HEWLETT hp PACKARD

2100 series computers

relocatable subroutines

HEWLETT hp PACKARD

2100 series computers

relocatable subroutines

HP MANUAL PART NO. 02116-91780
MICROFICHE PART NO. 02116-91806

October 1973

LIST OF EFFECTIVE PAGES

Title	Oct. 1973
iii to xiv	Oct. 1973
1-1 to 1-95	Oct. 1973
2-1 to 2-75	Oct. 1973
3-1 to 3-32	Oct. 1973
I-1	Oct. 1973
II-1	Oct. 1973
III-1	Oct. 1973
IV-1	Oct. 1973
V-1 to V-2	Oct. 1973

PRINTING HISTORY

First Edition . . . . Sep. 1970
Second Edition . . . . Oct. 1973

PREFACE

Relocatable Subroutines is a programmer's reference to all the subroutines of the various Hewlett-Packard Relocatable Libraries. It should be used in conjunction with the appropriate language and system manuals.

The Introduction explains the Relocatable Libraries, their relationships, and their uses. Section I contains all of the mathematical subroutines from all of the libraries, arranged alphabetically by subroutine name. Section II provides a similar listing of the utility subroutines. Section III is dedicated entirely to the three versions of the Formatter.

There are also indices that give page references for each routine of each library and an index of all the entry points mentioned in the book.

CONTENTS

INTRODUCTION

INTRODUCTION

Every Hewlett-Packard operating system that has a relocating loader (BCS, RTE, DOS, DOS-M) also has one or more Relocatable Libraries. The subroutines in these libraries perform mathematical and utility functions for user programs. The Relocating Loader links each user program with the subroutines that it needs.

From the library point of view, an operating system has four characterisitcs:

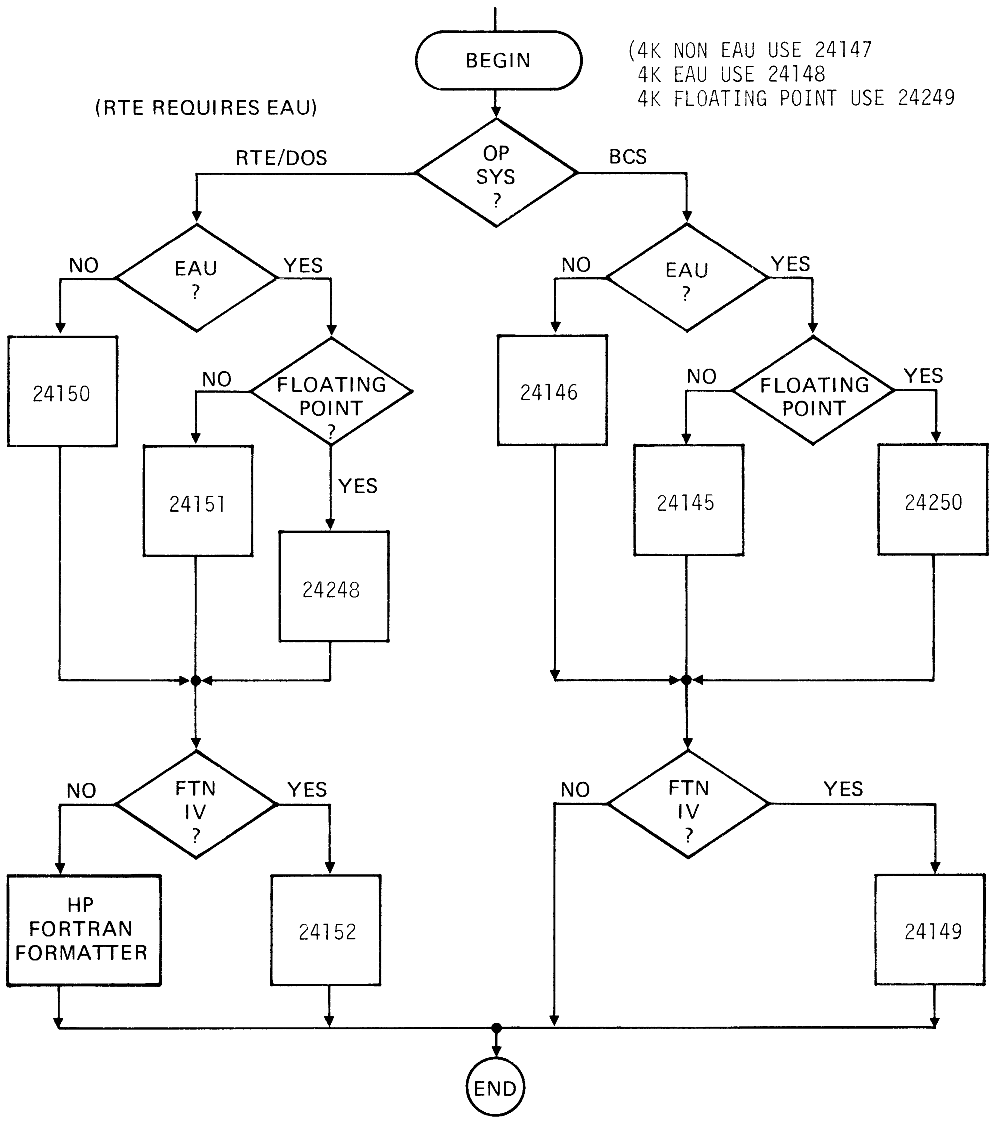
1. The system is disc-based (RTE, DOS, etc.) or is not (BCS).
2. The system includes EAU (Extended Arithmetic Unit) or does not.
3. The system includes the hardware floating-point option or does not.
4. The system includes extended precision arithmetic and formatting (FORTRAN IV library) or does not.

For each possible operating system there are two appropriate libraries: a standard library (BCS or disc-based, EAU or non-EAU, hardware floating-point or non-hardware floating-point) and an optional FORTRAN IV Library. (There are also special libraries for 4K BCS installations.) Each library has a five-character product number. These libraries include:

24147	Non-EAU FORTRAN Library (4K)
24148	EAU FORTRAN Library (4K)
24249	4K BCS Relocatable Library-Floating Point
24145	EAU Relocatable Library (BCS)
24146	Non-EAU Relocatable Library (BCS)
24149	BCS FTN IV Library
24250	BCS Relocatable Library-Floating Point
24150	Non-EAU RTE/DOS Relocatable Library (without HP FORTRAN Formatter)
24151	EAU RTE/DOS Relocatable Library (no Formatter)
24152	RTE/DOS FORTRAN IV Library (with FORTRAN IV Formatter)
24153	RTE/DOS HP FORTRAN Formatter
24248	RTE/DOS Relocatable Library-Floating Point

In addition there are two Plotter libraries that support the printing of graphs: a BCS version and an RTE/DOS version.

The chart below shows the decision process for choosing the correct libraries for any possible system configuration:



EAU AND FLOATING-POINT PHILOSOPHY

The floating-point hardware option provides hardware floating-point add, subtract, multiply, divide and conversion between fixed- and floating-point values.

The Extended Arithmetic Unit provides hardware multiply, divide and double load-store. In order to promote compatability between different systems, all compilers generate non-EAU code. That is, code generated by the FORTRAN compiler calls the multiply subroutine rather than using the hardware instruction. At run-time, if the system contains an EAU library, these subroutine calls are replaced by the corresponding EAU hardware instruction. Similar operation occurs for floating-point instructions.

ORGANIZATION OF THIS BOOK

This book is organized into three sections plus several indices. Since many subroutines appear in more than one library, each subroutine is documented only once. All mathematical subroutines are grouped into Section I, ordered alphabetically by name. All utility subroutines are covered in Section II, also ordered alphabetically. Section III covers all the Formatters.

For each library, there is an index that lists the subroutines in the order they appear in the library. With each subroutine is a page reference.

The final index provides an alphabetic list, with page references, of every entry point mentioned in the book. This is provided in case you know the entry point of a routine, but not the name.

The Page Format

Each subroutine is documented on a page of standard format. (See the sample page.) The following items may appear for each subroutine:

"NAME"	The name of the routine record in the NAM record.
Purpose	The use of the routine.
Entry Points	The entry points to the routine. If these are centered, they apply to both the BCS version and the DOS/RTE version of the routine. An entry of "N/A" means the routine is not available in that system. After the DOS/RTE entry point, there is a letter in parentheses giving the type of the routine: U for utility, P for privileged, and R for re-entrant.

External References

These are other subroutines that are called by the subroutine. If centered, they apply to both versions of the routine; if divided into two columns, they are different for the two versions. If the DOS/RTE version is type P or R, then it also references $LIBR and $LIBX.

Calling Sequences

This is the assembly language calling sequence for each entry point. If there is only one calling sequence, it is centered. The arrow (→) indicates a return point. "A" and "B" indicate the A- and B- registers.

Method

This gives the algorithm for producing the result and/or the accuracy of the routine.

Attribute Chart

For each entry point, this chart gives the following information:

a. Parameters: their type (real, integer, double real or complex) and whether they are loaded into the A- and B- registers.

b. Result: the type of the result and the registers (if any) where it is returned.

c. Basic FORTRAN: whether the routine is an intrinsic function (i.e., ABS(x)), callable subroutine, or uncallable in HP FORTRAN.

d. FORTRAN IV: Whether the routine is an intrinsic function, callable subroutine, or uncallable in HP FORTRAN IV.

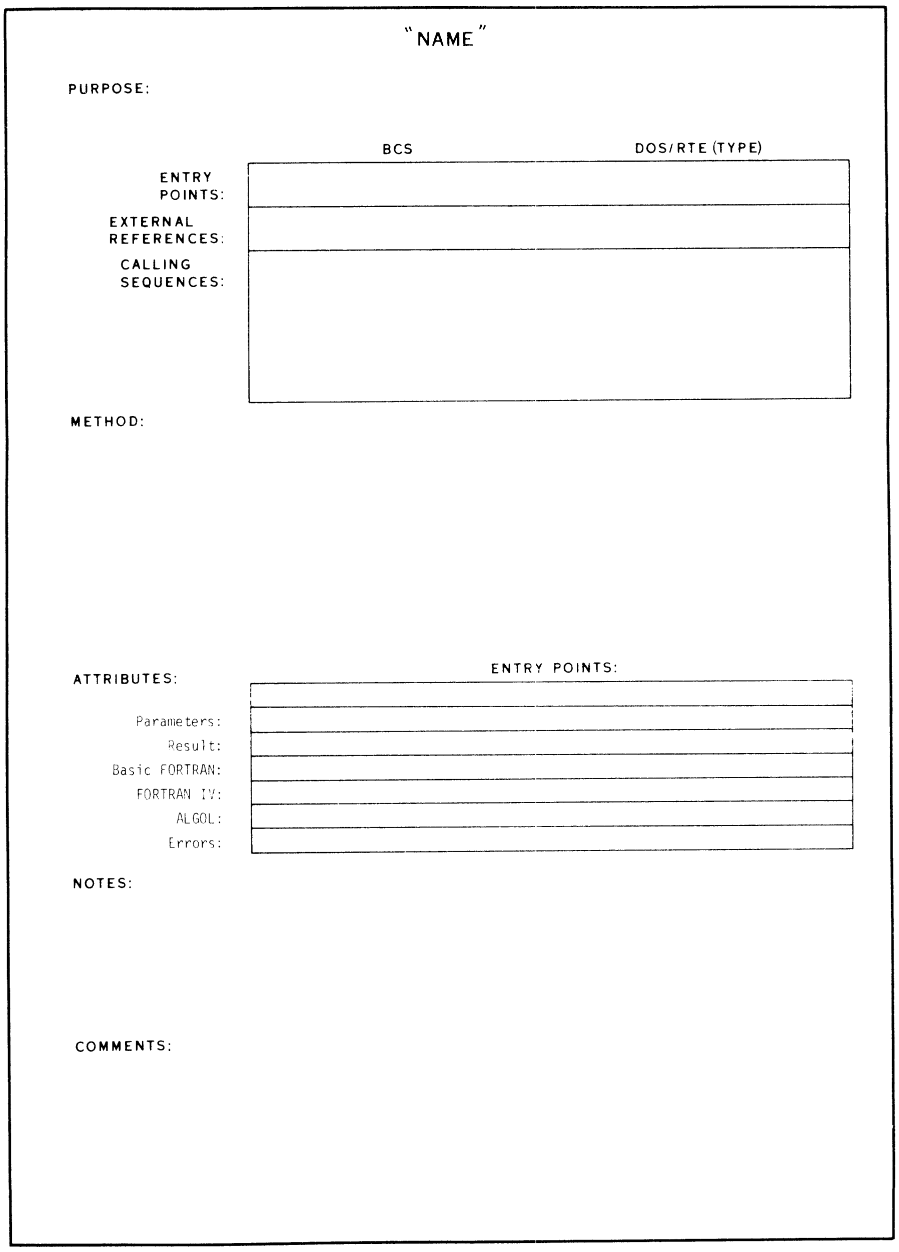
e. ALGOL: whether the routine is an intrinsic, callable or uncallable procedure in HP ALGOL.

f. ERRORS: This gives a summary of any error conditions in this format:

condition → (message or code)

(If the condition occurs, the message is printed.)

SAMPLE PAGE FORMAT



"NAME"

PURPOSE:

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		
EXTERNAL REFERENCES:		
CALLING SEQUENCES:		

METHOD:

ATTRIBUTES:

	ENTRY POINTS:
Parameters:	
Result:	
Basic FORTRAN:	
FORTRAN IV:	
ALGOL:	
Errors:	

NOTES:

COMMENTS:

LOADING SEQUENCES

If two libraries are used with an operating system, they must be loaded in a specific manner.

In BCS, the FORTRAN IV Library must be loaded _before_ the standard library.

In disc-based systems, either the FORTRAN IV Library _or_ the HP FORTRAN Formatter must be loaded in addition to the standard library.

SECTION I

MATHEMATICAL SUBROUTINES

ABS

PURPOSE: Calculate the absolute value of a real *x*.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	ABS (P)	
EXTERNAL REFERENCES:	..FCM	
CALLING SEQUENCES:	DLD *x* JSB ABS → result in A & B	

ATTRIBUTES:

	ENTRY POINTS:
	ABS
Parameters:	Real: A & B
Result:	Real: A & B
Basic FORTRAN:	Function: ABS (*x*)
FORTRAN IV:	Function: ABS (*x*)
ALGOL:	Intrinsic: ABS (*x*)
Errors:	None

AIMAG

PURPOSE: Extract the imaginary part of a complex *x*.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	AIMAG (P)	
EXTERNAL REFERENCES:	.ENTR	.ENTP
CALLING SEQUENCES:	JSB AIMAG DEF *+2 DEF *x* → result in A & B	

ATTRIBUTES:

	ENTRY POINTS:
	AIMAG
Parameters:	Complex
Result:	Two-word imaginary: A & B
Basic FORTRAN:	Callable as function
FORTRAN IV:	Intr. function: AIMAG (*x*)
ALGOL:	Callable as real procedure
Errors:	None

AINT

PURPOSE: Truncate a real x:

y = SIGN (x). (largest integer $\leq |x|$), or $y = [x]$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		AINT (P)
EXTERNAL REFERENCES:		.FLUN .PACK
CALLING SEQUENCES:		DLD x JSB AINT $\rightarrow y$ in A & B

ATTRIBUTES:

	ENTRY POINTS:
	AINT
Parameters:	Real: A&B
Result:	Real: A&B
Basic FORTRAN:	Not callable
FORTRAN IV:	Function: AINT (x)
ALGOL:	Not callable
Errors:	None

ALOG

PURPOSE: Calculate the natural logarithm of a real X:

$$Y = \ln(X)$$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	LN (R) ALOG	
EXTERNAL REFERENCES:	.FLUN, .MANT, FLOAT, .ERRR	.FLUN, .MANT, FLOAT
CALLING SEQUENCES:	DLD X JSB ALOG (or LN) → return (Y in A, B)	DLD X JSB ALOG (or LN) JSB ERRØ (error return) → return (Y in A & B)

METHOD: Let F = mantissa (X) I = characteristic (X) (that is, $X = 2^I \times F$) Then $Y = I + \log_2 F \cdot (\log_e 2) = \log_e 2 \left(I + z \left[c_1 + \frac{c_2}{c_3 - z^2} \right]^{-1/2} \right)$

where

$$z = \frac{F - \sqrt{2}/2}{F + \sqrt{2}/2}$$

and

$c_1 = 1.2920070987$
$c_2 = 2.6398577035$
$c_3 = 1.6567626301$

ATTRIBUTES:

	ENTRY POINTS:	
	ALOG	LN
Parameters:	Real: A & B	Real: A & B
Result:	Real: A & B	Real: A & B
Basic FORTRAN:	Function: ALOG(X)	Not Callable
FORTRAN IV:	Function: ALOG (X)	Not Callable
ALGOL:	Not Callable	Intrinsic Procedure
Errors:	$X \le 0$ → (Ø2 UN)	Same

NOTES: ALOG is the FORTRAN entry point; LN is the ALGOL entry point.

ALOGT

PURPOSE: Calculate the common logarithm (base 10) of real x:

$$y = \log_{10} x$$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	ALOGT (U)	
EXTERNAL REFERENCES:	ALOG	
CALLING SEQUENCES:	DLD x JSB ALOGT → normal return (result in A&B)	DLD x JSB ALOGT → error return → normal return (result in A&B)

METHOD:

$y = \log_{10} x = \log_{10} e * \log_e x$

Accuracy depends on the accuracy of ALOG.

ATTRIBUTES:

	ENTRY POINTS:
	ALOGT
Parameters:	Real
Result:	Real: A&B
Basic FORTRAN:	Not Callable
FORTRAN IV:	Intr. Function: ALOGT (x)
ALGOL:	Not Callable
Errors:	If $x \leq 0 \rightarrow$ (Ø2 UN)

AMOD

PURPOSE: Calculate the real remainder of *X*/*Y* for real *X* and *Y*:

$Z = X \text{ modulo } Y$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	AMOD (P)	
EXTERNAL REFERENCES:	.ENTR AINT	.ENTP AINT
CALLING SEQUENCES:	JSB AMOD DEF * + 3 DEF *X* DEF *Y* → *Z* in A & B	

METHOD:

$Z = X - [X/Y] * X$

ATTRIBUTES:

	ENTRY POINTS:
	AMOD
Parameters:	Real
Result:	Real: A&B
Basic FORTRAN:	Callable as Function
FORTRAN IV:	Intrinsic Function: AMOD (*X*,*Y*)
ALGOL:	Callable as Real Procedure
Errors:	If $Y = 0$, then $Z = X$

ATAN

PURPOSE: Calculate the arctangent of a real x:

$$y = \tan^{-1}(x)$$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	ARCTA (R) ATAN	
EXTERNAL REFERENCES:	.CHEB	
CALLING SEQUENCES:	DLD x JSB ATAN (or ARCTA) → return (y in A&B)	

METHOD:

```
if abs ($x$) > 1 then $u = 1/x$ else $u = x$
$y = u$ * Cheby($2*u*u - 1$)
if abs ($x$) $\leq$ 1 then answer = $y$
else if $x \geq 0$ then answer = $\pi/2-y$
else answer = $-\pi/2-y$
```

ATTRIBUTES:

	ENTRY POINTS:	
	ATAN	ARCTA
Parameters:	Real: A & B	Real: A & B
Result:	Real: A & B (radians)	Real: A & B (radians)
Basic FORTRAN:	Function: ATAN (x)	Not Callable
FORTRAN IV:	Function: ATAN (x)	Not Callable
ALGOL:	Not Callable	Intrinsic Function: ARCTAN(x)
Errors:	None	None

NOTES:

1. ATAN is the FORTRAN entry point and ARCTA is the ALGOL entry point.
2. Result ranges from $-\pi/2$ to $\pi/2$.

ATAN2

PURPOSE: Calculate the real arctangent of the quotient of two reals: z = arctan (Y/X)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	ATAN2 (R)	
EXTERNAL REFERENCES:	.ENTR, SIGN, ATAN	.ENTP, SIGN, ATAN
CALLING SEQUENCES:	JSB ATAN2 DEF * + 3 DEF Y DEF X $\rightarrow z$ in A & B	

METHOD:

If X = 0, z = sign (Y) $\pi/2$

If X > 0, z = arctan (Y/X)

If X < 0, z = arctan (Y/X) + sign (Y) .π

Accuracy depends on accuracy of ATAN.

ATTRIBUTES:

	ENTRY POINTS:
	ATAN 2
Parameters:	Real
Result:	Real: A & B
Basic FORTRAN:	Callable as Function
FORTRAN IV:	Intr. Function: ATAN2 (Y,X)
ALGOL:	Callable as Real Procedure
Errors:	None

CABS

PURPOSE: Calculate the real absolute value (modulus) of complex x: $y = |x|$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	CABS (R)	
EXTERNAL REFERENCES:	.ENTR, SQRT	.ENTP, SQRT
CALLING SEQUENCES:	JSB CABS DEF *+2 DEF x $\rightarrow y$ in A & B	

METHOD:

$$y = |x| = |x_1 + i * x_2| = \sqrt{x_1^2 + x_2^2}$$

Accuracy depends on the accuracy of SQRT.

ATTRIBUTES:

	ENTRY POINTS:
	CABS
Parameters:	Complex
Result:	Real: A&B
Basic FORTRAN:	Callable as Function
FORTRAN IV:	Intr. Function: CABS (x)
ALGOL:	Callable as Real Procedure
Errors:	None

CADD

PURPOSE: Add complex x to complex y: $z = x + y$ (z is complex)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.CADD (P) CADD	
EXTERNAL REFERENCES:	GETAD ADRES	.PCAD
CALLING SEQUENCES:	JSB .CADD DEF z (result) DEF x DEF y →	or JSB CADD DEF * + 4 DEF z (result) DEF x DEF y →

ATTRIBUTES:

	ENTRY POINTS:	
	.CADD	CADD
Parameters:	Complex	Complex
Result:	Complex	Complex
Basic FORTRAN:	Not Callable	Callable
FORTRAN IV:	Not Callable	Callable
ALGOL:	Not Callable	Callable
Errors:	None	None

CDIV

PURPOSE: Divide complex *X* by complex *Y*: $Z = X/Y$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.CDIV (P) CDIV	
EXTERNAL REFERENCES:	GETAD ADRES	.PCAD

CALLING SEQUENCES:

```
JSB .CDIV          or    JSB CDIV
DEF z (result)           DEF * + 4
DEF x                    DEF z (result)
DEF y                    DEF x
→                        DEF y
                         →
```

ATTRIBUTES:

ENTRY POINTS:

	.CDIV	CDIV
Parameters:	Complex	Complex
Result:	Complex	Complex
Basic FORTRAN:	Not Callable	Callable
FORTRAN IV:	Not Callable	Callable
ALGOL:	Not Callable	Callable
Errors:	None	None

CEXP

PURPOSE: Calculate the complex exponential of a complex x.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	CEXP (R)	
EXTERNAL REFERENCES:	.ENTR, EXP, SIN, COS	.ENTP, EXP, SIN, COS
CALLING SEQUENCES:	JSB CEXP DEF *+3 DEF y (result) DEF x → Normal return	JSB CEXP DEF *+3 DEF y (result) DEF x → Error return → Normal return

METHOD:

$y = y_1 + i.y_2 = e^x = e^{x_1 + i\, x_2} = e^{x_1} (\cos x_2 + i.\sin x_2)$

Accuracy: depends on the accuracy of EXP and SIN.

ATTRIBUTES:

	ENTRY POINTS:
	CEXP
Parameters:	Complex
Result:	Complex
Basic FORTRAN:	Not Callable
FORTRAN IV:	Intr. Function: CEXP(x)
ALGOL:	Not Callable
Errors:	Note 1

NOTES: 1. If $x_1 \cdot \log_2 e \geq 124, \rightarrow$ (Ø7 OF).

If $\frac{1}{2} \left| \frac{x_2}{\pi} + \frac{1}{2} \right| > 2^{14} \rightarrow$ (Ø5 OR).

CHEBY

PURPOSE: Evaluate the chebyshev series at a real *x* for a particular table of coefficients *c*.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		.CHEB (R)
EXTERNAL REFERENCES:		None
CALLING SEQUENCES:		DLD *x* JSB .CHEB DEF *c* (table, note 1) → result in A & B

METHOD: $T_i = 2 \cdot T_{i-1} - T_{i-2} + c_{n-i}$ (i = ∅,1,....,n-1)

where

$T_{-2} = T_{-1} = 0$

n = number of coefficients

$\text{Answer} = \dfrac{T_{n-1} - T_{n-3}}{2}$

ATTRIBUTES:

	ENTRY POINTS:
	.CHEB
Parameters:	Real
Result:	Real
Basic FORTRAN:	Not Callable
FORTRAN IV:	Not Callable
ALGOL:	Not Callable
Errors:	TAN(*x*) for *x* close to π/2

NOTES: 1. Table *c* consists of a series of real coefficients terminated by an integer zero.

CLOG

PURPOSE: Calculate the complex natural logarithm of a complex x.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	CLOG (R)	
EXTERNAL REFERENCES:	.ENTR, ALOG CABS, ATAN2	.ENTP, ALOG CABS, ATAN2
CALLING SEQUENCES:	JSB CLOG DEF *+3 DEF y (result) DEF x → Normal return	JSB CLOG DEF *+3 DEF y (result) DEF x → Error return → Normal return

METHOD:

$$y = y_1 + i \cdot y_2 = \log_e x = \log_e (x_1 + i . x_2) = \log_e(r) + i . \Theta$$

where

$$r = \sqrt{x_1^2 + x_2^2}$$

$$\Theta = \arctan\left(\frac{x_2}{x_1}\right)$$

Accuracy depends on the accuracy of ALOG and SQRT.

ATTRIBUTES:

	ENTRY POINTS:
	CLOG
Parameters:	Complex
Result:	Complex
Basic FORTRAN:	Not Callable
FORTRAN IV:	Intr. Function: CLOG(x)
ALGOL:	Not Callable
Errors:	If x = 0 → (Ø2 UN)

CMPLX

PURPOSE: Combine a real *x* and an imaginary *y* into a complex *z*.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	CMPLX (P)	
EXTERNAL REFERENCES:	.ENTR	.ENTP
CALLING SEQUENCES:	JSB CMPLX DEF *+4 DEF *z* DEF *x* DEF *y* →	

ATTRIBUTES:

	ENTRY POINTS:
	CMPLX
Parameters:	Real & Imaginary
Result:	Complex
Basic FORTRAN:	Callable
FORTRAN IV:	Intr. Function: CMPLX (*x*,*y*)
ALGOL:	Callable
Errors:	None

CMPY

PURPOSE: Multiply complex x by complex y: $z = x \cdot y$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.CMPY CMPY (P)	
EXTERNAL REFERENCES:	GETAD ADRES	.PCAD
CALLING SEQUENCES:	JSB .CMPY DEF z (result) DEF x DEF y →	or JSB CMPY DEF * + 4 DEF z (result) DEF x DEF y →

ATTRIBUTES:

ENTRY POINTS:

	.CMPY	CMPY
Parameters:	Complex	Complex
Result:	Complex	Complex
Basic FORTRAN:	Not Callable	Callable
FORTRAN IV:	Not Callable	Callable
ALGOL:	Not Callable	Callable
Errors:	None	None

CONJG

PURPOSE: Form the conjugate *y* of a complex *x*.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	CONJG (P)	
EXTERNAL REFERENCES:	.ENTR ..FCM	.ENTP ..FCM
CALLING SEQUENCES:	JSB CONJG DEF ⋅⋅ + 3 DEF *y* (result) DEF *x* →	

METHOD: If $x = x_1 + i \cdot x_2$, then $y = x_1 - i \cdot x_2$

ATTRIBUTES:

	ENTRY POINTS:
	CONJG
Parameters:	Complex
Result:	Complex
Basic FORTRAN:	Callable
FORTRAN IV:	Intr. Function: CONJG (*x*)
ALGOL:	Callable
Errors:	None

CSNCS

PURPOSE: Calculate the complex sine or cosine of complex x: y = sine (x)
y = cosine (x)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	CSIN (R) CCOS	
EXTERNAL REFERENCES:	.ENTR, SIN, COS EXP, ..FCM	.ENTP, SIN, COS EXP, ..FCM
CALLING SEQUENCES:	JSB CSIN (or CCOS) DEF * + 3 DEF y DEF x JSB error routine → Normal return	

METHOD: Sine: $y = Y_1 + i \cdot Y_2 = \sin(x) = \sin(x_1 + i \cdot x_2) =$

$$\frac{\sin(x_1)}{2}(e^{x_2} + e^{-x_2}) + i\left(\frac{\cos(x_2)}{2}\right)(e^{x_2} - e^{-x_2})$$

Cosine: $y = Y_1 + Y_2 \,.\, i = \cos(x) = \cos(x_1 + i.x_2) =$

$$\left(\frac{\cos(x_1)}{2}\right)(e^{x_2} + e^{-x_2}) + \left(\frac{i \cdot \sin(x_1)}{2}\right)(e^{x_2} - e^{-x_2})$$

Accuracy depends on the accuracy of EXP and SIN.

ATTRIBUTES:

	ENTRY POINTS:	
	CSIN	CCOS
Parameters:	Complex	Complex
Result:	Complex	Complex
Basic FORTRAN:	Not Callable	Not Callable
FORTRAN IV:	Intr. Function: CSIN (x)	Intr. Function: CCOS (x)
ALGOL:	Not Callable	Not Callable
Errors:	Note 1.	Note 1.

NOTES: 1. $\frac{1}{2}\left|\frac{x}{\pi} + \frac{1}{2}\right| > 2^{14} \rightarrow$ (Ø5 OR)

$x_2 \cdot \log_2 e \geq 124 \rightarrow$ (Ø7 OF)

CSQRT

PURPOSE: Calculate the complex square root of complex *x*: $y = y_1 + i \cdot y_2 = \sqrt{x_1 + i . x_2}$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	CSQRT (R)	
EXTERNAL REFERENCES:	.ENTR, ..DLC, SQRT, CABS	.ENTP, ..DLC, SQRT, CABS
CALLING SEQUENCES:	JSB CSQRT DEF * + 3 DEF *y* (result) DEF *x* →	

METHOD:

If $x = 0$, $y = 0$

If $X_1 \geq 0$; $Y_1 = \sqrt{\frac{X_1 + |X|}{2}}$, $Y_2 = \frac{X_2}{2Y_1}$

If $X_1 < 0$; $Y_2 = \text{sign}(X_2) \sqrt{\frac{-X_1 + |X|}{2}}$, $Y_1 = \frac{X_2}{2Y_2}$

Accuracy depends on the accuracy of SQRT.

ATTRIBUTES:

	ENTRY POINTS:
	CSQRT
Parameters:	Complex
Result:	Complex
Basic FORTRAN:	Callable
FORTRAN IV:	Intr. Function: CSQRT (*x*)
ALGOL:	Callable
Errors:	None

CSUB

PURPOSE: Subtract complex *y* from complex *x*: $z = x - y$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.CSUB (P) CSUB	
EXTERNAL REFERENCES:	GETAD ADRES	.PCAD
CALLING SEQUENCES:	JSB .CSUB DEF *z* (result) DEF *x* DEF *y* → or	JSB CSUB DEF * + 4 DEF *z* (result) DEF *x* DEF *y* →

ATTRIBUTES:

ENTRY POINTS:

	.CSUB	CSUB
Parameters:	Complex	Complex
Result:	Complex	Complex
Basic FORTRAN:	Not Callable	Callable
FORTRAN IV:	Not Callable	Callable
ALGOL:	Not Callable	Callable
Errors:	None	None

DABS

PURPOSE: Calculate the absolute value of a double real x: $y = |x|$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	DABS (P)	
EXTERNAL REFERENCES:	..DCM, .XFER	
CALLING SEQUENCES:	JSB DABS DEF *+3 DEF y DEF x →	

ATTRIBUTES:

	ENTRY POINTS:
	DABS
Parameters:	Double Real
Result:	Double Real
Basic FORTRAN:	Callable
FORTRAN IV:	Function: DABS (x)
ALGOL:	Callable
Errors:	NOTE 1

NOTES:

1. If x = Smallest negative number (-2^{127}), then
   y = Largest positive number $[(1-2^{-39})\cdot 2^{127}]$
   and the overflow bit is set.

DATAN

PURPOSE: Calculate the double real arctangent of double real x: y = arctan (x)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	DATAN (R)	
EXTERNAL REFERENCES:	.XADD, .XSUB, .XMPY, .XDIV, .XFER, ..DCM, .FLUN	
CALLING SEQUENCES:	JSB DATAN DEF *+3 DEF y (result) DEF x →	

METHOD:

If $x < 0$, $y = -\arctan(-x)$

If $|x|>1$, let $z = \frac{1}{|x|}$, then $y = \frac{\pi}{2} - \arctan(z)$

If $|x|<1$, let $z = |x|$

If $z \leq \sqrt{2}-1$, set $v = \tan\frac{\pi}{16}$, $w=\frac{\pi}{16}$

If $z < \sqrt{2}-1$, set $v = \tan\frac{3\pi}{16}$, $w = \frac{3\pi}{16}$

Then $T = \frac{z-v}{1+z\cdot v}$

$\text{Arctan}(z) = w + \arctan(T)$

$$\text{Arctan}(T) = T\left[C_0 + \frac{C_1[(T^2+B_2)(T^2+B_3)+C_3]}{(T^2+B_1)[(T^2+B_2)(T^2+B_3)+C_3]+C_2(T^2+B_3)}\right]$$

C_0 = .208979591837

C_1 = 2.97061224490 $\quad B_1$ = 5.10299532839

C_2 = -3.35025248131 $\quad B_2$ = 2.58417875505

C_3 = -.128720995297 $\quad B_3$ = 1.21282591656

Accuracy: The relative error in $y = \arctan(x+\Delta x)$ is $R = \frac{\Delta x}{(x^2+1)\arctan(x)}$ where Δx represents the round-off error in x. Hence, at $x = \pm.001$, the accuracy will be 9 significant digits due to the round-off error in the 39th bit of x. As x diverges from 0, the accuracy becomes 11 significant digits.

ATTRIBUTES:

	ENTRY POINTS:
	DATAN
Parameters:	Double Real
Result:	Double Real
Basic FORTRAN:	Callable
FORTRAN IV:	Intrinsic Function: DATAN (x)
ALGOL:	Callable
Errors:	None

DATN2

PURPOSE: Calculate the double real arctangent of the quotient of two double reals:

$z = \arctan (Y/X)$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	DATN2 (R)	
EXTERNAL REFERENCES:	.ENTR, DSIGN, DATAN .XADD, .XDIV, .XFER	.ENTP, DSIGN, DATAN .XADD, .XDIV, .XFER
CALLING SEQUENCES:	JSB DATN2 DEF *+4 DEF *z* (result) DEF *Y* DEF *X* →	

METHOD:

If $X = 0$, $z = \text{sign}(Y) \cdot \frac{\pi}{2}$

If $X > 0$, $z = \arctan (Y/X)$

If $X < 0$, $z = \arctan (Y/X) + \text{sign}(Y) \cdot \pi$

Accuracy depends on accuracy of DATAN.

ATTRIBUTES:

	ENTRY POINTS:
	DATN2
Parameters:	Double Real
Result:	Double Real
Basic FORTRAN:	Callable
FORTRAN IV:	Intrinsic Function: DATN2 (Y,X)
ALGOL:	Callable
Errors:	None

DBLE

PURPOSE: Convert a real *x* to a double real *y*.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	DBLE (P)	
EXTERNAL REFERENCES:	.FLUN, .XFER, .XPAK	
CALLING SEQUENCES:	JSB DBLE DEF *+3 DEF *y* (result) DEF *x* →	

ATTRIBUTES:

	ENTRY POINTS:
	DBLE
Parameters:	Real
Result:	Double Real
Basic FORTRAN:	Callable
FORTRAN IV:	Intrinsic Function: DBLE (*x*)
ALGOL:	Callable
Errors:	None

DCOS

PURPOSE: Calculate the double real cosine of double real x: $y = \cos(x)$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	DCOS (R)	
EXTERNAL REFERENCES:	.ENTR, DSIN, .XFER, .XADD	.ENTP, DSIN, .XFER, .XADD
CALLING SEQUENCES:	JSB DCOS DEF *+3 DEF *y* (result) DEF *x* →	

METHOD:

$y = \cos(x) = \sin(x + \pi/2)$

Accuracy depends on the accuracy of DSIN.

ATTRIBUTES:

	ENTRY POINTS:
	DCOS
Parameters:	Double Real
Result:	Double Real
Basic FORTRAN:	Callable
FORTRAN IV:	Intrinsic Function: DCOS (x)
ALGOL:	Callable
Errors:	None

DDINT

PURPOSE: Truncate a double real x to a double real y:
y = sign (x). (Largest integer $\leq |x|$), or $y = [x]$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	DDINT (P)	
EXTERNAL REFERENCES:	.ENTR, .XFER, .FLUN, .XPAK	.ENTP, .XFER .FLUN, .XPAK
CALLING SEQUENCES:	JSB DDINT DEF *+3 DEF y DEF x →	

ATTRIBUTES:

	ENTRY POINTS:
	DDINT
Parameters:	Double Real
Result:	Double Real
Basic FORTRAN:	Callable
FORTRAN IV:	Intrinsic Function: DDINT (x)
ALGOL:	Callable
Errors:	None

DEXP

PURPOSE: Calculate the double real exponential of a double real x: $y = e^x$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	DEXP (R)	
EXTERNAL REFERENCES:	.ENTR,.ERRR,.XPAK,.XADD,.XSUB,.XMPY,.XDIV,.FLUN,DDINT,SNGL,IFIX,.XFER	.ENTP,.XADD,.XSUB,.XMPY,.XDIV,DDINT,SNGL,IFIX,.FLUN,.XPAK,.XFER
CALLING SEQUENCES:	JSB DEXP DEF *+3 DEF *y* (result) DEF *x* → normal return	JSB DEXP DEF *+3 DEF *y* (result) DEF *x* → error return → normal return

METHOD:

$e^x = 2^N e^z$ where: $z = \ln 2\,(x\log_2 e - N)$

$N = [x\log_2 e \pm 1/2]$ (see DDINT)

$$e^z = Co + \frac{C_1(z(z^2+C_4)+C_3 z)}{(z+B_1)(z(z^2+C_4)+C_3 z)+C_2(z^2+C_4)}$$

$Co = 1.0$	$C_2 = 138.0$	$C_4 = 12.17391304348$
$C_1 = 40.0$	$C_3 = 29.8260869565$	$B_1 = -20.0$

Accuracy: The relative error in $y = e^{x + \Delta x}$ is $R = \Delta x$ where Δx represents the error in the argument. THus for $|x| < 1$, the accuracy will be 11 significant digits, but for $|x|$ near 100, the accuracy will be 8 significant digits.

ATTRIBUTES:

	ENTRY POINTS:
	DEXP
Parameters:	Double Real
Result:	Double Real
Basic FORTRAN:	Not Callable
FORTRAN IV:	Intrinsic Function: DEXP (*x*)
ALGOL:	Not Callable
Errors:	If $e^x > (1-2^{-39})\,2^{127} \rightarrow$ (10 OF)

DIM

PURPOSE: Calculate the positive difference between real x and y: $z = x - \min(x,y)$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	DIM (P)	
EXTERNAL REFERENCES:	.ENTR	.ENTP
CALLING SEQUENCES:	JSB DIM DEF *+3 DEF *x* DEF *y* → z in A & B	

ATTRIBUTES:

	ENTRY POINTS:
	DIM
Parameters:	Real
Result:	Real
Basic FORTRAN:	Callable
FORTRAN IV:	Intrinsic Function: DIM (x,y)
ALGOL:	Callable as Real Procedure
Errors:	None

DIV

(non-EAU Libraries only)

PURPOSE: Divide a two-word integer *I* by the one-word integer *J*: $K = I/J$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.DIV (U)	
EXTERNAL REFERENCES:	None	
CALLING SEQUENCES:	DLD *I* DIV *J* → result in A, remainder in B	or DLD *I* JSB,.DIV DEF *J* → result in A, remainder in B

ATTRIBUTES:

	ENTRY POINTS:
	.DIV
Parameters:	Two-word integer (Note 1), integer
Result:	Integer quotient and remainder in A&B
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	-32768 > quotient > 32767 → overflow, quotient ← 32767

NOTES:

1. B contains most significant bits, A least.
   See MPY.

DLDST
(non-EAU libraries only)

PURPOSE: Store *x*, a two-word quantity in the A and B registers, into memory

	BCS		DOS/RTE (TYPE)
ENTRY POINTS:	.DLD (U) .DST		
EXTERNAL REFERENCES:	GETAD, ADRES		
CALLING SEQUENCES:	JSB .DLD DEF *x* →	or	DLD *x* →
	JSB .DLD DEF *x* →	or	DST *x* →

ATTRIBUTES:

	ENTRY POINTS:	
Call:	.DLD	.DST
Parameters:	Two-word quantity	Two-word quantity: A&B
Result:	Two-word quantity: A&B	Two-word quantity
Basic FORTRAN:	Not callable	Not callable
FORTRAN IV:	Not callable	Not callable
ALGOL:	Not callable	Not callable
Errors:	None	None

DLOG

PURPOSE: Calculate the double real natural logarithm of a double real x:

$$Y = \log_e x$$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	DLOG (R)	
EXTERNAL REFERENCES:	.ENTR, .ERRR, .XADD, .XSUB, .XMPY, .XDIV, .XFER, .FLUN, FLOAT, DBLE	.ENTP, .XADD, .XSUB .XMPY, .XDIV, XFER, .FLUN, FLOAT, DBLE
CALLING SEQUENCES:	JSB DLOG DEF *+3 DEF Y (result) DEF X → normal return	JSB DLOG DEF *+3 DEF Y (result) DEF X → error return → normal return

METHOD:

$$\ln(x) = (n-1/2)\ln 2 + \ln\left(\frac{1+z}{1-z}\right)$$

where: $n = \text{EXPON}(x)$

$m = \text{MANT}(x)$

$$z = \frac{m - \sqrt{2}/2}{m + \sqrt{2}/2}$$

$$\ln \frac{1+z}{1-z} = z\left[\frac{c_1[(z^2+B_2)(z^2+B_3)+C_3]}{(z^2+B_1)[(z^2+B_2)(z^2+B_3)+C_3]+C_2(z^2+B_3)}\right]$$

$C_1 = -18.4800000000$ $\quad B_1 = -15.8484848485$

$C_2 = -23.643709825$ $\quad B_2 = -3.75400078147$

$C_3 = -.246270037272$ $\quad B_3 = -1.39751437005$

Accuracy: See NOTE 1.

ATTRIBUTES:

	ENTRY POINTS:
	DLOG
Parameters:	Double real
Result:	Double real
Basic FORTRAN:	Not callable
FORTRAN IV:	Intrinsic function: DLOG (x)
ALGOL:	Not callable
Errors:	If $x \leq 0 \rightarrow$ (11 UN)

NOTES:

1. The relative error in $Y = \ln(x+\Delta x)$ is $R = \frac{\Delta x}{x \ln x}$. Hence, the relative error increases as x approaches 1. At $x = 1.000 \pm .001$ the accuracy will be 9 significant digits due to an error in the 39th bit in the representation of x. As x diverges from 1 the accuracy becomes 11 significant digits.

DLOGT

PURPOSE: Calculate the double real comman logarithm of double real x:
$y = \log_{10} x$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	DLOGT (U)	
EXTERNAL REFERENCES:	.ENTR, DLOG, .XMPY	.ENTP, DLOG, .XMPY
CALLING SEQUENCES:	JSB DLOGT DEF *+3 DEF y (result) DEF x → normal return	JSB DLOGT DEF *+3 DEF y (result) DEF x → error return → normal return

METHOD:

$y = \log_{10} x = \log_e x$
Accuracy depends on the accuracy of DLOG.

ATTRIBUTES:

	ENTRY POINTS:
	DLOGT
Parameters:	Double Real
Result:	Double Real
Basic FORTRAN:	Not Callable
FORTRAN IV:	Intrinsic Function: DLOGT (x)
ALGOL:	Not Callable
Errors:	If $x < 0$ → (11 UN)

DMOD

PURPOSE: Calculate the double real remainder of two double real values:

$z = x \bmod y \ (z = x - [x/y]y)$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	DMOD (P)	
EXTERNAL REFERENCES:	.ENTR, .XSUB, .XMPY, .XDIV, DDINT	.ENTP, .XSUB, .XMPY, .XDIV, DDINT

CALLING SEQUENCES:

```
JOB DMOD
DEF *+4
DEF z (result)
DEF x
DEF y
→
```

ATTRIBUTES:

	ENTRY POINTS:
	DMOD
Parameters:	Double Real
Result:	Double Real
Basic FORTRAN:	Callable
FORTRAN IV:	Intrinsic Function: DMOD (x,y)
ALGOL:	Callable
Errors:	If $y = 0$, then $z = x$

DSIGN

PURPOSE: Transfer the sign of a double real x to a double real y:

$$z = \text{sign}(y) \cdot |x|$$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	DSIGN (P)	
EXTERNAL REFERENCES:	.ENTR, .XFER, ..DCM	.ENTP, .XFER, ..DCM

CALLING SEQUENCES:

```
JSB DSIGN
DEF *+4
DEF z (result)
DEF x
DEF y
→
```

ATTRIBUTES:

	ENTRY POINTS:
	DSIGN
Parameters:	Double Real
Result:	Double Real
Basic FORTRAN:	Callable
FORTRAN IV:	Intrinsic Function: DSIGN (x,y)
ALGOL:	Callable
Errors:	If $y = 0$, $z = 0$.

DSIN

PURPOSE: Calculate the double real sine of double real x:

$$y = \sin(x)$$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	DSIN (R)	
EXTERNAL REFERENCES:	.ENTR, .XFER, .XPLY, .XADD, .XSUB, .XMPY, .XDIV	.ENTP, .XFER, .XENT .XPLY, .XADD, .XSUB, .XMPY, .XDIV
CALLING SEQUENCES:	JSB DSIN DEF *+3 DEF *Y* DEF *X* →	

METHOD: x is reduced to the range $-\frac{\pi}{2} \leq x < \frac{\pi}{2}$

If $x < 10^{-6}$, $\sin(x) = x$.

Otherwise $\sin(x) = \left(\sum_{i=1}^{6} C_i x^{2i} + 1\right) x$

C_1 = -.166666666667 E+0 C_3 = -.198412663895 E-3 C_5 = -.250294478915 E-7

C_2 .833333331872 E-2 C_4 = .275569300800 E-5 C_6 = .154001500048 E-9

When x is near a non-zero multiple of π, the accuracy of the result is limited by the accuracy of the subtraction $n\pi - x$.

ATTRIBUTES:

	ENTRY POINTS:
	DSIN
Parameters:	Double Real
Result:	Double Real
Basic FORTRAN:	Callable
FORTRAN IV:	Intrinsic Function: DSIN (*x*)
ALGOL:	Callable
Errors:	None

DSQRT

PURPOSE: Calculate the double real square root of double real x: y = sqrt (x)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	DSQRT (R)	
EXTERNAL REFERENCES:	.ENTR, DBLE, SNGL, SQRT, .XDIV, .XADD, .FLUN, .XPAK, .XFER	.ENTP, DBLE, SNGL, SQRT, .XDIV, .XADD, .FLUN, .XPAK, .XFER
CALLING SEQUENCES:	JSB DSQRT DEF *+3 DEF y (result) DEF x → normal return	JSB DSQRT DEF *+3 DEF y (result) DEF x → error return → normal return

METHOD:

A first approximation is found using the single precision SQRT: z = SQRT (x)

Then $y = \frac{z+x/z}{2}$

Accuracy is 11 significant digits.

ATTRIBUTES:

	ENTRY POINTS:
	DSQRT
Parameters:	Double Real
Result:	Double Real
Basic FORTRAN:	Not Callable
FORTRAN IV:	Intrinsic Function: DSQRT (x)
ALGOL:	Not Callable
Errors:	If $x < 0$ → (Ø3 UN)

ENTIE

PURPOSE: 1) Calculate the greatest integer not algebraically exceeding a real *x* (ENTIE);
2) Round a real *x* to the nearest integer; for ties the algebraically larger integer (.RND).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	ENTIE (U) .RND	
EXTERNAL REFERENCES:	None	
CALLING SEQUENCES:	DLD *x* JSB ENTIE → sign in A, integer in B	DLD *x* JSB .RND → result in A

ATTRIBUTES:

	ENTRY POINTS:	
	ENTIE	.RND
Parameters:	Real	Real
Result:	Integer	Integer
Basic FORTRAN:	Not Callable	Not Callable
FORTRAN IV:	Not Callable	Not Callable
ALGOL:	Intr. Funct: ENTIER (*x*)	Not Callable
Errors:	None	None

ENTIX

PURPOSE: Calculate ENTIER of double real *x*:
y = ENTIER (*x*) = greatest integer not algebraically exceeding *x*.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.XENT (P) ENTIX	
EXTERNAL REFERENCES:	.ENTR, XFER, .FLUN, .XPAK	.ENTP, .XFER, .FLUN, .XPAK
CALLING SEQUENCES:	JSB .XENT(or ENTIX) DEF * + 3 DEF *y* DEF *x* →	

ATTRIBUTES:

	ENTRY POINTS:	
	ENTIX	.XENT
Parameters:	Double Real	Double Real
Result:	Double Real	Double Real
Basic FORTRAN:	Callable	Not Callable
FORTRAN IV:	Callable	Not Callable
ALGOL:	Callable	Not Callable
Errors:	None	None

EXP

PURPOSE: Calculate e^x, where x is real.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	EXP (R)	
EXTERNAL REFERENCES:	.ERRR, .IENT, FLOAT, .PWR2	.IENT, FLOAT, .PWR2
CALLING SEQUENCES:	DLD *x* JSB EXP → (*y* in A & B)	DLD *x* JSB EXP JSB ERRØ (error) → (*y* in A & B)

METHOD: Let i=ENTIER(x), and f=x*log$_2$e - 1 (see .IENT)

$$y = 2^i * \left[1 + \frac{2f}{c_4 + c_3 f^2 - f - c_2/(f^2 + c_1)} \right]$$

where

c_1 = 87.417497202

c_2 = 617.9722695

c_3 = 0.03465735903

c_4 = 9.9545957821

ATTRIBUTES:

	ENTRY POINTS:
	EXP
Parameters:	Real: A & B
Result:	Real: A & B
Basic FORTRAN:	Function: EXP (x)
FORTRAN IV:	Function: EXP (x)
ALGOL:	Intr. Proc.: EXP (x)
Errors:	x*log$_2$e ≥ 124 → (Ø7 OF)

NOTES: 1. If the error condition occurs, the overflow bit is set.

FADSB

(floating-point libraries only)

PURPOSE:

Add real X to Y:	Subtract real Y from X:
$Z = X + Y$	$Z = X - Y$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.FAD, .FSB (P)	
EXTERNAL REFERENCES:	.FLUN, .PACK	.FLUN, .PACK, .ZRLB
CALLING SEQUENCES:	DLD X JSB .FAD (.FSB) DEF Y → result in A&B	or DLD X FAD (FSB) Y → result in A&B

ATTRIBUTES:

	ENTRY POINTS:	
	.FAD	.FSB
Parameters:	Real	Real
Result:	Real	Real
Basic FORTRAN:	Not Callable	Not Callable
FORTRAN IV:	Not Callable	Not Callable
ALGOL:	Not Callable	Not Callable
Errors:	See Note 1	See Note 1

NOTES:

1. If the result is outside the range of representable floating point numbers $[-2^{127}, 2^{127}(1-2^{-23})]$ the overflow flag is set and the result $2^{128}(1-2^{-23})$ is returned. If an underflow occurs, (result within the range $(-2^{-129}(1+2^{-22}), 2^{-129})$) the overflow flag is set and the result 0 is returned.

FDV

(non-floating point libraries only)

PURPOSE: Divide real *X* by *Y*: $Z = X/Y$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.FDV (P)	
EXTERNAL REFERENCES:	.FLUN, .PACK	.FLUN, .PACK, .ZRLB
CALLING SEQUENCES:	DLD *X* JSB .FDV DEF *Y* → quotient in A&B or	DLD *X* FDV *Y* → quotient in A&B

ATTRIBUTES:

	ENTRY POINTS:
Call:	.FDV
Parameters:	Real
Result:	Real
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	See FADSB

FLOAT

(non-floating point libraries only)

PURPOSE: Convert integer *I* to real *X*

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	FLOAT (P)	
EXTERNAL REFERENCES:	.PACK	
CALLING SEQUENCES:	LDA *I* JSB FLOAT → (*X* in A & B)	

ATTRIBUTES:

	ENTRY POINTS:
	FLOAT
Parameters:	Integer: A
Result:	Real: A & B
Basic FORTRAN:	Function: FLOAT (*I*)
FORTRAN IV:	Function: FLOAT (*I*)
ALGOL:	Not Callable
Errors:	None

NOTE: 1. The FLIB routine exists in floating-point libraries only. It incorporates and replaces the FADSB, FDV, FMP, FLOAT, and IFIX routines found in non-floating-point libraries. FLIB entry points are: .FAD, .FSB, .FDV, .FMP, FLOAT, and IFIX. The calling sequences for the floating-point routines are the same as for the non-floating-point routines. The floating-point routines modify the calling JSB instruction to a hardware call; a routine is therefore entered only once for each subroutine call.

FMP

(non-floating point libraries only)

PURPOSE: Multiply real *X* by *Y*: *Z* = *X***Y*

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.FMP (P)	
EXTERNAL REFERENCES:	.FLUN, .PACK	.FLUN, .PACK, .ZRLB
CALLING SEQUENCES:	DLD *Y* JSB .FMP DEF *X* → product in A&B	or DLD *Y* FMP *X* → product in A&B

ATTRIBUTES:

	ENTRY POINTS:
Call:	.FMP
Parameters:	Real
Result:	Real
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	See FADSB

IABS

PURPOSE: Calculate absolute value of integer *I*.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		IABS (P)
EXTERNAL REFERENCES:		None
CALLING SEQUENCES:		LDA *I* JSB IABS → (result in A)

ATTRIBUTES:

	ENTRY POINTS:
	IABS
Parameters:	Integer: A
Result:	Integer: A
Basic FORTRAN:	Function: IABS (*I*)
FORTRAN IV:	Function: IABS (*I*)
ALGOL:	Not Callable
Errors:	None

IAND

PURPOSE: Take the logical product of integers *I* and *J*.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	IAND (U)	
EXTERNAL REFERENCES:	None	
CALLING SEQUENCES:	JSB IAND DEF *I* DEF *J* → result in A	

ATTRIBUTES:

	ENTRY POINTS:
	IAND
Parameters:	Integer
Result:	Integer
Basic FORTRAN:	Callable as function.
FORTRAN IV:	Callable as function
ALGOL:	Not Callable
Errors:	None

IDIM

PURPOSE: Calculate the positive difference between integers *I* & *J*:

$K = I - \min(I,J)$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	IDIM (P)	
EXTERNAL REFERENCES:	.ENTR	.ENTP
CALLING SEQUENCES:	JSB IDIM DEF *+3 DEF *I* DEF *J*. → *K* in A	

ATTRIBUTES:

	ENTRY POINTS:
	IDIM
Parameters:	Integer
Result:	Integer
Basic FORTRAN:	Callable
FORTRAN IV:	Intr. function: IDIM (*I*,*J*)
ALGOL:	Callable as integer procedure
Errors:	None

IDINT

PURPOSE: Truncate a double real X to an integer J:

J = Sign (x) . (largest integer $\leq |x|$), or $J = |x|$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		IDINT (P)
EXTERNAL REFERENCES:		SNGL, IFIX, DDINT
CALLING SEQUENCES:		JSB IDINT DEF *+2 DEF x → J in A

ATTRIBUTES:

	ENTRY POINTS:
	IDINT
Parameters:	Double real
Result:	Integer
Basic FORTRAN:	Callable as function
FORTRAN IV:	Function: IDINT (x)
ALGOL:	Callable as integer procedure
Errors:	NOTE 1.

NOTES:

1. If IDINT (x) is out of range, then J = 32767 and the overflow bit is set.

IFIX

(non-floating point libraries only)

PURPOSE: Convert a real x to an integer I.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	IFIX (P)	
EXTERNAL REFERENCES:	.FLUN	
CALLING SEQUENCES:	DLD x JSB IFIX → (I in A)	

ATTRIBUTES:

	ENTRY POINTS:
	IFIX
Parameters:	Real: A & B
Result:	Integer: A
Basic FORTRAN:	Function: IFIX (x)
FORTRAN IV:	Function: IFIX (x)
ALGOL:	Not Callable
Errors:	None

NOTES:

1. Any fractional portion of the result is truncated. If the integer portion is greater than or equal to 2^{15}, the result is set to 32767.

INT

PURPOSE: Truncate a real x to an integer J:

J = Sign (x) . (largest integer $\leq |x|$), or $J = |x|$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	INT (U)	
EXTERNAL REFERENCES:	IFIX	
CALLING SEQUENCES:	DLD x JSB INT $\rightarrow J$ in A	

ATTRIBUTES:

	ENTRY POINTS:
	INT
Parameters:	Real
Result:	Integer
Basic FORTRAN:	Not callable
FORTRAN IV:	Function: INT (x)
ALGOL:	Not callable
Errors:	Note 1

NOTES:

1. If INT (x) is out of range, then J = 32767 and the overflow bit is set.

IOR

PURPOSE: Take logical inclusive - or of integers *I* and *J*.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	IOR (U)	
EXTERNAL REFERENCES:	None	
CALLING SEQUENCES:	JSB IOR DEF *I* DEF *J* → result in A	

ATTRIBUTES:

	ENTRY POINTS:
	IOR
Parameters:	Integer
Result:	Integer
Basic FORTRAN:	Callable as function
FORTRAN IV:	Callable as function
ALGOL:	Not Callable
Errors:	None

ISIGN

PURPOSE: Calculate the sign of *z* times the absolute value of *I*, where *z* is real or integer and *I* is integer: Y=sign(z)*|I|

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		ISIGN (P)
EXTERNAL REFERENCES:		None
CALLING SEQUENCES:		JSB ISIGN DEF *I* DEF *z* → (*Y* in A)

METHOD: Same as SIGN

ATTRIBUTES:

	ENTRY POINTS:
	ISIGN
Parameters:	Real (or int) & integer
Result:	Integer: A
Basic FORTRAN:	Function: ISIGN (*I*,*z*)
FORTRAN IV:	Function: ISIGN (*I*,*z*)
ALGOL:	Not Callable
Errors:	None

MANT

PURPOSE: Extract mantissa of a real x where $x = \text{MANT}(x) * 2^{\text{EXP}(x)}$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		.MANT (P)
EXTERNAL REFERENCES:		None
CALLING SEQUENCES:		DLD JSB.MANT → Real Mantissa in A & B

METHOD: Accuracy is 23 bits.

ATTRIBUTES:

	ENTRY POINTS:
	.MANT
Parameters:	Real
Result:	Real
Basic FORTRAN:	Not Callable
FORTRAN IV:	Not Callable
ALGOL:	Not Callable
Errors:	None

MOD

PURPOSE: Calculate the integer remainder of *I/J* for integer *I* & *J*;
K = *I* modulo *J*

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	MOD (P)	
EXTERNAL REFERENCES:	.ENTR	.ENTP
CALLING SEQUENCES:	JSB MOD DEF *+3 DEF *I* DEF *J* → *K* in A & B	

METHOD: *K = I - [I/J]*J*

ATTRIBUTES:

	ENTRY POINTS:
	MOD
Parameters:	Integer
Result:	Integer
Basic FORTRAN:	Callable as function
FORTRAN IV:	Intrinsic function: MOD (*I*,*J*)
ALGOL:	Callable as integer procedure
Errors:	If *J* = 0, then *K* = *I*

MPY

(non-EAU libraries only)

PURPOSE: Multiply integer *I* and *J*: *K* = *I***J*

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.MPY (U)	
EXTERNAL REFERENCES:	None	
CALLING SEQUENCES:	LDA *J* JSB .MPY DEF *I* → *K* in A&B (Note 1)	or LDA *J* MPY *I* → *K* in A&B (Note 1)

ATTRIBUTES:

	ENTRY POINTS:
	.MPY
Parameters:	Integer
Result:	Two-word integer (Note 1)
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

NOTES:

1. B contains most significant bits of product;
   A contains least significant bits.

MXMND

PURPOSE: Calculate the maximum or minimum of a series of double real values:
Y = max (*A*,*B*,*C*,....) *Y* = min (*A*,*B*,*C*,....)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	DMAX1 (R) DMIN1	
EXTERNAL REFERENCES:	.XSUB .XFER	
CALLING SEQUENCES:	JSB DMAX1(or DMIN1) DEF *+*N*+2 DEF *Y (result)* DEF *A* (1) DEF *B* (2) : · : DEF *X* (*N*) →	

ATTRIBUTES:

	ENTRY POINTS:	
	DMAX1	DMIN1
Parameters:	Double Real	Double Real
Result:	Double Real	Double Real
Basic FORTRAN:	Callable as Subroutine	Callable as Subroutine
FORTRAN IV:	Note 1	Note 1
ALGOL:	Note 2	Note 2
Errors:	If *N* < 2, then *Y* = 0	If *N* < 2, then *Y* = 0

NOTES:

1. Intrinsic functions: DMAX1 (*A*,*B*,*C*, ...)
   DMIN1 (*A*,*B*,*C*, ...)
2. Callable, but only with a fixed number and parameters.

COMMENTS: Requires at least two parameters.

MXMNI

PURPOSE: Calculate the maximum or minimum of a series of integer values:

Y = MAX ($A,B,C,$ ......) Y = MIN ($A,B,C,$ ......)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	AMAXØ, MAXØ, AMINØ, MINØ (R)	
EXTERNAL REFERENCES:	FLOAT	
CALLING SEQUENCES:	JSB Entry Point DEF *+N+1 DEF A (1) DEF B (2) : DEF X (N) → Result in A or A & B	

ATTRIBUTES:

ENTRY POINTS:

	AMAXØ	MAXØ	AMINØ	MINØ
Parameters:	Integer	Integer	Integer	Integer
Result:	Real	Integer	Real	Integer
Basic FORTRAN:	Note 1	Note 1	Note 1	Note 1
FORTRAN IV:	Note 1	Note 1	Note 1	Note 1
ALGOL:	Note 2	Note 2	Note 2	Note 2
Errors:	Note 3	Note 3	Note 3	Note 3

NOTES:
1. Functions: AMAXØ (A,B,C ....), MAXØ (A,B,C ....) AMNØ (A,B,C ....), MINØ (A,B,C....)
2. Callable as integer or real procedure, but only with a fixed number of parameters.
3. If the number of parameters is less than 2, Y = Ø.

COMMENTS: Requires at least two parameters.
AMAXØ provides a real maximum.
MAXØ provides an integer maximum.
AMINØ provides a real minimum.
MINØ provides an integer minimum.

MXMNR

PURPOSE: Calculate the maximum or minimum of a series of real values:

Y = Max (*A*,*B*,*C* ....)   *Y* = Min (*A*,*B*,*C* ....)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	AMAX1, MAX1, AMIN1, MIN1 (R)	
EXTERNAL REFERENCES:	IFIX	
CALLING SEQUENCES:	JSB Entry Point DEF *+ *N* + 1 DEF *A* (1) DEF *B* (2) : : DEF *X* (*N*) → *Y* in A or A & B	

ATTRIBUTES:

ENTRY POINTS:

	AMAX1	MAX1	AMIN1	MIN1
Parameters:	Real	Real	Real	Real
Result:	Real	Integer	Real	Integer
Basic FORTRAN:	Note 1	Note 1	Note 1	Note 1
FORTRAN IV:	Note 1	Note 1	Note 1	Note 1
ALGOL:	Note 2	Note 2	Note 2	Note 2
Errors:	Note 3	Note 3	Note 3	Note 3

NOTES:

1. Functions: AMAX1 (*A*,*B*,*C*, ...), MAX1 (*A*,*B*,*C*, ...),
   AMIN1 (*A*,*B*,*C*, ...), MIN1 (*A*,*B*,*C*, ...).
2. Callable as integer or real procedure, but only with a fixed number or parameters.
3. If the number of parameters is less than 2, *Y* = Ø.

COMMENTS: Requires at least two parameters.

AMAX1 provides a real maximum.
MAX1 provides an integer maximum.
AMIN1 provides a real minimum.
MIN1 provides an integer minimum.

PWR2

PURPOSE: Calculate $x.2^n$ for real x and integer n: $y = x.2^n$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.PWR2 (P)	
EXTERNAL REFERENCES:	.FLUN	
CALLING SEQUENCES:	DLD x JSB .PWR2 DEC n $\rightarrow y$ in A & B	

METHOD:

Exponent of x is increased by n.
Accuracy is 23 bits.

ATTRIBUTES:

	ENTRY POINTS:
	.PWR2
Parameters:	Real & Integer
Result:	Real
Basic FORTRAN:	Not Callable
FORTRAN IV:	Not Callable
ALGOL:	Not Callable
Errors:	None

REAL

PURPOSE: Extract the real part of a complex *x*.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	REAL (P)	
EXTERNAL REFERENCES:	.ENTR	.ENTP
CALLING SEQUENCES:	JSB REAL DEF *+2 DEF *x* → result in A & B	

ATTRIBUTES:

	ENTRY POINTS:
	REAL
Parameters:	Complex
Result:	Real
Basic FORTRAN:	Callable as Function
FORTRAN IV:	Intr. Function: REAL (*x*)
ALGOL:	Callable as real procedure
Errors:	None

SICOS

PURPOSE: Calculate the sine or cosine of a real x (radians): y = sine (x) or
y = cosine (x)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	SIN (R) COS	
EXTERNAL REFERENCES:	..FCM, .IENT, .PWR2, FLOAT, .CHEB	
CALLING SEQUENCES:	DSD x JSB SIN (or COS) → y in A&B	DLD x JSB SIN (or COS) JSB ERRØ → y in A&B

METHOD:

```
x = x *2/Π                                   y = COS (x) = - SIN (x - π/2)
x = x -4 *ENTIER ((x+1)4)  (See .IENT)
 If x > 1 then x = 2 - x
 y = x *CHEBY (2*x*x-1)
```

ATTRIBUTES:

	ENTRY POINTS:	
	SIN	COS
Parameters:	Real Radians: A and B	Real Radians: A and B
Result:	Real: A and B	Real: A and B
Basic FORTRAN:	Function: SIN (x)	Function: COS (x)
FORTRAN IV:	Function: SIN (x)	Function: COS (x)
ALGOL:	Intr. Proc: SIN (x)	Intr. Proc: COS (x)
Errors:	See note 2.	Same

NOTES:

1. If the error condition occurs, the overflow bit is set.
2. $\frac{1}{2}\left|\frac{X}{\pi} + \frac{1}{2}\right| > 2^{14} \rightarrow$ (Ø5 OR)

SIGN

PURPOSE: Calculate the sign of *z* times the absolute value of *x*, where *z* is real or integer and *x* is real; if *z* = Ø, then the result equals Ø.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		SIGN (P)
EXTERNAL REFERENCES:		..FCM
CALLING SEQUENCES:		JSB SIGN DEF *x* DEF *z* → (result in A & B)

ATTRIBUTES:

	ENTRY POINTS:
	SIGN
Parameters:	Real or Integer and Real
Result:	Real
Basic FORTRAN:	Function: SIGN (*x*, *z*)
FORTRAN IV:	Function: SIGN (*x*, *z*)
ALGOL:	Not Callable
Errors:	None

SNGL

PURPOSE: Convert a double real *x* to a real *y*

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		SNGL (P)
EXTERNAL REFERENCES:		.XFER, .FLUN, .PACK
CALLING SEQUENCES:		JSB SNGL DEF *+2 DEF *x* → *y* in A & B

ATTRIBUTES:

	ENTRY POINTS:
	SNGL
Parameters:	Double Real
Result:	Real
Basic FORTRAN:	Callable
FORTRAN IV:	Intr. Function: SNGL (*x*)
ALGOL:	Callable as Real Procedure
Errors:	Note 1

NOTES:

1. If $x > (1-2^{-23})*2^{127}$ (the maximum real number), then $y = (1-2^{-23})*2^{127}$, and the overflow bit is set.

SQRT

PURPOSE: Calculate the square root of a real x: $y = \sqrt{x}$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	SQRT (R)	
EXTERNAL REFERENCES:	.FLUN, .PWR2, .ERRR	.FLUN, .PWR2
CALLING SEQUENCES:	DLD x JSB SQRT → (y in A and B)	DLD JSB SQRT JSB ERRØ (error) → (y in A and B)

METHOD: Choose f such that $x=2^{2b}(f)$, $.25 \le f < 1$ Then $\sqrt{x} = 2^b * \sqrt{f}$.

$\sqrt{f}$ is approximated by $p_1 = c_1 f + c_2$, where for $.25 \le f < .5$, $c_1=.875$, $c_2=.27863$ and for $.5 \le f < 1$, $c_1=.578125$, $c_2=.421875$

This approximation is improved by two Newton iterations: $p_2 = (p_1 + f/p_1)/2$

$p_3 = (p_2 + f/p_2)/2$

p_3 is the final result

ATTRIBUTES:

	ENTRY POINTS:
	SQRT
Parameters:	Real: A & B
Result:	Real: A & B
Basic FORTRAN:	Function: SQRT (x)
FORTRAN IV:	Function: SQRT (x)
ALGOL:	Intr. Proc: SQRT (x)
Errors:	$x < Ø$ → (Ø3 UN)

NOTES: 1. If the error condition occurs, the overflow bit is set.

TAN

PURPOSE: Calculate the tangent of a real x (radians): y = tangent (x)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	TAN (R)	
EXTERNAL REFERENCES:	.PWR2, ..FCM, .IENT, .CHEB, FLOAT, .ERRR	.PWR2, ..FCM, .IENT, .CHEB, FLOAT
CALLING SEQUENCES:	DLD x JSB TAN → (y in A & B)	DLD x JSB TAN JSB ERR∅ (error) → (y in A & B)

METHOD:

```
x = 4*x/π
x = x-4* ENTIER((x+1)/4) (See .IENT)
If x>1 then w = 2-x else w = x
w = w *CHEBY(2*w*w-1)
If x>1 then y = 1/w else y = w
```

ATTRIBUTES:

	ENTRY POINTS:
	TAN
Parameters:	Real: A and B
Result:	Real: A and B
Basic FORTRAN:	Function: TAN (x)
FORTRAN IV:	Function: TAN (x)
ALGOL:	Intr. Proc: TAN (x)
Errors:	$x>2^{14}$ →(∅9 OR), tan (x) $>2^{128}$ → overflow

NOTES: 1. If the error condition occurs, the overflow bit is set.

TANH

PURPOSE: Calculate the hyperbolic tangent of a real x: y=TANH (x)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	TANH (R)	
EXTERNAL REFERENCES:	.PWR2, EXP, ..FCM, ABS, .FLUN	
CALLING SEQUENCES:	DLD x JSB TANH → (y in A and B)	

METHOD:

1. $x \geq 0$
   - a. $x \geq 16$ TANH (x) = 1
   - b. $.125 \leq x < 16$ TANH (x) = (EXP(2*x)-1)/(EXP(2*x) + 1)
   - c. $.00005 \leq x < .125$ $\text{TANH}(x) = \left(c_1 + f^2\left[c_2 + c_3(c_4 + f^2)^{-1}\right]\right)^{-1}$

     where:

     $f = 4*x*\log_2 e$

     c_1= 5.7707801636

     c_2= .01732867951

     c_3=14.1384114018

     c_4=349.6699888
   - d. $x < .00005$ TANH(x) = x
2. $x < 0$ TANH (x) = -TANH ($-x$)

ATTRIBUTES:

	ENTRY POINTS:
	TANH
Parameters:	Real: A and B
Result:	Real: A and B
Basic FORTRAN:	Function: TANH (x)
FORTRAN IV:	Function: TANH (x)
ALGOL:	Intr. Proc: TANH (x)
Errors:	None

XADSB

PURPOSE: Double real addition and subtraction: $z = x + y$ $z = x - y$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.XADD (P) XADD .XSUB XSUB	
EXTERNAL REFERENCES:	.XFER, .FLUN, .XPAK, .XCOM, GETAD, ADRES	.PCAD, .XFER, .FLUN, .XPAK, .XCOM

CALLING SEQUENCES:

```
JSB(.XADD or .XSUB)                  JSB(XADD or XSUB)
DEF z (result)               or      DEF*+4
DEF x                                DEF z (result)
DEF y                                DEF x
→                                    DEF y
                                     →
```

ATTRIBUTES:

ENTRY POINTS:

	.XADD	XADD	.XSUB	XSUB
Parameters:	Double Real	Double Real	Double Real	Double Real
Result:	Double Real	Double Real	Double Real	Double Real
Basic FORTRAN:	Not Callable	Callable	Not Callable	Callable
FORTRAN IV:	Not Callable	Callable	Not Callable	Callable
ALGOL:	Not Callable	Callable	Not Callable	Callable
Errors:	Note 1	Note 1	Note 1	Note 1

NOTES: 1. If z is outside the range: $[-2^{128}, 2^{127}(1-2^{-39})]$, then the overflow bit is set and $z = 2^{127}(1-2^{-39})$.

If the result is within the range: $[-2^{-129}(1+2^{-22}), 2^{-129}]$, then the overflow bit is set and $z = 0$.

XDIV

PURPOSE: Divide a double real *x* by double real *y*: *z* = *x* / *y*

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.XDIV (P) XDIV	
EXTERNAL REFERENCES:	.XFER, .XCOM, .FLUN, .XPACK, GETAD, ADRES	.PCAD, .XFER, .XCOM, .FLUN, .XPAK
CALLING SEQUENCES:	JSB .XDIV DEF *z* (result) DEF *x* DEF *y* →	or JSB XDIV DEF * + 4 DEF *z* (result) DEF *x* DEF *y* →

ATTRIBUTES:

	ENTRY POINTS:	
	XDIV	.XDIV
Parameters:	Double Real	Double Real
Result:	Double Real	Double Real
Basic FORTRAN:	Callable	Not Callable
FORTRAN IV:	Callable	Not Callable
ALGOL:	Callable	Not Callable
Errors:	See XADSB	See XADSB

XMPY

PURPOSE: Multiply double real *X* by double real *Y*: *Z* = *X***Y*

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.XMPY (P) XMPY	
EXTERNAL REFERENCES:	.XFER, .FLUN, .XPAK, .XCOM, GETAD, ADRES	.PCAD, .XFER, .FLUN, .XPAK, .XCOM

CALLING SEQUENCES:

```
JSB .XMPY            or          JSB XMPY
DEF Z (result)                   DEF * + 4
DEF X                            DEF Z (result)
DEF Y                            DEF X
→                                DEF Y
                                 →
```

ATTRIBUTES:

ENTRY POINTS:

	XMPY	.XMPY
Parameters:	Double Real	Double Real
Result:	Double Real	Double Real
Basic FORTRAN:	Callable	Not Callable
FORTRAN IV:	Callable	Not Callable
ALGOL:	Callable	Not Callable
Errors:	See XADSB	See XADSB

XPOLY

PURPOSE: Evaluate double real polynomial: $Y = c_1 X^{n-1} + c_2 X^{n-2} + \ldots + c_{n-1} X + c_n$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.XPLY (R) XPOLY	
EXTERNAL REFERENCES:	.ENTR, .XFER, .XADD, .XMPY	
CALLING SEQUENCES:	JSB .XPLY or XPOLY DEF * + 5 DEF Y DEF n (degree + 1) DEF X DEF c_1 (first element of coefficient array)	

ATTRIBUTES:

ENTRY POINTS:

	.XPLY	XPOLY
Parameters:	Double Real,Integer	Double Real, Integer
Result:	Double Real	Double Real
Basic FORTRAN:	Not Callable	Callable
FORTRAN IV:	Not Callable	Callable
ALGOL:	Not Callable	Callable
Errors:	If $n \leq 0$, $Y = 0$	If $n \leq 0$, $Y = 0$

.CDBL

PURPOSE: Converts a complex *x* to real *y*

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		.CDBL (U)
EXTERNAL REFERENCES:		REAL DBLE
CALLING SEQUENCES:		JSB .CDBL DEF *y* (DP result) DEF *x* (complex) →

ATTRIBUTES:

	ENTRY POINTS:
	.CDBL
Parameters:	Complex
Result:	Double real
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

.CFER

PURPOSE: Transfer a complex *x* to complex *y*

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		.CFER (U)
EXTERNAL REFERENCES:		GETAD ADRES
CALLING SEQUENCES:		JSB .CFER DEF *y* DEF *x* →

ATTRIBUTES:

	ENTRY POINTS:
	.CFER
Parameters:	Complex
Result:	Complex
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

.CINT

PURPOSE: Convert a complex *x* to an integer.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		.CINT (U)
EXTERNAL REFERENCES:		REAL IFIX
CALLING SEQUENCES:		JSB .CINT DEF *x* →result in A

ATTRIBUTES:

	ENTRY POINTS:
	.CINT
Parameters:	Complex
Result:	Integer in A
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

.CTOI

PURPOSE: Raise a complex x to an integer power I: $z = x^I$ (z is complex)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.CTOI (R)	
EXTERNAL REFERENCES:	.ERRR, .CMPY, .CDIV, GETAD, ADRES	.CMPY, .CDIV, .PCAD
CALLING SEQUENCES:	JSB .CTOI DEF z (result) DEF x DEF J → Normal Return	JSB .CTOI DEF z (result) DEF x DEF J → Error Return → Normal Return

METHOD:

See .RTOI

ATTRIBUTES:

	ENTRY POINTS:
	.CTOI
Parameters:	Complex & integer
Result:	Complex
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	$x = 0$, $I \leq 0$ → (14 UN)

.DCPX

PURPOSE: Converts a double real *x* to a complex *y*.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		.DCPX (U)
EXTERNAL REFERENCES:		SNGL CMPLX
CALLING SEQUENCES:		JSB .DCPX DEF *y* DEF *x* →

ATTRIBUTES:

	ENTRY POINTS:
	.DCPX
Parameters:	Double real
Result:	Complex
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

.DINT

PURPOSE: Converts a double real *x* to an integer.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		.DINT (U)
EXTERNAL REFERENCES:		SNGL IFIX
CALLING SEQUENCES:		JSB .DINT DEF *x* → result in A

ATTRIBUTES:

	ENTRY POINTS:
	.DINT
Parameters:	Double real
Result:	Integer in A
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

.DTOD

PURPOSE: Raise a double real x to a double real power y:
$z = x^y$ (z is double real)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.DTOD (R)	
EXTERNAL REFERENCES:	DEXP, DLOG .XMPY, .XFER	
CALLING SEQUENCES:	JSB .DTOD DEF z (result) DEF x DEF y → normal return	JSB .DTOD DEF z (result) DEF x DEF y → error return → normal return

METHOD:

If $x = 0$ and $y>0$, $z = 0$.
If $x \neq 0$ and $y = 0$, $z = 1$.
If $x>0$ and $y \neq 0$, $z = \text{EXP}(y*\log(x))$

Accuracy depends on the accuracy of DLOG and DEXP.

ATTRIBUTES:

	ENTRY POINTS:
	.DTOD
Parameters:	Double real
Result:	Double real
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	Note 1

NOTES: 1.

$x = 0$, $y \leq 0$	→ (13 UN)
$x<0$, $y \neq 0$	→ (13 UN)
$x>(1-2^{-39})2^{127}$	→ (10 OF)

.DTOI

PURPOSE: Calculate a double real X raised to an integer power I:

$Y = X^I$ (Y is double real)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.DTOI (R)	
EXTERNAL REFERENCES:	.ERRR, .XMPY, .XDIV, .XFER	.XMPY, .XDIV, .XFER
CALLING SEQUENCES:	JSB .DTOI DEF Y (result) DEF X DEF I → Normal return	JSB .DTOI DEF Y (result) DEF X DEF I → Error return → Normal return

METHOD:

See .RTOI

ATTRIBUTES:

	ENTRY POINTS:
	.DTOI
Parameters:	Double real & integer
Result:	Double real
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	If $X = 0$, $I \leq 0$ → (12 UN)

.DTOR

PURPOSE: Raise a double real x to a real power y:

$z = x^y$ (z is double real)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.DTOR (U)	
EXTERNAL REFERENCES:	.DTOD DBLE	
CALLING SEQUENCES:	JSB .DTOR DEF z (result) DEF x DEF y → normal routine	JSB .DTOR DEF z (result) DEF x DEF y → error return → normal return

METHOD:

Convert y to double precision and call .DTOD.

ATTRIBUTES:

	ENTRY POINTS:
	.DTOR
Parameters:	Real & double real
Result:	Double real
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	See .DTOD

.EAU.

(EAU libraries only)

PURPOSE: Replace calls to .MPY, .DIV, .DLD, and .DST with hardware EAU instructions.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.MPY (U) .DIV .DLD .DST	
EXTERNAL REFERENCES:	None	
CALLING SEQUENCES:	See MPY, DIV, DLDST	

.FLUN

PURPOSE: "Unpack" a real *x*; place exponent in A, lower part of mantissa in B.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.FLUN (P)	
EXTERNAL REFERENCES:	None	.ZRLB
CALLING SEQUENCES:	DLD *x* JSB .FLUN → exponent in A Lower mantissa in B	

ATTRIBUTES:

	ENTRY POINTS:
	.FLUN
Parameters:	Real
Result:	A & B
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

.ICPX

PURPOSE: Converts an integer *I* to a complex *Y*.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		.ICPX (U)
EXTERNAL REFERENCES:		FLOAT CMPLX
CALLING SEQUENCES:		LDA *I* JSB .ICPX DEF *Y* →

ATTRIBUTES:

	ENTRY POINTS:
	.ICPX
Parameters:	Integer in A
Result:	Complex
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

.IDBL

PURPOSE: Converts an integer *I* to double real *Y*.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		.IDBL (U)
EXTERNAL REFERENCES:		FLOAT DBLD
CALLING SEQUENCES:		.LDA *I* JSB .IDBL DEF *Y* →

ATTRIBUTES:

	ENTRY POINTS:
	.IDBL
Parameters:	Integer in A
Result:	Double
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

.IENT

PURPOSE: Calculate ENTIER (x) for real x: I = ENTIER (x)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		.IENT (P)
EXTERNAL REFERENCES:		IFIX, .FLUN, FLOAT
CALLING SEQUENCES:		DLD *x* JSB .IENT JSB *error routine* → *I* in A

ATTRIBUTES:

	ENTRY POINTS:
	.IENT
Parameters:	Real
Result:	Integer
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	EXPO (x) > 14, user must supply error routine

.ITOI

PURPOSE: Calculate I^J for integer I and J: $K = I^J$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.ITOI (P)	
EXTERNAL REFERENCES:	.ERRR	None
CALLING SEQUENCES:	JSB .ITOI DEF I DEF J $\rightarrow K$ in A	JSB .ITOI DEF I DEF J JSB ERRØ (error return) $\rightarrow K$ in A

ATTRIBUTES:

	ENTRY POINTS:
	.ITOI
Parameters:	Integer
Result:	Integer
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	See Note 1.

NOTES: 1.

Condition	Error Code
$I = 0,\ J \leq 0$	Ø8 UN
$I^J \geq 2^{23}$	Ø8 OF

On error return, overflow bit is set.

.PACK

PURPOSE: Convert signed mantissa of real *x* into normalized real format.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.PACK (P)	
EXTERNAL REFERENCES:	None	.ZRLB
CALLING SEQUENCES:	DLD *x* JSB .PACK BSS 1 (exponent) → result in A & B	

ATTRIBUTES:

	ENTRY POINTS:
	.PACK
Parameters:	Mantissa in A & B
Result:	Real
Basic FORTRAN:	Not Callable
FORTRAN IV:	Not Callable
ALGOL:	Not Callable
Errors:	None

.RTOD

PURPOSE: Raise a real x to a double real power y: $z=x^y$ (z is double real)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.RTOD (U)	
EXTERNAL REFERENCES:	.DTOD DBLE	
CALLING SEQUENCES:	JSB .RTOD DEF *z* (result) DEF *x* DEF *y* → Normal Return	JSB .RTOD DEF *z* (result) DEF *x* DEF *y* → Error Return → Normal Return

METHOD: Convert *x* to double real and call .DTOD.

ATTRIBUTES:

	ENTRY POINTS:
	.RTOD
Parameters:	Real and Double Real
Result:	Double Real
Basic FORTRAN:	Not Callable
FORTRAN IV:	Not Callable
ALGOL:	Not Callable
Errors:	See .DTOD

.RTOI

PURPOSE: Calculate x^I for real x and integer I: $Y=X^I$.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.RTOI (R)	
EXTERNAL REFERENCES:	.ERRR	None
CALLING SEQUENCES:	JSB .RTOI DEF X DEF I $\rightarrow Y$ in A & B	JSB .RTOI DEF X DEF I JSB ERRØ $\rightarrow Y$ in A & B

METHOD: The only possibility of inaccuracy is that introduced by roundoff in the FMP or the FDV routine if $I < 0$.

X^I gives the same result as the expression:

$$\underbrace{X*X*X*...*X}_{I\text{ times}} \quad \text{or} \quad 1/\underbrace{X*X*X*....*X}_{I\text{ times}}$$

ATTRIBUTES:

	ENTRY POINTS:
	.RTOI
Parameters:	Real & Integer
Result:	Real
Basic FORTRAN:	Not Callable
FORTRAN IV:	Not Callable
ALGOL:	Not Callable
Errors:	See Note 1

NOTES: 1.

Condition	Error Code
$X = 0$, $I < 0$	Ø6 UN
$X^{\lvert I \rvert} > 2^{128}$	(floating point overflow)

On error return, overflow bit is set.

.RTOR

PURPOSE: Calculate x^Y for real x and Y: $z = x^Y$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.RTOR (R)	
EXTERNAL REFERENCES:	ALOG, EXP, .ERRR	ALOG, EXP
CALLING SEQUENCES:	JSB .RTOR DEF x DEF Y → z in A & B	JSB .RTOR DEF x DEF Y JSB ERRØ → z in A & B

ATTRIBUTES:

	ENTRY POINTS:
	.RTOR
Parameters:	Real
Result:	Real
Basic FORTRAN:	Not Callable
FORTRAN IV:	Not Callable
ALGOL:	Not Callable
Errors:	See Note 1

NOTES: 1.

Condition	Error Code
$x = 0$, $Y \leq 0$ } < 0, $\neq 0$ }	Ø4 UN
$\lvert x$*ALOG(x)$\rvert \geq 124$	Ø7 OF

On error return, the overflow bit is set.

.XCOM

PURPOSE: Complements a double real unpacked mantissa in place. Upon return, A-register = 1 if exponent should be adjusted; otherwise A = 0.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		.XCOM (P)
EXTERNAL REFERENCES:		.XFER
CALLING SEQUENCES:		JSB .XCOM DEF *x* ADA (exponent) STA (exponent)

ATTRIBUTES:

	ENTRY POINTS:
	.XCOM
Parameters:	Double real
Result:	Double real
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

.XFER

PURPOSE: Double real transfer: $Y = X$

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.XFER (P) .DFER	
EXTERNAL REFERENCES:	None	
CALLING SEQUENCES:	LDA (address of X) LDB (address of Y) JSB .XFER →	OR JSB .DFER DEF Y DEF X →

ATTRIBUTES:

ENTRY POINTS:

	.XFER	.DFER
Parameters:	Double real	Double real
Result:	Double real	Double real
Basic FORTRAN:	Not callable	Not callable
FORTRAN IV:	Not callable	Not callable
ALGOL:	Not callable	Not callable
Errors:	None	None

.XPAK

PURPOSE: Double real mantissa is normalized, rounded, and packed with exponent; result is double real.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		.XPAK (P)
EXTERNAL REFERENCES:		.XFER
CALLING SEQUENCES:		LDA exponent JSB .XPAK DEF *x* (3-word mantissa) → result in *x*

ATTRIBUTES:

	ENTRY POINTS:
	.XPAK
Parameters:	Double real, exponent
Result:	Double real
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	See XADSB

..CCM

PURPOSE: Complements a complex variable *x* in place.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		..CCM (U)
EXTERNAL REFERENCES:		GETAD ADRES ..FCM
CALLING SEQUENCES:		JSB ..CCM DEF *x* →

ATTRIBUTES:

	ENTRY POINTS:
	..CCM
Parameters:	Complex
Result:	Complex
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

..DCM

PURPOSE: Double real compliment.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	..DCM (P)	
EXTERNAL REFERENCES:	.FLUN, .XCOM .XPAK, .XFER	
CALLING SEQUENCES:	JSB ..DCM DEF *x* →	

ATTRIBUTES:

	ENTRY POINTS:
	..DCM
Parameters:	Double
Result:	Double real
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

..DLC

PURPOSE: Load and complement a real *x*.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		..DLC (P)
EXTERNAL REFERENCES:		None
CALLING SEQUENCES:		JSB ..DLC DEF *x* → compliment in A & B.

ATTRIBUTES:

	ENTRY POINTS:
	..DLC
Parameters:	Real
Result:	Real
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

..FCM

PURPOSE: Complement real *x*

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	..FCM (P)	
EXTERNAL REFERENCES:	..DLC	
CALLING SEQUENCES:	DLD *x* JSB ..FCM → result in A & B	

ATTRIBUTES:

	ENTRY POINTS:
	..FCM
Parameters:	Real
Result:	Real
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

SECTION II

UTILITY SUBROUTINES

AXIS

PURPOSE: Plots one axis (*x* or *y*) of a graph with a specified axis label, a specified length, and specified values at each inch marker.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	AXIS (U)	
EXTERNAL REFERENCES:	NUMB, SYMB, PLOT, and numerous library subroutines	

CALLING SEQUENCES:

```
JSB AXIS
DEF *+9
DEF floating-point origin: x coordinate
DEF floating-point origin: Y coordinate
DEF axis label
DEF integer number of characters in label (positive to
    place label counterclockwise to axis--as in Y axis--
    and negative to place label clockwise--as in X axis)
DEF length of axis in floating-point inches
DEF angle of axis in floating-point degrees
DEF minimum value of axis (calculated by SCALE)
DEF incremental value (calculated by SCALE)
→ normal return
```

ATTRIBUTES:

	ENTRY POINTS:
	AXIS
Parameters:	Mixed
Result:	N/A
Basic FORTRAN:	Callable as subroutine
FORTRAN IV:	Callable as subroutine
ALGOL:	Callable as CODE procedure
Errors:	None

NOTES:

1. SCALE must be called before AXIS.
2. AXIS calls SYMB to plot the labels 0.14 inches high.
3. Sample calls to AXIS:

   Plot the X axis, starting at (0,0) with the label "POWER" on the clockwise side, 6.5 inches long, at 0 degrees.

   ```
   CALL AXIS (Ø.Ø,Ø.Ø,IPWR,-5,6.5,0.0,X(51),X(52))
   ```

   Plot a similar Y axis with the label "PSI" on the counterclockwise side, ten inches long at 90 degrees.

   ```
   CALL AXIS (Ø.Ø,Ø.Ø,IPSI,3,1Ø.Ø,9Ø.Ø,Y(51),Y(52))
   ```

BINRY

PURPOSE: Reads or writes data at a specified location (logical unit number, track, sector, and offset) of a disc.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	N/A	BREAD, BWRIT (U)
EXTERNAL REFERENCES:		EXEC, .OPSY
CALLING SEQUENCES:		JSB BREAD (of BWRIT) (Note 1) DEF *+7 DEF buffer DEF buffer length (words) DEF logical unit DEF track DEF sector DEF offset (Note 2) →

ATTRIBUTES:

	ENTRY POINTS:	
	BREAD	BWRIT
Parameters:	Mixed	Mixed
Result:	Mixed	Mixed
Basic FORTRAN:	Callable	Callable
FORTRAN IV:	Callable	Callable
ALGOL:	Callable	Callable
Errors:	None	None

NOTES:

1. BREAD is the read entry point and BWRIT is the write entry point.

2. Offset: If the offset equals Ø, the transfer begins on the sector boundary; if the offset equals n, the transfer skips n words into the sector before starting.

CLRIO

PURPOSE: Performs a system clear request which makes all I/O devices available for the initiation of a new operation. (In RTE/DOS, CLRIO is a dummy compatibility routine.)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	CLRIO (U)	
EXTERNAL REFERENCES:	.IOC.	None
CALLING SEQUENCES:	JSB CLRIO DEF *+1 →	

ATTRIBUTES:

	ENTRY POINTS:
	CLRIO
Parameters:	None
Result:	None
Basic FORTRAN:	Callable
FORTRAN IV:	Callable
ALGOL:	Callable
Errors:	None

CODE

PURPOSE: Provides internal conversion according to a FORMAT from one core area to another core area.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		CODE (P) ACODE
EXTERNAL REFERENCES:		None
CALLING SEQUENCES:		JSB CODE DEF *+1 Read or write request (see Note 1)

METHOD: Utilizes the internal conversion capability of the Formatter.

ATTRIBUTES:

	ENTRY POINTS:	
	CODE	ACODE
Parameters:	None	None
Result:	None	None
Basic FORTRAN:	Callable	Callable
FORTRAN IV:	Callable	Callable
ALGOL:	Not callable (Note 2)	Callable (Note 2)
Errors:	None	None

NOTES:

1. The call to CODE must immediately precede a READ or WRITE request where the identifier of an ASCII record buffer replaces the logical unit number. Any labels must be attached to the CODE call, as it and the READ/WRITE call are treated as one statement.

   In FORTRAN the calling sequences are:

   ```
   CALL CODE                  CALL CODE
   READ (v,n)L                WRITE (v,n)L
   ```

   where *v* is the unsubscripted identifier of an ASCII record buffer;
   n is the number of a FORMAT Statement; and
   L is an Input/Output List of variables.

   On read, the contents of the ASCII record *v* are converted according to the FORMAT *n* and are stored in the variables listed in *L*.

   On write, the contents of the variables listed in *L* are converted to ASCII according to FORMAT *n* and the ASCII characters are stored in *v*.

2. ALGOL programmers must use the entry point ACODE instead of CODE.

DBKPT

PURPOSE: Processes breakpoints for DOS/RTE DEBUG. Never called by user programs. See DEBUG

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	N/A	$DBP2, $MEMR (U)
EXTERNAL REFERENCES:		$DBP1, DEBUG

DEBUG

PURPOSE: Aids the user in debugging his relocatable assembly language programs.

METHOD:

The operator links DEBUG to a program at load-time. See the manual for your operating system.

COMMENTS:

The BCS DEBUG executes programs interpretively and allows the operator to set values in memory and registers, dump memory, set relocation bases, establish a breakpoint at an instruction or operand, and set up a trace.

The RTE/DOS DEBUG does not interpret programs; it places jump subroutine instructions in each breakpoint location and allows the program to execute normally until it reaches a breakpoint. The operator can set a relocation base, set instruction breakpoints, dump memory, and set values in memory or registers.

ENDIO

PURPOSE: Delays further program execution until all current input/output operations are completed.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	ENDIO	N/A
EXTERNAL REFERENCES:	.IOC.	
CALLING SEQUENCES:	JSB ENDIO DEF *+1 → returns when all I/O is completed.	

METHOD:

Executes a system status request.

ATTRIBUTES:

	ENTRY POINTS:
	ENDIO
Parameters:	None
Result:	None
Basic FORTRAN:	Callable
FORTRAN IV:	Callable
ALGOL:	Callable
Errors:	None

ERRØ

PURPOSE: Prints a 4 character error code on the list device (the BCS version is a dummy routine for compatability).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	ERRØ (U)	
EXTERNAL REFERENCES:	NONE	EXEC, .OPSY
CALLING SEQUENCES:	LDA *NN* } see below LDB *XX* } JSB ERRØ →	

METHOD:

NN is the routine identifier } pairs of ASCII characters.
XX is the error type }

Prints this on the list device: *name NN XX*

where *name* is the name of the program.

ATTRIBUTES:

	ENTRY POINTS:
	ERRØ
Parameters:	ASCII Characters
Result:	Printed
Basic FORTRAN:	Not Callable
FORTRAN IV:	Not Callable
ALGOL:	Not Callable
Errors:	None

EXEC

PURPOSE: Provides program termination for RTE/DOS compatable programs when run in BCS.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	EXEC	N/A, part of system
EXTERNAL REFERENCES:	.STOP	

CALLING SEQUENCES:	`JSB EXEC`
	`DEF *+2`
	`DEF RCODE`
	`→`
	`:`
	`RCODE DEC 6`	

METHOD: Calls .STOP.

ATTRIBUTES:

	ENTRY POINTS:
	EXEC
Parameters:	Integer
Result:	None
Basic FORTRAN:	Use END statement in main program.
FORTRAN IV:	Use END statement in main program
ALGOL:	Use END$
Errors:	None

GETAD

PURPOSE: Determines the true address of a parameter passed to a subroutine and places the address in ADRES.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	GETAD, (U) ADRES	
EXTERNAL REFERENCES:	NONE	
CALLING SEQUENCES:	JSB GETAD DEF SUB,I LDA ADRES see below	

METHOD:

```
     JSB SUB
     DEF X[,I]
      :
 SUB NOP
     JSB GETAD
     DEF SUB,I
     LDA ADRES
     →
```

ATTRIBUTES:

ENTRY POINTS:

	GETAD	ADRES
Parameters:	Integer Address	NA
Result:	Address	Integer
Basic FORTRAN:	Not Callable	Not Callable
FORTRAN IV:	Not Callable	Not Callable
ALGOL:	Not Callable	Not Callable
Errors:	None	None

NOTES:

1. May not be called by privileged or re-entrant routines; see .PCAD.

INDEX

PURPOSE: Returns the address (.INDA) or value (.INDR) of an ALGOL array.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.INDA (U) .INDR	
EXTERNAL REFERENCES:	.IOC.	EXEC
CALLING SEQUENCES:	JSB .INDA (or .INDR) DEF array table (see below) DEF- number of indices DEF subscript 1 ⋮ DEF subscript *N* →result in A or A & B	

METHOD: Array Table:

```
TABLE ABS number of indices (+ = real, - = integer)
      ABS size of 1st dimension
      ABS -lower bound of 1st dimension
      DEF array address
       :
      ABS size of last dimension
      ABS - lower bound of last dimension
```

ATTRIBUTES:

	ENTRY POINTS:	
	.INDA	.INDR
Parameters:	Integer	Integer
Result:	Address: A	Value: A or A & B
Basic FORTRAN:	Not Callable	Not Callable
FORTRAN IV:	Not Callable	Not Callable
ALGOL:	Not Callable	Not Callable
Errors:	See Note 1	See Note 1

NOTES: 1. If array not properly defined:

A = Address of Call
Prints INDEX? on teleprinter.

When RUN is pushed, routine returns with result = 0.

ISSW

PURPOSE: Sets the sign bit (15) of A-Register equal to bit *N* of the switch register.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		ISSW (U)
EXTERNAL REFERENCES:		NONE
CALLING SEQUENCES:		LDA *N* JSB ISSW → result in A

ATTRIBUTES:

	ENTRY POINTS:
	ISSW
Parameters:	Integer
Result:	Integer
Basic FORTRAN:	Function: ISSW (*N*)
FORTRAN IV:	Function: ISSW (*N*)
ALGOL:	Not callable directly; see ALGOL manual.
Errors:	None

LEADR

PURPOSE: Produces consecutive feed frames (octal zeroes) on punched tape to serve as leader.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	LEADR	N/A
EXTERNAL REFERENCES:	.IOC., .ENTR	
CALLING SEQUENCES:	JSB LEADR DEF * + 3 DEF *U* (see below) DEF *N* (see below →	

METHOD: *U* is the octal unit-reference number of the punched tape unit;

N is the decimal number of inches of leader to be punched.

ATTRIBUTES:	**ENTRY POINTS:**
	LEADR
Parameters:	Integer
Result:	None
Basic FORTRAN:	Callable
FORTRAN IV:	Callable
ALGOL:	Callable
Errors:	If *U* is not a paper tape device → computer halts (A = 0).

LINE

PURPOSE: Plots a line and/or symbols through the successive data points in arrays previously scaled by the SCALE routine.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	LINE (U)	
EXTERNAL REFERENCES:	SYMB, PLOT, numerous library routines	

CALLING SEQUENCES:

```
JSB LINE
DEF *+7
DEF X (real array scaled for the abscissa)
DEF Y (real array scaled for the ordinate)
DEF N (integer number of points to be plotted)
DEF K (repeat factor, same as in SCALE)
DEF J (integer control value)
DEF L (number of centered symbols to be plotted;
      see SYMB for table)
→
```

```
where J = 0, for a line plot only;
        = 1, for a symbol at every point; no line;
        = -1, for a line and a symbol at every point;
        = -2, for a line and a symbol at every second point;
        = -N, for a line and a symbol at every Nth point.
```

ATTRIBUTES:

	ENTRY POINTS:
	LINE
Parameters:	Mixed
Result:	N/A
Basic FORTRAN:	Callable as subroutine
FORTRAN IV:	Callable as subroutine
ALGOL:	Callable as CODE procedure
Errors:	None

NOTES:

1. Since the LINE routine requires the adjusted minimum and delta values produced by the SCALE routine, SCALE must be called before LINE for each graph.

2. Sample calls to LINE:

   CALL LINE (*X*, *Y*, 5Ø, 1Ø, Ø)

   (plots a line of 50 *XY* values)

   CALL LINE (*X*, *Y*, 5Ø, 1, -5, 3)

   (plots a line of 50 points with a "+" symbol at every fifth point)

MAGTP

PURPOSE: Performs utility functions on magnetic tape and other devices: checks status, performs rewind/standby, writes a gap, issues a clear request, and does blocked input/output.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	IEOF,IERR,IEOT,ISOT,LOCAL,IWRDS, RWSTB,GAP3,CLEAR,IUNIT,BFINP,BFOUT.	IEOF,IERR,IEOT,ISOT,LOCAL, IWRDS(N/A in RTE),RWSTB
EXTERNAL REFERENCES:	.ENTR, .IOC.	.ENTR, EXEC

ATTRIBUTES:

	ENTRY POINTS:	
	IEOF,IERR,IEOT,ISOT,LOCAL,IWRDS,IUNIT	RWSTB,GAP3,CLEAR,BFINP,BFOUT
Parameters:	Integer	Integer
Result:	Integer: A	N/A
Basic FORTRAN:	Callable as function	Callable as subroutine
FORTRAN IV:	Callable as function	Callable as subroutine
ALGOL:	Callable as integer procedure	Callable as subroutine
Errors:	Returns on illegal call	Returns on illegal call

CALLING SEQUENCES:

The calling sequence and purpose of each entry point is:

JSB IEOF DEF *+2 DEF unit →	Returns a negative value in A if an end-of-file was encountered during last tape operation on the logical unit specified.
JSB IERR DEF*+2 DEF unit →	Returns a negative value in A if a parity or timing error was not cleared after three read attempts during the last operation on the specified unit (cannot occur if EOF occurs).
JSB IEOT DEF *+2 DEF unit →	Returns a negative value in A if an end-of-tape was encountered during the last forward movement of the specified unit.
JSB ISOT DEF *+2 DEF unit →	Returns a negative value in A if the start-of-tape marker is under the tape head of the specified unit.
JSB LOCAL DEF *+2 DEF unit →	Returns a negative value in A if the specified unit is in local mode.
JSB IWRDS DEF *+2 DEF unit →	(Not available in RTE.) Returns the value of the transmission log of the last read/write operation on the specified unit. (In the formatter environment, this value is always a positive number of characters.)
JSB IUNIT DEF *+2 DEF unit →	(Not available in DOS/RTE.) Returns the status word (EQT word #2) of the specified logical unit. If the unit is busy, the word is negative. If the specified unit is Ø, the routine returns system status.

MAGTP

CALLING SEQUENCES:

```
JSB RWSTB          Rewinds the specified logical unit and sets it
DEF *+2            to LOCAL.
DEF unit
→

JSB GAP3           (Not available in DOS/RTE.)  Writes a gap on the
DEF *+2            specified logical unit.
DEF unit
→

JSB CLEAR          (Not available in DOS/RTE.)  Issues a clear
DEF *+2            request to the specified unit.
DEF unit
→

JSB BFINP          Performs buffered input from the specified unit
DEF *+4            to the specified buffer.  (Not available in
DEF unit           DOS/RTE.) Unit is positive for binary, negative
DEF buffer         for ASCII.  Buffer length is positive for words,
  address          negative for characters.
DEF buffer
  length
→


JSB BFOUT          Performs buffered output from the specified
DEF *+4            buffer to the specified unit.  (Not available
DEF unit           in DOS/RTE.)  Unit is positive for binary,
DEF buffer         negative for ASCII.  Buffer length is positive
  address          for words, negative for characters.
DEF buffer
  length
→
```

The previous two calls should be followed by IUNIT tests for completion of operation in systems which are not using buffered .IOC..

MEMRY

PURPOSE: Performs memory allocation for buffered .IOC.; user program requests buffers to be allocated and released from the memory available after program loading.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.ALC., .RTN., .CLR.	N/A
EXTERNAL REFERENCES:	.MEM., .IOC.	

CALLING SEQUENCES:

To allocate a buffer:

```
JSB .ALC
DEC number of words
→
```

Upon return, if the buffer request is filled:

A=address of first word of buffer

B=number of words allocated

If the buffer is not allocated because sufficient memory is temporarily unavailable:

A=0

B=maximum buffer length that can be allocated without releasing some previous buffer space.

If the buffer is not allocated because sufficient memory is not available even when all buffers are released:

A=-1

B=maximum buffer length that can be allocated if all other buffers are released.

To determine the largest possible buffer that can be allocated if all other buffers are released:

```
JSB .ALC.
DEC 32767
→
```

The results are returned in the registers:

A=-1

B=maximum buffer length

To release a specified area of buffer:

```
JSB .RTN.
DEF address of first word of buffer to be released
DEC number of words to be released
→
```

To release all storage allocated:

```
CLA
STA .CLR.
```

ATTRIBUTES:

ENTRY POINTS:

	.ALC.	.RTN.	.CLR.
Parameters:	Integer	Integers	N/A
Result:	Values: A&B	N/A	N/A
Basic FORTRAN:	Not callable	Not callable	Not callable
FORTRAN IV:	Not callable	Not callable	Not callable
ALGOL:	Not callable	Not callable	Not callable
Errors:	None	None	None

NUMB

PURPOSE: Plots a floating-point number, with or without the decimal point, at a specified height, location, and angle.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		NUMB (U)
EXTERNAL REFERENCES:		SYMB and numerous library subroutines

CALLING SEQUENCES:

```
JSB NUMB
DEF *+7
DEF floating-point x coordinate of
    lower left corner where number
    is to be plotted
DEF floating-point y coordinate
DEF height, in floating-point inches,
    of number
DEF the floating-point number to be plotted
DEF the angle, in floating-point degrees,
    at which the number is to be plotted
DEF N
→
```

where *N* = 0, for print the decimal point of an integer;

= 1, for suppress decimal point of an integer.

ATTRIBUTES:

	ENTRY POINTS:
	NUMB
Parameters:	Mixed
Result:	N/A
Basic FORTRAN:	Callable as subroutine
FORTRAN IV:	Callable as subroutine
ALGOL:	Callable as CODE procedure
Errors:	None

NOTES:

1. Sample call to NUMB:

   Plot three numbers .1 inches high, with decimal point suppressed, at 8.79 inches above 0,0 and at 5.32, 6.3 and 7.16 inches to the right of 0,0.

```
CALL NUMB (5.32, 8.79, Ø.1Ø, FLOAT (I), Ø.Ø, -1)
CALL NUMB (6.3Ø, 8.79, Ø.1Ø, FLOAT (J), Ø.Ø, -1)
CALL NUMB (7.16, 8.79, Ø.1Ø, FLOAT (K), Ø.Ø, -1)
```

OVF

PURPOSE: Returns value of overflow bit in bit 15 of the A-Register and clears the overflow bit.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	OVF (U)	
EXTERNAL REFERENCES:	None	
CALLING SEQUENCES:	JSB OVF → result in A	

METHOD: If overflow bit is set (on), the A-Register is set negative; if the overflow bit is off, the A-Register is set positive.

ATTRIBUTES:

	ENTRY POINTS:
	OVF
Parameters:	None
Result:	Integer: A
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

PAUSE

PURPOSE: Prints the following message on the teleprinter: *name*: PAUSE *xxxx* or *name*: STOP *xxxx* where *name* is the calling program name and *xxxx* is the specified integer *I*.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	N/A	.PAUS, .STOP (U)
EXTERNAL REFERENCES:		EXEC
CALLING SEQUENCES:		LDA *I* JSB .PAUS (or .STOP) → See Note 1.

ATTRIBUTES:

	ENTRY POINTS:	
	.PAUS	.STOP
Parameters:	Integer	Integer
Result:	None	None
Basic FORTRAN:	Not callable	Not callable
FORTRAN IV:	Not callable	Not callable
ALGOL:	Not callable	Not callable
Errors:	None	None

NOTES:

1. When .PAUS is used, the program may be restarted using GO (RTE) or :GO (DOS).

PLOT

PURPOSE: Moves the pen of a plotter to any location on the graph with the pen up or down, establishes new origin points, sets the plotter logical unit (RTE/DOS), determines current position, varies the plot factor, and allows external buffers to be established.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	PLOT,WHERE,FACT,PLOTB	PLOT,PLTLU,WHERE,FACT,PLOTB (U)
EXTERNAL REFERENCES:	.ENTR,.IOC.,IFIX,FLOAT	.ENTR,EXEC,IFIX,FLOAT

ATTRIBUTES:

	ENTRY POINTS:				
	PLOT	WHERE	FACT	PLOTB	PLTLU
Parameters:	Mixed	None	Real	Mixed	Integer
Result:	N/A	Real	N/A	N/A	N/A
Basic FORTRAN:	Callable	Callable	Callable	Callable	Callable
FORTRAN IV:	Callable	Callable	Callable	Callable	Callable
ALGOL:	Callable	Callable	Callable	Callable	Callable
Errors:	None	None	None	None	None

PLOT

CALLING SEQUENCES:

PLOT Calling Sequences

The PLOT routine is called by a FORTRAN CALL statement or an Assembly Language calling sequence.

FORTRAN: CALL PLOT *X*, *Y*, *IC*

X and *Y* = the coordinates to which the pen is to be moved. All X and Y coordinates must be expressed as floating-point inches in deflection from the origin.

IC = an integer constant or variable name set equal to one of the following:

-2 = Move with the pen down; consider the point where the pen stops (*X*,*Y*) as the new origin.

-3 = Move with the pen up; consider the point where the pen stops (*X*,*Y*) as the new origin.

+2 = Move with the pen down; origin unchanged.

+3 = Move with the pen up; origin unchanged.

ASSEMBLY LANGUAGE:

```
JSB  PLOT
DEF  *+4
DEF   X  (Defines address of x coordinate)
DEF   Y  (Defines address of y coordinate)
DEF  IC  (Defines address of pen command)
→
```

PLOT Coding Requirements

Before a call is made to PLOT in DOS/RTE, an initial call to the PLTLU entry point must be made to insure that the logical unit number of the referenced plotter is placed in the I/O request.

A single FORTRAN statement moves the pen to the desired location.

All X and Y coordinates must be expressed as floating-point inches in deflection from the origin.

PLOT

PLOT Example

To plot a rectangle 8.5" by 11" starting at the origin, four calls to the PLOT routine must be made (assuming that the pen starts at the origin).

```
      CALL  PLTLU  (ILU)
```

Initial call to PLTLU for plotter's logical unit number

```
              :
              :
      CALL  PLOT  (11.,Ø.,+2)
```

Moves the pen from X,Y = (0,0) to X,Y = (11,0)

```
      CALL  PLOT  (11.,8.5,+2)
```

Moves the pen from X,Y = (11,0) to X,Y = (11,8.5)

```
      CALL  PLOT  (Ø.,8.5,+2)
```

Moves the pen from X,Y = (11,8.5) to X,Y = (0,8.5)

```
      CALL  PLOT  (Ø.,Ø.,+2)
```

Moves the pen from X,Y = (0,8.5) to the origin

PLOT Associated Functions

The PLOT routine can perform additional functions when calls are made to the following entry points: PLTLU, WHERE, FACT and PLOTB.

PLTLU ENTRY POINT (RTE/DOS ONLY)

A call to the PLTLU entry point allows the user to designate the logical unit number for the plotter. The logical unit number must be designated before a call to the PLOT routine. Otherwise, the user program will be terminated when an I/O request containing a logical unit value of zero is made by the PLOT routine. The logical unit number may be varied by the user program to direct output to more than one plotter. (In BCS, the PLOT routine examines the equipment table to find the first plotter; all output is then made to that plotter.)

PLTLU Calling Sequences

The PLTLU function can be called by a FORTRAN CALL statement or an Assembly Language calling sequence.

FORTRAN: CALL PLTLU (*ILU*)

ILU = An integer value representing the logical unit number. Refer to the Real-Time Software and DOS reference manuals for discussion of logical unit values.

ASSEMBLY LANGUAGE:

```
      JSB   PLTLU
      DEF   *+2
      DEF   ILU    (Defines address of logical unit value)
      →
```

WHERE ENTRY POINT

A call to the WHERE entry point allows the user to determine the current plotter pen position.

WHERE Calling Sequences

The WHERE function can be called by a FORTRAN CALL statement or an Assembly Language calling sequence.

FORTRAN: CALL WHERE (*X*,*Y*)

X and *Y* = The addresses in which the X and Y coordinates of the current pen position are stored (in floating-point format) by the WHERE function.

ASSEMBLY LANGUAGE:

```
JSB   WHERE
DEF   *+3
DEF   X    {Define the locations where the current }
DEF   Y    {pen positions are to be stored.        }
→
```

FACT ENTRY POINT

A call to the FACT entry point allows the user to vary the plot scale factor.

FACT Calling Sequences

The FACT function can be called by a FORTRAN CALL statement or an Assembly Language calling sequence.

FORTRAN: CALL FACT (*N*)

N = The floating point number used to establish the new scaling factor. Note that *N* is multiplied by 100.00 for the 100 plotter increments/inch when the new scaling factor is established. The plot factor is initialized at 1.

ASSEMBLY LANGUAGE:

```
JSB   FACT
DEF   *+2
DEF   FCT   (Defines the address of the factor modifier)
→
```

PLOTB ENTRY POINT

A call to the PLOTB entry point allows the user to specify a "plot work" buffer external to the PLOT routine. This entry point is initialized using an internal ten-word buffer.

PLOTB Calling Sequences

The PLOTB function can be called by a FORTRAN CALL statement or an Assembly Language calling sequence.

```
FORTRAN:      CALL   PLOTB   (A,L)
```

A = The starting address of the external buffer. An address of zero specifies the ten-word internal buffer.

L = A positive decimal integer specifying the buffer length in words.

ASSEMBLY LANGUAGE:

```
JSB   PLOTB
DEF   *+3
DEF   A    (Defines the buffer starting address)
DEF   L    (Defines the buffer length)
→
```

PTAPE

PURPOSE: Positions a magnetic tape unit by spacing forward or backward a number of files and/or records.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	PTAPE (U)	
EXTERNAL REFERENCES:	.IOC., .ENTR	EXEC, .ENTR

CALLING SEQUENCES:

```
JSB PTAPE
DEF *+4
DEF logical unit
DEF file count      } see below
DEF record count
→
```

File count: positive for forward, negative for backward.

For example:

Ø means make no file movements.

-1 means backspace to the beginning of the current file.

1 means forward space to beginning of the next file.

-2 means backspace to the beginning of the previous file.

Record count: positive for forward, negative for backward.

The file count is executed first, then the record count.

EOF marks count as a record.

For example:

Ø,-1 means move back one record.

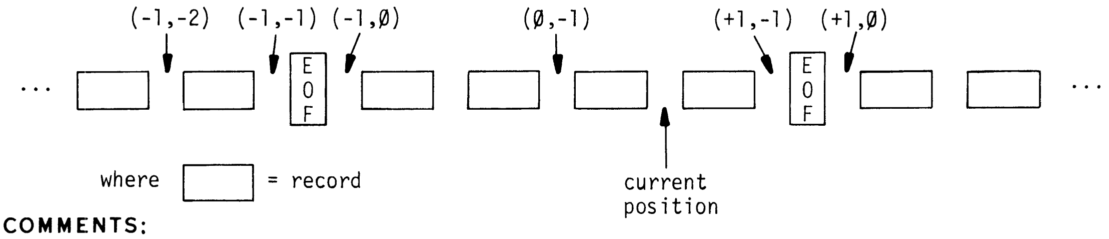
-1,0 means backspace to the first record of the current file.

See Note 1.

ATTRIBUTES:

	ENTRY POINTS:
	PTAPE
Parameters:	Integers
Result:	None
Basic FORTRAN:	Callable
FORTRAN IV:	Callable
ALGOL:	Callable
Errors:	None

NOTES: 1. The diagram below shows how the position of the magnetic tape would change with several example file/record counts.



COMMENTS:

1. After using PTAPE, always check status with MAGTP.

RMPAR

PURPOSE: Retrieves parameters passed by operator when a suspended program is resumed.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	N/A	RMPAR (U)
EXTERNAL REFERENCES:		None
CALLING SEQUENCES:		Suspend call JSB RMPAR DEF *+2 DEF ARRAY → ARRAY BSS 5

ATTRIBUTES:

	ENTRY POINTS:
	RMPAR
Parameters:	Integer
Result:	Integer
Basic FORTRAN:	Callable
FORTRAN IV:	Callable
ALGOL:	Callable
Errors:	None

SCALE

PURPOSE: Scales an array of floating-point numbers to fit a specified size graph; the values generated are used by the LINE and AXIS routines.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	SCALE (U)	
EXTERNAL REFERENCES:	Numerous library subroutines	
CALLING SEQUENCES:	(Separate calls required for X and Y axes) JSB SCALE DEF *+5 DEF array containing real values DEF length of axis in floating-point inches DEF integer number of points to be plotted DEF *K* (integer which specifies the points to be scaled: K=1, every point; K=2, every other point; K=3, every third point; etc.)	

ATTRIBUTES:

	ENTRY POINTS:
	SCALE
Parameters:	Mixed
Result:	N/A
Basic FORTRAN:	Callable as subroutine
FORTRAN IV:	Callable as subroutine
ALGOL:	Callable as CODE procedure
Errors:	None

NOTES:

1. The adjusted minimum value is a number less than or equal to the minimum data value. The adjusted delta value is the result of subtracting the minimum data value from the maximum data value, divided by the length of the axis and adjusted to provide one-inch increments that will cover the data. The adjusted scale values are used by the LINE and AXIS routines.

2. The adjusted values are stored following the array. The minimum value for *Y* is stored in *Y(NP*K+1)*, where *NP* is the number of points to be plotted; the delta value is stored in *Y(NP*K+2)*. Therefore, the array must be dimensioned (*K+2*) locations larger than (*NP*K*), which is the number of locations necessary for data points. Normally, *K*=1, so an array ZIP of ten data points would be dimensioned as ZIP(12).

3. Sample use of SCALE: Scale every point in a 50-point array, fitting X values on a 6.5-inch X axis and Y values on a 10-inch Y axis:

```
DIMENSION X(52),Y(52)
    :
CALL SCALE (X,6.5,50,1)
CALL SCALE (Y,1Ø.Ø,5Ø,1)
```

SREAD

PURPOSE: Reads a record from a device specified by a logical unit number (used only by system programs).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	N/A	%READ, %JFIL, %RDSC, (U)
EXTERNAL REFERENCES:		.OPSY, EXEC

SYMB

PURPOSE: Plots a string of characters at a specified location on the plotter; number, height, and angle of characters can be varied.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	SYMB (U)	
EXTERNAL REFERENCES:	PLOT,SIN,COS,.ENTR	PLOT,SIN,COS,.ENTR,ERRØ

CALLING SEQUENCES:

```
JSB SYMB
DEF *+7
DEF X  (defines address of floating-point X coordinate)
DEF Y  (defines floating-point Y coordinate)
DEF size  (defines height in floating-point inches)
DEF IASCI (defines address of ASCII array or special symbol number)
DEF THETA (defines angle in degrees)
DEF N     (defines number or type of characters)
```

If *N*>0, N = number of ASCII characters to be plotted from array *IASCI*.
N=0, plot only lower character from *IASCI*.
N<0, plot special symbol #*N* (*N*=-1, move with pen up, *N*<-1, pen down)

ATTRIBUTES:

	ENTRY POINTS:
	SYMB
Parameters:	Mixed
Result:	N/A
Basic FORTRAN:	Callable as subroutine
FORTRAN IV:	Callable as subroutine
ALGOL:	Callable as CODE procedure
Errors:	None

NOTES:

1. Example:

   Plot a line of 39 symbols from IBUFF along the X axis, starting 1 inch to the right and 9 inches above (0,0), with characters 0.14 inches high.

   ```
   CALL SYMB(1.Ø,Ø.Ø,Ø.14,IBUFF,Ø.Ø,39)
   ```

   Plot a right-direction arrow (symbol #20) 4 inches above and to the right of (0,0). The desired height is 0.5 inches.

   ```
   CALL SYMB(4.Ø,4.Ø,Ø.5,NUMB,Ø.Ø,-1)
   ```

   where NUMB contains a decimal 20.

COMMENTS:

1. See Table SYMB-1 for a list of characters that the SYMB routine can plot.

   Any program that calls SYMB in the RTE/DOS environment must call PLOTLU first to establish the plotter logical unit. (See PLOT.)

TABLE 2-1. SYMBOL/CHARACTER TABLE

SYMBOLS

Centered Symbols[1]

0	☐	5	◇	10	⌑
1	⊘	6	↑	11	*
2	△	7	⨯	12	⧖
3	+	8	Z	13	'
4	X	9	Y	14	✡

Uncentered Symbols[2]

15	_	20	→
16	!	21	Σ
17	↓	22	≥
18	≤	23	△
19	=	24	≠
25	±		

ASCII CHARACTERS[2]

126	@	39	M	52	Z	65	'	78	4
27	A	40	N	53	[	66	(	79	5
28	B	41	O	54	\	67	)	80	6
29	C	42	P	55	]	68	*	81	7
30	D	43	Q	56	↑	69		82	8
31	E	44	R	57	←	70	,	83	9
32	F	45	S	58	pen up	71	-	84	:
33	G	46	T	59	!	72	.	85	;
34	H	47	U	60	"	73	/	86	<
35	I	48	V	61	#	74	Ø	87	=
36	J	49	W	62	$	75	1	88	>
37	K	50	X	63	%	76	2	89	?
38	L	51	Y	64	&	77	3		

[1]Centered symbols are centered with respect to their reference point; they are useful in point plotting, with or without an accompanying line plot.

[2]Uncentered symbols are plotted such that the lower left corner of the symbol starts from the specified reference point; these symbols are useful mainly in captions and notes on the graph. ASCII characters are likewise plotted uncentered.

#COS

PURPOSE: Entry to CCOS with no error return.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	#COS (U)	
EXTERNAL REFERENCES:	.ENTR, CCOS	ERRØ, .ENTR, CCOS

CALLING SEQUENCES:

```
JSB #COS
DEF *+3
DEF y
DEF x
→
```

ATTRIBUTES:

	ENTRY POINTS:
	#COS
Parameters:	Complex
Result:	Complex
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

#EXP

PURPOSE: Entry to CEXP with no error return.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	#EXP (U)	
EXTERNAL REFERENCES:	.ENTR, CEXP	ERRØ, ENTR, CEXP

CALLING SEQUENCES:

```
JSB #EXP
DEF *+3
DEF y
DEF x
→
```

ATTRIBUTES:

	ENTRY POINTS:
	#EXP
Parameters:	Complex
Result:	Complex
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

#LOG

PURPOSE: Entry to CLOG with no error return.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	#LOG (U)	
EXTERNAL REFERENCES:	.ENTR, CLOG	ERRØ, .ENTR, CLOG
CALLING SEQUENCES:	JSB #LOG DEF *+3 DEF *Y* DEF *X* →	

ATTRIBUTES:

	ENTRY POINTS:
	#LOG
Parameters:	Complex
Result:	Complex
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

#SIN

PURPOSE: Entry to CSIN with no error routine.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	#SIN (U)	
EXTERNAL REFERENCES:	.ENTR, CSIN	ERRØ, .ENTR, CSIN
CALLING SEQUENCES:	JSB #SIN DEF *+3 DEF *Y* DEF *X* →	

ATTRIBUTES:

	ENTRY POINTS:
	#SIN
Parameters:	Complex
Result:	Complex
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

$EXP

PURPOSE: Entry to DEXP with no alternate error routine.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	$EXP (U)	$EXP (U)
EXTERNAL REFERENCES:	.ENTR, DEXP	ERRØ, .ENTR, DEXP
CALLING SEQUENCES:	JEB $EXP DEF *+3 DEF *y* DEF *x* →	JEB $EXP DEF *+3 DEF *y* DEF *x* →

ATTRIBUTES:

	ENTRY POINTS:
	$EXP
Parameters:	Double real
Result:	Double real
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

$LOG

PURPOSE: Entry to DLOG with no error return.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	$LOG (U)	
EXTERNAL REFERENCES:	.ENTR, DLOG	ERRØ, .ENTR, DLOG
CALLING SEQUENCES:	JSB %EXP DEF *+3 DEF *Y* DEF *X* →	

ATTRIBUTES:

	ENTRY POINTS:
	$LOG
Parameters:	Double real
Result:	Double real
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

$LOGT

PURPOSE: Entry to DLOGT with no error return.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	$LOGT (U)	
EXTERNAL REFERENCES:	.ENTR, DLOGT	DLOGT, .ENTR, ERRØ
CALLING SEQUENCES:	JSB $LOGT DEF *+3 DEF *Y* DEF *X* →	

ATTRIBUTES:

	ENTRY POINTS:
	$LOGT
Parameters:	Double real
Result:	Double real
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

$SQRT

PURPOSE: Entry to DSQRT with no error return.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	$SQRT (U)	
EXTERNAL REFERENCES:	DSQRT, .ENTR	DSQRT, ERRØ, .ENTR
CALLING SEQUENCES:	JSB $SQRT DEF *+3 DEF *Y* DEF *X* →	

ATTRIBUTES:

	ENTRY POINTS:
	$SQRT
Parameters:	Double real
Result:	Double real
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

%ABS

PURPOSE: Call-by-name entry to IABS(*I*)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		%ABS (U)
EXTERNAL REFERENCES:		IABS
CALLING SEQUENCES:		JSB %ABS DEF *+2 DEF *I* → result in A

ATTRIBUTES:

	ENTRY POINTS:
	%ABS
Parameters:	Integer: A
Result:	Integer: A
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

%AN

PURPOSE: Call-by-name entry to TAN(x).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		%AN (U)
EXTERNAL REFERENCES:		TAN
CALLING SEQUENCES:		JSB %AN DEF *+2 DEF x → result in A&B

ATTRIBUTES:

	ENTRY POINTS:
	%AN
Parameters:	Real
Result:	Real: A&B
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

%AND

PURPOSE: Call-by-name entry to IAND(*I*,*J*).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		%AND (U)
EXTERNAL REFERENCES:		IAND
CALLING SEQUENCES:		JSB %AND DEF *+3 DEF *I* DEF *J* → result in A

ATTRIBUTES:

	ENTRY POINTS:
	%AND
Parameters:	Integer
Result:	Integer
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

%ANH

PURPOSE: Call-by-name entry to TANH(*x*).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		%ANH (U)
EXTERNAL REFERENCES:		TANH
CALLING SEQUENCES:		JSB %ANH DEF *+2 DEF *x* → result in A&B

ATTRIBUTES:

	ENTRY POINTS:
	%ANH
Parameters:	Real
Result:	Real: A&B
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

%BS

PURPOSE: Call-by-name entry to ABS(*x*).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		%BS (U)
EXTERNAL REFERENCES:		ABS
CALLING SEQUENCES:		JSB %BS DEF *+2 DEF *x* → result in A&B

ATTRIBUTES:

	ENTRY POINTS:
	%BS
Parameters:	Real
Result:	Real: A&B
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

%FIX

PURPOSE: Call-by-name entry to IFIX(*x*).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		%FIX (U)
EXTERNAL REFERENCES:		IFIX
CALLING SEQUENCES:		JSB %FIX DEF *+2 DEF *x* → result in A

ATTRIBUTES:

	ENTRY POINTS:
	%FIX
Parameters:	Real
Result:	Integer: A
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

%IGN

PURPOSE: Call-by-name entry to SIGN (*x*, *z*)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		%IGN (U)
EXTERNAL REFERENCES:		SIGN
CALLING SEQUENCES:		JSB %IGN DEF *+3 DEF *x* DEF *z* →result in A & B

ATTRIBUTES:

	ENTRY POINTS:
	%IGN
Parameters:	Real or integer and real
Result:	Real
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

%IN

PURPOSE: Call-by-name entry to SIN (*x*).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		%IN (U)
EXTERNAL REFERENCES:		SIN
CALLING SEQUENCES:		JSB %IN DEF *+2 DEF *x* →result in A & B

ATTRIBUTES:

	ENTRY POINTS:
	%IN
Parameters:	Real
Result:	Real: A & B
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

%INT

PURPOSE: Call-by-name entry to AINT (*x*).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	%INT (U)	
EXTERNAL REFERENCES:	AINT	
CALLING SEQUENCES:	JSB %INT DEF *+2 DEF *x* →result in A & B	

ATTRIBUTES:

	ENTRY POINTS:
	%INT
Parameters:	Real
Result:	Real
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

%LOAT

PURPOSE: Call-by-name entry to FLOAT (*I*)

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		%LOAT (U)
EXTERNAL REFERENCES:		FLOAT
CALLING SEQUENCES:		JSB %LOAT DEF *+2 DEF *I* → result in A&B

ATTRIBUTES:

	ENTRY POINTS:
	%LOAT
Parameters:	Integer
Result:	Real: A&B
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

%LOG

PURPOSE: Call-by-name entry to ALOG (*x*).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		%LOG (U)
EXTERNAL REFERENCES:		ALOG
CALLING SEQUENCES:		JSB %LOG DEF *+2 DEF *x* → result in A&B

ATTRIBUTES:

	ENTRY POINTS:
	%LOG
Parameters:	Real
Result:	Real: A&B
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

% LOGT

PURPOSE: Call-by-name entry to ALOGT (*x*).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		%LOGT (U)
EXTERNAL REFERENCES:		ALOGT
CALLING SEQUENCES:		JSB %LOGT DEF *+2 DEF *x* → result in A&B

ATTRIBUTES:

	ENTRY POINTS:
	%LOGT
Parameters:	Real
Result:	Real
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

%NT

PURPOSE: Call-by-name entry to INT (*x*).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		%NT (U)
EXTERNAL REFERENCES:		INT
CALLING SEQUENCES:		JSB %NT DEF *+2 DEF *x* (real) → result in A

ATTRIBUTES:

	ENTRY POINTS:
	%NT
Parameters:	Real
Result:	Integer
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

%OR

PURPOSE: Call-by-name entry to IOR (*I*, *J*).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		%OR (U)
EXTERNAL REFERENCES:		IOR
CALLING SEQUENCES:		JSB %OR DEF *+3 DEF *I* DEF *J* → result in A

ATTRIBUTES:

	ENTRY POINTS:
	%OR
Parameters:	Integer
Result:	Integer: A
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

%OS

PURPOSE: Call-by-name entry to COS (x).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		%OS (U)
EXTERNAL REFERENCES:		COS
CALLING SEQUENCES:		JSB %OS DEF *+2 DEF *x* → result in A&B

ATTRIBUTES:

	ENTRY POINTS:
	%OS
Parameters:	Real
Result:	Real: A&B
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

%OT

PURPOSE: Standard call-by-name subroutine for NOT function.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		%OT (U)
EXTERNAL REFERENCES:		None
CALLING SEQUENCES:		JSB %OT DEF *+2 DEF *I* → result in A

METHOD:

Executes complement of *I*.

ATTRIBUTES:

	ENTRY POINTS:
	%OT
Parameters:	Integer
Result:	Integer: A
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

%QRT

PURPOSE: Call-by-name entry to SQRT (*x*).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		%QRT (U)
EXTERNAL REFERENCES:		SQRT
CALLING SEQUENCES:		JSB %QRT DEF *+2 DEF *x* → result in A&B

ATTRIBUTES:

	ENTRY POINTS:
	%QRT
Parameters:	Real
Result:	Real: A&B
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

% SIGN

PURPOSE: Call-by-name entry to ISIGN (*I*, *Z*).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	%SIGN (U)	
EXTERNAL REFERENCES:	ISIGN	
CALLING SEQUENCES:	JSB %SIGN DEF *+3 DEF *I* DEF *Z* → result in A	

ATTRIBUTES:

	ENTRY POINTS:
	%SIGN
Parameters:	Real (or integer) & integer
Result:	Integer: A
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

%SSW

PURPOSE: Call-by-name entry to ISSW (*N*).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		%SSW (U)
EXTERNAL REFERENCES:		ISSW
CALLING SEQUENCES:		JSB %SSW DEF *+2 DEF *N* (integer) → result in A

ATTRIBUTES:

	ENTRY POINTS:
	%SSW
Parameters:	Integer
Result:	Integer: A
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

%TAN

PURPOSE: Call-by-name entry to ATAN (x).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		%TAN (U)
EXTERNAL REFERENCES:		ATAN
CALLING SEQUENCES:		JSB %TAN DEF *+2 DEF x → result in A&B

ATTRIBUTES:

	ENTRY POINTS:
	%TAN
Parameters:	Real
Result:	Real: A&B
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

%WRIS

PURPOSE: Writes a disc source file (used only by system programs).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	N/A	%WRIS, %WRIN, %WEOF, (U)
EXTERNAL REFERENCES:		EXEC, .OPSY

COMMENTS:

1. This routine can only be called in the RTE System.

%WRIT

PURPOSE: Writes a load-and-go file on disc (used only by system programs).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	N/A	%WRIT, %WRIF, (U)
EXTERNAL REFERENCES:		.OPSY, EXEC

%XP

PURPOSE: Call-by-name entry to EXP (*x*).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		%XP (U)
EXTERNAL REFERENCES:		EXP
CALLING SEQUENCES:		JSB %XP DEF *+2 DEF *x* → result in A&B

ATTRIBUTES:

	ENTRY POINTS:
	%XP
Parameters:	Real
Result:	Real: A&B
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

.ENTR

PURPOSE: Transfers the true addresses of parameters from a calling sequence into a subroutine; adjusts return address to the true return point.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.ENTR	.ENTR, .ENTP (P)
EXTERNAL REFERENCES:	None	

CALLING SEQUENCES:

For all BCS subroutines, all DOS/RTE Utility routines:

```
PARAM BSS N      (N = maximum number of parameters)
  SUB NOP        (entry point to subroutine)
      JSB .ENTR
      DEF PARAM
```

For all privileged routines:

```
PARAM BSS N      (N = maximum number of parameters)
  SUB NOP        (entry point)
      JSB $LIBR
      NOP
      JSB .ENTP
      DEF PARAM
```

For all re-entrant routines:

```
  TDB NOP        (re-entrant processing table)
      DEC Q+N+3  (size of table)
      NOP
VARBL BSS Q      (subroutine variables)
PARAM BSS N      (parameter addresses)
  SUB NOP        (entry point)
      JSB $LIBR
      DEF TDB
      JSB .ENTP
      DEF PARAM
      STA TDB+2  (sets return address)
```

ATTRIBUTES:

	ENTRY POINTS:	
	.ENTR	.ENTP
Parameters:	Address	Address
Result:	Address	Address
Basic FORTRAN:	Not callable	Not callable
FORTRAN IV:	Not callable	Not callable
ALGOL:	Not callable	Not callable
Errors:	None	None

NOTES:

1. The true parameter address is determined by eliminating all indirect references.
2. .ENTR assumes the subroutine call is of the form:

```
JSB SUB
DEF *+n+1
DEF P1
 :
DEF Pn
→
```

.ERRR

PURPOSE: Writes a specified ASCII error code on the list device.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.ERRR	N/A
EXTERNAL REFERENCES:	.IOC.	
CALLING SEQUENCES:	JSB .ERRR ASC 1, *xx* ASC 1, *yy* *xx* and *yy* are error codes.	

ATTRIBUTES:

	ENTRY POINTS:
	.ERRR
Parameters:	ASCII characters
Result:	None
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

.GOTO

PURPOSE: Transfers control to the location indicated by a FORTRAN computed GO TO statement: GO TO (K_1, K_2, ... K_N) J

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.GOTO (U)	
EXTERNAL REFERENCES:	None	
CALLING SEQUENCES:	JSB .GOTO DEF *+N+1 DEF J DEF K_1 : DEF K_N →	

ATTRIBUTES:

	ENTRY POINTS:
	.GOTO
Parameters:	Addresses
Result:	None
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

.MAP.

PURPOSE: Returns actual address of a particular element of a two-dimensional FORTRAN array.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		.MAP. (U)
EXTERNAL REFERENCES:		None
CALLING SEQUENCES:		JSB .MAP. DEF array DEF first subscript DEF second subscript OCT first dimension, as below → result in A

METHOD:

Length of first dimension is actual for a real array, two's complement for an integer array.

ATTRIBUTES:

	ENTRY POINTS:
	.MAP.
Parameters:	Integer
Result:	Integer
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

.OPSY

PURPOSE: Determines, for disc-based systems, which operating system (RTE, DOS, DOS-M) is in control.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	N/A	.OPSY (P)
EXTERNAL REFERENCES:		None

CALLING SEQUENCES:		JSB .OPSY
		→ result in A
		A = 0: DOS
		A = 1: DOS-M
		A = -2: RTE

ATTRIBUTES:

	ENTRY POINTS:
	.OPSY
Parameters:	None
Result:	Integer
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

.PAUS

PURPOSE: Prints PAUSE on the teleprinter and halts the computer with a specified integer (*I*) in the A-Register. Returns to calling program when restarted.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.PAUS .	N/A, see PAUSE
EXTERNAL REFERENCES:	.IOC.	
CALLING SEQUENCES:	LDA *I* JSB .PAUS → return when RUN is pushed.	

ATTRIBUTES:

	ENTRY POINTS:
	.PAUS
Parameters:	Integer
Result:	None
Basic FORTRAN:	Not callable (Note 1)
FORTRAN IV:	Not callable (note 1)
ALGOL:	Not callable (Note 1)
Errors:	None

NOTES: 1. In FORTRAN and ALGOL use PAUSE statement.

.PCAD

PURPOSE: Return the true address of a parameter passed to a subroutine.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	N/A	.PCAD (P)
EXTERNAL REFERENCES:		None
CALLING SEQUENCES:		JSB .PCAD DEF SUB, I → result in A (See below for context)

METHOD:

```
        JSB SUB        (call to subroutine; indirect bit is optional
        DEF X[,I]       on parameter)
          :
    SUB NOP            (entry point to subroutine)
          :
        JSB .PCAD
        DEF SUB, I
        → address of X in A
```

ATTRIBUTES:

	ENTRY POINTS:
	.PCAD
Parameters:	Indirect Address
Result:	Direct Address: A
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

NOTES:

1. .PCAD has the same purpose as GETAD.
2. .PCAD is used by re-entrant or privileged by subroutines because they cannot use GETAD.

.PRAM

PURPOSE: Processes parameter values and/or addresses passed to Assembly language subroutines by ALGOL programs.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		.PRAM (U)
EXTERNAL REFERENCES:		None
CALLING SEQUENCES:		JSB .PRAM <code words> <parameters> See the ALGOL manual (HP 02116-9072).

ATTRIBUTES:

	ENTRY POINTS:
	.PRAM
Parameters:	Integer
Result:	Integer & Real
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

COMMENTS: Used in Assembly language subroutines to retrieve parameters from calling sequence inside the ALGOL calling program.

.STOP

PURPOSE: Prints STOP on the teleprinter and halts the computer with a specified integer (*I*) in the B-Register.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.STOP	N/A, see PAUSE
EXTERNAL REFERENCES:	.IOC.	
CALLING SEQUENCES:	LDA *I* JSB .STOP (no return)	

METHOD:

Returns to entry point HALT in the BCS loader. In stand-alone mode the HALT 77B is irrecoverable. In MTS mode control returns to .IPL..

ATTRIBUTES:

	ENTRY POINTS:
	.STOP
Parameters:	Integer
Result:	None
Basic FORTRAN:	Not callable (Note 1)
FORTRAN IV:	Not callable (Note 1)
ALGOL:	Not callable (Note 1)
Errors:	None

NOTES: 1. In FORTRAN and ALGOL use the STOP statement.

.SWCH

PURPOSE: Switches execution control to the *I*th of a sequence of *N* labels (implements ALGOL switch statement).

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.SWCH (U)	
EXTERNAL REFERENCES:	None	

CALLING SEQUENCES:

```
   LDA I
   JSB S
   → return if I is out of range
   :
 S NOP
   JSB .SWCH
   ABS N (see below)
   DEF Label 1
   DEF Label 2
   :
   DEF Label N
```

N is the number of labels.
If *I* is out of range, .SWCH returns.

ATTRIBUTES:

	ENTRY POINTS:
	.SWCH
Parameters:	Addresses
Result:	N/A
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	If *I* is out of range, returns.

.TAPE

PURPOSE: Performs magnetic tape rewind, backspace or end-of-file operations on a specified logical unit.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	.TAPE (U)	
EXTERNAL REFERENCES:	.IOC.	EXEC
CALLING SEQUENCES:	LDA *constant* JSB .TAPE constant = see below	

METHOD: *Constant:*

$30xyy_8$
x = 4 for REWIND
= 2 for BACKSPACE
= 1 for END FILE
yy = logical unit number

ATTRIBUTES:

	ENTRY POINTS:
	.TAPE
Parameters:	Integer
Result:	None
Basic FORTRAN:	Not callable (Note 1)
FORTRAN IV:	Not callable (Note 1)
ALGOL:	Not callable (Note 1)
Errors:	None

NOTES: 1. In FORTRAN and ALGOL use utility statements.

.ZRLB

PURPOSE: Eliminates calls to $LIBR and $LIBX that are unnecessary in DOS and DOS-M. .ZRLB is called by FADSB, FMP, FDV, .FLUN, and .PACK.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:		.ZRLB (P)
EXTERNAL REFERENCES:		$LIBR,$LIBX,.OPSY
CALLING SEQUENCES:		see below

```
 SUB NOP
     JSB $LIBR
     NOP
     JSB .ZRLB
     DEF EXIT
      :
EXIT JSB $LIBX
     DEF SUB
     JMP SUB,I
```

METHOD:

In RTE, DOS, and DOS-M, this routine replaces the instruction "JSB .ZRLB" in the calling sequence with an "RSS". In DOS and DOS-M only, the instructions "JSB $LIBR" and "JSB $LIBX" are both replaced by "RSS".

ATTRIBUTES:

	ENTRY POINTS:
	.ZRLB
Parameters:	None
Result:	None
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

..MAP

PURPOSE: Computes the address of a specified element of a 2 or 3 dimension array; returns the address in the A-Register.

	BCS	DOS/RTE (TYPE)
ENTRY POINTS:	..MAP (U)	
EXTERNAL REFERENCES:	None	

CALLING SEQUENCES:

```
For 2 dimensions:                  For 3 dimensions:
LDA = DØ                           LDA = D-1
LDB N (see below)                  LDB N (see below)
JSB ..MAP                          JSB ..MAP
DEF base address                   DEF base address
DEF 1st subscript                  DEF 1st subscript
DEF 2nd subscript                  DEF 2nd subscript
DEF length of 1st dimension        DEF 3rd subscript
→ address in A                     DEF length of 1st dimension
                                   DEF length of 2nd dimension
                                   → address in A
```

N = number of words per variable.

ATTRIBUTES:

	ENTRY POINTS:
	..MAP
Parameters:	Integer
Result:	Integer
Basic FORTRAN:	Not callable
FORTRAN IV:	Not callable
ALGOL:	Not callable
Errors:	None

SECTION III

THE FORMATTER

THE FORMATTER

The Formatter is a subroutine that is called by relocatable programs to perform formatted data transfers, to interpret formats, to provide unformatted input and output of binary data, to provide free field input, and to provide buffer-to-buffer conversion. The Formatter is first given a string of ASCII characters that constitutes a format code. This "format" tells the Formatter the variables to transfer, the order, and the conversion (on input, ASCII characters are converted to binary values and on output, binary values are converted to ASCII). Then the calling program gives the Formatter a string of variables to be output or filled by input.

In FORTRAN and ALGOL programming, the programmer first defines a FORMAT string through FORMAT statements.

For example:

```
FORTRAN:  1Ø FORMAT (I5,A2,5F12.3)
          identifier   actual format

ALGOL:    FORMAT F23 (I5,A2,5F12.3);
          identifier   actual format
```

Then the programmer uses a READ or WRITE statement giving the logical unit number of the device to be used, the format identifier, and a list of variables.

For example:

```
FORTRAN:  2Ø WRITE (2,1Ø) INT,LETR,ARRAY
            logical  format     variable
              unit   identifier   list

ALGOL:    WRITE (2,F23, INT, LETR, VARI);
            logical  format     variable
              unit   identifier   list
```

The FORTRAN and ALGOL Compilers automatically generate the correct calls to the Formatter. In assembly language, the programmer is responsible for all calls to the Formatter, as will be discussed later.

There are three different formatters used in relocatable Hewlett-Packard software systems:

1. 4K Formatter
2. Basic FORTRAN Formatter
3. FORTRAN IV Formatter

The 4K Formatter is the simplest formatter, as it must operate in 4,096 words of memory. The Basic FORTRAN Formatter includes all the features of the 4K Formatter, plus several more. The FORTRAN IV Formatter is expanded even further to include double precision number conversion.

These three formatters are distributed as follows:

1. 4K Formatter:
    a. 24147 Non-EAU 4K FORTRAN Library
    b. 24148 EAU 4K FORTRAN Library
2. Basic FORTRAN Formatter:
    a. 24146 non-EAU Relocatable Program Library
    b. 24145 EAU Relocatable Program Library
    c. RTE/DOS Basic FORTRAN Formatter (separate)
3. FORTRAN IV Formatter:
    a. 24149 BCS FORTRAN IV Library
    b. 24152 RTE/DOS FORTRAN IV Library

FORMATTED INPUT/OUTPUT

Formatted input/output is distinguished from unformatted input/output by the presence of a format identifier in the READ or WRITE statement. The format identifier refers to a format that is a string of ASCII characters bounded by parentheses. The ASCII characters consist of a series of format specifications or codes. Each code specifies either a conversion or an editing operation. Conversion specifications tell the formatter how to handle each variable in the data list.

To summarize:

Format specifications may be nested (enclosed in parenthesis) to a depth of one level. In FORTRAN IV they may be nested to a depth of four levels.

Conversion specifications tell the formatter how to convert variables into ASCII output and how to convert ASCII input into binary variable data.

Editing specifications tell the formatter what literal strings to put on output, when to begin new records and when to insert blanks.

FORMAT SPECIFICATIONS

A format has the following form: (*spec*,...,r(*spec*,...),*spec*,...)

where:

spec is a format specification and r is an optional repeat factor which must be an integer.

Conversion Specifications

rEw.d	Real number with exponent	
rFw.d	Real number without exponent	
rIw	Decimal Integer	
r@w, rKw,rOw	Octal Integer	[Not available on 4K]
rAw,rRw	ASCII character	[Not available on 4K]

FORTRAN IV FORMATTER ONLY:

srDw.d	Double precision number with exponent
srGw.d	Real number with digits
rLw	Logical variable

Editing Specifications

nX Blank field
nH *character string*
r"*character string*"
r/ begin new record

where:

r is an integer repetition factor,
w and n are non-zero integer constants representing the width of a field in the external character string,
d is an integer constant representing the digital fraction in the part of the string, and
s is an optional scale factor.

THE E SPECIFICATION

The E specification defines a field for a real number with exponent.

Output

On output, the E specification converts numbers (integers, real, or double precision) in memory into character form. The E field is defined in a format by the presence of the E specification (Ew.d). The field is w positions in the output record. The variable is printed out in floating-point form, right justified in the field as

$$\overbrace{-.\underbrace{x_1 \ldots x_d}_{d}\ E\pm ee}^{w}$$

where

$x_1 \ldots x_d$ are the most significant digits of the value, the e's are the digits of the exponent w is the width of the field, d is the number of significant digits, and the minus sign is present if the number is negative.

The w must be large enough to contain the significant digits (d), the sign, the decimal point, E, and the exponent. In general, w should be greater than or equal to d + 6.

If w is greater than the number of positions required for the output value, the quantity is right justified in the field with spaces to the left. If w is not large enough (e.g., less than d + 6), then the value of d is truncated to fit in the field. If this is not possible, the entire field is filled with dollar signs ($).

EXAMPLES:

FORMAT	DATA ITEM	RESULT
E1Ø.3	+12.34	‸‸.123E+Ø2
E1Ø.3	-12.34	‸-.123E+Ø2
E12.4	+12.34	‸‸‸.1234E+Ø2
E12.4	-12.34	‸‸-.1234E+Ø2
E7.3	+12.34	.12E+Ø2
E5.1	+12.34	$$$$$

Input

The E specification on input tells the formatter to interpret the next w positions in the record as a real number with exponent. The formatter then converts the field into a number and stores it into the variable specified in the variable list.

The input field may consist of integer, fraction, and exponent subfields

$$\overbrace{\pm n....n}^{\text{integer field}}.\overbrace{n....n}^{\text{fraction field}}\overbrace{E\pm\ ee}^{\text{exponent field}}$$

where the format equals Ew.d.

Rules for E Field Input

1. The width of the input item must not be greater than w characters, with $w > d$.
2. Initial + and E are optional.
   Example: 123. = +123., 12.+6 = 12.E6
3. If E is present, the initial + of the exponent is optional.
   Example: 123.4EØ6
4. If the decimal point is left out, the formatter inserts it by multiplying the integer field by 10^{-d}.
   Example: If format = E9.4, 123456E+6 = 12.3456E+6
5. Spaces are ignored in the Basic FORTRAN and 4K Formatter, but in the FORTRAN IV Formatter blanks are evaluated as zeroes (0).
6. Any combination of integer field, fraction field, and exponent field is legal:

```
123.456E6
   .456E6
   .456
123.E6
123.
    E6
(all blanks = Ø)
```

NOTE: Input to F, G, D and I fields is interpreted in the same way as the E field.

THE F SPECIFICATION

The F Specification defines a field for a fixed point real number (no exponent).

Output

On output, the F specification converts numbers (integer, real, or double precision) in a format by the presence of the F specification (Fw.d). The field is w positions in the output record. The variable is printed out right-justified in fixed-point form with d digits to the right of the decimal point:

$$\underbrace{\overbrace{-x....x}^{\text{integer field}}.\overbrace{xx..x}^{\text{fraction field (d)}}}_{w}$$

Where w is the total width of the field, the negative sign (-) is optional (positive numbers are unsigned), d is the length of the fraction field (empty if d=0).

If w is greater than the number of positions required for the output value, the quantity is right justified in the field with spaces to the left. If w is not large enough to hold the data item, then the value of d is reduced to fit. If this is not possible, the entire field is filled with dollor signs ($).

EXAMPLES:

FORMAT	DATA ITEM	RESULT
F1Ø.3	+12.34	^^^^12.34Ø
F1Ø.3	-12.34	^^^-12.34Ø
F12.3	+12.34	^^^^^^12.34Ø
F12.3	-12.34	^^^^^-12.34Ø
F4.3	+12.34	12.3
F4.3	+12345.12	$$$$

Input

Input to an F field is identical to an E field. All the rules under the E specification apply equally to the F specification.

THE D SPECIFICATION

The D specification is available only on the FORTRAN IV formatter. The effect is exactly the same as using an E specification with exception that on output "D" begins the exponent field instead of "E".

EXAMPLES:

```
        D1Ø.3
        D12.4
        D7.3
```

THE G SPECIFICATION

The G specification defines an external field for a real number. The magnitude of the number determines whether or not there is an exponent field.

Output

On output, the G specification converts numbers (integer, real, or double precision) in memory into character form. The G field is defined in a format by the presence of the G specification (Gw.d). The field is w spaces wide, with d significant digits. The format of the output depends on the magnitude of the number (N):

Magnitude	Output Conversion
$0.1 \le N<1$	F(W-4).d,4X
$1 \le N<10$	F(W-4).(d-1),4X
⋮	⋮
$10^{d-2} \le N<10^{d-1}$	F(W-4).1,4X
$10^{d-1} \le N<10^{d}$	F(W-4).0,4X
Otherwise	sEw.d (s is scale factor)

NOTE: The scale factor is applied only when the G conversion is done as E.

Example Output

The following real numbers are converted under a G10.3 specification:

Number	Output Format
.Ø5234	^^.523E-Ø1
.5234	^^.523^^^^
52.34	^^52.3^^^^
523.4	^^523.^^^^
5234.	^^.523E+Ø4

Input

Input processing of a Gw.d specification is identical to that of an Ew.d specification.

THE SCALE FACTOR (FORTRAN IV ONLY)

The optional scale factor for F,E,G, and D conversions is of the form:

nP

When n, the scale factor, is an integer constant or a minus followed by an integer constant. Upon initialization of the formatter, the scale factor equals zero. Once a scale factor is encountered, it remains in effect for all subsequent F,E,G and D fields until another scale is encountered.

The scale factor effects are as follows:

1. F,E,G,D input (provided no exponent exists in the external field): internally represented number equals externally represented number times ten raised to the -nth power. That is, $IN=XN*10^{-n}$ where IN and XN represent internal and external numbers, respectively.
2. F,E,G,D, input with exponent field in external field: no effect.
3. F output: external number equals internal number times ten raised to the nth power. ie,

   $$XN = IN*10^{n}$$

4. E,D output: mantissa is multiplied by 10^{n} and the exponent is reduced by n. If $n \le 0$, there will be -n leading zeroes and d + n significant digits to the right of the decimal point. If n>o, there will be n significant digits to the left of the decimal point and d-n + 1 to the right. The scale factor when applied to E and D output has the effect of shififting the decimal point to the left or right and adjusting the exponent accordingly. Note that when n > 0, there are d + 1 significant digits in the external field.
5. G output: If F conversion is used, the scale factor has no effect. If E conversion is used, the scale factor has the same effect as with E output.

For example,

Input conversion

External field	Format	Internal number
528.6	1PF1Ø.3	52.86
.5286E+Ø3	1PG1Ø.3	528.6
528.6	-2PD1Ø.3	5286Ø.

Output conversion

Internal number	Format	External field
528.6	1PF8.2	^5286.ØØ
.5286	2PE1Ø.4	52.86ØE-Ø2
5.286	-1D1Ø.4	^.0529D+Ø2
52.86	1PG1Ø.3	^^52.9^^^^
-5286.	1PG1Ø.3	-5.286E+Ø3

THE I SPECIFICATION

The I specification defines a field for decimal integer.

Output

On output, the I specification converts numbers (integer, real, or double precision) in memory into character form. The I field is defined in a format by the presence of the I specification (Iw). The field occupies w positions in the output record. The variable is converted to an integer, if necessary, and printed out right-justified in the field (spaces to the left) as:

$$\underbrace{\ldots \wedge \underline{\wedge}\ x,\ \ldots xd}_{w}$$

where

x, ... xd are the digits of the value, (max = 5), w is the width of the field in characters, and the minus sign (-) is present if the number is negative.

If the output field is too short, the field is filled with dollar signs ($).

Format	Data Item	Result
I5	-1234	-1234
I5	+12345	12345
I4	+12345	$$$$
I6	+12345	^12345

Input

The I specification on input (Iw) is equivalent to an Fw.0 specifications. The input field is read in, the number is converted to the form suitable to the variable (integer, real, double real), and the binary value is stored in the variable location.

During input, if a value is less than -32768_{10}, the value is converted to +32767.

EXAMPLES:

Format	Input Field	Internal Result
I5	-^123	-123
I5	12ØØ3	12ØØ3
I4	^1Ø2	1Ø2
I1	3	3

O,K,@ SPECIFICATION (NOT AVAILABLE WITH 4K FORMATTER)

These three specification types (O,K,@) are equivalent; they are all used to convert octal (base eight) numbers.

Output

On output, the octal specification (O,K,@) converts an integer value in memory into octal digits for output. The octal field is defined in a format by the presence of the O(Ow), K(Kw), or @(@w) specification. The field is w octal digits wide. The integer value is converted and right justified in the field as:

$$\underbrace{\ldots_{\wedge\wedge} d_1 \ldots d_n}_{w}$$

where

$d_1 \ldots d_n$ are the octal digits (6 maximum), $\ldots_{\wedge\wedge}$ are lead spaces, and w is the width.

If w is less than 6, the w least significant octal digits are written.

Input

On input, the octal specification tells the formatter to interpret the next w positions in the input record as an octal number. The formatter converts the digits into an octal integer and stores it into an integer variable.

If w is greater than or equal to six, up to six octal digits are stored; non-octal digits with the field are ignored.

If w is less than six or if less than six octal digits occur in the field, the result is right-justified in the variable with zeroes (0) to the left.

If the value of the octal digits in the field is greater than 177777, the results are unpredictable.

EXAMPLES:

Format	Input Field	Internal Result
@6	123456	123456
@7	-123456	123456
2K5	2342342342	Ø23423 and Ø42342
2@4	,396E-Ø5	ØØØ36 and ØØØØØ5

L SPECIFICATION

The L specification allows input or output of logical values:

TRUE = T (external), negative (internal)
FALSE = F (external), non-negative (internal)

Output

On output, the L specification converts numbers (integer, real, or double precision) in memory into their external logical value (T or F). The L field is defined by the presence of the L specification (Lw). The field is w spaces wide, consisting of w-1 blanks followed by a T or F.

Input

On input, the L specification converts an external character field into the internal true of false of value. The L specification (Lw) specifies a field w spaces wide, consisting of optional blank, a T or F and optional trailing characters. A T is converted to -32,768 (100000_8) and an F is converted to 0.

A AND R SPECIFICATIONS (NOT AVAILABLE WITH 4K FORMATTER)

The A and R specifications define a field of one or two ASCII characters. ASCII characters are stored as two 8-bit codes per integer variable.

Output

On output, the A and R specifications transfer ASCII character codes from memory to an external medium. The field is defined by an A or R specification (Aw or Rw). The field is w positions wide in the output record. For w $\geq$ 2, A and R are equivalent: the field consists of w-2 blanks followed by two characters (first the character in the left or high-order bits part of the variable, then the character in the right part of the variable).

For example:

Variable	Format	Output Format
AB	A2	AB
AB	R4	^^AB
AB	A6	^^^^AB

In order to output a string of characters, the repeat factor must be used.

For example:

Variable	Format	Output
AB, CD, EF, GH	4A2	ABCDEFGH

For w = 1, the FORTRAN IV and Basic FORTRAN formatters differ.

Basic FORTRAN Formatter

In Basic FORTRAN, the A specification is the same as the R. For w = 1, A and R specify the character in the right half of the variable.

For example:

Variable	Format	Output
XY	Al or Rl	Y

FORTRAN IV Formatter

The R specification is the same as in the Basic FORTRAN Formatter.

The Al specification takes the character from the left half of the variable.

For example:

Variable	Format	Output
XY	Al	X

Input

On input, the A and R specifications transfer ASCII character codes from an external medium to internal memory. The field is defined by an A or R specification (Aw or Rw). The field is w positions wide. If w $\geq$ 2, the right most two characters are taken from the input field.

For example:

Input Field	Format	Variable
MN	A2	MN
MNOP	R4	OP
MNOPQR	A6	QR

In order to read in a string of more than two characters, the repeat factor must be used.

For example:

Input Field	Format	Variable
MNOPQRSTUV	5A2	MN,OP,QR,ST,UV
FGHIJK	3A2	FG,HI,JK

For w = 1, the FORTRAN IV and Basic FORTRAN Formatter differ.

Basic FORTRAN Formatter

In Basic FORTRAN the A is the same as the R. For w = 1, A and R read in one character and places it in the right half of the variable with binary zeroes in the left.

For example:

Input	Format	Variable
X	A1 or R1	$\boxed{00000000_2}\boxed{X}$
		left right
		computer word

FORTRAN IV Formatter

The R spcification is the same as in the Basic FORTRAN Formatter.

For A1, one character is read in and placed in the left half of the computer word. An ASCII blank is placed in the right half.

For example:

Input	Format	Variable
X	A1	X‸

Compatibility

The FORTRAN IV Formatter can be modified at run-time to interpret A as in Basic FORTRAN. This is done by calling the OLDIO entry point:

```
CALL  OLDIO
```

To change back to a FORTRAN IV A specification call NEWIO:

```
CALL NEWIO
```

The FORTRAN IV Formatter always begins operation in the NEWIO state.

X SPECIFICATION

The X specification produces spaces on output and skips characters on input. The comma (,) following X in the format is optional.

Output

On output, the X specification causes spaces to be inserted in the output record. The X field is defined by the presence of an X specification (nX) in the format, where n is the number of spaces to be inserted. (X alone = 1X; ØX is not permitted.)

Examples

Format

E8.3,5X,F6.2,5X,I4

Data Values

+123.4, -12.34, -123

Output Field

.123E+03^^^^^-12.34^^^^^-123

Input

On input, the X specification causes characters to be skipped in the input record. The X field is defined by the presence of an X specification (nX) in the format, where n is the number of characters to be skipped. (X alone = 1X; ØX is not permitted.)

Example

Format

8X,I2,1ØX,F4.2,1ØX,F5.2

Input Field

WEIGHT ^^1Ø^^PRICE^^$1.98^^TOTAL^^$19.80

Internal Values

1Ø, 1.98, 19.8Ø

" ", H SPECIFICATIONS (Literal Strings)

The H and " " specifications provide for the transfer, without conversion, of a series of ASCII characters (except that quotation marks -"- cannot be transferred using " "). A comma after this specification is optional.

Output

On output, the ASCII characters in the format specification (there is no associated variable since this is only an editing specification) are output as headings, comments, titles, etc. The specifications are of the form:

$$nHc_1c_2...c_n \quad \text{or} \quad "c_1c_2...c_n"$$

where

n is the numbers of characters to be transmitted, $c_1c_2...c_n$ are the characters themselves, and H or "..." are the specification types.

(H alone = 1H; ØH is not permitted.)

Note that with " ", the field length is not specified; that is determined by the number of characters between the quotation marks.

Examples:

Format	Result
2ØH‸THIS‸IS‸AN‸EXAMPLE	‸THIS IS AN EXAMPLE
"THIS‸ALSO‸IS‸AN‸EXAMPLE"	THIS‸ALSO‸IS‸AN‸EXAMPLE
3"ABC"	ABCABCABC

Input

If H is used on input, the number of characters needed to fill the specification is transmitted from the input record to the format. A subsequent output statement will transfer the new heading to the output record. In this way, headings can be altered at run-time.

If " " is used on input, the number of characters within the quotation marks is skipped on the input field.

Example:

Format

```
31H‸‸‸‸‸‸‸‸‸‸‸‸‸‸‸‸‸‸‸‸‸‸‸‸‸‸‸‸‸‸‸
```

Input

```
H‸INPUT‸ALLOWS‸VARIABLE‸HEADERS
```

Result

```
31HH‸INPUT‸ALLOWS‸VARIABLE‸HEADERS
```

/ SPECIFICATION

The / specification terminates the current record. The / may appear anywhere in the format and need not be set off by commas. Several records may be skipped by preceding the slash with a repetition factor (r-1 reocrds are skipped for r/).

On output, a new record means a new line (list device), a carriage return-linefeed (punch device), or an end-of-record (magnetic tape). Formatted I/O records can be up to 67 words (134 characters) long.

On input, a new record is a new "unit record" (card reader), is terminated by a carriage return-linefeed (teleprinter), or is terminated by an end-of-record (magnetic tape).

> *NOTE: When the formatter reaches the end of a format and still has values to output, it starts a new record.*

Examples:

Format

```
22X,6HBUDGET/// 6HWEIGHT,6X, 5HPRICE,9X,
5HTOTAL,8X
```

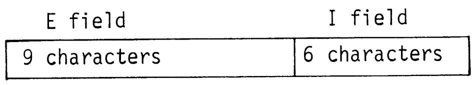
Result

```
(line 1) ^^^^^^^^^^^^^^^^^^^^^^BUDGET
(line 2)
(line 3)
(line 4) WEIGHT ^^^^^^PRICE^^^^^^^^^TOTAL^^^^^^^^
```

HOW TO PUT FORMATS TOGETHER

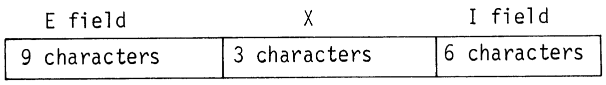
1. When two specifications follow each other they are concatenated.

   Format: E9.4,I6



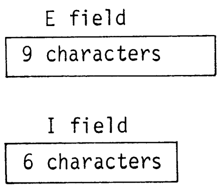
2. To leave space between numbers use X.

   Format: E9.4,3X,I6



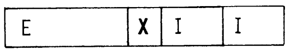
3. To start a new Line, use /

   Format: E9.4/I6



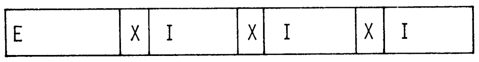
4. Specifications can be gathered together into groups and surrounded by parenthesis.

   Example: (E9.3, 2X, I6)

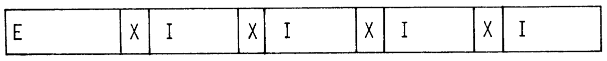


These groups can be nested one level deep, except in the FORTRAN IV Formatter they can be four levels deep. For example,

(E9.3,3(2X,I6))

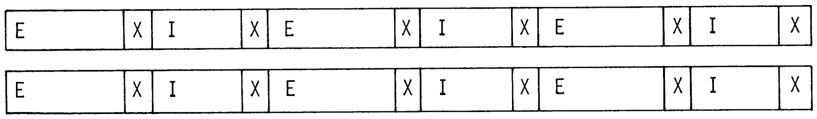


(E9.3,3(2X,I6)2X,2(I8))



5. Use the repetition factor to repeat single specifications (except nH) or groups of specifications. This is done by preceding the specification or parenthetical groups with a repeat count, r. THe conversion is repeated up to r times, unless the list of variables is exhausted first.

   3(E9.3,2X,I6,2X)/



6. Use the principle of unlimited groups -- when the formatter has exhausted the specifications of a format and still has list items left, it inputs a new record for a READ or outputs the present record for a WRITE and returns to the last, outer-most unlimited group within the format. An unlimited group is a set of specifications enclosed In parenthesis. If the format has no unlimited groups, the formatter returns to the beginning of the format.

Example: Format = (I5,2(3X,F8.4,8(I2))

Format = (I5,2(3X,F8.4,8(12I2)),4X,3(I6))

Format = (I5,3X,4F8.4,3X)

7. Keep in mind the accuracy limitations of your data. Although the formatter will print out or read in as many digits as specified, only certain digits are significant:

   Integer variables can be between $-32,768_{10}$ and $+32,767_{10}$.
   Floating-point numbers can guarantee 6 digits of accuracy (plus exponent).
   Double precision can guarantee 11 digits of accuracy (plus exponent).

8. On input to the FORTRAN IV formatter blanks are interpreted as zero digits, while on input to the other two formatters, blanks are not evaluated as part of the data item.

   The FORTRAN IV Formatter can be made to act exactly as the other Formatters do by calling entry point OLDIO. This condition can be reversed by calling entry point NEWIO. These calls are made in FORTRAN as:

```
CALL OLDIO
CALL NEWIO
```

   In Assembly Language as:

```
JSB OLDIO       JSB NEWIO
DEF *+1         DEF *+1
→               →
```

FREE FIELD INPUT

When free field input is used, a format is not necessary. Special symbols included within the data direct the conversion process:

space or,	Data item delimiters
/	Record terminator
+ -	Sign of item
. E + - D	Floating point number
@	Octal integer
"..."	Comments

All other ASCII non-numeric characters are treated as spaces (and delimiters). Free field input may be used for numeric data only.

FORTRAN

In FORTRAN, a free field input operation has this form:

READ (*unit*, *) *variable list*

ALGOL

In ALGOL, a free field input operation has this form:

READ (*unit*, *, *variable list*);

Data Item Delimiters

Any contiguous string of numeric and special formatting characters occuring between two commas, a comma and a space, or two spaces, is a data item whose value corresponds to a list element. A string of consecutive spaces is equivalent to one space. Two consecutive commas indicate that no data item is supplied for the corresponding list element; the current value of the list element is unchanged. An initial comma causes the first list element to be skipped.

EXAMPLES:

1)

```
        READ(5,*)I,J,K,L
```

Input data: 1720, 1966, 1980, 1392

2)

```
        READ(5,*)I,J,K,L
```

Input data: 1266,,1794,2000

Result: I contains 1720
        J contains 1966
        K contains 1980
        L contains 1392

Result: I contains 1266
        J contains 1966
        K contains 1974
        L contains 2000

Floating Point Input

The symbols used to indicate a floating point data item are the same as those used in representing floating point data for FORMAT statement directed input:

```
 Integer  Fraction Exponent
  Field \   Field    Field
          \    |     /
  ______  _____  ___
 +n...n.n...n+ee
 -      ^     -
 n      |     E
        |     D (in FORTRAN IV only)
        |
     decimal point
```

If the decimal point is not present, it is assumed to follow the last digit.

EXAMPLES:

```
      READ(5,*)A,B,C,D,E
```

Input Data: 3.14, 314E-2, 3140-3, .0314+2, .314E1

All are equivalent to 3.14

Octal Input

An octal input item has the following format:

@$x_1 \ldots x_d$

The symbol @ defines an octal integer. The x's are octal digits each in the range of 0 through 7. List elements corresponding to the octal data items must be type integer.

Record Terminator

A slash within a record causes the next record to be read as a continuation of the data list; the remainder of the current record is skipped as comments.

Example:

```
      READ(5,*)II,JJ,KK,LL,MM
```

```
Input data:  987, 654, 321, 123/DESCENDING
             456

Result:  II  contains 987
         JJ  contains 654
         KK  contains 321
         LL  contains 123
         MM  contains 456
```

List Terminator

If a line terminates (with a carriage return and linefeed) and a slash has not been encountered, the input operation terminates even though all list elements may not have been processed. The current values of remaining elements are unchanged.

EXAMPLES:

```
      READ(5,*)A,B,C,J,X,Y,Z
```

Input Data:

```
      A=7.987  B=5E2  C=4.6859E-3 carriage return and linefeed
      J=3456 carriage return and linefeed
```

```
Result:  A contains 7.987
         B contains 5E2
         C contains 4.6859E-3

         J,X,Y,Z are unchanged.
```

Comments

All characters appearing between a pair of quotation marks in the same line are considered to be comments and are ignored.

EXAMPLES:

"6.7321"	is a comment and ignored
6.7321	is a real number

UNFORMATTED INPUT /OUTPUT

Read and write operations can be unformatted as well as formatted. On an unformatted operation, binary values are transferred to or from a specified logical unit without any conversion.

In FORTRAN, an unformatted READ statement has the form:

READ (*unit*) *variable list*

Binary records are read in from *unit* and assigned serially to the locations in the variable list.

In ALGOL, an unformatted READ statement has the form:

READ (*unit*,*variable list*)

In FORTRAN, an unformatted WRITE statement has the form:

WRITE (*unit*) *variable list*

The values in the variable list are packed into physical records of 60 words each and are output to the *unit* specified. The variable list which may consist of several physical records, is called a logical record.

In ALGOL, an unformatted WRITE statement has the form:

WRITE (*unit, variable list*)

In Assembly Language, the program calls the formatter directly. (See "Assembly Language Calling Sequences.")

Records

Unformatted I/O through the formatter is limited to physical records of 60 words maximum. If a variable list contains more than 60 words of data, the data is broken into more than one record. (For example, 100 words of data are broken into two records of 60 and 40 words each.)

When paper tape or unit record devices are used, (teleprinter, mark sense card reader etc.) however, only 59 words of each record contain data. The first word issued is for the record length.

ASSEMBLY LANGUAGE CALLING SEQUENCES

In FORTRAN and ALGOL, when the programmer uses a READ or WRITE statement the compiler generates all the necessary calls to the Formatter.

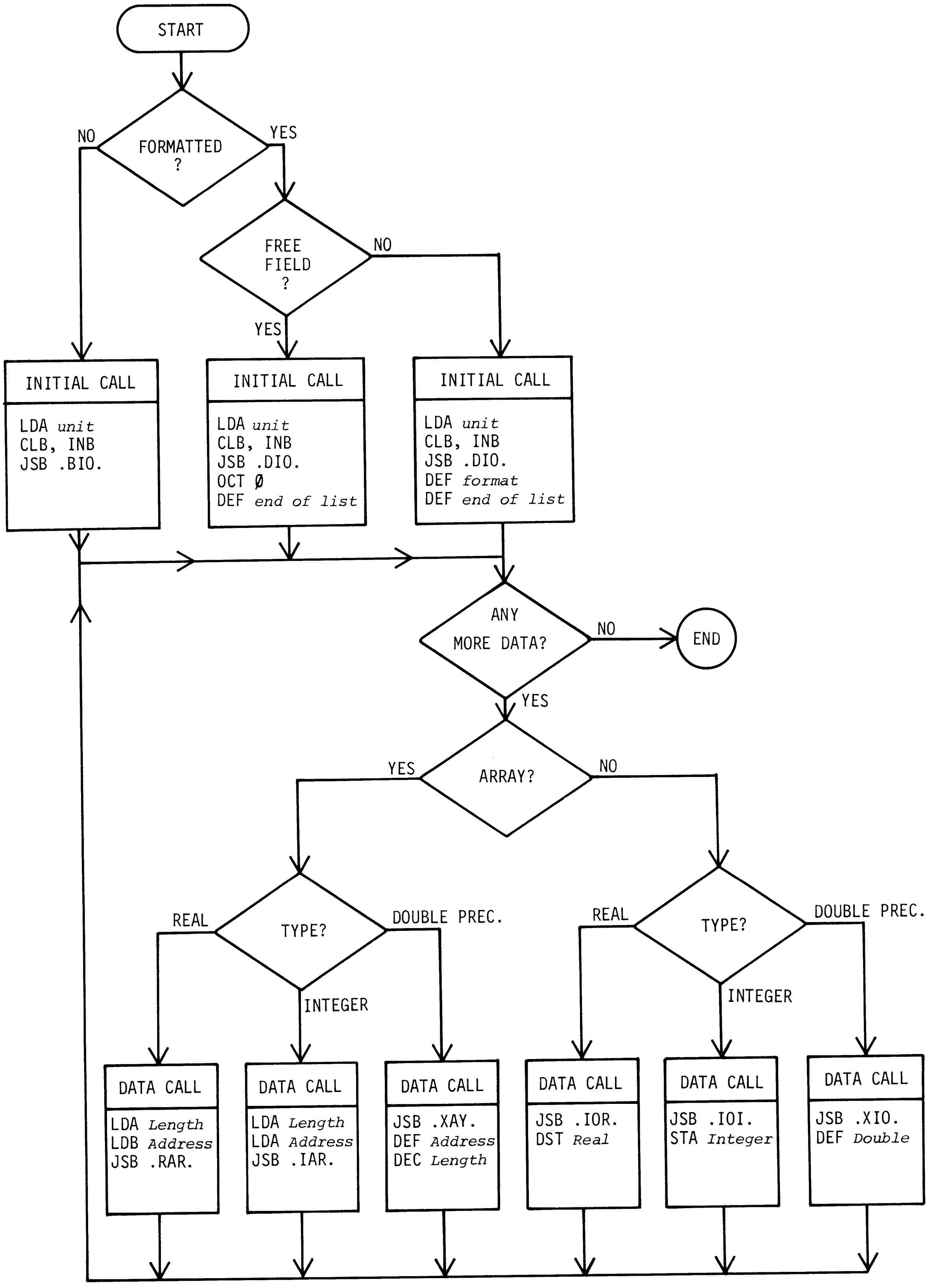
In Assembly Language the programmer is responsible for all calls to the Formatter. For each I/O operation, the program must first make an "Initialization" call (entry points .DIO and .BIO). This call establishes the format to be used (if any), and the logical unit and a way to say whether the operation is input or output. Then, for each data item, the program must make a separate call which depends on the type of data. Finally, for output only, the program must make a termination call that tells the Formatter to output the last record.

Figure 3-1 flowcharts the process of selecting an input calling sequence. Figure 3-2 flowcharts the output calling sequence.

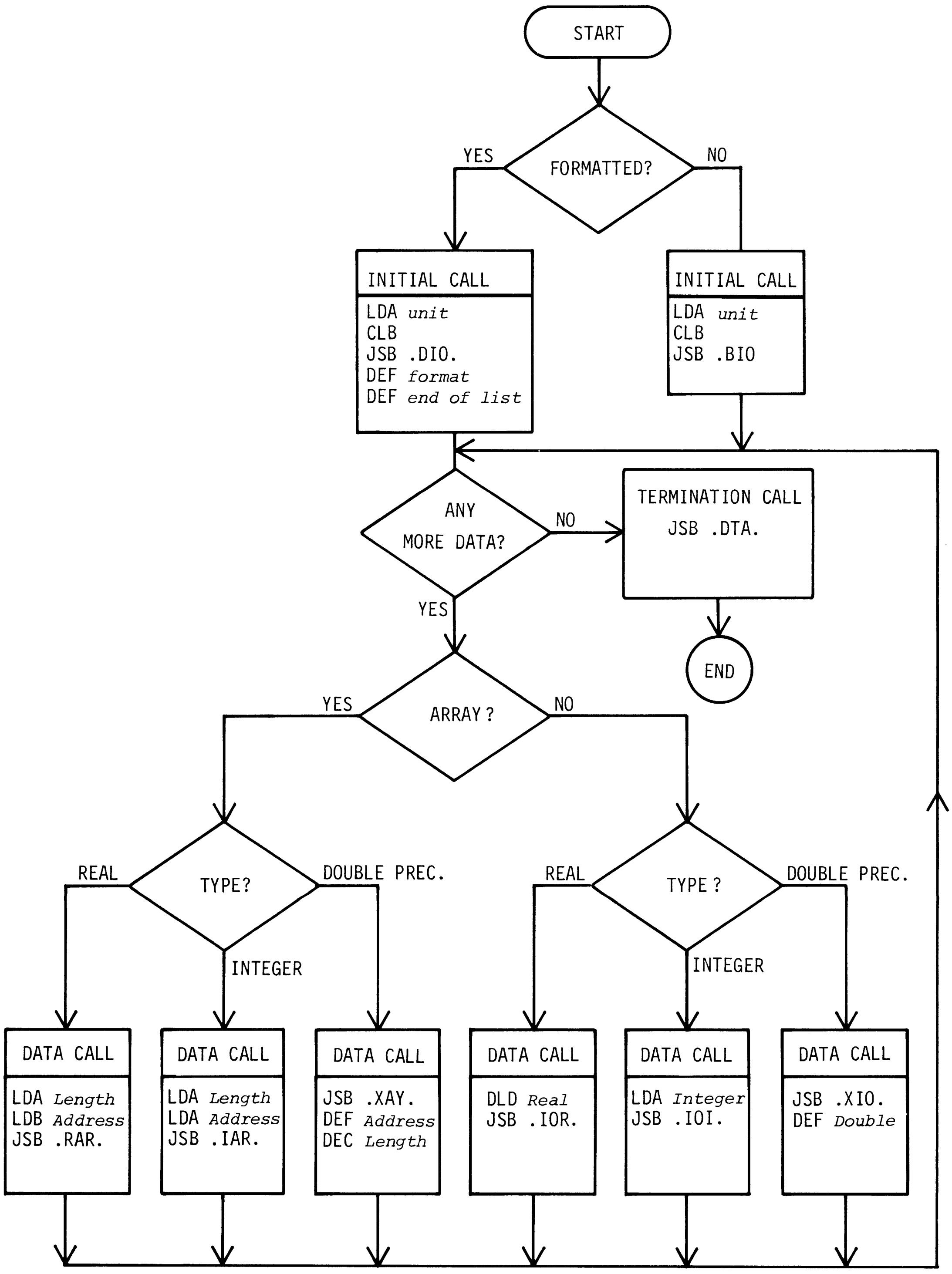
Variable items in the calling sequences include:

unit	is the logical unit number of the desired I/O device.
format	is the label of an ASC psuedo-instruction that defines the format specification.
end of list	is the location following the last data call to the formatter. When an error occurs in the format specification or the input data, the formatter returns to this location.
real	is the address of the real variable.
integer	is the address of the integer variable.
double	is the address of the double precision variable
length	is the number of elements (not the number of memory locations) in the array. Maximum length of an external physical record is 67 words for formatted data and 60 words for binary data. Formatted data blocks can be of any length if the format breaks the data in multiple records using "/" and unlimited groups. If binary data exceeds 60 words, the record is read in or written out and the formatter skips to the next record. (Note: For this reason, binary data should be read in with the same variable list as that used to write it out.)
address	is the first location of the array.

INPUT
SELECTING A CALLING SEQUENCE



OUTPUT
SELECTING A CALLING SEQUENCE



NOTES

1. Double precision calls are available only in the FORTRAN IV Formatter.

2. In the FORTRAN IV Formatter, there are alternative calling sequences for data items:

```
     Real Variable                    Real Arrays
        JSB .RIO.                        JSB .RAY.
        DEF real                         DEF address
                                         DEC length
     Integer Variables                Integer Arrays
        JSB .IIO.                        JSB .IAY.
        DEF integer                      DEF address
                                         DEC length
```

3. Also in the FORTRAN IV Formatter, the statement "DEC *length*" in array calling sequences can be replaced by "DEF L,I" where L is defined elsewhere in the program as the array length.

INTERNAL CONVERSION

The Formatter provides the programmer with the option of using the conversion parts of the Formatter only without any input or output. This process is called "internal conversion."

On "input", ASCII data is read from a buffer and converted according to a format (or free field) into a variable list. (This is known as decoding.)

On "output", binary data is converted to ASCII according to a format and stored in a buffer. (This is known as encoding.)

Internal Conversion Calling Sequence

Output (Binary to ASCII Conversion): ENCODING

```
CLA
CLB
JSB .DIO.
DEF buffer (destination)
DEF format
DEF end of list
 :
Calls to define each variable
     (Same as regular calls)
 :
Termination Call
     (Same as regular calls)
```

where *buffer* is a storage area for the ASCII "output" to be stored into.

Input (ASCII to Binary Conversion): DECODING

```
Formatter           Free Field

CLA                 CLA
CLB, INB            CLB, INB
JSB  .DIO.          JSB  .DIO.
DEF  buffer         DEF  buffer
DEF  format         ABS  Ø
DEF  end of list    DEF  end of list
 :                   :
Calls to define each variable
     (Same as regular calls)
```

where *buffer* is a storage area containing ASCII characters which will be converted by the Formatter into binary values.

NOTES

1. Internal conversion ignores "/" specifications or unlimited groups. The concept of records does not apply during internal conversion.
2. In FORTRAN, internal conversion can be used through the subroutine CODE. A call to CODE must immediately precede a READ or WRITE request where the identifier of an ASCII record buffer (array name) replaces the logical unit number. Any labels must be attached to the CODE call, as it and the READ/WRITE statement are treated as one statement. Also, the identifier of the ASCII record buffer should not be a dummy argument (formal parameter) of a subprogram.

   The calling sequences are:

   CALL CODE
   READ (*buffer, format*) *variable list*

   CALL CODE
   WRITE (*buffer, format*) *variable list*

3. In ALGOL, the entry point ACODE is used for internal conversion (since CODE is a reserved word). ACODE must be declared an "code" procedure before being called.

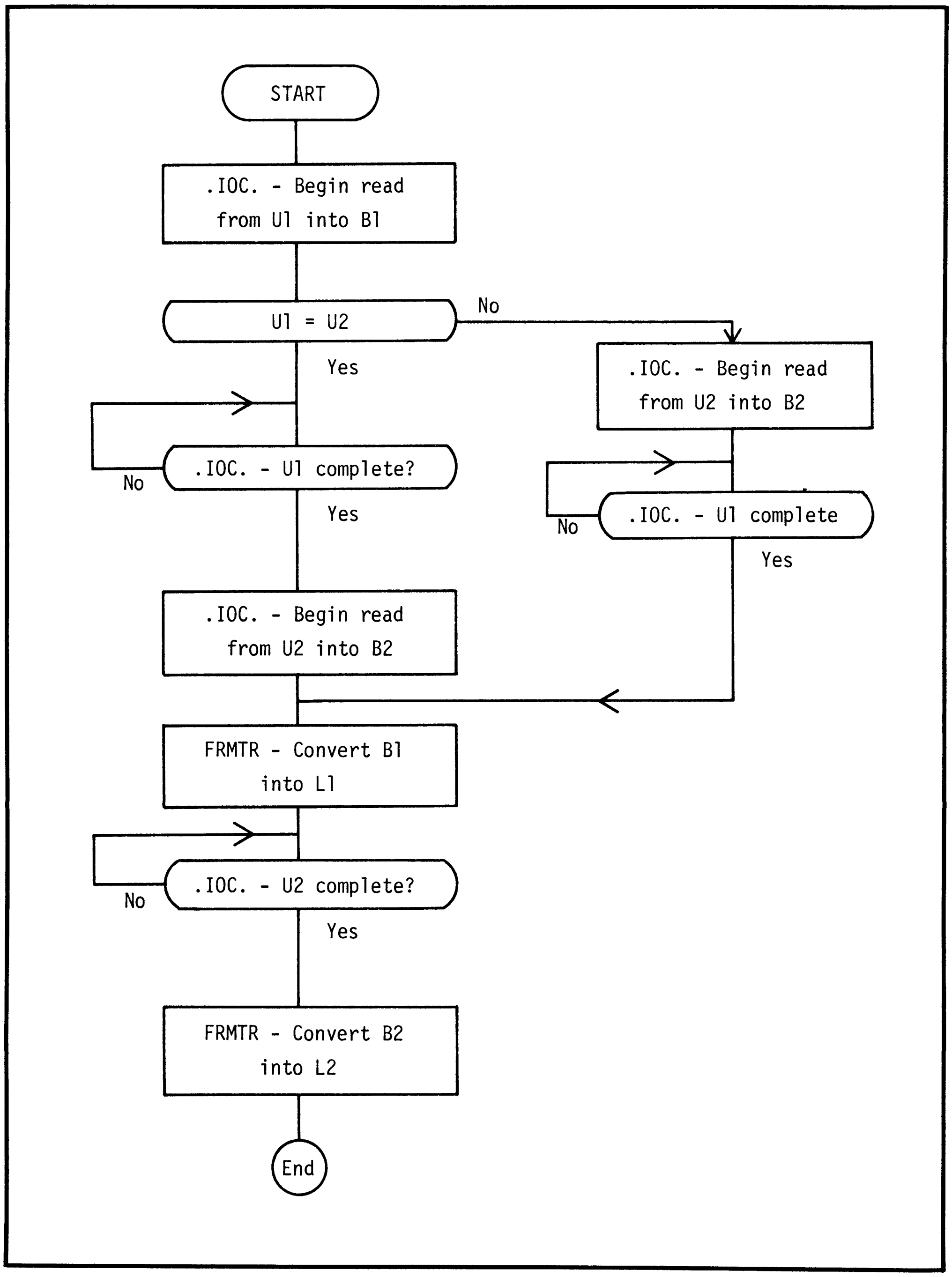
4. FORTRAN IV does not allow a subscripted variable in a READ statement. Therefore, the calling sequence must be:

   DIMENSION IRRAY(10)
   ⋮
   CALL CODE
   READ (IRRAY, *format*) *variable list*
   or
   ⋮
   CALL CODE
   WRITE (IRRAY, *format*) *variable list*

BUFFERED I/O WITH THE FORMATTER

Normally, when a program uses the Formatter, it can only execute one I/O operation at a time. However, the internal conversion feature of the Formatter can be used with direct calls to .IOC. (through the MAGTP subroutine) to provide both buffered and formatter I/O.

The following flowchart shows how a program can read in data from two units (U1 and U2) into two buffers (B1 and B2) at the same time by calling .IOC.. When unit U1 is complete, buffer B1 is converted into list L1 by the Formatter (while input continues on unit U2).


START
.IOC. - Begin read from U1 into B1
U1 = U2
No
Yes
.IOC. - Begin read from U2 into B2
.IOC. - U1 complete?
No
Yes
.IOC. - U1 complete
No
Yes
.IOC. - Begin read from U2 into B2
FRMTR - Convert B1 into L1
.IOC. - U2 complete?
No
Yes
FRMTR - Convert B2 into L2
End

EXAMPLE CALLING SEQUENCES

EXAMPLE 1: FORMATTED INPUT

Purpose

A 20 character double precision number and a 10 character integer are read and converted from the first record. 80 characters are read from the second record and stored in ASCII form in the array ALPHA. Execution continues with the instruction at ENDLS.

```
        LDA     INPUT                       Input unit number
        CLB,INB                             Input flag
        JSB     .DIO.                       Initialization enterance
        DEF     FMT                         Location of format
        DEF     ENDLS                       End of list
        JSB     .XIO.                       Declare double precision variable
        DEF     DP                          Location of variable
        JSB     .IIO.                       Declare integer variable
        DEF     I                           Location
        JSB     .IAY.                       Declare integer array
        DEF     ALPHA                       Location
        DEC     80                          Number of elements
ENDLS   →                                   (Continue program here)
        .
        .
        .
INPUT   DEC     1                           Unit number
DP      BSS     3                           Double precision variable
I       BSS     1                           Integer variable
ALPHA   BSS     80                          Integer array
FMT     ASC     9,(D20.12,I1Ø/80A1)         Format specification
```

EXAMPLE 2: UNFORMATTED OUTPUT EXAMPLE

Purpose

1000 2-word elements in the array ARRAY are punched on the standard punch unit. The output will consist of 60 word records (59 data words and 1 control word) until the entire array is punched.

```
             LDA  PUNCH         Output unit number
             CLB                Output flag
             JSB  .BIO          Binary initialization enterance
             LDA  =D1ØØØ        Number of elements in array
             LDB  ADRES         Location of array
             JSB  .RAR.         Real (2-word) array enterance
             JSB  .DTA.         Output termination
             →
             :
       PUNCH DEC  4             Unit number
       ADRES DEF  ARRAY         Location of ARRAY
       ARRAY BSS  2ØØØ          Defines 1000 2-word elements.
```

EXAMPLE 3: INTERNAL CONVERSION AND FREE FIELD INPUT

Purpose

The ASCII data starting at BUFFR is converted in free field form to binary. R will contain the binary representation of .ØØØ1234 and I will contain the binary representation of 28.

```
        CLA                          Internal conversion flag
        CLB,INB                      ASCII to binary flag
        JSB      .DIO.               Initialization enterance
        DEF      BUFFR               Location of ASCII data
        ABS      Ø                   Specifies ASCII data is in free-field form
        DEF      ENDLS               End of list
        JSB      .IOR.               Declare real variable
        DST      R                   Store binary item in R
        JSB      .IOI.               Declare integer variable
        STA      I                   Store in I
ENDLS   →
        :
R       BSS      2                   Real variable
I       BSS      1                   Integer variable
BUFFR   ASC      6,123.4E-6,28       ASCII data to be converted to binary.
```

INDICES

INDEX I

LIB.*N*, K4N.*N*, EAU.*N*, K4E.*N*

N is the revision letter

In EAU.*N* and K4E.*N*, .EAU. replaces MPY, DIV, DLDST.

INDEX II

FTN4*N* and F4D.*N*

N is the revision letter

INDEX III

F2E.*N* and F2N.*N*

N is the revision letter

In F2E.*N*, .EAU. replaces MPY, DIV, DLDST.

INDEX IV

Plotter Libraries

INDEX V

ALL ENTRY POINTS

INDEX

02116—91780