LIBRARY

The SPIREL System is composed of the SPIREL operating system plus a library. The library is a collection of named programs and constants which may be utilized by a user system loaded into SPIREL.

The names of all library items are known to the Genie compiler. The user does not have to make any declarations about library items in a Genie program.

The names of library items are not known to the assembly program. The user must use pseudo-orders to establish linkage to library items from APl programs.

Memory space for the full library is required while user programs are being loaded. But then the library may be compressed to just those programs required for support of the user programs, the decision about what is necessary being made automatically. Dynamically, memory space is required for only the essential library items.

- Scalar Routines

These programs are mathematical functions, taking a single floating point argument and giving a floating point result. Detailed explanations are given in the Library Routines section.

All programs are for implicit execution, as discussed in Programmed Use of Library Routines.

NAME	DESCRIPTION
SQR†	square root argument ≥ 0; if argument < 0, gives result = 0 and prints error message
EXP†	exponential |argument| ≤ 170.0; if argument < -170.0, gives result = 0; if argument > 170.0, gives result for argument = 170.0 and prints error message
LOG†	natural logarithm argument > 0; if argument ≤ 0, gives result = argument and prints error message
LOG10	logarithm, base 10 argument > 0; if argument ≤ 0, gives result = 0 and prints error message
SIN†	sine
COS†	cosine
TAN†	tangent
COT†	cotangent
ASIN†	arc sine |result| < π/2, |argument| ≤ 1.0; if |argument| > 1.0, gives result = 0 and prints error message
ATAN†	arc tangent |result| < π/2
SINH†	hyperbolic sine |argument| ≤ 170.0; if |argument| > 170.0, gives result for |argument| = 170.0 and prints error message
COSH†	hyperbolic cosine |argument| ≤ 170.0; if |argument| > 170.0, gives result for |argument| = 170.0 and prints error message
TANH†	hyperbolic tangent |argument| ≤ 170.0; if |argument| > 170.0, gives |result| = 1.0
ASINH†	arc hyperbolic sine
ACOSH†	arc hyperbolic cosine argument ≥ 1.0; if argument < 1.0, gives result = 0 and prints error message

NAME	DESCRIPTION
ATANH†	arc hyperbolic tangent |argument| < 1.0; if |argument| ≥ 1.0, gives result = 0 and prints error message
GAMMA	gamma function -27.0 < argument < 55.0 and argument not a negative integer; if argument out of range or a negative integer, gives result = 0 and prints error message
LGAMMA	log gamma argument ≥ 0; if argument < 0, gives result = 0 and prints error message

†In the Genie language, if argument is complex, corresponding complex routine will be used.

- Complex Scalar Routines

These programs are functions of a single complex argument, giving a real or complex result. Detailed explanations are given in the Library Routines section.

All programs are for implicit execution, as discussed in Programmed Use of Library Routines.

NAME	DESCRIPTION
RE†	real part of a complex number
IM†	imaginary part of a complex number
CARTN†	conversion of complex number from polar to Cartesian form
POLAR†	conversion of complex number from Cartesian to polar form
MOD	modulus of a complex number
CONJ†	conjugate of a complex number
ITIMES†	i times a complex number
CSQR	complex square root
CEXP	complex exponentiation |real part of argument| ≤ 170.0; if real part < -170.0, gives result = 0; if real part > 170.0, gives result for real part = 170.0 and prints error message
CLOG	complex log argument ≠ 0; if argument = 0, gives result = 0 and prints error message
CSIN	complex sine |imaginary part of argument| ≤ 170.0; if |imaginary part| > 170.0, gives result for |imaginary part| = 170.0 and prints error message
CCOS	complex cosine |imaginary part of argument| ≤ 170.0; if |imaginary part| > 170.0, gives result for |imaginary part| = 170.0 and prints error message
CTAN	complex tangent |imaginary part of argument| ≤ 85.0; if |imaginary part| > 85.0, gives result for |imaginary part| = 85.0 and prints error message; if argument near singularity, gives result = tangent of real part of argument and prints error message
CCOT	complex cotangent |imaginary part of argument| ≤ 85.0; if |imaginary part| > 85.0, gives result for |imaginary part| = 85.0 and prints error message

NAME	DESCRIPTION
CASN	complex arc sine
CATN	complex arc tangent argument ≠ ±i; if argument = ±i, gives result = 0 and prints error message
CSNH	complex hyperbolic sine |real part of argument| ≤ 170.0; if |real part| > 170.0, gives result for |real part| = 170.0 and prints error message
CCSH	complex hyperbolic cosine |real part of argument| ≤ 170.0; if |real part| > 170.0, gives result for |real part| = 170.0 and prints error message
CTNH	complex hyperbolic tangent |real part of argument| ≤ 85.0; if |real part| > 85.0, gives result for |real part| = 85.0 and prints error message; if argument near singularity, gives result = tangent of real part and prints error message
CASNH	complex arc hyperbolic sine
CACSH	complex arc hyperbolic cosine
CATNH	complex arc hyperbolic tangent argument ≠ ±1; if argument = ±1, gives result = 0 and prints error message

†In the Genie language, if argument is vector or matrix, corresponding complex matrix routine will be used.

- Matrix Routines

These programs operate on standard vectors and matrices in the STEX domain. Detailed explanations are given in the Library Routines section.

A matrix routine prints an error message if a non-scalar operand does not exist and then performs no operation.

The EXECUTION column below gives the type of routine as explained in Programmed Use of Library Routines.

NAME	DESCRIPTION	EXECUTION
MCOPY	copy of vector or matrix	special
MADD	addition of two vectors or matrices if dimensions not compatible, prints error message	special
MSUB	subtraction of two vectors or matrices if dimensions not compatible, prints error message	special
MMPY	multiplication of two vecotrs, two matrices, or a vector and a matrix -- if dimensions not compatible, prints error message	special
INV†	inverse of matrix if matrix contains more rows than columns or is singular, performs no operation and prints error message	implicit
TRAN†	transpose of matrix	implicit
SMMPY	multiplication of scalar and vector or matrix	special
SMDIV	division of vector or matrix by a scalar if scalar = 0, performs no operation and prints error message	special
MFLT	floating point equivalent of integer matrix	special
MCMPL	complex equivalent of real floating point matrix	special
MINDEX	change initial index and B-mods for a vector or matrix	explicit
MINSERT	insert or delete elements of a vector or rows or columns of a matrix	explicit
MPATCH	move part of one vector or matrix into another vector or matrix	explicit

NAME	DESCRIPTION	EXECUTION
MPOWER	matrix to an integer power if matrix not square and power ≥ 0, prints error message and uses square portion; if matrix not square and power < 0, prints error message and performs no operation; if power < 0 and matrix singular, performs no operation and prints error message	special
DIAG	diagonalization of a matrix, with eigenvectors if desired	explicit
ORTHOG	Gram-Schmidt orthonormalization of the rows of a matrix -- if rows not linearly independent, prints error message and performs no operation	implicit
SOLN†	solution of a system of linear equations if dimensions not proper or solution not defined, prints error message and performs no operation	implicit
DET†	determinant of a matrix if matrix not square, gives result = 0 and prints error message	implicit
STNDV	standard deviation of a vector	implicit
CHISQ	χ^2 for two vectors if vectors not the same length, gives result = 0 and prints error message	implicit
QCONF	χ^2 confidence level between two vectors if vectors not the same length, gives result = 1.0 and prints error message; if lengths < 2, gives result = 0 and prints error message	implicit
CRCOR	vector cross-correlation or auto-correlation if length of result > sum of input lengths - 1 for cross-correlation or > length of input for auto-correlation, prints error message and performs no operation	implicit
CMCON	construct complex vector or matrix if parts do not have same dimensions prints error message and performs no operation	special
FTRAN	Fourier transform of a real vector if arguments are not meaningful, prints error message and performs no operation	implicit

NAME	DESCRIPTION	EXECUTION
ITRAN	inverse Fourier transform of a complex vector -- if arguments are not meaningful, prints error message and performs no operation	implicit
VREV	order reversal of vector elements	explicit
CONVL	convolution of two vectors	implicit

+In the Genie language, if argument is complex, corresponding complex routine will be used.

- Complex Matrix Routines

These programs operate on standard complex vectors and matrices in the STEX domain. Detailed explanations are given in the Library Routines section.

A complex matrix routine prints an error message if a non-scalar operand does not exist and then performs no operation.

The EXECUTION column below gives the type of routine as explained in Programmed Use of Library Routines.

NAME	DESCRIPTION	EXECUTION
CMCPY	copy of a complex vector or matrix	special
CMADD	addition of two complex vectors or matrices -- if dimensions not compatible, prints error message	special
CMSUB	subtraction of two complex vectors or matrices -- if dimensions not compatible, prints error message	special
CMMPY	multiplication of two complex vectors, two complex matrices, or a complex vector and a complex matrix -- if dimensions not compatible, prints error message	special
CINV	inverse of complex matrix if matrix contains more rows than columns or is singular, performs no operation and prints error message	implicit
CTRAN	transpose of complex matrix	implicit
CDET	determinant of complex matrix if matrix not square, gives result = 0 and prints error message	implicit
CSOLN	solution of a system of linear equations with complex coefficients -- if dimensions not proper or solution not defined, prints error message and performs no operation	implicit
CSMMP	multiplication of a complex scalar and a complex vector or matrix	special
CSMDV	division of a complex vector or matrix by a complex scalar -- if scalar = 0, performs no operation and prints error message	special

NAME	DESCRIPTION	EXECTUION
MRE	real part of a complex vector or matrix	implicit
MIM	imaginary part of a complex vector or matrix	implicit
MCARTN	conversion of a complex vector or matrix from polar to Cartesian form	implicit
MPOLAR	conversion of a complex vector or matrix from Cartesian to polar form	implicit
MCONJ	conjugate of a complex vector or matrix	implicit
MITIMES	i times a complex vector or matrix	implicit

- Software Routines

These programs perform miscellaneous operations on data. Detailed explanations are given in the Library Routines section.

The EXECUTION column below gives the type of routine as explained in Programmed Use of Library Routines.

NAME	DESCRIPTION	EXECUTION
LENGTH†	length of a vector	implicit
CLENGTH	length of a complex vector	implicit
ROW†	number of rows in a matrix	implicit
CROW	number of rows in a complex matrix	implicit
COL†	number of columns in a matrix	implicit
CCOL	number of columns in a complex matrix	implicit
MAX	index of maximum element in vector	implicit
MIN	index of minimum element in vector	implicit
VSPACE†	dynamic creation of standard vector of zeroes	explicit
CVSPACE	dynamic creation of a standard complex vector of zeroes	explicit
MSPACE†	dynamic creation of standard matrix of zeroes	explicit
CMSPACE	dynamic creation of a standard complex matrix of zeroes	explicit
MTAKE†	dynamic creation of n-dimensional array of zeroes	explicit
CMTAKE	dynamic creation of n-dimensional complex array of zeroes	explicit
FXEXP	integer or floating point number to an integer power -- if base = 0 and exponent ≤ 0, gives result = 0 and prints error message	special
FLEXP	floating point number to a floating point power -- if base ≤ 0, gives result = 0 and prints error message	special
EVEN	test integer for being even	implicit
FIX	integer nearest to floating point number, if |argument| ≥ 16383.5, gives result = 0 and prints error message	implicit
FLOAT	floating point equivalent of integer	implicit

NAME	DESCRIPTION	EXECUTION
RANDM	floating point random number between 0.0 and 1.0	implicit
CADD	addition of complex scalars	special
CSUB	subtraction of complex scalars	special
CMPY	multiplication of complex scalars	special
CDIV	division of complex scalars	special
CXEXP	complex scalar to an integer power -- if base = 0 and exponent ≤ 0, gives result = 0 and prints error message	special
CFEXP	complex scalar to a real floating point power -- if base = 0, gives result = 0 and prints error message	special
CCEXP	complex scalar to a complex power if base = 0, gives result = 0 and prints error message	special
CONTROL†	application of SPIREL to non-complex item	explicit
CCONTROL	application of SPIREL to complex item	explicit
SCRIBE	formatted line printing	explicit (Genie programs only)
PRESCRIBE	formatted line printing with page control	explicit (Genie programs only)
PLOT	plot on printer of one vector versus another or of a vector versus its indices	explicit

†In the Genie language, if argument is complex, corresponding complex routine will be used.

- Debug Routines

These utility programs are never used in assembly language coding. Detailed explanations are given in the Library Routines section.

NAME	DESCRIPTION	EXECUTION
←COMP	library compression	internal library use only
←INOUT	input/output from compiled programs	used only by Genie-generated code
←ERPR	prints error message	internal library use only
EDIT	communication for compression and system maintenance	console use only
←ENTRY	records information for error print	internal library use only
INPUT	user's input routine	used only by Genie-generated code
OUTPUT	user's output routine	used only by Genie-generated code

- Constants

NAME	DESCRIPTION
LINCT	number of lines used on current page updated by SCRIBE and PRESCRIBE
PAGCT	number of page being printed updated by PRESCRIBE
←TEMP	used for storage control by EDIT only
CMPLX	complex scalar accumulator, double word operand with name "ditto" (←←←←) on second part
←ELOC	information used in printing error message

- Types of Routines

The library routines may be classified by execution characteristics.

Functions are programs which accept arguments by the following rules:

for a single scalar argument, its value in T7 -- may not be output

for a single non-scalar argument, * codeword address in T7 -- may be input, output, or both

for N arguments, N > 1, the address of a scalar and * codeword address of a non-scalar on the B6-list at B6-N,..., B6-1 -- any may be input, output, or both (B6 decremented by N on exit)

A function which provides no output or has one or more output arguments is for explicit execution only.

A function which provides a single output which is not specified as an argument is for implicit execution. The single output is provided as follows:

real scalar in U and T7

complex scalar in the complex scalar accumulator, CMPLX

real non-scalar (vector or matrix) in the real non-scalar accumulator, *10 (octal)

complex non-scalar (vector or matrix) in the complex non-scalar accumulator, CSTAR

Note: Each complex argument is actually two arguments: the real part, then the imaginary part. Thus, for a function with one or more complex arguments, the number of arguments N ≥ 2 always. Two words are used on the B6-list for each complex argument.

Programs which do not accept arguments in T7 or on the B6-list by the above rules are not functions. They require special set-up for execution, e.g., arguments may be given in index registers.

- Use in the Genie Language

In the Genie language only function execution may be specified.

Explicit execution of the function EXPLC with arguments A,B, and C is specified by the command

```
EXECUTE EXPLC(A ,B ,C)
```

Example:

```
EXECUTE VSPACE(V ,K)
```

to execute the program VSPACE for dynamic creation of the vector V of length K.

Implicit execution of the function IMPLC with arguments A,B, and C is specified on the righthand side of an equation, alone or in an expression:

```
... = ...IMPLC(A ,B ,C)...
```

Example:

$P = (SIN(X^2)+SIN(Y^2))/(X-Y)$

involving two executions of the SIN program.

An argument which is input only may be an expression. An output argument must be given as a simple name, scalar (not an element of a vector or matrix) or non-scalar as appropriate.

Scalar input arguments may be specified numerically, e.g.,-5.39. Non-scalar arguments must be specified by name, not by number, e.g., the vector with codeword at +200 (octal) may not be referred to as +200 but must be assigned the value by

```
LET #V = +200
```

and then be referred to as V.

Genie language may cause code to be generated which calls for execution of library routines which are not functions.

Example: For scalar S and matrix M, the command

$Q = S^3M^2$

causes execution of FXEXP, MPOWER, and SMMPY which are library routines for special execution. The command

```
PRINT     S ,M ,Q
```

causes execution of the library routine ←INOU.

- Use in Assembly Language

In either AP1 or AP2 execution of any library routine may be specified by code to set up arguments and then TSR to the program.

In an AP1 program every named item which is referenced, including library routines and constants, must be given a cross-reference word through which it is indirectly addressed. This is accomplished with a REF pseudo-order as explained in the assembly language literature. Cross-reference words are set up automatically in Genie programs for all external named items referenced in the Genie language or AP2, including library routines and constants.

- Compression

The set of library routines may be reduced to just that set necessary for support of any user system. This compression is provided by execution, from the console, of the program EDIT. All memory space occupied by unnecessary programs is made available for user storage.

Library routines have negative Symbol Table indices (...,-2, -1,-0), and their Symbol Table entries bear a tag 1 initially.

Private named items receive positive Symbol Table indices (1,2,...).

Printing or punching ST-VT through SPIREL and execution of the library program EDIT cause context to be determined. This just means that all ST entries with negative indices have tag 1 changed to tag 0 (no tag) if the item represented is required for support of private programs loaded (on the basis of reference only, not dynamic use). Thus, only library items not in context bear tag 1 when a ST-VT print-out is obtained through SPIREL. All private named and numbered items are always in context.

To use EDIT, obtain SPIREL and load private programs and any data which may be located outside the STEX domain. Do not activate STEX or execute any program. Execute the named program EDIT with a control word to SPIREL from the console (manually or off paper tape). Context is determined, and space occupied by all items not in context is freed. All free space is consolidated into a single area. If SL14 is off, a print-out is given of ST-VT entries for items in context.

The "compressed" system may be written on tape for future use or may be used immediately for a run. Activate STEX, if desired, and load items into the STEX domain. Use an execute control word to start the system running.

A "compressed" system (one in which EDIT has been executed) may not be compressed again. This would be meaningless and is impossible since the EDIT routine "erases" itself.

- Diagnostic Procedures

The first version of a program should contain ample print outs that provide display of intermediate results. These may be edited out of the final version of the program or they may be executed conditionally on the basis of sense light settings.

A program should be tested with sense light 14 off. This causes monitoring on the printer of all SPIREL operations, all input-output operations, and all space taking operations. Such information is often a valuable debugging aid.

All the SPIREL diagnostic features available: tracing, arithmetic error monitoring, block bounds checking, diagnostic dump, memory dump. In addition, improper input to many library routines will cause an error message on the printer. Information provided includes the error made, the location of the transfer to the program detecting the error, and the name or codeword address of the calling program.

In this section the library routines are listed in alphabetical order. The details of input, output, and operation are given for each program. The programs ←ERPR and ←ENTRY and the constant ←ELOC may be used by any routine; they are not mentioned as support.

ACOSH, arc hyperbolic cosine

Function: This routine computes the arc hyperbolic cosine of a number.

Execution: implicit

ACOSH(A)

where argument A is floating point scalar input

result is floating point scalar

Errors: If A < 1.0, ACOSH gives result = 0 and prints error message.

Support: programs LOG, SQR

ASIN, arc sine

Function: This routine computes the arc sine of a number.

Execution: implicit

ASIN(A)

where argument A is floating point scalar input

result is floating point scalar, $|ASIN(A)| < \pi/2$

Errors: If $|A| > 1.0$, ASIN gives result = 0 and prints error message.

Support: program SQR

ASINH, arc hyperbolic sine

Function: This routine computes the arc hyperbolic sine of a number.

Execution: implicit

ASINH(A)

where argument A is floating point scalar input

result is floating point scalar

Errors: none

Support: programs LOG, SQR

ATAKE, array take

Function: This routine creates an m-dimensional array, the lowest level of which contains primary codewords addressing n-dimensional arrays of zeroes.

Execution: explicit

ATAKE(A,D_1...D_m,Z,D_1...D_n,N)

where argument A is the real array to be created,
argument D_i is the length in the i^{th} dimension,
argument Z indicates the break between m and n, and
argument N ≤ 6 is the number of dimensions (m+n)+1.

Space formerly addressed as A is freed. An array of size D_1x...xD_m is created, to be indexed by registers B1...Bm. Then arrays of size D_1x...xD_n (with primary codewords at level D_m) are created, to be indexed by registers B1...Bn.

Errors: If N > 6, or if any D_i < 1, or if the Z parameter is missing or improperly located, an error message is printed.

Support: MTAKE

ATAN, arc tangent

Function: This routine computes the arc tangent of a number.

Execution: implicit

ATAN(A)

where argument A is floating point scalar input

result is floating point scalar, $|ATAN(A)| < \pi/2$

Errors: none

Support: none

ATANH, arc hyperbolic tangent

Function: This routine computes the arc hyperbolic tangent of a number.

Execution: implicit

ATANH(A)

where argument A is floating point scalar input

result is floating point scalar

Errors: If $|A| \geq 1.0$, ATANH gives result = 0 and prints error message.

Support: program LOG

CACSH, complex arc hyperbolic cosine

Function: This routine computes the complex arc hyperbolic cosine of a complex number.

Execution: implicit

CACSH(A)

where argument A is complex scalar input

result is complex scalar in CMPLX

In the Genie language, same execution is specified by

ACOSH(A)

for complex argument A.

Errors: none

Support: program CASNH

CADD, complex add

Function: This routine forms the sum of two complex scalars.

Execution: special

input (B1) = address of real part of first operand

(B2) = address of real part of second operand

result in U, R and in complex scalar accumulator, CMPLX.

Complex scalars must occupy consecutive memory locations, real part followed by imaginary part.

Errors: none

Support: scalar CMPLX

CARTN, polar to Cartesian conversion

Function: This routine converts a double word scalar in polar form to a complex scalar in Cartesian form.

Execution: implicit

CARTN(A)

where argument is double word scalar input in polar form, i.e., represented by floating point scalars r and θ such that $A = re^{i\theta}$

result is complex scalar in standard Cartesian form, i.e., represented by floating point scalars x and y such that $A = re^{i\theta} = x + iy$

The polar form of a complex operand is a complex operand in the Genie language, but the arithmetic operations are not defined for this representation; input, output, and storage across an equals are meaningful for the polar form and useful.

Errors: none

Support: program SIN; scalar CMPLX

CASN, complex arc sine

Function: This routine computes the complex arc sine of a complex number.

Execution: implicit

```
CASN(A)
```

where argument A is complex scalar input
result is complex scalar in CMPLX

In the Genie language, same execution is specified by

```
ASIN(A)
```

for complex argument A.

Errors: none

Support: program CASNH; scalar CMPLX

CASNH, complex arc hyperbolic sine

Function: This routine computes the complex arc hyperbolic sine of a complex number.

Execution: implicit

CASNH(A)

where argument A is complex scalar input

result is complex scalar in CMPLX

In the Genie language, same execution is specified by

ASINH(A)

for complex argument A.

Errors: none

Support: programs CADD, CLOG, CMPY, CSQR; scalar CMPLX

CATAKE, complex array take

Function: This routine creates an m dimensional complex array; the lowest level of which contains codewords addressing n-dimensional arrays of zeroes.

Execution: explicit

CATAKE(A,D_1...D_m,Z,D_1...D_n,N)

where argument A is the complex array to be created,
argument D_i is the length in the i^{th} dimension,
argument Z indicates the break between m and n, and
argument N ≤ 6 is the number of dimensions (m+n)+1.

Space formerly addressed as A is freed. A complex array of size D_1x...D_m is created, to be indexed by registers B1...Bm. Then complex arrays of size D_1x...xD_n (with primary codewords at level D_m) are created, to be indexed by registers B1...Bn.

Errors: If n > 6, or any D_i < 1, or if the Z parameter is either missing or improperly located, an error message is printed.

Support: ATAKE,MTAKE

CATN, complex arc tangent

Function: This routine computes the complex arc tangent of a complex number.

Execution: implicit

CATN(A)

where argument A is complex scalar input

result is complex scalar in CMPLX

In the Genie language, same execution is specified by

ATAN(A)

for complex argument A.

Errors: If A = 0±/1.0, CATN gives result = 0 and prints error message.

Support: programs CASN, CATNH; scalar CMPLX

CATNH, complex arc hyperbolic tangent

Function: This routine computes the complex arc hyperbolic tangent of a complex number.

Execution: implicit

CATNH(A)

where argument A is complex scalar input

result is complex scalar in CMPLX

In the Genie language, same execution is specified by

ATANH(A)

for complex argument A.

Errors: If A = ±1.0, CATNH gives result = 0 and prints error message.

Support: programs CDIV, CLOG; scalar CMPLX

CCEXP, complex-complex exponentiation

Function: This routine computes the exponentiation of a complex scalar to a complex power.

Execution: special

input address portion of (U) = address of real part of complex scalar to be raised to power

address portion of (R) = address of real part of complex power

result in U,R and in complex scalar accumulator, CMPLX.

Complex scalars must occupy consecutive memory locations, real part followed by imaginary part.

Errors: If base = 0, CCEXP gives result = 0 and prints error message.

Support: programs CEXP, CLOG, CMPY; scalar CMPLX

CCOL, number of columns in a complex matrix

Function: This routine provides the number of columns in a complex matrix.

Execution: implicit

CCOL(A)

where argument A is complex matrix input

result is the integer number of columns in matrix A.

In the Genie language the same execution is specified by

COL(A)

for complex argument A.

Errors: If A does not exist, result = 0 and an error message is printed.

Support: program COL

CCONTROL, application of SPIREL to complex operand

Function: This routine composes control words and applies SPIREL to the real then the imaginary part of a named complex quantity.

Execution: Explicit

CCONTROL(N,WXYZ,R,CNAME)

where arguments N, WXYZ, and R are control word fields as for CONTROL

argument CNAME is the complex scalar or non-scalar to which the SPIREL operation is to be applied

See description of CONTROL for further details and examples. In the Genie language the same exeuction is specified by

CONTROL(N,WXYZ,R,CNAME)

for complex argument CNAME.

Errors: none

Support: program CONTROL

CCOS, complex cosine

Function: This routine computes the cosine of a complex number.

Execution: implicit

CCOS(A)

where argument A is complex scalar input

result is complex scalar in CMPLX

In the Genie language, same execution is specified by

COS(A)

for complex argument A.

Errors: If |imaginary part of A| > 170.0, CCOS gives result for |IM(A)| = 170.0 and prints error message.

Support: programs SIN, SINH; scalar CMPLX

CCOT, complex cotangent

Function: This routine computes the cotangent of a complex number.

Execution: implicit

CCOT(A)

where argument A is complex scalar input

result is complex scalar in CMPLX

In the Genie language, same execution is specified by

COT(A)

for complex argument A.

Errors: If |imaginary part of A| > 85.0, CCOT gives result for |IM(A)| = 85.0 and prints error message.

Support: programs COT, SIN, SINH; scalar CMPLX

CCSH, complex hyperbolic cosine

Function: This routine computes the complex hyperbolic cosine of a complex number.

Execution: implicit

CCSH(A)

where argument A is complex scalar input

result is complex scalar in CMPLX

In the Genie language, same execution is specified by

COSH(A)

for complex argument A.

Errors: If |real part of A| > 170.0, CCSH gives result for |RE(A)| = 170.0 and prints error message.

Support: programs SIN, SINH; scalar CMPLX

CDET, complex determinant

Function: This routine computes the determinant of a square standard complex matrix in the STEX domain.

Execution: implicit

CDET(A)

where argument A is square complex matrix input

result is complex scalar in CMPLX

SPIREL monitoring for creation of intermediate results is provided if SL14 is off. In the Genie language the same execution is specified by

DET(A)

for complex argument A.

Errors: If A does not exist, CDET prints an error message and performs no operation. If A is not square, CDET gives result = 0 and prints an error message.

Support: program CINV; non-scalar CSTAR

CDIV, complex divide

Function: This routine forms the quotient of two complex scalars.

Execution: special

input (B1) = address of real part of numerator
(B2) = address of real part of denominator

result in U,R and in complex scalar accumulator, CMPLX.

Complex scalars must occupy consecutive memory locations, real part followed by imaginary part.

Errors: none

Support: scalar CMPLX

CEXP, complex exponential

Function: This routine computes the exponential of a complex number.

Execution: implicit

CEXP(A)

where argument A is complex scalar input

result is complex scalar in CMPLX

In the Genie language, same execution is specified by

EXP(A)

for complex argument A.

Errors: If real part of A < -170.0, CEXP gives result = 0; if real part of A > 170.0, CEXP gives result for RE(A) = 170.0 and prints error message.

Support: programs EXP, SIN; scalar CMPLX

CFEXP, complex-floating point exponentiation

Function: This routine computes the exponentiation of a complex scalar to a real floating point power.

Execution: special

input address portion of (U) = address of real part of complex scalar to be raised to power

(R) = floating point scalar power

result in U,R and in complex scalar accumulator, CMPLX

Complex scalars must occupy consecutive memory locations, real part followed by imaginary part.

Errors: If base = 0, CFEXP gives result = 0 and prints error message.

Support: programs CARTN, FLEXP, POLAR; scalar CMPLX

CHISQ, χ^2

Function: This routine computes χ^2, measure of fit, for two floating point vectors of equal length.

Execution: implicit

CHISQ(A,B)

where argument A is the theoretical distribution, real floating point vector input

argument B is the observed distribution, real floating point vector input

result is real scalar, computed as

$$\sum_{i=1}^{n} \left(\frac{(B_i - A_i)^2}{A_i} \right)$$

where vectors are of length n

Errors: If A and B are not equal in length or if either does not exist, CHISQ prints an error message and gives result = 0.

Support: none

CINV, complex matrix inverse

Function: This routine forms the inverse of a standard complex matrix in the STEX domain.

Execution: implicit

CINV(A)

where argument A is standard complex matrix input

result is in complex non-scalar accumulator, CSTAR

Input matrix is not destroyed. SPIREL monitoring for creation of the result is provided if SL14 is off.

The usual application of CINV is to compute the inverse of a square matrix. CINV will work on a matrix containing more columns than rows, say n rows and m columns with m > n. In this case m - n systems of linear equations are represented

$$p^{th}\ \text{system} \begin{cases} A_{1,1}X_1+\ldots+A_{1,n}X_n-A_{1,n+p} = 0 \\ \\ A_{n,1}X_1+\ldots+A_{n,n}X_n-A_{n,n+p} = 0 \end{cases}$$

for p = 1,...,m-n. Column p of the result matrix contains the solution of the p^{th}system:

$$X_1 \text{ in } A_{1,n+p},\ldots,X_n \text{ in } A_{n,n+p}$$

The left square portion (n Columns) of the matrix result is the inverse of the left square portion of the input. In the Genie language the same execution is specified by

INV(A)

for complex argument A.

Errors: If A does not exist or if A contains more rows than columns, CINV prints an error message and performs no operation. If A is singular, no result is given, and CINV prints an error message.

Support: programs CADD, CDIV, CMCPY, CSUB, MOD; scalar CMPLX, non-scalar CSTAR

CLENGTH, length of a complex vector

Function: This routine provides the length of a complex vector.

Execution: implicit

CLENGTH(A)

where argument A is complex vector input

result is integer length of vector A.

In Genie programs the same execution is specified by

LENGTH(A)

for complex argument A.

Error: If A does not exist, result = 0 and an error message is printed.

Support: program LENGTH

CLOG, complex natural logarithm

Function: This routine computes the natural logarithm of a complex number.

Execution: implicit

CLOG(A)

where argument A is complex scalar input

result is complex scalar in CMPLX with imaginary part ≥ 0

In the Genie language, same execution is specified by

LOG(A)

for complex argument A.

Errors: If A = 0 (both real and imaginary parts = 0), CLOG gives result = 0 and prints error message.

Support: programs LOG, POLAR; scalar CMPLX

CMADD, complex matrix add

Function: This routine forms the sum of two standard complex vectors or two standard complex matrices in the STEX domain.

Execution: special

input (B1) = codeword address of real part of first operand

(B2) = codeword address of real part of second operand

result in complex non-scalar accumulator, CSTAR

If either (B1) or (B2) null on entry, the corresponding operand is taken as the complex non-scalar accumulator, CSTAR. An operand which is not CSTAR is not destroyed. The two codewords for a complex non-scalar must occupy consecutive memory locations, real part followed by imaginary part. SPIREL monitoring for creation of the result is provided if neither operand is CSTAR and SL14 is off.

Errors: If either operand does not exist, CMADD prints error message and performs no operation. If dimensions of the two operands are not the same, CMADD uses the subset of the larger which corresponds to the smaller, performs the addition, and prints error message.

Support: programs MADD, MSUB; non-scalar CSTAR

CMCON, complex matrix construction

Function: This routine forms a standard complex vector or matrix from two standard vectors or matrices in the STEX domain. The operands must be of the same dimensions and should contain floating point elements.

Execution: special

input (B1) = codeword address of real part

(B2) = codeword address of imaginary part

result in complex non-scalar accumulator, CSTAR

If either (B1) or (B2) null on entry, the corresponding operand is taken as the non-scalar accumulator, *10. An operand which is not *10 is not destroyed. SPIREL monitoring for creation of the result is provided if SL14 is off.

Errors: If either operand does not exist or if the operands are not of the same dimension, CMCON prints error message and performs no operation.

Support: program MCOPY, non-scalar CSTAR

CMCPY, complex matrix copy

Function: This routine copies a standard vector or matrix in the STEX domain.

Execution: Special

Input: (B1) = codeword address of real part of copy

(B2) = codeword address of real part of vector or matrix to be copied.

If (B1) is null on entry, (B2) is copied into CSTAR, the complex non-scalar accumulator. (B2) is never erased after the copy. If (B1) = (B2), (B2) is assumed to be a duplicate codeword (see SPIREL section on Storage Control). Then an actual copy is made of (B2) and the duplicate backreference is removed. SPIREL monitoring for creation of the copy is provided if SL14 is off.

Errors: If the complex non-scalar (B2) to be copied does not exist, CMCPY prints an error message and performs no operation.

Support: program MCOPY; non-scalar CSTAR.

CMMPY, complex matrix multiply

Function: This routine forms the product of two standard complex vectors (dot product, a scalar), a standard complex vector and a standard complex matrix (a vector), or two standard complex matrices (a matrix). Operands must be in the STEX domain.

Execution: special

input (B1) = codeword address of real part of lefthand operand

(B2) = codeword address of real part of righthand operand

scalar result in CMPLX; non-scalar in CSTAR

If either (B1) or (B2) null on entry, the corresponding operand is taken as the complex non-scalar accumulator, CSTAR. An operand which is not CSTAR is not destroyed. Note that vector × matrix treats the vector as a one-column matrix. The dot product given by vector A × vector B is defined as $\Sigma A_i \bar{B}_i$ where $\bar{B}_i$ is the conjugate of B_i. The two codewords for a complex non-scalar must occupy consecutive memory locations, real part followed by imaginary part. SPIREL monitoring for creation of the result is provided if SL14 is off.

Errors: If either operand does not exist, CMMPY prints error message and performs no operation. If the non-scalar operands do not have dimensions compatible for multiplication, CMMPY uses the subset of the operand with the larger pertinent dimension which corresponds to the operand with the smaller pertinent dimension, performs the multiplication, and prints an error message.

Support: programs MADD, MMPY, MSUB; scalar CMPLX; non-scalar CSTAR

CMPY, complex multiply

Function: This routine forms the product of two complex scalars.

Execution: special

input (B1) = address of real part of first operand
(B2) = address of real part of second operand
result in U,R and in complex scalar accumulator, CMPLX

Complex scalars must occupy consecutive memory locations, real part followed by imaginary part.

Errors: none

Support: scalar CMPLX

CMSPACE, complex matrix space

Function: This routine creates a standard complex matrix of zeroes in the STEX domain.

Execution: explicit

CMSPACE(A,B,C)

where argument A is complex matrix to be created
argument B is integer number of rows in A
argument C is integer number of columns in A

Storage addressed formerly as A is freed; then, if both B > 0 and C > 0, a complex matrix (two matrices with adjacent codewords) with B rows and C columns is created. SPIREL monitoring for freeing or creating is provided if SL14 is off. In the Genie language the same execution is specified by

MSPACE(A,B,C)

for complex argument A.

Errors: none

Support: program MSPACE

CMSUB, complex matrix subtract

Function: This routine forms the difference of two standard complex vectors or two standard complex matrices in the STEX domain.

Execution: special

input (B1) = codeword address of real part of operand to be subtracted from

(B2) = codeword address of real part of operand to be subtracted

result in complex non-scalar accumulator, CSTAR

If either (B1) or (B2) null on entry, the corresponding operand is taken as the complex non-scalar accumulator, CSTAR. An operand which is not CSTAR is not destroyed. The two codewords for a complex non-scalar must occupy consecutive memory locations, real part followed by imaginary part. SPIREL monitoring for creation of the result is provided if neither operand is CSTAR and SL14 is off.

Errors: If either operand does not exist, CMSUB prints error message and performs no operation. If dimensions of the two operands are not the same, CMSUB uses the subset of the larger which corresponds to the smaller, performs the subtraction, and prints error message.

Support: program CMADD

CMTAKE, complex matrix take

Function: This routine creates an n-dimensional complex array of zeroes.

Execution: explicit

CMTAKE(A,D_1,...,D_N,N)

where argument A is complex array to be created
argument D_i is length in the i^{th} dimension
argument $N \leq 5$ is the number of dimensions

Space addressed formerly as A is freed; then an array of size D_1x...xD_N is created, to be indexed by registers B1,...,BN. SPIREL monitoring for creating A is provided if SL14 is off. In the Genie language the same execution is specified by

MTAKE(A,D_1,...,D_N,N)

for complex argument A.

Errors: If any $D_i < 1$, $N < 1$, or $N > 5$, CMTAKE gives no result and prints an error message.

Support: program MTAKE

COL, number of columns in a matrix

Function: This routine provides the number of columns in a matrix.

Execution: implicit

COL(A)

where argument A is a matrix input

result is the integer number of columns in matrix A.

Errors: If A does not exist, result = 0 and an error message is printed.

Support: none

CONJ, conjugate of complex scalar

Function: This routine provides the conjugate of a complex scalar.

Execution: implicit

CONJ(A)

where argument A is complex scalar in standard Cartesian form

result is complex scalar in CMPLX

Errors: none

Support: scalar CMPLX

CONTROL, application of SPIREL

Function: This routine composes a control word and applies SPIREL to a named quantity.

Execution: explicit

CONTROL(N,WXYZ,R,NAME)

where argument N is an integer, the "N" field (first five traids) of the control word to be executed

argument WXYZ is an integer, the "wxyz" field (next four triads) of the control word to be executed, usually given as an octal configuration

argument R is an integer, the "R" field (next four triads) of the control word to be executed

argument NAME is the scalar or non-scalar to which the SPIREL operation is to be applied

If NAME is a scalar, the N field in the control word must be 1, the x triad in the control word must = 0 so that WXYZ would be given as +w0yz in Genie. If NAME is a non-scalar (vector, matrix, or program), the x triad in the control word should = 4 so that WXYZ would be given as +w4yz in Genie.

Examples:

EXECUTE CONTROL(n,+4400,k,BLOCK)

in the Genie language would cause SPIREL to print n words of the vector or program BLOCK in octal, starting at the k^{th}.

EXECUTE CONTROL(0,+5440,0,DATA)

in the Genie language would cause SPIREL to punch all data of the array DATA in hexad with tag and checksum format.

EXECUTE CONTROL(1,+4030,0,RATE)

in the Genie language would cause SPIREL to print the scalar RATE in decimal.

Errors: none

Support: none

CONVL, convolution

Function: This routine computes the convolution of two real floating point vectors.

Execution: implicit

CONVL(A,B)

where argument A is the shorter (filter) input vector
argument B is the longer (data) input vector
result is real floating point vector in the non-scalar accumulator, *10, of length = (length of B)-(length of A)+1

In computing dot products, CONVL does not extend the filter beyond the ends of the data; but "slow start-up" may be obtained by having sufficient zeroes at the ends of the data.

Errors: If A or B does not exist, CONVL prints an error message and performs no operation.

Support: program VREV

COS, cosine

Function: This routine computes the cosine of a number.

Execution: implicit

COS(A)

where argument A is floating point scalar input

result if floating point scalar

Also, (R) = SIN(A) on exit.

Errors: If $|A| \geq 2^{47}$, COS gives result = 0 and prints an error message.

Support: program SIN

COSH, hyperbolic cosine

Function: This routine computes the hyperbolic cosine of a number.

Execution: implicit

COSH(A)

where argument A is floating point scalar input

result is floating point scalar

Also, (R) = SINH(A) on exit.

Errors: If $|A| > 170.0$, COSH gives result for $|A| = 170.0$ and prints error message.

Support: program SINH

COT, cotangent

Function: This routine computes the cotangent of a number.

Execution: implicit

COT(A)

where argument A is floating point scalar input

result is floating point scalar

Error: If |A| = multiple of $3\pi/2$, COT gives result = 0 and prints an error message.

Support: program TAN

CRCOR, cross-correlation or auto-correlation

Function: This routine performs the cross-correlation of two vectors of equal length or auto-correlation on a single vector.

Execution: implicit

CRCOR(A,B,C)

where argument A is real floating point vector input

argument B is real floating point vector input, B≢A for cross-correlation, B≡A for auto-correlation

argument C is integer specifying length of result, ≤ 2(input length)-1 for cross-correlation, ≤ input length for auto-correlation, C≡Z for maximum length result

result is standard floating point vector in the non-scalar accumulator, *10

Errors: If C > 2(input length)-1 for cross-correlation or C > input length for auto-correlation, CRCOR prints an error message and gives maximum length result. If A and B are not equal in length or either does not exist, CRCOR prints an error message and performs no operation.

Support: none

CROW, number of rows in a complex matrix

Function: This routine provides the number of rows in a complex matrix.

Execution: implicit

CROW(A)

where argument A is complex matrix input

result is integer number of rows in matrix A.

In the Genie language the same execution is specified by

ROW(A)

for complex argument A.

Error: If A does not exist, result = 0 and an error message is printed.

Support: program CLENGTH

CSIN, complex sine

Function: This routine computes the sine of a complex number.

Execution: implicit

CSIN(A)

where argument A is complex scalar input

result is complex scalar in CMPLX

In the Genie language, same execution is specified by

SIN(A)

for complex argument A.

Errors: If |imaginary part of A| > 170.0, CSIN gives result for |IM(A)| = 170.0 and prints error message.

Support: programs COS, SINH; scalar CMPLX

CSNH, complex hyperbolic sine

Function: This routine computes the complex hyperbolic sine of a complex number.

Execution: implicit

CSNH(A)

where argument A is complex scalar input

result is complex scalar in CMPLX

In the Genie language, same execution is specified by

SINH(A)

for complex argument A.

Errors: If |real part of A| > 170.0, CSNH gives result for |RE(A)| = 170.0 and prints error message.

Support: programs SIN, SINH; scalar CMPLX

CSMDV, complex scalar-matrix divide

Function: This routine divides a standard complex vector or matrix in the STEX domain by a complex scalar.

Execution: special

input (B1) = codeword address of real part of non-scalar operand
(U) = address of real part of scalar operand

result in complex non-scalar accumulator, CSTAR

If (B1) null on entry, the non-scalar operand is taken as CSTAR. A non-scalar which is not CSTAR is not destroyed. The two words for a complex scalar and the two codewords for a complex non-scalar must occupy consecutive memory locations, real part followed by imaginary part. SPIREL monitoring for creation of the result is provided if SL14 is off.

Errors: If the non-scalar operand does not exist or if the scalar = 0, CSMDV prints error message and performs no operation.

Support: programs CMCPY, CDIV; scalar CMPLX; non-scalar CSTAR

CSMMP, complex scalar-matrix multiply

Function: This routine forms the product of a complex scalar and a standard complex vector or matrix in the STEX domain.

Execution: special

input (B1) = codeword address of real part of non-scalar operand
(U) = address of real part of scalar operand
result in complex non-scalar accumulator, CSTAR

If (B1) null on entry, the non-scalar operand is taken as CSTAR. A non-scalar which is not CSTAR is not destroyed. The two words for a complex non-scalar must occupy consecutive memory locations, real part followed by imaginary part. SPIREL monitoring for creation of the result is provided if SL14 is off.

Errors: If non-scalar operand does not exist, CSMMP prints error message and performs no operation.

Support: programs CMCPY, CMPY; non-scalar CSTAR

CSOLN, complex linear equations solution

Function: This routine provides the solution to a system of linear equations represented by a square standard complex matrix of coefficients and a standard complex vector of constants. The operands must be in the STEX domain.

Execution: implicit

CSOLN(A,B)

where argument A is square complex matrix of coefficients

argument B is complex vector of constants

representing a system of n equations of the form

$$A_{i,1}X_1+\ldots+A_{i,n}X_n = B_i \quad , \quad i = 1,2,\ldots,n$$

result is complex vector in the complex non-scalar accumulator, CSTAR, with the value of X_i as the i^{th} element

SPIREL monitoring for creation of the result is provided if SL14 is off. In the Genie language the same execution is specified by

SOLN(A)

for complex argument A. In assembly language coding the argument B may be a matrix of m columns representing m systems of equations of the form

$$A_{i,1}X_{1,j}+\ldots+A_{i,n}X_{n,j} = B_{i,j} \quad , \quad i = 1,2,\ldots,n \text{ and } j = 1,2,\ldots,m$$

Then the result in CSTAR is a matrix with the solution to the j^{th} system as the j^{th} column

$$X_{1,j}, X_{2,j}, \ldots, X_{n,j}$$

Errors: If A or B does not exist or if dimensions are not proper of if a solution is not defined, CSOLN prints an error message and performs no operation.

Support: program CINV; non-scalar CSTAR

CSQR, complex square root

Function: This routine computes the square root a complex number.

Execution: implicit

CSQR(A)

where argument A is complex scalar input

result is complex scalar in CMPLX with imaginary part ≥ 0

In the Genie language, same execution is specified by

SQR(A)

for complex argument A.

Errors: none

Support: programs CARTN, POLAR, SQR; scalar CMPLX

CSUB, complex subtract

Function: This routine forms the difference of two complex scalars.

Execution: special

input (B1) = address of real part of the operand to be subtracted from

(B2) = address of real part of the operand to be subtracted

result in U,R and in complex scalar accumulator, CMPLX

Complex scalars must occupy consecutive memory locations, real part followed by imaginary part.

Errors: none

Support: scalar CMPLX

CTAN, complex tangent

Function: This routine computes the tangent of a complex number.

Execution: implicit

CTAN(A)

where argument A is complex scalar input

result is complex scalar in CMPLX

In the Genie language, same exeuction is specified by

TAN(A)

for complex argument A.

Errors: If |imaginary part of A| > 85.0, CTAN gives result for |IM(A)| = 85.0 and prints error message. If A is near singularity of complex tangent, CTAN gives result = tangent of real part of A and prints error message.

Support: programs SIN, SINH, TAN; scalar CMPLX

CTNH, complex hyperbolic tangent

Function: This routine computes the complex hyperbolic tangent of a complex number.

Execution: implicit

CTNH(A)

where argument A is complex scalar input

result is complex scalar in CMPLX

In the Genie language, same execution is specified by

TANH(A)

for complex argument A.

Errors: If |real part of A| > 85.0, CTNH gives result for |RE(A)| = 85.0 and prints error message. If A is near singularity, CTNH gives result = tangent of real part of A and prints error message.

Support: programs CASN, CTAN; scalar CMPLX

CTRAN, complex matrix transpose

Function: This routine forms the transpose of a standard complex matrix in the STEX domain.

Execution: implicit

CTRAN(A)

where argument A is standard complex matrix input

result is in complex non-scalar accumulator, CSTAR

Note that CTRAN forms the matrix B, transpose of A, such that $B_{i,j}=A_{j,i}$. SPIREL monitoring for creation of the result is provided if SL14 is off. In the Genie language the same execution is specified by

TRAN(A)

for complex argument A.

Errors: If A does not exist, CTRAN prints an error message and performs no operation.

Support: programs CMCPY, TRAN; non-scalar CSTAR

CVSPACE, complex vector space

Function: This routine creates a standard complex vector of zeroes in the STEX domain.

Execution: explicit

CVSPACE(A,B)

where argument A is complex vector to be created
argument B is integer length of A

Storage addressed formerly as A is freed; then, if B > 0, a complex vector (two vectors with adjacent codewords) of length B is created. SPIREL monitoring for freeing or creation of A is provided if SL14 is off. In the Genie language the same execution is specified by

VSPACE (A,B)

for complex argument A.

Errors: none

Support: program VSPACE

CXEXP, complex-fixed point exponentiation

Function: This routine computes the exponentiation of a complex scalar to an integer power.

Execution: special

input address portion of (U) = address of real part of complex scalar to be raised to power

(R) = integer power

result in U,R and in complex scalar accumulator, CMPLX, zero if base = 0 and input (R) > 0

Complex scalars must occupy consecutive memory locations, real part followed by imaginary part.

Errors: If base = 0 and input (R) ≤ 0, CXEXP gives result = 0 and prints error message.

Support: programs CARTN, FXEXP, POLAR; scalar CMPLX

DET, determinant

Function: This routine computes the determinant of a square standard matrix of floating point type.

Execution: implicit

DET(A)

where argument A is square floating point matrix input

result is floating point scalar

A is destroyed only if it is the non-scalar accumulator, *10. In any case *10 is freed. SPIREL monitoring is provided for creation of an intermediate non-scalar if SL14 is off.

Errors: If A does not exist, DET prints an error message and performs no operation. If A is not square, DET gives result = 0 and prints an error message.

Support: program INV

DIAG, matrix diagonalization

Function: This routine diagonalizes a symmetric matrix of floating point type in the STEX domain. The initial indices of the matrix must be one. It also provides eigenvectors if desired.

Execution: explicit

DIAG(A,B,C)

where argument A is the matrix to be diagonalized and will contain the result

argument B is the matrix to contain eigenvectors as rows, null if no eigenvectors desired

argument C floating point scalar to be used as upper bound on off-diagonal elements of diagonalized matrix, null for upper bound as 2^{-48} times smallest diagonal element in result

The input matrix may be stored in upper triangular form, initial row indices 1,2,...,n for an n X n matrix

Errors: none

Support: program SQR

EDIT, library edit

Function: This routine performs library maintenance operations and is executed for library compression.

Execution: from the console only -- from word 1 for compression (see RUNNING section), from word 2 for punching (see MAINTENANCE section), from word 3 for initialization only (see MAINTENANCE section).

Errors: none

Support: program ←COMP

EVEN, test integer even

Function: This routine tests an integer for being even.

Execution: implicit

EVEN(A)

where argument A is integer input

result is Boolean value TRUE (represented by integer -0) if A is even, Boolean value FALSE (represented by integer -1) if A is odd

Errors: none

Support: none

EXP, exponential

Function: This routine computes the exponential of a number.

Execution: implicit

EXP(A)

where argument A is floating point scalar input

result is floating point scalar

Errors: If A < -170.0, EXP gives result = 0. If A > 170.0, EXP gives result for A = 170.0 and prints error message.

Support: none

FFT, fast Fourier transform

Function: This program does a discrete Fourier transform or inverse, as directed by the parameters; it is also used by FFTC.

$$A'_j = \frac{1}{B} \sum_{k=0}^{N-1} A_k e^{-i2\pi jk/N}$$ (exponent sign is + for an inverse)

Execution: explicit

FFT(A,B,C)

where argument A is the input/output vector (complex)

B is the real scale constant

C is a Boolean variable: true if the sign of the exponent is negative, false otherwise.

result is stored in A since the computation is done in place. FFT is most efficient for highly composite N, that is, if $N = n_1 \cdot n_2 \cdot n_3 \cdot \ldots \cdot n_m$. If N is prime the running time is on the order of N^2, otherwise it is on the order of

$$N\left(\sum_{i=1}^{m} n_i\right).$$

Errors: if A is non-existent, or the values require too much scratch storage, an error message is printed.

Support: program COS.

FFTC, fast Fourier transform control program

Function: This program causes a discrete Fourier transform or inverse to be done on an original input vector, as directed by the combination of input parameters. See FFT and RTRAN.

Execution: explicit

FFTC(A,B,C,D,E)

where argument A is the input vector
B is the output vector
C is the real scale constant
D is a Boolean variable: true for a transform, false for an inverse
E is a Boolean variable: true if the complex vector is conjugate symmetric (i.e., only the right half of the vector is supplied); false, if not.

The Fourier transform requires a complex input; therefore if input A is real, FFTC makes it complex under the following conditions:

1. if a) input A has odd length N, and output B is specified as symmetric,
   or b) output B is specified as non-symmetric (parameter E false),

   FFTC makes A complex by creating an imaginary part: a vector of zeroes N long. The output is complex vector B, each part of which is N long.
2. If input A has even length N and B is specified as symmetric (E true), FFTC saves time by creating a complex vector from A: the real part is the odd elements of A and the imaginary part is the even elements. This complex vector is N/2+1 long (the +1 being a zero added by the program to provide necessary working space for RTRAN). FFT is then entered at the third instruction, causing the transform to

be done on N/2 elements. RTRAN is done next, on N/2+1 elements and the result is complex vector B (each part of which is N/2+1 long).

If the input A is already complex, FFTC does the following:

1. if A is N long and not symmetric, the transform or inverse is done directly and the output is a complex vector N long (the input and output vectors may both be A if the input doesn't need to be saved).
2. If A is symmetric, FFTC considers its length N to represent one half of the vector plus 1 (the mid-point), and the real output B will be 2(N-1) long. (This case is the reverse of 2 above, i.e., it does RTRAN first, followed by FFT entered at the third instruction. By definition then, this case is an inverse and parameter D must be false).

The scale constant for a transform is usually 1.0; for an inverse 1.0/length N. If the input and output vectors are different lengths, use the length of the one which is real.

Errors: The following combinations of parameters produce error messages:

1. A real, B real, C,D,E
2. A complex, B complex, C,D,E true
3. A complex, B real, C,D,E false

Support: programs FFT, RTRAN, MCOPY, CMCPY; non-scalars ←CSTAR, USTAR, ←DUMY.

FIX, convert to integer

Function: This routine computes the integer closest to a floating point scalar.

Execution: implicit

FIX(A)

where argument A is floating point scalar input

result is integer closest to A, rounded up in absolute value

Errors: If $|A| \geq 16383.5$, FIX gives result = 0 and prints error message.

Support: none

FLEXP, floating point exponentiation

Function: This routine computes exponentiation of a floating point number to a floating point power.

Execution: special

input (U) = floating point scalar to be raised to power

(R) = floating point scalar power

result in U and T7 which is $(U)^{(R)}$

Errors: If input (U) $\leq$ 0, FLEXP gives result = 0 and prints error message.

Support: programs EXP, LOG

FLOAT, convert to floating point

Function: This routine provides the floating point equivalent of an integer.

Execution: implicit

FLOAT(A)

where argument A is scalar input, integer or floating point
result is floating point equivalent of A, just A if A is floating point

Errors: none

Support: none

FTRAN, Fourier transform

Function: This routine computes the complex frequency (Hz) spectrum of a real function of time. The discrete time series must be stored in a floating point vector in the STEX domain, and the values must be equally spaced in time.

Execution: implicit

FTRAN(A,B,C,D,E,F)

where argument A is the real time series vector input,
argument B is the upper time limit, floating point scalar
argument C is the lower time limit, floating point scalar
argument D is the upper frequency limit, floating point scalar
argument E is the lower frequency limit, floating point scalar
argument F is the frequency increment, floating point scalar.
result is complex vector in CSTAR, the complex non-scalar accumulator

The Fourier integral is approximated by the trapezoidal quadrature formula. SPIREL monitoring for creation of the result is provided if SL14 is off.

Errors: If A does not exist or B-C $\leq$ 0 or D-E < 0, FTRAN prints an error message and performs no operation.

Support: programs CVSPACE, SIN; non-scalar CSTAR.

FXEXP, fixed point exponentiation

Function: This routine computes exponentiation of a number to an integer power.

Execution: special

input (U) = floating point or integer scalar to be raised to power

(R) = integer power

result in U and T7 which is $(U)^{(R)}$ and of same type as input (U), zero if input (U) is integer and $|(U)| > 1$ and input (R) < 0, zero if input (U) = 0 and input (R) > 0

Errors: If input (U) = 0 and input (R) ≤ 0, FXEXP gives result = 0 and prints error message.

Support: none

GAMMA, gamma function

Function: This routine computes the gamma function of a real floating point number.

Execution: implicit

GAMMA(A)

where argument A is real floating scalar input

result is real floating point scalar computed by Stirling's logarithmic approximation

Errors: If A<-27, A>55.0, or A is a negative integer, GAMMA gives result=0 and prints an error message.

Support: programs EXP, LOG

IM, imaginary part

Function: This routine provides the imaginary part of a complex scalar.

Execution: implicit

IM(A)

where argument A is complex scalar input
result is real floating point scalar

If coded in the Genie language, the library routine is not used; but the routine may be used in assembly language coding. IM may be used on any double word scalar argument to provide the second part as a single word scalar.

Errors: none

Support: none

INPUT, special input

Function: This routine is supplied by the user for special input to programs written in the Genie language.

Execution: in Genie language only, by the command

INPUT list

where the program INPUT is entered once for each named variable in the list. A complex variable is treated as two items, the real part with the name of the variable and the imaginary part with the name "ditto". The program INPUT must be coded in the assembly language, AP1. Information is given in T7 on entry to INPUT as follows:

bits 1-30 name in BCD as given in list
31-33 not used
34-36 octal 0 for scalar
2 for vector
4 for matrix
39-41 not used
40-54 address of scalar, codeword address for non-scalar

INV, matrix inverse

Function: This routine forms the inverse of a standard matrix in the STEX domain. The matrix must be of floating point type.

Execution: implicit

INV(A)

where argument A is floating point standard matrix input

result is floating point standard matrix in the non-scalar accumulator, *10

Input matrix which is not *10 is not destroyed. SPIREL monitoring for creation of the result is provided if SL14 is off. The determinant of A is given in T7 on exit, floating point scalar.

The usual application of INV is to compute the inverse of a square matrix. INV will work on a matrix containing more columns than rows, say n rows and m columns with m > n. In this case m - n systems of linear equations are represented

$$p^{th} \text{ system } \begin{cases} A_{1,1}X_1+\ldots+A_{1,n}X_n-A_{1,n+p} = 0 \\ \\ A_{n,1}X_1+\ldots+A_{n,n}X_n-A_{n,n+p} = 0 \end{cases}$$

for p = 1,...,m-n. Column p of the result matrix contains the solution of the p^{th} system:

$$X_1 \text{ in } A_{1,n+p},\ldots,X_n \text{ in } A_{n,n+p}$$

The left square portion (n columns) of the matrix result is the inverse of the left square portion of the input, and the determinant computed is that of the left square portion of the input.

Errors: If A does not exist or if A contains more rows than columns, INV prints an error message and performs no operation. If A is singular, no result is given, and INV prints an error message.

Support: program MCOPY

ITIMES, i times complex scalar

Function: This routine computes i times a complex scalar.

Execution: implicit

ITIMES(A)

where argument A is complex scalar, x+iy

result is complex scalar in CMPLX, -y+ix

Errors: none

Support: scalar CMPLX

ITRAN, Inverse Fourier Transform

Function: This routine computes the real time spectrum of a complex function of frequency (HZ), where $F(HZ)=\overline{F(-HZ)}$. The positive portion of the frequency domain must be stored in a complex, floating point vector in the STEX domain.

Execution: implicit

ITRAN(A,B,C,D,E,F)

where argument A is the complex frequency vector input,
argument B is the upper frequency limit, floating point scalar,
argument C is the lower frequency limit, floating point scalar,
argument D is the upper time limit, floating point scalar,
argument E is the lower time limit, floating point scalar,
argument F is the time increment, floating point scalar,
result is real vector in the real non-scalar accumulator, *10.

The Fourier integral is approximated by the trapezoidal quadrature formula. SPIREL monitoring for creation of the result is provided if SL14 is off.

Errors: If A does not exist or B-C≤0 or D-E<0, ITRAN prints an error message and performs no operation.

Support: programs SIN, VSPACE.

LENGTH, length of vector

Function: This routine provides the length of a vector.

Execution: implicit

LENGTH(A)

where argument A is vector input

result is integer length of vector A.

Errors: If A does not exist, result = 0 and an error message is printed.

Support: none

LGAMMA, log gamma

Function: This routine computes the logarithm of the gamma function of a non-negative real floating point number.

Execution: implicit

LGAMMA(A)

where argument A is real floating point scalar input

result is real floating point scalar

Error: If A<0, LGAMMA gives result=0 and prints an error message.

Support: programs GAMMA, LOG

LOG, natural logarithm

Function: This routine computes the natural logarithm of a number.

Execution: implicit

LOG(A)

where argument A is floating point scalar input

result is floating point scalar

Errors: If A ≤ 0, LOG gives result = argument and prints error message.

Support: none

LOG10, logarithm, base 10

Function: This routine computes the common logarithm of a number.

Execution: implicit

LOG10(A)

where argument A is floating point scalar input

result is floating point scalar

Errors: If A ≤ 0, LOG10 gives result = 0 and prints error message.

Support: program LOG

MADD, matrix add

Function: This routine forms the sum of two standard vectors or two standard matrices in the STEX domain. The operands must agree in type, floating point or integer, and the result will be of the same type.

Execution: special

input (B1) = codeword address for first operand
(B2) = codeword address for second operand

result in non-scalar accumulator, *10

If either (B1) or (B2) null on entry, the corresponding operand is taken as the non-scalar accumulator, *10. An operand which is not *10 is not destroyed. SPIREL monitoring for creation of the result is provided if neither operand is *10 and SL14 is off.

Errors: If either operand does not exist, MADD prints error message and performs no operation. If dimensions of the two operands are not the same, MADD uses the subset of the larger which corresponds to the smaller, performs the addition, and prints error message.

Support: none

MAX, vector maximum

Function: This routine computes the index of the element with the largest numeric value in a vector of floating point numbers.

Execution: implicit

MAX(A)

where argument A is floating point vector input

result is integer

Errors: If A does not exist, MAX prints an error message and gives result=0.

Support: none

MCARTN, matrix polar to Cartesian conversion

Function: This routine converts a double word vector or matrix in the STEX domain in polar form to a complex vector or matrix in Cartesian form.

Execution: implicit

MCARTN(A)

where argument is double word vector or matrix input in polar form, i.e., each element represented by floating point scalars r and θ such that the element = $re^{i\theta}$

result is complex vector or matrix in CSTAR in standard Cartesian form, i.e., each element represented by floating point scalars x and y such that the element = $re^{i\theta}$ = x+iy

SPIREL monitoring for creation of the result is given if SL14 is off. The polar form of a complex operand is a complex operand in the Genie language, but the arithmetic operations are not defined for this representation; input, output, and storage across an equals are meaningful for the polar form and useful. In the Genie language same execution is specified by

CARTN(A)

for non-scalar argument.

Errors: If A does not exist, MCARTN prints an error message and performs no operation A.

Support: programs CARTN, CMCPY, POLAR; scalar CMPLX; non-scalar CSTAR

MCMPL, matrix complex

Function: This routine provides the complex equivalent of a real floating point vector or matrix in the STEX domain.

Execution: special

input (B1)=codeword address for operand

result in complex non-scalar accumulator, CSTAR

If (B1) null on entry, the operand is taken as the real non-scalar accumulator, *10.

Errors: If operand does not exist, MCMPL prints error message and performs no operation.

Support: program MCOPY; non-scalar CSTAR

MCONJ, matrix conjugate

Function: This routine provides the conjugate of a complex vector or matrix.

Execution: implicit

MCONJ(A)

where argument A is complex vector or matrix input
result is complex vector or matrix in CSTAR, each element being the conjugate of the corresponding element of A
SPIREL monitoring for creation of the result is given if SL14 is off. In the Genie language same execution is specified by

CONJ(A)

for non-scalar argument A.

Errors: If A does not exist, MCONJ prints an error message and performs no operation.

Support: program CMCPY; non-scalar CSTAR

MCOPY, matrix copy

Function: This routine copies a standard vector or matrix in the STEX domain.

Execution: Special

Input: (B1) = codeword address for copy,
(B2) = codeword address of vector or matrix to be copied.

If (B1) is null on entry, (B2) is copied into *10, the non-scalar accumulator. (B2) is never erased after the copy. If (B1) = (B2), (B2) is assumed to be a duplicate codeword (see SPIREL section on Storage Control). Then an actual copy is made of (B2) and the duplicate backreference is removed. SPIREL monitoring for creation of the copy is provided if SL14 is off.

Errors: If the non-scalar (B2) to be copied does not exist, MCOPY prints an error message and performs no operation.

Support: programs MSPACE and VSPACE.

MEAN, Average Values

Function: This routine computes the mean of a standard vector.

Execution: implicit

MEAN(A)

where argument A is a floating point vector

result is a floating point scalar

Errors: If A does not exist, MEAN gives result = 0 and prints an error message.

Support: none

MFLT, matrix float

Function: This routine provides the floating point equivalent of an integer vector or matrix in the STEX domain.

Execution: special

input (B1)=codeword address for operand

result in non-scalar accumulator, *10

If (B1) null on entry, the operand is taken as the non-scalar accumulator, *10.

Errors: If operand does not exist, MFLT prints error message and performs no operation.

Support: program MCOPY

MIM, matrix imaginary part

Function: This routine provides the imaginary part of a complex vector or matrix.

Execution: implicit

MIM(A)

where argument A is complex non-scalar input

result is in non-scalar accumulator, *10

MIM may be used on any double word non-scalar argument to provide the second part as a single word non-scalar. SPIREL monitoring for creation of the result is provided if SL14 is off. In the Genie language the same exeuction is specified by

IM(A)

for non-scalar argument A.

Errors: If A does not exist, MIM prints error message and performs no operation.

Support: program MCOPY

MIN, vector minimum

Function: This routine computes the index of the element with the smallest numeric value in a vector of floating point numbers.

Execution: implicit

MIN(A)

where argument A is floating point vector input

result is integer

Errors: If A does not exist, MIN prints an error message and gives result=0.

Support: program MAX

MINDEX, matrix index

Function: This routine changes the initial indices and B-mods for a vector or matrix in the STEX domain.

Execution: explicit

MINDEX(i,b,V) for vector

where argument i is integer, initial index (for i = zero, use -Z)

argument b is integer (=1,2,...,7) for B-mod or zero to not change B-mod

argument V is vector operand

If both i and b are zero, the vector V is changed to standard form (initial index = 1 and B1-mod).

MINDEX(i_r,b_r,i_c,b_c,M) for matrix

where arguments i_r and i_c are integers, row and column initial indices respectively

arguments b_r and b_c are integers (=1,2,...,7) for row and column B-mods respectively or zero to not change B-mod

argument M is matrix operand

If arguments i_r,b_r,i_c, and b_c are zero, the matrix M is changed to standard form (initial indices = 1 and B1-mod for rows, B2-mod for columns).

Errors: If operand does not exist, MINDEX prints error message and performs no operation.

Support: none

MINSERT, matrix insert

Function: This routine inserts or deletes elements of a vector in the STEX domain or rows or columns of a matrix in the STEX domain.

Execution: explicit

MINSERT(n,r,V) for vector

where argument n is integer, number of elements to insert as zeroes if > 0, number of elements to delete if < 0

argument r is integer, index of first element inserted or deleted

argument V is vector operand

MINSERT(n,r,k,M) for matrix

where argument n is integer, number of rows or columns to insert as zeroes if > 0, number of rows or columns to delete if < 0

argument r is integer, index of first row or column inserted or deleted

argument k is integer, k = 1 to operate on rows, k = 2 to operate on columns

If argument n is zero, MINSERT deletes element, row, or column r and all following. If argument r is null, MINSERT inserts n elements, rows, or columns after the last.

Errors: none

Support: none

MITIMES, i times complex matrix

Function: This routine computes i times a complex vector or matrix.

Execution: implicit

MITIMES(A)

where argument A is complex vector or matrix input
result is complex vector or matrix in CSTAR, each element being i times the corresponding element of A

SPIREL monitoring for creation of the result is given if SL14 is off. In the Genie languate same execution is specified by

ITIMES(A)

for non-scalar argument A.

Errors: If A does not exist, MITIMES prints an error message and performs no operation

Support: program MCONJ; non-scalar SCTAR

MMPY, matrix multiply

Function: This routine forms the product of two standard vectors (dot product, a scalar), a standard vector and a standard matrix (a vector), or two standard matrices (a matrix). Operands must be in the STEX domain; they must agree in type, floating point or integer, and the result will be of the same type.

Execution: special

input (B1) = codeword address for lefthand operand

(B2) = codeword address for righthand operand

scalar result in U and T7; non-scalar in accumulator, *10

If either (B1) or (B2) null on entry, the corresponding operand is taken as *10. An operand which is not *10 is not destroyed. Note that vector X matrix treats the vector as a one-row matrix, and matrix X vector treats the vector as a one-column matrix. SPIREL monitoring for creation of the result is provided if SL14 is off.

Errors: If either operand does not exist, MMPY prints error message and performs no operation. If the non-scalar operands do not have dimensions compatible for multiplication, MMPY uses the subset of the operand with the larger pertinent dimension which corresponds appropriately to the operand with the smaller pertinent dimension, performs the multiplication, and prints an error message.

Support: none

MOD, modulus of complex scalar

Function: This routine computes the modulus of a complex scalar.

Execution: implicit

MOD(A)

where argument A is complex scalar input

result is real floating point scalar

Errors: none

Support: program SQR

MODUL, Compute Remainder

Function: This routine computes A modulo B

Execution: implicit

MODUL(A,B)

where A and B are integers.

result is an integer

Errors: none

Support: none

MPATCH, matrix patch

Function: This routine moves part of one vector or matrix in the STEX domain into another vector or matrix in the STEX domain.

Execution: explicit

MPATCH([what],[from],[to])

where [from] arguments are i, FV to move from vector FV starting at element i (integer)

[from] arguments are i, j, FM to move from matrix FM starting at element i, j (integers)

[to] arguments are k, TV to move to vector TV starting at element k (integer)

[to] arguments are k, ℓ, TM to move to matrix TM starting at element k, ℓ (integers)

[what] arguments are given by the chart:

from \ to	TV_k	$TM_{k,\ell}$
FV_i	m elements [what] as m	m elements into row k [what] as 1,m n elements into col ℓ [what] as n,1
$FM_{i,j}$	m elements from row i [what] as 1,m n elements from col j [what] as n,1	n rows × m cols [what] as n,m

Errors: none

Support: none

MPOLAR, matrix Cartesian to polar conversion

Function: This routine converts a complex vector or matrix in the STEX domain in Cartesian form to a double word vector or matrix in polar form.

Execution: implicit

MPOLAR(A)

where argument A is complex vector or matrix input in standard Cartesian form, i.e., each element represented by floating point scalars x and y such that the element = x + iy

result is double word vector or matrix in CSTAR in polar form, i.e., each element represented by floating point scalars r and θ such that the element = $x+iy = re^{i\theta}$; $0 \leq \theta < 2\pi$; if $x = y = 0$, then $r = \theta = 0$

SPIREL monitoring for creation of the result is given if SL14 is off. The polar form of a complex operand is a complex operand in the Genie language, but the arithmetic operations are not defined for this representation; input, output, and storage across an equals are meaningful for the polar form and useful. In the Genie language the same execution is specified by

POLAR(A)

for non-scalar argument A.

Errors: If A does not exist, MPOLAR prints an error message and performs no operation.

Support: program MCARTN

MPOWER, matrix power

Function: This routine raises a square standard matrix in the STEX domain to an integer power, generating a unit matrix for zero power, inverting for a negative power, and multiplying for powers > 1 in absolute value. The matrix must be of floating point type.

Execution: special

input (U) = codeword address of matrix
(R) = integer power

result in non-scalar accumulator, *10

If (U) null on entry, the input matrix is taken as *10. A matrix which is not *10 is not destroyed. SPIREL monitoring for creation of the result is provided if SL14 is off and the input is not *10 with power one.

Errors: If matrix does not exist, MPOWER prints error message and performs no operation. If power ≥ 0 and matrix is not square, MPOWER uses square portion, performs operation, and prints error message. If power < 0 and matrix is not square, MPOWER prints error message and performs no operation. If power < 0 and matrix is singular, no result is given, and MPOWER prints error message.

Support: programs INV, MCOPY, MMPY

MRE, matrix real part

Function: This routine provides the real part of a complex vector or matrix.

Execution: implicit

MRE(A)

where argument A is complex non-scalar input

result is in non-scalar accumulator, *10

MRE may be used on any double word non-scalar argument to provide the first part as a single word non-scalar. SPIREL monitoring for creation of the result is provided if SL14 is off. In the Genie language the same execution is specified by

RE(A)

for non-scalar argument A.

Errors: If A does not exist, MRE prints error message and performs no operation.

Support: program MCOPY

MSPACE, matrix space

Function: This routine creates a standard matrix of zeroes in the STEX domain.

Execution: explicit

MSPACE(A,B,C)

where argument A is matrix to be created
argument B is integer number of rows in A
argument C is integer number of columns in A

Storage addressed formerly as A is freed; then, if both B > 0 and C > 0, a matrix with B rows and C columns is created. SPIREL monitoring for freeing or creating A is provided if SL14 is off. If SL14 is on, MSPACE takes "fast" space by bypassing XCWD(*126).

Errors: none

Support: none

MSUB, matrix subtract

Function: This routine forms the difference of two standard vectors or two standard matrices in the STEX domain. The operands must agree in type, floating point or scalar, and the result will be of the same type.

Execution: special

input (B1) = codeword address for the operand to be subtracted from

(B2) = codeword address for the operand to be subtracted

result in non-scalar accumulator, *10

If either (B1) or (B2) null on entry, the corresponding operand is taken as the non-scalar accumulator, *10. An operand which is not *10 is not destroyed. SPIREL monitoring for creation of the result is provided if neither operand is *10 and SL14 is off.

Errors: If either operand does not exist, MSUB prints error message and performs no operation. If dimensions of the two operands are not the same, MSUB uses the subset of the larger which corresponds to the smaller, performs the subtraction, and prints error message.

Support: program MADD

MTAKE, matrix take

Function: This routine creates an n-dimensional array of zeroes.

Execution: explicit

$$MTAKE(A,D_1,\ldots,D_N,N)$$

where argument A is array to be created
argument D_i is length in the i^{th} dimension
argument $N \leq 5$ is the number of dimensions

Space addressed formerly as A is freed; then an array of size $D_1 x \ldots x D_N$ is created, to be indexed by registers B1,...,BN. SPIREL monitoring for creating A is provided if SL14 is off. If SL14 is on, MTAKE takes "fast" space by bypassing XCWD(*126).

Errors: If any $D_i < 1$, $n < 1$, or $n > 5$, MTAKE gives no result and prints an error message.

Support: none

ODD, test integer odd

Function: This routine tests an integer for being odd.

Execution: implicit

ODD(A)

where argument A is the integer input

result is Boolean value TRUE (represented by integer-0) if A is odd; Boolean value FALSE (represented by -1) if A is even.

Errors: none

Support: none

ORTHOG, matrix orthonormalization

Function: This routine orthonormalizes (by the Gram-Schmidt method) the rows of a standard matrix in the STEX domain. The matrix must be of floating point type.

Execution: implicit

ORTHOG(A)

where argument A is floating point standard matrix input
result is floating point standard matrix in the non-scalar accumulator, *10

Input matrix which is not *10 is not destroyed. SPIREL monitoring for creation of result is provided if input is not *10 and SL14 is off.

Errors: If A does not exist or if the rows are not linearly independent, ORTHOG prints an error message and performs no operation.

Support: programs MCOPY, SQR

OUTPUT, special output

Function: This routine is supplied by the user for special output from programs written in the Genie language.

Execution: in the Genie language only, by the command

OUTPUT list

where the program OUTPUT is entered once for each named variable in the list. A complex variable is treated as two items, the real part with the name of the variable and the imaginary part with the name "ditto". The program OUTPUT must be coded in the assembly language, AP1. Information is given in T7 on entry to OUTPUT as follows:

bits	1-30	name in BCD as given in list
	31-33	not used
	34-36	octal 0 for scalar
		2 for vector
		4 for matrix
	39-41	not used
	40-54	address for scalar, codeword address for non-scalar

PLOT, plot on the printer

Function: This routine plots on the printer one floating point vector versus another or a floating point vector versus its indices.

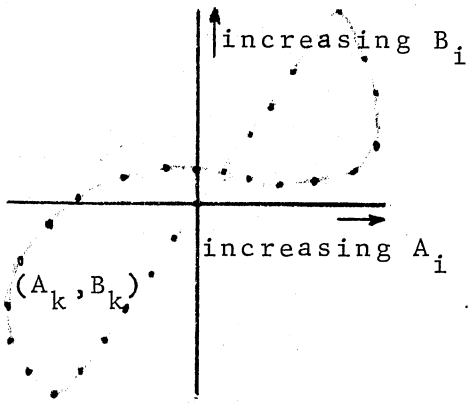
Execution: explicit

PLOT(A,B)

where argument A is a vector of x-values to be plotted across the page from min on the left to max on the right

argument B is a vector of y-values to be plotted down the page from max at the top to min at the bottom

result is one-page plot of points (A_k, B_k), k in the index range of both A and B

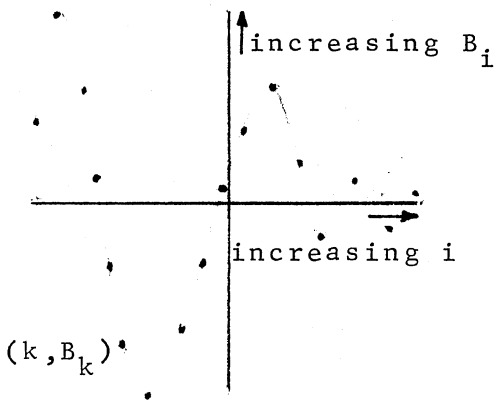


PLOT(Z,B)

where argument Z (actually zero on the B6-list) specifies use of indices for x-values to be plotted across the page from min on the left to max on the right

argument B is a vector of y-values to be plotted down the page from max at the top to min at the bottom

result is one-page plot of points (k, B_k), k in the index range of B



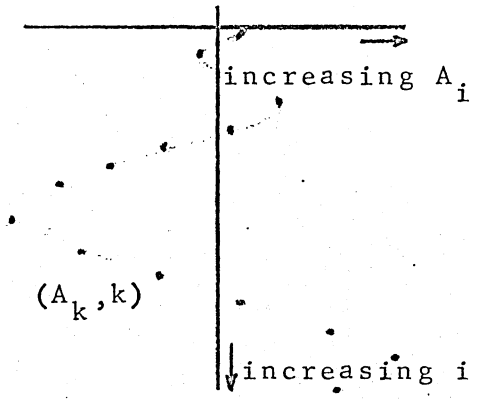
PLOT (continued)

PLOT(A ,Z)

where argument A is a vector of x-values to be plotted across the page from min on the left to max on the right

argument Z (actually zero on the B6-list) specifies use of indices for y-values to be plotted down the page from min at the top to max at the bottom

result is plot on one or more pages of points (A_k,k), k in the index range of A, one point per index value



No vector operands are destroyed.

Errors: If both arguments are Z, an error message is printed.

Support: none

POLAR, Cartesian to polar conversion

Function: This routine converts a complex scalar in Cartesian form to a double word operand in polar form.

Execution: implicit

POLAR(A)

where argument A is complex scalar input in standard Cartesian form, i.e., represented by floating point scalars x and y such that $A = x + iy$

result is double word operand in polar form, i.e., represented by floating point scalars r and θ such that $A = x + iy = re^{i\theta}$; $0 \leq \theta < 2\pi$; if $x = y = 0$, then $r = \theta = 0$

The polar form of a complex operand is a complex operand in the Genie language, but the arithmetic operations are not defined for this representation; input, output and storage across an equals are meaningful for the polar form and useful.

Errors: none

Support: programs ASIN, MOD; scalar CMPLX

PRESCRIBE, line print with format and page control

Function: This routine is used in Genie programs only instead of SCRIBE to produce printed SCRIBE output with page control, headings, and page numbers.

Execution: explicit*

PRESCRIBE(A1,...,AK,F,N,TITLE,LIMIT)

where argument F is the name of the format to be used
arguments A1,...,AK are variables whose values are to be substituted successively for dummy variables in the format
} exactly as for SCRIBE

argument N is the number of spaces after output of SCRIBE (A1,...,AK,F)

argument TITLE is either the name of a format containing only text or the name of a vector containing hexad data, to be used for title on pages (may be more than 108 characters to exceed one line)

argument LIMIT is the number of lines to be printed per page of output

and * one additional argument is supplied automatically by the Genie compiler -- minus the number of arguments A1,...,AK,N,TITLE,LIMIT stored directly on the B6-list as a negative integer after the arguments A1,...,AK,N,TITLE,LIMIT

If LINCT exceeds LIMIT on entry, PRESCRIBE prints a heading containing the specified title at the top of the next page. Then SCRIBE(A1,...,AK,F) is executed. Finally, N spaces are provided. LINCT is updated to reflect all printing and spacing by PRESCRIBE.

The heading is provided by PRESCRIBE after incrementing PAGCT by 1. It consists of a 1/2 inch margin (3 blank lines) at the top of the page, then the lines:

...data and time... ...page no...

...title supplied by user...

...blank line...

PAGCT is used as the page number, and LINCT is set to 5 plus the

PRESCRIBE (continued)

number of lines in the title after a heading print.

In a fresh Genie SPIREL, LINCTR = PAGCTR = 0. Either may be used within private programs. Both should be updated if printing is done other than through SCRIBE or PRESCRIBE. Setting LINCTR = 0 forces a new page on the next entry to PRESCRIBE. LIMIT = 60 provides a 1/2 inch margin at the bottom of the page to match the margin at the top.

Errors: same as for SCRIBE

Support: program SCRIBE; constants LINCT, PAGCT

QCONF, χ^2 confidence

Function: This routine computes the χ^2 confidence level between two floating point vectors of equal length.

Execution: implicit

QCONF(A,B)

where argument A is the theoretical distribution, real floating point vector input

argument B is the observed distribution, real floating point vector input

result is real scalar, computed as

$$Q(\chi^2 | \nu) = \int_{\chi^2}^{\infty} e^{-\frac{t}{2}} t^{\frac{\nu}{2}-1} dt$$

with degrees of freedom ν=vector length -1

Errors: If A and B are not equal in length or if either does not exist, QCONF prints an error message and gives result=1.0. If vector length <2, QCONF prints an error message and gives result=0.

Support: programs CHISQ, EXP, SQR

RANDM, random number generator

Function: This routine computes the first or the next in a sequence of pseudo-random floating point numbers evenly distributed between 0.0 and 1.0.

Execution: implicit

RANDM(A)

where argument A $\neq$ 0 causes generation of the first random number, i.e., restarts the generation procedure

argument A $\equiv$ Z causes generation of the next random number (starts the generation procedure on first execution)

result is floating point scalar

Errors: none

Support: none

RE, real part

Function: This routine provides the real part of a complex scalar.

Execution: implicit

RE(A)

where argument A is complex scalar input

result is real floating point scalar

If coded in the Genie language, the library routine is not used; but the routine may be used in assembly language coding. RE may be used on any double word scalar argument to provide the first part as a single word scalar.

Errors: none

Support: none

ROW, number of rows in a matrix

Function: This routine provides the number of rows in a matrix.

Execution: implicit

ROW(A)

where argument A is matrix input

result is integer number of rows in matrix A.

Errors: If A does not exist, result = 0 and an error message is printed.

Support: program LENGTH

RTRAN, real fast Fourier transformation

Function: this program is used in conjunction with FFT by FFTC in those cases where 1) the complex input is conjugate symmetric and the output real; or 2) the input is real and even in length, and the output is complex and conjugate symmetric.

Execution: explicit (see FFTC)

RTRAN(A,B,C)

where argument A is the input/output vector (complex)

B is a Boolean variable: true if the sign of the exponent is negative, otherwise false.

C is a Boolean variable: true for a transform, otherwise false.

Errors: If A doesn't exist, an error message is printed.

Support: programs COS, SIN.

SCRIBE, line print with format

Function: This routine substitutes variables for dummy fields in a line skeleton called a format and prints the result. It may be used in Genie programs only.

Execution: explicit*

SCRIBE(A1,...,AK,F)

where argument F is the name of the format to be used

argument A1,...,AK are variables whose values are to be substituted successively for dummy variables in the format

and * one additional argument is supplied automatically by the Genie compiler -- minus the number of arguments A1,...,AK,F stored directly on the B6-list as negative integer after the arguments A1,...,AK,F

result is that printing occurs and the constant LINCT is incremented by 1 for each line.

A format is a line skeleton written as a FORMAT statement in the Genie language, as a BCD pseudo-order in APl. A format contains text and dummy variables. Special characters are used to form dummy variables: lower case letters 'a,b,c,d,e,f', the characters '+,-,.' with 'd' and 'e', and the digits '0' thru '9' with 'f'. A dummy variable is any consecutive sequence of special characters in the format. All other characters in the format are characters of text. The use of special characters to form dummy variables is explained below.

SCRIBE operates by transferring text directly to the printed output and substituting argument values for dummy variables. The number of arguments need not equal the number of dummy variables in the specified format. If the number of arguments is less than the number of dummy variables, processing will cease when a dummy variable is encountered for which there is no argument. If the number of arguments is greater than the number of dummy variables,

SCRIBE (continued)

the format will be used as many times as necessary to substitute all arguments, and each re-use of the format will cause a new line of printing to be initiated. The processing of a non-scalar argument is handled by replacement of successive dummy variables in the specified format with successive array elements -- all words of a program, all vector elements, all matrix elements by row, and generally all data words of any array.

One or more lines may be printed on a single entry to SCRIBE. Line termination occurs due to:

- special position notation 'f0f'
  -- causes printing and initialization of the next line at print position 1 (as on entry), but scan of the format continues.
- end of format
  -- if no arguments remain to be processed, causes printing and exit
  -- if more arguments remain to be processed, causes printing, initialization of the next line at print position 1, and reinitialization of the format scan.
- dummy variable in format and no arguments remain to be processed
  -- causes printing and exit

A dummy variable consists of a string of special characters with no embedded blanks. The representation determines the type of conversion to be applied to an argument and the appearance of the output. The types of dummy variables are as follows:

A hexad dummy is formed by a string of 'a's with possibly embedded 'c's. The occurrence of any 'a' specifies hexad conversion of the argument. Each 'a' specifies the position of a hexad character, and each 'c' specifies a space within the field. Hexads are taken from the left end of the word: a hexad dummy with three 'a's will cause the three leftmost hexads of the argu-

ment specified to be printed. A machine word contains nine hexads; if there are more than nine 'a's in single hexad dummy, then more than one word of input must be used. If the argument is a scalar, successive words in memory will be used. If the argument is a non-scalar, successive array elements will be used.

An octal dummy is formed by a string of 'b's with possibly embedded 'c's. The occurrence of any 'b' specifies octal conversion of the argument. Each 'b' specifies the position of an octal digit, and each 'c' specifies a space within the field. A machine word contains eighteen octal digits, so no more than eighteen 'b's in a dummy variable are meaningful. Octal digits are taken from the right end of the word: a dummy variable with four 'b's will cause the four rightmost octal digits of the argument specified to be printed.

A decimal dummy is formed by a string of 'd's with possibly embedded 'c's, and perhaps the special characters '+,-,.,e'. The occurrence of any 'd' specifies decimal conversion of the argument. Each character in the dummy specifies a position in the decimal output. The general form of a decimal dummy is:

±d···d.d···de±d···d

The decimal point '.' will appear in the output as in the dummy. It must appear to get the fractional part of a floating point argument, and then the fractional part is rounded in the last digit. If no decimal point appears, the last digit in the integer is rounded.

The character 'e' appears in the printed output and indicates that the integer following it is the power of ten for the number in front of it. The 'e' causes output of a full field of decimal digits before the decimal point and an appropriate exponent.

Each 'd' specifies a decimal digit position in the output, before or after the decimal point or in the exponent.

A character '+' or '-' specifies a sign position in

the output, either on the number or on its exponent. '-' specifies to print a sign (minus) only if the number which follows is negative. '+' specifies to always print a sign (plus or minus). A sign is always printed immediately to the left of the most significant digit of the number to which it applies.

A decimal dummy without 'e' may be overflowed by an argument value. The number will then be truncated on the left, X being printed as the leftmost character. Thus with a dummy -dd.ddd the value 5763.4587 will be printed as X63.459.

A position dummy does not use an argument; it is formed by a pair of 'f's which bracket an unsigned integer which is the print position to move to in forming the line of output. The 'f's and the bracketed number do not appear on the printed output. The print positions are numbered from 1 to 108 from left to right. Any number of pairs of 'f's may appear anywhere except within variables on a dummy line. As a special case, 'f0f' causes printing and initialization of the next line at print position 1.

Examples: [Note _ denotes space]

dummy variable	argument value	output
aaa	hexad ABCDEFGHI	ABC
aacaa	hexad ABCDEFGHI	AB_CD
aaaaaaaaa	hexad THE_END__	THE_END_ _
bb	octal ...461	61
bb	octal 0...01	_1
bbcb	octal ...461	46_1
d.d	decimal 3.59	3.6
dd.d	decimal 3.51	_3.5
d	decimal 3.5	4
dd	decimal 3	_3
dd.d	decimal 3	_3.0
-d.d	decimal 3.52	_3.5

SCRIBE (continued)

dummy variable	argument value	output
-dd.d	decimal -0.52	_-0.5
+d.d	decimal 3.52	+3.5
+d.d	decimal -0.52	-0.5
d.d	decimal -3.52	3.5
d.d	decimal 35.67	5.7 or X.7
ddcddd	decimal 1024	_1_024
-dd.ddde+dd	decimal 45784.734	_45.785e_+3
+dd.ddde-dd	decimal 45.784834	+45.785e__0
+dd.dddce-dd	decimal 45.784734	+45.785_e__0

In Genie, the format given by the statement

```
SKEL      FORMAT
FIRST RESULTS    aaaacaaa    A=bbbcbbb    B=-d.dddd    C=-dd.ddce+d
```

might be used in the explicit execution command

```
EXECUTE        SCRIBE(ALPHA ,AONE ,BTWO ,A+B ,SKEL)
```

to cause the printed output

```
FIRST RESULTS    TRUE END    A=147 003    B= 4.5969    C=-47.594 e-1
```

Errors: X in decimal field as explained under decimal dummy.

Support: constant LINCT

SIN, sine

Function: This routine computes the sine of a number.

Execution: implicit

SIN(A)

where argument A is floating point scalar input

result is floating point scalar

Also, (R) = COS(A) on exit.

Errors: If $|A| \geq .2^{47}$, SIN gives result = 0 and prints an error message.

Support: none

SINH, hyperbolic sine

Function: This routine computes the hyperbolic sine of a number.

Execution: implicit

SINH(A)

where argument A is floating point scalar input

result is floating point scalar

Also, (R) = COSH(A) on exit.

Errors: If $|A| > 170.0$, SINH gives result for $|A| = 170.0$ and prints error message.

Support: program EXP

SMDIV, scalar-matrix divide

Function: This routine divides a standard vector or matrix in the STEX domain by a scalar. The operands must agree in type, floating point or integer, and the result will be of the same type.

Execution: special

input (B1) = codeword address for non-scalar operand
(U) = scalar operand

result in non-scalar accumulator, *10

If (B1) null on entry, the non-scalar operand is taken as *10. A non-scalar operand which is not *10 is not destroyed. SPIREL monitoring for creation of the result is provided if the non-scalar operand is not *10 and SL14 is off.

Errors: If the non-scalar operand does not exist or if the scalar = 0, SMDIV prints an error message and performs no operation.

Support: program SMMPY

SMMPY, scalar-matrix multiply

Function: This routine forms the product of a scalar and a standard vector or matrix in the STEX domain. The operands must agree in type, floating point or integer, and the result will be of the same type.

Execution: special

input (B1) = codeword address for non-scalar operand
(U) = scalar operand

result in non-scalar accumulator, *10

If (B1) null on entry, the non-scalar operand is taken as *10. A non-scalar operand which is not *10 is not destroyed. SPIREL monitoring for creation of the result is provided if the non-scalar operand is not *10 and SL14 is off.

Errors: If non-scalar operand does not exist, SMMPY prints an error message and performs no operation.

Support: program MCOPY

SOLN, linear equations solution

Function: This routine provides the solution to a system of linear equations represented by a square standard matrix of coefficients and a standard vector of constants, both of floating point type and in the STEX domain.

Execution: implicit

SOLN(A,B)

where argument A is a square floating matrix of coefficients

argument B is a floating point vector of constants

representing a system of n equations of the form

$$A_{i,1}X_1 + \ldots + A_{i,n}X_n = B_i \quad , \; i=1,2,\ldots,n$$

result is floating point vector in the non-scalar accumulator, *10, with the value of X_i as the i^{th} element.

An operand which is not *10 is not destroyed. SPIREL monitoring for creation of the result is provided if SL14 is off.

Errors: If A or B does not exist or if dimensions are not proper or if a solution is not defined, SOLN prints an error message and performs no operation.

Support: program INV

SQR, square root

Function: This routine computes the square root of a number.

Execution: implicit

SQR(A)

where argument A is floating point scalar input

result is floating point scalar

Errors: If A < 0, SQR gives result = 0 and prints error message.

Support: none

STNDV, standard deviation

Function: This routine computes the standard deviation of any vector of floating point numbers.

Execution: implicit

STNDV(A)

where argument A is floating point vector input

result is floating point scalar

Errors: If A does not exist, STNDV prints an error message and performs no operation.

Support: program SQR

TAN, tangent

Function: This routine computes the tangent of a number.

Execution: implicit

TAN(A)

where argument A is floating point scalar input

result is floating point scalar

Errors: If $|A| \geq 2^{47}$, or if $|A|$ is a multiple of $\pi/2$, TAN gives result = 0 and prints an error message.

Support: none

TANH, hyperbolic tangent

Function: This routine computes the hyperbolic tangent of a number.

Execution: implicit

TANH(A)

where argument A is floating point scalar input

result is floating point scalar

If |A| > 170.0, TANH gives |result| = 1.0.

Errors: none

Support: program EXP

TRAN, matrix transpose

Function: This routine forms the transpose of a standard matrix in the STEX domain.

Execution: implicit

TRAN(A)

where argument A is standard matrix input

result is standard matrix in the non-scalar accumulator,*10 Input matrix which is not *10 is not destroyed. SPIREL monitoring for creation of the result is provided if SL14 is off.

Errors: If A does not exist, TRAN prints an error message and performs no operation.

Support: none

TTAKE, Triangular Matrix Take

Function: This routine creates an upper triangular matrix of zeroes in the STEX domain.

Execution: explicit

TTAKE(A,N)

where argument A is the matrix to be created
argument N is the size of the matrix (N×N)

Storage formerly addressed as A is freed; then, if N > 0, a triangular matrix is created. If N = 0, any storage for A is freed and A is cleared.

Errors: none

Support: none

VREV, vector reversal

Function: This routine reverses the order of the elements of a vector.

Execution: explicit

VREV(A)

where argument A is vector input

result replaces vector A

Errors: If A does not exist, VREV prints an error message and gives no result.

Support: none

VSPACE, vector space

Function: This routine creates a standard vector of zeroes in the STEX domain.

Execution: explicit

VSPACE(A,B)

where argument A is vector to be created

argument B is integer length of A

Storage addressed formerly as A is freed; then, if B > 0, vector A of length B is created. SPIREL monitoring for freeing or creation of A is provided if SL14 is off. If SL14 is on, VSPACE takes "fast" space by bypassing XCWD(*126).

Errors: none

Support: None

←CDUP, duplicate U, R to CSTAR

Function: This routine takes a complex codeword in U, R and duplicates it into CSTAR, the complex non-scalar accumulator. It is used when complex arrays are subscripted to one level in GENIE.

Support: CSTAR, complex non-scalar

←COMP, compression

Function: This routine performs compression of the library. It is never executed by a user, either manually or under program control.

Execution: internal library use only -- receives control from EDIT

Errors: none

Support: none

←CSAV, save CSTAR on B6 list

Function: This routine saves or duplicates CSTAR, the complex non-scalar accumulator, on to the B6 list. It is used for complicated complex non-scalar expressions in GENIE.

Support: CSMT, complex non-scalar

←CSWP, swap (B1), (B2), or (B1) + 1 to CSTAR

Function: This routine swaps the codeword in CSTAR to (B1), (B2), and changes the backreference. (B1), (B2) is erased first if normal entry is used but not so if entry is at order 2. In the latter case, the input is assumed to be (B1) and (B1) + 1. It is used in complex non-scalar stores in GENIE.

Support: CSTAR, complex non-scalar

←ENTRY, entry to library

Function: This routine records information about entry to a library routine, the name of the routine and the PF setting.

Execution: internal library use only

Errors: none

Support: constant ←ELOC

←ERPR, error print

Function: This routine prints error message containing text from the calling program and information about the PF setting at time of the error.

Execution: internal library use only, from all programs supplying error messages

Errors: none

Support: constant ←ELOC

←EXEC, Arithmetic Evaluator

Function: This routine carries out arithmetic operations called for by arithmetic statements input to CONSOL

Execution: internal system use only, called by ←IFE

Errors: none

Support: ←TYPE, vector ←CT, FXEXP, FLEXP, ←INOU

←INOUT, input/output

Function: This routine does input and output from compiled programs to carry out DPUNCH, READ, PRINT, PUNCH, DATA, INPUT, OUTPUT, DISPLAY, and ACCEPT commands in the Genie language.

Execution: from Genie-generated code only -- by TRA (not TSR) which may be traced

input (B1) specifies operation: 0,1,2,3,4,5,6,7,10 for DPUNCH READ, PRINT, PUNCH, DATA, INPUT, OUTPUT, DISPLAY, ACCEPT respectively

list of arguments one per word following TRA, terminated by null word; each word containing name in BCD and addressing information

return to location following null word

Monitoring of the input/output operation is provided if SL14 is off.

Errors: none

Support: programs INPUT, OUTPUT

←FETC, Interpreter Fetch

Function: Used by ←IFE to fetch characters from *TEXT (174).

Execution: Internal system use only.

Errors: none

Support: none

←IFE, Interpretive Formula Evaluator

Function: This routine performs statement scanning of arithmetic expressions input to CONSOL.

Execution: Internal system use only.

Errors: If illegal syntactic expression found, prints error message and returns to CONSOL.

Support: ←LASC, ←LOOK, ←TYPE, ←FETC
vector ←CT.

- Description of Use:

  Names: The name of any library subroutine may not be used for a private name. Also any 2 character sequence which is the mnemonic for a SPIREL command may not be used as a private name. All names are external system quantities, as they are on the Symbol Table (*113).

  Type of Results: Floating point always takes precedence over fixed point. The type of a variable becomes fixed when it first appears on the L.H.S. of an equation. Storing of an integer R.H.S. to a floating point L.H.S. will cause the integer to be floated before the store. However, a floating point R.H.S. will always be stored that way regardless of the type of the L.H.S.

  Arithmetic operators: The standard set of operators are available:

  binary: +, -, /, x(lower case x)
  Multiplication may be implied as in GENIE, when unambiguous. However if A1 and B are names, A1B will not be taken as A1 x B; but A1 B will be, where the ' ' represents a space.

  unary: -, | |(abs. value bars), and +(indicates what follows is to be interpreted as an octal number).

Superscripts and subscripts are allowed following the standard GENIE conventions.

NOTE: Genie interprets $-A^B$ as $(-A)^B$. ←IFE interprets it the way it looks: $-(A^B)$

Functions: Any library or user function may be executed implicitly in an ←IFE statement. Arguments may be scalar or non-scalar, but in no case may they be complex. ←IFE cannot operate in any case on complex quantities.

For convenience of notation, functions may be raised to a power immediately after the function name and preceeding the '(args)'. For example

$$A = SIN(X)^2 + COS(X)^2$$

may be written as

$$A = SIN^2(X) + COS^2(X).$$

However, this operation is meaningful only if the function has a single, non-complex, scalar result.

NOTE: For any function which has a single scalar parameter, that parameter will be floated before execution takes place. Therefore the function FLOAT may not be used in the ←IFE language.

Explicit execution: A function may be explicitly executed with args in the following manner. A dummy variable is used on the L.H.S. of a statement that would otherwise call for implicit

execution. For example:

```
A = TTAKE(B,5)
```

will create a triangular matrix at B.
If a function executed in this manner has a non-scalar result, and does not work in place, the resulting array will be in USTAR (*10).

Summary and Further Comments:

1) Type of variables: real or integer scalar only, except that real non-scalars may appear in function arguments. (A single element of a non-scalar is a scalar).
2) Rank of operations:
   +, -, x, /, -(unary), | |, function call, ↓, ↑.
3) Number format:
   integer: 5, 376
   real: .5, 5.1, 5.3*-3
   octal: +533
4) Statement length: May not exceed four (4) lines on the display scope.
5) Special Display Option: If the L.H.S. is a non-subscripted variable, an equal sign ('=') if placed at the end of the statement will cause the value stored to be displayed on the scope. An HTR -- will occur. Press continue to return to CONSOL COMMUNICATION LOOP.
6) More than one statement may be included in a line, provided they are separated by a comma ','. No interdependencies are accounted for. The statements

will be evaluated in the order in which they appear. For example:

```
A = B+C =, F = SIN(A) =, G = G =
```

will display A(=B+C) and then compute and display F, and display G.

7) Other I-O:
Printing must be done with the standard SPIREL print command. There is no ←IFE equivalent to the GENIE-DATA statement.

8) ←IFE statements and SPIREL commands may not appear on the same line.

9) ←IFE and all associated programs are edited out of the library with EX EDIT. To keep them in such a system, a dummy APl program must be included with a single REF to ←IFE.

←LASC, Statement Scanner

Function: Performs conversion to reverse polish of an arithmetic expression of the form accepted by ←IFE.

Execution: internal system use only.

Errors: none

Support: ←EXEC, vector ←CT

←LOOK, check for special command

Function: Used by ←IFE to look for special command sequences.

Execution: internal system use only.

Errors: none

Support: none

←RDUP, duplicate U to *10

Function: This routine takes a codeword in U and duplicates it into *10, the non-scalar accumulator. It is used when arrays are subscripted to one level in GENIE.

Support: none

←RSAV, save *10 on B6 list

Function: This routine saves or duplicates *10, the non-scalar accumulator, on to the B6 list. It is used for complicated non-scalar expressions in GENIE.

Support: none

←RSWP, swap (B1) to *10

Function: This routine swaps the codeword in 10 to (B1) and changes the backreference. (B1) is erased first if normal entry is used but not so if entry is at order 2. It is used in non-scalar stores in GENIE.

Support: none

←TYPE, determine shape of ST entry

Function: Used by ←IFE and ←EXEC to determine shape of symbol table entry.

Execution: internal system use only

Error: none

Support: none

- Punching

The program EDIT is used for punching all library items.

The punch procedure is:

(1) Load SPIREL from paper tape or magnetic tape.

(2) Load all necessary updates to the library routines -- corrections, new versions of programs and new programs.

(3) Erase (ER at console) any programs to be deleted from the package.

(4) Do not activate STEX or execute any program other than EDIT.

(5) Execute from word 2 of the program EDIT with a control word to SPIREL, manually or off paper tape.

(6) Initialization occurs -- ST and VT indices set so that all entries have tag 1 and negative indices and last entry in use is at -0. ST and VT are alphabetized.

(7) Punching of the package starts. Interrupt by turning on SL15 when enough tape is punched. CONTINUE to resume punching. EDIT exits when punching is finished.

(8) To check new punched tapes, load with SL15 on.

- Editing

The program EDIT is used for updating the SPIREL library in memory or on magnetic tape.

The edit procedure is:

(1) Load SPIREL from paper tape or magnetic tape.

(2) Load all necessary updates to library routines, named and numbered -- corrections, new versions of programs, and new programs.

(3) Do not activate STEX or execute any program other than EDIT.

(4) Execute from word 3 of the program EDIT with a control word to SPIREL, manually or off paper tape.

(5) Initialization occurs -- ST and VT indices set so that all entries have tag 1 and negative indices and last entry in use is at -0.

(6) Control returns to the console communications loop, and the library is updated in memory.

(7) Write on magnetic tape, if desired.