Request for Quotations

OpenCL 2.2 CTS

September 2016

Notice
ALL KHRONOS SPECIFICATIONS AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." KHRONOS 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, Khronos 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 or otherwise under any patent or patent rights of Khronos. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied.

Trademarks
Khronos, SPIR and SPIR-V is a trademark of the Khronos Group Inc. OpenCL is a trademark of Apple Inc. used under license by Khronos. All other product names, trademarks, and/or company names are used solely for identification and belong to their respective owners.

1. Background

OpenCL™ and SPIR-V™ are open, royalty-free standards for heterogeneous parallel programming. OpenCL 2.2 incorporates the OpenCL C++ kernel language for significantly enhanced parallel programming productivity. OpenCL 2.2 also adds some minor API functionality. SPIR-V 1.1 extends the intermediate representation defined by Khronos with native support for shader and compute kernel features to fully support the OpenCL C++ kernel language.

2. Goals

The **primary** goal of this project is to enhance and expand the existing OpenCL 2.1 conformance tests to create an OpenCL 2.2 test suite to be used to define conformance for OpenCL 2.2 implementations.

The **secondary** goal is to analyze the resulting OpenCL 2.2 test suite to determine any licenses necessary to bring the OpenCL test suite into open source under the Khronos Apache 2.0 license.

Upon project completion the OpenCL 2.2 test suite will be made available to OpenCL Adopters under the existing Khronos Adopters Agreement, and Khronos may also use the data from this project to initiate bringing the OpenCL 2.2 test suite into open source.

3. Scope

The OpenCL working group has defined the work packages needed to complete this project with a first estimate of man days to complete each package:

Work Package	Man Days
Add test for new OpenCL C++ features and representations	40
LIT tests for broad range of C++ language features	21
Add tests for new OpenCL 2.2 API features	10
Analyze Test Sources for Open Source Licensing	5
Total Man Days	**76**

A detailed description and breakdown of the new tests contained in these work packages are provided in Attachment A.

4. Deliverables, Methodology and Acceptance Criteria

Deliverables will include:

1. Code for enhanced tests as described in attachment A, with associated documentation. These new tests shall be added to the existent OpenCL conformance framework in the Khronos GitLab repository and shall be written to be buildable across Windows, Linux, Android and Mac;
2. Open source license analysis document for the complete OpenCL 2.2 conformance suite. This should be a spreadsheet, listing all code modules and the associated source license used in each module. A summary document should contain a list of all the licenses in use, with their actual wording. No legal analysis or recommendations are required.

5. Selection Schedule and Process

Khronos will follow the schedule below to select a Contractor:

1. **September 20th** – Khronos Releases RFQ;

2. **October 18th** – RFQ responses received by Khronos;

3. **November 1st** – Contractor selected and notified;

4. **November 14th** – Contract executed and start of work.

If a bidder is not a Khronos member, NDA access will be provided to any necessary information to create a bid. The contractor will be selected from any received bids by the Khronos OpenCL working group. The selected contractor will be required to execute the standard Khronos Membership Agreement (with membership fees waived) if they are not already a Khronos member, and execute the standard Khronos Contactors Agreement with milestones and costs entered into Exhibit B and Contractor Disclosures entered into Exhibit C.

No work shall begin, and Khronos shall be liable for no costs or expenses, until the selected contractor is in receipt of an executed contractor's agreement.

6. RFQ Responses

The RFQ responses will form the basis for detailed milestone and cost negotiations for the final contract with the selected vendor(s). Vendors are encouraged to quote for a subset of the deliverables if they feel they are able to specifically address that subset as Khronos will consider splitting this project between multiple respondents.

Please provide the following information in the format of your choice:

1. Identification of which elements of the Project Scope outlined in Section 3 on which you wish to bid;

2. A description of your familiarity with OpenCL, SPIR-V and experience with conformance testing and open source methodology;

3. Proposed testing approach for the added tests;

4. Proposed schedule to complete deliverables, including when you are available to commence work and incremental milestones;

5. Whether you are bidding a fixed cost or a T&M contract. Fixed cost responses are strongly preferred;

6. Confirmation that you are willing to work under the Khronos Contractor Agreement and execute the Khronos Membership Agreement for the duration of the project;

7. Any particular issues or risk factors that you wish to highlight;

8. Supporting materials, including background materials about your company, highlighting relevant experience and expertise for this project, personnel backgrounds (e.g., resume and GitHub profiles), in particular success in delivering projects requiring close attention to detail and strong writing and communication skills.

Responses and all subsequent communication regarding this RFQ should be sent to opencl-rfq@khronos.org.

Attachment A - Work Package Breakdowns

A1. Add test for new OpenCL C++ features and representations

Task	Work Days	Description
New representation of types		
images, samplers	3	Test should verify the following scenarios for 2D and 3D images: - reading from image - sampling from image - writing to image - inline samplers and samplers passed by a kernel argument
pipes, resevation	2	Test should verify: - reading from pipe without reservations - writing from pipe without reservations - reading from pipe with reservations - writing from pipe with reservations
device enqueue, events	3	Test should verify: - enqueuing two simple child kernel + event dependencies - creating and using user events - using enqueue_marker method
address spaces	2	Test should verify: - simple use of all pointer types (global_ptr, local_ptr, constant_ptr, private_ptr) - simple use of all storage types (global, local, priv, constant) - use local memory in program scope
New representation of functions		
convert_cast and as_type	2	Test should verify: - a few fixed conversions between vector types with and without specifing rounding modes - a few as_type casts between a few vector types
atomics	1	Simple use of new atomic type with e.g. add operation
work-item functions	1	Simple kernel using all functions specified in the spec
work-group functions	2	Simple use of work_group functions with the fixed input for 1 type (e.g. int)
synchronization functions	1	Simple use of sub_group_barrier and work_group_barrier functions
math functions	2	Simple use of all functions for 1 type (e.g. float) with the fixed input
integer functions	2	Simple use of all functions for 1 type (e.g. int) with the fixed input

common functions	1	Simple use of all functions for 1 type (e.g. float) with the fixed input
geometric functions	1	Simple use of all functions for 1 type (e.g. float) with the fixed input
relational functions	1	Simple use of all functions for 1 type (e.g. float) with the fixed input
vload and vstore functions	2	Simple use of all functions
New in OpenCL C++		
spec constants	3	Test should verify: - simple use of spec constant in if statement - using spec constants in kernel attributes - initializing variables with a spec constant value - int and float spec constants
KHR named barriers	2	Test should verify: - all configurations of flags and scope - using work_group_named_barrier type in use cases described in OpenCL C++ spec
required_num_sub_groups attribute	1	Simple scenario using required_num_sub_groups attribute
ivdep attribute	1	Simple scenario using ivdep attribute
max_size attribute	2	Test should verify: - max_size attribute with local_ptr, constant_ptr, local*, constant* - fixed size values - size specified in a spec constant
constructors and destructors of global scope objects	3	Test should verify: - program scope object with non-trivial constructors and destructors and if they are correctly exeucted
program scope pipes / pipe storage	3	Test should verify: - creating read pipe accessor from program scope pipes/pipe storage - creating write pipe accessor from program scope pipes/pipe storage - reading and writing with and without reservations - various sizes of program scope pipes

A2. LIT tests for broad range of C++ language features – 21 Days

Preparation		
Adding -emit-spirv switch to the LIT tests	1	Tasks to be done: - modify all libclcxx tests to use -emit-spirv instead of -emit-llvm - modify clang/test/OpenCL/OpenCL22/attributes test to use -emit-spirv instead of -emit-llvm
SPIR-V checks		
Checking SPIR-V instead of LLVM	20	Tasks to be done: - add SPIR-V checks to atomic, conversions, functions, half, images, pipes, printf, queue, reinterpret, sampler, Vectors, vload_vstore, work_group, work_item tests from libclcxx - add SPIR-V checks to attribute tests from Clang

A3. Add tests for new OpenCL 2.2 API features – 20 Days

Task	Work Days	Description
Test support for constructors and destructors for program scope global variables	7	a. Verify queries clGetProgramInfo correctly return the presence of contructors and/or destructors in the program (using option CL_PROGRAM_SCOPE_CTORS_PRESENT/CL_PROGRAM_SCOPE_DTORS_PRESENT) (both are present, either one is present, none is present) b. Verify that constructors are executed before any kernel is executed. Verify that when present, multiple constructors are executed. The order between constructors is undefined, but they should all execute. c. Verify that constructors are only executed once when multiple kernels from a program are executed. For example, when the first kernel is run the program scope variable contains the value assigned by the constructor. Assume that first program modifies the value of the program scope global. Then when second kernel is executed it will observe the value assigned by the first kernel, not from constructor. d. Verify that when constructor is executed, the ND-range used is (1,1,1). e. Verify that destructor is executed (need to create a destructor with a side effect). f. Verify that multiple destructors, if present, are executed. Order between multiple destructors is undefined. g. Verify that each destructor is executed only once. h. Verify that ND-range during destructor execution is set to (1,1,1)
Test support for specialization constants	3	a. Verify that if left unset the specialization constant defaults to the default value set in SPIR-V (zero). b. Verify that setting an existing specialization constant affects only the value of that constant and not of other specialization constants. c. Verify that the API correctly handles the size of a specialization constant by exercising the API for specialization constants of different types (int, bool, float, etc.)