Floating-Point Support

Many Microsoft run-time library functions require floating-point support from a math coprocessor or from the floating-point libraries that accompany the compiler. Floating-point support functions are loaded only if required.

When you use a floating-point type specifier in the format string of a call to a function in the **printf** or **scanf** family, you must specify a floating-point value or a pointer to a floating-point value in the argument list to tell the compiler that floating-point support is required. The math functions in the Microsoft run-time library handle exceptions the same way that the UNIX V math functions do.

The Microsoft run-time library sets the default internal precision of the math coprocessor (or emulator) to 64 bits. This default applies only to the internal precision at which all intermediate calculations are performed; it does not apply to the size of arguments, return values, or variables. You can override this default and set the chip (or emulator) back to 80-bit precision by linking your program with LIB/FP10.OBJ. On the linker command line, FP10.OBJ must appear before LIBC.LIB, LIBCMT.LIB, or MSVCRT.LIB.

Floating-Point Functions

Routine	**Use**
abs	Return absolute value of **int**
acos	Calculate arccosine
asin	Calculate arcsine
atan, atan2	Calculate arctangent
atof	Convert character string to double-precision floating-point value
Bessel functions	Calculate Bessel functions **_j0**, **_j1**, **_jn**, **_y0**, **_y1**, **_yn**
_cabs	Find absolute value of complex number
ceil	Find integer ceiling
_chgsign	Reverse sign of double-precision floating-point argument
_clear87, _clearfp	Get and clear floating-point status word
_control87, _controlfp	Get old floating-point control word and set new control-word value
_copysign	Return one value with sign of another
cos	Calculate cosine
cosh	Calculate hyperbolic cosine
difftime	Compute difference between two specified time values
div	Divide one integer by another, returning quotient and remainder
_ecvt	Convert **double** to character string of specified length
exp	Calculate exponential function

fabs	Find absolute value
_fcvt	Convert **double** to string with specified number of digits following decimal point
_finite	Determine whether given double-precision floating-point value is finite
floor	Find largest integer less than or equal to argument
fmod	Find floating-point remainder
_fpclass	Return status word containing information on floating-point class
_fpieee_flt	Invoke user-defined trap handler for IEEE floating-point exceptions
_fpreset	Reinitialize floating-point math package
frexp	Calculate exponential value
_gcvt	Convert floating-point value to character string
_hypot	Calculate hypotenuse of right triangle
_isnan	Check given double-precision floating-point value for not a number (NaN)
labs	Return absolute value of **long**
ldexp	Calculate product of argument and 2 to specified power
ldiv	Divide one **long** integer by another, returning quotient and remainder
log	Calculate natural logarithm
log10	Calculate base-10 logarithm
_logb	Extract exponential value of double-precision floating-point argument
_lrotl, _lrotr	Shift **unsigned long int** left (**_lrotl**) or right (**_lrotr**)
_matherr	Handle math errors
__max	Return larger of two values
__min	Return smaller of two values
modf	Split argument into integer and fractional parts
_nextafter	Return next representable neighbor
pow	Calculate value raised to a power
printf, wprintf	Write data to **stdout** according to specified format
rand	Get pseudorandom number
_rotl, _rotr	Shift **unsigned int** left (**_rotl**) or right (**_rotr**)
_scalb	Scale argument by power of 2
scanf, wscanf	Read data from **stdin** according to specified format and write data to specified location

sin	Calculate sine
sinh	Calculate hyperbolic sine
sqrt	Find square root
srand	Initialize pseudorandom series
_status87, _statusfp	Get floating-point status word
strtod	Convert character string to double-precision value
tan	Calculate tangent
tanh	Calculate hyperbolic tangent