

Name	Arguments	Purpose
ABEND		for abnormal end
ALGAMA	(X)	=> ALOG(Gamma function)
ALOGAM	(X)	=> ALOG(Gamma function)
AMAXMU	(A,IDO,IW,NA)	find  max  of scattered vector
ATG	(Y,X)	PROXIM(ATAN2(Y,X),PI)
BESIO	(X)	modified Bessel Function I0(X)
BESI1	(X)	modified Bessel Function I1(X)
BESJ0	(X)	Bessel function J0(X)
BESJ1	(X)	Bessel function J1(X)
BESK0	(X)	modified Bessel Function K0(X)
BESK1	(X)	modified Bessel Function K1(X)
BESY0	(X)	Bessel function Y0(X)
BESY1	(X)	Bessel function Y1(X)
BINSIZ	(A1,A2,NAA,BL,BH,NB,BWID)	determines good histogram bins
BITPOS	(IWORDS,NBITS,IXV,NX)	finds set bits in word or array
BLOW	(IN,OUT,NBYTES,NBITS)	unpacks small integers from array
BUNCH	(IN,OUT,NBYTES,NBITS)	packs small integers into array
CALDAT	(IINDEX,CHREP,BINREP,RETC)	converts date representations
CBYT	(IA,JA,IX,J,NBITS)	copies byte JA of IA to J of IX
CCMMPY	(M,N,X11,X12,X21,Y1,Y2,Z1,Z2)	$Z_i = + X'_{ij} Y_j$
CCOPIV	(FR,TO,NCH)	copies CHARACTER FR to TO in reverse order
CCOPYL	(FR,TO,NCH)	copies CHARACTER FR to TO starting at 1
CCOPYR	(FR,TO,NCH)	copies CHARACTER FR to TO starting at NCH
CCOSUB	(FROM,NFR,TO,JL,JR,TOKEN,SUB)	copy with string substitution
CCUMPY	(N,U11,U12,U22,Y1,Y2,Z1,Z2)	$Z_j = \text{sum}(U'_{jk} * Y_k) \quad k=j,n$
CENVIR	(FROM,NFR,TO,JL,JR,IFLAG)	copy with global variable subs.
CEQINV	(N,A,IDIM,IR,IFAIL,K,B)	sets $A=1/A$ , finds $X=B/A$
CEQN	(N,A,IDIM,IR,IFAIL,K,B)	solves $X=B/A$ , A is any non-singular matrix
CFACT	(N,A,IDIM,IR,IFAIL,DET,JFAIL)	set A and R for DFEQN and DFINV
CFCLOS	(LUNDES,MEDIUM)	close disc file
CFEQN	(N,A,IDIM,IR,K,B)	solves $X=B/A$ , A prepared by CFACT
CFFT	(A,MSIGN)	Complex Fast Fourier Transform
CFGET	(LUNDES,MEDIUM,NWREC,NWTAK,MBUF,ISTAT)	get record from file
CFILL	(CHI,LINE,JL,JR)	fills LINE(JL:JR) with copies of CHI
CFINV	(N,A,IDIM,IR)	puts $A=1/A$ (A prepared by CFACT)
CFOPEN	(LUNDES,MED,NWREC,MODE,NBUF,NAME,ISTAT)	open CFIO stream
CFPERM	(IPERM)	set permissions for future disc access
CFPUT	(LUNDES,MEDIUM,NWREC,MBUF,ISTAT)	put record on file
CFREW	(LUNDES,MEDIUM)	rewind disc file
CFROMI	(IPACK)	-> CHARACTER*4
CFSEEK	(LUNDES,MEDIUM,NWREC,NRECC,ISTAT)	set current record number
CFSIZE	(LUNDES,MEDIUM,NWREC,NRECT,ISTAT)	get # records in file
CFTELL	(LUNDES,MEDIUM,NWREC,NRECC,ISTAT)	get current record number
CFWEOF	(LUNDES,MEDIUM,NEOF)	dummy (write NEOF ends-of-file)
CGAMMA	(Z)	COMPLEX gamma function
CHARN	(I)	-> right adjusted decimal ASCII
CHTOI	(CHAR,INTGR,*label)	convert 1 character to integer
CICLOS	(LUNDES)	close disc file
CIGET	(LUNDES,CHBUF,NBDO,NBDONE,ISTAT)	get bytes from disc file
CIGETW	(LUNDES,MBUF,NWDO,NWDONE,ISTAT)	get words from disc file
CINV	(N,A,IDIM,R,IFAIL)	finds $1/A$ , A is any non-singular matrix
CIOPEN	(LUNDES,CHMODE,CHNAME,ISTAT)	open UNIX style stream
CIPERM	(IPERM)	set permissions for future disc access

CIPUT	(LUNDES,CHBUF,NBDO,ISTAT)	send bytes to disc file
CIPUTW	(LUNDES,MBUF,NWDO,ISTAT)	send words to disc file
CIREW	(LUNDES)	rewind disc file
CISEEK	(LUNDES,NBYTC,ISTAT)	set pointer in disc file
CISIZE	(LUNDES,NBYTT,ISTAT)	get size of disc file
CITELL	(LUNDES,NBYTC,ISTAT)	get pointer in disc file
CKRACK	(LINE,JL,JR,IFLD)	reads INTEGER or REAL from LINE(JL:JR)
CLEFT	(LINE,JL,JR)	squeezes blanks in LINE(JL:JR) to right
CLGAMA	(Z)	COMPLEX gamma function
CLOGAM	(Z)	COMPLEX gamma function
CLTOU	(STR)	converts lower case letters in STR to upper case
CMADD	(M,N,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21)	$z = x + y$
CMBIL	(N,V1,V2,X11,X12,X21,Y1,Y2)	$V_k * X_{kj} * Y_j \Rightarrow \text{funct}$
CMCPY	(M,N,X11,X12,X21,Z11,Z12,Z21)	$z = x$
CMDMP	(M,N,D1,D2,X11,X12,X21,Z11,Z12,Z21)	$Z_{ij} = D_i * X_{ij}$
CMMLA	(M,N,K,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21)	$z = xy + z$
CMMLS	(M,N,K,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21)	$z = xy - z$
CMMLT	(M,N,K,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21, T)	$Z = XY$
CMMLTC	(M,N,K,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21, T)	$Z = XY'$
CMMNA	(M,N,X11,X12,X21,Y1,Y2,Z1,Z2)	$Z_i = -X_{ij}Y_j + Z_i$
CMMNS	(M,N,X11,X12,X21,Y1,Y2,Z1,Z2)	$Z_i = -X_{ij}Y_j - Z_i$
CMMPA	(M,N,X11,X12,X21,Y1,Y2,Z1,Z2)	$Z_i = +X_{ij}Y_j + Z_i$
CMMPs	(M,N,X11,X12,X21,Y1,Y2,Z1,Z2)	$Z_i = +X_{ij}Y_j - Z_i$
CMMPY	(M,N,X11,X12,X21,Y1,Y2,Z1,Z2)	$Z_i = +X_{ij}Y_j$
CMMPYC	(M,N,X11,X12,X21,Y1,Y2,Z1,Z2)	$Z_i = +X_{ij}Y'_j$
CMNMA	(M,N,K,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21)	$z = -xy + z$
CMNMS	(M,N,K,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21)	$z = xy - z$
CMRAN	(M,N,A,B,Z11,Z12,Z21)	$Z_{ij} = \text{random in range } [A,B]$
CMsCL	(M,N,S,X11,X12,X21,Z11,Z12,Z21)	$Z_{ij} = S * X_{ij}$
CMSET	(M,N,S,Z11,Z12,Z21)	$z = s$
CMsUB	(M,N,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21)	$z = x - y$
CMUTL	(N,X11,X12,X21)	$X_{jk} = X_{kj} \quad (j > k)$
COMBI	(IA,N,J)	generate possible combinations
CORGEN	(C,X,N)	generates random n-vectors in X according to
CORSET	(V,C,N)	creates C(N,N) from error matrix V(N,N)
CPLNML	(X,N,C,MODE)	make polynomial sum
CRIGHT	(LINE,JL,JR)	squeezes blanks in LINE(JL:JR) to left
CROSS	(A,B,C)	$C = \text{vector } A \times B$
CSETDI	(INT,LINE,JL,JR)	stores integer INT in decimal LINE(JL:JR)
CSETHI	(INT,LINE,JL,JR)	stores integer INT in hex LINE(JL:JR)
CSETOI	(INT,LINE,JL,JR)	stores integer INT in octal LINE(JL:JR)
CSETVI	(NI,INTV,NBIAS,LINE,JL,JR,NCOL,IFLSQ)	vector of integers to characters
CSETVM	(NI,INC ,IGO ,LINE,JL,JR,NCOL,IFLSQ)	generated integers to characters
CSQMBL	(LINE,JL,JR)	squeeze multiple blanks from LINE(JL:JR)
CSQMCH	(CH,LINE,JL,JR)	squeeze multiple characters from LINE(JL:JR)
CTRANS	(CHOLD,CHNEW,LINE,JL,JR)	change CHOLD to CHNEW in LINE(JL:JR)
CUMNA	(N,U11,U12,U22,Y1,Y2,Z1,Z2)	$Z_j = Z_j - \sum(U_{jk} * Y_k) \quad k=j,n$
CUMNS	(N,U11,U12,U22,Y1,Y2,Z1,Z2)	$Z_j = -Z_j - \sum(U_{jk} * Y_k) \quad k=j,n$
CUMPA	(N,U11,U12,U22,Y1,Y2,Z1,Z2)	$Z_j = Z_j + \sum(U_{jk} * Y_k) \quad k=j,n$
CUMPS	(N,U11,U12,U22,Y1,Y2,Z1,Z2)	$Z_j = -Z_j + \sum(U_{jk} * Y_k) \quad k=j,n$
CUMPY	(N,U11,U12,U22,Y1,Y2,Z1,Z2)	$Z_j = \sum(U_{jk} * Y_k) \quad k=j,n$
CUMPYC	(N,U11,U12,U22,Y1,Y2,Z1,Z2)	$Z_j = \sum(U_{jk} * Y'_k) \quad k=j,n$
CUTOL	(STR)	converts upper case letters in STR to lower case
CVADD	(N,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = X_i + Y_i, \quad i=1,N$
CVCPY	(N,X1,X2,Z1,Z2)	$Z_i = X_i, \quad i=1,N$

CVDIV	(N,X1,X2,Y1,Y2,Z1,Z2,IFAIL)	$Z_i = X_i/Y_i, i=1,N$
CVMPA	(N,X1,X2,Y1,Y2,S)	$\Rightarrow S + \sum(X_i, Y_i), i=1,N$
CVMPAC	(N,X1,X2,Y1,Y2,S)	$\Rightarrow S + \sum(X_i, Y_i'), i=1,N$
CVMPY	(N,X1,X2,Y1,Y2)	$\Rightarrow \sum(x_i * y_i) i=1,N$
CVMPYC	(N,X1,X2,Y1,Y2)	$\Rightarrow \sum(x_i * y_i') i=1,N$
CVMUL	(N,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = X_i * Y_i, i=1,N$
CVMULA	(N,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = Z_i + X_i * Y_i, i=1,N$
CVMUNA	(N,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = Z_i - X_i * Y_i, i=1,N$
CVRAN	(N,A,B,Z1,Z2)	$Z_i = \text{random}[A \text{ to } B], i=1,N$
CVSCA	(N,S,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = S * X_i + Y_i, i=1,N$
CVSCL	(N,S,X1,X2,Z1,Z2)	$Z_i = S * X_i, i=1,N$
CVSCS	(N,S,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = S * X_i - Y_i, i=1,N$
CVSET	(N,S,Z1,Z2)	$Z_i = S, i=1,N$
CVSUB	(N,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = X_i - Y_i, i=1,N$
CVSUM	(N,X1,X2)	$\Rightarrow \sum(X_i), i=1,N$
CVXCH	(N,X1,X2,Y1,Y2)	$X_i = Y_i \text{ while } Y_i = X_i, i=1,N$
DADAPT	(F,A,B,NSEG,RELTOL,ABSTOL,RES,ERR)	DP Gaussian quadrature
DATIME	(ID,IT)	get date and time in BCD
DATIMH	(ND,NT)	get date and time in ASCII
DBESIO	(X)	REAL*8 modified Bessel Function I0(X)
DBESI1	(X)	REAL*8 modified Bessel Function I1(X)
DBESJ0	(X)	REAL*8 Bessel function J0(X)
DBESJ1	(X)	REAL*8 Bessel function J1(X)
DBESK0	(X)	REAL*8 modified Bessel Function K0(X)
DBESK1	(X)	REAL*8 modified Bessel Function K1(X)
DBESY0	(X)	REAL*8 Bessel function Y0(X)
DBESY1	(X)	REAL*8 Bessel function Y1(X)
DEBSIO	(X)	$\text{DEXP}(- X ) * \text{DBESIO}(X)$
DEBSI1	(X)	$\text{DEXP}(- X ) * \text{DBESI1}(X)$
DEBSK0	(X)	$\text{DEXP}( X ) * \text{DBESK0}(X)$
DEBSK1	(X)	$\text{DEXP}( X ) * \text{DBESK1}(X)$
DEQINV	(N,A,IDIM,IR,IFAIL,K,B)	sets $A=1/A$ , finds $X=B/A$
DEQN	(N,A,IDIM,IR,IFAIL,K,B)	solves $X=B/A$ , A is any non-singular matrix
DERF	(RX)	$\Rightarrow$ Double Precision Error function
DERFC	(RX)	$\Rightarrow 1 - \text{Double Precision Error function}$
DFACT	(N,A,IDIM,IR,IFAIL,DET,JFAIL)	set A and R for DFEQN and DFINV
DFEQN	(N,A,IDIM,IR,K,B)	solves $X=B/A$ , A prepared by DFACT
DFINV	(N,A,IDIM,IR)	puts $A=1/A$ (A prepared by DFACT)
DFREQ	(RX)	$\Rightarrow$ Double Precision Normal frequency function
DGAMMA	(X)	$\Rightarrow$ Double precision Gamma function
DGAMMF	(X)	$\Rightarrow$ Double precision Gamma function
DGAUSN	(P)	$\Rightarrow$ inverse Gaussian (REAL*8)
DGS56P	(F,A,B,RES,ERR)	Gaussian quadrature with 5&6 points
DINV	(N,A,IDIM,R,IFAIL)	finds $1/A$ , A is any non-singular matrix
DIVDIF	(F,A,NN,X,MM)	tabular interpolation
DLGAMA	(X)	$\Rightarrow \text{DLOG}(\text{Gamma function})$
DLOGAM	(X)	$\Rightarrow \text{DLOG}(\text{Gamma function})$
DMADD	(M,N,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21)	$z = x + y$
DMBIL	(N,V1,V2,X11,X12,X21,Y1,Y2)	$V_k * X_{kj} * Y_j \Rightarrow \text{funct}$
DMCPY	(M,N,X11,X12,X21,Z11,Z12,Z21)	$z = x$
DMDMP	(M,N,D1,D2,X11,X12,X21,Z11,Z12,Z21)	$Z_{ij} = D_i * X_{ij}$
DMINFC	(F,A,B,EPS,DELTA,X,Y,LLM)	finds local minimum of function
DMMLA	(M,N,K,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21)	$z = xy + z$
DMMLS	(M,N,K,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21)	$z = xy - z$
DMMLT	(M,N,K,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21, T)	$Z = XY$
DMMNA	(M,N,X11,X12,X21,Y1,Y2,Z1,Z2)	$Z_i = -X_{ij}Y_j + Z_i$

DMMNS	(M,N,X11,X12,X21,Y1,Y2,Z1,Z2)	$Z_i = -X_{ij}Y_j - Z_i$
DMPMPA	(M,N,X11,X12,X21,Y1,Y2,Z1,Z2)	$Z_i = +X_{ij}Y_j + Z_i$
DMMPSP	(M,N,X11,X12,X21,Y1,Y2,Z1,Z2)	$Z_i = +X_{ij}Y_j - Z_i$
DMMPY	(M,N,X11,X12,X21,Y1,Y2,Z1,Z2)	$Z_i = +X_{ij}Y_j$
DMNMA	(M,N,K,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21)	$z = -xy + z$
DMNMS	(M,N,K,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21)	$z = -xy - z$
DMRAN	(M,N,A,B,Z11,Z12,Z21)	$Z_{ij} = \text{random in range [A,B]}$
DMSCL	(M,N,S,X11,X12,X21,Z11,Z12,Z21)	$Z_{ij} = S * X_{ij}$
DMSET	(M,N,S,Z11,Z12,Z21)	$z = s$
DMSUB	(M,N,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21)	$z = x - y$
DMUTL	(N,X11,X12,X21)	$X_{jk} = X_{kj} \quad (j > k)$
DOTI	(A,B)	4-vector dot product
DPLNML	(X,N,C,MODE)	make polynomial sum
DRKSTP	(N,H,X,Y,SUB,W)	differential equations (Runge-Kutta)
DSEQN	(N,A,IDIM,IFAIL,K,B)	solves $X=B/A$ , A becomes lower triangular
DSFACT	(N,A,IDIM,IFAIL,DET,JFAIL)	form lower triangular matrix
DSFEQN	(N,A,IDIM,K,B)	solves $X=B/A$ (A is lower triangular)
DSFINV	(N,A,IDIM)	solves $A = 1/A$ (A originally lower triangular)
DSINV	(N,A,IDIM,IFAIL)	finds $A=1/A$ (symmetric)
DUMNA	(N,U11,U12,U22,Y1,Y2,Z1,Z2)	$Z_j = Z_j - \sum(U_{jk} * Y_k) \quad k=j,n$
DUMNS	(N,U11,U12,U22,Y1,Y2,Z1,Z2)	$Z_j = -Z_j - \sum(U_{jk} * Y_k) \quad k=j,n$
DUMPA	(N,U11,U12,U22,Y1,Y2,Z1,Z2)	$Z_j = Z_j + \sum(U_{jk} * Y_k) \quad k=j,n$
DUMPS	(N,U11,U12,U22,Y1,Y2,Z1,Z2)	$Z_j = -Z_j + \sum(U_{jk} * Y_k) \quad k=j,n$
DUMPY	(N,U11,U12,U22,Y1,Y2,Z1,Z2)	$Z_j = \sum(U_{jk} * Y_k) \quad k=j,n$
DVADD	(N,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = X_i + Y_i, i=1,N$
DVCPY	(N,X1,X2,Z1,Z2)	$Z_i = X_i, i=1,N$
DVDIV	(N,X1,X2,Y1,Y2,Z1,Z2,IFAIL)	$Z_i = X_i/Y_i, i=1,N$
DVMPA	(N,X1,X2,Y1,Y2,S)	$\Rightarrow S + \sum(X_i, Y_i), i=1,N$
DVMPY	(N,X1,X2,Y1,Y2)	$\Rightarrow \sum(x_i*y_i) \quad i=1,N$
DVMUL	(N,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = X_i * Y_i, i=1,N$
DVMULA	(N,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = Z_i + X_i*Y_i, i=1,N$
DVMUNA	(N,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = Z_i - X_i*Y_i, i=1,N$
DVRAN	(N,A,B,Z1,Z2)	$Z_i = \text{random}[A \text{ to } B], i=1,N$
DVSCA	(N,S,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = S*X_i + Y_i, i=1,N$
DVSCL	(N,S,X1,X2,Z1,Z2)	$Z_i = S*X_i, i=1,N$
DVSCS	(N,S,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = S*X_i - Y_i, i=1,N$
DVSET	(N,S,Z1,Z2)	$Z_i = S, i=1,N$
DVSUB	(N,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = X_i - Y_i, i=1,N$
DVSUM	(N,X1,X2)	$\Rightarrow \sum(X_i), i=1,N$
DVXCH	(N,X1,X2,Y1,Y2)	$X_i = Y_i \text{ while } Y_i = X_i, i=1,N$
DZERO	(A,B,X0,R,EPS,MXF,F)	finds zero of REAL*8 function
EBESIO	(X)	$\text{EXP}(- X ) * \text{BESIO}(X)$
EBESI1	(X)	$\text{EXP}(- X ) * \text{BESI1}(X)$
EBESK0	(X)	$\text{EXP}( X ) * \text{BESK0}(X)$
EBESK1	(X)	$\text{EXP}( X ) * \text{BESK1}(X)$
ERF	(RX)	$\Rightarrow$ Error function
ERFC	(RX)	$\Rightarrow 1 - \text{Error function}$
FFGET	(CHOPT,IVALUE)	interrogate internal parameters of 'ffread'
FFGO	()	reads data cards for 'ffread'
FFINIT	(NW)	initialise 'ffread' with common length NW words
FFKEY	(KEY, ADDRESS, LENGTH, CHTYPE)	defines keywords for 'ffread'
FFSET	(CHOPT,IVALUE)	set optional values for 'ffread'
FFUSER	(KEY)	dummy
FINT	(NARG,ARG,NENT,ENT,TABLE)	multidim. linear interpolation
FLOARG	(WORD)	returns WORG in floating format
FLPSOR	(A,N)	sorts floating array A, length N
FREQ	(RX)	$\Rightarrow$ Normal frequency function

FUNLUX	(ARRAY,XRAN,LEN)	XRAN returns LEN random numbers according to ARRAY
FUNLXP	(F,XFC,X2L,X2H)	prepares user function F for FUNLUX
GAMMA	(X)	=> Gamma function
GAMMF	(X)	=> Gamma function
GATHER	(NW,A,B,INDX)	$A(I) = B(INDX(I))$ , $I=1,NW$
GAUSIN	(P)	=> inverse Gaussian (REAL*4)
GETARG	(IARG,GOTEXT)	returns the command line arguments
GETBIT	(I,M,L)	finds value of a bit in a bit string
GETBYT	(ADDR,IBEG,ILEN,IRES)	extracts byte from bit string
IARGC	()	returns # arguments
ICDECI	(LINE,JL,JR)	returns integer from decimal LINE(JL:JR)
ICEQU	(CHA,CHB,N)	returns comparison of strings CHA and CHB
ICFILA	(CHIS,LINE,JL,JR)	find last occurrence of CHIS in LINE(JL:JR)
ICFIND	(CHIS,LINE,JL,JR)	find first occurrence of CHIS in LINE(JL:JR)
ICFMUL	(CHI,LINE,JL,JR)	finds any character of CHI in LINE(JL:JR)
ICFNBL	(LINE,JL,JR)	finds first non-blank in LINE(JL:JR)
ICHARN	(STRING)	-> integer (converted from decimal ASCII)
ICHEXI	(LINE,JL,JR)	returns integer from hex LINE(JL:JR)
ICINQ	(TEXT,POSS,NPOSS)	searches POSS(NPOSS) for TEXT
ICINQL	(TEXT,POSS,NPOSS)	searches POSS(NPOSS) for TEXT (lower case)
ICINQU	(TEXT,POSS,NPOSS)	searches POSS(NPOSS) for TEXT (upper case)
ICLOC	(CHI,NI,LINE,JL,JR)	is INDEX(LINE(JL:JR),CHI(1:NI))
ICLOCL	(CHI,NI,LINE,JL,JR)	finds CHI(1:NI) in LINE(JL:JR) converted
ICLOCU	(CHI,NI,LINE,JL,JR)	finds CHI(1:NI) in LINE(JL:JR) converted
ICLUNS	(LINE,JL,JR)	finds first non-printing character in LINE(JL:JR)
ICNEXT	(LINE,JL,JR)	finds first non-blank & subsequent blank in LINE
ICNTH	(TEXT,POSS,NPOSS)	searches POSS(NPOSS) for TEXT
ICNTHL	(TEXT,POSS,NPOSS)	searches POSS(NPOSS) (lower case) for TEXT
ICNTHU	(TEXT,POSS,NPOSS)	searches POSS(NPOSS) (upper case) for TEXT
ICNUM	(LINE,JL,JR)	finds first non-numeric character in LINE(JL:JR)
ICNUMA	(LINE,JL,JR)	finds first non-alphanumeric in LINE(JL:JR)
ICNUMU	(LINE,JL,JR)	finds first non-(alphanumeric or _) in LINE
ICOCTI	(LINE,JL,JR)	returns integer from octal LINE(JL:JR)
ICTYPE	(CHIS)	finds the type of a single character
IFROMC	(STRING*4)	-> packed integer
IILZ	(NW,A,INC)	=> # leading zeros in $A(1+I*INC)$ , $I=0,NW-1$
ILSUM	(NW,LA,INC)	=> # .TRUE. elements in $LA(1+I*INC)$ , $I=0,NW-1$
INCBYT	(INC,X,JX,MPACK)	increments a packed histogram bin
INDEXA	(STR)	==> location of first alphabetic character
INDEXB	(STR,SSTR)	==> location of last occurrence of SSTR in STR
INDEXC	(STR,SSTR)	==> first location where SSTR .NE. STR
INDEXN	(STR)	==> location of first numeric digit in STR
INDEXS	(STR)	==> location of first non-alphanumeric on STR
INDXAC	(STR)	==> location of first non-alphabetic character in STR
INDXBC	(STR,SSTR)	==> last location where SSTR .NE. STR
INDXNC	(STR)	==> location of first non-numeric character in STR
INTARG	(WORD)	returns WORD in INTEGER format
INTRAC		returns .TRUE. showing session is interactive
INTSOR	(IA,N)	sorts integer array IA, length N
IRNDM	()	returns random integer from 1 to $2^{**31}-1$
ISCAN	(STR,SET)	==> first location in STR of any character in SET
ISHFTC	(M,K,IC)	returns least sig. IC bits of M rotated left K bits
ITOCH	(INTGR,CHAR,*label)	converts integer to single character
IUBACK	(CH,JL,JR)	reads integer from end CH(JL,JR)
IUBIN	(X,PAR,SPILL)	finds histogram channel
IUCHAN	(X,PAR,SPILL)	finds histogram channel

IUCOLA	(IT,IVEC,N)	finds the last word in IVEC equal to IT
IUCOMH	(CH1,CH2,N)	compares string 1 with string 2
IUCOMP	(IT,IVEC,N)	finds the first word in IVEC equal to IT
IUEND	(NDA)	returns ND in NDA and NE in IUEND
IUFILA	(IT,IVEC,JL,JR)	finds the last word in IVEC(JL,JR) = IT
IUFIND	(IT,IVEC,JL,JR)	finds the first word in IVEC(JL,JR) = IT
IUFNBL	(CH,JL,JR)	finds the first non-blank in CH(JL,JR)
IUFORW	(CH,JL,JR)	reads the integer from beginning of CH(JL,JR)
IUHIST	(X,PAR,SPILL)	finds histogram channel
IUHUNT	(IT,IVEC,N,INC)	finds IT in IVEC(I), I=1,N,INC
IULAST	(IT,IVEC,N)	finds last word in IVEC not equal to IT
IULOOK	(N,CH,JL,JR)	returns first N characters of CH(JL,JR) as nH
IUMODE	(WORD)	returns 0 for integer, nonzero otherwise
IUNEXT	(CH,JL)	returns the first entry in CH(JL...) not blank
IUSAME	(VECT,JL,JR,MIN,JSAME)	finds set of identical words
JBIT	(IW,J)	gets bit J from IW (l.s. is bit 1)
JBYT	(IW,J,NBITS)	gets byte length NBITS from IW (l.s. is bit 1)
JBYTET	(IA,IW,J,NBITS)	=> AND of IA and byte J of IW
JBYTOR	(IA,IW,J,NBITS)	=> OR of IA and byte J of IW
JBYTPK	(MA,JA,MPACK)	=> byte JA of packed vector MA
JRSBYT	(IA,IX,J,NBITS)	=> byte J of IX, replaces byte J by IA
JUMPAD	(TARGET)	return address of EXTERNAL routine
JUMPST	(IAD)	save address of EXTERNAL routine
JUMPX0		transfer control to the routine defined by JUMPST
JUMPX1	(P1)	transfer control with 1 argument
JUMPX2	(P1,P2)	transfer control with 2 arguments
KERNGT	(LUN)	print library version number
LENOCC	(TXT)	faster than LNBLNK for many blanks
LLSQ	(N,X,Y,A0,A1,IFAIL)	least squares fit to a straight line
LNBLNK	(CHV)	find last non-blank in CHV
LOCATD	(D,N,T)	finds where T lies in momotonic real*8 array D
LOCATF	(A,N,T)	finds where T lies in momotonic array A
LOCATI	(IA,N,IT)	finds where IT lies in momotonic array IA
LOCATR	(A,N,T)	finds where T lies in momotonic array A
LOCB	(X)	gets address of X
LOCBYT	(IT,VECT,N,INC,L,NBITS)	=> index of word containing byte
LOCF	(X)	gets 4-byte address of X
LOREN4	(PS,PI,PF)	Lorentz transform PI to PS frame -> PF
LORENB	(U,PS,PI,PF)	Lorentz transform PI from PS frame -> PF
LORENF	(U,PS,PI,PF)	Lorentz transform PI to PS frame -> PF
LSQ	(N,X,Y,M,A)	least squares fit to a polynomial
LVMAX	(A,N)	index of maximum of A()
LVMAXA	(A,N)	index of maximum of  A()
LVMIN	(A,N)	index of minimum of A()
LVMINA	(A,N)	index of minimum of  A()
LVSDMI	(DA,N,INC)	find location of minimum in REAL*8 DA(I) I=1,N*INC
LVSDMX	(DA,N,INC)	find location of maximum in REAL*8 DA(I) I=1,N*INC
LVSIMI	(IA,N,INC)	find location of minimum in IA(I) I=1,N*INC,INC
LVSIMX	(IA,N,INC)	find location of maximum in IA(I) I=1,N*INC,INC
LVSMI	(A,N,INC)	index of minimum of A(I), I=1,N*INC,INC
LVSMX	(A,N,INC)	index of maximum of A(I), I=1,N*INC,INC
MBYTET	(IA,IW,J,NBITS)	=> IW AND IA shifted to byte at J
MBYTOR	(IA,IW,J,NBITS)	=> IW OR IA shifted to byte at J
MCBYT	(IA,JA,IX,J,NBITS)	=> IW with byte at J replaced by JA of IA
MINVAR	(X,Y,R,XEPSI,STEP,MAXFUN,A,B,F)	finds min of function
MSBIT	(IA,IX,J)	returns IX with bit J set to IA (l.s. bit is 1)
MSBIT0	(IX,J)	returns IX with bit J set to 0 (l.s. bit is 1)

MSBIT1	(IX,J)	returns IX with bit J set to 1 (l.s. bit is 1)
MSBYT	(IA,IX,J,NBITS)	returns IX with NBITS at J set to IA (l.s is b
MVBITS	(M,I,LEN,N,J)	moves LEN bits from bit I in M to bit J in N
MXMAD	(A,B,C,NI,NJ,NK)	$C_{ik} = A_{ij} * B_{jk} + C_{ik}$
MXMAD1	(A,Q,C,NI,NJ,NK)	$C_{ik} = A_{ij} * Q'_{kj} + C_{ik}$
MXMAD2	(P,B,C,NI,NJ,NK)	$C_{ik} = P'_{ji} * B_{jk} + C_{ik}$
MXMAD3	(P,Q,C,NI,NJ,NK)	$C_{ik} = P'_{ji} * Q'_{kj} + C_{ik}$
MXMLRT	(A,B,C,NI,NJ)	$C_{il} = A_{ij} * B_{jk} * A'_{kl}$
MXMLTR	(A,B,C,NI,NJ)	$C_{il} = A'_{ij} * B_{jk} * A_{kl}$
MXMPY	(A,B,C,NI,NJ,NK)	$C_{ik} = A_{ij} * B_{jk}$
MXMPY1	(A,Q,C,NI,NJ,NK)	$C_{ik} = A_{ij} * Q'_{kj}$
MXMPY2	(P,B,C,NI,NJ,NK)	$C_{ik} = P'_{ji} * B_{jk}$
MXMPY3	(P,Q,C,NI,NJ,NK)	$C_{ik} = P'_{ji} * Q'_{kj}$
MXMUB	(A,B,C,NI,NJ,NK)	$C_{ik} = A_{ij} * B_{jk} - C_{ik}$
MXMUB1	(A,Q,C,NI,NJ,NK)	$C_{ik} = A_{ij} * Q'_{kj} - C_{ik}$
MXMUB2	(P,B,C,NI,NJ,NK)	$C_{ik} = P'_{ji} * B_{jk} - C_{ik}$
MXMUB3	(P,Q,C,NI,NJ,NK)	$C_{ik} = P'_{ji} * Q'_{kj} - C_{ik}$
MXTRP	(A,B,NI,NJ)	$B_{ji} = A_{ij}$
MXUTY	(A,N)	$A(N \times N) =$ Unity matrix
NCDECI	(CHV)	returns integer from decimal CHV
NCHEXI	(CHV)	returns integer from hex CHV
NCOCTI	(CHV)	returns integer from octal CHV
NUMBIT	(X)	=> number of one-bits in X
PERMU	(IA,N)	form permutaions of IA, length N
PERMUT	(NRP,N,IA)	form permutaions of IA, length N
PKBYT	(IB,X,JX,N,MPACK)	packs N words into packed byte array
PKCHAR	(INT,CHAR,N,IPAR)	packs N words into continuous byte string
POLINT	(F,ARG,M,Z,SUM)	tabular interpolation
PROB	(CHI2,N)	=> Chisquared probability
PROBKL	(X)	=> Kolmogorov probability
PROXIM	(B,A)	returns angle B (+2nPI) nearest to A
PSCALE	(NSC,NMAX,A,NST)	finds power of 10 scale for printing
QNEXT		dummy calls STOP
QNEXT		calls QNEXT always at the same level
RADAPT	(F,A,B,NSEG,RELTOL,ABSTOL,RES,ERR)	REAL Gaussian quadrature
RANECQ	(IS1,IS2,ISQ,CH)	set or retrieve random seeds
RANECU	(R,N,KSEQ)	sets vector of N random numbers of sequence KSEQ
RANLUX	(RVEC,LENV)	returns a vector RVEC(LENV) of random numbers
RANMAR	(RVEC,LENV)	generate LENV random numbers in RVEC
RANNOR	(A,B)	returns random Gaussians
RCHAR	(STRING)	-> REAL from ASCII decimal with optional '.'
RDMIN	(ISEED)	installs new seed (odd integer)
RDMOUT	(ISEED)	returns current seed
REPEAT	(STR,N)	==> string with n copies of STR
REQINV	(N,A,IDIM,IR,IFAIL,K,B)	sets $A=1/A$ , finds $X=B/A$
REQN	(N,A,IDIM,IR,IFAIL,K,B)	solves $X=B/A$ , A is any non-singular matrix
RFACT	(N,A,IDIM,IR,IFAIL,DET,JFAIL)	set A and R for RFEQN and RFINV
RFEQN	(N,A,IDIM,IR,K,B)	solves $X=B/A$ , A prepared by RFACT
RFFT	(A,MSIGN)	real Fast Fourier Transform
RFINV	(N,A,IDIM,IR)	puts $A=1/A$ (A prepared by RFACT)
RGS56P	(F,A,B,RES,ERR)	Gaussian quadrature with 5&6 points
RINV	(N,A,IDIM,R,IFAIL)	finds $1/A$ , A is any non-singular matrix
RKSTP	(N,H,X,Y,SUB,W)	differential equations (Runge-Kutta)
RLUXAT	(LOUT,INOUT,K1,K2)	gets four integer state of RANLUX
RLUXGO	(LUX,INS,K1,K2)	restarts RANLUX from four integer state
RLUXIN	(ISDEXT)	restarts RANLUX from 25-vector ISDEXT

RLUXUT	(ISDEXT)	extracts state of RANLUX as 25-vector ISDEXT
RM48	(DVEC,LENV)	returns a vector DVEC(LENV) REAL*8 random numbers
RM48IN	(I1,N1,N2)	initializes rm48 from initial seed and two counts
RM48UT	(I1,N1,N2)	returns the initial seed and two counts from RM48
RMADD	(M,N,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21)	$z = x + y$
RMARIN	(IJKLIN,NTOTIN,NT02IN)	initialise random numbers for RANMAR
RMARUT	(IJKLUT,NTOTUT,NT02UT)	get state of RANMAR
RMBIL	(N,V1,V2,X11,X12,X21,Y1,Y2)	$V_k * X_{kj} * Y_j \Rightarrow \text{funct}$
RMCPY	(M,N,X11,X12,X21,Z11,Z12,Z21)	$z = x$
RMDMP	(M,N,D1,D2,X11,X12,X21,Z11,Z12,Z21)	$Z_{ij} = D_i * X_{ij}$
RMINFC	(F,A,B,EPS,DELTA,X,Y,LLM)	finds local minimum of function
RMMAQ	(ISEED,ISEQ,CHOPT)	control for RMMAR
RMMAR	(RVEC,LENV,ISEQ)	generate LENV random numbers in RVEC
RMMLA	(M,N,K,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21)	$z = xy + z$
RMMLS	(M,N,K,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21)	$z = xy - z$
RMMLT	(M,N,K,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21,T)	$Z = XY$
RMMNA	(M,N,X11,X12,X21,Y1,Y2,Z1,Z2)	$Z_i = -X_{ij}Y_j + Z_i$
RMMNS	(M,N,X11,X12,X21,Y1,Y2,Z1,Z2)	$Z_i = -X_{ij}Y_j - Z_i$
RMMPA	(M,N,X11,X12,X21,Y1,Y2,Z1,Z2)	$Z_i = +X_{ij}Y_j + Z_i$
RMMPs	(M,N,X11,X12,X21,Y1,Y2,Z1,Z2)	$Z_i = +X_{ij}Y_j - Z_i$
RMMPY	(M,N,X11,X12,X21,Y1,Y2,Z1,Z2)	$Z_i = +X_{ij}Y_j$
RMNMA	(M,N,K,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21)	$z = -xy + z$
RMNMS	(M,N,K,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21)	$z = -xy - z$
RMRAN	(M,N,A,B,Z11,Z12,Z21)	$Z_{ij} = \text{random in range [A,B]}$
RMSCL	(M,N,S,X11,X12,X21,Z11,Z12,Z21)	$Z_{ij} = S * X_{ij}$
RMSET	(M,N,S,Z11,Z12,Z21)	$z = s$
RMSUB	(M,N,X11,X12,X21,Y11,Y12,Y21,Z11,Z12,Z21)	$z = x - y$
RMUTL	(N,X11,X12,X21)	$X_{jk} = X_{kj} \quad (j > k)$
RN2DIM	(X,Y,R)	gets random point (X,Y) on circle radius R
RN3DIM	(X,Y,Z,R)	gets random point (X,Y,Z) on sphere radius R
RNDM	()	returns random real*4 ( $0.0 \leq R < 1.0$ )
RNGAMA	(P)	returns random value according to GAMMA(P)
RNHPRE	(Y,N)	convert Y histogram to normalised cumulative
RNHRAN	(Y,N,XLO,XWID,XRAN)	$XRAN = \text{random value in histogram Y}$
RNORML	(D,N)	generates random normal distribution n-vector
RNORMX	(D,N,RSUB)	generates random normal distribution n-vector
RNPSET	(AMX)	new minimum for RPSSN approximation
RNPSSN	(AMU,N,IERR)	returns random poisson N of mean AMU
ROT	(A,TH,B)	$B = A$ rotated through TH about z-axis
RPLNML	(X,N,C,MODE)	make polynomial sum
RSEQN	(N,A,IDIM,IFAIL,K,B)	solves $X=B/A$ , A becomes lower triangular
RSFACT	(N,A,IDIM,IFAIL,DET,JFAIL)	form lower triangular matrix
RSFEQN	(N,A,IDIM,K,B)	solves $X=B/A$ (A is lower triangular)
RSFINV	(N,A,IDIM)	solves $A = 1/A$ (A originally lower triangular)
RSINV	(N,A,IDIM,IFAIL)	finds $A=1/A$ (symmetric)
RUMNA	(N,U11,U12,U22,Y1,Y2,Z1,Z2)	$Z_j = Z_j - \sum(U_{jk} * Y_k) \quad k=j,n$
RUMNS	(N,U11,U12,U22,Y1,Y2,Z1,Z2)	$Z_j = -Z_j - \sum(U_{jk} * Y_k) \quad k=j,n$
RUMPA	(N,U11,U12,U22,Y1,Y2,Z1,Z2)	$Z_j = Z_j + \sum(U_{jk} * Y_k) \quad k=j,n$
RUMPS	(N,U11,U12,U22,Y1,Y2,Z1,Z2)	$Z_j = -Z_j + \sum(U_{jk} * Y_k) \quad k=j,n$
RUMPY	(N,U11,U12,U22,Y1,Y2,Z1,Z2)	$Z_j = \sum(U_{jk} * Y_k) \quad k=j,n$
RVADD	(N,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = X_i + Y_i, i=1,N$
RVCPY	(N,X1,X2,Z1,Z2)	$Z_i = X_i, i=1,N$
RVDIV	(N,X1,X2,Y1,Y2,Z1,Z2,IFAIL)	$Z_i = X_i/Y_i, i=1,N$
RVMPA	(N,X1,X2,Y1,Y2,S)	$\Rightarrow S + \sum(X_i,Y_i), i=1,N$
RVMPY	(N,X1,X2,Y1,Y2)	$\Rightarrow \sum(x_i*y_i) i=1,N$



RVMUL	(N,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = X_i * Y_i, i=1,N$
RVMULA	(N,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = Z_i + X_i * Y_i, i=1,N$
RVMUNA	(N,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = Z_i - X_i * Y_i, i=1,N$
RVRAN	(N,A,B,Z1,Z2)	$Z_i = \text{random}[A \text{ to } B], i=1,N$
RVSCA	(N,S,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = S * X_i + Y_i, i=1,N$
RVSCL	(N,S,X1,X2,Z1,Z2)	$Z_i = S * X_i, i=1,N$
RVSCS	(N,S,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = S * X_i - Y_i, i=1,N$
RVSET	(N,S,Z1,Z2)	$Z_i = S, i=1,N$
RVSUB	(N,X1,X2,Y1,Y2,Z1,Z2)	$Z_i = X_i - Y_i, i=1,N$
RVSUM	(N,X1,X2)	$\Rightarrow \text{sum}(X_i), i=1,N$
RVXCH	(N,X1,X2,Y1,Y2)	$X_i = Y_i \text{ while } Y_i = X_i, i=1,N$
RZERO	(A,B,X0,R,EPS,MXF,F)	finds zero of REAL*4 function
SBIT	(IA,IX,J)	sets bit J in IX to IA (l.s. bit is 1)
SBIT0	(IX,J)	sets bit J in IX to 0 (l.s. bit is 1)
SBIT1	(IX,J)	sets bit J in IX to 1 (l.s. bit is 1)
SBYT	(IA,IX,J,NBITS)	sets NBITS in IX starting at J to IA (l.s. is bi
SBYTOR	(IA,IX,J,NBITS)	ORs IA into byte J of IX
SBYTPK	(IT,MX,JX,MPACK)	stores IT into byte JX of packed vector MX
SCATTER	(NW,A,INDX,B)	$A(\text{INDX}(I)) = B(I), I=1,NW$
SETBIT	(I,M,L)	sets value of a bit in a bit string
SETBYT	(ADDR,IBEG,ILEN,IBYT)	sets byte in bit string
SORCHA	(A,ICH1,ICH2,N,ITYP)	sorts CHARACTER array A of N elements
SORTD	(MX,NC,NR,NCS)	sort rows of a marix of real*4
SORTI	(MX,NC,NR,NCS)	sort rows of a marix of integers
SORTR	(MX,NC,NR,NCS)	sort rows of a marix of real*4
SORTZV	(A,INDEX,N,MODE,NWAY,NSORT)	sorts 1-d array of any type
SPACES	(STR,NS)	$\Rightarrow$ STR with blank spaces padded with N blanks
STRIP	(STR,CHOPT,CHD)	$\Rightarrow$ STR with leading/trailing CHD removed
STUDIN	(F,N)	$\Rightarrow$ inverse Student t-distribution
STUDIS	(T,N)	$\Rightarrow$ Student t-distribution
SUBWORD	(STR,IW,NW)	$\Rightarrow$ words IW to IW+NW-1 from STR
TCDUMP	(LABEL,IA,NW,IND)	dumps area of memory to stream 6
TIMED	(T)	returns time since last call to TIMED
TIMEL	(T)	execution time remaining
TIMEST		dummy
TIMEX	(T)	execution time so far
TKOLMO	(A,NA,B,NB,PROB)	finds Kolmogorov probablility
TLERR	(A,X,AUX,IPIV)	gets fitted error matrix for l.s.fit
TLRES	(A,B,AUX)	gets fitted residuals from l.s.fit
TLS	(A,B,AUX,IPIV,EPS,X)	unconstrained least squares fitting
TLSC	(A,B,AUX,IPIV,EPS,X)	constrained least squares fit
TLSMSQ	(B,L,M)	(part of TLS)
TLSTEP	(A,B,IASEP,IBSEP,NR,NC,BETA)	(part of TLS)
TLWOP	(A,AD,N,NR)	(part of TLS)
TLUK	(A,IASEP,NR,SIG,BETA)	(part of TLS)
TMINIT		does nothing
TMPRO	(TEXT)	sends prompt to screen
TMREAD	(MAXCH,CHLINE,NCH,ISTAT)	read line from stangard input
TRAAT	(A,S,M,N)	$S(\text{sym } M) = A * A' (M \times N)$
TRACEP	(N,NAME,LB,LC)	name, (beginning) and (call) of Nth traceback
TRACEQ	(LUN,N)	print traceback of depth N
TRAL	(A,V,B,M,N)	$B(M \times N) = A(M \times N) * V' (\text{lower tri } N)$
TRALT	(A,V,B,M,N)	$B(M \times N) = A(M \times N) * V' (\text{lower tri } N)$
TRAS	(A,R,C,M,N)	$C(M \times N) = A(M \times N) * R(\text{sym } N)$
TRASAT	(A,S,R,M,N)	$R(\text{sym } M) = A(M \times N) * S(\text{sym } N) * A'$
TRATA	(A,R,M,N)	$R(\text{sym } M) = A * A' (N \times M)$
TRATS	(B,R,C,M,N)	$C(M \times N) = B' (M \times N) * R(\text{sym } N)$

TRATSA	(B,S,R,M,N)	$R(\text{sym } M) = B' * S(\text{sym } N) * B(N \times M)$
TRCHLU	(A,B,N)	Choleski decompose A(sym) to B'B(lower tri)
TRCHUL	(A,B,N)	Choleski decompose A(sym) to BB'(lower tri)
TRINV	(T,S,N)	invert T(lower tri) to S(lower tri)
TRLA	(W,A,B,M,N)	$B(M \times N) = W(\text{lower tri } M) * A(M \times N)$
TRLTA	(W,A,B,M,N)	$B(M \times N) = W'(\text{lower tri } M) * A(M \times N)$
TRPCK	(A,S,M)	$S(\text{sym}) = A(\text{full})$
TRQSQ	(Q,T,R,M)	$R(\text{sym } M) = QTQ(\text{sym } M)$
TRSA	(S,A,C,M,N)	$C(M \times N) = S(\text{sym } M) * A(M \times N)$
TRSAT	(S,B,C,M,N)	$C(M \times N) = S(\text{sym } M) * B'(M \times N)$
TRSINV	(S,R,N)	inverts S(sym) to R(sym)
TRSM LU	(W,R,N)	$R(\text{sym}) = WW'(\text{lower tri})$
TRSMUL	(W,S,N)	$S(\text{sym}) = W'W(\text{lower tri})$
TRUPCK	(S,A,M)	$A(\text{full}) = S(\text{sym})$
UBITS	(IWORDS,NBITS,IXV,NX)	finds set bits in word or array
UBLANK	(BUF,L1,L2)	$BUF(I) = 4H \quad (I=L1,L2)$
UBLOW	(VM,V1,NCH)	converts 4H array into 1H array
UBUNCH	(V1,VM,NCH)	converts 1H array to 4H array
UCOCOP	(IN,OUT,N,IW,NIN,NOUT)	copies dispersed to compressed vector
UCOPIV	(A,X,N)	invert order of A into X
UCOPY	(IN,OUT,N)	$OUT() = IN()$ where vectors do not overlap
UCOPY2	(IN,OUT,N)	$OUT() = IN()$ where vectors may overlap
UCOPYN	(IA,IX,N)	$IX() = -IA()$
UCTOH	(VC,VJ,NH,NCH)	copy string VC to Hollerith(NH) array VJ
UCTOH1	(VC,V1,NCH)	converts string to 1H format
UDICOP	(IN,OUT,N,IW,NIN,NOUT)	copy compressed to dispersed array
UFILL	(BUF,L1,L2,VAR)	$BUF(I) = VAR \quad (I=L1,L2)$
UFLINT	(VECT,NW,MODE)	assure integer or floating
UH1TOC	(V1,VC,NCH)	converts NCH 1H characters in V1 to string VC
UHOLLR	(IOUT,NC,IHOLL)	copy hollerith string to array
UHTOC	(VI,I,VC,NCH)	converts Hollerith (nl) array VI to string VC
ULEFT	(ch,jl,jr)	left justifies characters in CH(JL,JR)
UOPT	(IACT,IPOSS,IOPT,N)	selects options
UOPTC	(CHACT,CHPOSS,IOPT)	selects options
UPKBYT	(MA,JA,IY,N,MPACK)	unpacks N bytes from packed vector MA
UPKCH	(char,int,n,ipar)	unpacks words from continuous byte string
URIGHT	(CH,JL,JR)	right justifies string CH(JL,JR)
USET	(INT,CH,JL,JR)	writes INT into CH(JL,JR) right justified
USWOP	(A,B,N)	swops arrays A and B
UTRANS	(VI,VJ,NCH,I,J)	converts one format Hollerith array to another
UZERO	(BUF,L1,L2)	$BUF(I) = 0 \quad (I=L1,L2)$
VADD	(A,B,X,N)	$X() = A() + B()$
VASUM	(A,N)	returns sum( A() )
VBIAS	(A,ALPHA,X,N)	$X() = A() + ALPHA$
VBLANK	(BUF,N)	$BUF() = 4H$
VCOPYN	(A,X,N)	$X() = -A()$
VDIST	(A,B,N)	returns vector distance A()-B()
VDIST2	(A,B,N)	returns $(A() - B())^{**2}$
VDOT	(A,B,N)	dot product A().B()
VDOTN	(A,B,N)	Cosine of angle between A() and B()
VDOTN2	(A,B,N)	$(A().B())^{**2} / (A()^{**2} * B()^{**2})$
VERIFY	(STR,SET)	$\Rightarrow$ location (integer!) of first character not in S
VEXCUM	(A,EX,N)	finds minimum, maximum and sum of A()
VFILL	(BUF,N,VAR)	$BUF() = VAR$
VFIX	(A,IX,N)	$IX() = A()$
VFLOAT	(IA,X,N)	$X() = IA()$
VIZPRI	(LUN,TEXT)	sends banner text to stream LUN

VLINCO	(A,F1,B,F2,X,N)	$X() = A()*F1 + B()*F2$
VMATL	(G,C,X,N,M)	$X(j)=C(i)*G(i,j) \quad i=1,M \quad j=1,N$
VMATR	(A,G,V,N,M)	$V(j)=G(j,i)*A(i), \quad i=1,N \quad j=1,M$
VMAX	(A,N)	maximum of A()
VMAXA	(A,N)	maximum of  A()
VMIN	(A,N)	minimum of A()
VMINA	(A,N)	minimum of  A()
VMOD	(A,N)	A()
VMUL	(A,B,X,N)	$X() = A()*B()$
VSCALE	(A,ALPHA,X,N)	$X() = A()*ALPHA$
VSUB	(A,B,X,N)	$X() = A() - B()$
VSUM	(A,N)	sum A()
VUNIT	(A,X,N)	$A() = A()/ A() $
VXINVB	(IW,N)	inverts bytes in N words of IW
VXINVC	(IV,IW,N)	inverts bytes in N words of IV into IW
VZERO	(BUF,N)	BUF() = 0
WGAMMA	(Z)	COMPLEX*16 gamma function
WHNEQ	(NW,IA,INC,IT,INDX,NF)	finds pointers to IA(I)=IT
WHNFGE	(NW,A,INC,T,INDX,NF)	finds pointers to A(I)>=T
WHNFGT	(NW,A,INC,T,INDX,NF)	finds pointers to A(I)>T
WHNFLE	(NW,A,INC,T,INDX,NF)	finds pointers to A(I)<=T
WHNFLT	(NW,A,INC,T,INDX,NF)	finds pointers to A(I)<T
WHNIGE	(NW,IA,INC,IT,INDX,NF)	finds pointers to IA(I)>=IT
WHNIGT	(NW,IA,INC,IT,INDX,NF)	finds pointers to IA(I)>IT
WHNILE	(NW,IA,INC,IT,INDX,NF)	finds pointers to IA(I)<=IT
WHNILT	(NW,IA,INC,IT,INDX,NF)	finds pointers to IA(I)<IT
WHENNE	(NW,A,INC,IT,INDX,NF)	finds pointers to IA(I)<>IT
WLGAMA	(Z)	COMPLEX*16 gamma function
WLOGAM	(Z)	COMPLEX*16 gamma function
WORD	(STR,IW)	==> word IW from STR
WORDS	(STR)	==> # words in STR
WORDSEP	(SEP)	set SEParator
WPLNML	(Z,N,C,MODE)	make polynomial sum
XINB	(LUN,XV,NX)	reads binary of variable length
XINBF	(LUN,XV,NX)	reads binary of fixed length
XINBS	(LUN,XAV,NA,XV,NX)	reads binary in split mode
XOUTB	(LUN,XV,NX)	writes binary of variable length
XOUTBF	(LUN,XV,NX)	writes binary of fixed length
XOUTBS	(LUN,XAV,NA,XV,NX)	writes binary in split mode