HEWLETT hp PACKARD

Part No. 32102-90002
Product No. 32102B

Printed in U.S.A. SEP 1977

J. GEREMIN.

HP 3000 Computer Systems

FORTRAN POCKET GUIDE

HEWLETT hp PACKARD

PRINTING HISTORY

First Edition	Nov 1976
Second Edition	Sep 1977

CHARACTER SET

A-Z, 0-9, any printable ASCII character. Blanks ignored except in Hollerith and string constants.

VARIABLE NAMES

1 to 15 characters. First character must be letter.

FIXED FORMAT RULES

Statements:	Labels in cols. 1-5, text in cols. 7-72.
Continuation Lines:	Non-zero and non-blank col. 6, max. 19 continuation lines per statement.
Comments:	C in col. 1, text in cols. 2-72
Compiler Commands:	$ in col. 1, command in cols. 2-72, to continue use & as last character.
Sequencing:	Cols. 73-80.

FREE-FIELD FORMAT RULES

Maximum 80 characters per line or maximum 71 characters (plus sequence field) if begins in pos. 2.

Sequence field max. 8 characters (terminated by first blank). Must begin in pos. 1.

Comments:	# in first pos. after sequence field and blank.
Commands:	$ in first pos. after sequence field and blank except $CONTROL FREE which starts in pos. 1. To continue use & as last character.
Statement labels:	Max. value = 99999.

Text is first non-blank, non-numeric character after sequence field.

ARRAY DECLARATORS

Use with DIMENSION, COMMON, Type statements.	Example:
name $(b_1, \ldots, b_n)$	A(25,5,6)

DO-IMPLIED LISTS

Use in READ, WRITE, ACCEPT, and DISPLAY.	Example:
(*list, variable = init,limit,step*)	(X(J),Y(J),J=2+I,2*I,K)
(*list,variable = init,limit*)	(Z(I),I=1,9)

Note: *variable* must be integer or double integer simple variable.

PARTIAL WORD DESIGNATORS

operand [first bit:number of bits]

where

operand is integer or logical constant, variable, function reference, subexpression

first bit and *number of bits* are integer constants

Example: VAR [4:12]

SUBSTRING DESIGNATORS

name [*first character:number of characters*]

where

name is character variable

first character and *number of characters* are integer constants or variables.

Examples: CHAR1 [3:58] or CHAR1 [J:12]

DATA TYPES

Type	Examples	Size
INTEGER (or INTEGER*2)	+45 –365 %4777 –%17 +%"A8" –%'XY'	16 bits
INTEGER*4	2J –467J %46J –%777J %"DEC"J –%"X"J	32 bits
REAL	20.5 .205E2 205.E–1 2.4E+3 %37R %"V"R	32 bits
DOUBLE PRECISION	3D–6 1.73D4 –%3776D %45D %"ABCDE"D	64 bits*
COMPLEX	(3.0,–2.5E3) (%37R,%736R) (45J, (%[5/12] R, %[3/2] R)) Double Precision not allowed.	32 bits real 32 bits imaginary
LOGICAL	%177L %"AB"L % 3/7,4/7 L	16 bits
CHARACTER	"W"	8 bits
CHARACTER**x*	"WORD" 'ANOTHER WORD'	*x* 8-bit characters

* 48 for Series I

Composite Numbers:

%[3/9,4/12,2/1] L	%[2/%5,23/%6] R
%[4/15,6/%13,2/1]	J and D also allowed.

EXPRESSIONS

Type	Result	Conversion Hierarchy for Operands	Operators-Hierarchy
Arithmetic	Integer	Lowest	**
	Double Integer		* and /
	Real		+ and –
	Double Precision		
	Complex	Highest	
Logical	.TRUE.		
	.FALSE.		
	16-bit mask		
Character	Character		

Logical Expressions

Simple Logical: *operand*
operand relational operator operand

where

operand is constant, variable, function reference, subexpression, or arithmetic or character expression.

relational operator is .EQ., .NE., .LT., .LE., .GT., .GE.
(Expressions of complex numbers with .NE. and .EQ. only)

Complex Logical: *simple logical expression logical operator simple logical expression*

where

logical operator is .NOT., .AND., .XOR., .OR.

Hierarchy: *high* $\longrightarrow$ *low*

STATEMENTS

Form	Purpose
ACCEPT list	Input free-field data from any input file defined as logical unit 5. Prompts with ? if terminal. Example: ACCEPT A,B(2),(X(J),Y(J),J=1,5,2)
ASSIGN statement label TO variable	Assign **statement label** value (integer constant) to integer simple variable. Example: ASSIGN 210 TO IVAR
(Assignment statement see name)	
BACKSPACE unit	Position record pointer for **unit** file reference to preceding record. Example: BACKSPACE 12
BLOCK DATA **BLOCK DATA name**	First line of block data subprogram optionally identified by **name.** Example: BLOCK DATA BLOCKNUMBER1
CALL name **CALL name(param,param,. . param)** **CALL name(param,. . .param, $label,. . ,$lebel)**	References and transfers control to external procedure, with control optionally returned to specific statement labels. Example: CALL FORMS CALL FORMS(A,B,C) CALL FORMS (A,B,$30,$40)
CHARACTER list **CHARACTER *x list**	Assigns type character to variables, arrays, functions in **list.** Reserves memory space for arrays. Example: CHARACTER A(3), NAME*20 CHARACTER*4 A(3), NAME*20 CHARACTER Z*(I) ← **variable length must be surrounded by parentheses**
COMMON list **COMMON /blockname/list . . . /blockname/list**	Reserves block of storage space that can be referenced by several different program units. Only one unlabelled common block allowed.

Form	Purpose
	Example: COMMON A, B(10), XARRAY COMMON /XBLOCK/VAR,I2(20)/ /YBLOCKS/S, T, U//Z(3)
COMPLEX list	Assigns type complex to variables, arrays, functions in **list.** Reserves memory space for arrays. Example: COMPLEX JXVAR, RVAR
CONTINUE **label CONTINUE**	A null statement typically used as last statement in DO loop. Example: 410 CONTINUE
DATA list/d_1,d_2, . . . ,d_n/ ,list/d_1,d_2, . . . ,d_m/	Assigns initial values to data elements at load time. Constants must be same type as variable except real variable can have integer constant. Example: DATA A,B,C(3)/4.1,5.6,–.12/, X/5*1.5/
DIMENSION name(bounds), name(bounds), . . . , name(bounds)	Defines the dimensions and bounds of arrays. Example: DIMENSION ARR(3,4,2),XARR(9)
DISPLAY list	Prints values of data elements in list in free-field format to unit number 6. Example: DISPLAY A,B(2),(X(J),J=2,8,2)
DO label variable=init,limit, step **DO label variable=init,limit**	Controls execution of group of statements by causing statements to be repeated a certain number of times. Example: DO 120 I = 1,25,5 DO 20 J = 10,1,–1
DOUBLE PRECISION list	Assigns type double precision to variables, arrays, function in **list.** Reserves memory space for arrays. Example: DOUBLE PRECISION RVAR,D,X3

Form	Purpose
END	Informs compiler end of program unit's code has been reached. Example: END
ENDFILE unit	Writes end-of-file record on specified unit. Example: ENDFILE 12
ENTRY entryname **ENTRY entryname(parameter name,parameter name, . . . , parameter name)**	Specifies secondary entry points for function or subroutine. Example: ENTRY JMULT ENTRY XMULT(B1,C1)
EQUIVALENCE(list),(list), . . . (list)	Associates simple variables and array elements so share allocated storage space in same program unit. Example: EQUIVALENCE (DSETP(1),CS), (XX,YY,ZZ),(INAM,JNAM)
EXTERNAL name,name, . . name	Identifies function and subroutine names referenced in one program unit but defined in another. Example: EXTERNAL ALPHA,BETA,SOLVE
label FORMAT(format and/or edit specifications)	Describes how input or output information is to be formatted. See format and edit specification tables. Example: 210 FORMAT (I3,6F10.2) 220 FORMAT (3(E12.4),M12.1)
FUNCTION name(param, param, ... ,param) **type FUNCTION name(param, param, . . , param)**	Assigns a symbolic name to and specifies dummy parameters of a function, and optionally, its type. Example: FUNCTION RANDOM(X) INTEGER FUNCTION I1 (M,N)
GO TO label	Transfers control unconditionally to a given labeled statement. Example: GO TO 40

Form	Purpose
GO TO(label,label, . . .,label), index expression	Passes control to one of several labeled statements depending on result of **index expression.** Example: GO TO (100,200,240),X-24
GO TO variable **GO TO variable (label,label, . . . ,label)**	Passes control to statement label assigned to given integer or double integer variable. **Labels** have no effect on execution. Example: GO TO NEXT GO TO IXV(20,25,30)
IF (expression)label$_1$, label$_2$,label$_3$	Directs control to one of three statements. (**label$_1$** if **expression** $\leqslant$ 0, **label$_2$** if = 0, **label$_3$** if > = 0) Example: IF (A–B*3) 100,200,300
IF (logical expression) statement	Executes **statement** if result of **logical expression** is true. Example: IF (Z.LT.X.AND.Z.NE.2) Z=X+2.4
IMPLICIT type(letter, ... ,letter). . . type(letter, ... ,letter) **IMPLICIT type(letter-letter)**	Overrides or confirms the type associated with the first letter of a variable. Example: IMPLICIT INTEGER (A,B), CHARACTER(X) IMPLICIT INTEGER*4(J–M)
INTEGER list **INTEGER*2 list**	Assigns type integer to variables, arrays, functions in **list.** Reserves memory for arrays. Example: INTEGER XVAR, STOCKNO INTEGER*2 CLASSID
INTEGER*4 list	Assigns type double integer to variables, arrays, functions. Reserves memory for arrays. Example: INTEGER*4 XARR(3,5)

Form	Purpose
LOGICAL list	Assigns type logical to variables, arrays, functions in **list**. Reserves memory for arrays. Example: LOGICAL TESTVAL

name = expression	Assigns a value to the variable **name** which is result of **expression.**
	Example:
	X = A*D/RVAL**3
	Y = SQRT(Z/2300)

name(param,param, . . . ,param) = expression	Defines a statement function. Must occur before first executable statement and after all other declaration statements except DATA. Example: TAX(PRIC,DSCNT)= (PRIC-DSCNT) *.06
ON error condition CALL subroutine ON error condition ABORT ON CONTROLY CALL subroutine	Transfers control to subroutine or aborts if trap condition is encountered or Control Y is pressed. (See trap handling tables.) Example: ON INTEGER DIV 0 CALL FAIL ON SYSTEM ERROR ABORT ON FORMAT ERROR CALL FIXIT ON CONTROLY CALL CYSUB
PARAMETER name=constant, . . . name=constant	Assigns symbolic name to constant. Type determined by constant. Example: PARAMETER PI=3.1416,MULT=5
PAUSE PAUSE integer PAUSE "character string" PAUSE 'character string'	Causes program break in interactive mode, prints PAUSE and identifies by **integer** or prints message. Example: PAUSE PAUSE 6 PAUSE "CONTINUE?" PAUSE 'RESUME?'
PROGRAM name	Assigns a symbolic name to a main program. (Default name is MAIN.) Example: PROGRAM PROGX

Form	Purpose
READ(source,format, END=sn_1, ERR=sn_2) list	Transfers information from an external file to data elements in **list**, or from character variable to element in **list**. (–**source** = MPE file no.) Example: READ(3,200) IVAL,JVAL READ(3 @ 20,410) A READ(5,30,END=460,ERR=500) X,Y MPENO=–FOPEN (. . . . .) READ(MPENO,150) B
READ(source,*) list	Same as above for free-field input. Example: READ(5,*) A,IVAL,X(9)
READ(source) list READ(source@record) list	Transfers unformatted (binary) information from external file to data elements in **list** (sequentially or directly). Example: READ(9) A,B,C READ(9@IVAL) A,B,C
REAL list	Assigns type real to variables, arrays, functions in **list**. Reserves memory for arrays. Example: REAL INVERSX, JPERCNT

**RETURN
RETURN n**
	Transfers control from a subprogram back to calling program.
	Example:
	RETURN 2
	RETURN INTX

REWIND unit	Positions record pointer to first record of referenced file. Example: REWIND 12
(Statement function see name)	

Form	Purpose
STOP **STOP integer** **STOP "character string"** **STOP 'character string'**	Terminates program execution and prints **integer** or message. Example: STOP STOP 3 STOP "WRONG DEVICE" STOP 'BAD PASSWORD'
SUBROUTINE name **SUBROUTINE name(param, param, ... , param)**	Assigns a symbolic name to and specifies dummy parameters of a subroutine. Example: SUBROUTINE EVAL SUBROUTINE XSUB(I,RS1)
SYSTEM INTRINSIC name, . . . , name	Specifies that the SPLINTER file is to be searched for names appearing in list. Example: SYSTEM INTRINSIC BINARY, FOPEN
(Type statements see INTEGER, INTEGER*2, INTEGER*4, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL, CHARACTER.)	
WRITE(destination,format, END=sn$_1$, ERR=sn$_2$) list	Transfers information from data elements in **list** to external file or character variable in memory. (–**destination** = MPE file no.) Example: WRITE(6,100,END=500,ERR=999) XARRAY, Z WRITE(ZIP,150) "94040" MPENO=–FOPEN(. . . . .) WRITE(MPENO,250) B
WRITE(destination,*)	Same as above for free-field output. Example: WRITE(6,*) A,JVAL,X(8)
WRITE(destination) list **WRITE(destination@record) list**	Transfers unformatted (binary) information to external file from data elements in **list** (sequentially or directly). Example: WRITE(10) A,B,C WRITE(10@IVAL) A,B,C

COMPILER COMMANDS

FORM		PARAMETERS	DEFAULT
$CONTROL *parameter list*		BOUNDS	Clear
		CHECK= *number*	CHECK = 3
		CODE (NOCODE)	NOCODE
		CROSSREF	No crossref
		CROSSREF ALL	No crossref
		ERRORS= *number*	50 severe errors
	required for each logical unit referenced in program	FILE= *number*	
		FILE= *number*$_1$–*number*$_n$	
		FIXED	FIXED
		FREE	
		INIT	Clear
		LABEL (NOLABEL)	NOLABEL
		LIST (NOLIST)	LIST
		LOCATION (NOLOCATION)	NOLOCATION
		MAP (NOMAP)	NOMAP
		SEGMENT= *name*	SEGMENT=SEG'
		SOURCE (NOSOURCE)	SOURCE (Batch) NOSOURCE (Interactive)
		STAT (NOSTAT)	NOSTAT
		USLINIT	Clear
		WARN (NOWARN)	WARN
$EDIT *parameter list*		FIXED	
		FREE	
		INC= *number*	
		NOSEQ	
		SEQNUM = *sequence number*	
		VOID = *sequence number*	

$IF X*n* = $\begin{matrix}\text{ON}\\ \text{OFF}\end{matrix}$

$INTEGER*4

$PAGE

$PAGE *character string list*

$SET X0 = $\begin{matrix}\text{ON}\\ \text{OFF}\end{matrix}$, SET X1 = $\begin{matrix}\text{ON}\\ \text{OFF}\end{matrix}$, ... , SET X*n* = $\begin{matrix}\text{ON}\\ \text{OFF}\end{matrix}$ or

$SET X0 = $\begin{matrix}\text{ON}\\ \text{OFF}\end{matrix}$, X1 = $\begin{matrix}\text{ON}\\ \text{OFF}\end{matrix}$, . . X_n= $\begin{matrix}\text{ON}\\ \text{OFF}\end{matrix}$

$TITLE *character string list*

$TRACE *program unit; identifier, identifier, . . . , identifier*

INTRINSIC FUNCTIONS

FUNCTION REFERENCE	TYPE OF	
	ARG	FN
ABS*(a)*	R	R
IABS*(a)*	I	I
JABS*(a)*	DI	DI
DABS*(a)*	DP	DP
AINT*(a)*	R	R
INT*(a)*	R or L	I
IJINT*(a)*	DI	I
JINT*(a)*	R	DI
JINT*(a)*	I	DI
IDINT*(a)*	DP	I
JDINT*(a)*	DP	DI
DDINT*(a)*	DP	DP
AMOD(a_1, a_2)	R	R
MOD(a_1, a_2)	I	I
JMOD(a_1, a_2)	DI	DI
AMAX0 $(a_1, a_2, \ldots a_n)$	I	R
AMAX1 $(a_1, a_2, \ldots a_n)$	R	R
MAX0 $(a_1, a_2, \ldots a_n)$	I	I
MAX1 $(a_1, a_2, \ldots a_n)$	R	I
DMAX1 $(a_1, a_2, \ldots a_n)$	D	D
AJMAX0	DI	R
JMAX0 $(a_1, a_2, \ldots, a_n)$	DI	DI

FUNCTION REFERENCE	TYPE OF	
	ARG	FN
JMAX1 $(a_1, a_2, \ldots a_n)$	R	DI
AMIN0 $(a_1, a_2, \ldots a_n)$	I	R
AMIN1 $(a_1, a_2, \ldots a_n)$	R	R
MIN0 $(a_1, a_2, \ldots a_n)$	I	I
MIN1 $(a_1, a_2, \ldots a_n)$	R	I
DMIN1 $(a_1, a_2, \ldots a_n)$	DP	DP
AJMIN0 $(a_1, a_2, \ldots a_n)$	DI	R
JMIN0 $(a_1, a_2, \ldots a_n)$	DI	DI
JMIN1 $(a_1, a_2, \ldots a_n)$	R	DI
FLOAT *(a)*	I	R
FLOATJ (a_1)	DI	R
IFIX*(a)*	R	I
JFIX*(a)*	R	DI
SIGN(a_1, a_2)	R	R
ISIGN(a_1, a_2)	I	I
DSIGN(a_1, a_2)	DP	DP
JSIGN(a_1, a_2)	DI	DI
DIM(a_1, a_2)	R	R
IDIM(a_1, a_2)	I	I
JDIM(a_1, a_2)	DI	DI

INTRINSIC FUNCTIONS (continued)

FUNCTION REFERENCE	TYPE OF	
	ARG	FN
SNGL*(a)*	DP	R
REAL*(a)*	CX	R
AIMAG*(a)*	CX	R
DBLE*(a)*	R	DP
CMPLX(a_1, a_2)	R	CX
CONJG*(a)*	CX	CX
INDEX(a_1, a_2)	CH	I
INUM*(a)*	CH	I
JNUM*(a)*	CH	DI
RNUM*(a)*	CH	R
DNUM*(a)*	CH	DP
STR(a_1, a_2)	I, R, DP, or DI	CH
BOOL*(a)*	I	L

CH	=	Character
CX	=	Complex
DI	=	Double Integer
DP	=	Double Precision
I	=	Integer
L	=	Logical
R	=	Real

BASIC EXTERNAL FUNCTIONS

FUNCTION REFERENCE	TYPE OF ARG	TYPE OF FN	FUNCTION REFERENCE	TYPE OF ARG	TYPE OF FN
EXP(*a*)	R	R	DSQRT(*a*)	DP	DP
DEXP(*a*)	DP	DP	CSQRT(*a*)	CX	CX
CEXP(*a*)	CX	CX	ATAN(*a*)	R	R
ALOG(*a*)	R	R	DATAN(*a*)	DP	DP
DLOG(*a*)	DP	DP	ATAN2(a_1, a_2)	R	R
CLOG(*a*)	CX	CX	DATAN2(a_1, a_2)	DP	DP
ALOG10(*a*)	R	R	DMOD(a_1, a_2)	DP	DP
DLOG10(*a*)	DP	DP	CABS(*a*)	CX	R
SIN(*a*)	R	R	SINH(*a*)	R	R
DSIN(*a*)	DP	DP	DSINH(*a*)	DP	DP
CSIN(*a*)	CX	CX	CSINH(*a*)	CX	CX
COS(*a*)	R	R	COSH(*a*)	R	R
DCOS(*a*)	DP	DP	DCOSH(*a*)	DP	DP
CCOS(*a*)	CX	CX	CCOSH(*a*)	CX	CX
TAN(*a*)	R	R	TANH(*a*)	R	R
DTAN(*a*)	DP	DP	DTANH(*a*)	DP	DP
CTAN(*a*)	CX	CX	CTANH(*a*)	CX	CX
SQRT(*a*)	R	R			

CX	=	Complex
DP	=	Double Precision
R	=	Real

GENERIC FUNCTIONS

GENERIC NAME	SPECIFIC NAME	TYPE OF ARG	TYPE OF RESULT
ABS	ABS	R	R
	IABS	I	I
	DABS	DP	DP
	CABS	CX	CX
	JABS	DI	DI
INT	INT	R or L	I
	IFIX	R	I
	IDINT	DP	I
	IJINT	DI	I
(if INTEGER*4 is invoked)	JINT	R	DI
	JDINT	DP	DI
	JFIX	R	DI
	JIINT	I	DI
JINT	JINT	R	DI
	JDINT	DP	DI
	JFIX	R	DI
	JIINT	I	DI
REAL	FLOAT	I	R
	SNGL	DP	R
	REAL	CX	R
	FLOATJ	DI	R
MOD	MOD	I	I
	AMOD	R	R
	DMOD	DP	DP
	JMOD	DI	DI
SIGN	SIGN	R	R
	ISIGN	I	I
	DSIGN	DP	DP
	JSIGN	DI	DI

CX	=	Complex	I	=	Integer
DI	=	Double Integer	L	=	Logical
DP	=	Double Precision	R	=	Real

GENERIC FUNCTIONS (continued)

GENERIC NAME	SPECIFIC NAME	TYPE OF	
		ARG	RESULT
DIM	DIM	R	R
	IDIM	I	I
	JDIM	DI	DI
MAX	MAX0	I	I
	AMAX1	R	R
	DMAX1	DP	DP
	JMAX0	DI	DI
MIN	MIN0	I	I
	AMIN1	R	R
	DMIN1	DP	DP
	JMIN0	DI	DI
SQRT	SQRT	R	R
	DSQRT	DP	DP
	CSQRT	CX	CX
EXP	EXP	R	R
	DEXP	DP	DP
	CEXP	CX	CX
LOG	ALOG	R	R
	DLOG	DP	DP
	CLOG	CX	CX
SIN	SIN	R	R
	DSIN	DP	DP
	CSIN	CX	CX
COS	COS	R	R
	DCOS	DP	DP
	CCOS	CX	CX

CX	=	Complex	I	=	Integer
DI	=	Double Integer	L	=	Logical
DP	=	Double Precision	R	=	Real

GENERIC FUNCTIONS (continued)

GENERIC NAME	SPECIFIC NAME	TYPE OF	
		ARG	RESULT
TAN	TAN	R	R
	DTAN	DP	DP
	CTAN	CX	CX
ATAN	ATAN	R	R
	DATAN	DP	DP
	ATAN2	R	R
	DATAN2	DP	DP
SINH	SINH	R	R
	DSINH	DP	DP
	CSINH	CX	CX
COSH	COSH	R	R
	DCOSH	DP	DP
	CCOSH	CX	CX
TANH	TANH	R	R
	DTANH	DP	DP
	CTANH	CX	CX

CX	=	Complex	I	=	Integer
DI	=	Double Integer	L	=	Logical
DP	=	Double Precision	R	=	Real

FILES

Standard Device Files

FTN05 (logical unit 5) standard input file $STDIN

FTN06 (logical unit 6) standard output file $STDLIST

Default File Parameters

FILEDESIGNATOR	FTN*dd*, where *dd* is the UNIT number in the FLUT (for example, FTN03).
FOPTIONS	Bits are set or cleared for the following file options: Domain (bits 15 and 14 clear): NEW
	BINARY file (bit 13 clear)[1]
	Default File Designator (bits 12, 11, 10): 000[2]
	Record Format (bits 9 and 8): If sequential, then VARIABLE (bits 9 and 8 = 01), else (direct) FIXED (bits 9 and 8 = 00).
	Carriage Control (bit 7 clear): none[3]
	Disallow File Equation (bit 5 clear): allow :FILE
	(Bits 4 through 0 are spares.)
AOPTIONS	Access Type: READ/WRITE
	No Multirecord
	No Dynamic Locking
	Exclusive Access: Default
	Buffered
RECSIZE	System default value: 128 words.
DEVICE	System default: DISC.
FORMMSG	None.
USERLABELS	System default: 0

[1] Except for FTN05 or FTN06: ASCII file (bit 13 set).
[2] Except for FTN05: 100 for $STDIN, and FTN06: 001 for $STDLIST.
[3] Except for FTN06: yes (bit 7 set).

BLOCKFACTOR	System default value: 128/physical record size
NUMBUFFERS	System default: OUTPRI = 8 COPIES = 1 BUFFERS = 2
FILESIZE	System default: 1023 records
NUMEXTENTS	System default: 8 extents
INITIALLOC	System default value: 1 extent
FILECODE	System default: 0.

UNITCONTROL PROCEDURE

CALL UNITCONTROL(*unit,opt*)

unit is FLUT entry (1 to 99)

opt is
-1 REWIND
0 BACKSPACE
1 ENDFILE
2 SKIP BACKWARD TO TAPE MARK
3 SKIP FORWARD TO TAPE MARK
4 UNLOAD TAPE AND CLOSE FILE
5 LEAVE TAPE AND CLOSE FILE
6 CONVERT FILE TO PRE-SPACING
7 CONVERT FILE TO POST-SPACING
8 CLOSE FILE

MPE COMMANDS TO COMPILE, PREPARE, AND EXECUTE

Note: *entrypoint* and all keyword parameters delimited by semicolon are optional.

:FORTRAN *textfile,uslfile,listfile,masterfile,newfile*

:FORTPREP *textfile,progfile,listfile,masterfile,newfile*

:FORTGO *textfile,listfile,masterfile,newfile*

:PREP *uslfile,progfile;*PMAP;MAXDATA=*segsize*;STACK=*stacksize*;
ZERODB; DL=*dlsize*;CAP=*caplist*;RL=*filename*

:PREPRUN *uslfile*;NOPRIV;PMAP;DEBUG;LMAP;MAXDATA=
segsize;PARM=*parameternum*;STACK=*stacksize*;DL=*dlsize*;
ZERODB; LIB=*library*;CAP=*caplist*;RL=*filename*;NOCB

:RUN *progfile*;NOPRIV;LMAP;DEBUG;MAXDATA=*segsize*;
PARM=*parameternum*;STACK=*stacksize*;DL=*dlsize*;LIB=*library* ;
NOCB

FORTRAN Compiler File Designators

File	Purpose	Formal File Designator	Default File Designator
Textfile	Contains source program, correction text to be merged, and/or compiler subsystem commands.	FTNTEXT	$STDIN
Listfile	Destination of listing output.	FTNLIST	$STDLIST
Uslfile	Destination of object program code.	FTNUSL	$NEWPASS
Masterfile	Old source program to be merged and edited with new text input from *textfile.*	FTNMAST	$NULL
Newfile	New source program resulting from (optional) merging of *textfile* and *masterfile.*	FTNNEW	$NULL
Progfile	Destination of executable object program.	None	$NEWPASS

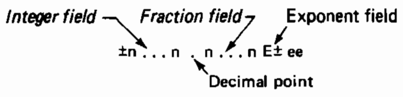
TRAP HANDLING

Error Condition	Subroutine Parameters
Arithmetic Errors:	
REAL DIV 0 REAL OVERFLOW REAL UNDERFLOW	1. real variable
DOUBLE PRECISION DIV 0 DOUBLE PRECISION OVER-FLOW DOUBLE PRECISION UNDER-FLOW	1. double precision variable
INTEGER*2 DIV 0 INTEGER*2 OVERFLOW	1. integer variable
INTEGER*4 OVERFLOW INTEGER*4 DIV 0	1. double integer variable
INTEGER DIV 0 INTEGER OVERFLOW	1. integer variable (or double integer variable if $INTEGER*4 used).
System Errors: SYSTEM ERROR	1. integer array (8 parameters)
Basic External Function Errors: EXTERNAL ERROR	1. single integer variable (compiler library errors 1-50) 2. result 3. first operand 4. second operand (if one exists)
Internal Function Errors: INTERNAL ERROR	1. single integer variable (compiler library errors 51-99) 2. result
Format Errors: FORMAT ERROR	1. single integer variable (compiler library errors 101-149) 2. format (character array) 3. offset to location of error in format 4. I/O buffer (character array) 5. offset to error in I/O buffer 6. single integer variable (unit number) 7. single integer variable (MPE file number)
Plot Errors: PLOT ERROR	1. single integer variable (compiler library errors 150-157)

FORMATS

Format Specifications

Rules for Input: Input field can include integer, fraction and exponent subfields



as well as $ and commas in appropriate places.

Examples:

2314
+2314
−2314
2314+2
+2314E−4
−2314E+4
2.314
+2.314
−.2314
2.314+2
+2.314E−4
−2.314E+4
$2,314
$2,314.00
−$2,314.00
2,314
+2,314.30
−2,314,000

1. Number of characters, including $ and commas, must not exceed *w* in field descriptor.
2. Exponent field input can be ±e ±ee Ee E±e Eee E±ee De D±e D±ee Dee
3. Embedded or trailing blanks are treated as zeros; leading blanks are ignored. Field of all blanks is treated as zero.
4. Internal storage type based on type of variable currently using field descriptor. Type INTEGER and DOUBLE INTEGER truncate, REAL and DOUBLE PRECISION round.

Scale Factor

*n*P*f* or *n*P*rf*

n	=	positive or negative scale factor
f	=	field descriptor
r	=	repeat specification

Example: (E10.3,2PF12.4,I9)

Output:	Dw.d,Ew.d and Gw.d–selected	−*n* or 0 leading zeros, *d*+*n* significant digits in fraction field (rounded)
	Ew.d	+*n* significant digits in integer field and *d*−*n* in fraction field (rounded)
	Fw.d, Mw.d, Nw.d	10^n times internal value
Input:	Use for integer or fixed-field to Dw.d, Ew.d, Fw.d, Gw.d, Mw.d and Nw.d to multiply by 10^{-n}. No effect if input has exponent field.	

Edit Specifications

edit descriptor or *repeat specification edit descriptor*

Note: Nesting maximum, 4 levels

Edit Descriptors	**Output**	**Input**
" . . . "	"THIS IS A FIXED STRING"	Skip *n* characters.
' . . . '	'THIS IS A FIXED STRING'	Skip *n* characters.
*n*H	6HOUTPUT	Replace edit descriptor characters.
*n*X	4X prints 4 blanks	Skip *n* characters.
T*n*	T20 tabs to 20th position.	
/	Terminates current record.	Terminates current record.
%*n*C (first character)	%40 (blank) = single space	
	%60 (0) = double space	
	%61 (1) = page eject	
	%53 (+) = no space (suppress)	
	%2nn = space nn lines (n=0–7)	
	%300 = page eject (tape channel 1)	
	%301 = skip to bottom of form (tape channel 2)	
	%302 = single space (automatic page eject, tape channel 3)	
	%303 = single space next odd-numbered line (automatic page eject, tape channel 4)	
	%304 = triple space (automatic page eject, tape channel 5)	

%*n* C (first character) (cont)	%305 = space 1/2 page (automatic page eject, channel 6)
	%306 = space 1/4 page (automatic page eject, channel 7)
	%307 = space 1/6 page (automatic page eject, channel 8)
	%310 = space to bottom of form (tape channel 9)
	%311 = skip to channel 10
	%312 = skip to channel 11
	%313 = skip to channel 12
	%320 = no space, no return.
	%0–%37,%41–%52,%54–%57, %62–%77,%314–%317,%321–%377 same as %40
	%400 or %100(@) = set post-space movement option
	%401 or %101(A) = set pre-space movement option
	%402 or %102(B) = set single-space option with auto-eject.
	%403 or %103(C) = set single-space option without automatic page eject (66 lines)

Free-field Input

Data types allowed:	octal, integer, double integer, floating-point real, double precision floating-point, and character string.
Data item delimiters:	comma, blank, any ASCII character (not part of data item)
Decimal Data:	Any field descriptor form except monetary or numeration. Leading, embedded or trailing blanks, commas, $ or any ASCII character not part of data item are delimiters.
Octal Data:	%$i_1 \ldots i_n$ (max. digits: 6 for integer, 11 for double integer or real, 22 for double precision Series II, 16 for Series I)
Character string data:	Any series of ASCII characters (usually enclosed in blanks) Left-justified in variable. If end of record, assumes rest of data in next record.
Complex data:	Parentheses enclosing a pair of values separated by comma.
Record terminator:	Slash (/), except within character data items, terminates current record, delimits current data item. If list not satisfied, remainder of record skipped, transfer resumes first character next record.

Free-field Output

Data types:	integer (I6), floating-point real (G12.6), double integer (I11), double-precision floating-point (G23.17 for Series II, G17.11 for Series I), character (" . . . "), logical (L1), complex (2G12.6).
Data item delimiter:	1 blank space
Record terminator:	Slash (/) output if numeric data item exceeds current record and item continued on next line. If character item, / not output but item continues on subsequent records.

Field Descriptors

Descriptor	Result	Comments
Ew.d	$\overset{\longleftarrow w \longrightarrow}{-.x_1 \ldots x_dE\pm ee}$	w > =d+6 Decimal optional. Truncates. Trailing blanks are zeros.
Fw.d	$\overset{\longleftarrow w \longrightarrow}{+i_1 \ldots i_n.f_1 \ldots f_d}$	w > =d+n+3 Decimal optional.
Gw.d	same as Ew.d if absolute value after rounding > d or < .1, otherwise same as Fw.d with 4 trailing blanks.	
Mw.d	$\overset{\longleftarrow w \longrightarrow}{-\$i_1 \ldots,\ldots,\ldots i_n.f_1 \ldots f_d}$	w ⩾ d+n+c+4 c = number of commas (On input, $ and commas accepted but ignored)
Nw.d	$\overset{\longleftarrow w \longrightarrow}{-i_1 \ldots,\ldots,\ldots i_n.f_1 \ldots f_d}$	w ⩾ d+n+c+3 c = number of commas (On input, commas accepted but ignored)
Dw.d	$\overset{\longleftarrow w \longrightarrow}{-.x_1 \ldots x_dD\pm ee}$	w ⩾ d+6
Iw	$\overset{\longleftarrow w \longrightarrow}{-i_1 \ldots i_n}$	w ⩾ n+2 Input same as Fw.d with d = 0.
Ow	$\overset{\longleftarrow w \longrightarrow}{i_1 \ldots i_n}$	Output min., Input max.: 6 integer, 11 real or double integer, 22 double precision Series II, 16 double precision Series I.

Descriptor	Result	Comments
Zw	$\overleftrightarrow{w}$ $i_1 \ldots i_n$	Output min., Input max.: 4 integer, 8 real or double integer, 16 double precision for Series II, 12 for Series I.
Lw	$\overleftrightarrow{w}$ $b_1 \ldots b_n c$	b = blank, c = T if least significant digit is 1 or F if it is 0. On input, T becomes −1, F becomes 0.
Aw	$\overleftrightarrow{w}$ $b_1 \ldots b_r c_1 \ldots c_n$	Output: $w < n$, leftmost characters output Input: $w > n$, inputs last n characters $n < w$, inputs w, adds trailing blanks
Rw	$\overleftrightarrow{w}$ $b_1 \ldots b_r c_1 \ldots c_n$	Output: $w < n$, rightmost characters output Input: $w > n$, inputs last n characters $n < w$, inputs w right-justified, left fills with binary zeros.
S	$\overleftrightarrow{w}$ $c_1 \ldots c_n$	w = length attribute of character variable

Note: E, F, G, M, N, D round least significant digit.
All output right-justified.
All input, next w positions in ASCII record read.

· NOTES