60497900

CONTROL DATA CORPORATION

FORTRAN EXTENDED
VERSION 4
INSTANT MANUAL

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

60497900

CONTROL DATA CORPORATION

FORTRAN EXTENDED VERSION 4 INSTANT MANUAL

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

REVISION RECORD

Revision	Description
A (03/29/76)	Original printing, documenting FORTRAN Extended Version 4.6.
B (06/12/81)	This revision documents Version 4.8 of FORTRAN Extended at PSR level 533. Features documented include CRM products BAM and AAM, the Post Mortem Dump facility, the CYBER Interactive Debug interface, and the STATIC option for FORTRAN Extended.

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	Revision
Front Cover	-
Title Page	-
ii	B
iii/iv	B
v/vi	B
vii/viii	B
1 thru 49	B
Back Cover	-

PREFACE

This instant provides a brief description of the major language features of FORTRAN Extended Version 4.8. FORTRAN Extended is designed to comply with the American National Standards Institute FORTRAN language, as described in X3.9-1966.

The FORTRAN Extended compiler operates in conjunction with the COMPASS 3 assembly language processor under control of the following operating systems:

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

Relocatable binaries compiled by versions of FORTRAN Extended prior to Version 4.7 cannot be run with CRM BAM 1.5 or AAM 2; they must be recompiled. In this instant CONTROL DATA extensions to the language are indicated by shading.

More detailed information can be found in the following publications:

Publication	Publication Number
FORTRAN Extended Version 4 Reference Manual	60497800
FORTRAN Common Library Mathematical Routines Reference Manual	60498200
CID Version 1 Guide for Users of FORTRAN Extended Version 4	60482700
FORTRAN Extended Version 4 User's Guide	60499700
NOS Version 1 Reference Manual, Volume 1 of 2	60435400

NOS/BE Version 1 Reference Manual	60493800
SCOPE Version 2 Reference Manual	60342600

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

This manual describes a subset of the features and parameters documented in the FORTRAN Extended Version 4 Reference Manual. Control Data cannot be responsible for the proper functioning of any features or parameters not documented in the FORTRAN Extended Version 4 Reference Manual.

CONTENTS

TABLES

LANGUAGE ELEMENTS

SYMBOLIC NAMES

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

FORTRAN CHARACTER SET

Alphabetic:	A to Z						
Numeric:	0 to 9						
Special:	=	equal	/	slash	.	decimal point
	+	plus	(	left parenthesis	$	dollar sign
	–	minus	)	right parenthesis		blank
	*	asterisk	,	comma	≠ or "	quote

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

FORTRAN STATEMENTS

Column	Contents
1	C or $ or * indicates comment line
1-2	C$ indicates debug directive if in debug mode
1-2	C/ indicates list directive
1-5	Statement label
6	Any character other than blank or zero denotes continuation, except on comment lines or list directives
7-72	Statement
73-80	Identification field; not processed by compiler, but printed with source listing

Statements are written in columns 7-72; blanks are ignored except in Hollerith fields. All 80 columns can be used for data input. Statements can be labeled by an integer constant in the range 1-99999. If a C, $ or * appears in column 1 without a $ or / in column 2, the remainder of the card is ignored by the compiler but printed with the source listing as a comment.

$ may be used to separate multiple statements on a card except with FORMAT statements or list or debug directives.

CONSTANTS

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

TABLE 1. LIST OF CONSTANTS

Constants	Form	Examples
Integer	$n_1 n_2 \ldots n_m$ $1 \leqslant m \leqslant 18$ Range: $-(2^{59}-1)$ to $2^{59}-1$ Integer addition and subtraction results can range from $-(2^{59}-1)$ to $2^{59}-1$. Integer multiplication and division operands and results can range from $-(2^{48}-1)$ to $2^{48}-1$. Integers used as a DO index or as a subscript must be in the range from 1 to $2^{17}-1$.	2 247 31456932
Real	n.n .n n. n.nE±s .nE±s n.E±s nE±s n Coefficient ⩽ 15 decimal digits E±s Exponent s Base 10 scale factor Range 10^{-293} to 10^{+322} Accurate to approximately 14 decimal digits	 7.5 3.22 42.E1 314.E05 700.E-2 .5 0.
Double Precision	n.nD±s .nD±s n.D±s nD±s n Coefficient ⩽ 29 decimal digits D±s Exponent s Base 10 scale factor Range 10^{-293} to 10^{+322} Accurate to approximately 29 decimal digits	5.834D2 7.D2 9.2D03 14.D-5 3120D4 1.D0
Complex	(r1,r2) r1 Real part r2 Imaginary part Each part has same range as a real constant	(1.,7.54) (-2.1E1,3.24) (0.,-1.) (4.0,5.0)

TABLE 1. LIST OF CONSTANTS (Contd)

Constants	Form	Examples
Octal	$n_1 \cdots n_m B$ $1 \leq m \leq 20$	7777777777B 525252B
Hollerith	nHf nRf nLf ≠f≠ or "f" $1 \leq n \leq 10$ in expression	
	H left justified with blank fill	6HABCDEF
	R right justified with binary zero fill L left justified with binary zero fill A Hollerith string delimited by paired symbols ≠ ≠ or quotation marks can be used anywhere the H form of the Hollerith constant can be used. For example: IF(V.EQ.≠YES≠) GO TO 20 PRINT 1,≠ SQRT = ≠ ,SQRT(.5) 1 FORMAT (A10,F10.2)	7RJUSTIFY 7LTHE END ≠ABCDEF≠ "ABCDEF"
Logical	.TRUE. or .T. .FALSE. or .F.	LOGICAL X1, Z2 X1 = .TRUE. Z2 = .FALSE.

VARIABLES

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

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

A variable not defined in a type declaration is real if the first character of the symbolic name is any letter other than I,J,K,L,M,N and if the type is not changed by an IMPLICIT statement in that program unit. The default implicit typing is as follows:

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

TABLE 2. LIST OF VARIABLES

Variables	Form	Examples
Integer	Range $-(2^{59}-1)$ to $2^{59}-1$. As subscript or index of a DO statement, maximum value is $2^{17}-1$. As a result of multiplication or division or conversion between real and integer, maximum value is $2^{48}-1$.	ITEM JSUM KOOL INTEGER X
Real	Range 10^{-293} to 10^{+322}, approximately 14 significant digits.	AVAR SUM TUF BETA REAL I
Double Precision	Must be defined explicitly in type declaration. Range 10^{-293} to 10^{+322}, approximately 29 significant digits. Occupies two storage words.	DOUBLE PRECISION *OMEGA,X,B DOUBLE X,Y
Complex	Must be defined explicitly in type declaration. Occupies two storage words; each contains a number in real variable format, and each number can range from 10^{-293} to 10^{+322}.	COMPLEX A,D COMPLEX P2
Logical	Must be defined explicitly in type declaration.	LOGICAL L3,C LOGICAL L2,R

ARRAYS

An array name can have up to three subscripts. Zero and negative subscripts are not allowed. Subscripts can be any valid arithmetic expression. A non-integer subscript value is truncated to integer. If the number of subscripts in a reference is less than the declared dimensions of the array, the compiler assumes a value of one for missing subscripts. The number of subscript expressions in a reference must not exceed the number of declared dimensions.

Table 3 can be used to find an element in the linear sequence of storage locations.

TABLE 3. ARRAY ELEMENT POSITION

Number of Dimensions	Array Dimension	Subscript	Location of Element Relative to Starting Location
1	ALPHA(K)	ALPHA(k)	(k-1)*E
2	ALPHA(K,M)	ALPHA(k,m)	(k-1+K*(m-1))*E
3	ALPHA(K,M,N)	ALPHA(k,m,n)	(k-1+K*(m-1+M*(n-1)))*E

The following notations are used in table 3:

- K, M, and N are dimensions of the array.
- k, m, and n are actual subscript values of the array.
- 1 is subtracted from each subscript value because the subscript starts with 1, not 0.
- E is length of the element. For real, logical, and integer arrays, E = 1. For complex and double precision arrays, E = 2.

PRINTER CONTROL CHARACTERS

The first characters of each line in a print file perform the following functions:

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
+	No advance before printing; allows overprinting
Any other character	Refer to operating system reference manual

STATEMENT FORMS

The following symbols are used in the descriptions of statements:

v	variable or array element
sn	statement label
iv	integer variable
name	symbolic name
u	input/output unit: 1- or 2-digit decimal integer constant; integer variable with value of: 0-99 or a Hollerith value denoting the file name, left justified with zero fill.
fn	format designator; such as a statement label, array name, variable, or array element
p	dummy arguments that agree in order, type, and number to the actual arguments passed to the subroutine
b	dummy statement label arguments that agree in order, type, and number to the actual statement labels passed to the subroutine

m	indexing parameter
n	string of 1–5 octal digits
c. . . c	string of 1–70 characters
eam	arithmetic or masking expression
erl	relational or logical expression
a	variable or array name
iolist	input/output list specifying items to be transmitted
stat	any unlabeled executable statement other than END, DO, or another logical IF.

ASSIGNMENT STATEMENTS

Form	Examples
v = arithmetic expression	A = B + C
logical v = erl	LOGICAL L,M,N L = M .AND.N
v = masking expression	CAT = 5252B .OR. Z

MULTIPLE ASSIGNMENT

$v_1 = v_2 = \ldots v_n$ = expression	X = Y = Z = (10. + B)/SUM(1)

FLOW CONTROL STATEMENTS

GO TO sn	GO TO 30
GO TO ($sn_1, \ldots, sn_m$), iv	GO TO (1,4,7,2), N
GO TO ($sn_1, \ldots, sn_m$) iv	GO TO (3,6,10,1) J
GO TO ($sn_1, \ldots, sn_m$), eam	GO TO (1,2,9,4), A + B
GO TO ($sn_1, \ldots, sn_m$) eam	GO TO (3,4,5,6) N + J
GO TO iv ($sn_1, \ldots, sn_m$)	GO TO NEXT (1,2,3,4)
GO TO iv,($sn_1, \ldots, sn_m$)	GO TO LSWTCH, (10,20,30,40)
ASSIGN sn TO iv	ASSIGN 10 TO LSWTCH
IF (eam) sn_1,sn_2,sn_3	IF (I-N) 3,4,6
IF (eam) sn_1,sn_2	IF (I*Y*K) 100, 200
IF (erl) sn	IF (P.AND.Q) RES = 7.2
IF (erl) sn_1,sn_2	IF (K.EQ. 100) 60,70
DO sn iv = m_1,m_2,m_3	DO 100 I = 1,10,2
DO sn iv = m_1,m_2	DO 2 J = 1,5

Form	Examples
sn CONTINUE	100 CONTINUE
PAUSE	PAUSE
PAUSE n	PAUSE 2
PAUSE ≠c . . . c≠	PAUSE ≠ CHANGE TAPE ≠
STOP	STOP
STOP n	STOP 25
STOP ≠c . . . c≠	STOP ≠ END OF RUN ≠
END	END

TYPE DECLARATION

Arrays can be dimensioned in type specifications.

INTEGER $name_1, \ldots, name_n$	INTEGER A,B,C(10)
TYPE INTEGER $name_1, \ldots, name_n$	TYPE INTEGER X,Y,N
REAL $name_1, \ldots, name_n$	REAL NEXT,X(5)
TYPE REAL $name_1, \ldots, name_n$	TYPE REAL N,J,CAT
COMPLEX $name_1, \ldots, name_n$	COMPLEX CC,J
TYPE COMPLEX $name_1, \ldots, name_n$	TYPE COMPLEX NON,Z(3)
DOUBLE PRECISION $name_1, \ldots, name_n$	DOUBLE PRECISION DP1,DP2
DOUBLE $name_1, \ldots, name_n$	DOUBLE DP3
TYPE DOUBLE PRECISION $name_1, \ldots, name_n$	TYPE DOUBLE PRECISION CAT,DOG
TYPE DOUBLE $name_1, \ldots, name_n$	TYPE DOUBLE HEN,DUCK
LOGICAL $name_1, \ldots, name_n$	LOGICAL L1,L2
TYPE LOGICAL $name_1, \ldots, name_n$	TYPE LOGICAL LL,LN
IMPLICIT $type_1(ac), \ldots, type_n(ac)$	IMPLICIT REAL (I-N),COMPLEX(A,Q,R-T)

ac one or more alphabetic characters or ranges of characters (first and last separated by a minus sign) separated by commas

EXTERNAL DECLARATION

EXTERNAL $name_1, \ldots, name_n$	EXTERNAL ABS

STORAGE ALLOCATION

Form	Examples
type $name_1$ (d_1)	REAL CJ (3,3)
TYPE type $name_1$ (d_1)	TYPE REAL DJ (10)
DIMENSION $name_1$ (d_1), . . . , $name_n$ (d_n)	DIMENSION SUM (10)

d_i array declarator, one to three integer constants; in a subprogram, one to three integer constants or variables

type INTEGER, REAL, COMPLEX, DOUBLE PRECISION, LOGICAL, or DOUBLE

COMMON v_1, . . . , v_n	COMMON A,B,C
COMMON/blkname$_1$/v_1, . . . , v_n . . . /blkname$_n$/v_1, . . . , v_n	COMMON/BLK/D,E,F/CAT/X,Y,Z(10)
COMMON//v_1, . . . , v_n	COMMON//NEXT,JAY(3)

blkname symbolic name or 1–7 digits

// blank common

v_i variable or array name

DATA $vlist_1$/$dlist_1$/, . . . , $vlist_n$/$dlist_n$/	DATA A,B,C/3.,27.5,5.0/
DATA ($vlist_1$=$dlist_1$), . . . ,($vlist_n$=$dlist_n$)	DATA (X=3.),(Y=5.)

$vlist_i$ array names, array elements, variable names or implied DO list, separated by commas

$dlist_i$ one or more constant list forms separated by commas, with rf an integer constant repetition factor:

constant	(constant list)
rf*constant	rf*(constant list)
	rf(constant list)

EQUIVALENCE ($glist_1$), . . . ,($glist_n$)	EQUIVALENCE (N,J),(X,Y)

$glist_i$ two or more variables, array elements, or array names separated by commas

LEVEL ℓ,a_1, . . . , a_n	LEVEL 3,X,Y,Z

ℓ unsigned integer 1, 2 or 3, indicating level to which list is allocated

a_i variable or array name

PROGRAM UNITS

PROGRAM name ($file_1$, . . . , $file_n$)	PROGRAM A(INPUT,OUTPUT)
PROGRAM name	PROGRAM B

Form	Examples
FUNCTION name($p_1, \ldots, p_n$)	FUNCTION GRATER(A,B)
type FUNCTION name ($p_1, \ldots, p_n$)	REAL FUNCTION D(X,Y)
BLOCK DATA	BLOCK DATA
BLOCK DATA name	BLOCK DATA BD3
SUBROUTINE name ($p_1, \ldots, p_n$)	SUBROUTINE X(C,D,E)
SUBROUTINE name	SUBROUTINE PGM

SUBROUTINE name ($p_1, \ldots, p_n$), RETURNS ($b_1, \ldots, b_m$)

SUBROUTINE SUB(X,Y), RETURNS (M,N)

SUBROUTINE name, RETURNS ($b_1, \ldots, b_m$)

SUBROUTINE SUB2, RETURNS(J,K,L)

ENTRY POINT

ENTRY name	ENTRY BOX

STATEMENT FUNCTIONS

name ($p_1, \ldots, p_n$) = expression	ADD(X,Y,C,D) = X+Y+C+D

SUBPROGRAM CONTROL STATEMENTS

CALL name	CALL JIM
CALL name ($p_1, \ldots, p_n$)	CALL JIM (A,2)

CALL name ($p_1, \ldots, p_n$),RETURNS ($b_1, \ldots, b_m$)

CALL JOHN (X,Y),RETURNS (2,3)

CALL name,RETURNS ($b_1, \ldots, b_m$)	CALL SUB4,RETURNS (8,2,2)
RETURN	RETURN
RETURN i	RETURN M

i a dummy argument in a RETURNS list

INPUT/OUTPUT

PRINT fn,iolist	PRINT 4,A,B,N
PRINT fn	PRINT 20
PRINT(u,fn) iolist	PRINT (6,17) A

Form	Examples
PRINT*,iolist	PRINT *, N, (C(I),I=1,N)
PRINT(u,fn)	PRINT(NUM, 46)
PRINT(u,*) iolist	PRINT (19,*) X,SQRT(X)
PUNCH fn,iolist	PUNCH 2,X,Y,Z
PUNCH fn	PUNCH 30
PUNCH(u,fn) iolist	PUNCH(NUM,FMT)J,K
PUNCH*,iolist	PUNCH *,≠X =≠,X
PUNCH(u,*) iolist	PUNCH (4,*) A(I), A(J)
PUNCH(u,fn)	PUNCH (45,66)
WRITE(u,fn) iolist	WRITE (4,27) X,Y,Z
WRITE(u,fn)	WRITE (2,30)
WRITE fn,iolist	WRITE 203, B(2)
WRITE fn	WRITE 66
WRITE (u) iolist	WRITE (3) A,B,C
WRITE (u)	WRITE (3)
WRITE(u,*) iolist	WRITE (4,*) ABCD
WRITE*,iolist	WRITE *,J,K
READ(u,fn) iolist	READ (5,100) X,Y,Z
READ(u,fn)	READ (5,100)
READ fn,iolist	READ 100,A,B,C
READ fn	READ 100
READ(u) iolist	READ (3) JN,AB
READ(u)	READ (5)
READ(u,*) iolist	READ (5,*) Q,R
READ*,iolist	READ*,(C(J),D(J),J=I,N)
BUFFER IN (u,p) (a,b)	BUFFER IN (1,1)(R(1),R(512))
BUFFER OUT(u,p) (a,b)	BUFFER OUT(1,J) (B(M),B(N))
a,b first and last word of data block to be transferred p integer constant or integer variable: 0, even parity; 1, odd parity	
NAMELIST/group name$_1$/a_1, . . . , a_n/ . . . /group name$_n$/a_1, . . . a_n	NAMELIST/SHIP/I1,I2,A,B

Form	Examples
READ (u,group name)	READ (5,SHIP)
WRITE (u,group name)	WRITE (6,SHIP)
READ group name	READ SHIP
PRINT group name	PRINT SHIP
PUNCH group name	PUNCH SHIP

group name symbolic name identifying the group $a_1, \ldots a_n$

INTERNAL TRANSFER OF DATA

ENCODE (c,fn,v) iolist	ENCODE (40,1,ALPHA) A,B,C
DECODE (c,fn,v) iolist	DECODE (77,17,CARD) INK

c length of record in characters; unsigned integer constant or simple integer variable

v starting location of record; variable or array name

FILE MANIPULATION

REWIND u	REWIND 3
BACKSPACE u	BACKSPACE LUN
ENDFILE u	ENDFILE 4

FORMAT SPECIFICATION

sn FORMAT ($fs_1, \ldots, fs_n$) 100 FORMAT (I6,F7.3,214)

fs_i one or more field specifications separated by commas and/or slashes and optionally grouped by parentheses

DATA CONVERSION

Leading blanks are not significant in numeric input conversions; other blanks are treated as zeros. When an all-blank field is read with a Hollerith input specification, each blank character is translated to a display code 55_8. The output field is right-justified and blank-filled for all output conversion.

srEw.d	Single precision floating point with exponent	2E13.3
srEw.dEe	Floating point with specified exponent length	E10.2E1
srEw.dDe	Floating point with specified exponent length	E30.20D3
srFw.d	Single precision floating point without exponent	F7.3
srGw.d	Single precision floating point with or without exponent	G14.6
srDw.d	Double precision floating point with exponent	2D10.4

Form		Examples
rIw	Decimal integer conversion	4I9
rIw.z	Integer with specified minimum digits	I6.2
rLw	Logical conversion	2L5
rAw	Alphanumeric conversion	A7
rRw	Alphanumeric conversion	4R10
rOw	Octal integer conversion	O5
rOw.z	Octal integer with specified minimum digits	O24.16
rZw	Hexadecimal conversion	Z8

s optional scale factor of the form: nP
r optional repetition factor
w nonzero unsigned integer constant indicating field width
d unsigned integer constant indicating digits to right of decimal point
e nonzero unsigned integer indicating digits in exponent field
z integer specifying minimum number of digits

nX	Intraline spacing	9X
nH . . . * . . . * ≠ . . . ≠ " . . . "	Hollerith	8H THE END *FINIS* ≠ TEST 7 ≠ "NEW TEST"
/	Format field separator; indicates end of FORTRAN record	/8HNEW LINE/
Tn	Column tabulation; control skips to column n	T10
V	Display code substitution; the rightmost six bits from the current variable in the input/output list are interpreted as display code	V8.2
=	Numeric substitution; the current variable in the input/output list supplies a positive integer value to be used in place of the = for the next variable in the input/output list	A=

OVERLAYS

CALL OVERLAY (fname,i,j,recall,k)

CALL OVERLAY (4HTEXT,1,0,6HRECALL)

fname	name of file or overlay in H format
i,j	octal with a B or decimal equivalent level numbers
recall	6HRECALL stops reloading of overlay already in memory
k	L format Hollerith constant: name of library containing overlay Any other non-zero value: overlay loaded from global library

Form	Examples
OVERLAY(fname,i,j,origin,ov=m	OVERLAY(TEST,0,0,OV=4)

fname	file where generated overlay is to be written
i,j	octal level numbers of the overlay
origin	overlay origin; optional (not allowed for 0,0 overlay)
ov=m	total decimal number(m) of the high level overlay structure; optional (valid only for 0,0 overlay)

DEBUG STATEMENTS

The D option on FTN control statement selects debugging mode; if it is not specified, debugging statements are treated as comments.

Debug statements are written in columns 7-72; columns 1 and 2 of each statement must contain C$. Any character, other than blank or zero, in column 6 denotes a continuation line. Columns 3, 4, and 5 of a continuation line must be blank.

C$ DEBUG

C$ DEBUG ($name_1, \ldots, name_n$)

C$ AREA $bounds_1, \ldots, bounds_n$
C$ DEBUG
} within program unit

C$ AREA/$name_1$/$bounds_1, \ldots, bounds_n, \ldots$
/$name_n$/$bounds_1, \ldots, bounds_n$
C$ DEBUG ($name_1, \ldots, name_n$)
or
C$ DEBUG
} external debug deck

bounds (n_1,n_2) n_1 initial line position; n_2 terminal line position
(n_3) n_3 single line position to be debugged
(n_1,*) n_1 initial line position; * last line of program
(*,n_2) * first line of program; n_2 terminal line position
(*,*) first and last lines of program

C$ ARRAYS ($a_1, \ldots, a_n$)

C$ ARRAYS

a_i array names

C$ CALLS ($s_1, \ldots, s_n$)

C$ CALLS

s_i subroutine names

C$ FUNCS ($f_1, \ldots, f_n$)

C$ FUNCS

f_i function names

C$ GOTOS

C$ NOGO

C$ STORES ($c_1, \ldots, c_n$)

c_i variable name

variable name.relational operator.constant

variable name.relational operator.variable name

variable name.checking operator.

checking operators:

RANGE	out of range
INDEF	indefinite
VALID	out of range or indefinite

C$ TRACE (lv)

C$ TRACE

lv level number: 0, tracing outside DO loops; n, tracing up to and including level n in DO nest

C$ OFF

C$ OFF ($x_1, \ldots, x_n$)

x_i any debug option

LIST DIRECTIVES

List directive statements contain C/ in columns 1 and 2 with the option, LIST, ALL or LIST, NONE, appearing anywhere within columns 7–72.

C/	LIST,NONE	Stops source program listing and can suppress the other listings
C/	LIST,ALL	Resumes source program listing

COMPASS SUBPROGRAM IDENTIFICATION

IDENT name	(Starts in column 11)	Begins COMPASS
END	(Starts in column 11)	Terminates subprogram

FORTRAN INTRINSIC FUNCTIONS

An intrinsic function is a compiler-defined procedure that returns a single value. The FORTRAN Extended intrinsic functions are shown in table 4.

TABLE 4. FORTRAN INTRINSIC FUNCTIONS

Definition	Arguments		Type of Function	Symbolic Name	Example
	Max. No.	Type			
Absolute value |A|	1	Real	Real	ABS	Y=ABS(X)
		Integer	Integer	IABS	J=IABS(I)
		Double	Double	DABS	DOUBLE A,B B=DABS(A)
Truncate A: sign of A times largest integer $\leq$ |A|†	1	Real	Real	AINT	Y=AINT(X)
		Real	Integer	INT	I=INT(X)
		Double	Integer	IDINT	DOUBLE Z J=IDINT(Z)
Remaindering†† A1(mod A2)	2	Real	Real	AMOD	B=AMOD(A1,A2)
		Integer	Integer	MOD†	J=MOD(I1,I2)
Choosing largest of 2-63 values: max (A1, . . . , An)	63	Integer	Real	AMAX0	X=AMAX0(I,J,K)
		Real	Real	AMAX1	A=AMAX1(X,Y,Z)
		Integer	Integer	MAX0	L=MAX0(I,J,K,N)
		Real	Integer	MAX1	I=MAX1(A,B)
		Double	Double	DMAX1	DOUBLE W,X,Y,Z W=DMAX1(X,Y,Z)
Choosing smallest of 2-63 values: min (A1, . . . , An)	63	Integer	Real	AMIN0	Y=AMIN0(I,J)
		Real	Real	AMIN1	Z=AMIN1(X,Y)
		Integer	Integer	MIN0	L=MIN0(I,J)
		Real	Integer	MIN1	J=MIN1(X,Y)
		Double	Double	DMIN1	DOUBLE A,B,C C=DMIN1(A,B)
Convert from integer to real	1	Integer	Real	FLOAT	X1=FLOAT(I)
Convert from real to integer	1	Real	Integer	IFIX	IY=IFIX(Y)
Transfer sign of A2 to |A1|, +A1 if A2=+0, -A1 if A2=-0	2	Real	Real	SIGN	Z=SIGN(X,Y)
		Integer	Integer	ISIGN	J=ISIGN(I1,I2)
		Double	Double	DSIGN	DOUBLE X,Y,Z Z=DSIGN(X,Y)
Positive difference: if A1 > A2 then A1-A2. If A1 $\leq$ A2 then 0	2	Real	Real	DIM	A=DIM(C,D)
		Integer	Integer	IDIM	J=IDIM(I1,I2)

†Absolute value of arguments must be $\leq 2^{48}-1$.

††MOD or AMOD (a,b) is defined as a-[a/b] b, where [X] is the largest integer that does not exceed the magnitude of X with sign the same as X.

TABLE 4. FORTRAN INTRINSIC FUNCTIONS (Contd)

Definition	Arguments		Type of Function	Symbolic Name	Example
	Max. No.	Type			
Logical product: bit-by-bit logical AND of 2 or more values	63	any type†	no mode	AND	C=AND(X,Y,Z)
Logical sum: bit-by-bit logical OR of 2 or more values	63	any type†	no mode	OR	D=OR(X,Y,Z)
Exclusive OR: bit-by-bit exclusive OR of 2 or more values	63	any type†	no mode	XOR	D=XOR(X,Y,Z)
Complement: bit-by-bit Boolean complement of A	1	any type†	no mode	COMPL	B=COMPL(A)
Shift A1 by I bit positions: left circular if I is positive; right end off with sign extension if I is negative, $0 \leq \lvert I \rvert \leq 60$††	2	A1: any type† I: integer	no mode	SHIFT	B=SHIFT(A,I)
Mask by setting I bits to 1 starting at left of word. $0 \leq I \leq 60$††	1	Integer	no mode	MASK	A=MASK(I)
Obtain most significant part of double precision argument	1	Double	Real	SNGL	DOUBLE Y X=SNGL(Y)
Obtain real part of complex argument	1	Complex	Real	REAL	COMPLEX A B=REAL(A)
Obtain imaginary part of complex argument	1	Complex	Real	AIMAG	COMPLEX A D=AIMAG(A)
Express single precision argument in double precision form	1	Real	Double	DBLE	DOUBLE Y Y=DBLE(X)

†For a double precision or complex argument, only the high order or real part is used.

††Function is undefined if outside these bounds.

TABLE 4. FORTRAN INTRINSIC FUNCTIONS (Contd)

Definition	Arguments		Type of Function	Symbolic Name	Example
	Max. No.	Type			
Express two real arguments in complex form: A1+A2i where $i^2=-1$	2	Real	Complex	CMPLX	COMPLEX C C=CMPLX(A1,A2)
Obtain conjugate of complex argument: a-bi where A=a+bi	1	Complex	Complex	CONJG	COMPLEX X,Y Y=CONJG(X)
Return random numbers uniformly distributed over range (0,1); dummy argument is ignored	1	any type	Real	RANF	Y=RANF(A)
Obtain address of named variable or array element, or entry point of named external subprogram	1	any type	Integer	LOCF	J=LOCF(Q)

FORTRAN LIBRARY BASIC EXTERNAL FUNCTIONS

A basic external function is a predefined procedure (used to evaluate standard mathematical functions) included with the FORTRAN common library. Table 5 shows the FORTRAN library basic external functions.

TABLE 5. FORTRAN LIBRARY BASIC EXTERNAL FUNCTIONS

Definition	Arguments		Type of Function	Symbolic Name	Example
	Max. No.	Type			
Exponent: e to Ath power	1	Real	Real	EXP	Z=EXP(Y)
		Double	Double	DEXP	DOUBLE X,Y Y=DEXP(X)
		Complex	Complex	CEXP	COMPLEX A,B B=CEXP(A)
Natural logarithm of A	1	Real	Real	ALOG	Z=ALOG(Y)
		Double	Double	DLOG	DOUBLE X,Y Y=DLOG(X)
		Complex	Complex	CLOG	COMPLEX A,B B=CLOG(A)
Logarithm to base 10 of A	1	Real	Real	ALOG10	B=ALOG10(A)
		Double	Double	DLOG10	DOUBLE D,E E=DLOG10(D)
Trigonometric sine of A	1	Real	Real	SIN	Y=SIN(X)
		Double	Double	DSIN	DOUBLE D,E E=DSIN(D)

TABLE 5. FORTRAN LIBRARY BASIC EXTERNAL FUNCTIONS (Contd)

Definition	Arguments		Type of Function	Symbolic Name	Example
	Max. No.	Type			
Trigonometric sine of A (cont)		Complex	Complex	CSIN	COMPLEX CC,F CC=CSIN(F)
Trigonometric cosine of A	1	Real	Real	COS	X=COSY(Y)
		Double	Double	DCOS	DOUBLE D,E E=DCOS(D)
		Complex	Complex	CCOS	COMPLEX CC,F CC=CCOS(F)
Hyperbolic sine of A	1	Real	Real	SINH	B=SINH(A)
		Double	Double	DSINH	DOUBLE D,E E=DSINH(D)
Hyperbolic cosine of A	1	Real	Real	COSH	B=COSH(A)
		Double	Double	DCOSH	DOUBLE D,E E=DCOSH(D)
Hyperbolic tangent of A	1	Real	Real	TANH	B=TANH(A)
		Double	Double	DTANH	DOUBLE D,E E=DTANH(D)
Error Function	1	Real	Real	ERF	A1=ERF(A)
Complementary Error Function	1	Real	Real	ERFC(A)	Y=ERFC(X)
Hyperbolic Artangent of A where $\|A\|<1$	1	Real	Real	ATANH	Y=ATANH(D)
Trigometric Sine in degrees of A where $\|A\|<2^{47}$	1	Real	Real	SINH	Z=SINH(C)
Trigometric Cosine in degrees of A where $\|A\|<2^{47}$	1	Real	Real	COSD	W=COSD(E)
Trigometric Tangent in degrees of A where $\|A\|<2^{47}$	1	Real	Real	TAND†	Y=TAND(X)
Square root	1	Real	Real	SQRT	Y=SQRT(X)
		Double	Double	DSQRT	DOUBLE D,E E=DSQRT(D)
		Complex	Complex	CSQRT	COMPLEX CC,F CC=CSQRT(F)

TABLE 5. FORTRAN LIBRARY BASIC EXTERNAL FUNCTIONS (Contd)

Definition	Arguments		Type of Function	Symbolic Name	Example
	Max. No.	Type			
Arctangent of A	1	Real Double	Real Double	ATAN DATAN	Y=ATAN(X) DOUBLE D,E E=DATAN(D)
Arctangent of A1/A2	2	Real Double	Real Double	ATAN2 DATAN2	B=ATAN2(A1,A2) DOUBLE D,D1,D2 D=DATAN2(D2,D2)
Remaindering††	2	Double	Double	DMOD	DOUBLE DM,D1,D2 DM=DMOD(D1,D2)
Modulus: square root (a^2+b^2) for A=a+bi	1	Complex	Real	CABS	COMPLEX C CM=CABS(C)
Arccosine of A	1	Real Double	Real Double	ACOS DACOS	X=ACOS(Y) DOUBLE D,E D=DACOS(E)
Arcsine of A	1	Real Double	Real Double	ASIN DASIN	X=ASIN(Y) DOUBLE D,E D=DASIN(E)
Trigonometric tangent of A	1	Real Double	Real Double	TAN DTAN	X=TAN(Y) DOUBLE E,D E=DTAN(D)

†The argument for TAND must not be an odd multiple of 90.

††DMOD (a,b) is defined as a-[a/b]b, where [X] is the largest integer that does not exceed the magnitude of X with sign the same as X.

LIBRARY SUBROUTINES AND FUNCTIONS

The following utility subprograms are supplied by the system. The DATE, IDATE, SECOND, TIME and CLOCK routines can be used as functions or subroutines. The value for these routines is always returned via the argument and the normal function return.

DATE(a) or CALL DATE (a)	Returns current date in format Δmm/dd/yyΔ
JDATE(a) or CALL JDATE (a)	Returns current date in format 5Ryyddd. (not available under SCOPE 2.1)
SECOND(t) or CALL SECOND (t)	Returns elapsed central processor time as real number
TIME(a) or CALL TIME (a) CLOCK(a) or CALL CLOCK(a)	Returns current time in format Δhh.mm.ssΔ
CALL DISPLA (H,k)	Places 1-80 character Hollerith message H and value from expression or variable k in dayfile
CALL REMARK (H)	Places 1-80 character Hollerith message H in dayfile; 1-90 characters for SCOPE 2.1
CALL SLITE(i)	Turns on sense light 1-6; 0 turns off all
CALL SLITET(i,j)	Sets j=2 if sense light i is off, j=1 if on
CALL SSWTCH (i,j)	Sets j=2 if switch i is off; j=1 if on
CALL OVERLAY (fname,i,j,recall,k)	Calls an overlay
CALL EXIT	Terminates program
CALL CHEKPTX (filelist,n)	Takes a checkpoint dump of all files if n is zero; if n is nonzero a checkpoint dump is taken of all the files specified by filelist
CALL RECOVR (subr,flags,ck)	Calls subr on abnormal termination (not available under SCOPE 2.1)
CALL DUMP ($a_1,b_1,f_1,\ldots,a_n,b_n,f_n$)	Dumps storage and terminates program execution
CALL PDUMP ($a_1,b_1,f_1,\ldots,a_n,b_n,f_n$)	Dumps storage and returns control to calling program

a,b first and last word of storage area to be dumped

f=0 or 3, octal dump
f=1, real dump
f=2, integer dump

Adding 4 to any f values causes the values of a and b to be used as addresses.

CALL STRACE	Prints traceback
LEGVAR (a)	Checks variables: a -1 is indefinite, +1 is out of range, and 0 is normal
CALL SYSTEM (num,msg)	Prints error message and aborts if fatal error
CALL SYSTEMC (num,speclist)	Prints non-standard error message
CALL LIMERR(lim)	Enables user to input data without risk of termination up to the limit of lim
NUMERR(n)	Returns the number of errors since last LIMERR call; n is a dummy argument
CALL RANSET (n)	Sets initial value of RANF seed to n
CALL RANGET (nam)	Puts current seed of RANF in nam
CALL OPENMS (u,ix,lngth,t)	Opens mass storage file
CALL READMS (u,fwa,n,k)	Transmits data from mass storage to central memory
CALL WRITMS (u,fwa,n,k,r,s)	Transmits data from central memory to mass storage
CALL STINDX (u,ix,lngth,t)	Changes file index in central memory to base specified in call
CALL CLOSMS (u)	Writes index from central memory to file and closes file

u	unit designator
ix	Name of the array containing the master index
lngth	Length of index buffer: Number index, lngth $\geq$ entries in master index + 1; name index, lngth $\geq$ 2* maximum number of entries in master index + 1
t	Index type: 0 is number index; 1 is name index
fwa	First word address in central memory of data buffer area
n	Number of 60-bit words in data record to be transferred
k	Record key: Number index $1 \leq k \leq$ lngth-1; name index, k can refer to any 60-bit quantity except ± 0
r	Rewrite in place request: +1 is rewrite; -1 is rewrite if new record length $\leq$ old record length; otherwise, write at end-of-information; 0 is write at end-of-information (default). Can be omitted if no subindex flag parameter is required
s	Subindex flag, may be omitted; 0 is flag is not included (default value). 1 is write subindex marker flag in control word for record

UNIT (u)	Returns buffer status on unit u: -1 is unit ready, no error; +0 is unit ready, EOF encountered; +1 is unit ready, parity error encountered
EOF (u)	Checks for end of file; if zero, no end of file encountered
IOCHEC (u)	Returns parity status on non-buffer unit; if zero, no read parity error
LENGTH (u) or CALL LENGTHX (u,nw,ubc)	Returns number of words read and unused bit count on previous buffer or mass storage input/output request
CALL LABEL (u,labinfo)	Sets tape label information; labinfo is a 4-word array containing label information
CALL MOVLEV (a,b,n)	Transfers n consecutive words of data between extended core storage, central memory, SCM, or LCM. a is starting address of the data to be moved and b is starting address of receiving location.
CALL READEC (a,b,n)	Transfers n consecutive words of data beginning with a in central memory and b in extended core storage or LCM block.
CALL WRITEC (a,b,n)	Transfers n consecutive words of data beginning with a in central memory and b in extended core storage or LCM block.
CALL CONNEC (u) or CALL CONNEC (u,cs) cs character set	Connects file to a terminal (not available under SCOPE 2.1)
CALL DISCON (u)	Disconnects file from a terminal (not available under SCOPE 2.1)

A job deck submitted for execution through a card reader begins with a job statement and ends with an end-of-information card (a 6/7/8/9 multipunch in column 1. The deck is divided into sections by an end-of-record card (a 7/8/9 multipunch in column 1). Refer to the appropriate operating system reference manual for information concerning control statements.

COMPILING AND EXECUTING A FORTRAN PROGRAM

The following deck structure is used for simple compilation and execution of a FORTRAN source program.

```
Job statement
USER statement              NOS only
CHARGE statement            NOS only
ACCOUNT statement           NOS/BE and SCOPE 2 only
FTN.
LGO.
7/8/9
     PROGRAM MAIN
          .
          .
          .
     END
     FUNCTION RTSM(A,B)
          .
          .
          .
     END
     SUBROUTINE RUN(C)
          .
          .
          .
     END
7/8/9

     Data used in execution

6/7/8/9
```

COMPILING IN TIMESHARING MODE

The following deck structure is used when TS compilation mode is desired.

Job statement	
USER statement	NOS only
CHARGE statment	NOS only
ACCOUNT statment	NOS/BE and SCOPE 2 only
FTN(TS)	Timesharing mode is requested for compilation

7/8/9

FORTRAN source deck

6/7/8/9

COMPILING A FORTRAN PROGRAM AND PRODUCING BINARY CARDS

The following deck structure is used when the OPT=2 (full optimization) option of the FTN control card is desired.

Job statement	
USER statement	NOS only
CHARGE statement	NOS only
ACCOUNT statement	NOS/BE and SCOPE 2 only
FTN(B=PUNCHB,OPT=2)	The file PUNCHB is punched at end-of-job

7/8/9

FORTRAN source deck

6/7/8/9

LOADING AND EXECUTING A BINARY PROGRAM

The following deck structure is used to load and execute a program composed of binary object code.

Job statement	
USER statement	NOS only
CHARGE statment	NOS only
ACCOUNT statement	NOS/BE and SCOPE 2 only
MAP(OFF)	
INPUT.	
7/8/9	
Binary deck	
6/7/9	End-of-file; terminates load (NOS only)
7/8/9	{ Empty record; terminates load (NOS/BE and SCOPE 2 only)
7/8/9	
Data used during execution	
6/7/8/9	

COMPILING AND EXECUTING WITH DIFFERENT DATA DECKS

The following deck structure is used to execute a program using two different data decks. The program is compiled only once but executes twice.

```
Job statement
USER statement              NOS only
CHARGE statement            NOS only
ACCOUNT statement           NOS/BE and SCOPE 2 only
FTN.

LGO,,TAPE1.                 { Output is written on two separate files;
LGO,,TAPE2.                 { data 1 is written on TAPE1, data 2 is
                            { written on TAPE2.

7/8/9
     PROGRAM MAIN(INPUT,OUTPUT)
          .
          .
          .
     END
7/8/9

     Data 1 used during first execution.

7/8/9

     Data 2 used during second execution.

6/7/8/9
```

COMPILING AND EXECUTING A FORTRAN SUBROUTINE AND A COMPASS SUBPROGRAM

The following deck structure is used when compiling and executing a FORTRAN main program, FORTRAN subroutine, and a COMPASS subprogram. FORTRAN and COMPASS program unit source decks can be in any order. COMPASS source decks must conform with the format required by the COMPASS language.

```
Job statement
USER statement              NOS only
CHARGE statement            NOS only
ACCOUNT statement           NOS/BE and SCOPE 2 only

FTN(EL=A)                   { All diagnostics including ANSI are listed
                            { on file OUTPUT

LGO.
7/8/9

    PROGRAM DONE(INPUT,OUTPUT)
        .
        .
        .
    END
    SUBROUTINE S1(P1,P2)
        .
        .
        .
    RETURN
    END
        IDENT   SUB
        ENTRY   A1
                .
                .
                .
        END

7/8/9

    Data used during execution

6/7/8/9
```

COMPILING AND EXECUTING WITH A RELOCATABLE BINARY PROGRAM

The following deck structure is used when a FORTRAN source program is compiled and executed with a relocatable binary deck.

Job statement	
USER statement	NOS only
CHARGE statement	NOS only
ACCOUNT statement	NOS/BE and SCOPE 2 only
FTN.	
LOAD(LGO)	
LOAD(INPUT)	
EXECUTE.	
7/8/9	
FORTRAN source program	
7/8/9	
Binary deck	
6/7/9	End-of-file; terminates load (NOS only)
7/8/9	Empty record; terminates load (NOS/BE and SCOPE 2 only)
7/8/9	
Data used during execution	
6/7/8/9	

COMPILING AND EXECUTING WITH OVERLAYS

The following sample deck structure is used when compilation and execution of a program with overlays is desired.

```
Job statement
USER statement              NOS only
CHARGE statement            NOS only
ACCOUNT statement           NOS/BE and SCOPE 2 only
FTN.
LOAD(LGO)
NOGO.
SUM.
7/8/9

    OVERLAY(SUM,0,0)                    Main overlay source deck

    PROGRAM LEO(INPUT,OUTPUT)
        .
        .
        .
    CALL GROUND(40,0)
    CALL OVERLAY(3HSUM,1,0)             Call to primary overlay
        .
        .
        .
    END
    SUBROUTINE GROUND(x,y)
        .
        .
        .
    END
    OVERLAY(SUM,1,0)                    Primary overlay source deck

    PROGRAM RDY
        .
        .
        .
    CALL OVERLAY(3HSUM,1,1)             Call to secondary overlay

    END
    OVERLAY(SUM,1,1)                    Secondary overlay source deck

    PROGRAM MLT
        .
        .
        .
    END
7/8/9

    Data used during execution

6/7/8/9
```

EXTERNAL DEBUGGING DECK

The following deck structure is used when debugging of all program units is desired. The external debugging deck is placed immediately in front of the first source line of the first program unit. All program units (here, Program A and Subroutine B) are debugged unless limiting bounds are specified in the deck.

```
Job statement
USER statement              NOS only
CHARGE statement            NOS only
ACCOUNT statement           NOS/BE and SCOPE 2 only
FTN(D)
LGO.
7/8/9
C$  DEBUG
        .
        .                  } External debugging deck
        .
    PROGRAM A
        .
        .
        .
    END
    SUBROUTINE B
        .
        .
        .
    RETURN
7/8/9

    Data used during execution

6/7/8/9
```

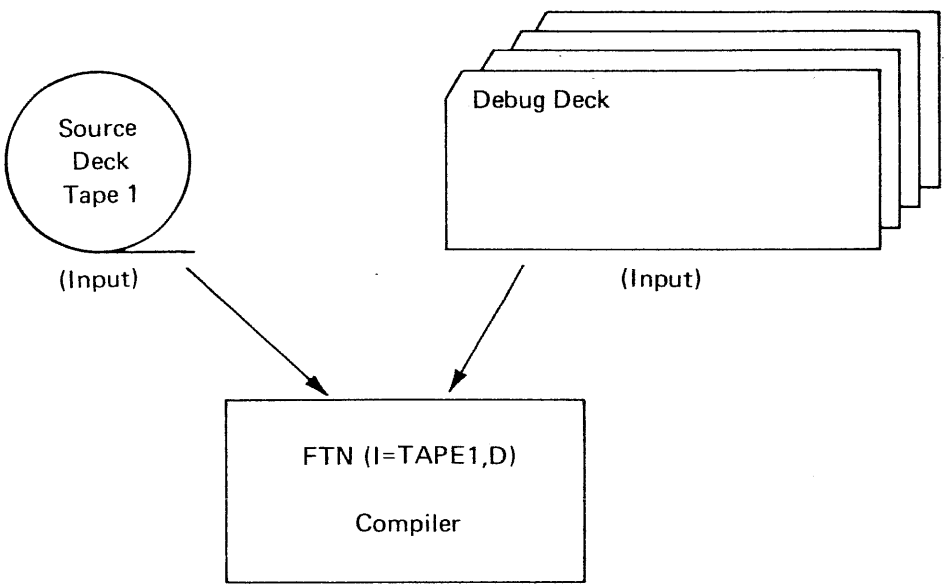
INTERSPERSED DEBUGGING STATEMENTS

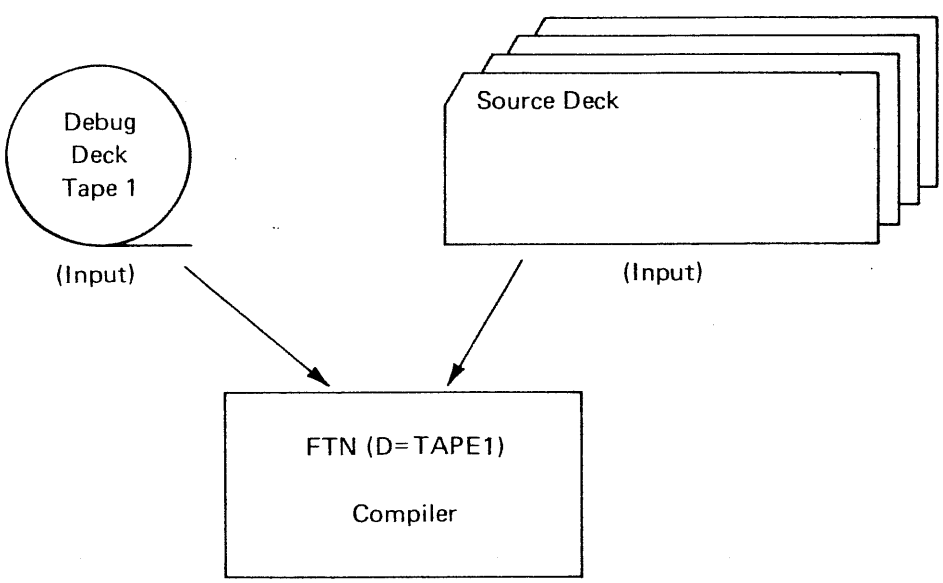
The following sample deck structure is used when interspersed debugging statements are desired. Debugging statements are inserted at the point in the program where they are activated.

```
Job statement
USER statement                  NOS only
CHARGE statement                NOS only
ACCOUNT statement               NOS/BE and SCOPE 2 only
FTN(D)
LGO.
7/8/9
     PROGRAM MAIN
     Specification statements
          .
          .
          .
     Executable statements
C$   FUNCS
C$   CALLS                      DEBUG statements
     Executable statements
          .
          .
          .
C$   STORES(A)
C$   OFF(FUNCS)                 DEBUG statements
     Executable statements
          .
          .
          .
7/8/9

     Data used during execution

6/7/8/9
```

EXTERNAL DECK ON SEPARATE FILE

The debugging deck is on file INPUT and the source deck on a named file. Alternatively, the debugging deck is placed on a separate file (external debugging deck) named by the D parameter on the FTN control statement and called in during compilation. All program units will be debugged (unless the program units to be debugged are specified in the deck). This positioning is useful when several jobs can be processed using the same debugging deck.





SORT/MERGE 4 AND 1 INTERFACE

Sort/Merge 4 and 1 processing can be initiated through FORTRAN Extended call statements. Sort/Merge 4 and 1 uses the unused part of field length as a scratch area; if necessary, additional field length is obtained from the system. For this reason, the STATIC option of the FTN control statement must not be specified with programs using Sort/Merge 4 and 1. Refer to the Sort/Merge 4 and 1 Reference Manual for detailed information.

The following statements initiate Sort/Merge 4 and 1 processing.

CALL SMSORT(mrl)	Sort-only or sort and merge processing.
CALL SMSORTB(mrl,ba)	Balanced tape sort under NOS and NOS/BE only.
CALL SMSORTP(mrl,ba)	Polyphase tape sort under NOS and NOS/BE only.
CALL SMMERGE(mrl,ba)	Merge only processing.

mrl Maximum record length in characters of record to be sorted.

ba Optional number of words of central memory used by Sort/Merge for working storage; applies only to NOS and NOS/BE (default 22000_8)

Optional total in decimal of large core memory used as a buffer area for all intermediate scratch files applies only to SCOPE 2.

CALL SMFILE(dis,i/o,lfn,action)

One call to SMFILE must be issued for each file unless the output file is handled by SMOWN.

dis File processing; specified as follows:

"SORT"	File to be sorted.
"MERGE"	File to be merged.
"OUTPUT"	File to receive output.

i/o Type of input/output; specified as follows:

"FORMATTED" "CODED"	File acessed with formatted input/ output.
"BINARY"	File accessed with unformatted input/ output.
0	File accessed with CYBER Record Manager interface routines

lfn Logical file name; can either be a unit number or the file name in nL_filename format. If i/o is specified as zero , lfn can be an array containing the file information table.

action File disposition after sort or merge; specified as follows:

"REWIND"

"UNLOAD"

"NONE" (default)

CALL SMKEY(charpos,bitpos,nchar,nbits,code,colseq,order)

SMKEY must be called once for each sort key.

charpos Integer specifying the starting position of sort key counting from 1.

bitpos Integer specifying the first bit of charpos counting from 1.

nchar Number of characters in sort key.

nbits Number of bits in excess of nchar.

The remaining three parameters are optional.

code Code used to interpret keys; specified as follows:

"DISPLAY"	Internal display code
"FLOAT"	Floating point data
"INTEGER"	Signed integer data
"LOGICAL"	Unsigned integer data (default)

The following identifiers must be separated by commas as indicated.

"DISPLAY", "SIGN", "LEADING"	Numeric data in display code; sign present as overpunch at beginning of field (NOS and NOS/BE only).
"DISPLAY", "SIGN", "TRAILING"	Numeric data in display code; sign present as overpunch at end of field.
"DISPLAY", "SEPARATE", "LEADING"	Numeric data in display code; sign is a separate character at beginning of field (NOS and NOS/BE only).
"DISPLAY", "SEPARATE", "TRAILING"	Numeric data in display code; sign is a separate character at end of field (NOS and NOS/BE only).

colseq Name of user-supplied collating sequence; can be used only if code is "DISPLAY". Possible collating sequences are the following:

"ASCII6"	ASCII 6-bit collating sequence.
"COBOL6"	COBOL 6-bit collating sequence.
"DISPLAY"	Internal display code collating sequence.
"INTBCD"	Internal BCD collating sequence.
Seqnam	Collating sequence defined by SMSEQ

order Specifies order of sort processing as follows:

"A"	Ascending (default)
"D"	Descending

CALL SMSEQ(seqnam,seqspec)

SMSEQ defines a user's collating sequence.

seqnam Name of user-supplied collating sequence

seqspec Name of integer array specifying the characters to be collated. Each character should be in either nRf or octal format. The array must be terminated by a negative number.

CALL SMEQU(colseq,equspec)

SMEQU defines two or more characters in a collating sequence as equal for comparison purposes.

colseq Name of the collating sequence; determined by a previous call to SMKEY or SMSEQ.

equspec Name of integer array specifying the characters to be collated. Each character should be in either nRf or octal format. The array must be terminated by a negative number.

CALL SMOPT(optlist)

SMOPT specifies special record handling options and must be called immediately after SMSORT or SMMERGE. If SMOPT is called more than once the last call overrides all previous calls.

optlist Non-ordered series of options separated by commas; specified as follows:

"VERIFY"	Check output for correct sequencing
"RETAIN"	Retain records with identical sort keys in order of appearance on input file.

"VOLDUMP"	Checkpoint dump at end-of-volume (NOS and NOS/BE only).
"DUMP"	Checkpoint dump after 50000 records (NOS and NOS/BE only).
"DUMP",n	Checkpoint dump after n (decimal) records (NOS and NOS/BE only).
"NODUMP"	No checkpoint dump (NOS and NOS/BE only).
"NODAY"	Suppress dayfile messages (NOS and NOS/BE only).
"ORDER",mo	Determines intermediate merge order (mo); $2 \leq mo \leq 64$ (NOS and NOS/BE only).
"COMPARE"	Use key comparison technique (NOS and NOS/BE only).
"EXTRACT"	Use key extraction technique (NOS and NOS/BE only).

CALL SMOWN($exnum_1$,$subname_1$, . . . ,$exnum_n$,$subname_n$)

$exnum_i$ Number of the owncode exit.

$subname_i$ Name of the user-supplied EXTERNAL owncode exit subroutine which exits through a call to system subroutine SMRTN.

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.

No parameters are needed for exit number 1 if no input files are used.

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
a,b	Integer array of rl/10 words that contains the record stored when subname is called.
rl	Record length in characters.

CALL SMTAPE(taplist)

SMTAPE specifies all magnetic tape intermediate merge files (NOS and NOS/BE only).

taplist List of names assigned to intermediate merge files.

CALL SMEND

SMEND is required to initiate execution of Sort/Merge 4 and 1.

CALL SMABT

SMABT terminates a sequence of Sort/Merge 4 and 1 calls without execution of Sort/Merge.

CYBER RECORD MANAGER INTERFACE

The FORTRAN user can access CYBER Record Manager facilities directly by calling the following subprograms. File names used in the calls must not appear in a PROGRAM statement. FIT field mnemonics used to represent parameters in the CALL statements are the same as the FIT field mnemonics defined in the CYBER Record Manager reference manual. These subprograms are not callable through SCOPE 2.1. Except for CALL FILExx, the order of parameters is fixed so that all parameters to the left of a desired option must be specified. Refer to either the CYBER Record Manager Basic Access Methods Reference Manual or to the CYBER Record Manager Advanced Access Methods Reference Manual for more detailed information.

CALL CHECK (fit)	Determines I/O status for SQ and WA file types.
CALL CLOSEM (fit,cf,typ)	Closes file for all file types.
CALL DLTE (fit,ka,kp,0,ex)	Deletes record for IS, DA, and AK file types.
CALL ENDFILE (fit)	Writes end-of-partition for SQ file type.
CALL FILExx (fit,keyword$_1$,value$_1$,..., keyword$_n$,value$_n$)	Establishes FIT for all file types.
CALL FITDMP (fit,d)	Dumps the contents of the FIT to error file ZZZZEG.
CALL FLUSH (afit)	Performs all file close operations but the files remain open.
CALL FLUSH1 (fit)	Performs all the file close operations for a single file. The file remains open.
CALL GET (fit,wsa,{ka / wa},kp,mkl,{1 / rl}, {ex / dx})	Reads record for all file types.
CALL GETN (fit,wsa,ka,ex)	Reads next record for IS, DA, and AK file types.
CALL GETP (fit,wsa,ptl, 4LSKIP,dx)	Reads partial record for SQ and WA file types.
CALL OPENM (fit,pd,of)	Opens file for all file types.
CALL PUT (fit,wsa,rl,{ka / wa},kp,pos,ex)	Writes record for all file types.
CALL PUTP (fit,wsa,ptl,rl,ex)	Writes partial record for SQ and WA file types.
CALL REPLC (fit,wsa,rl,ka,kp,0,ex)	Replaces record for SQ, IS, DA, and AK file types. The parameters rl, ka and kp must be specified as zero for SQ file type.
CALL REWND (fit)	Rewinds file for SQ, IS, DA, and AK file types.

CALL RMKDEF (fit,rkw,rkp,kl,0,kf, ks,kg,kc,nl,ie,ch)	Defines primary or alternate key field in a record for IS, DA, and AK multiple index file types.
CALL SEEKF (fit,ka,kp,mkl,ex)	Initiates record search for IS, DA, and AK file types.
CALL SKIP (fit,±count)	Repositions file for SQ, IS, and AK file types.
CALL STARTM (fit,ka,kp,mkl,ex)	Positions an IS or AK file to a record that meets a specific condition.
CALL STOREF (fit,keyword,value)	Sets FIT value in field named for all file types.
CALL WEOR (fit,lev)	Terminates S type record after PUTP, writes end-of-section or end-of-partition for SQ file type.
CALL WTMK (fit)	Writes tape mark for SQ file type.
IFETCH (fit,keyword)	Integer function. Retrieves contents of FIT field named for all file types.

xx — File organization mnemonic:

SQ	sequential files
WA	word addressable files
IS	indexed sequential files
DA	direct access files
AK	actual key files

Multiple index processing is allowed for IS, DA, and AK file organizations.

fit — Name of the 35-word array containing the file information table (FIT)

afit — Name of the array containing a list of addresses of FITS; the array must be terminated by a word of zeros.

keyword — Name of a FIT field in L format

value — Value for FIT field specified by keyword

pd — Type of processing:

5LINPUT	read only
6LOUTPUT	write only
3LI-O	read and write
3LNEW	IS, AK, or DA file to be created (write only)

of — Open flag; specifies position of file when opened as follows:

- 1LR — Rewind.
- 1LN — No file positioning.
- 1LE — File is positioned just before end-of-information.

cf — Close flag; specifies position of file after close as follows:

- ILR — Rewind.
- 1LN — No rewind.
- ILU — Unload.
- 3LRET — Return.
- 3LDIS — Disconnect (terminal files only).
- 3LDET — No positioning; release buffer space and remove from active file list.

typ — Type of close to be performed on SQ files only; omitted for all other file organizations:

- 6LVOLUME — Volume close.
- 4LFILE — File close.

wsa — Working storage area.

ka — Address of key for DA, AK, or IS records.

wa — Word address for read or write.

kp — Beginning character position of key within ka; 0-9.

mkl — Major key length on IS files; symbolic key type.

rl — Record length in characters; 1 indicates positioning rather than length.

ex — Name of user owncode error exit subroutine; must be specified in an EXTERNAL statement.

dx — Name of end-of-data exit subroutine (EXTERNAL).

pos — Duplicate key processing; indicates action to be taken when a duplicate key is encountered:

- 0 — First record in a duplicate key chain will be deleted or replaced.

count — Number of logical records to be skipped; positive for forward skip, negative for backward skip.

lev — Level number for end-of-section; 0 to 17_8.

ptl — Partial transfer length in characters.

4LSKIP — Skip to beginning of next record before reading.

nl Null suppression: 0 = null values recorded (default).

ie Include/exclude sparse control character; specified as follows:

- E Exclude alternate key value
- I Include alternate key value

ch Characters that qualify as sparse control characters (maximum 36).

id FIT identifier.

rkw Relative keyword (0 = first word).

rkp Relative key position (0 = keyword aligned starting at rkw position.

kl Key length in characters (1-255 for symbolic key; 10 for signed key).

kf Key type: 0 = symbolic; 1 = signed integer; 2 = unsigned integer.

ki Summary index; reserved 0.

ks Optional: substructure for each primary key list in the index: I = index-sequential; F = FIFO; U (default) = unique; can be specified as L format Hollerith constant. If ks is specified, kg and kc must be defined.

- kg size of repeating group in which key resides (default = 0), or the index block size.
- kc occurrences of group (default = 0).

POST MORTEM DUMP

Post Mortem Dump (PMD) analyzes the execution-time errors in FORTRAN programs. To use PMD, the PMD parameter must be specified on the FTN control statement. PMD is then activated by a fatal error or by one of the user-callable subroutines PMDLOAD or PMDSTOP.

CALL PMDARRY(i,j,k)	Prints one, two, and three dimensional arrays when an abort, PMDLOAD or PMDSTOP is called; i, j, and k represent the first, second, and third dimensions respectively.
CALL PMDDUMP	Causes a dump of variables in the calling routine when an abort or PMDLOAD or PMDSTOP is called.
CALL PMDLOAD	Causes an immediate dump of variables in the calling routine, all routines in the traceback chain, and any routines that have called PMDDUMP.

CALL PMDSTOP	Causes an immediate dump of variables in the calling routine, all routines in the traceback chain, and any routines that have called PMDDUMP. The job is aborted.

COMMON MEMORY MANAGER INTERFACE

All Common Memory Manager (CMM) interfaces for NOS and NOS/BE are on the library SYMLIB. For a run using CMM interface routines, the user must either include the loader statement LDSET(LIB=SYMLIB) in the loader directives, or include a CALL SYMLIB subroutine call in the main program. SCOPE 2 users must specify SYMIO in the LDSET statement instead of SYMLIB.

CALL CMMALF(ibksz,iszcde,igrpid,iblfwa)	Allocates a fixed position block
CALL CMMFRF(iblfwa)	Frees a fixed position block when no longer needed

iblksz — Number of words required for block

iszcds — Size code:

0	Fixed size block
1	Block can grow at last word address
2	Block can shrink at last word address
4	Block can shrink at first word address
5	Block can grow at first word address and shrink at last word address
6	Block can shrink at first word address and last word address
7	Block can shrink at first word address and last word address and grow at last word address

igrpid — Group identifier:

0	Item does not belong to a group
>0	The block is assigned to this group. (See Common Memory Manager Reference Manual.)

iblfwa — The first word address of a block allocated by CMM, value returned by a call to CMMALF.

FORTRAN-CYBER INTERACTIVE DEBUG INTERFACE

CYBER Interactive Debug (CID) is a debugging facility, available under NOS and NOS/BE only, which allows the user to monitor and control program execution from an interactive terminal. Debug tables must be produced during compilation to make all of the CID features available for use with FORTRAN programs. Debug tables are created if, prior to compilation, debug mode is turned on via the DEBUG control statement.

DEBUG or DEBUG(ON)	Turns on debug mode
DEBUG(OFF)	Turns off debug mode

Alternatively, debug tables can be created by compiling the program using the DB=ID or DB option of the FTN control statement. If debug tables are not created during compilation, CID can be utilized by turning debug mode on via the DEBUG control statement prior to execution; however, not all of the CID features are available. Refer to the CYBER Interactive Debug Reference Manual for more information about CID.

FORTRAN CONTROL STATEMENT

The control statement FTN4 can be used interchangeably with the following forms of the FTN control statement:

FTN.comments

FTN($p_1, \ldots, p_n$)comments

FTN,$p_1, \ldots, p_n$.comments

The optional parameters, p_i, can appear in any order. All parameters must be separated by commas.

FORTRAN CONTROL STATEMENT PARAMETERS

- A EXIT

A=0 omitted	System advances to the next control statement whether or not fatal a compile-time error occurs.
A	System branchs to the next EXIT or teminates if a fatal compile-time error occurs.

- B BINARY OBJECT CODE FILE

B omitted	Generated binary object code is placed on file LGO.
B=lfn	Generated binary object code is placed on file lfn.
B=0	No binary object code is produced.

- BL BURSTABLE LISTING

BL=0 omitted	Output listing is generated in compact form.
BL	Page ejects are generated between parts of output listing.

- C COMPASS ASSEMBLY

  C=0 or omitted — The FORTRAN internal assembler is selected.

  C — The COMPASS assembler is selected.

- CC CONTROL STATEMENT CONTINUATION

  CC=0 or omitted — The FTN control statement appears on one line only.

  CC — The FORTRAN compiler interprets the following control statement as a continuation of the FTN control statement.

- D DEBUGGING MODE PARAMETER

  D=0 or omitted — Debug statements are ignored.

  D=lfn — Debug statements are on file lfn.

  D — Debug statements are on file INPUT.

- DB CYBER INTERACTIVE DEBUG

  DB=0 or omitted — No debug tables are created.

  DB=ID or DB — Debug tables are created. This option must be specified if the DEBUG control statement is not specified but the program is to be debugged using CID.

- E EDITING

  E=0 or omitted — Object file is generated in normal binary code.

  E=lfn — Object file is output as COMPASS line images on file lfn.

  E — Object file is output as COMPASS line images on file COMPS.

- EL ERROR LEVEL

  EL=I or omitted — Informative and fatal diagnostics are listed if OPT=0, 1, or 2; note, warning and fatal diagnostics are listed if in TS mode.

  EL=A — All non-ANSI and fatal diagnostics are listed. Informative diagnostics are listed if OPT=0, 1, or 2; note and warning diagnostics are listed if in TS mode.

  EL=N — Fatal diagnostics are listed if OPT=0, 1, or 2; note, warning, and fatal diagnostics are listed if in TS mode.

EL=W — Fatal diagnostics are listed if OPT=0, 1, or 2; warning and fatal diagnostics are listed if in TS mode.

EL=F — Fatal diagnostics are listed.

- ER ERROR RECOVERY

ER=0 — Code is generated for object-time reprieve. (Default for OPT=1, 2)

ER — No code is generated for object-time reprieve. (Default for TS and OPT=0)

- G SOURCE OF SYSTEM TEXT

G=0
omitted — System text loading from sequential binary file is prevented.

G=lfn — First sytem text overlay is loaded from file lfn.

G=lfn/ovl — System text overlay is loaded from file lfn.

G — System text overlay is loaded from file SYSTEXT.

- GO LOAD AND GO

GO=0
omitted — The binary object file is not loaded and executed at the end of compilation.

GO — The binary object file is loaded and executed at the end of compilation.

- I SOURCE INPUT FILE

I=INPUT
omitted — Source code is on file INPUT.

I=lfn — Source code is on file lfn.

I — Source code is on file COMPILE.

- L LIST OUTPUT FILE

L
omitted — Listable output is written on file OUTPUT.

L=lfn — Listable output is written on file lfn.

L=0 — Fatal diagnostics and the statements that caused them are listed on the file OUTPUT. All other compile-time output is suppressed.

- LCM LEVEL 2 AND LEVEL 3 STORAGE ACCESS

  LCM=D or omitted — Direct mode (17-bit address mode) is selected for level 2 or 3 data.

  LCM=I — Indirect mode (21-bit address mode) is selected for level 2 or 3 data. TS mode always uses 21-bit addressing for level 2 or 3 data.

  Level 2 data applies only to the CYBER 170 Model 176, CYBER 70 Model 7600 computers.

- ML MODLEVEL MICRO

  ML or omitted — The current date in the form yyddd is used for the MODLEVEL micro.

  ML=n — The value specified by n (1-7 letters or digits) is used for the MODLEVEL micro.

- OL OBJECT LIST

  OL=0 or omitted — No object code is listed.

  OL — Object code is listed on list output file.

- OPT OPTIMIZATION

  OPT=1 or omitted — Standard compilation and execution are activated.

  OPT=0 — Fast compilation and the T and ER options are activated.

  OPT=2 or OPT — Fast execution is activated.

- P PAGINATION OF OUTPUT LISTING

  P=0 or omitted — Page numbering begins at 1 for each subprogram.

  P — Page numbering is continuous across program units.

- PD PRINT DENSITY

  PD=6 or omitted — Compile-time listings are 6 lines per inch.

  PD=8 or PD — Compile-time listings are 8 lines per inch.

- PL PRINT LIMIT

  PL=n — Maximum number of records that can be written to file OUTPUT at execution time is set by n. (Default PL=5000.)

- PMD POST MORTEM DUMP

  PMD=0 omitted — No symbol tables are created.

  PMD — Symbol tables that are accessed by the Post Mortem Dump facility are generated. PMD must be specified if the facility is to be used.

- PS PAGE SIZE

  PS=n — Maximum number of lines per page including headers for compiler listings is set by n. (Default PS=60.)

- PW PAGE WIDTH

  PW=n — Number of characters (50-136) on a line of listable output in TS mode is set by n. (Default PW=126.)

  PW — Number of characters on a line of listable output is set to 72. (Default for files connected to a terminal.)

- Q PROGRAM VERIFICATION

  Q=0 omitted — Normal compilation is produced.

  Q — The compiler performs a full syntactic scan of the program, but no object code is produced.

- R SYMBOLIC REFERENCE MAP

  R=1 omitted — A short map (symbols, addresses, properties, DO loop map) is produced.

  R=0 — No map is produced.

  R=2 R — A long map (short map plus reference by line number) is produced.

  R=3 — A long map plus a listing of common block numbers and equivalence classes is produced. In TS mode R=3 is identical to R=2.

- ROUND ROUNDED ARITHMETIC COMPUTATIONS

  ROUND=0 or omitted — Arithmetic computations are not rounded.

  ROUND=op — The hardware rounding features are selected for the arithmetic operators specified by op (+ - * /).

  ROUND — The hardware rounding features are selected for the arithmetic operations + - * /.

- S SYSTEM TEXT FILE

  S=ovl — System text overlay loaded from job's current library set.

  S=lib/ovl — The system text overlay named ovl is loaded from the library called lib which can be either a user library file or a system library.

  S=0 — No system text file is loaded if COMPASS is called to assemble any intermixed program. (Default if G≠0.)

  S — The system text overlay named SYSTEXT is loaded from the job's current library set. (Default if G=0.)

- SEQ SEQUENCED INPUT FILE

  SEQ=0 or omitted — Source code is in standard FORTRAN format.

  SEQ — Source code is in sequenced line format.

- SL SOURCE LIST

  SL or omitted — The source code is listed on the file specified by the L parameter.

  SL=0 — No source code is listed.

- STATIC STATIC LOADING

  STATIC=0 or omitted — Dynamic memory management is used at execution time; no LDSET directives are generated.

  STATIC — Dynamic memory management is inhibited; a set of LDSET and USE directives (NOS and NOS/BE only) are generated.

- SYSEDIT SYSTEM EDITING

  SYSEDIT=0 or omitted — All input/output references are accomplished directly.

  SYSEDIT — All input/output references are accomplished indirectly.

- T ERROR TRACEBACK

  T=0 or omitted — No traceback occurs when an error is detected.

  T — A full error traceback occurs when an error is detected.

- TS TIMESHARING MODE

  TS=0 or omitted — OPT=1 is selected.

  T — TS mode single pass compilation is selected.

- UO UNSAFE OPTIMIZATION

  UO=0 or omitted — Unsafe optimization is not performed.

  UO — Potentially unsafe operations are performed when OPT=2.

- X EXTERNAL TEXT NAME

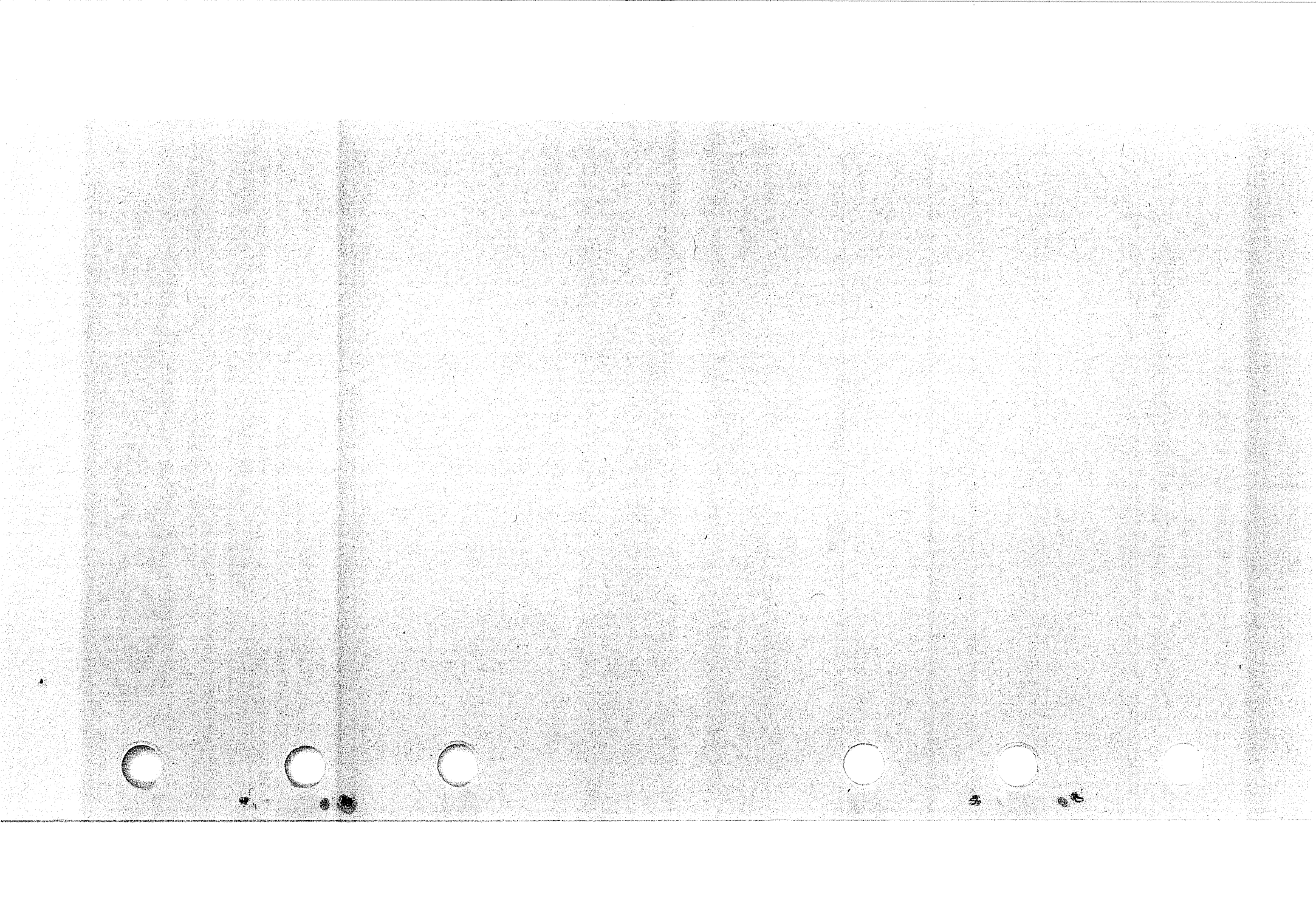
  X or omitted — The file OPL is the source of external text (XTEXT).

  X=lfn — The file lfn is the source of external text (XTEXT).

- Z ZERO

  Z=0 or omitted — No zero word parameter list is passed.

  Z — All subroutine calls having no parameters are forced to pass a parameter list consisting of a zero word.



CONTROL DATA CORPORATION

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

SALES OFFICES AND SERVICE CENTERS
IN MAJOR CITIES THROUGHOUT THE WORLD