



LIBNVVM API

v5.5 | July 2013

API Reference Manual



TABLE OF CONTENTS

Chapter 1. Modules.....	1
1.1. Error Handling.....	1
nvvmResult.....	1
nvvmGetErrorString.....	2
1.2. General Information Query.....	2
nvvmVersion.....	2
1.3. Compilation.....	2
nvvmProgram.....	2
nvvmAddModuleToProgram.....	3
nvvmCompileProgram.....	3
nvvmCreateProgram.....	5
nvvmDestroyProgram.....	5
nvvmGetCompiledResult.....	6
nvvmGetCompiledResultSize.....	6
nvvmGetProgramLog.....	7
nvvmGetProgramLogSize.....	7
nvvmVerifyProgram.....	8

Chapter 1.

MODULES

Here is a list of all modules:

- ▶ Error Handling
- ▶ General Information Query
- ▶ Compilation

1.1. Error Handling

enum nvvmResult

NVVM API call result code.

Values

`NVVM_SUCCESS = 0`

`NVVM_ERROR_OUT_OF_MEMORY = 1`

`NVVM_ERROR_PROGRAM_CREATION_FAILURE = 2`

`NVVM_ERROR_IR_VERSION_MISMATCH = 3`

`NVVM_ERROR_INVALID_INPUT = 4`

`NVVM_ERROR_INVALID_PROGRAM = 5`

`NVVM_ERROR_INVALID_IR = 6`

`NVVM_ERROR_INVALID_OPTION = 7`

`NVVM_ERROR_NO_MODULE_IN_PROGRAM = 8`

`NVVM_ERROR_COMPILATION = 9`

const char *nvvmGetErrorString (nvvmResult result)

Get the message string for the given nvvmResult code.

Parameters

result

NVVM API result code.

Returns

Message string for the given nvvmResult code.

1.2. General Information Query

nvvmResult nvvmVersion (int *major, int *minor)

Get the NVVM version.

Parameters

major

NVVM major version number.

minor

NVVM minor version number.

Returns

- ▶ NVVM_SUCCESS

1.3. Compilation

typedef _nvvmProgram *nvvmProgram

NVVM Program.

An opaque handle for a program

nvvmResult nvvmAddModuleToProgram (nvvmProgram prog, const char *buffer, size_t size, const char *name)

Add a module level NVVM IR to a program.

Parameters

prog

NVVM program.

buffer

NVVM IR module in the bitcode or text representation.

size

Size of the NVVM IR module.

name

Name of the NVVM IR module. If NULL, "<unnamed>" is used as the name.

Returns

- ▶ NVVM_SUCCESS
- ▶ NVVM_ERROR_OUT_OF_MEMORY
- ▶ NVVM_ERROR_INVALID_INPUT
- ▶ NVVM_ERROR_INVALID_PROGRAM

Description

The buffer should contain an NVVM IR module either in the bitcode representation or in the text representation.

nvvmResult nvvmCompileProgram (nvvmProgram prog, int numOptions, const char **options)

Compile the NVVM program.

Parameters

prog

NVVM program.

numOptions

Number of compiler options passed.

options

Compiler options in the form of C string array.

Returns

- ▶ NVVM_SUCCESS

- ▶ NVVM_ERROR_OUT_OF_MEMORY
- ▶ NVVM_ERROR_IR_VERSION_MISMATCH
- ▶ NVVM_ERROR_INVALID_PROGRAM
- ▶ NVVM_ERROR_INVALID_IR
- ▶ NVVM_ERROR_INVALID_OPTION
- ▶ NVVM_ERROR_NO_MODULE_IN_PROGRAM
- ▶ NVVM_ERROR_COMPILATION

Description

The valid compiler options are:

- ▶ -g (enable generation of debugging information)
- ▶ -opt=
 - ▶ 0 (disable optimizations)
 - ▶ 3 (default, enable optimizations)
- ▶ -arch=
 - ▶ compute_20 (default)
 - ▶ compute_30
 - ▶ compute_35
- ▶ -ftz=
 - ▶ 0 (default, preserve denormal values, when performing single-precision floating-point operations)
 - ▶ 1 (flush denormal values to zero, when performing single-precision floating-point operations)
- ▶ -prec-sqrt=
 - ▶ 0 (use a faster approximation for single-precision floating-point square root)
 - ▶ 1 (default, use IEEE round-to-nearest mode for single-precision floating-point square root)
- ▶ -prec-div=
 - ▶ 0 (use a faster approximation for single-precision floating-point division and reciprocals)
 - ▶ 1 (default, use IEEE round-to-nearest mode for single-precision floating-point division and reciprocals)
- ▶ -fma=
 - ▶ 0 (disable FMA contraction)
 - ▶ 1 (default, enable FMA contraction)

`nvvmResult nvvmCreateProgram (nvvmProgram *prog)`

Create a program, and set the value of its handle to *prog.

Parameters

prog

NVVM program.

Returns

- ▶ `NVVM_SUCCESS`
- ▶ `NVVM_ERROR_OUT_OF_MEMORY`
- ▶ `NVVM_ERROR_INVALID_PROGRAM`

Description

See also:

[nvvmDestroyProgram\(\)](#)

`nvvmResult nvvmDestroyProgram (nvvmProgram *prog)`

Destroy a program.

Parameters

prog

NVVM program.

Returns

- ▶ `NVVM_SUCCESS`
- ▶ `NVVM_ERROR_INVALID_PROGRAM`

Description

See also:

[nvvmCreateProgram\(\)](#)

`nvvmResult nvvmGetCompiledResult (nvvmProgram prog, char *buffer)`

Get the compiled result.

Parameters

prog

NVVM program.

buffer

Compiled result.

Returns

- ▶ `NVVM_SUCCESS`
- ▶ `NVVM_ERROR_INVALID_PROGRAM`

Description

The result is stored in the memory pointed by 'buffer'.

`nvvmResult nvvmGetCompiledResultSize (nvvmProgram prog, size_t *bufferSizeRet)`

Get the size of the compiled result.

Parameters

prog

NVVM program.

bufferSizeRet

Size of the compiled result (including the trailing NULL).

Returns

- ▶ `NVVM_SUCCESS`
- ▶ `NVVM_ERROR_INVALID_PROGRAM`

`nvvmResult nvvmGetProgramLog (nvvmProgram prog, char *buffer)`

Get the Compiler/Verifier Message.

Parameters

prog

NVVM program program.

buffer

Compilation/Verification log.

Returns

- ▶ `NVVM_SUCCESS`
- ▶ `NVVM_ERROR_INVALID_PROGRAM`

Description

The NULL terminated message string is stored in the memory pointed by 'buffer' when the return value is `NVVM_SUCCESS`.

`nvvmResult nvvmGetProgramLogSize (nvvmProgram prog, size_t *bufferSizeRet)`

Get the Size of Compiler/Verifier Message.

Parameters

prog

NVVM program.

bufferSizeRet

Size of the compilation/verification log (including the trailing NULL).

Returns

- ▶ `NVVM_SUCCESS`
- ▶ `NVVM_ERROR_INVALID_PROGRAM`

Description

The size of the message string (including the trailing NULL) is stored into 'buffer_size_ret' when the return value is `NVVM_SUCCESS`.

`nvvmResult nvvmVerifyProgram (nvvmProgram prog, int numOptions, const char **options)`

Verify the NVVM program.

Parameters

prog

NVVM program.

numOptions

Number of compiler options passed.

options

Compiler options in the form of C string array.

Returns

- ▶ `NVVM_SUCCESS`
- ▶ `NVVM_ERROR_OUT_OF_MEMORY`
- ▶ `NVVM_ERROR_IR_VERSION_MISMATCH`
- ▶ `NVVM_ERROR_INVALID_PROGRAM`
- ▶ `NVVM_ERROR_INVALID_IR`
- ▶ `NVVM_ERROR_INVALID_OPTION`
- ▶ `NVVM_ERROR_NO_MODULE_IN_PROGRAM`

Description

The valid compiler options are:

NONE. The `numOptions` and `options` parameters are ignored, and are for future use.

Notice

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication of otherwise under any patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all other information previously supplied. NVIDIA Corporation products are not authorized as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

Trademarks

NVIDIA and the NVIDIA logo are trademarks or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2007-2013 NVIDIA Corporation. All rights reserved.