Intrinsic Functions

The following table lists the intrinsic functions available in HLSL. Each function has a brief description, and a link to a reference page that has more detail about the input argument and return type.

Name	Description	Minimum shader model
abort	Terminates the current draw or dispatch call being executed.	4
abs	Absolute value (per component).	1[1]
acos	Returns the arccosine of each component of x.	1[1]
all	Test if all components of x are nonzero.	1[1]
AllMemoryBarrier	Blocks execution of all threads in a group until all memory accesses have been completed.	5
AllMemoryBarrierWithGroupSync	Blocks execution of all threads in a group until all memory accesses have been completed and all threads in the group have reached this call.	5
any	Test if any component of x is nonzero.	1[1]
asdouble	Reinterprets a cast value into a double.	5
asfloat	Convert the input type to a float.	4
asin	Returns the arcsine of each component of x.	1[1]
asint	Convert the input type to an integer.	4
asuint	Reinterprets the bit pattern of a 64-bit type to a uint.	5
asuint	Convert the input type to an unsigned integer.	4
atan	Returns the arctangent of x.	1[1]
atan2	Returns the arctangent of of two values (x,y).	1[1]
ceil	Returns the smallest integer which is greater than or equal to x.	1[1]

CheckAccessFullyMapped	Determines whether all values from a **Sample** or **Load** operation accessed mapped tiles in a tiled resource.	5
clamp	Clamps x to the range [min, max].	1[1]
clip	Discards the current pixel, if any component of x is less than zero.	1[1]
cos	Returns the cosine of x.	1[1]
cosh	Returns the hyperbolic cosine of x.	1[1]
countbits	Counts the number of bits (per component) in the input integer.	5
cross	Returns the cross product of two 3D vectors.	1[1]
D3DCOLORtoUBYTE4	Swizzles and scales components of the 4D vector xto compensate for the lack of UBYTE4 support in some hardware.	1[1]
ddx	Returns the partial derivative of x with respect to the screen-space x-coordinate.	2[1]
ddx_coarse	Computes a low precision partial derivative with respect to the screen-space x-coordinate.	5
ddx_fine	Computes a high precision partial derivative with respect to the screen-space x-coordinate.	5
ddy	Returns the partial derivative of x with respect to the screen-space y-coordinate.	2[1]
ddy_coarse	Computes a low precision partial derivative with respect to the screen-space y-coordinate.	5
ddy_fine	Computes a high precision partial derivative with respect to the screen-space y-coordinate.	5
degrees	Converts x from radians to degrees.	1[1]
determinant	Returns the determinant of the square matrix m.	1[1]
DeviceMemoryBarrier	Blocks execution of all threads in a group until all device memory accesses have been completed.	5
DeviceMemoryBarrierWithGroupSync	Blocks execution of all threads in a group until all device memory accesses have been completed and all threads in the group have reached this call.	5
distance	Returns the distance between two points.	1[1]
dot	Returns the dot product of two vectors.	1

dst	Calculates a distance vector.	5
errorf	Submits an error message to the information queue.	4
EvaluateAttributeAtCentroid	Evaluates at the pixel centroid.	5
EvaluateAttributeAtSample	Evaluates at the indexed sample location.	5
EvaluateAttributeSnapped	Evaluates at the pixel centroid with an offset.	5
exp	Returns the base-e exponent.	1[1]
exp2	Base 2 exponent (per component).	1[1]
f16tof32	Converts the float16 stored in the low-half of the uint to a float.	5
f32tof16	Converts an input into a float16 type.	5
faceforward	Returns -n * sign(dot(i, ng)).	1[1]
firstbithigh	Gets the location of the first set bit starting from the highest order bit and working downward, per component.	5
firstbitlow	Returns the location of the first set bit starting from the lowest order bit and working upward, per component.	5
floor	Returns the greatest integer which is less than or equal to x.	1[1]
fma	Returns the double-precision fused multiply-addition of a * b + c.	5
fmod	Returns the floating point remainder of x/y.	1[1]
frac	Returns the fractional part of x.	1[1]
frexp	Returns the mantissa and exponent of x.	2[1]
fwidth	Returns abs(ddx(x)) + abs(ddy(x))	2[1]
GetRenderTargetSampleCount	Returns the number of render-target samples.	4
GetRenderTargetSamplePosition	Returns a sample position (x,y) for a given sample index.	4
GroupMemoryBarrier	Blocks execution of all threads in a group until all group shared accesses have been completed.	5
GroupMemoryBar	Blocks execution of all threads in a group until all group shared accesses	5

rierWithGroupSync	have been completed and all threads in the group have reached this call.	
InterlockedAdd	Performs a guaranteed atomic add of value to the dest resource variable.	5
InterlockedAnd	Performs a guaranteed atomic and.	5
InterlockedCompareExchange	Atomically compares the input to the comparison value and exchanges the result.	5
InterlockedCompareStore	Atomically compares the input to the comparison value.	5
InterlockedExchange	Assigns value to dest and returns the original value.	5
InterlockedMax	Performs a guaranteed atomic max.	5
InterlockedMin	Performs a guaranteed atomic min.	5
InterlockedOr	Performs a guaranteed atomic or.	5
InterlockedXor	Performs a guaranteed atomic xor.	5
isfinite	Returns true if x is finite, false otherwise.	1[1]
isinf	Returns true if x is +INF or -INF, false otherwise.	1[1]
isnan	Returns true if x is NAN or QNAN, false otherwise.	1[1]
ldexp	Returns x * 2exp	1[1]
length	Returns the length of the vector v.	1[1]
lerp	Returns x + s(y - x).	1[1]
lit	Returns a lighting vector (ambient, diffuse, specular, 1)	1[1]
log	Returns the base-e logarithm of x.	1[1]
log10	Returns the base-10 logarithm of x.	1[1]
log2	Returns the base-2 logarithm of x.	1[1]
mad	Performs an arithmetic multiply/add operation on three values.	5
max	Selects the greater of x and y.	1[1]
min	Selects the lesser of x and y.	1[1]
modf	Splits the value x into fractional and integer parts.	1[1]

msad4	Compares a 4-byte reference value and an 8-byte source value and accumulates a vector of 4 sums.	5
mul	Performs matrix multiplication using x and y.	1
noise	Generates a random value using the Perlin-noise algorithm.	1[1]
normalize	Returns a normalized vector.	1[1]
pow	Returns x^y.	1[1]
printf	Submits a custom shader message to the information queue.	4
Process2DQuadTessFactorsAvg	Generates the corrected tessellation factors for a quad patch.	5
Process2DQuadTessFactorsMax	Generates the corrected tessellation factors for a quad patch.	5
Process2DQuadTessFactorsMin	Generates the corrected tessellation factors for a quad patch.	5
ProcessIsolineTessFactors	Generates the rounded tessellation factors for an isoline.	5
ProcessQuadTessFactorsAvg	Generates the corrected tessellation factors for a quad patch.	5
ProcessQuadTessFactorsMax	Generates the corrected tessellation factors for a quad patch.	5
ProcessQuadTessFactorsMin	Generates the corrected tessellation factors for a quad patch.	5
ProcessTriTessFactorsAvg	Generates the corrected tessellation factors for a tri patch.	5
ProcessTriTessFactorsMax	Generates the corrected tessellation factors for a tri patch.	5
ProcessTriTessFactorsMin	Generates the corrected tessellation factors for a tri patch.	5
radians	Converts x from degrees to radians.	1
rcp	Calculates a fast, approximate, per-component reciprocal.	5
reflect	Returns a reflection vector.	1
refract	Returns the refraction vector.	1[1]

reversebits	Reverses the order of the bits, per component.	5
round	Rounds x to the nearest integer	1[1]
rsqrt	Returns 1 / sqrt(x)	1[1]
saturate	Clamps x to the range [0, 1]	1
sign	Computes the sign of x.	1[1]
sin	Returns the sine of x	1[1]
sincos	Returns the sine and cosine of x.	1[1]
sinh	Returns the hyperbolic sine of x	1[1]
smoothstep	Returns a smooth Hermite interpolation between 0 and 1.	1[1]
sqrt	Square root (per component)	1[1]
step	Returns (x >= a) ? 1 : 0	1[1]
tan	Returns the tangent of x	1[1]
tanh	Returns the hyperbolic tangent of x	1[1]
tex1D(s, t)	1D texture lookup.	1
tex1D(s, t, ddx, ddy)	1D texture lookup.	2[1]
tex1Dbias	1D texture lookup with bias.	2[1]
tex1Dgrad	1D texture lookup with a gradient.	2[1]
tex1Dlod	1D texture lookup with LOD.	3[1]
tex1Dproj	1D texture lookup with projective divide.	2[1]
tex2D(s, t)	2D texture lookup.	1[1]
tex2D(s, t, ddx, ddy)	2D texture lookup.	2[1]
tex2Dbias	2D texture lookup with bias.	2[1]
tex2Dgrad	2D texture lookup with a gradient.	2[1]
tex2Dlod	2D texture lookup with LOD.	3

tex2Dproj	2D texture lookup with projective divide.	2[1]
tex3D(s, t)	3D texture lookup.	1[1]
tex3D(s, t, ddx, ddy)	3D texture lookup.	2[1]
tex3Dbias	3D texture lookup with bias.	2[1]
tex3Dgrad	3D texture lookup with a gradient.	2[1]
tex3Dlod	3D texture lookup with LOD.	3[1]
tex3Dproj	3D texture lookup with projective divide.	2[1]
texCUBE(s, t)	Cube texture lookup.	1[1]
texCUBE(s, t, ddx, ddy)	Cube texture lookup.	2[1]
texCUBEbias	Cube texture lookup with bias.	2[1]
texCUBEgrad	Cube texture lookup with a gradient.	2[1]
texCUBElod	Cube texture lookup with LOD.	3[1]
texCUBEproj	Cube texture lookup with projective divide.	2[1]
transpose	Returns the transpose of the matrix m.	1
trunc	Truncates floating-point value(s) to integer value(s)	1

[1] see reference page for restrictions.

Component and Template Types

The HLSL intrinsic function declarations use component types and template types for input parameter arguments and return values. The available types are listed in the following table.

These Template Types	Description	Support These Data Types
matrix	up to 16 components depending on the declaration	Basic HLSL Types
object	sampler object	*sampler*, *sampler1D*, *sampler2D*, *sampler3D*, *samplerCUBE*

scalar	1 component	Basic HLSL Types
vector	1 component minimum, 4 components maximum (inclusive)	Basic HLSL Types

See also

Reference for HLSL