libNVVM API

API Reference Manual

vRelease Version | July 2019

Table of Contents

Chapter 1. Modules

Here is a list of all modules:

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 nvvmIRVersion (int *majorIR, int *minorIR, int *majorDbg, int *minorDbg)

Get the NVVM IR version.

Parameters

majorIR
NVVM IR major version number.

minorIR
NVVM IR minor version number.

majorDbg
NVVM IR debug metadata major version number.

minorDbg
NVVM IR debug metadata minor version number.

Returns

- NVVM_SUCCESS

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. The module may have NVVM IR version 1.5 in LLVM 5.0 bitcode format. Alternatively, the module may have NVVM IR version 1.2 either in the LLVM 3.4 bitcode representation or in the LLVM 3.4 text representation. Support for reading the text representation of NVVM IR is deprecated and may be removed in a later version.

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_OPTION
- NVVM_ERROR_NO_MODULE_IN_PROGRAM
- NVVM_ERROR_COMPILATION

Description

The NVVM IR modules in the program will be linked at the IR level. The linked IR program is compiled to PTX.

The target datalayout in the linked IR program is used to determine the address size (32bit vs 64bit).

The valid compiler options are:

- -g (enable generation of debugging information, valid only with -opt=0)
- -generate-line-info (generate line number information)
- -opt=
  - 0 (disable optimizations)
  - 3 (default, enable optimizations)
- -arch=
  - compute_35
  - compute_37
  - compute_50

- compute_52 (default)
- compute_53
- compute_60
- compute_61
- compute_62
- compute_70
- compute_72
- compute_75
- compute_80

- -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 nvvmLazyAddModuleToProgram (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 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. The module may have NVVM IR version 1.5 in LLVM 5.0 bitcode format. Alternatively, the module may have NVVM IR version 1.2 in the LLVM 3.4 bitcode representation.

A module added using this API is lazily loaded - the only symbols loaded are those that are required by module(s) loaded using nvvmAddModuleToProgram. It is an error for a program to have all modules loaded using this API. Compiler may also optimize entities in this module by making them internal to the linked NVVM IR module, making them eligible for other optimizations. Due to these optimizations, this API to load a module is more efficient and should be used where possible.

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:

Same as for nvvmCompileProgram().

See also:

nvvmCompileProgram()

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

VESA DisplayPort

DisplayPort and DisplayPort Compliance Logo, DisplayPort Compliance Logo for Dual-mode Sources, and DisplayPort Compliance Logo for Active Cables are trademarks owned by the Video Electronics Standards Association in the United States and other countries.

HDMI

HDMI, the HDMI logo, and High-Definition Multimedia Interface are trademarks or registered trademarks of HDMI Licensing LLC.

OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

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

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com