example:

```
CALL SYSTEMC(115,IRAY)
```

- CALL LIMERR(lim)

  Enables the user to input data without the risk of termination when improper data is encountered. Parameter lim is the limit for the number of errors.

  Example:

```
CALL LIMERR(200)
```

- NUMERR( )

  NUMERR returns the number of errors since the last LIMERR call.

  Example:

```
IF (NUMERR( ) .GT. 0) GO TO 500
```

COLLATING SEQUENCE CONTROL

The following utility subroutines provide collating sequence control:

- CALL COLSEQ(a)

  Selects a processor-defined, user-specified weight table. Parameter a is a character expression that is one of the following: ASCII6, COBOL6, DISPLAY, or STANDARD.

  Example:

```
CALL COLSEQ('COBOL6')
```

- CALL WTSET(ind,wt)

  Modifies the user-specified weight table. Parameter ind is a character expression. Parameter wt is an integer expression.

  Example:

```
CALL WTSET('$',ICHAR('.'))
```

- CALL CSOWN(str)

  Specifies a partial collating sequence. Parameter str is a character expression.

SAMPLE DECK STRUCTURES

Refer to the operating system reference manual for required job and accounting statements and for details of control statements.

COMPILE AND EXECUTE

A sample deck structure to compile and execute a FORTRAN program is shown in figure 1.

COMPILE AND PRODUCE BINARY CARDS

A sample deck structure to compile and produce binary cards is shown in figure 2.

LOAD AND EXECUTE BINARY PROGRAM

A sample deck structure to load and execute binary programs is shown in figure 3.

COMPILE ONCE AND EXECUTE WITH DIFFERENT DATA DECKS

A sample deck structure to compile once and execute with different data decks is shown in figure 4.

COMPILATION AND EXECUTION WITH OVERLAYS

Preparation of Overlays 0,0; 1,0; and 1,1 are shown in figure 5.

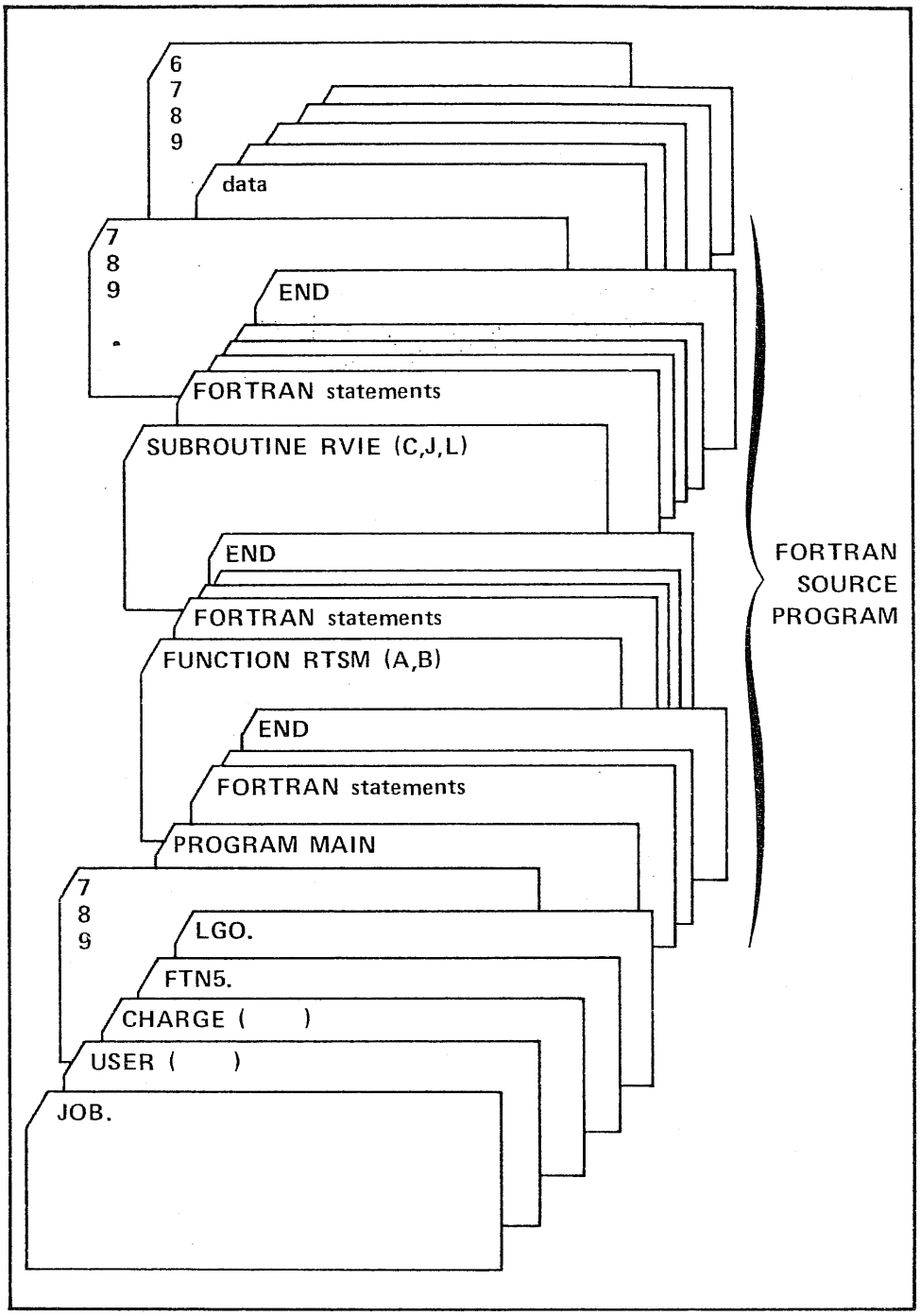

Figure 1. Sample Deck Structure to Compile and Execute

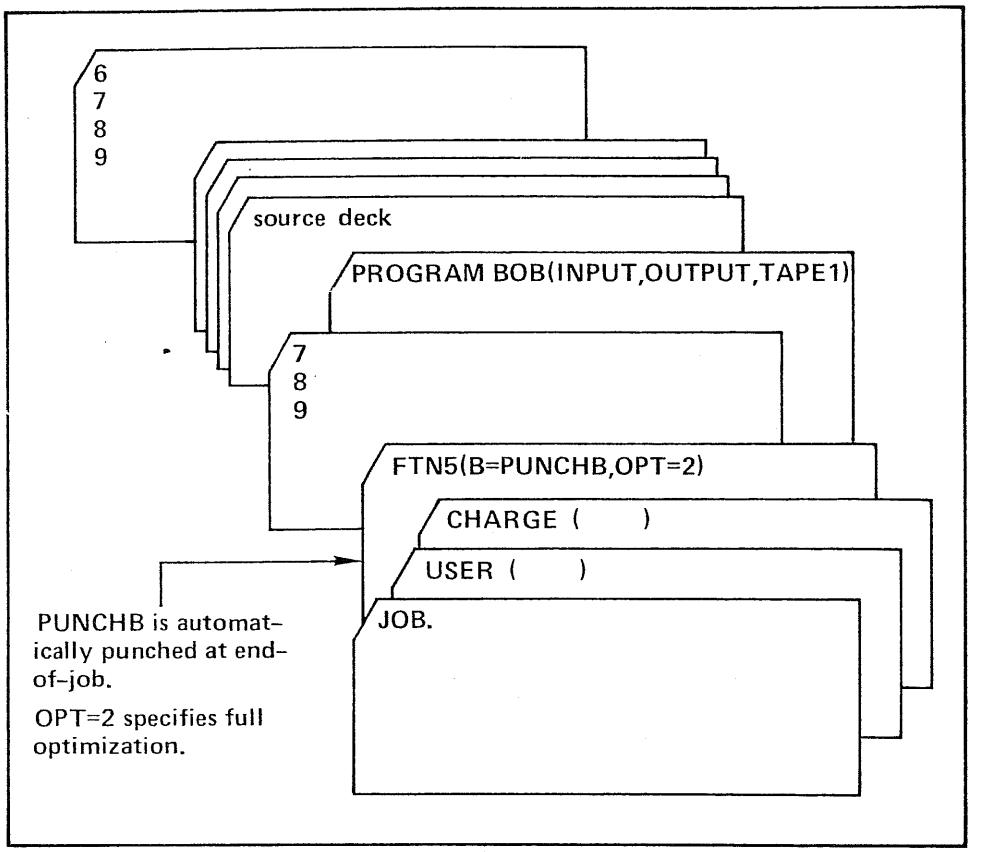

Figure 2. Sample Deck Structure to Compile and Produce Binary Cards

TERMINAL USAGE FORMATS

Terminal usage formats to compile and execute under NOS in the FORTRAN subsystem and the BATCH subsystem are described in this section. For other terminal usage refer to the NOS reference manual.

The formats in figure 6, create programs and input data at the terminal.

The format in figure 7, compiles and executes existing programs at the terminal.

SORT/MERGE INTERFACE

Sort/Merge version 4 and 1 processing can be initiated through FORTRAN 5 call statements. Field length reduction must be off or additional field length requested. A list and description of the Sort/Merge calls are shown in table 12.

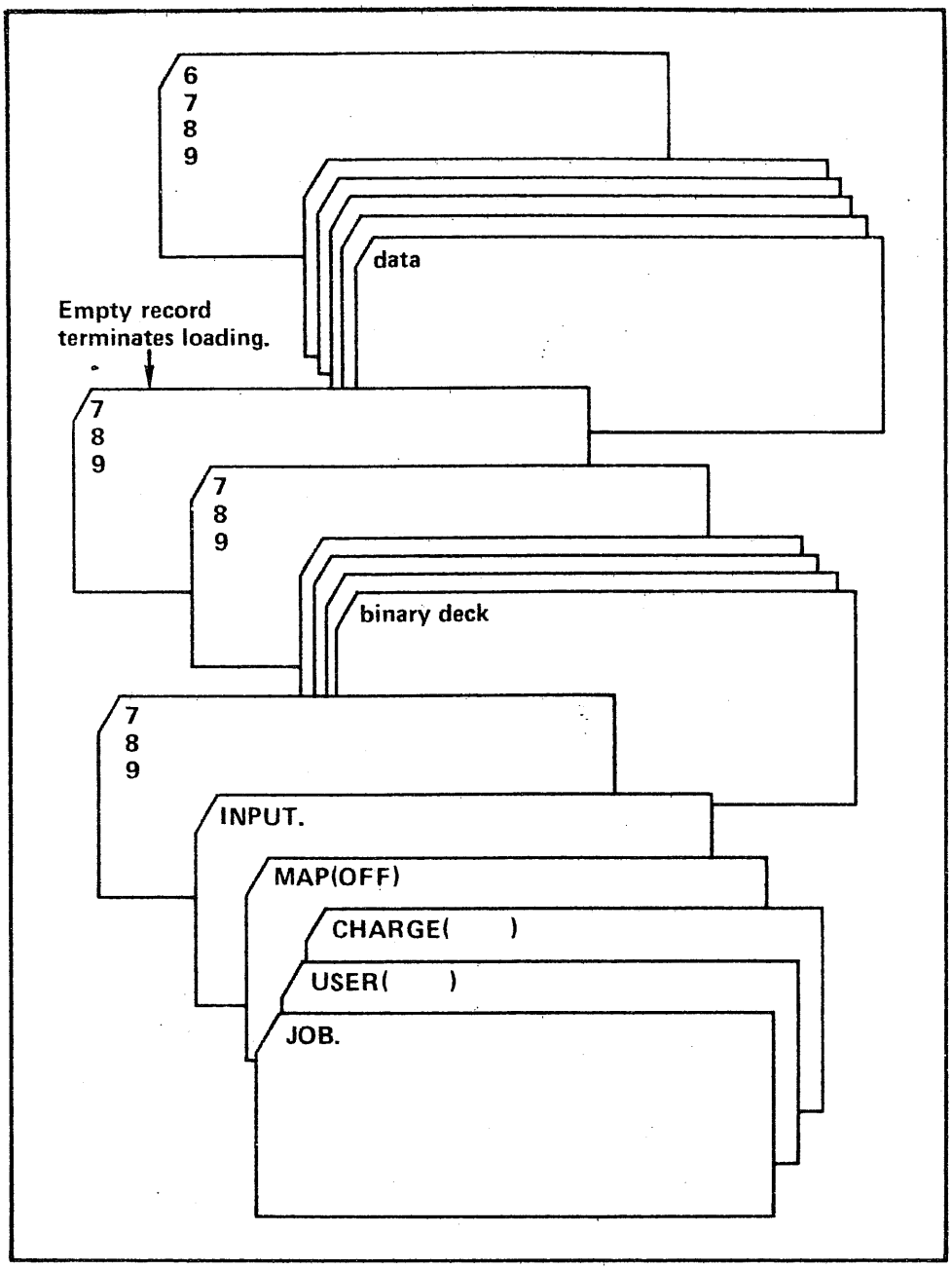

Figure 3. Sample Deck Structure to Load and Execute Binary Programs

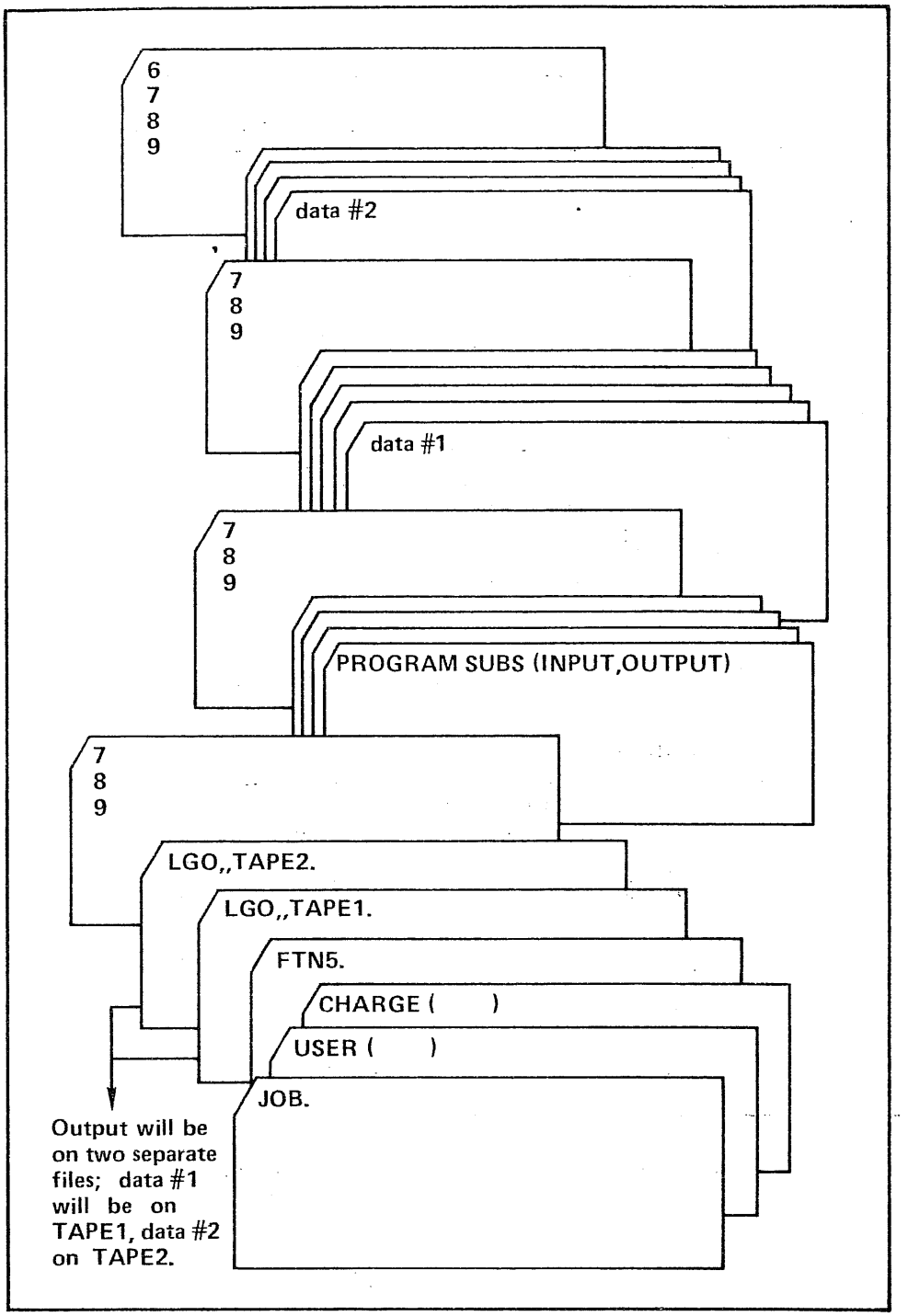

Figure 4. Sample Deck Structure to Compile Once and Execute With Different Data Decks

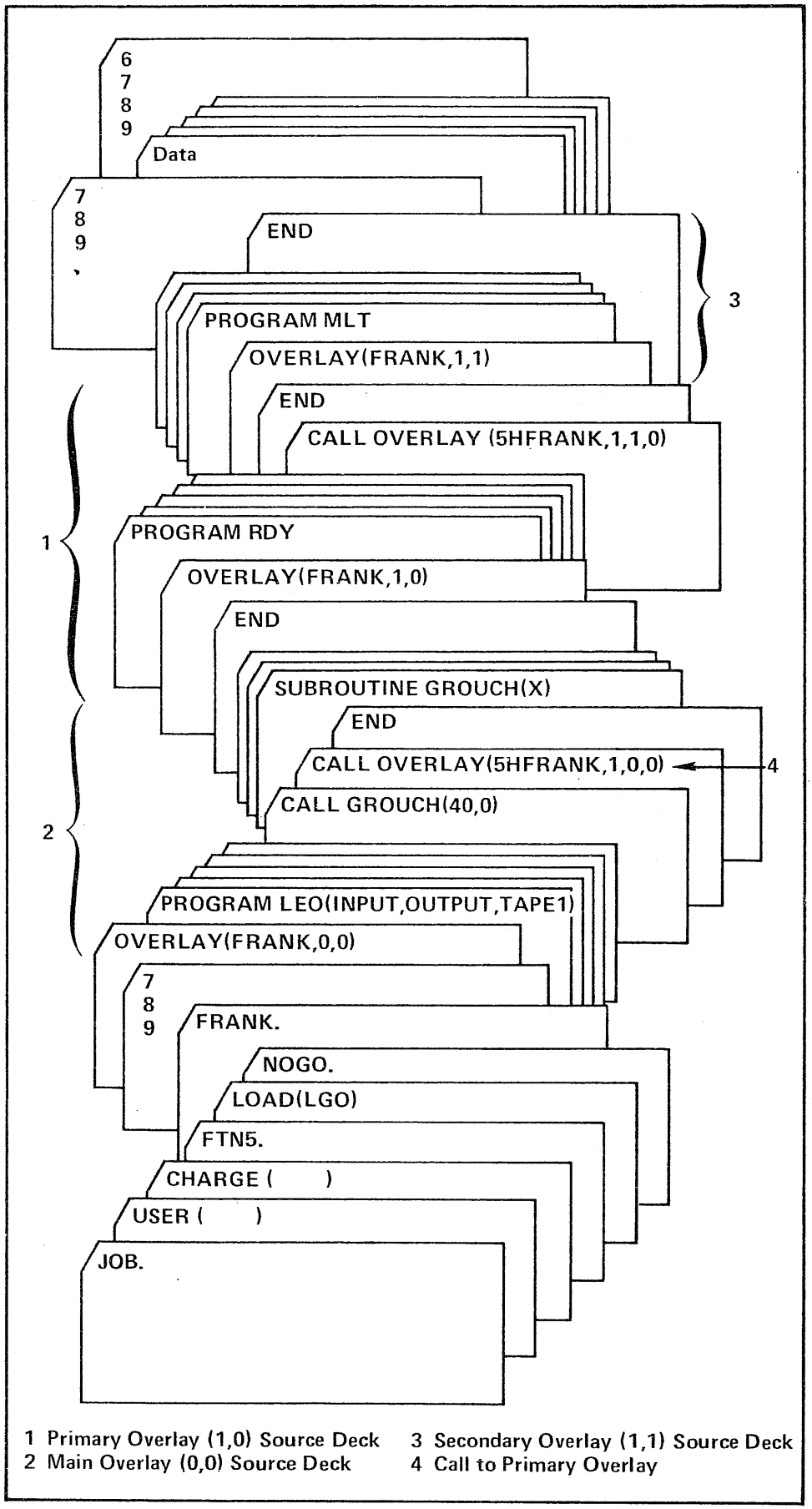

Figure 5. Sample Deck Structure of Overlays

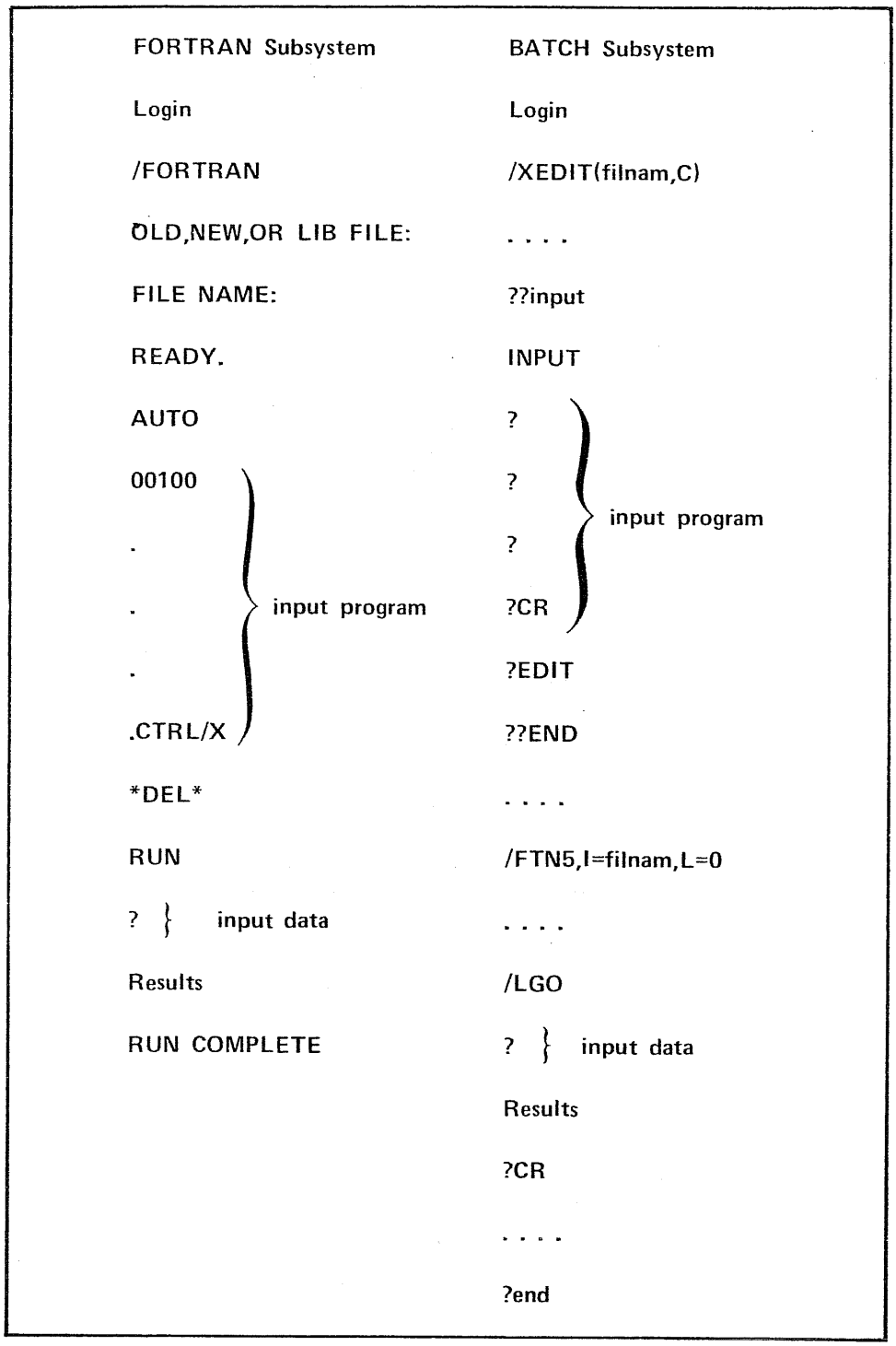

Figure 6. Formats to Compile and Execute Programs Created at the Terminal

```
Login
/GET,FILEX
/GET,FILDATA
/FTN5,I=FILEX,L=0
. . . .
/LGO
```

Figure 7. Format to Compile and Execute Existing Programs at Terminal

FORTRAN CONTROL STATEMENT

The FORTRAN control statement calls the compiler specifies the files to be used for input and output and indicates the type of output to be produced Table 13 lists the FTN5 statement parameters, th defaults, and the initial values.

TABLE 12. SORT/MERGE CALLS

Call	Description
CALL SMSORT(mrl,ba)	Rotating mass-storage sort.
CALL SMMERGE(mrl,ba)	Merge-only processing.
CALL SMSORTB†(mrl,ba)	Balanced tape sort.
CALL SMSORTP†(mrl,ba)	Polyphased tape sort.
	mrl — Maximum record length in characters.
	ba — Number of words of central memory to be used by Sort/Merge for working storage. Parameter ba is the size of the LCM buffer area when applicable to SCOPE 2.
CALL SMFILE(dis,i/o,lfn,action)	Identifies the file to be sorted or merged.
	dis — File disposition: 'SORT', 'MERGE', 'OUTPUT'.
	i/o — Type of file input/output: 'FORMATTED' or 'CODED', 'BINARY', 0†
	lfn — Logical file name: u, L"filename", fit† .

	action — File disposition after sort or merge: 'REWIND', 'UNLOAD', 'NONE'(default).
CALL SMKEY(charpos,bitpos,nchar, nbits,code,colseq,order)	The sort key to be used. charpos — Starting position of sort key counting from 1. bitpos — First bit of charpos counting from 1. nchar — Number of characters in sort key. nbits — Number of bits in excess of nchar. code — Type of code (optional): 'DISPLAY', 'FLOAT', 'INTEGER', 'LOGICAL', ('SIGN', 'LEADING'), ('SIGN', 'TRAILING'), ('SEPARATE', 'LEADING'), ('SEPARATE', 'TRAILING'). colseq — Name of collating sequence only if code is DISPLAY (optional): 'ASCII6', 'COBOL6', 'DISPLAY', 'INTBCD', seqname (collating sequence in SMSEQ). order — Character expression specifying order of sort processing: 'A' ascending, 'D' descending.
CALL SMSEQ(seqnam,seqspec)	A user's collating sequence. seqnam — Name of user-supplied collating sequence.

TABLE 12. SORT/MERGE CALLS (Contd)

Call	Description
	seqspec Name of integer array, terminated by a negative number, with characters in R or octal format in order of collation.
CALL SMEQU(colseq,equspec)	Two or more characters in the collating sequence are equal for comparison purposes. colseq From SMKEY or SMSEQ. equspec Name of integer array, terminated by a negative number, with characters in R or octal format to collate equal to the last character, which must be included in colseq.
CALL SMOPT(optlist)	Specifies special record-handling options. Call immediately after SMSORT or SMMERGE on SCOPE 2. optlist Nonordered series of options separated by commas: 'VERIFY', 'RETAIN', 'VOLDUMP'†, 'DUMP', 'DUMP', n†, 'NODUMP'†, 'ORDER', mo†, 'COMPARE', 'EXTRACT'.
CALL SMOWN(exitnum$_1$,subname$_1$ [,exitnum$_2$,subname$_2$]...)	Owncode exits to be used during Sort/Merge processing.

exitnum Number of the owncode exit.

subname Name of the user-supplied EXTERNAL owncode exit subroutine which exits through a call to system subroutine SMRTN. The owncode exit numbers are:

entry: SUBROUTINE subname(a,rl)

exit 1†† or 3: CALL SMRTN(retaddr), for retaddr=1 or 3
CALL SMRTN(retaddr,b,rl), for retaddr=0 or 2.

entry: SUBROUTINE subname

exit 2 or 4: CALL SMRTN(retaddr), for retaddr=0
CALL SMRTN(retaddr,b,rl), for retaddr=1.

entry: SUBROUTINE subname(a_1,rl_1,a_2,rl_2)

exit 5: CALL SMRTN(b_1,rl_1,b_2,rl_2), for retaddr=0
CALL SMRTN(b_1,rl_1), for retaddr=1.

retaddr Return address:
0 Normal return address
1 Normal return address+1
2 Normal return address+2
3 Normal return address+3

TABLE 12. SORT/MERGE CALLS (Contd)

Call	Description
	a — Integer array of (rl+9/10) words. Record stored by Sort/Merge when subname is called. b — Integer array of (rl+9/10) words. Record stored by user when subname is called. rl — Record length in characters
CALL SMTAPE(tapelist)	Set of tapes to be merged. tapelist — The logical file names, each in the form L"filename", to be used in balanced or polyphase tape merge.
CALL SMEND	Execution of the sort or merge.
CALL SMABT	A sequence of Sort/Merge interface calls without initiating execution of Sort/Merge.

†Does not apply to SCOPE 2. If a file is to be accessed with Record Manager subroutines, OPENM should be called prior to SMFILE.

††No parameters needed for exit number 1 if no input files exist.

TABLE 13. FORTRAN CONTROL STATEMENT DEFAULTS

Parameter	First Default (parameter omitted)	Second Default (keyword only)	Initial Values
ANSI	ANSI=0	ANSI=T	
ARG	ARG=0	ARG=-COMMON/FIXED	ARG=0
B	B=LGO	B=BIN	
BL	BL=0	BL	
CS	CS=USER	CS=FIXED	
DB	DB=0 if opt=1, 2, or 3; DB=ER if opt=0	DB=TB/SB/SL/ER/PMD	DB=TB/SB/SL/ER/MD
DO	DO=0	DO=OT	DO=0
DS	DS=0	DS	
E	E=OUTPUT	E=ERRS	
EL	EL=T	EL=F	
ET	ET=0	ET=F	
G	G=0	G=SYSTEXT	
GO	GO=0	GO	
I	I=INPUT	I=COMPILE	
L	L=OUTPUT	L=LIST	
LCM	LCM=D	LCM=I	

CORPORATE HEADQUARTERS, 8100 34th AVE. SO.
MINNEAPOLIS, MINN. 55440

SALES OFFICES AND SERVICE CENTERS
IN MAJOR CITIES THROUGHOUT THE WORLD