XShaderCompiler

0.10 Alpha

Generated by Doxygen 1.8.11

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

Class Documentation

2.1 XscBindingSlot Struct Reference

Binding slot of textures, constant buffers, and fragment targets.

`#include <ReflectionC.h>`

Public Attributes

- const char * ident

  Identifier of the binding point.
- int location

  Zero based binding point or location. If this is -1, the location has not been set explicitly.

2.1.1 Detailed Description

Binding slot of textures, constant buffers, and fragment targets.

The documentation for this struct was generated from the following file:

- ReflectionC.h

2.2 XscFormatting Struct Reference

Formatting descriptor structure for the output shader.

`#include <XscC.h>`

Public Attributes

- bool alwaysBracedScopes

  If true, scopes are always written in braces. By default false.
- bool blanks

  If true, blank lines are allowed. By default true.
- bool compactWrappers

  If true, wrapper functions for special intrinsics are written in a compact formatting (i.e. all in one line). By default false.
- const char * indent

  Indentation string for code generation. By default 4 spaces.
- bool lineMarks

  If true, line marks are allowed. By default false.
- bool lineSeparation

  If true, auto-formatting of line separation is allowed. By default true.
- bool newLineOpenScope

  If true, the '{'-braces for an open scope gets its own line. If false, braces are written like in Java coding conventions. By default true.

2.2.1 Detailed Description

Formatting descriptor structure for the output shader.

The documentation for this struct was generated from the following file:

- XscC.h

2.3 XscIncludeHandler Struct Reference

Include handler structure.

`#include <IncludeHandlerC.h>`

Public Attributes

- XSC_PFN_HANDLE_INCLUDE handleIncludePfn

  Function pointer to handle the '#include'-directives.
- const char ** searchPaths

  Pointer to an array of search paths. This must be either NULL, point to an array where the last entry is always NULL.

2.3.1 Detailed Description

Include handler structure.

The documentation for this struct was generated from the following file:

- IncludeHandlerC.h

2.4 XscLog Struct Reference

Output log structure.

`#include <LogC.h>`

Public Attributes

- XSC_PFN_HANDLE_REPORT handleReportPfn
  Function pointer to handle compiler reports.

2.4.1 Detailed Description

Output log structure.

The documentation for this struct was generated from the following file:

- LogC.h

2.5 XscNameMangling Struct Reference

Name mangling descriptor structure for shader input/output variables (also referred to as "varyings"), temporary variables, and reserved keywords.

`#include <XscC.h>`

Public Attributes

- const char * inputPrefix
  Name mangling prefix for shader input variables. By default "xsv_".
- const char * outputPrefix
  Name mangling prefix for shader output variables. By default "xsv_".
- const char * reservedWordPrefix
  Name mangling prefix for reserved words (such as "texture", "main", "sin" etc.). By default "xsr_".
- const char * temporaryPrefix
  Name mangling prefix for temporary variables. By default "xst_".
- const char * namespacePrefix
  Name mangling prefix for namespaces like structures or classes. By default "xsn_".
- bool useAlwaysSemantics
- bool renameBufferFields
  If true, the data fields of a 'buffer'-objects is renamed rather than the outer identifier. By default false.

2.5.1 Detailed Description

Name mangling descriptor structure for shader input/output variables (also referred to as "varyings"), temporary variables, and reserved keywords.

2.5.2 Member Data Documentation

2.5.2.1 const char* XscNameMangling::inputPrefix

Name mangling prefix for shader input variables. By default "xsv_".

Remarks

This can also be empty or equal to "outputPrefix".

2.5.2.2 const char* XscNameMangling::namespacePrefix

Name mangling prefix for namespaces like structures or classes. By default "xsn_".

Remarks

This can also be empty, but if it's not empty it must not be equal to any of the other prefixes.

2.5.2.3 const char* XscNameMangling::outputPrefix

Name mangling prefix for shader output variables. By default "xsv_".

Remarks

This can also be empty or equal to "inputPrefix".

2.5.2.4 bool XscNameMangling::renameBufferFields

If true, the data fields of a 'buffer'-objects is renamed rather than the outer identifier. By default false.

Remarks

This can be useful for external diagnostic tools, to access the original identifier.

2.5.2.5 const char* XscNameMangling::reservedWordPrefix

Name mangling prefix for reserved words (such as "texture", "main", "sin" etc.). By default "xsr_".

Remarks

This must not be equal to any of the other prefixes and it must not be empty.

2.5.2.6 const char* XscNameMangling::temporaryPrefix

Name mangling prefix for temporary variables. By default "xst_".

Remarks

This must not be equal to any of the other prefixes and it must not be empty.

2.5.2.7 bool XscNameMangling::useAlwaysSemantics

If true, shader input/output variables are always renamed to their semantics, even for vertex input and fragment output. Otherwise, their original identifiers are used. By default false.

The documentation for this struct was generated from the following file:

- XscC.h

2.6 XscNumThreads Struct Reference

Number of threads within each work group of a compute shader.

```
#include <ReflectionC.h>
```

Public Attributes

- int **x**
- int **y**
- int **z**

2.6.1 Detailed Description

Number of threads within each work group of a compute shader.

The documentation for this struct was generated from the following file:

- ReflectionC.h

2.7 XscOptions Struct Reference

Structure for additional translation options.

```
#include <XscC.h>
```

Public Attributes

- bool allowExtensions

  If true, the shader output may contain GLSL extensions, if the target shader version is too low. By default false.
- bool autoBinding

  If true, binding slots for all buffer types will be generated sequentially, starting with index at 'autoBindingStartSlot'. By default false.
- int autoBindingStartSlot

  Index to start generating binding slots from. Only relevant if 'autoBinding' is enabled. By default 0.
- bool explicitBinding

  If true, explicit binding slots are enabled. By default false.
- bool obfuscate

  If true, code obfuscation is performed. By default false.
- bool optimize

  If true, little code optimizations are performed. By default false.
- bool preferWrappers

  If true, intrinsics are prefered to be implemented as wrappers (instead of inlining). By default false.
- bool preprocessOnly

  If true, only the preprocessed source code will be written out. By default false.
- bool preserveComments

  If true, commentaries are preserved for each statement. By default false.
- bool rowMajorAlignment

  If true, matrices have row-major alignment. Otherwise the matrices have column-major alignment. By default false.
- bool separateSamplers

  If true, generated GLSL code will contain separate sampler and texture objects when supported. By default true.
- bool separateShaders

  If true, generated GLSL code will support the 'ARB_separate_shader_objects' extension. By default false.
- bool showAST

  If true, the AST (Abstract Syntax Tree) will be written to the log output. By default false.
- bool showTimes

  If true, the timings of the different compilation processes are written to the log output. By default false.
- bool unrollArrayInitializers

  If true, array initializations will be unrolled. By default false.
- bool validateOnly

  If true, the source code is only validated, but no output code will be generated. By default false.

2.7.1 Detailed Description

Structure for additional translation options.

2.7.2 Member Data Documentation

2.7.2.1 bool XscOptions::autoBinding

If true, binding slots for all buffer types will be generated sequentially, starting with index at 'autoBindingStartSlot'. By default false.

Remarks

This will also enable 'explicitBinding'.

The documentation for this struct was generated from the following file:

- XscC.h

2.8 XscReflectionData Struct Reference

Structure for shader output statistics (e.g. texture/buffer binding points).

`#include <ReflectionC.h>`

Public Attributes

- const char ** macros
  All defined macros after pre-processing.
- size_t macrosCount
  Number of elements in 'macros'.
- const char ** uniforms
  Single shader uniforms.
- size_t uniformsCount
  Number of elements in 'uniforms'.
- const struct XscBindingSlot * textures
  Texture bindings.
- size_t texturesCount
  Number of elements in 'textures'.
- const struct XscBindingSlot * storageBuffers
  Storage buffer bindings.
- size_t storageBuffersCount
  Number of elements in 'storageBuffers'.
- const struct XscBindingSlot * constantBuffers
  Constant buffer bindings.
- size_t constantBufferCounts
  Number of elements in 'constantBuffers'.
- const struct XscBindingSlot * inputAttributes
  Shader input attributes.
- size_t inputAttributesCount
  Number of elements in 'inputAttributes'.
- const struct XscBindingSlot * outputAttributes
  Shader output attributes.
- size_t outputAttributesCount
  Number of elements in 'outputAttributes'.
- const struct XscSamplerState * samplerStates
  Static sampler states (identifier, states).
- size_t samplerStatesCount
  Number of elements in 'samplerStates'.
- struct XscNumThreads numThreads
  'numthreads' attribute of a compute shader.

2.8.1 Detailed Description

Structure for shader output statistics (e.g. texture/buffer binding points).

The documentation for this struct was generated from the following file:

- ReflectionC.h

2.9 XscReport Struct Reference

Report structure for warning and error messages.

`#include <ReportC.h>`

Public Attributes

- enum XscReportType type
  Specifies the report type.
- const char * context
  Specifies the context description string (e.g. a function name where the report occured). This may also be NULL.
- const char * message
  Specifies the message string.
- const char * line
  Specifies the line string where the report occured. This line never has new-line characters at its end. This may also be NULL.
- const char * marker
  Specifies the line marker string to highlight the area where the report occured. This may also be NULL.
- const char ** hints
  Specifies the list of optional hints of the report. This may also be NULL.
- size_t hintsCount
  Specifies the number of hints. If 'hints' is NULL, this is 0.

2.9.1 Detailed Description

Report structure for warning and error messages.

The documentation for this struct was generated from the following file:

- ReportC.h

2.10 XscSamplerState Struct Reference

Static sampler state descriptor structure (D3D11_SAMPLER_DESC).

`#include <ReflectionC.h>`

Public Attributes

- const char * **ident**
- enum XscFilter **filter**
- enum XscTextureAddressMode **addressU**
- enum XscTextureAddressMode **addressV**
- enum XscTextureAddressMode **addressW**
- float **mipLODBias**
- unsigned int **maxAnisotropy**
- enum XscComparisonFunc **comparisonFunc**
- float **borderColor** [4]
- float **minLOD**
- float **maxLOD**

2.10.1 Detailed Description

Static sampler state descriptor structure (D3D11_SAMPLER_DESC).

Remarks

All members and enumerations have the same values like the one in the "D3D11_SAMPLER_DESC" structure respectively. Thus, they can all be statically casted from and to the original D3D11 values.

See also

https://msdn.microsoft.com/en-us/library/windows/desktop/ff476207(v=vs.↩
85).aspx

The documentation for this struct was generated from the following file:

- ReflectionC.h

2.11 XscShaderInput Struct Reference

Shader input descriptor structure.

```
#include <XscC.h>
```

Public Attributes

- const char * filename

  Specifies the filename of the input shader code. This is an optional attribute, and only a hint to the compiler. By default NULL.
- const char * sourceCode

  Specifies the input source code. This must not be null when passed to the "XscCompileShader" function!
- enum XscInputShaderVersion shaderVersion

  Specifies the input shader version (e.g. XscEInputHLSL5 for "HLSL 5"). By default XscEInputHLSL5.
- enum XscShaderTarget shaderTarget

  Specifies the target shader (Vertex, Fragment etc.). By default XscUndefinedShader.
- const char * entryPoint

  Specifies the HLSL shader entry point. By default "main".
- const char * secondaryEntryPoint

  Specifies the secondary HLSL shader entry point. By default NULL.
- unsigned int warnings

  Compiler warning flags. This can be a bitwise OR combination of the "XscWarnings" enumeration entries. By default 0.
- unsigned int extensions

  Language extension flags. This can be a bitwise OR combination of the "Extensions" enumeration entries. By default 0.
- struct XscIncludeHandler includeHandler

  Include handler member which contains a function pointer to handle '#include'-directives.

2.11.1 Detailed Description

Shader input descriptor structure.

2.11.2 Member Data Documentation

2.11.2.1 unsigned int XscShaderInput::extensions

Language extension flags. This can be a bitwise OR combination of the "Extensions" enumeration entries. By default 0.

Remarks

This is ignored, if the compiler was not build with the 'XSC_ENABLE_LANGUAGE_EXT' macro.

See also

Extensions

2.11.2.2 const char* XscShaderInput::secondaryEntryPoint

Specifies the secondary HLSL shader entry point. By default NULL.

Remarks

This is only used for a Tessellation-Control Shader (alias Hull Shader) entry point, when a Tessellation-Control Shader (alias Domain Shader) is the output target. This is required to translate all Tessellation-Control attributes (i.e. "partitioning" and "outputtopology") to the Tessellation-Evaluation output shader. If this is empty, the default values for these attributes are used.

2.11.2.3 unsigned int XscShaderInput::warnings

Compiler warning flags. This can be a bitwise OR combination of the "XscWarnings" enumeration entries. By default 0.

See also

XscWarnings

The documentation for this struct was generated from the following file:

- XscC.h

2.12 XscShaderOutput Struct Reference

Shader output descriptor structure.

```
#include <XscC.h>
```

Public Attributes

- const char * filename

  Specifies the filename of the output shader code. This is an optional attribute, and only a hint to the compiler.
- const char ** sourceCode

  Specifies the output source code. This will contain the output code. This must not be null when passed to the "XscCompileShader" function!
- enum XscOutputShaderVersion shaderVersion

  Specifies the output shader version. By default XscEOutputGLSL (to auto-detect minimum required version).
- const struct XscVertexSemantic * vertexSemantics

  Optional list of vertex semantic layouts, to bind a vertex attribute (semantic name) to a location index (only used when 'explicitBinding' is true). By default NULL.
- size_t vertexSemanticsCount

  Number of elements the 'vertexSemantics' member points to. By default 0.
- struct XscOptions options

  Additional options to configure the code generation.
- struct XscFormatting formatting

  Output code formatting descriptor.
- struct XscNameMangling nameMangling

  Specifies the options for name mangling.

2.12.1 Detailed Description

Shader output descriptor structure.

The documentation for this struct was generated from the following file:

- XscC.h

2.13 XscVertexSemantic Struct Reference

Vertex shader semantic (or rather attribute) layout structure.

```
#include <XscC.h>
```

Public Attributes

- const char * semantic

  Specifies the shader semantic (or rather attribute).
- int location

  Specifies the binding location.

2.13.1 Detailed Description

Vertex shader semantic (or rather attribute) layout structure.

The documentation for this struct was generated from the following file:

- XscC.h

Index